Algorithmes

2e édition

Antoine Cornuéjols - Laurent Miclet

Apprentissage artificiel

Concepts et algorithmes

Préface de Jean-Paul Haton

Copyright © 2010 Eyrolles.

EYROLLES

CHEZ LE MÊME ÉDITEUR

Dans la même collection

G. Dreyfus *et al.* – **Apprentissage statistique et réseaux de neurones**.
Méthodologie et applications.
N°12229, 3e édition 2008, 464 pages avec CD-Rom.

P. Naïm, P.-H. Wuillemin, P. Leray, O. Pourret, A. Becker. – **Réseaux bayésiens**.
N°11972, 3e édition, 2007, 424 pages (collection Algorithmes).

G. Fleury, P. Lacomme et A. Tanguy. – **Simulation à événements discrets**.
Modèles déterministes et stochastiques – Exemples d'applications implémentés en Delphi et en C++.
N°11924, 2006, 444 pages avec CD-Rom.

J. Richalet *et al.* – **La commande prédictive**.
Mise en oeuvre et applications industrielles.
N°11553, 2004, 256 pages.

P. Lacomme, C. Prins, M. Sevaux – **Algorithmes de graphes**.
N°11385, 2003, 368 pages, avec CD-Rom.

J. Dréo, A. Pétrowski, P. Siarry, E. Taillard – **Métaheuristiques pour l'optimisation difficile**.
Recuit simulé, recherche tabou, algorithmes évolutionnaires et algorithmes génétiques, colonies de fourmis...
N°11368, 2003, 368 pages.

Y. Collette, P. Siarry – **Optimisation multiobjectif**.
N°11168, 2002, 316 pages.

C. Guéret, C. Prins, M. Sevaux. – **Programmation linéaire.**
65 problèmes d'optimisation modélisés et résolus avec Visual XPress.
N°9202, 2000, 365 pages, *avec CD-ROM.*

Autres ouvrages

I. Hurbain, avec la contribution d'E. Dreyfus. – **Mémento Unix/Linux**
N°11954, 2006, 14 pages.

C. Jacquet. – **Mémento LaTeX**
N°12244, 2007, 14 pages.

R. Rimelé. – **Mémento MySQL.**
N°12720, 2e édition 2010, 14 pages.

R. M. Stallman *et al.* – **Richard Stallman et la révolution du logiciel libre.** *Une biographie autorisée.*
N°12609, 2010, 344 pages.

S. Bordage, D. Thévenon, L. Dupaquier, F. Brousse. – **Conduite de projet Web.**
N°12665, 5e édition, 2010, 432 pages.

S. Jaber. – **Programmation GWT 2**. *Développer des applications Ajax avec le Google Web Toolkit.*
N°12569, 2010, 484 pages

Copyright © 2010 Eyrolles.

2e édition

Apprentissage artificiel

Antoine Cornuéjols - Laurent Miclet

Copyright © 2010 Eyrolles.

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Remerciements à Eric Bernauer pour la relecture de cet ouvrage.

Copyright © 2010 Eyrolles.



Le code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée notamment dans les établissements d'enseignement, provoquant une baisse brutale des achats de livres, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.
En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans autorisation de l'éditeur ou du Centre Français d'Exploitation du Droit de Copie, 20, rue des Grands-Augustins, 75006 Paris.
● Groupe Eyrolles, 2002, 2010, ISBN : 978-2-212-12471-2

Préface à la deuxième édition

Apprendre.

Apprendre par l'exemple, par l'environnement, par la lecture, par le professeur, par cœur...

L'apprentissage est multiforme et c'est une caractéristique de l'intelligence. On comprend la remarque, très opportunément mise en exergue par les auteurs, d'Alan Turing, un des pionniers de l'intelligence artificielle. Prétendre doter une machine de cette faculté propre à l'être humain, ou à tout le moins à l'animal supérieur, pouvait paraître une gageure à l'époque où cette remarque a été énoncée. Ce n'est plus le cas aujourd'hui et le vaste champ de l'apprentissage par une machine est un domaine de recherche en pleine expansion et dans lequel il y a encore beaucoup à faire !

L'apprentissage occupe une place privilégiée au sein de l'intelligence artificielle, et plus généralement de l'informatique. Cette place ne cessera de croître. Les succès des programmes incluant un certain niveau d'apprentissage automatique ou semi-automatique sont déjà nombreux. Il suffit de songer à la reconnaissance de la parole, la vision par ordinateur, le rejet de pourriels, la détection de transactions frauduleuses, le diagnostic, les jeux, la prédiction et la prévision, la fouille de données, etc. Les progrès spectaculaires enregistrés sont dûs pour une bonne part aux efforts des chercheurs qui sont parvenus à une meilleure compréhension des processus d'apprentissage, qu'ils soient implantés sur une machine ou qu'ils existent dans le cortex d'un animal.

Le moment est donc opportun de faire le point sur les connaissances acquises et les applications. La décision de proposer une profonde révision de la première édition de l'ouvrage de A. Cornuéjols et L. Miclet arrive ainsi à point nommé. Ces deux auteurs, aux compétences complémentaires, sont particulièrement bien indiqués pour couvrir le vaste champ pluridisciplinaire de l'apprentissage. La première édition, de très grande qualité, a connu un succès considérable et justifié, auprès d'un public varié : étudiants, enseignants-chercheurs, ingénieurs. Elle est devenue un ouvrage de référence pour la communauté francophone proposant la somme la plus complète d'idées, de concepts, d'algorithmes et d'applications sur le sujet.

Le même fil directeur original a été conservé pour cette seconde édition. L'accroissement des connaissances se traduit directement dans le nombre de pages et l'on ne peut que se féliciter qu'il existe encore en France des éditeurs acceptant de faire paraître un ouvrage scientifique original de plus de 800 pages...

Je ne doute pas du succès de cette édition dont je recommande chaudement la lecture à toute personne désirant faire le point sur l'apprentissage, un des plus grands défis lancé à la recherche en intelligence artificielle.

Jean-Paul Haton
Nancy, 28 mars 2010

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

The idea of a learning machine may appear paradoxical to some readers.
A. M. Turing, 1950.

à Isabelle, Claire, Aurélie, Sébastien, Fanny
et à Maura, Fabien, Marion

Présentation de la deuxième édition

La première édition de cet ouvrage, parue en septembre 2002, a reçu un très bon accueil, montrant l'intérêt d'un livre couvrant largement les aspects de l'apprentissage artificiel. Par ailleurs son organisation raisonnée autour de trois grandes parties : l'apprentissage par exploration, par optimisation, et par approximation et interpolation, l'uniformité des notations et un fil directeur tenu de bout en bout ont visiblement séduit à côté de l'offre des ouvrages existant en langue anglaise.

Au fil des années, nous avons reçu de nombreux courriels témoignant à la fois de la variété du public intéressé : étudiants, enseignant-chercheurs du domaine, spécialistes de domaines connexes, et grand public, et de l'étendue géographique des publics touchés : la zone francophone bien sûr, y compris le Canada et les pays du Maghreb, mais aussi des pays d'Europe centrale.

Plus rapidement qu'attendu, les presque 2000 exemplaires de la première édition ont été épuisés. La question s'est alors posée du choix entre une simple ré-impression ou bien une mise à jour conduisant à une deuxième édition. L'expérience de la première édition aurait du nous rendre prudents, mais la mémoire humaine étant volatile et étant donnée la vitalité du domaine de l'apprentissage artificiel, il nous a paru pertinent de choisir la deuxième voie. Petit à petit cependant, nous avons réalisé que non seulement les techniques et les résultats avaient progressé, mais que, plus largement, de nouvelles questions et de nouvelles approches étaient apparues depuis 2002. Il devenait difficile de se contenter de simplement adapter la première édition, de nouveaux chapitres étaient nécessaires. Par ailleurs, une ré-organisation de certaines parties était également souhaitable pour tenir compte de nouvelles perspectives ou de l'accent porté à des approches classiques mais remises au goût du jour, comme les méthodes linéaires. D'un ravalement de façade, nous sommes insensiblement passés à un chantier comprenant une ré-organisation des espaces et des cloisons et à l'élaboration d'extensions significatives. Comme pour tout chantier, les délais prévus ont été largement dépassés, puis dépassés encore, et à certains moments, notre éditeur pourtant conciliant, et nos familles pourtant très compréhensives, ont pu croire que la fiction de Dino Buzzati, « Le désert des tartares », se réalisait. Ce n'est donc finalement qu'en 2010 qu'apparaît cette deuxième édition.

Le fil directeur allant de l'exposition des fondements conceptuels et méthodologiques, puis progressant depuis des apprentissages très guidés par l'existence d'une relation de généralité dans l'espace des hypothèses à des apprentissages s'appuyant sur des espaces de plus en plus démunis

Copyright © 2010 Eyrolles.

de structure a été maintenu. Nous avons cependant ajouté une partie inédite sur de nouvelles questions et de nouveaux types d'apprentissage qui dépassent les cadres classiques des apprentissages supervisés, non supervisés et par renforcement. Ainsi, par exemple, les apprentissages d'ordonnancement, les apprentissages semi-supervisés, actifs, à partir de flux de données et en ligne, font l'objet de sections ou de chapitres nouveaux.

Par ailleurs, le développement considérable des méthodes à base de fonctions noyau nous a conduit à dédier tout un chapitre aux méthodes linéaires classiques, et un grand chapitre aux méthodes à noyaux. De même, les méthodes d'ensemble, boosting, bagging, etc. font maintenant l'objet d'un chapitre à part entière.

Finalement, tous les chapitres ont été mis à jour pour tenir compte des progrès réalisés. De nombreuses figures ont été refaites pour les rendre plus lisibles, et beaucoup d'autres ont été ajoutées. La typographie a évolué afin de mieux mettre en évidence les définitions, les théorèmes et les formules principales. L'index a été entièrement revu et largement augmenté afin de faciliter l'accès direct aux concepts. Au bout du compte, et malgré notre souci de rester concis, le nombre de pages est passé en huit ans de 630 à 830. Cela reflète la vitalité du domaine et l'accroissement des idées, concepts et méthodes utiles à connaître.

Nous espérons que cette deuxième édition séduira un public aussi large que pour la première édition. Bienvenue dans le nouvel édifice. Nous vous souhaitons une visite agréable, une installation heureuse et l'envie d'apporter de nouvelles idées, d'ouvrir de nouvelles fenêtres, et de dessiner de nouveaux horizons.

Antoine CORNUÉJOLS et Laurent MICLET

Paris, Lannion, France
Le 27 Mars 2010

Nous tenons à remercier particulièrement les personnes suivantes pour leur aide, leurs commentaires, leurs encouragements, et en général pour leurs contributions à la réalisation de cet ouvrage. Notre gratitude va aussi aux lecteurs critiques des versions préliminaires, ce qui inclut notablement une certaine proportion de nos étudiants. Merci à vous et aussi à ceux que nous avons pu oublier ici mais qui sont importants pour nous.

Abdel Belaïd, Sami Bengio, Younes Bennani, Christophe Bernard, Marc Bernard, Olivier Boëffard, Cédric Buche, Michel Cartier, Christophe Choisy, Delphine Cosandier, François Coste, François Denis, Gérard Douaire, Pierre Dupont, Béatrice Duval, Lou Fedon, Daniel Fredouille, Mirta Gordon, Colin de la Higuera, Ghazal Jaber, Yves Kodratoff, Israël-César Lerman, Gaëlle Loosli, Christine Martin, Tristan Mary-huard, Stan Matwin, Maurice Milgram, Engelbert Mephu Nguifo, Tom Mitchell, Jacques Nicolas, Laurent Orseau, Yann Prudent, Arpad Rimmel, Céline Rouveirol, Michèle Sebag, Dominique Snyers, Franck Thollard, Fabien Torre, Stéphane Vandenmersch et Jean-Daniel Zucker.

Merci aussi à notre éditrice, Muriel Shan-Sei-Fan, et à Sophie Hincelin pour une relecture complète du manuscrit. Heureusement qu'il existe encore des éditeurs de cette qualité.

Copyright © 2010 Eyrolles.

Préface

L'apprentissage artificiel s'intéresse à l'écriture de programmes d'ordinateur capables de s'améliorer automatiquement au fil du temps, soit sur la base de leur propre expérience, soit à partir de données antérieures fournies par d'autres programmes. Dans le domaine scientifique relativement jeune de l'informatique, l'apprentissage artificiel joue un rôle de plus en plus essentiel. Au début de son existence, dans les années 1950, l'informatique se résumait principalement à programmer des machines en leur spécifiant ligne après ligne la séquence d'instructions que l'ordinateur aurait à suivre. Autour des années 1990, les logiciels étaient devenus si complexes qu'une alternative s'imposait naturellement : développer des techniques pour que les programmes puissent s'entraîner sur des exemples. Le résultat est qu'il existe aujourd'hui de nombreux domaines d'application de l'informatique dans lesquels les méthodes de l'apprentissage artificiel sont employées pour entraîner les logiciels. Mieux, le code résultant dépasse de beaucoup en performance les réalisations les plus abouties de programmation manuelle « ligne après ligne ». C'est ainsi que tous les meilleurs logiciels commercialisés de reconnaissance de la parole sont fondés sur l'entraînement de leurs programmes à la reconnaissance des différents sons et mots. La plupart d'entre eux permettent même à l'utilisateur d'accoutumer le système aux caractéristiques de sa voix. D'autres exemples existent dans des domaines tels que la vision par ordinateur, le traitement automatique du texte et la commande de robot.

L'apprentissage artificiel peut donc déjà revendiquer des succès dans un grand nombre de domaines d'application. Il en est ainsi de logiciels de fouille de données utilisés à grande échelle pour découvrir la prescription la plus efficace pour un patient, à partir de l'analyse de fichiers médicaux antérieurs. D'autres applications vont de la prédiction de la demande en énergie, étant connu l'historique des consommations antérieures, à l'apprentissage de la reconnaissance de transactions frauduleuses par carte de crédit, par examen des transactions passées avérées frauduleuses. Alors que nous passons des cinquante premières années de l'informatique aux cinquante prochaines années, il semble évident que le rôle de l'apprentissage artificiel ne cessera de croître au centre de cette science.

Pourquoi cette progression? La réponse fondamentale est que nous possédons désormais la compréhension de plusieurs principes calculatoires qui guident tout processus d'apprentissage, qu'il soit implémenté sur une machine ou sur un humain. La discipline de l'apprentissage artificiel possède désormais de riches fondements théoriques : on commence à savoir répondre à des questions comme : « Combien au mimimum d'exemples d'entraînement faut-il fournir à un programme d'apprentissage pour être certain qu'il apprenne avec une efficacité donnée? » et « Quelles méthodes d'apprentissage sont les plus efficaces pour tel ou tel type de problème? » Ces fondements proviennent de la théorie statistique de l'estimation, de la théorie de l'identification et de la commande optimale, de travaux pionniers sur la complexité de l'apprentissage de grammaires ou plus récents sur l'inférence bayésienne algorithmique.

Cet ouvrage fournit au lecteur francophone l'introduction la plus complète à ce jour à l'apprentissage artificiel. Il traite de la théorie et des applications de cette discipline sous un grand nombre d'aspects, en couvrant des sujets comme les méthodes d'apprentissage bayésien, l'inférence grammaticale ou l'apprentissage par renforcement. C'est avec plaisir que je recommande au lecteur de découvrir ce livre, et à travers lui les idées et les méthodes de l'apprentissage artificiel.

Tom M. MITCHELL

Pittsburgh, Pennsylvania, USA
Le 29 Mai 2002

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Table des matières

Avant-propos xiii
Quelques applications de l'apprentissage artificiel . . . xiv
Quelques définitions de base . . . xiv
Deux champs industriels de l'apprentissage artificiels : la reconnaissance des formes et la fouille de données . . . xv
Les caractéristiques de l'apprentissage artificiel . . . xvii
Trois exemples d'apprentissage . . . xviii
Organisation et plan de l'ouvrage . . . xxi
Guide de lecture . . . xxiii

Notations xxvii

I Les fondements de l'apprentissage 1

1 De l'apprentissage naturel à l'apprentissage artificiel 3
1 L'apprentissage artificiel . . . 4
2 Deux exemples : apprendre à jouer, apprendre à lire . . . 6
3 Deux approches : la cybernétique et les sciences cognitives . . . 10
4 Les concepts de base de l'apprentissage . . . 14
5 L'induction comme un jeu entre espaces . . . 22
6 Retour sur l'organisation de l'ouvrage . . . 29

2 Première approche théorique de l'induction 37
1 Poser un problème d'apprentissage . . . 40
2 Approches bayésiennes et approche directe pour décider . . . 46
3 Le critère inductif . . . 49
4 Analyse du principe de *minimisation du risque empirique* . . . 60
5 Le lien entre le passé et le futur et le *no-free-lunch theorem* . . . 73
6 Notes historiques et bibliographiques . . . 78

Copyright © 2010 Eyrolles.

3 L'environnement méthodologique de l'apprentissage 81
1 L'espace des données d'apprentissage . . . 85
2 L'espace des hypothèses d'apprentissage . . . 101
3 Les protocoles d'apprentissage . . . 112
4 L'évaluation de l'apprentissage . . . 113
5 La comparaison des méthodes d'apprentissage . . . 124
6 Autres problèmes pratiques . . . 129

II Apprentissage par exploration 135

4 Induction et relation d'ordre : l'espace des versions 137
1 Les concepts de base . . . 140
2 La structuration de l'espace des hypothèses . . . 144
3 La construction de l'espace des versions . . . 152
4 La représentation des connaissances par un treillis de Galois . . . 156

5 La programmation logique inductive 161
1 La programmation logique inductive : le cadre général . . . 165
2 La logique des prédicats et les programmes logiques : terminologie . . . 171
3 La structuration de l'espace des hypothèses en logique des prédicats . . . 175
4 L'exploration de l'espace des hypothèses . . . 182
5 Deux exemples de systèmes de PLI . . . 186
6 La probabilisation de la PLI . . . 190
7 Les domaines d'application de la PLI . . . 191
8 Les chantiers de la PLI . . . 193

6 Transfert de connaissances et apprentissage par analogie 199
1 L'apprentissage en présence de théorie . . . 200
2 L'apprentissage par examen de preuve (EBL) . . . 201
3 Abstraction et reformulation des connaissances . . . 208
4 Changement de repère, raisonnement par analogie et RàPC . . . 210
5 L'apprentissage par proportion analogique . . . 213
6 Bilan . . . 217

7 L'inférence grammaticale 219
1 Définitions et notations . . . 225
2 Les protocoles de l'inférence : quelques résultats théoriques . . . 233
3 L'espace de recherche de l'inférence régulière . . . 240
4 L'inférence régulière sans échantillon négatif . . . 241
5 L'inférence régulière sous contrôle d'un échantillon négatif . . . 246
6 L'inférence de grammaires algébriques . . . 250
7 L'inférence d'automates probabilistes . . . 257
8 Quelques approches complémentaires . . . 260

Copyright © 2010 Eyrolles.

8 Apprentissage par évolution simulée **263**
1 Trois espaces au lieu de deux 265
2 Un modèle formel simplifié de l'évolution 268
3 Les algorithmes génétiques 269
4 Les stratégies d'évolution 281
5 La programmation génétique 281
6 La coévolution 291

III Apprentissage par optimisation **297**

9 L'apprentissage de modèles linéaires **299**
1 Introduction 300
2 Régression linéaire 301
3 Séparatrices linéaires 306
4 Modèles linéaires par morceaux et combinaisons de modèles locaux 317
5 La recherche des facteurs pertinents 320

10 L'apprentissage de réseaux connexionnistes **325**
1 Les différents éléments d'un réseau connexionniste 328
2 L'architecture multicouche 330
3 L'algorithme d'apprentissage 335
4 Quelques résultats théoriques sur les réseaux connexionnistes 344
5 Comment choisir l'architecture d'un réseau ? 345
6 Les réseaux à architecture profonde 346
7 Réseaux et régime dynamique : le *Reservoir Computing* 348

11 L'apprentissage de réseaux bayésiens **353**
1 Les modèles graphiques 355
2 Les réseaux d'inférence bayésiens 357
3 Les inférences dans les réseaux bayésiens 365
4 L'apprentissage des réseaux bayésiens 369
5 L'inférence de relations causales 378
6 Applications 379
7 Quelques logiciels 380

12 L'apprentissage de modèles de Markov cachés **383**
1 Les modèles de Markov observables 386
2 Les modèles de Markov cachés (HMM) 387
3 Les HMM comme règles de classification de séquences 392
4 L'évaluation de la probabilité d'observation 393
5 Le calcul du chemin optimal : l'algorithme de Viterbi 395
6 L'apprentissage 398
7 Approfondissements 403
8 Applications 404

13 Apprentissage par inférence d'arbres **407**
1 Les arbres de décision 409
2 Les arbres de régression 423

Copyright © 2010 Eyrolles.

IV Apprentissage par approximation et interpolation 427

14 Méthodes à noyaux 429
1 Trois voies vers les méthodes à noyau . . . 431
2 Philosophie des méthodes à noyaux . . . 441
3 Les Séparatrices à Vastes Marges (SVM) . . . 443
4 Autres types d'induction avec fonctions noyau . . . 453
5 Ingénierie des fonctions noyau . . . 458
6 Les méthodes à noyaux en pratique . . . 477
7 Bilan et perspectives . . . 483

15 L'apprentissage bayésien et son approximation 487
1 L'apprentissage bayésien . . . 490
2 Les méthodes paramétriques . . . 499
3 Les méthodes non paramétriques . . . 509
4 Les méthodes semi-paramétriques . . . 523

16 L'apprentissage de réflexes par renforcement 531
1 Description du problème . . . 534
2 Si tout est connu : l'utilité de la fonction d'utilité . . . 541
3 L'apprentissage des fonctions d'utilité quand l'environnement est connu . . . 543
4 Sans modèle du monde : la méthode de Monte-Carlo . . . 547
5 La méthode des différences temporelles . . . 548
6 La généralisation dans l'apprentissage par renforcement . . . 552
7 Contrôle optimal par recherche arborescente et algorithme UCT . . . 558
8 Le cas des environnements partiellement observables . . . 561
9 Exemples d'application . . . 562
10 Bilan et perspectives . . . 565

V Au-delà de l'apprentissage supervisé 569

17 Apprentissage de combinaisons d'experts 571
1 Principes des méthodes par combinaison . . . 572
2 Le vote de plusieurs classificateurs . . . 576
3 Les codes correcteurs de classes . . . 577
4 Le *boosting* d'un algorithme d'apprentissage . . . 582
5 Le *bagging* . . . 590
6 Les forêts aléatoires (*random forests*) . . . 591
7 L'apprentissage en cascade (*cascading*) . . . 591

18 La classification non supervisée et la fouille de données 593
1 Introduction . . . 596
2 Les méthodes de classification fondées sur les distances . . . 597
3 Les méthodes de classification par des modèles probabilistes . . . 609
4 Méthodes spectrales de catégorisation . . . 610
5 La classification de données symboliques . . . 612
6 La fouille de données . . . 615
7 Les analyses en composantes . . . 625

Copyright © 2010 Eyrolles.

19 L'apprentissage semi-supervisé **635**
1 Présentation 636
2 Les modèles génératifs : apprentissage dans l'espace joint $\mathcal{X} \times \mathcal{Y}$ 640
3 L'auto-apprentissage 643
4 Le co-apprentissage 644
5 L'utilisation d'hypothèses fondamentales sur les données 646
6 Quelques directions pour l'apprentissage semi-supervisé 655
7 Conclusion et petite discussion 657

20 Vers de nouvelles tâches et de nouvelles questions **659**
1 Apprentissage actif 661
2 Apprentissages en ligne, incrémental et par transferts 669
3 Apprentissage à partir de flux de données 677
4 Apprentissage de sorties structurées 682
5 Apprentissage pour le filtrage collaboratif 686

21 Analyse de l'induction : approfondissements et ouvertures **695**
1 Généralisation de l'analyse du principe MRE 696
2 Principes inductifs contrôlant l'espace des hypothèses 702
3 Prise en compte de l'algorithme d'apprentissage dans la théorie 715
4 Autres types d'analyses 720

VI Annexes techniques **727**

22 Annexes techniques **729**
1 Exemples de fonctions de perte en induction 729
2 Le calcul de l'intervalle de confiance de l'estimation de la probabilité d'une règle de classification 732
3 Estimation d'une densité de probabilité en un point. 733
4 L'estimation des paramètres d'une distribution gaussienne. 734
5 Pourquoi et comment la règle du PPV converge-t-elle ? 735
6 Pourquoi la règle de décision bayésienne est-elle optimale ? 736
7 Apprentissage par estimation-maximisation 736
8 Optimisation par descente de gradient 740
9 La rétropropagation du gradient de l'erreur 744
10 L'analyse de l'induction de Vapnik 747
11 L'induction par compression d'information 758

Index **795**

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Avant-propos

Ce livre présente les théories, les algorithmes et les applications de l'apprentissage artificiel. Son ambition est d'une part d'unifier le cadre méthodologique, et d'autre part de décrire un ensemble d'algorithmes utiles, de manière cohérente avec ce cadre, et de présenter ses applications existantes et potentielles.

À quoi sert l'apprentissage artificiel? La plupart des programmes d'intelligence artificielle possèdent aujourd'hui un module d'apprentissage et tous les programmes de reconnaissance des formes sont fondés sur des algorithmes d'apprentissage. Et que font ces programmes? Ils sont capables de reconnaître la parole humaine et de l'interpréter. Ils réalisent une analyse automatique de photos satellites pour détecter certaines ressources sur la Terre. Ils assistent les experts pour prendre des décisions dans des environnements complexes et évolutifs, par exemple le marché financier ou le diagnostic médical. Ils fouillent d'immenses bases de données hétérogènes comme les millions de pages Web accessibles à tous. Ils analysent les données clientèle des entreprises pour les aider à mieux cibler leurs campagnes de publicité. Ils participent aussi à des tournois : le 11 mai 1997, le tenant du titre de champion du monde du jeu d'échecs, Gary Kasparov, a été battu en match par un programme.

On sait donc programmer les ordinateurs pour leur faire exécuter des tâches considérées comme intelligentes, de multiples façons et de manière de plus en plus efficace. Cet ouvrage s'intéresse à un aspect particulier de cette intelligence artificielle : la faculté d'apprentissage.

L'apprentissage artificiel est une discipline dont les outils et les champs d'applications sont assez disparates. Cependant, les connaissances de base nécessaires à sa compréhension sont essentiellement une culture généraliste que l'on trouve par exemple dans les ouvrages de mathématiques pour l'informatique : notions d'algèbre linéaire, de probabilités, de combinatoire, d'analyse élémentaire, d'algorithmique, de théorie des langages, de logique. Dans la mesure du possible, ces notions de base sont brièvement rappellées selon la nécessité des chapitres de ce livre.

À qui s'adresse cet ouvrage ?

On peut tirer profit de cet ouvrage en autodidacte, comme le fera par exemple un ingénieur qui cherche à connaître ce qui se cache derrière les mots ou à acquérir une initiation à des techniques qu'il ignore encore. On pourra aussi s'en servir comme d'un appui pour compléter un enseignement : ce sera le cas pour un étudiant au niveau maîtrise, DEA ou école d'ingénieurs, ou comme d'un ouvrage de référence pour faire un cours sur le domaine.

Copyright © 2010 Eyrolles.

Quelques applications de l'apprentissage artificiel

Voyons maintenant comment rendre un programme plus efficace en le dotant d'une possibilité d'apprentissage. Reprenons pour cela les applications de l'intelligence artificielle et de la reconnaissance des formes citées ci-dessus.

- Les performances d'un programme d'application augmentent au fur et à mesure de son utilisation par la même personne : c'est une expérience qu'il est aujourd'hui facile de mener en pratique en utilisant un logiciel personnel de dictée vocale.
- Un programme de détection des ressources terrestres apprend à reconnaître une zone de pollution au milieu de la mer, à partir d'une base de données d'exemples d'images de zones connues comme propres ou comme polluées : cette base de données lui sert d'expérience pour déterminer sa décision sur une zone inconnue.
- Un programme de diagnostic sur un ensemble d'informations évolutives prises sur un patient doit avoir été pourvu de connaissances, à partir de diagnostics de praticiens et d'experts sur des situations types. Mais il doit aussi avoir été doté d'un module de généralisation, de façon à réagir correctement à des situations auxquelles il n'a jamais été confronté exactement.
- Les moteurs de recherche sur le Web pourraient être munis d'un module d'adaptation au style de navigation de l'usager : c'est une faculté souhaitable pour augmenter l'ergonomie de leur utilisation. Les programmes ne sont pas encore réellement agrémentés de cette propriété, mais il est clair que c'est une condition nécessaire pour franchir certains obstacles de communication si évidents actuellement.
- L'exploitation des fichiers client d'une entreprise est souvent réalisée par un expert ou un programme expert qui utilise des règles explicites pour cibler un segment de clientèle susceptible d'être intéressé par un nouveau produit. Mais ces règles peuvent être acquises automatiquement, par un apprentissage dont le but est de fournir de nouvelles connaissances expertes, à la fois efficaces et intelligibles pour l'expert.
- Un programme de jeu d'échecs possède en général une très bonne efficacité *a priori* ; mais il est naturel d'essayer de le doter d'un module où il puisse analyser ses défaites et ses victoires, pour améliorer ses performances moyennes dans ses parties futures. Ce module d'apprentissage existe dans un certain nombre de programmes de jeux.

Quelques définitions de base

Apprentissage (sous-entendu : artificiel, automatique) (*Machine Learning*)

Cette notion englobe toute méthode permettant de construire un modèle de la réalité à partir de données, soit en améliorant un modèle partiel ou moins général, soit en créant complètement le modèle. Il existe deux tendances principales en apprentissage, celle issue de l'intelligence artificielle et qualifiée de *symbolique*, et celle issue des statistiques et qualifiée de *numérique*.

Fouille de données (*Data Mining*) ou **Extraction de connaissances à partir des données** (*Knowledge Discovery in Data*)

La fouille de données prend en charge le processus complet d'extraction de connaissances : stockage dans une base de données, sélection des données à étudier, si nécessaire : nettoyage

Copyright © 2010 Eyrolles.

des données, puis utilisation des apprentissages numériques et symboliques afin de proposer des modèles à l'utilisateur, et enfin validation des modèles proposés. Si ces modèles sont invalidés par l'utilisateur, le processus complet est répété.

Précision *vs.* généralisation .

Le grand dilemme de l'apprentissage. La précision est définie par un écart entre une valeur mesurée ou prédite et une valeur réelle. Apprendre avec trop de précision conduit à un « sur-apprentissage », comme l'apprentissage par cœur, pour lequel des détails insignifiants (ou dûs au bruit) sont appris. Apprendre avec trop peu de précision conduit à une « sur-généralisation » telle que le modèle s'applique même quand l'utilisateur ne le désire pas. Les deux types d'apprentissage, numérique et symbolique, ont défini des mesures de généralisation et c'est à l'utilisateur de fixer le seuil de généralisation qu'il juge optimal.

Intelligibilité (devrait être *Comprehensibility* mais tend à devenir *Understandability*).

Depuis quelques années, principalement sous la poussée des industriels, les chercheurs se sont mis à essayer de contrôler aussi l'intelligibilité du modèle obtenu par la fouille de données. Jusqu'à présent les méthodes de mesure de l'intelligibilité se réduisent à vérifier que les résultats sont exprimés dans le langage de l'utilisateur et que la taille des modèles n'est pas excessive. Des méthodes spécifiques de visualisation sont aussi utilisées.

Classification, classement et régression. La classification, telle qu'elle est définie en analyse de données, consiste à regrouper des ensembles d'exemples non supervisés en classes. Ces classes sont souvent organisées en une structure (*clustering*). Si cette structure est un arbre, alors on parle de taxonomie ou de taxinomie (*taxonomy*). Sous l'influence du mot anglais *classification*, on a tendance à confondre classification et classement. Ce dernier mot désigne le processus de reconnaissance en intension (par leur propriétés) de classes décrites en extension (par les valeurs de leurs descripteurs). Lorsque les valeurs à prédire sont des classes en petit nombre, on parle de classification. Il s'agit par exemple de prévoir l'appartenance d'un oiseau observé à la classe « canard » ou « oie ». La *régression* traite des cas où les valeurs à prédire sont numériques, par exemple : *nombre d'exemplaires de cet ouvrage qui seront vendus* = 3900.

Deux champs industriels de l'apprentissage artificiels : la reconnaissance des formes et la fouille de données

En quarante ans et plus d'existence, l'apprentissage artificiel a fourni un grand nombre d'outils aux industriels et aux entrepreneurs. Nous les regroupons selon deux grands axes : la *reconnaissance des formes* et la *fouille de données* ou pour être plus précis, l'*extraction de connaissances des données*.

Le second domaine est le moins bien connu des deux bien qu'il soit porteur de réelles possibilités économiques.

Quant au premier, rappellons seulement que les méthodes de l'apprentissage artificiel sont à la base de la reconnaissance des images (écriture manuscrite, signatures, détection de ressources par satellite, pilotage automatique, etc.), de la reconnaissance de la parole, du traitement avancé des signaux bio-médicaux, etc. Pour mesurer l'extraordinaire vitalité des applications et des potentialités de la reconnaissance des formes, il suffit par exemple de suivre la parution incessante des livres dans ce domaine. Pour ne citer que lui, l'éditeur World Scientific propose une cinquantaine

Copyright © 2010 Eyrolles.

de titres à son catalogue sous la rubrique « Applications de la reconnaissance des formes » et les renouvelle à raison de près d'une dizaine par an.

Revenons maintenant à la fouille de données. Les problèmes pratiques que peut résoudre en ce domaine l'apprentissage artificiel se posent constamment dans la vie industrielle : distinguer un bon client d'un mauvais, et reconnaître un mauvais procédé de fabrication et l'améliorer sont deux exemples frappants parmi d'autres. On constate pourtant que l'ancrage de ce type d'application dans la vie industrielle ne date que des années 1990, avec la naissance d'une discipline nouvelle, créée sous le nom de « fouille de données » (*data mining*) ou ECD : « extraction de connaissances à partir des données » (*Knowledge Discovery in Databases*, KDD). Nous présentons rapidement le domaine avant d'en donner l'état de l'art industriel dans le dernier paragraphe de cet avant-propos.

L'ECD est née de la constatation que les trois approches qui permettaient de construire des modèles, à savoir les statistiques exploratoires, l'analyse des données et l'apprentissage symbolique automatique (ASA), souffraient de deux défauts communs : exiger des données présentées sous une forme très rigide et faire peu de cas de l'intelligibilité des résultats. De plus, chacune présentait un défaut particulier gênant leur emploi : les statistiques exploratoires et l'analyse des données s'adressaient à des données essentiellement numériques et l'ASA se limitait aux données symboliques ou discrétisées en intervalles de valeurs.

Depuis, ces domaines ont évolué et les critiques à leur adresser ont changé, mais tel était l'état de l'art dans les années 1990. L'ECD est donc née d'un quadruple effort :

- permettre aux utilisateurs de fournir des données dans l'état où elles sont : ceci a donné naissance aux techniques de nettoyage des données (ce point sera développé au chapitre 3) ;
- utiliser les données enregistrées sous forme de bases de données (en général relationnelles) : ceci a provoqué un large courant de recherche au sein de la communauté des BD intéressée par la création de modèles ;
- fournir aux utilisateurs des outils capables de travailler sur des données mixtes, numériques et symboliques ;
- construire des outils produisant une connaissance intelligible aux utilisateurs.

C'est ainsi que l'ECD a pu trouver la large reconnaissance industrielle dont elle jouit actuellement. Elle a commencé à résoudre les deux problèmes industriels principaux de l'analyse des données, ceux qui coûtent le plus cher : le fait que le client est souvent imprécis dans la définition du problème qu'il se pose et le fait que les données dont il dispose sont souvent de qualité discutable.

L'étude des applications industrielles de l'ECD montre qu'il existe une assez forte demande en outils de création de modèles, autrement dit en apprentissage artificiel. Ceci se traduit par le fait qu'environ cent cinquante compagnies se sont spécialisées dans ce domaine. Certaines de ces entreprises existent depuis plusieurs années et d'autres se sont vendues fort cher. L'ensemble revèle bien un secteur en progression raisonnable sur plusieurs années.

Notre estimation est que le marché de l'ECD est occupé par 60 % d'outils d'apprentissage statistiques et 40 % d'outils d'apprentissage symbolique. Ces dernières techniques étant moins enseignées que les premières dans les universités, on constate un hiatus entre l'enseignement et l'industrie. En tous cas, le présent livre cherche à aller dans le sens d'un meilleur enseignement des méthodes de l'apprentissage artificiel, symboliques comme statistiques.

Copyright © 2010 Eyrolles.

Les caractéristiques de l'apprentissage artificiel

Certaines des facultés que l'on peut lier naturellement à l'apprentissage ont été citées dans les exemples ci-dessus : entraînement, reconnaissance, généralisation, adaptation, amélioration, intelligibilité.

Rappellons la définition classique de l'apprentissage en sciences cognitives : « capacité à améliorer les performances au fur et à mesure de l'exercice d'une activité ». Cette définition s'applique en particulier au comportement d'un joueur d'échecs au fil des parties, où l'assimilation de l'expérience et la puissance de raisonnement se combinent dans sa progression. Elle est aussi pertinente pour des tâches perceptives : on s'habitue à un accent, à une écriture. On accumule des bonnes et des mauvaises expériences. À partir d'elles, on sait, consciemment ou non, en abstraire ou faire évoluer des règles pour mieux effectuer la tâche.

Nous avons mentionné une autre facette de l'apprentissage, souvent entremêlée à la précédente : la faculté à généraliser rationnellement. Si une expérience accumulée sur un certain nombre d'exemples a permis de tirer des règles de comportement, ces règles doivent s'appliquer aussi à des situations non encore rencontrées. Prenons quelqu'un qui apprend à conduire sur une berline de petite puissance. Dès qu'il a mérité le permis, la loi l'autorise à conduire une camionnette utilitaire ou une voiture de sport. C'est que les règles qu'il a apprises et les réflexes qu'il a acquis s'appliquent aussi (plus ou moins directement) à ces véhicules.

Qu'en est-il des machines? Dès les débuts de l'intelligence artificielle, c'est-à-dire en vérité dès l'apparition des ordinateurs, les chercheurs et les ingénieurs se sont posés le problème de l'apprentissage [1]. L'apprentissage artificiel dans sa situation actuelle est donc le produit d'une histoire de cinquante ans de recherches et de réalisations. Comme on l'a vu, un grand nombre de tâches d'intelligence artificielle et de reconnaissance des formes s'appuient ou sont fondées sur des modules d'apprentissage.

On verra dans cet ouvrage comment des programmes peuvent mettre en œuvre un apprentissage par amélioration du comportement, en général grâce à des techniques d'optimisation. On verra aussi qu'il est possible d'écrire des programmes qui réalisent un apprentissage par généralisation : quand on leur donne suffisamment d'exemples et le type du concept à apprendre, ils choisissent un concept qui n'est pas seulement valide sur les exemples qu'ils ont vus, mais qui sera également valable pour d'autres. C'est ainsi qu'un programme de reconnaissance de la parole ne peut pas « entendre » tous les sons avant d'élaborer une règle de décision. Il est écrit pour extraire une méthode de classification de ceux qu'on lui a présentés et traiter ensuite du mieux possible tous les sons qu'il aura à décoder.

En réalité, d'un point de vue informatique, la problématique n'est pas fondamentalement différente dans les deux cas. Il s'agit dans le premier de faire évoluer des règles de comportement au fil des exemples et dans le second d'extraire des règles à partir d'un ensemble d'exemples donné *a priori*. De même que dans l'apprentissage naturel, un panachage de ces deux modes de fonctionnement est facile à concevoir dans l'apprentissage artificiel.

Il y a une autre facette de l'apprentissage que l'intelligence artificielle explore. Quand un expert extrait des connaissances d'un ensemble de données, il apprend une certaine façon de les résumer. Mais le résultat de cet apprentissage ne sera opératoire que si la connaissance extraite est intelligible, transmissible de l'expert aux utilisateurs, interprétable « en clair ». Il en est de

[1] Alan Turing, dans son article *Computing Machine and Intelligence*, de la revue *Mind* en Octobre 1950 (Vol LIX, No 236) avait intitulé un paragraphe *Learning Machines*. On peut consulter un fac-similé du manuscrit sur le site `http ://data.archives.ecs.soton.ac.uk/turing/` et le texte à : `http ://www.abelard.org/turpap/turpap.htm`

Copyright © 2010 Eyrolles.

même pour un agent artificiel : certaines tâches d'apprentissage ne se mesurent pas seulement par leur qualité de prédiction, mais aussi par la manière dont les résultats sont expliqués. Cet aspect est relié opérationnellement à l'intelligence artificielle symbolique, aux systèmes experts en particulier : mieux vaut souvent un petit nombre de règles compréhensibles qu'un fouillis de règles sophistiquées, même avec une performance objective supérieure.

Avant de décrire plus en détail les motivations et l'organisation de cet ouvrage, précisons à travers trois exemples comment s'organise l'apprentissage dans des situations concrètes. Cela nous permettra de donner une typologie des méthodes et de présenter le plan de cet ouvrage.

Trois exemples d'apprentissage

Un exemple ornithologique

Imaginons un étang sur lequel nagent des oies et des cygnes (nous admettons qu'il n'y a pas d'autres oiseaux dans cette région). Le brouillard est tombé, quand arrivent deux avimateurs dont l'un est expert et l'autre débutant. Ils n'aperçoivent en arrivant qu'une partie des animaux, de manière peu distincte. Pour l'expert, l'identification est cependant facile (il n'est pas expert pour rien). Quant au débutant, il doit se contenter de mesurer ce qui lui paraît caractéristique : le niveau de gris du plumage et la taille de la bête. Pour représenter le problème, il va donc prendre ces deux mesures sur chaque animal qu'il voit et établir un graphique : il se place ainsi dans un certain espace de représentation (figure -1.1, à gauche).

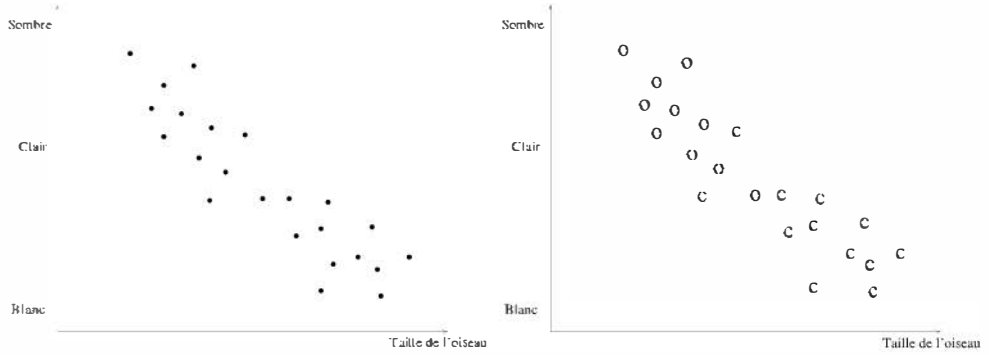


FIG. -1.1: *Le premier graphique de l'avimateur débutant représente les oiseaux observés placés dans son espace de représentation. Le second graphique représente les mêmes oiseaux, mais il est étiqueté par l'expert. La lettre O signifie que l'oiseau est une oie, C qu'il est un cygne.*

Maintenant, comment lancer une phase d'apprentissage ? Il faut que le débutant se place en situation d'apprenant vis-à-vis de l'expert, en lui demandant quelle est la décision correcte pour chaque oiseau. Une fois que l'expert a agi comme un professeur en donnant toutes les réponses, notre apprenant possède un graphique enrichi (figure -1.1, à droite) qui va lui permettre de démarrer l'apprentissage proprement dit.

Le problème d'apprentissage est maintenant bien posé. Il peut s'énoncer ainsi : comment trouver une règle qui décide, dans l'espace de représentation choisi, avec le moins d'erreurs possibles, quel oiseau est une oie et quel oiseau est un cygne ? La règle trouvée doit posséder

Copyright © 2010 Eyrolles.

de bonnes propriétés de généralisation, c'est-à-dire fonctionner au mieux non seulement sur ces exemples expertisés, mais par la suite sur des oiseaux non encore observés.

Que sera une telle règle ? L'apprenant peut imaginer de tracer dans le plan de représentation une ligne (courbe ou droite) qui sépare les cygnes des oies. À partir des exemples connus, il aura alors induit une loi générale : tout oiseau observé dont la représentation est située « sous » cette ligne sera un cygne. Ce sera une oie dans le cas contraire. Mais on peut tracer une infinité de telles lignes. C'est ici que l'apprenant doit préciser le type des connaissances à acquérir, le type du concept à apprendre, en l'espèce quelle est la forme générale de la ligne.

Si l'apprenant impose que la ligne soit droite, le but de l'apprentissage sera de trouver la meilleure ligne droite, en optimisant un critère dont il est maître. On remarque d'ailleurs qu'aucune droite ne sépare parfaitement les exemples, mais c'est le prix à payer pour un concept aussi simple. La figure -1.2 de gauche représente la règle de décision que notre débutant en ornithologie peut raisonnablement produire. S'il n'impose pas de restriction aussi stricte sur la forme de la ligne, il pourra obtenir une décision comme celle de la figure -1.2 à droite.

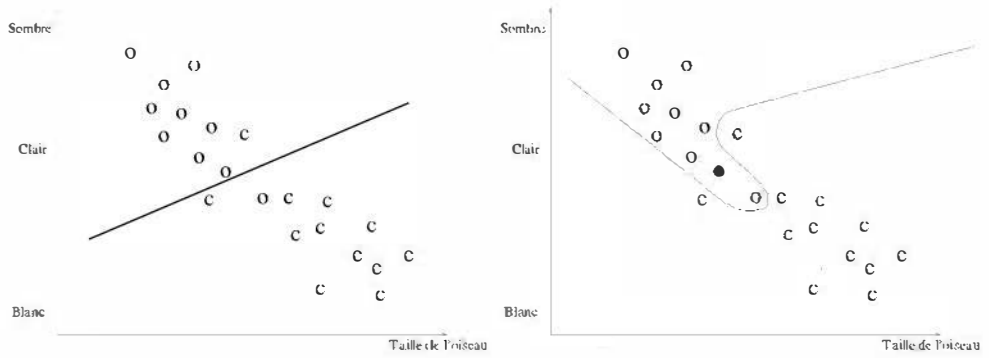


FIG. -1.2: *Une règle de décision simple et une règle de décision complexe pour séparer les oies des cygnes.*

Quand le brouillard se lève, d'autres oiseaux deviennent visibles. L'apprenant peut alors vérifier la qualité de la règle qu'il a apprise, toujours avec l'aide de son professeur. Dans l'exemple donné sur la figure -1.3, il est facile de constater que la droite qu'il a choisie mène à une erreur environ une fois sur cinq[2]. Pas trop mal, pour un débutant ! Il est assez facile de transposer cet exemple à l'écriture d'un programme d'apprentissage. Remarquons bien qu'un tel programme n'apprend pas tout court mais apprend quelque chose, en l'occurence une règle de décision sous la forme d'une équation de droite dans le plan. Cet exemple est caractéristique de ce que font les programmes de reconnaissance des formes. Ce type d'apprentissage par généralisation est d'une immense importance méthodologique et pratique.

Un exemple linguistique

Abordons un autre exemple. Supposons que nous disposions d'un ensemble de phrases d'une certaine langue. Est-il possible d'écrire un programme pour en apprendre automatiquement la grammaire ? Pour une langue naturelle, le problème est certainement complexe, ne serait-ce que parce qu'un très grand nombre d'exemples est nécessaire. Mais on peut essayer de le résoudre dans le cas d'un langage artificiel comme ceux qui servent à interroger les bases de données ou

[2] La ligne courbe donnerait une erreur encore plus grande ; nous reviendrons sur ce phénomène aux chapitres 2 et 3.

Copyright © 2010 Eyrolles.

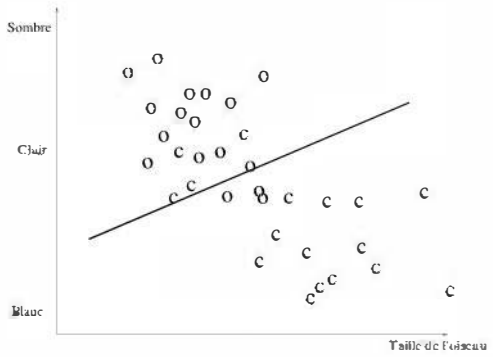


FIG. -1.3: *Le test de la règle simple sur d'autres oiseaux.*

pour un sous-ensemble bien délimité de langage naturel. Le langage des échanges entre un client et un employé d'agence de voyage en est un exemple.

Dans de tels cas, il est effectivement possible d'apprendre une grammaire. Il faut cependant imposer au programme des restrictions sur le type de la syntaxe que l'on cherche. L'espace de représentation est ici l'ensemble de toutes les séquences de mots possibles, dont on ne connaît que certaines, linguistiquement correctes. Mais comment définir la grammaire à apprendre ? On verra au chapitre 7 que si on oblige cette grammaire à être un automate fini, on peut démontrer que tous les automates finis qui sont compatibles avec les exemples forment un ensemble limité et structuré par une relation d'ordre. Le programme d'apprentissage a alors pour tâche de chercher le meilleur automate dans cet ensemble structuré, encore une fois au sens d'un critère à lui préciser. Remarquons encore que le programme n'apprend pas tout court, mais apprend quelque chose : en l'espèce, une grammaire représentée par un automate fini.

Un exemple d'extraction de connaissances

Une compagnie d'assurances cherche à lancer un nouveau produit, destiné à couvrir le risque de vol d'objets de valeur à domicile. Elle veut faire une campagne de publicité ciblée auprès d'une partie de ses clients. Cette compagnie ne dispose que de peu de produits du même type et par conséquent sa base de données ne comporte qu'une petite proportion d'enregistrements où un client est déjà associé à une assurance contre le vol à domicile. De plus, comme ces clients possèdent déjà un produit analogue, ce n'est pas vers eux qu'il faut principalement cibler la campagne. Mais comment savoir si un client qui n'a pas encore d'assurance de ce type sera intéressé par le nouveau produit ?

Une solution est de chercher un profil commun aux clients qui se sont déjà montrés intéressés par un produit de ce type pour cibler parmi tous les clients ceux qui ont un profil analogue. Que sera un tel profil? Dans la base de données, chaque client est décrit par un certain nombre de champs, que l'on peut supposer binaires. Par exemples : « âge inférieur à trente ans », « possède une maison », « a un ou plusieurs enfants », « vit dans une zone à risque de vol », etc. Certains champs peuvent être non remplis : les clients qui ont seulement une assurance automobile n'ont pas été interrogés à la constitution de leur dossier sur l'existence d'un système d'alarme dans leur appartement.

Une façon de constituer un profil consiste à découvrir des associations dans les données, c'est-à-dire des implications logiques approximatives. Disons par exemple que la plupart des clients qui possèdent déjà une assurance contre le vol d'objets de valeur à domicile sont plutôt âgés et n'ont en général qu'une voiture, mais haut de gamme. Il semble raisonnable de démarcher parmi

Copyright © 2010 Eyrolles.

tous les clients ceux qui répondent au même profil. L'hypothèse est donc que posséder une seule voiture (mais de luxe) et être d'âge mûr est un profil qui implique sans doute la possession à domicile d'objets de valeur.

Ce type d'apprentissage relève de l'extraction de connaissances et de l'apprentissage non supervisé. Ce dernier terme signifie que le programme d'apprentissage se débrouille sans professeur : l'expertise est présente dans les données, mais de manière implicite, c'est au programme de la découvrir et de l'utiliser. La combinatoire sous-jacente à ce type d'apprentissage est évidemment très importante.

Organisation et plan de l'ouvrage

L'organisation de cet ouvrage reflète en partie des caractéristiques importantes des exemples précédents. Nous disons que, pour le premier, l'espace de recherche est peu structuré. Pourquoi cela? Parce qu'on ne peut pas dire si telle droite est meilleure que telle autre sans les tester toutes les deux explicitement sur les données. Il s'agit de ce qu'on appelle en général un problème d'optimisation. En revanche, dans le second exemple, nous avons mentionné une relation d'ordre entre deux solutions, intimement liée à leur qualité relative. Dans ce cas, une exploration partielle de l'ensemble des solutions est possible ; elle sera guidée à la fois par sa structure algébrique et par les données, exemples et contre-exemples, alors que seules les données pouvaient conduire le programme dans le premier cas. Pour le troisième exemple, il n'y a même pas de guidage dans l'espace de recherche par des exemples et des contre-exemples.

Ces remarques sont cruciales pour la conception des algorithmes. C'est la raison pour laquelle nous avons choisi d'organiser le présent ouvrage selon le critère suivant : nous traitons les méthodes d'apprentissage en commençant par celles pour lesquelles l'espace de représentation des concepts à apprendre est fortement structuré, puis celles pour lesquelles cette hypothèse doit être affaiblie et enfin celles pour lesquelles l'information *a priori* sur la nature des concepts à apprendre est très faible ou nulle.

Partie 1 : Les fondements de l'apprentissage

Une partie de fondements méthodologiques est d'abord nécessaire. Nous y faisons une présentation générale de la problématique de l'apprentissage et nous donnons les définitions de base (**Chapitre 1 : De l'apprentissage naturel à l'apprentissage artificiel**). Le chapitre suivant propose une introduction aux théories de l'apprentissage par généralisation (**Chapitre 2 : Première approche théorique de l'induction**). Un approfondissement de ce thème sera mené au chapitre 21. Le troisième chapitre traite de la représentation des données et des connaissances et des types d'algorithmes qui sont mis en jeu par la suite (**Chapitre 3 : L'environnement méthodologique de l'apprentissage**).

Partie 2 : Apprentissage par exploration

Nous analysons dans la deuxième partie les méthodes d'apprentissage quand les représentations des concepts forment des ensembles fortement structurés. Nous l'avons appelée *l'apprentissage par exploration*. On y trouve d'abord une méthode très générale (**Chapitre 4 : Induction et relation d'ordre : l'espace des versions**), puis un chapitre sur l'apprentissage dans la logique des prédicats (**Chapitre 5 : La programmation logique inductive**). Le chapitre suivant complète ce point de vue en montrant comment modifier des concepts dans des espaces

Copyright © 2010 Eyrolles.

structurés ; il aborde aussi l'apprentissage par analogie (**Chapitre 6 : Transfert de connaissances et apprentissage par analogie**). Le septième chapitre (**Chapitre 7 : L'inférence grammaticale**) traite de l'apprentissage des automates et des grammaires. Enfin les méthodes d'apprentissage par évolution simulée, fondées sur l'exploration par algorithmes génétiques, sont exposées (**Chapitre 8 : Apprentissage par évolution simulée**).

Partie 3 : Apprentissage par optimisation et interpolation

Dans la troisième partie, les connaissances sur la structure des espaces sont plus faibles. En fait, il n'existe plus qu'une mesure de qualité des hypothèses induites par le critère inductif et une distance dans l'espace des hypothèse permettant de mettre en œuvre des techniques d'optimisation par gradient. Il s'agit de l'*apprentissage par optimisation*. On y trouve le problème de l'apprentissage de droites, soit pour la régression, soit pour la classification, mentionné ci-dessus, et leur généralisation à des hyperplans (**Chapitre 9 : L'apprentissage de modèles linéaires**). Une extension désormais classique mène aux réseaux connexionnistes multicouche (**Chapitre 10 : L'apprentissage de réseaux connexionnistes**). Le fonctionnement et l'apprentissage des réseaux de probabilités conditionnelles est ensuite abordé (**Chapitre 11 : L'apprentissage de réseaux bayésiens**). Le chapitre suivant traite comme le chapitre 7 de l'apprentissage de certaines machines à produire des séquences, mais sous un aspect probabiliste (**Chapitre 12 : L'apprentissage de modèles de Markov cachés**).

Pour finir cette partie, nous abordons dans le chapitre suivant l'apprentissage des arbres de décision (**Chapitre 13 : Apprentissage par inférence d'arbres**).

Partie 4 : Apprentissage par approximation

La partie suivante, intitulée l'*apprentissage par approximation et interpolation*, traite des méthodes les moins informées, celles où l'espace des concepts cherchés possède le moins de propriétés.

Elle commence par un chapitre qui expose un ensemble de méthodes récentes qui permettent de prendre des décisions non linéaires tout en s'appuyant sur des méthodes linéaires (**Chapitre 14 : Méthodes à noyaux**).

Nous décrivons ensuite les techniques d'apprentissage de règles de classification par des méthodes statistiques qui cherchent à approcher la règle de décision bayésienne. Ce chapitre inclut aussi certains aspects de l'apprentissage par analogie, en particulier la méthodes des plus proches voisins (**Chapitre 15 : L'apprentissage bayésien et son approximation**). Le chapitre suivant s'intéresse à un apprentissage numérique de type « punition » ou « récompense », typiquement applicable à l'apprentissage de son comportement par un robot (**Chapitre 16 : L'apprentissage de réflexes par renforcement**).

Partie 5 : Au-delà de l'apprentissage supervisé.

La partie suivante, intitulée *au-delà de l'apprentissage supervisé* a pour ambition de regrouper un ensemble de techniques autour ou au-dessus de l'apprentissage supervisé. Le **Chapitre 17 : Apprentissage de combinaison d'experts** s'intéresse à la combinaison de plusieurs méthodes de classification pour qu'elles s'entraident sur un problème donné. La méthode du *boosting* y est en particulier traitée. Dans le chapitre suivant, on s'intéresse aux données non étiquetées par un expert : il s'agit de les organiser et d'y découvrir des régularités et des associations (**Chapitre 18 : La classification non supervisée et la fouille de données**). Ensuite, nous développons le cas où seulement une partie des exemples est supervisée : il s'agit de tirer parti des exemples non supervisés pour aider la classification (**Chapitre 19 : L'apprentissage semi-supervisé**).

Copyright © 2010 Eyrolles.

Le chapitre suivant dresse un panorama non exhaustif mais évocateur de nouvelles directions de recherche visant à dépasser les idées classiques de l'apprentissage artificiel (**Chapitre 20 : Vers de nouvelles tâches et de nouvelles questions**). Pour terminer, le chapitre suivant revient sur les aspects théoriques de l'apprentissage abordés au Chapitre 2 et explore la notion d'induction sous ses aspects mathématiques, avant d'évoquer d'autres cadres possibles d'apprentissage (**Chapitre 21 : Analyse de l'induction : approfondissements et ouvertures**).

Ce livre se termine par des **annexes**, qui détaillent des points techniques ; celles qui traitent de l'algorithme **estimation-maximisation** et de l'**optimisation par gradient** sont sans doute les plus référencées dans les chapitres précédents.

Finalement, une **bibliographie** fournit les ouvrages fondamentaux et les articles plus pointus cités dans le texte.

Guide de lecture

Après plus de quarante ans de recherches et de réalisations en apprentissage artificiel, il est difficile pour un non initié de savoir comment aborder ce domaine et comment s'y orienter. Nos collaborations avec des utilisateurs des techniques d'apprentissage et avec des chercheurs d'autres disciplines, comme notre activité d'enseignement et d'encadrement avec nos étudiants nous ont amplement montré l'intérêt d'un ouvrage d'introduction cohérent, articulé autour de grandes lignes directrices.

Il existe déjà des livres d'enseignement et de recherche sur pratiquement chaque sujet abordé dans les chapitres de ce livre et nous ne manquons pas d'y faire référence. Mais si la somme des connaissances qui s'y trouvent est au total bien supérieure à celle contenue dans notre livre, leur lecture conduit à des contradictions dans les notations, à des approfondissements théoriques de niveau très variable, à des analyses différentes du même problème et à des présentations redondantes voire contradictoires des mêmes sujets.

Il nous a donc paru que la discipline de l'apprentissage artificiel pouvait être présentée de manière unifiée, dans un but d'abord didactique. Ce souci commande le fond comme la forme de cet ouvrage.

Compte tenu de la variété technique des sujets abordés et de l'intérêt personnel de chaque lecteur (autodidacte, enseignant ou étudiant), des parcours de lecture différents peuvent être suivis.

Nous proposons en particulier les itinéraires suivants, mais ce n'est pas exclusif :

1. Pour une **vue d'ensemble** sur l'apprentissage artificiel, un rapide aperçu des méthodes et un point de vue sur leurs applications : chapitres 1, 2 (paragraphes 2.1 et 2.2), 3 et 4.
2. Pour un point de vue approfondi sur les **principes méthodologiques**, en particulier statistiques de l'apprentissage : chapitres 1, 2, 3, 8, 9, 13, 14, 15, 17, 18, 19, 20 et 21.
3. Pour comprendre l'apprentissage de phénomènes essentiellement numériques et pour les applications à la **reconnaissance des formes** : chapitres 1, 2 (paragraphes 2.1 et 2.2), 3, 9, 10, 12, 13, 15, 16, 17 et 18.
4. Pour acquérir un point de vue sur l'apprentissage dans les **systèmes experts** et le traitement des **données symboliques** : chapitres 1, 3, 4, 5, 6, 7, 8, 11, 15 et éventuellement 16.
5. Pour qui veut réaliser l'apprentissage de concepts à partir de **données séquentielles** (signaux, textes, etc.) : chapitres 1, 3, 5, 7 et 12.

Copyright © 2010 Eyrolles.

6. Pour qui s'intéresse plus à la **robotique**, l'apprentissage de **comportement** : chapitres 1, 3, 7, 11, 12, 15 et 16.
7. Pour qui s'intéresse à la **fouille de données**, à l'**extraction de connaissances** : chapitres 9, 10, 17, 11, 15 et 18. 9, 10, 11, 13, 15 et 18.

Copyright © 2010 Eyrolles.

Glossaire français-anglais [3]

Apprenant	*Learner*
Apprentissage actif	*Active learning*
Apprentissage en ligne	*On-line learning*
Apprentissage hors ligne	*Batch learning*
Apprentissage par examen de preuve (EBL)	*Explanation-based learning*
Apprentissage quand $\mathcal{F} \neq \mathcal{H}$	*Agnostic learning*
Apprentissage de tri	*Learning to rank*
Astuce de représentation unique	*Single representation trick*
Attribut	*Feature*
Biais de représentation	*Représentation bias*
Catégorisation	*Clustering*
Classes déséquilibrées	*Imbalanced data sets*
Connaissance *a priori*	*Domain theory (sometimes hints)*
Échantillon d'apprentissage	*Learning set*
Échantillon de test	*Test set*
Échantillon de validation	*Validation set*
Exemple critique	*Support vector or near-miss*
Fonction de coût	*Loss function*
Fonction cible	*Target function*
Fonction de performance	*Fitness function*
Hypothèse correcte (par rapport à des données)	*Consistent hypothesis*
Méthodes à noyaux	*Kernel methods*
Optimisation de performance	*Speed-up learning*
Perceptron multi-couches (PMC)	*Multi-layer perceptron (MLP)*
Point aberrant	*Outlier*
Principe de longueur minimale de description	*Minimum description length principle*
Programmation logique inductive (PLI)	*Inductive logic programing (ILP)*
Raisonnement à partir de cas	*Case-based reasoning*
Reconnaissance des formes	*Pattern recognition*
Réseaux bayésiens	*Bayes nets or graphical models*
Réseaux connexionnistes	*Neural networks*
Risque empirique	*Empirical risk*
Risque réel	*Expected risk or true risk*
Séparateurs à vaste marge	*Support vector machines*
Suradaptation *ou* surapprentissage	*Over-fitting*
Système de classification	*Classifier*

[3] Les termes importants sont traduits en *anglais* au fil du texte.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Notations

Notations générales

$\#\ldots$	Nombre de ...
$\mathbf{P}$	Une probabilité
$\mathbf{p}$	Une densité de probabilités
$\mathbb{E}[x]$	L'espérance de la variable x
$\mathbb{N}$	L'ensemble des entiers naturels
$\mathbb{R}^d$	L'espace euclidien de dimension d
$\mathbb{B}^d = \{0,1\}^d$	L'espace booléen de dimension d
$\mathcal{O}(\cdot)$	L'ordre de grandeur maximal de complexité d'un algorithme
$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$	Un vecteur
$\mathbf{x}^\top = (x_1, \ldots, x_d)$	Un vecteur transposé
$\mathbf{x} = (x_1, \ldots, x_d)^\top$	Un vecteur
$\langle \mathbf{xy} \rangle = \mathbf{x}^\top \mathbf{y}$	Le produit scalaire des vecteurs $\mathbf{x}$ et $\mathbf{y}$
$\| \mathbf{x} \|$	La norme du vecteur $\mathbf{x}$
$\mathbf{M}^{-1}$	La matrice inverse d'une matrice carrée M
$\mathbf{M}^\top$	La matrice transposée d'une matrice M
$\mathbf{M}^+$	La matrice pseudo-inverse d'une matrice M. Par définition, $\mathbf{M}^+ = \mathbf{M}^\top (\mathbf{M}\mathbf{M}^\top)^{-1}$
$\Delta(\mathbf{x}, \mathbf{y})$	La distance euclidienne entre deux vecteurs $\mathbf{x}$ et y de $\mathbb{R}^d$
$\frac{\partial}{\partial x} f(x, y)$	La dérivée partielle par rapport à x de la fonction f des deux variables x et y
$\nabla_{\mathbf{A}} J(\mathbf{A}, \mathbf{B})$	Le vecteur dérivé par rapport au vecteur $\mathbf{A}$ de la fonctionnelle J des deux vecteurs $\mathbf{A}$ et $\mathbf{B}$

Copyright © 2010 Eyrolles.

Les éléments en jeu dans l'apprentissage

$\mathcal{X}$	L'espace de représentation des objets (des formes)
$\mathcal{Y}$	L'espace de sortie
F	Espace de redescription ou « *Feature space* »
$\mathcal{S}$	L'échantillon d'apprentissage (un ensemble ou une suite d'exemples)
$\mathcal{S}_+$	Les exemples positifs
$\mathcal{S}_-$	Les exemples négatifs
$\mathcal{A}$	L'échantillon d'apprentissage quand on divise $\mathcal{S}$ en $\mathcal{A}$, $\mathcal{T}$ et $\mathcal{V}$
$\mathcal{T}$	L'échantillon de test
$\mathcal{V}$	L'échantillon de validation
m	La taille d'un échantillon d'apprentissage (le nombre d'exemples)
$\mathbf{z}_i = (\mathbf{x}_i, u_i)$	Un exemple (élément d'un échantillon d'apprentissage)
$\mathbf{x}_i$	La description d'un objet dans un espace de représentation
x_{ij}	La valeur de la coordonnée j de la description de l'objet x_i dans $\mathbb{R}^d$

Les principes de l'apprentissage inductif

$u_i \in \mathcal{Y}$	La supervision, ou sortie désirée, d'un exemple
$f : \mathcal{X} \to \mathcal{Y}$	La fonction cible (celle que l'on cherche à apprendre)
$\mathcal{H}$	L'espace des hypothèses d'apprentissage
$h \in \mathcal{H}$	Une hypothèse produite par un apprenant (un algorithme d'apprentissage)
$y = h(\mathbf{x}) \in \mathcal{Y}$	La prédiction faite par l'hypothèse h sur la description $\mathbf{x}$ d'un exemple
$\ell(f, h)$	La perte (ou distance) entre la fonction cible et une hypothèse
$R_{\text{Réel}}(h)$	Le risque réel associé à l'hypothèse h
$R_{\text{Emp}}(h)$	Le risque empirique associé à l'hypothèse h
$R^\star$	Le risque (optimal) de la règle de décision de Bayes
h^*	L'hypothèse de $\mathcal{H}$ qui minimise le risque réel
$h^*_{\mathcal{S}}$	L'hypothèse de $\mathcal{H}$ qui minimise le risque empirique sur $\mathcal{S}$
$\hat{h}^*_{\mathcal{S}}$	L'hypothèse trouvée par l'algorithme d'apprentissage ayant $\mathcal{S}$ en entrée et cherchant $h^*_{\mathcal{S}}$ dans $\mathcal{H}$

L'apprentissage d'une règle de classification

$\mathcal{C}$	L'ensemble des classes
C	Le nombre de classes
ω_i	Une classe de $\mathcal{C}$

La logique

$a \wedge b$	a *ET* b, quand a et b sont des valeurs binaires
$a \vee b$	a *OU* b
$\neg a$	*NON* a
$a \to b$	a implique b

Copyright © 2010 Eyrolles.

Première partie

Les fondements de l'apprentissage

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 1

De l'apprentissage naturel à l'apprentissage artificiel

Sommaire

1 L'apprentissage artificiel 4

2 Deux exemples : apprendre à jouer, apprendre à lire 6

2.1 Apprendre à jouer 6

2.2 Apprendre à reconnaître des caractères manuscrits 7

3 Deux approches : la cybernétique et les sciences cognitives 10

3.1 La cybernétique 11

3.2 Le pari du cognitivisme 12

4 Les concepts de base de l'apprentissage 14

4.1 Un scénario de base pour l'induction 14

4.2 Quelques notions clés 14

4.2.1 Le critère de succès 14

4.2.2 Notion de protocole d'apprentissage 15

4.2.3 Notion de tâche d'apprentissage 16

4.3 L'induction considérée comme estimation de fonction 19

5 L'induction comme un jeu entre espaces 22

5.1 L'apprentissage est impossible... 23

5.2 ...sans limiter l'espace des hypothèses 25

5.3 L'exploration de l'espace des hypothèses 27

6 Retour sur l'organisation de l'ouvrage 29

Copyright © 2010 Eyrolles.

1. L'apprentissage artificiel

Même les machines ont besoin d'apprendre.

Depuis plus d'un demi-siècle, les chercheurs en intelligence artificielle travaillent à programmer des machines capables d'effectuer des tâches qui requièrent l'exercice de l'intelligence. Nous citerons l'*aide à la décision*, par exemple l'aide au diagnostic médical ; la *reconnaissance de formes*, par exemple la reconnaissance de la parole ou la vision artificielle ; le *contrôle de processus*, par exemple la conduite de procédés industriels; la *prédiction*, par exemple la prédiction de consommation électrique ou la prédiction de cours boursiers; la *conduite de robots*, y compris d'équipes de robots comme dans la RoboCup[1] ; l'exploration de grandes bases de données (on dit aussi la *fouille de données*), tant il est vrai que si nous croulons sous les informations, il nous manque souvent la connaissance. Chacune de ces tâches et bien d'autres ont stimulé l'inventivité des chercheurs et donné lieu à de nombreuses réalisations impressionnantes. Cependant, programmer des machines capables de s'adapter à toutes les situations et éventuellement d'évoluer en fonction de nouvelles contraintes est difficile. L'enjeu est de contourner cette difficulté en dotant la machine de capacités d'apprentissage lui permettant de tirer parti de son expérience. C'est pourquoi, parallèlement aux recherches sur le raisonnement automatique, se sont développées des recherches sur l'apprentissage par les machines. Avant d'aborder ce type d'apprentissage, examinons rapidement certaines activités d'apprentissage par des organismes naturels.

L'apprentissage naturel

Dès sa naissance, un enfant apprend à *reconnaître* l'odeur de sa mère, puis sa voix et plus largement l'ambiance du lieu où il vit. Ensuite, il apprend à *coordonner* ses perceptions, comme sa vue ou son toucher, avec ses mouvements. Par des *essais* gratifiants ou pénalisants, il apprend plus tard à marcher, manifestant une grande capacité à *intégrer* des signaux différents : la vue, le sens de l'équilibre, la proprioception, la coordination motrice. Il apprend pendant le même temps à segmenter et *catégoriser* des sons et à les associer à des significations. Il apprend aussi la structure de sa langue maternelle et acquiert simultanément un répertoire organisé de *connaissances* sur le monde qui l'environne.

Il va aussi apprendre à lire. Il sait déjà faire la distinction entre texte et non texte, parce qu'il a souvent manipulé des livres illustrés où il a observé l'association des images et des symboles de l'écriture. Il apprend d'abord *par cœur* des mots associés à des sons et à leur signification. Plus tard, il extrait des *règles* permettant de distinguer des groupements syllabiques à l'intérieur des mots et de les prononcer. Cet apprentissage est long et *progressif*, et demande des répétitions et des séquences d'exercices bien choisies. Il est en partie *supervisé* par des adultes qui préparent les tâches d'apprentissage, accompagnent son cheminement et sanctionnent, par récompense ou punition, les résultats observés.

Au cours des années qui suivent, l'enfant apprend par étapes à maîtriser des concepts et des opérations de plus en plus abstraits. Finalement, cette fois sans professeur pour l'escorter, il découvrira et énoncera des points de vue personnels, des *théories* sur les phénomènes sociaux, sportifs, économiques, naturels et autres.

Les modalités de l'apprentissage naturel sont donc *multiples* : apprentissage par cœur, par instruction, par généralisation, par découverte, apprentissage impliquant des catégorisations voire

[1] La RoboCup est une compétition annuelle organisée depuis 1997 entre équipes de robots footballeurs. Il existe plusieurs types de compétitions mettant en jeu soit des agents simulés, soit des robots réels de tailles diverses. Pour plus d'information, voir par exemple http://www.robocup.org/.

Copyright © 2010 Eyrolles.

la formation de théories, apprentissage plus ou moins supervisé ou autonome, etc. Ces diverses formes d'apprentissage auront-elles une contrepartie lorsqu'il s'agira d'apprentissage par des machines ? Et comment envisagera-t-on l'apprentissage naturel après cette étude ?

Apprentissage « artificiel » ou apprentissage « automatique » ?

Au fait, comment appeler cette discipline ? Un terme courant est *apprentissage automatique*. Cependant, bien que consacré par l'habitude, il ne nous semble pas complètement satisfaisant. Il sous-entend en particulier une sorte d'activité inconsciente de bas niveau, qui s'exécute « en tâche de fond » comme disent les informaticiens pour parler d'un processus se déroulant au second plan sans perturber la tâche principale courante. Si certains types d'apprentissages, comme l'habituation, voire même certaines formes d'associations (comme chez le fameux chien de Pavlov), peuvent correspondre à ce schéma, celui-ci est cependant beaucoup trop restrictif.

On peut aussi penser à utiliser les expressions *apprentissage machine* pour traduire directement l'expression américaine *machine learning* ou à *apprentissage algorithmique* pour insister sur les aspects opérationnels.

Il nous semble que la notion d'*apprentissage artificiel* apporte quelque chose de plus profond que la simple idée d'« automatique ». Il est vrai que le mot *artificiel* évoque aussi quelque chose de factice, voire de frelaté et que nous savons combien le terme d'*intelligence artificielle* a souffert de ces connotations ; mais nous nous plaçons ici sous le patronage de Herbert Simon (1916-2001), prix Nobel d'économie et l'un des fondateurs de l'intelligence artificielle, qui a bien su montrer la marque et l'intérêt de la notion de *sciences de l'artificiel* [Sim81].

Sciences naturelles et sciences de l'artificiel

Le projet des *sciences naturelles* est de comprendre les phénomènes en formulant des lois sous-jacentes, de préférence simples. L'ambition fondamentale des *sciences de l'artificiel* n'est pas différente mais, par le but poursuivi et les moyens utilisés, elles s'en écartent cependant suffisamment pour se définir à part entière. Ainsi, le but des sciences de l'artificiel, en particulier de l'apprentissage artificiel, est bien de comprendre les phénomènes de la nature. Mais cette compréhension doit passer par la construction de *modèles* qui (naturellement pour des informaticiens) doivent être capables de réaliser des *simulations*.

Selon le point de vue des sciences de l'artificiel, comprendre implique la capacité de fabriquer pour *reproduire*. Connaître, dans cette optique, c'est concevoir un modèle opératoire du monde pour le soumettre à des manipulations réglées. Connaître, c'est donc prendre de la distance par rapport à l'objet et se donner les moyens de l'approcher dans son comportement, d'en faire varier des paramètres et d'énoncer des conditions de réalisabilité.

Les sciences de l'artificiel présentent deux aspects qui les distinguent des sciences naturelles.

- D'une part, elles conçoivent la connaissance et la compréhension comme une *capacité de simulation*, ce qui implique la possibilité d'explorer *effectivement* les conséquences de postulats initiaux.
- D'autre part, ce sont des sciences qui cherchent des normes permettant de définir ce qu'est un raisonnement valide, un apprentissage correct et les conditions nécessaires pour qu'il puisse avoir lieu. En ceci, les sciences de l'artificiel sont aussi des *sciences normatives*, par opposition à l'aspect principalement descriptif des sciences naturelles.

C'est dans ce double sens que nous désirons présenter l'apprentissage artificiel dans cet ouvrage. Certes, il sera bien question d'apprentissage automatisable, donc d'apprentissage automatique et d'un apprentissage réalisable sur des machines, donc d'apprentissage machine, mais l'un des soucis sous-jacents sera de rechercher les conditions de réalisabilité des modèles de l'apprentissage,

Copyright © 2010 Eyrolles.

c'est-à-dire les lois profondes qui règlent la possibilité d'apprendre. Ainsi, l'apprentissage artificiel est la science qui cherche et établit des liens entre les principes généraux d'*apprenabilité* et les méthodes et outils permettant de *réaliser* un apprentissage dans un contexte particulier. La première partie de l'ouvrage est davantage tournée vers l'exposé des principes tandis que le reste présente des techniques justifiées en particulier à la lumière des principes fondamentaux. Le théoricien et l'ingénieur établissent ainsi un dialogue. Nous avons cherché à conserver cet esprit dans l'organisation de l'ouvrage.

2. Deux exemples : apprendre à jouer, apprendre à lire

Avant de fixer un cadre méthodologique et de présenter les concepts de base sur lesquels s'organise cet ouvrage, examinons brièvement deux tâches d'apprentissage, simples en apparence, pour faire émerger les multiples questions sous-jacentes.

2.1 Apprendre à jouer

Prenons d'abord le cas d'un jeu à deux adversaires sans hasard et sans information cachée. Le jeu d'échecs, le jeu de go ou le morpion en sont les exemples les plus immédiats. Supposons que l'on veuille faire apprendre à une machine à jouer à l'un de ces jeux. Comment s'y prendre ?

Il faut d'abord définir exactement le but poursuivi. S'agit-il de faire apprendre les *règles du jeu* à la machine à partir d'observations de parties jouées ? S'agit-il de lui faire apprendre à *bien jouer*? S'agit-il de lui faire découvrir les variables pertinentes pour *prédire* comment jouera l'adversaire (son niveau, son style de jeu) ? S'agit-il de prédire le gagnant, ou bien le nombre de coups restant à jouer ? La liste est déjà variée, elle n'est pas exhaustive.

Prenons maintenant la situation du jeu d'échecs schématisée dans la figure 1.1. Imaginons que ce soit à la machine (pièces blanches) de choisir son prochain coup. En supposant qu'elle connaisse les règles du jeu, elle a le choix entre plusieurs dizaines de coups légaux. Lequel est le meilleur ? Pour simplifier, choisissons comme critère de succès une grandeur liée simplement au gain de la partie, en ignorant la durée de jeu, le nombre de coups restant, etc. Comment déterminer le coup à jouer ? L'approche classique en intelligence artificielle utilise l'algorithme MinMax |RN10| fondé sur la notion de *fonction d'évaluation*. En pratique, dans cet algorithme, la machine effectue une recherche en avant dans l'arbre des coups possibles, aussi loin que le temps et les ressources de calcul le lui permettent (aux échecs, une dizaine de demi-coups environ). Ensuite, elle *évalue* chaque position atteinte en fonction de certains critères (par exemple : l'occupation du centre, l'avantage matériel, etc.), et finalement joue le coup lui permettant de maximiser le gain que l'adversaire est obligé de lui concéder. Dans ce cadre, l'apprentissage consiste naturellement à apprendre cette fonction d'évaluation, car c'est elle qui détermine la qualité des décisions.

Mais d'autres possibilités sont envisageables. Par exemple, la machine pourrait simplement apprendre par cœur une table d'association entre une position et le coup à jouer, une *look-up table* géante. Évidemment, dans le cas des jeux intéressants, ceux qui comportent un très grand espace de situations possibles, cela semble absurde. Nous verrons cependant au chapitre 16, dédié à l'apprentissage par renforcement, comment rendre cette idée possible. D'autre part, il est clair que les joueurs humains prennent des décisions motivées par des considérations de stratégie et de tactique : ils opèrent rarement par une exploration exhaustive avec mise en œuvre d'une fonction d'évaluation « simple ». Une simulation plus fidèle de l'apprentissage naturel pourrait chercher à identifier tactiques et stratégies, et à les apprendre directement.

Copyright © 2010 Eyrolles.



FIG. 1.1: *Une position dans une partie d'échecs.*

Un problème qui n'a pas encore été abordé est celui du *choix* des données d'apprentissage : quelles sont les informations dont le système apprenant va bénéficier pour apprendre ? Dans le cas du jeu d'échecs, il peut s'agir d'observations de parties. Doit-on lui proposer des parties jouées par des champions, ou des parties médiocres feraient-elles aussi bien l'affaire, ou peut-être des parties jouées par l'apprenant contre lui-même ? Des exemples de coups faibles sont-ils favorables, voire indispensables, à l'apprentissage ? On peut aussi penser profiter de corpus de parties commentées par un professeur, comme dans la littérature échiquéenne. Dans le cas de l'apprentissage des règles du jeu, des exemples de coups illégaux seraient-ils favorables, voire indispensables, à l'apprentissage ?

Il se pose aussi la question du *séquencement* des leçons : y a t-il un ordre de présentation plus favorable qu'un autre ? Doit-on tenir compte d'une vitesse d'assimilation, comme chez les apprenants humains, liée aux capacités de calcul et de mémoire de l'apprenant ?

Autant de questions qui ressortent tant d'une analyse théorique sur les conditions de possibilité de l'apprentissage que d'une étude expérimentale.

Finalement on n'évitera pas le problème de la *validation* de l'apprentissage réalisé. Comment mesurer la performance de l'apprenant après l'apprentissage ? En comptant la proportion de parties gagnées contre un échantillon représentatif de joueurs ? En incluant le temps de réflexion ? En demandant des explications des décisions prises ? Et dans ce cas, qui jugera de leur validité?

Même dans le contexte familier et apparemment simple de l'apprentissage d'un jeu, on voit donc que l'ensemble des questions ouvertes est vaste.

2.2 Apprendre à reconnaître des caractères manuscrits

Maintenant, supposons que nous voulions entraîner une machine à reconnaître des caractères manuscrits tels qu'ils apparaîssent dans l'adresse figurant sur une enveloppe, c'est-à-dire en général assez bien tracés et séparés les uns des autres. La figure 1.2 donne un exemple de caractères tels qu'ils peuvent se présenter dans la réalité. Comment une machine peut-elle apprendre à identifier ces formes ?

Copyright © 2010 Eyrolles.

Le codage

La difficulté de base est que la variété des formes rencontrées est infinie. Il ne peut donc être question d'apprentissage par cœur. Il faut par conséquent, à partir d'un échantillon d'exemples « bien choisis »(comment ?) être capable de *généraliser*. De manière informelle, nous définissons un exemple comme l'association d'une forme et d'une étiquette. C'est ainsi que la forme de la figure 1.2 est associée à l'étiquette 'a' (lettre de la catégorie 'a'). Nous avons alors affaire à ce qu'on appelle de l'*apprentissage supervisé* [2].

Ici se pose la première question : comment coder les formes? Par une matrice binaire transcrivant l'éclairement des pixels de la rétine de la caméra ? Dans cette hypothèse, chaque caractère serait défini par une matrice, disons de taille 16×32 [3]. Avant de s'intéresser à l'exploitation de telles représentations, il faut résoudre un problème d'homogénéité. Les caractères seront-ils centrés sur la rétine ? Seront-ils tous à la même échelle ? Auront-ils une orientation imposée ? On voit que même dans le cas d'un codage très primitif des formes, un *prétraitement* est indispensable.

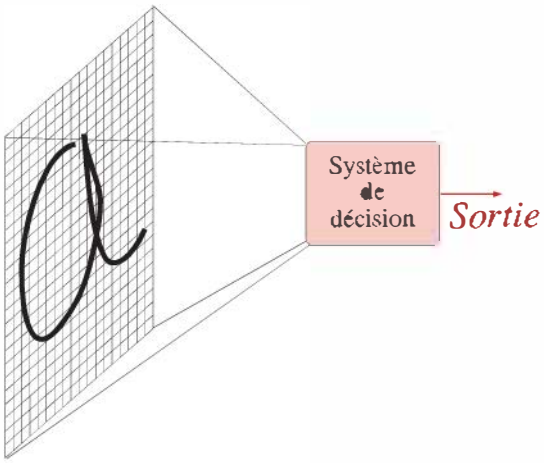


FIG. 1.2: *Une tâche de reconnaissance de caractères manuscrits.*

Pourquoi ne pas d'emblée coder les formes de manière plus pertinente, en extrayant des caractéristiques essentielles ou invariantes? Par exemple la présence de boucles, ou bien la hauteur ou la largeur ou le nombre de points de retour. Il est probable que l'apprentissage subséquent s'en trouverait facilité. Mais... qui a appris ce codage ? Et comment ?

Alors que la description des formes comme des projections sur la rétine de la caméra est immédiate, une redescription adaptée à l'apprentissage implique des opérations non triviales et surtout des connaissances *a priori* sur le mécanisme de cet apprentissage. Il s'agit d'éliminer les descripteurs non pertinents, par exemple la couleur de l'encre ou celle du fond de l'image, de recoder pour tenir compte des invariances par translation ou par changement d'échelle, voire d'introduire de nouveaux descripteurs. Certains de ces nouveaux descripteurs, non présents dans la description brute des données, n'impliquent pas nessairement des attributs complexes. Ainsi, pour distinguer un '*a*' d'un '*b*', il suffit en général de considérer le rapport de leur hauteur à leur largeur. Mais, le plus souvent, il faudra être capable d'inventer des descripteurs sophistiqués. Une autre technique consiste à calculer une grande collection de descripteurs à partir des descripteurs primitifs dont l'apprentissage essaiera de tirer le meilleur parti.

[2] Par contraste avec l'apprentissage non supervisé dans lequel les étiquettes ne sont pas fournies.

[3] Notons qu'après ce codage le nombre de formes possible n'est plus *stricto sensu* infini. Si chaque pixel est noir ou blanc, le nombre de formes différentes est cependant de $2^{16 \times 32} \simeq 10^{30}$.

Copyright © 2010 Eyrolles.

La mesure de performance

Retournons maintenant au problème de la définition du critère de performance. S'agit-il simplement du nombre d'erreurs de classification après apprentissage, que l'on peut ramener à une *probabilité de mauvaise classification*? S'agit-il d'une mesure de risque plus élaborée, prenant en compte le fait qu'il vaut mieux se tromper sur une lettre que sur un chiffre (le code postal est plus difficile à reconstruire que le nom de la commune) ? Ici encore, les possibilités sont nombreuses et c'est l'application qui commande le choix à faire.

Dans tous les cas, l'évaluation de l'apprentissage devra être conduite avec soin. En général, on mesure la performance après le déroulement de l'apprentissage sur un certain nombre de données que l'on appelle *échantillon d'apprentissage*. Si l'on fait varier la taille de cet échantillon, on obtient une *courbe d'apprentissage* comme celle donnée sur la figure 1.3. Cependant, il faut s'assurer que la mesure de performance s'effectue sur un *échantillon de test* différent de l'échantillon d'apprentissage. Autrement, ce ne serait pas la capacité de généralisation qui serait testée, mais une capacité à l'*apprentissage par cœur*, qui n'est pas pertinente dans ce contexte (mais qui pourrait éventuellement l'être dans le cas de caractères d'imprimerie).

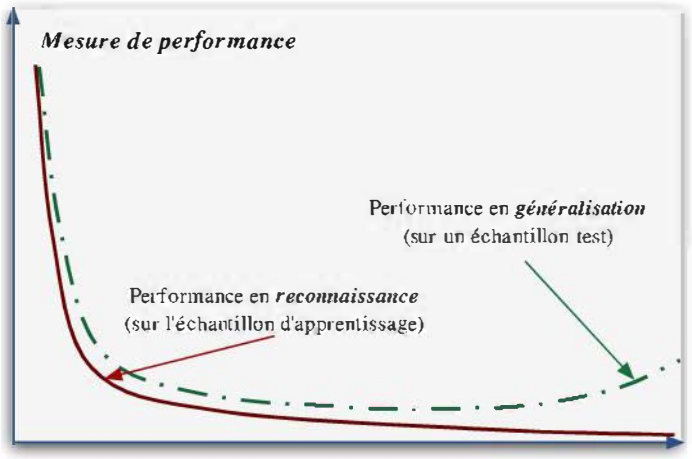


FIG. 1.3: Courbes de performance. *On suppose ici que l'apprentissage est supervisé : la machine peut exploiter les informations contenues dans un échantillon d'apprentissage en vue d'établir des prédictions sur des observations nouvelles. En abscisse figure l'exploitation de l'échantillon d'apprentissage, par exemple mesurée en nombre de fois où l'échantillon est utilisé par le système d'apprentissage. En ordonnée est portée une mesure de performance, par exemple le taux d'erreur du système en prédiction sur des exemples vus en apprentissage (erreur de reconnaissance), soit sur des exemples non vus (erreur en généralisation). Normalement, la performance s'améliore au fur et à mesure que l'algorithme exploite davantage l'information (le taux d'erreur diminue). Elle est généralement meilleure en reconnaissance qu'en généralisation. On observe aussi fréquemment une baisse des performances en généralisation lorsque l'apprenant exploite « trop » les données. Ce phénomène, appelé surapprentissage (ou parfois suradaptation), qui peut sembler paradoxal trouvera son explication dans la suite de l'ouvrage.*

Copyright © 2010 Eyrolles.

La modélisation

Finalement, il faudra décider de la forme d'apprentissage à réaliser, c'est-à-dire de ce qui est appris en interne par le système apprenant. Pour donner un exemple, on peut se figurer les caractères comme étant décrits dans un espace de descripteurs à plusieurs dimensions. Certains des points de cet espace correspondent à la lettre '*a*', d'autres à la lettre '*b*', etc. Le problème est alors d'apprendre à associer à chaque point la lettre correspondante. Ceci peut être effectué de plusieurs manières. Le but de cet ouvrage est de les présenter et d'offrir un cadre conceptuel pour orienter les choix à opérer. Pour donner déjà quelques exemples, on peut imaginer une approche *géométrique* : apprendre des frontières entre les régions correspondant aux différentes classes. Une nouvelle forme inconnue sera alors étiquetée en fonction de sa place par rapport aux frontières trouvées. On peut aussi adopter un point de vue *probabiliste* et apprendre des probabilités conditionnelles d'appartenance des points aux classes, ou réciproquement des probabilités conditionnelles des classes connaissant la description des points. On pourrait également envisager d'utiliser un critère de décision par les *plus proches voisins* dont on connaît l'étiquette. Il faudra alors disposer d'un nombre suffisant de points étiquetés et d'une relation de voisinage définie proprement. Et il existe encore bien d'autres possibilités...

3. Deux approches : la cybernétique et les sciences cognitives

Commencer à réfléchir à l'apprentissage, c'est naturellement évoquer d'autres notions. D'abord, celles d'*évolution*, de *transformation*, de *modification*. Il n'y a pas d'apprentissage sans évolution. Mais cette notion d'évolution n'est pas suffisante. Le vin aussi évolue, on ne dira pourtant pas qu'il apprend. Il faut donc y ajouter la notion de *progrès*. L'apprentissage vise à rendre plus performant, meilleur, qu'elle qu'en soit la mesure. Mais le vin qui se bonifie apprend-il ? Non. C'est que l'apprentissage implique aussi une notion d'*adaptation* à un environnement, à une tâche. Il faut qu'il y ait une sorte de rétroaction de l'environnement sur le système pour que l'on puisse qualifier d'apprentissage le processus de transformation qui en découle éventuellement. Si le vin réagit à son environnement, c'est de manière minimale, totalement passive, comme tout objet ou matière. Finalement, l'apprentissage évoque aussi souvent la notion de *transfert* à d'autres situations, voire de *généralisation*. Quelqu'un qui apprend par cœur n'est qu'un perroquet ou un « savant idiot ». On attend davantage d'un authentique apprentissage, à savoir qu'il y ait capacité à savoir tirer profit d'une expérience particulière pour faire face à une nouvelle situation suffisamment « proche » pour que le transfert de l'une à l'autre puisse être pertinent.

Transformation, progrès, adaptation, transfert et généralisation : chacune de ces notions renvoie à son tour à des questions que toute étude de l'apprentissage devra aborder.

- Qu'est-ce qui est transformé lors de l'apprentissage ? En particulier, dans une machine apprenante, comment représente-t-on ce qui détermine le comportement du système et qui subira éventuellement une modification ? Le neurobiologiste évoquera immédiatement le support biologique : les neurones, leurs connexions, les neurotransmetteurs ; le psychologue raisonnera en termes de croyances, de connaissances ; l'informaticien envisagera des réseaux de neurones artificiels, des assertions Prolog d'un système expert, etc.
- Comment peut s'effectuer le processus de transformation ?
- En réponse à quel type de sollicitation ? Qu'est-ce qui informe le système apprenant qu'il doit se transformer ? Comment mesurer son progrès ou son degré d'adaptation ?

Copyright © 2010 Eyrolles.

- Comment un observateur extérieur peut-il mesurer le progrès et attribuer à un système une capacité d'apprentissage et une amélioration de performance ?
- Comment caractériser le transfert entre deux situations? Comment mesurer leurs différences et ce que l'expérience de l'une apporte à l'expérience de l'autre ?

Nous n'épuisons pas ici l'ensemble des questions qui peuvent se poser à propos de l'apprentissage. Nous voulons seulement signaler les plus évidentes, les plus immédiates. Nous l'avons fait dans des termes volontairement généraux et relativement vagues pour ne pas d'emblée embrasser un point de vue, un type de questionnement. Dans cette section, sans vouloir, ni pouvoir naturellement être exhaustif, nous voulons brosser à grands traits deux modes d'approche qui ont constitué deux moments de l'étude de la cognition au XX^{e} siècle, et qui ont façonné le point de vue de l'apprentissage artificiel durant ces dernières décennies.

3.1 La cybernétique

Dans les années quarante et cinquante, certains scientifiques se donnent pour ambition d'édifier une science générale du fonctionnement de l'esprit. Sous l'influence du mathématicien Norbert Wiener (1894-1964), ils identifient ce programme par le terme de *cybernétique*. Leurs travaux, leurs discussions et débats donnèrent lieu à dix conférences tenues à New-York sous le nom de conférences Macy du nom de la fondation philanthropique les finançant. Une partie des articles fondateurs de ce mouvement sont reproduits dans |PT95|, tandis que |Dup94| retrace l'historique et les clivages conceptuels de la cybernétique et des sciences cognitives orthodoxes.

L'un des postulats des cybernéticiens de la première heure, c'est qu'il est vain de vouloir appréhender directement les notions d'intelligence, de conscience, de mémoire, d'anticipation, d'intentionalité. Pour étudier l'esprit, il faut le *naturaliser* ou le *matérialiser* et pour cela, d'une part *assimiler l'esprit à l'activité du cerveau* et d'autre part, *poser que celui-ci est une machine.* Il en découle une triple focalisation, d'une part sur l'agent cognitif considéré individuellement (plutôt que par exemple sur le fonctionnement global de la culture, de la société ou de l'environnement), d'autre part sur les mécanismes matériels, par exemple neurobiologiques, agissant à l'intérieur de la machine, et enfin sur les règles logiques qui sous-tendent le comportement mesuré.

Un *agent cognitif* devient ainsi un module opérant sur des *entrées* pour les transformer en *sorties*. La notion de sujet disparait pour faire place aux concepts de programme, syntaxe et information telles qu'elles figurent dans la définition de la machine de Turing (1912-1954), des systèmes asservis et autorégulés de Wiener, et de la théorie de l'information de Shannon (1916-2001). On cherche donc à préciser les entrées dont dispose un système, en essayant d'évaluer la quantité d'informations disponibles et corollairement l'effet d'une quantité plus ou moins grande d'informations sur la performance de l'agent. Les cybernéticiens se concentrent également sur les modules fonctionnels qui permettent de réaliser une fonction supérieure donnée, telle que la capacité à s'orienter dans l'espace ou à prendre une décision. De même, on cherche de quels comportements génériques est capable une structure donnée de modules fonctionnels. Plus tard, lors de la « deuxième cybernétique », on étendra cette quête à l'étude des totalités engendrées par une collection de relations et aux systèmes auto-organisés.

Dans ce cadre, les questions concernant l'apprentissage deviennent relatives à l'effet d'une quantité croissante d'informations sur la performance du système : comment celui-ci s'adapte-t-il à de nouvelles stimulations ou à un nouvel environnement, comment retrouve-t-il un nouvel équilibre? On cherche également ce qu'une modification de structure peut entraîner comme changement de performance. Avec la deuxième cybernétique, l'apprentissage devient complètement lié à la capacité du système à s'autoadapter ou à continuer à « exister » dans une nouvelle situation.

Copyright © 2010 Eyrolles.

Le cybernéticien se préoccupe de définir une structure d'unités en interaction et d'étudier, par simulation, ses capacités d'adaptation spontanée à toutes sortes de milieux.

Ce qui est intéressant, c'est que la dimension dynamique de l'apprentissage et sa fonction d'adaptation à un milieu sont prises en compte. En revanche, la mise en valeur trop exclusive de simulations expérimentales et de reproductions de comportements, par exemple éthologiques, a jusqu'à présent nuit à une construction théorique de l'apprentissage. Il y a pléthore d'expériences singulières, parfois spectaculaires, et pénurie de cadres théoriques.

Le cognitivisme, qui est en partie héritier de la cybernétique, offre curieusement une image presque inversée, nous allons le voir, avec une myopie sur l'aspect dynamique de l'apprentissage et son rôle adaptatif, mais avec une forte construction théorique.

3.2 Le pari du cognitivisme

Le cognitivisme reprend à son compte l'*approche fonctionnaliste de l'esprit* selon laquelle celui-ci peut, d'une part, être abordé par ses manifestations sans avoir à présupposer de quelconques facultés mentalistes, et, d'autre part, être considéré comme *une fonction calculable*, c'est-à-dire réalisable par une machine de Turing. Cette dernière idée permet en outre d'envisager la séparation de l'aspect matériel (le *hardware*) de la machine de son aspect logiciel. De cette manière, c'est entièrement sur le logiciel, les programmes, que se trouve reportée la charge de toutes les éventuelles propriétés cognitives du système.

Mais le cognitivisme va plus loin. Si pour lui, comme pour la cybernétique, *penser c'est calculer* comme un ordinateur, il ne s'agit pas en revanche de manipuler des symboles dénués de sens, mais de manipulation réglée de *symboles ayant le statut de représentation d'état du monde*. Pour le dire autrement, les symboles envisagés par le cognitivisme ont à la fois une réalité matérielle et une valeur sémantique. Ils représentent certains aspects du monde, et, de ce fait, les calculs opérés par la machine deviennent une simulation qui préserve la structure de la réalité. À toute opération « mentale » correspond ainsi une transformation possible du monde. Le cognitivisme pose de la sorte des contraintes beaucoup plus fortes sur le système de symboles manipulés que la cybernétique. Pour donner un exemple, les nombres manipulés dans les réseaux connexionnistes n'ont pas le statut de symboles pour le cognitivisme, et si un cognitiviste étudie un tel réseau, il le fera en l'abordant à un autre niveau, dans lequel il pourra lui attribuer des connaissances et des règles de raisonnement.

Il s'agit alors d'énoncer comment est constituée une représentation et ce qui peut rendre sa manipulation sémantiquement correcte. Pour cela, le cognitivisme s'est naturellement trouvé influencé par le mouvement de pensée le plus caractéristique et le plus influent du XX^{e} siècle, à savoir celui selon lequel la réalité est organisée comme un langage. À l'instar d'autres disciplines comme la biologie, maintenant entièrement conçue comme élucidation du code génétique, l'ethnologie de Claude Levi-Strauss (1908-2009) inscrite dans le mouvement structuraliste, la psychanalyse cherchant le code de l'inconscient, et même la physique[4], les sciences cognitives ont été chercher du côté de la philosophie analytique – essentiellement une philosophie du langage – une solution à leur problème. Selon cette approche, la pensée procède à partir de propositions portant sur le monde, dotées d'une syntaxe, et manipulées suivant des *règles d'inférence* strictes d'un langage formel, parmi lesquelles figurent au premier plan la déduction, l'abduction, la généralisation, etc., c'est-à-dire des règles d'inférence liées à la logique.

[4] Dont une grande partie de l'activité est tournée vers l'exploration des conséquences de manipulations du langage mathématique (par exemple la théorie des groupes qui permet de prédire, et avec quel succès, l'existence de particules). Si la physique s'enivre d'une telle puissance, elle ne peut que s'interroger sur « le pouvoir prédictif déraisonable des mathématiques » [Wig60].

Copyright © 2010 Eyrolles.

On ne s'attardera pas ici sur les difficultés de nature philosophique rencontrées par cette approche, touchant entre autres au problème de la référence et de l'intentionalité (voir par exemple l'excellent livre de Joëlle Proust [Pro97]). En revanche, il est important de souligner les conséquences de ce point de vue pour l'étude de l'apprentissage.

À partir du moment où la cognition est considérée comme la manipulation, suivant un ensemble de règles strictes, de propositions sur le monde, il devient naturel d'envisager l'ensemble de toutes les propositions possibles sur le monde et de toutes les théories correspondantes. Cet ensemble de « mondes possibles » sera assimilé à un espace d'hypothèses potentielles pouvant expliquer les manifestations observées du monde. Selon cette perspective, l'apprentissage devient la recherche d'une, ou plusieurs, hypothèse(s), s'accordant aux données recueillies jusque-là. Et l'étude de l'apprentissage selon ce point de vue amène alors naturellement à considérer un certain nombre de questions.

Nous invitons ici le lecteur à faire une pause et à s'interroger sur le programme de recherche qu'il envisagerait à ce point. Il s'agit là en effet de tester à quel point un parti pris philosophique sur un sujet détermine ensuite de manière très profonde la nature du questionnement qui va façonner toute la discipline.

Voici, par exemple, un ensemble de questions fondamentales qui façonnent le point de vue cognitiviste :

- Comment évolue l'espace d'hypothèses en fonction des données disponibles sur le monde ? Peut-on montrer qu'il se réduit ? Si oui, à quelle vitesse ? Peut-il y avoir espoir de converger vers une hypothèse unique ? Et si, après le recueil d'un certain nombre d'informations, il reste plusieurs hypothèses, peut-il y avoir espoir qu'elles aient une parenté ou une proximité pour pouvoir, sans trop de risque, en choisir une plutôt qu'une autre ?
- Comment peut-on rendre efficace l'exploration de l'espace des hypothèses en cours d'apprentissage ?
- Quelles sont les règles d'inférence appropriées pour rendre compte de l'apprentissage ? En particulier, par quels opérateurs peut-on modéliser les règles d'inférence telles que l'induction, l'analogie, l'abduction, qui correspondent à des formes de raisonnement permettant d'élaborer une représentation plus opératoire que la simple mémorisation des entrées ?

À ce stade, ces questions sont encore vagues. Nous verrons comment elles peuvent être précisées dans les modèles et les algorithmes de l'apprentissage, comme dans le cadre des théories de l'apprenabilité. Il ne faudra pour autant pas oublier le formidable réductionnisme dont elles sont issues, en particulier la projection opérée ainsi de la cognition sur le plan des représentations et des règles de la logique formelle. Par ailleurs, à trop se mettre au niveau de la proposition et de la règle d'inférence, la scène globale, le fonctionnement des théories et de la connaissance en général finissent par être occultés. C'est sans doute le prix à payer pour faire les premiers pas. Il ne faut cependant pas en devenir dupe et négliger l'objectif qui est la compréhension de l'apprentissage, en particulier dans sa dimension liée à la construction de connaissances, telle que l'envisage par exemple la didactique.

Nous pensons que le reste de l'ouvrage montrera que l'apprentissage artificiel tel qu'il est abordé actuellement est placé à la résultante des influences de la cybernétique et du cognitivisme. Du second, il a pris les concepts d'espace d'hypothèses, de langage de représentation, de règles d'inférence, de recherche dans un espace de possibilités. Cela a rendu possible les premières simulations, les premiers programmes. Mais il a fallu aussi l'apport du premier courant pour que, en relâchant les contraintes sur les symboles manipulés, on ose se servir de toute une panoplie d'outils mathématiques permettant de caractériser l'apprentissage comme un processus de convergence vers une fonction cible. Il reste sans doute maintenant à dépasser ces deux

Copyright © 2010 Eyrolles.

perspectives, mais pour cela il est utile de les connaître et d'en peser l'influence.

La suite de ce chapitre vise à fournir les concepts et notations de base nécessaires à la compréhension du reste de l'ouvrage.

4. Les concepts de base de l'apprentissage

Afin de présenter les principaux concepts permettant d'étudier l'apprentissage, nous allons nous référer dans ce chapitre introductif à un scénario certes limité, mais servant de cadre idéal à la majorité des recherches actuelles sur l'apprentissage artificiel : celui de l'induction supervisée. Nous envisagerons d'autres apprentissages plus tard.

Qu'est-ce que l'*induction*? C'est le processus par lequel on tire des lois de portée générale en partant de l'observation de cas particuliers. C'est ainsi que l'avimateur novice évoqué dans la préface cherche une loi générale lui permettant de distinguer les oies des cygnes. Pour ce faire, il dispose de quelques exemples de volatiles, sur lesquels il effectue des mesures comme la couleur ou la taille. De plus, il est assisté d'un expert qui lui dit à quelle espèce appartient chacun d'eux. Dans ce cas, dans lequel un « oracle » fournit la bonne réponse, on parle d'*induction supervisée*. Nous définissons ceci plus formellement ci-dessous.

4.1 Un scénario de base pour l'induction

Dans ce scénario, nous supposons qu'un *système apprenant* reçoit des données de l'univers dans lequel il est placé. Nous ne nous interrogerons pas ici sur ce qui permet de définir et de segmenter ces « données ». Dans le cas de l'apprentissage supervisé, chacune de ces données prend la forme d'un couple dans lequel on distingue d'une part la description d'une situation ou encore *observation* (par exemple une situation de jeu, ou bien une matrice de niveaux de gris), et d'autre part une réponse, que l'on appelle aussi fréquemment *sortie désirée*, (par exemple *situation de mat en trois coups* ou bien *lettre 'a'*, ou encore *volatile = cygne*) qui est supposée être fournie par un *oracle*. Une donnée $\mathbf{z}_i$ est ainsi définie par un couple (*observation, sortie désirée*) que nous noterons $(\mathbf{x}_i, u_i)$[5]. (Voir la figure 1.5).

4.2 Quelques notions clés

4.2.1 Le critère de succès

Dans le scénario de l'apprentissage supervisé, la tâche de l'apprenant est d'essayer d'approximer au mieux la sortie désirée u_i pour chaque entrée observée $\mathbf{x}_i$. Dans le cas idéal, l'apprenant devient capable, après un certain temps d'apprentissage, de prédire exactement, pour chaque entrée $\mathbf{x}_i$, la sortie désirée u_i. En général cependant, il faudra se contenter d'une approximation de la réponse de l'oracle. Nous formaliserons plus loin cette notion d'approximation grâce à une *fonction de risque*, encore appelée *critère de succès*, qui dépendra à la fois du domaine étudié et de l'objectif de l'apprentissage.

[5] Nous utilisons la lettre u en accord avec les notations de l'automatique et de la théorie du contrôle, pour la distinguer de la « vraie » réponse de l'environnement, notée y_i, pour une forme d'entrée $\mathbf{x}_i$. Par ailleurs, la sortie (désirée, u, ou vraie, y) est rarement un vecteur et c'est pourquoi nous n'utilisons pas de caractère gras pour la dénoter. Il ne faut cependant pas en déduire que la sortie ne peut pas être un vecteur. La plupart des résultats décrits dans cet ouvrage pourraient être étendus, souvent au prix d'une plus grande complexité calculatoire, au cas d'une sortie vectorielle, voire d'un tri.

Copyright © 2010 Eyrolles.

Le critère de succès est ce qui est mesuré dans l'évaluation de la performance. Il s'agit donc d'un *critère relatif à un observateur externe*. Par exemple, la performance sera mesurée en fonction du nombre d'erreurs commises par l'apprenant en cours d'apprentissage, ou en fonction de son taux d'erreur après apprentissage. Dans une tâche qui prend une importance grandissante avec le réseau Internet, celle qui consiste à chercher des documents relatifs à une requête particulière, la performance sera fonction à la fois du nombre de documents pertinents trouvés par le système rapporté au nombre réel de documents pertinents et du nombre de documents pertinents non trouvés. Plus généralement, la mesure de performance peut inclure des facteurs indépendants de l'adéquation aux données d'apprentissage et de natures très diverses. Par exemple, la *simplicité* du résultat d'apprentissage produit par l'apprenant, sa *compréhensibilité*, son *intelligibilité* par un expert, la facilité de son intégration dans une théorie courante, le faible coût computationnel nécessaire à son obtention, etc.

Il faut ici faire une remarque importante. Le critère de succès, mesuré par un observateur externe, n'est pas nécessairement identique à la *fonction de coût* ou *de perte* qui est *interne à l'apprenant* et le guide pour faire converger les paramètres de son modèle d'apprentissage. Par exemple, un algorithme d'apprentissage de réseau connexionniste cherche généralement à minimiser un écart quadratique entre ce qu'il prédit sur chaque exemple d'apprentissage et la sortie désirée. Cette mesure est interne au système, elle lui permet de mesurer la qualité de l'approximation de son modèle courant avec les données d'apprentissage, mais elle n'est pas généralement celle qui intéresse l'observateur externe qui examine par exemple le taux d'erreur ou prend éventuellement en compte d'autres critères comme ceux évoqués plus haut.

4.2.2 Notion de protocole d'apprentissage

L'apprentissage et son évaluation dépendent du *protocole* qui règle les interactions entre l'apprenant et son environnement, incluant l'oracle. Il faut ainsi distinguer l'*apprentissage hors ligne* (*batch learning*), dans lequel toutes les données d'apprentissage sont fournies d'un seul coup à l'apprenant, de l'apprentissage séquentiel, incrémental ou *apprentissage en ligne* (*on-line learning*) dans lequel les données arrivent en séquences et où l'apprenant doit délibérer et fournir une réponse après chaque entrée ou groupe d'entrées.

Le protocole stipule également le type d'entrées fournies à l'apprenant et le type de sorties attendues. Par exemple, un scénario peut spécifier qu'à chaque instant l'apprenant reçoit une observation x_i, qu'il doit alors fournir une réponse y_i, et que seulement alors l'oracle produit la réponse correcte u_i (voir figure 1.4, partie droite). Une illustration d'un tel protocole est fournie par le cas d'un système essayant de prédire le cours du lendemain d'un titre en bourse à partir d'informations sur le contexte économique courant. À chaque fois la réponse correcte est disponible le lendemain seulement. On parle alors naturellement de tâche de *prédiction*. Plus fondamentalement, les tâches dites de prédiction s'intéressent à prévoir correctement une réponse en un point précis : quel sera le cours du soja demain à midi, la consommation électrique dans trois mois, quelle est la pathologie de ce patient particulier, etc.

Les tâches de prédiction sont à contraster avec les *tâches d'identification* dans lesquelles le but est de trouver une explication globale parmi toutes celles possibles, qui une fois connue permettra l'établissement de prédictions quelle que soit la question. Un système d'analyse boursière pourrait ainsi chercher à identifier la fonction suivie par le cours du soja. De même, un fournisseur d'électricité pourrait vouloir connaître l'ensemble de la courbe de la consommation sur une année. Dans le cas de la médecine, une tâche d'identification consisterait à trouver des lois permettant de fournir un diagnostic pour n'importe quel malade et non pour un patient particulier.

Le scénario sera alors différent. Par exemple, il pourra prévoir que le système apprenant doit

Copyright © 2010 Eyrolles.

produire après chaque nouvelle entrée $(\mathbf{x}_i, u_i)$ une hypothèse sur la « fonction cachée » de l'oracle par laquelle celui-ci détermine u_i en fonction de $\mathbf{x}_i$. On conçoit que le critère de succès ne soit pas le même dans le cas d'une tâche de prédiction que dans celui d'une tâche d'identification. Dans ce dernier cas, en effet, on demande beaucoup plus à l'apprenant puisqu'on attend de lui une hypothèse explicite, donc une sorte d'explication de ses prédictions (voir figure 1.4, partie gauche).

Par ailleurs, l'apprenant peut être plus ou moins *actif*. Dans les protocoles décrits jusqu'ici, l'apprenant reçoit passivement les données sans avoir d'influence sur leur sélection. Il est possible d'envisager des scénarios dans lesquels l'apprenant a une certaine initiative dans la recherche d'informations. Dans certains cas, cette initiative est limitée, par exemple lorsque l'apprenant, sans avoir la totale maîtrise du choix de l'échantillon d'apprentissage, est simplement capable d'orienter sa distribution de probabilité. Les méthodes de *boosting*, décrites dans le chapitre 17, en sont une illustration. Dans d'autres cas, l'apprenant peut poser des questions sur la classe d'appartenance d'une observation, on parle alors d'*apprentissage par requête d'appartenance* (*membership queries*), ou même organiser des expériences sur le monde, et on parle alors d'*apprentissage actif*. Le jeu de MasterMind, qui consiste à deviner une configuration de pions de couleurs cachés en posant des questions suivant certaines règles, est un exemple simple d'apprentissage actif dans lequel l'apprenant possède l'initiative des questions.

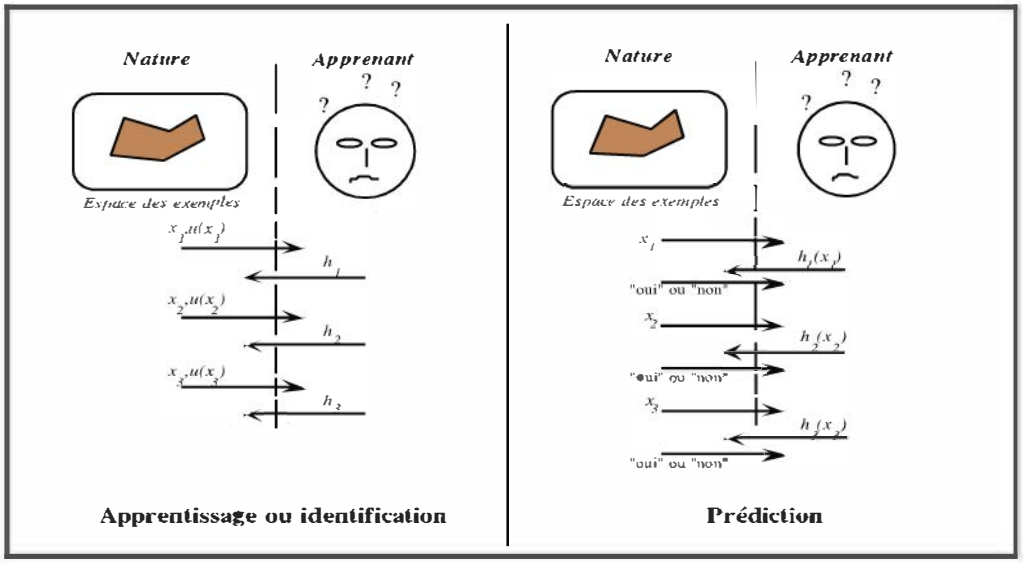


FIG. 1.4: *Différence entre un protocole d'*identification *(à gauche) et un protocole de* prédiction *(à droite). Dans le protocole d'identification, l'objectif de l'apprenant est de trouver une hypothèse h qui rende bien compte des données d'apprentissage. Dans le protocole de prédiction, l'apprenant doit « seulement » chercher à trouver la bonne réponse à une question particulière. Ici, on suppose qu'après chaque prédiction, l'apprenant reçoit une confirmation ou une infirmation de sa prédiction, ce qui correspond à un apprentissage en-ligne, par contraste avec un apprentissage « batch ».*

4.2.3 Notion de tâche d'apprentissage

Il est possible d'aborder l'objectif du processus d'apprentissage suivant plusieurs points de vue.

Copyright © 2010 Eyrolles.

Par rapport à la connaissance

Le but de l'apprentissage peut être de *modifier le contenu* de la connaissance[6]. On parle alors d'*acquisition de connaissances*, de *révision*, et, pourquoi pas, d'oubli. En parlant de manière informelle, l'apprenant sait désormais plus ou moins de choses. Mais cela ne préjuge pas de sa capacité à utiliser ses connaissances.

Le but de l'apprentissage peut aussi être, sans nécessairement modifier le « contenu » de la connaissance, de *le rendre plus efficace* par rapport à un certain but, par réorganisation, optimisation ou compilation par exemple. Ce pourrait être le cas d'un joueur d'échecs ou d'un calculateur mental qui apprend à aller de plus en plus vite sans pour autant connaître de nouvelles règles de jeu ou de calcul. On parle dans ce cas d'*optimisation de performance* (*speed-up learning*).

Par rapport à l'environnement

La tâche de l'apprentissage peut aussi être définie par rapport à ce que l'agent apprenant doit réaliser pour « survivre » dans son environnement. Cela peut inclure :

- *Apprendre à reconnaître* des formes (par exemple : des caractères manuscrits, des oiseaux, des prédateurs, une tendance haussière à la bourse, une appendicite, etc.). Lorsque l'apprentissage s'opère avec un professeur, ou oracle, qui fournit les réponses désirées, on parle d'*apprentissage supervisé*. Sinon, on parle d'*apprentissage non supervisé*. Dans ce cas, la tâche d'apprentissage consiste à la fois à découvrir des catégories et à trouver des règles de catégorisation.
- *Apprendre à prédire*. Il y a alors une notion de dépendance temporelle ou de causalité.
- *Apprendre à être plus efficace*. C'est le cas notamment des situations de résolution de problème, ou de recherche de plans d'action dans le monde.

Par rapport à des classes abstraites de problèmes

Indépendamment même d'un algorithme d'apprentissage, il est possible de caractériser l'apprentissage par une classe générale et abstraite de problèmes et de processus de résolution qui leur sont liés. C'est ainsi qu'un certain nombre de disciplines, en particulier issues des mathématiques ou de la théorie de l'information, se sont découvertes un intérêt pour les problèmes d'apprentissage.

- *Les théories de compression d'information*. En un certain sens, l'apprentissage peut être abordé comme un problème d'extraction et de compression d'information. Il s'agit d'extraire l'information essentielle ou le message initial d'un émetteur idéal, débarassé de toutes ses redondances. Ainsi, les sciences d'observation, telles l'astronomie ou l'ornithologie, procèdent par élimination des détails superflus ou redondants et par la mise en évidence de régularités cachées.
- *La cryptographie*. Dans une perspective analogue, proche des préoccupations de la théorie de l'information, l'apprentissage peut être considéré comme une *tâche de décodage* ou même de décryptage d'un message codé par l'émetteur idéal et intercepté en tout ou partie par l'agent. Après tout, c'est parfois ainsi qu'est vu le scientifique étudiant la nature. Il est alors logique de se demander sous quelles conditions un message peut être « cassé », c'est-à-dire sous quelles conditions un apprentissage est possible.

[6] Qui peut-être mesuré par exemple par sa clôture déductive, c'est-à-dire, dans une représentation logique, tout ce qui peut être déduit correctement à partir de la base de connaissances courante.

Copyright © 2010 Eyrolles.

- *L'analyse.* L'apprentissage peut également être examiné comme un *problème d'approximation.* C'est particulièrement clair lorsque l'on adopte le point de vue de la figure 1.5. La tâche de l'apprenant est bien alors de trouver une approximation aussi bonne que possible d'une fonction cachée connue uniquement par l'intermédiaire d'un échantillon de données. Le problème de l'apprentissage devient alors souvent celui de l'étude des conditions d'approximation et de convergence. Nous aurons largement l'occasion de développer ce point de vue, actuellement dominant, dans les chapitres 2 et 21.
- *L'induction.* Dans les années soixante-dix et au début des années quatre-vingts, sous l'influence du point de vue cognitiviste, une large communauté de chercheurs, particulièrement active en France, s'est penchée sur l'apprentissage en tant que *problème de généralisation.* Cette approche part de deux présupposés essentiels. D'une part, l'agent cognitif apprenant doit apprendre quelque chose qu'un autre agent cognitif équivalent connaît. Il est donc normalement capable d'atteindre parfaitement la connaissance cible. D'autre part, les connaissances et les données peuvent être décrites par un langage. On cherche alors les opérateurs dans ce langage qui peuvent correspondre à des opérations de généralisation ou de spécialisation utiles pour l'induction, et on construit des algorithmes les utilisant, permettant de résumer les données tout en évitant de les surgénéraliser et d'en tirer des conséquences illégitimes.
- *Les mathématiques appliquées.* Finalement, l'ingénieur peut être tenté de voir dans l'apprentissage un cas particulier de *résolution de problème inverse.* Dans le cas d'un problème direct, on se donne une « structure » et on en cherche les conséquences. Par exemple, tel avion est capable de supporter telle charge dans telles conditions. Dans le cas d'un problème inverse, on se donne des spécifications sur les capacités souhaitées et on cherche à concevoir un objet qui les vérifie. C'est évidemment typiquement le problème auquel sont confrontés les ingénieurs. Prenons trois exemples :
  - On peut dire que la théorie des probabilités est une théorie s'attachant à un problème direct (étant donné un modèle paramétré, quelles sont les probabilités associées à tel événement ?), tandis que la théorie des statistiques s'attaque à un problème inverse (étant donné un échantillon de données, quel modèle permet de l'expliquer, c'est-à-dire peut l'avoir produit ?).
  - Étant donnés deux nombres, il est facile d'en trouver le produit (problème direct), il est en revanche généralement impossible de trouver à partir d'un nombre ceux dont il est le produit (problème inverse).
  - Étant donné un échantillon d'apprentissage $\mathcal{S} = \langle(\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), \ldots, (\mathbf{x}_m, u_m)\rangle$[7], identifier la fonction qui permet de calculer l'étiquette u de n'importe quelle forme d'entrée $\mathbf{x}$. Il s'agit encore une fois d'un problème inverse.

  Les problèmes inverses sont ainsi souvent des problèmes que l'on dits *mal posés*, c'est-à-dire n'ayant pas de solution unique. Selon cette perspective, l'étude de l'apprentissage peut être vue comme celle des conditions permettant de résoudre un problème mal posé, c'est-à-dire des contraintes qu'il faudra ajouter pour que la procédure de résolution puisse trouver une solution particulière[8].

[7] Nous utilisons la notation $\langle\ldots\rangle$ pour l'échantillon d'apprentissage qui peut être un multi-ensemble (voir la section 5 et le chapitre 2 pour plus de détails).

[8] Déjà en 1902, Hadamard (1865-1963) affirmait qu'un problème *bien posé* devrait avoir la propriété qu'*il existe une solution unique, qui, en plus, devrait dépendre des données de manière continue.*

Copyright © 2010 Eyrolles.

Par rapport aux structures de données ou types d'hypothèses visées

Il arrive fréquemment que l'on impose le type de structure de données (ou de *langage d'expression d'hypothèses*) qui doit être recherché par le système apprenant. Cela permet de guider à la fois la détermination de l'algorithme d'apprentissage à utiliser, mais aussi les données qui seront nécessaires pour que l'apprentissage soit possible[9]. Sans chercher à être exhaustif, le chapitre 3 y reviendra plus longuement, les représentations les plus étudiées incluent :

- Les *expressions booléennes*, qui sont souvent appropriées pour apprendre des concepts définis sur un *langage attribut-valeurs* (par exemple des règles de système expert).
- Les *grammaires et les processus markoviens*, utilisés pour représenter des séquences d'événements.
- Les *fonctions* de décision, linéaires ou non linéaires, permettant soit de discriminer des formes appartenant à un sous-espace ou à son complémentaire (tâches de classification), soit d'identifier un processus sous-jacent (tâches de régression).
- Les *arbres de décision* qui autorisent l'expression de classifications par des hiérarchies de questions. Les arbres de décisions ont l'avantage d'être souvent à la fois concis et interprétables.
- Les *programmes logiques* auxquels il faut songer lorsque l'on cherche à apprendre des concepts relationnels.
- Les *réseaux bayésiens* permettant à la fois de représenter des univers structurés par des relations de causalité et de prendre en compte et d'exprimer des mesures de certitude ou de confiance.

Parfois l'apprentissage peut consister à *changer de structure de données* pour en trouver une équivalente mais plus efficace du point de vue calculatoire. C'est encore une fois, sous un autre angle, le problème de l'optimisation de performance.

4.3 L'induction considérée comme estimation de fonction

Après avoir brièvement passé en revue les facteurs et les points de vue en jeu dans l'apprentissage artificiel, nous esquissons dans cette section la manière dont est actuellement envisagé le processus d'apprentissage. Nous allons considérer la tâche de l'apprenant, ainsi que l'approche suivie pour la mener à bien. Il s'agit ici d'un premier exposé qui sera précisé, de manière plus formelle dans le chapitre 2 et dans son complément, le chapitre 21, et qui permettra de comprendre le fonctionnement des algorithmes d'apprentissage décrits dans les chapitres suivants.

Revenons sur le scénario esquissé dans la section 4.1 et dans la figure 1.5. Nous supposons que l'environnement, qu'il soit mesuré par les senseurs d'un robot ou qu'il s'exprime sous la forme d'une base de données, fournit une collection de formes $\mathbf{x}_i$ définies sur l'espace des entrées $\mathcal{X}$ et tirées aléatoirement chacune suivant une distribution de probabilités notée $\mathsf{P}_{\mathcal{X}}$ (on parle de tirage indépendant et identiquement distribué ou *tirage i.i.d.*). On peut ainsi imaginer qu'une webcam prenne des images à intervalles réguliers d'un carrefour à New-York, et que les formes $\mathbf{x}_i$ mesurées correspondent aux véhicules observés. On pourra supposer que ces véhicules sont indépendants les uns des autres (sauf dans le cas de cortèges officiels ou mortuaires), mais que leur distribution dépendra de la ville, New-York se révélant sans doute différent de Londres ou Nouakchott[10] sous cet aspect.

[9] Cette manière d'organiser les méthodes d'apprentissage artificiel a été utilisée par Pat Langley (1953-...) dans son livre [Lan96] de 1996.

[10] Capitale de la République islamique de Mauritanie.

Copyright © 2010 Eyrolles.

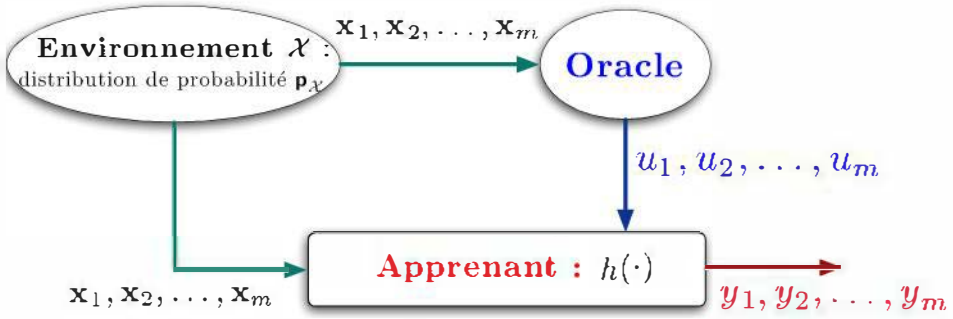


FIG. 1.5: *Le scénario classique de l'apprentissage par induction. L'environnement fournit des données* $\mathbf{x}_i$ *tirées aléatoirement suivant une distribution* $\mathsf{P}_{\mathcal{X}}$ *sur l'espace d'entrée* $\mathcal{X}$. *Ces données sont étiquetées par un oracle qui utilise pour cela une fonction* $f \in \mathcal{F}$. *L'apprenant reçoit un échantillon d'exemples ou couples* $(\mathbf{x}_i, u_i) = (\mathbf{x}_i, f(\mathbf{x}_i))$, *et à partir de cet échantillon, doit chercher à deviner* f, *ou au moins à en trouver une approximation* h.

Dans le cadre de l'apprentissage supervisé, nous supposons également qu'un oracle étiquette les formes $\mathbf{x}_i$ grâce à une fonction inconnue de l'apprenant, que nous appellerons *fonction cible*, notée f. L'apprenant reçoit donc un ensemble d'exemples (voir figure 1.5).

$$\mathcal{S} = \langle (\mathbf{x}_i, u_i) \rangle_{1 \leq i \leq m} = \langle (\mathbf{x}_i, f(\mathbf{x}_i)) \rangle_{1 \leq i \leq m}$$

On supposera que l'oracle choisit la fonction cible f dans une famille de fonctions notée $\mathcal{F}$, également inconnue. On supposera également ici, qu'une fois sélectionnée par l'oracle, la fonction f reste constante, de même que la distribution $\mathsf{P}_{\mathcal{X}}$ des formes $\mathbf{x}_i$. Nous étudierons plus tard le cas d'une fonction cible bruitée ou présentant une dérive au cours du temps, ainsi que le cas d'une distribution variable des exemples.

Cette description de l'apprentissage, plus précisément de l'induction, conduit naturellement à voir l'apprentissage comme une tâche d'estimation de fonction à partir d'un échantillon de son comportement. Il s'agit là effectivement du point de vue dominant actuellement.

Notons que le cadre de l'estimation de fonction cible est très général puisqu'il couvre de nombreuses tâches d'apprentissage classiques. Par exemple :

- Un *problème de régression* peut être considéré comme un problème d'estimation dans lequel il s'agit de trouver une fonction h telle que :

$$\forall \mathbf{x} \in \mathcal{X}, h(\mathbf{x}) \approx f(\mathbf{x}) = y$$

- L'*apprentissage d'une classification* d'objets (par exemple apprendre à classer des images de pièces de mobilier en types de meubles, des volatiles caractérisés par un certain nombre d'attributs en types d'oiseaux, etc.) peut-être vu comme l'estimation d'une fonction à valeur discrète, où à chaque entrée correspond une valeur associée à une classe. L'apprentissage de la reconnaissance des lettres de l'alphabet peut ainsi être abordé comme l'estimation d'une fonction définie sur un espace d'entrée (par exemple une matrice de pixels) vers un espace de sortie à 26 valeurs.

- L'*apprentissage de concept*, dans lequel il s'agit d'apprendre à reconnaître *une* classe d'objets parmi tous les autres objets, peut être considéré comme l'estimation d'une fonction binaire (on dit aussi fonction indicatrice), prenant la valeur 1 quand la forme d'entrée est de la classe cible et 0 autrement.

Copyright © 2010 Eyrolles.

- Un *problème d'optimisation multicritères* dans lequel on cherche à optimiser à la fois plusieurs critères objectifs peut être perçu comme l'estimation d'une fonction multi-valuée.

Toujours pour simplifier, nous supposerons que l'apprenant cherche une approximation de la fonction cible à l'intérieur d'une famille $\mathcal{H}$ de *fonctions hypothèses* h. C'est le cas par exemple d'un apprenant utilisant un réseau de neurones dont l'architecture contraint le type de fonctions réalisables à un certain ensemble de fonctions. De même, le désormais fameux avimateur novice ne cherche que des séparations linéaires dans l'espace des descriptions des oies et des cygnes.

Pour définir le problème d'apprentissage, il faut maintenant se donner un critère de performance. On évaluera la qualité de l'estimation h relativement à une espérance de performance dans un environnement donné. C'est pourquoi on spécifie généralement *le critère de performance* d'une fonction hypothèse h sous la forme d'une expression exprimant ce que coûtera le choix de la fonction hypothèse h si la vraie fonction inconnue est f. Par exemple, la performance d'un système d'apprentissage de diagnostic médical sera mesurée par l'espérance de coût de la décision $h(\mathbf{x})$ lorsque la vraie pathologie est $f(\mathbf{x})$. C'est ce que dénote l'équation suivante :

$$R(h) = \int_{\mathbf{x} \in \mathcal{X}} l(h(\mathbf{x}), f(\mathbf{x}))\, \mathsf{P}_{\mathcal{X}}\, d\mathbf{x} \tag{1.1}$$

dans laquelle $R(h)$ dénote une *fonction de risque*, tandis que l désigne une *fonction de perte* définie pour chaque exemple. L'intégrale est prise sur l'ensemble des formes $\mathbf{x} \in \mathcal{X}$ possibles suivant la distribution donnée $\mathsf{P}_{\mathcal{X}}$.

Par exemple, si l'apprenant se trouve à New-York, la distribution des voitures de couleur jaune est différente de celle observée à Londres. En supposant que le problème soit d'apprendre à reconnaître des taxis, il faut prendre en compte la distribution des véhicules dans l'environnement d'apprentissage. On suppose naturellement que cette distribution $\mathsf{P}_{\mathcal{X}}$ des formes est aussi celle qui sera rencontrée après l'apprentissage. C'est pourquoi cette distribution apparaît dans l'expression du risque. (Apprendre à reconnaître des taxis dans New-York peut se révéler d'une utilité limitée si l'on doit ensuite se débrouiller à Londres, ou, plus encore, en Mauritanie). La tâche d'induction sera bien plus compliquée si c'est un échantillon de données non représentatif de l'environnement qui sera rencontré ensuite par l'apprenant qui lui est fourni en apprentissage. Il faudra en effet lui demander d'en tirer une information qui le rende performant dans un nouvel environnement inconnu pour lui.

La fonction de risque (1.1) mesure donc l'espérance de perte *dans un environnement donné*, spécifié par la distribution $\mathsf{P}_{\mathcal{X}}$ des événements mesurables par l'apprenant.

Formellement, l'apprenant cherche donc à trouver l'hypothèse $h^\star \in \mathcal{H}$ minimisant l'espérance de coût, c'est-à-dire :

$$h^\star = \underset{h \in \mathcal{H}}{\mathrm{ArgMin}}\, R(h) = \underset{h \in \mathcal{H}}{\mathrm{ArgMin}} \int_{\mathbf{x} \in \mathcal{X}} l(h(\mathbf{x}), f(\mathbf{x}))\, \mathsf{P}_{\mathcal{X}}\, d\mathbf{x} \tag{1.2}$$

Cependant, comme la fonction cible f et la distribution des formes $\mathsf{P}_{\mathcal{X}}$ sont inconnues, l'apprenant ne peut directement chercher $h^\star$ et doit s'appuyer sur l'information à sa disposition, à savoir une connaissance *a priori* sous la forme d'un espace d'hypothèses $\mathcal{H}$ à considérer, et un échantillon $\mathcal{S} = \langle (\mathbf{x}_i, u_i) \rangle_{1 \leq i \leq m}$ de données supposées représentatives du monde, pour trouver une bonne hypothèse $\hat{h}$ à défaut de $h^\star$.

L'hypothèse $\hat{h}$ optimise un *critère inductif*, c'est-à-dire un critère défini sur l'échantillon d'apprentissage $\mathcal{S}$ et tenant lieu de la fonction objectif ou risque (voir équation 1.1) inconnu. Ce

Copyright © 2010 Eyrolles.

critère inductif, défini de $\mathcal{H} \times \mathcal{S} \rightarrow \mathbb{R}$ permet d'évaluer chaque hypothèse $h \in \mathcal{H}$. Par exemple, il peut prendre la forme d'un *risque empirique* :

$$R_{Emp}(h) = \frac{1}{m} \sum_{i=1}^{m} l(h(\mathbf{x}_i), y_i)) \tag{1.3}$$

Nous reviendrons plus formellement sur le problème de l'induction dans les chapitre 2 et 21. Pour le moment, ce cadre simplifié suffit à introduire les concepts et questions fondamentaux.

Nous avons ainsi défini la tâche d'apprentissage comme celui d'un problème d'estimation de fonction à partir de l'observation d'un échantillon de données. Nous nous tournons maintenant vers les principes permettant de réaliser cette estimation.

5. L'induction comme un jeu entre espaces

Dans le but de simplifier toute la discussion qui suit et de permettre une visualisation aisée des problèmes, nous nous focalisons dans cette section sur l'apprentissage supervisé de *concept*, c'est-à-dire sur l'apprentissage de *fonctions indicatrices* ou encore à valeur dans $\{0, 1\}$. Les notions abordées seront cependant d'une portée beaucoup plus générale et valables pour l'essentiel dans toutes les situations d'apprentissage.

L'*apprentissage supervisé de concept* consiste à chercher une fonction $f : \mathcal{X} \rightarrow \{0, 1\}$, c'est-à-dire un étiquetage de chaque forme $\mathbf{x} \in \mathcal{X}$ par 0 ($\mathbf{x}$ n'appartient pas au concept visé) ou 1 ($\mathbf{x}$ appartient au concept)[11]. Cette fonction est apprise à partir d'un échantillon de points étiquetés que l'on appelle *échantillon d'apprentissage*. Nous noterons[12]

$$\mathcal{S} = \langle (\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), ..., (\mathbf{x}_m, u_m) \rangle$$

un échantillon d'apprentissage de m points non nécessairement tous distincts (lorsqu'il sera important de préciser la taille de l'échantillon d'apprentissage, nous le noterons $\mathcal{S}_m$). Pour des raisons évidentes, on appelle souvent *exemples* ou *exemples positifs* les points étiquetés par 1 ou par '+', et *contre-exemples* ou *exemples négatifs* les points étiquetés par 0 ou par '−'. Il arrivera cependant dans la suite de l'ouvrage que nous parlions d'exemples pour dénoter les points étiquetés, qu'ils le soient positivement (exemples au sens propre) ou négativement (contre-exemples). La figure 1.6 schématise la tâche d'apprentissage de concepts.

Nous supposons maintenant, d'une part que l'échantillon d'apprentissage n'est pas bruité, c'est-à-dire que les exemples sont correctement décrits et étiquetés, d'autre part qu'il n'est pas incohérent, au sens où la même forme n'est pas à la fois exemple et contre-exemple.

Dans ce cadre, l'échantillon d'apprentissage $\mathcal{S} = \langle (\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), ..., (\mathbf{x}_m, u_m) \rangle$ fournit une information *cohérente* ou encore *consistante* (un anglicisme qui s'est introduit dans le jargon de l'apprentissage artificiel mais que, pour notre part, nous éviterons) à l'apprenant dans la mesure où la partie de $\mathcal{X}$ qu'il cherche doit couvrir tous les exemples positifs de l'échantillon (ce que

[11] Ces deux classes sont aussi notées $\{+, -\}$.

[12] Il arrivera également que nous notions $\mathcal{S} = \{(\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), ..., (\mathbf{x}_m, u_m)\}$ l'échantillon d'apprentissage quand la répétition des exemples n'est pas prise en compte par l'algorithme (ce qui est le cas par exemple de l'algorithme de l'espace des versions (chapitre 4)). Nous verrons également des cas dans lesquels les exemples sont associés à un poids non entier (cas du boosting par exemple, au chapitre 17).

Copyright © 2010 Eyrolles.

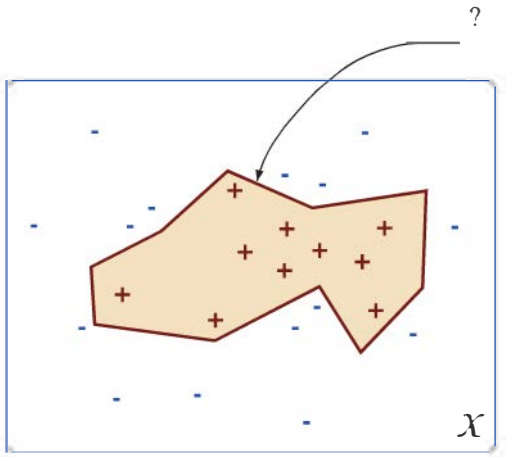


FIG. 1.6: *À partir d'un échantillon de points étiquetés, ici figurés par des points '+' et des points '−', l'apprenant cherche une partition de $\mathcal{X}$ permettant de discriminer les formes* $\mathbf{x}$ *appartenant au concept de celles n'y appartenant pas.*

l'on appelle la propriété de *complétude*) et ne couvrir aucun des exemples négatifs (ce que l'on appelle la propriété de *correction*).

Dans ce cadre restreint, on peut maintenant poser deux questions :

- Quelle information est fournie par chaque exemple ?
- Comment, sur la base de l'échantillon d'apprentissage, choisir une hypothèse, c'est-à-dire, dans le cas de l'estimation d'une fonction indicatrice, une partition de $\mathcal{X}$?

5.1 L'apprentissage est impossible...

Dans le cadre de l'induction de concept, donc d'une fonction indicatrice définie sur l'espace $\mathcal{X}$ des entrées, l'*apprentissage revient à chercher une partition de l'espace* $\mathcal{X}$. En effet, il s'agit d'identifier les régions de $\mathcal{X}$, donc les formes $\mathbf{x}$, correspondant au concept visé (voir figure 1.6).

Que peut nous apprendre un échantillon d'exemples $\mathcal{S}$ sur cette partition ?

Supposons que l'apprenant soit prêt à considérer toutes les partitions possibles de $\mathcal{X}$, donc que n'importe quel étiquetage des formes $\mathbf{x} \in \mathcal{X}$ soit possible *a priori*. Cela signifie que si le cardinal de $\mathcal{X}$, noté $|\mathcal{X}|$, est fini, il existe $2^{|\mathcal{X}|}$ partitions possibles de $\mathcal{X}$.

Supposons alors que nous cherchions à déterminer la classe d'un point $\mathbf{x} \in \mathcal{X}$ inconnu connaissant la classe de tous les points d'apprentissage $\mathbf{x}_i \in \mathcal{X}$. Comment procéder?

Puisque nous manipulons des partitions de $\mathcal{X}$, nous pourrions considérer toutes les partitions cohérentes avec l'échantillon d'apprentissage, puis décider alors de la classe de $\mathbf{x}$ en fonction de ces partitions. Si toutes les partitions cohérentes avec l'échantillon $\mathcal{S}$ prescrivent que $\mathbf{x}$ appartient au concept, ou au contraire n'y appartient pas, cela déterminera notre décision pour la classe de $\mathbf{x}$. Supposons même que toutes ces partitions ne soient pas d'accord sur la classe de $\mathbf{x}$, nous pourrions encore décider que la classe de $\mathbf{x}$ est la classe majoritaire parmi les prédictions de toutes les partitions cohérentes avec l'échantillon d'apprentissage.

Malheureusement, aucun de ces deux cas de figure ne se présente. Il se trouve que si l'on prend toutes les partitions cohérentes avec n'importe quel ensemble de points d'apprentissage $\mathcal{S}$ (c'est-à-dire prédisant correctement l'étiquette de chacun de ces points), et si l'on prend n'importe quel point $\mathbf{x} \notin \mathcal{S}$, alors il existe autant de partitions prédisant l'étiquette 1 pour $\mathbf{x}$ que de partitions prédisant l'étiquette 0. L'échantillon d'apprentissage à lui tout seul ne fournit donc pas une base

Copyright © 2010 Eyrolles.

suffisante pour décider de la classe d'un point nouveau. L'induction, c'est-à-dire l'extrapolation du connu à l'inconnu est impossible. Seul un apprentissage par cœur est réalisable.

Les deux questions soulignées dans la section précédente ont donc reçu une réponse qui jette pour le moins une ombre sur la possibilité de l'induction. Chaque exemple ne fournit aucune information sur une forme inconnue. Toutes les partitions de l'espace $\mathcal{X}$ cohérentes avec l'échantillon sont également probables et leurs prédictions s'annulent en chaque point inconnu. L'aventure de l'apprentissage artificiel tournerait-elle court ?

—— EXEMPLE **Apprentissage de fonction booléenne (1)** ——

Soit un ensemble $\mathcal{X}$ de points décrits par n attributs binaires. Chaque partition de $\mathcal{X}$ correspond à un étiquetage particulier des 2^n points de $\mathcal{X}$. Il existe donc 2^{2^n} partitions différentes de $\mathcal{X}$ ou encore 2^{2^n} fonctions indicatrices définies de $\mathcal{X}$ sur $\{0,1\}$.

Supposons que l'échantillon d'apprentissage comporte m exemples distincts. Le nombre de partitions de $\mathcal{X}$ compatibles avec ces m exemples est : $2^{2^n - m}$ puisque m points sur les 2^n sont fixés.

Prenons le cas de $n = 10$ attributs binaires et de $m = 512$ exemples d'apprentissage. Le cardinal de $\mathcal{X}$ est $|\mathcal{X}| = 2^{10}$, soit 1024 points différents, ce qui n'est pas un espace très grand. Il existe 2^{1024} manières différentes de les étiqueter par 1 ou 0. Après l'observation de la moitié de ces 1024 points, il reste $2^{1024-512}$ partitions possibles, soit 2^{512}. On voit que ces 512 exemples laissent un ensemble considérable de partitions possibles.

x_1	x_2	x_3	$f(\mathbf{x})$
0	0	0	+
0	0	1	−
0	1	0	+
0	1	1	?
1	0	0	+
1	0	1	?
1	1	0	?
1	1	1	−

FIG. 1.7: *Soit f une fonction binaire définie sur un espace d'entrée à trois attributs. La table fournit un échantillon de 5 exemples de cette fonction.*

Étudions un problème plus simple dans lequel les exemples sont décrits par trois attributs binaires. Cela fait $2^3 = 8$ formes possibles. Supposons que cinq exemples parmi ces huit aient été étiquetés par l'oracle, comme le montre la table 1.7. Pour fixer complètement une fonction, il faut déterminer la valeur des trois dernières formes. Il faut donc faire un choix entre $2^3 = 8$ fonctions. Supposons que nous voulions déterminer la valeur associée à l'entrée (0 1 1). Il y a quatre fonctions parmi les huit qui sont associées à la sortie + et quatre associées à la sortie -. Il est donc impossible d'avoir même seulement une préférence pour une prédiction plutôt qu'une autre concernant l'étiquette de ce point.

Nous nous sommes placés dans le cas où l'apprenant cherche directement une partition de l'espace d'entrée $\mathcal{X}$, c'est-à-dire qu'il cherche à déterminer l'étiquette de chaque forme $\mathbf{x} \in \mathcal{X}$. C'est évidemment impossible, sauf dans le cas d'espaces $\mathcal{X}$ très restreints pour lesquels un apprentissage par cœur est envisageable. En d'autres termes, il est généralement impossible d'apprendre une partition de $\mathcal{X}$ *en extension*, c'est-à-dire en énumérant toutes les formes et leur étiquette associée.

Copyright © 2010 Eyrolles.

5.2 ...sans limiter l'espace des hypothèses

C'est pourquoi on utilise généralement pour décrire des partitions de $\mathcal{X}$ un *langage de description des hypothèses*, que nous noterons $\mathcal{L}_{\mathcal{H}}$. Celui-ci permet de définir un espace d'expressions ou d'hypothèses $\mathcal{H}$, par exemple l'espace des hypothèses décrites par une conjonction de conditions sur les descripteurs[13].

Ainsi, dans l'exemple précédent, on pourrait décrire des fonctions binaires du type $(x_1 = 0) \wedge (x_2 = 1) \wedge (x_3 = 1)$. En revanche, ce langage interdit de considérer une fonction telle que $(x_1 = 0 \wedge x_2 = 1 \wedge x_3 = 1) \vee (x_1 = 0 \vee x_2 = 0 \wedge x_3 = 0)$.

La figure 1.8 donne des exemples de la restriction de l'espace des hypothèses par un langage de description.

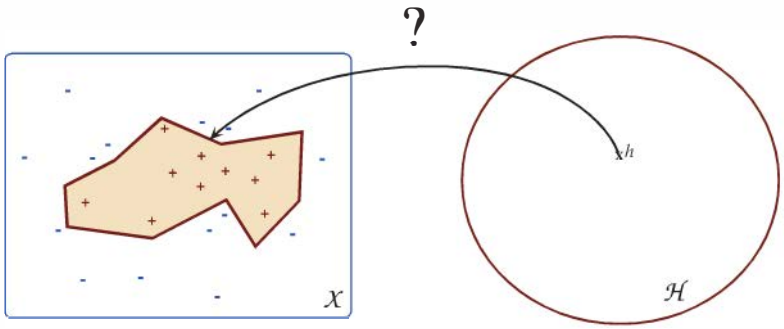


FIG. 1.8: *Introduction d'un espace d'hypothèses $\mathcal{H}$. Chaque point de $\mathcal{H}$, ou encore hypothèse, correspond à une partition de l'espace des entrées $\mathcal{X}$.*

Lorsqu'un espace d'hypothèses $\mathcal{H}$ est disponible, la recherche d'une partition de $\mathcal{X}$ s'effectue par l'intermédiaire de $\mathcal{H}$. Il s'agit de chercher *dans* $\mathcal{H}$, une hypothèse h correspondant à une partition de $\mathcal{X}$ appropriée.

Les avantages de l'utilisation explicite d'un espace d'hypothèses sont multiples :

1. D'abord, grâce au langage $\mathcal{L}_{\mathcal{H}}$, l'apprenant manipule des partitions de $\mathcal{X}$ *en intension* et non plus *en extension*. Il travaille sur des expressions du langage $\mathcal{L}_{\mathcal{H}}$ et non pas sur des ensembles définis par l'énumération de leurs éléments.
2. Ensuite, et c'est un point capital d'après la discussion de la section précédente, il devient possible d'effectuer une *induction* à partir d'un échantillon limité d'exemples. Il suffit pour cela que $\mathcal{L}_{\mathcal{H}}$ *ne permette pas* de décrire toutes les partitions de $\mathcal{X}$.

   Voyons pourquoi.

   Nous allons d'abord le montrer en reprenant l'exemple précédent.

   —— EXEMPLE **Apprentissage de fonction booléenne (2)** ——

   Supposons que pour une raison quelconque, l'apprenant qui reçoit des entrées décrites sur les trois descripteurs binaires x_1, x_2, x_3 ne puisse prendre en compte en fait que le premier et le troisième descripteurs, c'est-à-dire x_1 et x_3, pour décider de l'étiquette de la forme reçue. Cela revient à dire que le nombre de fonctions que l'apprenant peut considérer est de 4 (2^2) au lieu des 8 (2^3) possibles lorsque l'on prend en compte les trois descripteurs.

   Cela signifie en particulier que si l'échantillon d'apprentissage contient les exemples (000) → − et (010) → +, l'apprenant ne pourra pas construire une hypothèse, c'est-à-dire une fonction, qui permette d'en rendre compte.

[13] Nous verrons au chapitre 3 qu'il s'agit du langage CNF (Conjunctive Normal Form).

Copyright © 2010 Eyrolles.

En revanche, cette fois-ci l'échantillon d'apprentissage fourni dans la table donnée précédemment lui permet de faire une prédiction pour le point (0 1 1). Ceci parce que la seule fonction à valeur sur x_1, x_3 et cohérente avec les exemples d'apprentissage est la fonction dont le tableau est le suivant :

x_1	x_3	$f(\mathbf{x})$
0	0	+
0	1	−
1	0	+
1	1	−

Et selon cette fonction, l'étiquette de la forme (0 1 1) est '−'.

Nous voyons donc qu'une limitation de l'espace d'hypothèses rend possible l'induction. Naturellement, ce pouvoir a un prix. Si les « œillères » dont on a muni l'apprenant ne correspondent pas avec la fonction cible de la nature ou de l'oracle, on ne peut pas l'apprendre correctement.

La figure 1.9 est également une illustration de cette même idée.

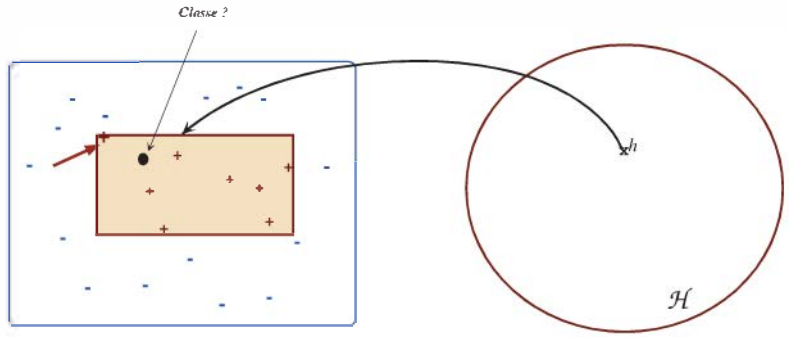


FIG. 1.9: *Supposons que le langage de représentation des hypothèses $\mathcal{L}_\mathcal{H}$ corresponde à une restriction aux parties de $\mathcal{X}$ qui sont des rectangles. Dans ce cas, la donnée du point '+' indiqué par la flèche implique que tous les points inscrits dans le rectangle dont il délimite un angle sont de classe '+'. On voit que dès lors, il devient possible d'induire la classe de points jamais observés dans l'échantillon d'apprentissage. Par exemple, selon ce biais, le point dénoté par un rond noir est prédit appartenir à la classe '+'.*

Pour qualifier ces « œillères » qui limitent l'espace des fonctions hypothèses que peut considérer l'apprenant, on parle d'un *biais de représentation*. Évidemment, tout biais de représentation correspond à un « acte de foi » sur le type d'hypothèses adéquat pour décrire le monde. Cet acte de foi peut être erroné auquel cas l'apprentissage peut donner de très mauvais résultats (voir figure 1.10). Il faudra parvenir à détecter cette situation.

Nous verrons plus loin que la notion de *biais* en apprentissage se définit comme toute restriction de l'ensemble des hypothèses potentielles, y compris des restrictions qui vont plus loin que les restrictions portant sur le langage d'expression des hypothèses.

3. Finalement, l'espace $\mathcal{H}$ des hypothèses peut offrir des structures permettant son exploration de manière plus ou moins systématique et plus ou moins efficace. En particulier, une relation d'ordre sur $\mathcal{H}$ corrélée avec la généralité de l'induction effectuée est très utile (voir le chapitre 4).

Copyright © 2010 Eyrolles.

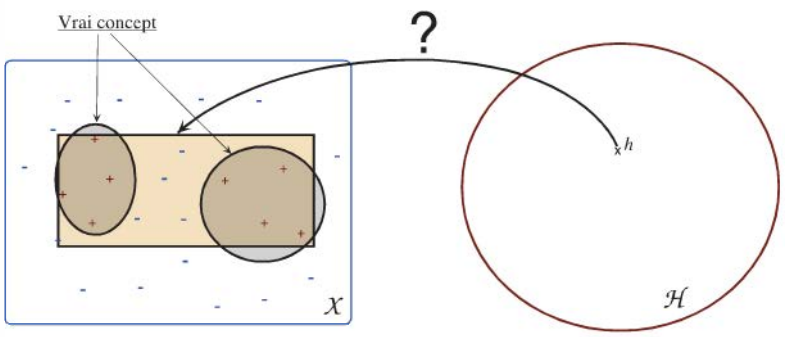


FIG. 1.10: *Supposons que le langage de représentation des hypothèses $\mathcal{L}_{\mathcal{H}}$ corresponde à une restriction aux parties de $\mathcal{X}$ qui sont des rectangles et que la partition « vraie » de la nature, correspondant aux exemples positifs, soit représentée par les deux « patatoïdes ». Dans ce cas, il est impossible d'approximer correctement le concept cible à l'aide d'une hypothèse de $\mathcal{H}$.*

5.3 L'exploration de l'espace des hypothèses

Soient un espace d'hypothèses $\mathcal{H}$, un espace d'entrée $\mathcal{X}$ et un échantillon d'apprentissage $\mathcal{S} = \langle(\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), ..., (\mathbf{x}_m, u_m)\rangle$. La tâche de l'apprenant est de trouver une hypothèse h approximant au mieux, au sens de la mesure de performance objectif – souvent le risque réel –, une fonction cible f sur la base de l'échantillon $\mathcal{S}$ dans lequel on suppose que chaque étiquette y_i a été calculée grâce à la fonction f appliquée à la forme $\mathbf{x}_i$.

Comment trouver une telle hypothèse $h \in \mathcal{H}$? Deux préalables sont nécessaires :

1. savoir *évaluer la qualité d'une hypothèse*, et ainsi pouvoir déterminer qu'une hypothèse satisfaisante (voire optimale) a été trouvée ;
2. savoir *organiser la recherche* dans $\mathcal{H}$.

Quel que soit le processus guidant l'exploration de $\mathcal{H}$, il est nécessaire que l'apprenant puisse *évaluer* les hypothèses h qu'il considère à un instant t de sa recherche. Pour cela, il utilise un critère inductif, tel que nous l'avons esquissé en section 1.3, par exemple un risque empirique. Cela permet à l'apprenant de mesurer sa performance sur l'échantillon d'apprentissage et de décider s'il doit poursuivre sa recherche dans $\mathcal{H}$ ou s'il peut s'arrêter.

Par exemple, dans le cas de l'apprentissage supervisé de concept, en supposant des descriptions non bruitées des entrées, l'apprenant cherche une hypothèse exprimable dans le langage $\mathcal{L}_{\mathcal{H}}$ couvrant tous les exemples positifs de l'échantillon d'apprentissage et ne couvrant aucun des exemples négatifs. Dans ce cas, le risque empirique sera en effet nul.

La figure 1.11 schématise la recherche d'une hypothèse dans le cas d'un apprentissage hors ligne (quand tout l'échantillon d'apprentissage est supposé d'emblée disponible). La figure 1.12 est relative à un apprentissage en ligne, dans lequel les exemples sont fournis séquentiellement. Dans ce dernier cas, on suppose ici que l'hypothèse courante h_t est comparée à l'entrée courante $\mathbf{z}_{t+1} = (\mathbf{x}_{t+1}, u_{t+1})$ et modifiée s'il y a lieu. Mais on peut également considérer une séquence d'hypothèses $\langle h_t \rangle$ dans le cas où l'apprenant calcule itérativement son hypothèse sur la base de l'échantillon d'apprentissage $\mathcal{S}$.

En supposant qu'à l'instant t, l'apprenant juge insatisfaisante son hypothèse courante h_t, comment peut-il en changer ? C'est là que se décide l'efficacité de l'apprentissage et que joue la structure exploitable sur l'espace $\mathcal{H}$. Plus celle-ci sera accordée à l'apprentissage, et plus il sera

Copyright © 2010 Eyrolles.

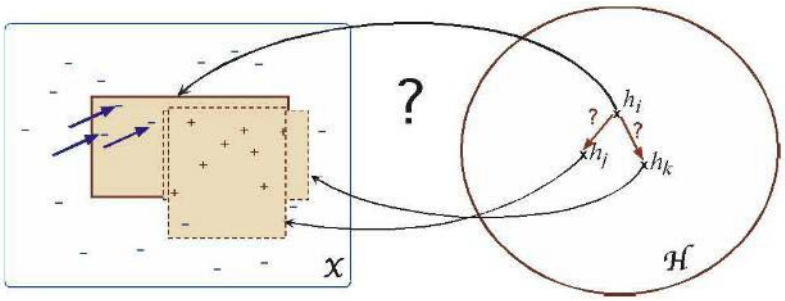


FIG. 1.11: *Si l'hypothèse courante h_t est insatisfaisante – ici elle n'exclue pas tous les exemples négatifs connus –, alors il faut que l'apprenant cherche une nouvelle hypothèse dans $\mathcal{H}$. La question est : où doit-il chercher ?*

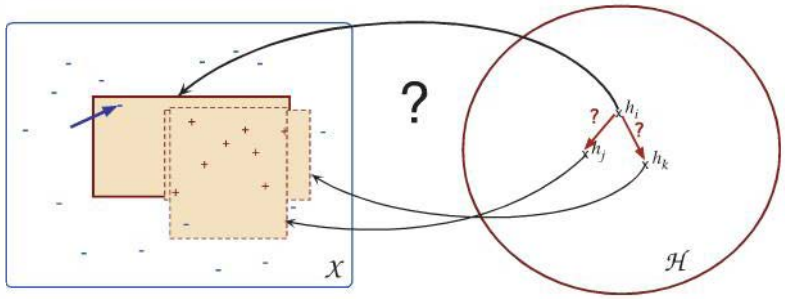


FIG. 1.12: *Si l'hypothèse courante h_t est insatisfaisante – ici elle ne couvre pas le nouvel exemple $z_{t+1} = (\mathbf{x}_{t+1}, \mathbf{u}_{t+1})$ –, alors il faut que l'apprenant cherche une nouvelle hypothèse dans $\mathcal{H}$. Encore une fois, la question est : où doit-il chercher ?*

envisageable d'organiser efficacement l'exploration de $\mathcal{H}$. Ainsi, on peut distinguer quatre grandes classes de possibilités correspondant à des connaissances *a priori* croissantes sur le monde.

1. **Il n'y a pas d'espace d'hypothèse**. Cela signifie que l'apprentissage ne met en jeu que l'espace d'entrée $\mathcal{X}$ et que le résultat de l'apprentissage ne peut être qu'une règle de décision ne s'appuyant que sur les exemples d'apprentissage. C'est le cas de la méthode des plus proches voisins (voir chapitre 15).
2. L'espace $\mathcal{H}$ des hypothèses ne présente **aucune structure**. Dans ce cas, seule une exploration aléatoire est possible. Rien ne permet de guider la recherche, ni même de tirer parti des informations déjà glanées sur $\mathcal{H}$. C'est le cas où l'on ne connaîtrait rien *a priori* sur $\mathcal{H}$.
3. **Une notion de voisinage est définissable sur $\mathcal{H}$**. Il est alors possible d'opérer une exploration par des techniques d'optimisation comme le gradient[14]. L'avantage de ces techniques, et ce qui les rend si populaire, c'est qu'elles sont d'un usage très général puisqu'il est souvent possible de définir une notion de voisinage sur un espace. Un problème fondamental est celui de la pertinence de cette notion de voisinage. Une mauvaise relation de voisinage peut en effet éloigner l'apprenant des régions prometteuses de l'espace ! Par ailleurs, c'est encore une structure faible qui, sauf dans des cas particuliers (différentiabilité, convexité, etc. de l'espace des hypothèses) ne permet pas une exploration rapide.

[14] Ce terme inclut ici des méthodes plus ou moins sophistiquées y compris les approches de type évolution simulée (algorithmes génétiques) et celles des réseaux connexionnistes.

Copyright © 2010 Eyrolles.

4. Il est parfois possible de disposer d'une structure plus forte permettant d'organiser l'exploration de $\mathcal{H}$. C'est le cas en particulier des **structures d'ordre partiel induites par des relations de généralité entre hypothèses.** Dans ce cas, par exemple, il devient possible de modifier une hypothèse erronée en la spécialisant juste assez pour qu'elle ne couvre plus le nouvel exemple négatif, ou au contraire en la généralisant juste assez pour qu'elle couvre le nouvel exemple positif fourni. Ce type d'exploration, possible en particulier quand l'espace des hypothèses est structuré par un langage (voir chapitre 4), est généralement mieux guidé et plus efficace qu'une exploration aveugle.

De ce qui précède, il est évident que plus la structuration de l'espace des hypothèses est forte et adaptée au problème d'apprentissage, et plus les connaissances *a priori*, s'exprimant en particulier dans les biais et dans le critère de performance, sont importantes, plus l'apprentissage sera facilité. En contrepartie, bien sûr, cela nécessitera un travail de réflexion préalable d'autant plus important.

6. Retour sur l'organisation de l'ouvrage

Toute étude de l'apprentissage artificiel peut se situer par rapport à trois pôles :

- Une *approche théorique* de l'apprentissage s'attachant à identifier ce qu'il est possible d'apprendre ou, plus précisément, ce qui est nécessaire pour qu'un apprentissage soit possible en principe.
- Une *approche d'ingénieur* concerné par la réalisation de méthodes d'apprentissage sous formes d'algorithmes et de programmes informatiques.
- Une *approche d'utilisateur* intéressé par les réalisations des programmes d'apprentissage et les problèmes qu'ils permettent de résoudre.

Nous avons essayé de rendre compte de ces trois points de vue tout au long de l'ouvrage, même si la progression logique impose de partir de prémices plutôt conceptuelles et théoriques pour aller vers la conception de systèmes d'apprentissage et, de là, vers les applications.

Tout ouvrage général sur l'apprentissage artificiel doit affronter la difficulté d'avoir à présenter une collection de méthodes et d'algorithmes parfois issus de communautés scientifiques différentes, pour des motivations diverses (métaphores biologiques, modèles de la physique, architecture cognitive...) et souvent décrits dans les articles scientifiques à l'aide de notations non homogènes. Chaque auteur doit alors faire un choix pour organiser, le moins arbitrairement possible, l'exposition de toutes ces techniques. Parmi nos illustres prédécesseurs, Tom Mitchell [Mit97] a choisi d'équilibrer tout au long de l'ouvrage théorie et pratique, à l'image de son cours à l'université de Carnegie-Mellon (CMU), sans suivre de principe directeur particulier et en assumant les différences de notations entre les écoles de pensée. Pat Langley [Lan96] a fait le pari audacieux de structurer tout son ouvrage sur les langages de représentation des hypothèses manipulés par les systèmes apprenants, en imposant une notation uniforme et des exemples de tâches d'apprentissage illustrant l'ensemble des méthodes. Plus récemment, Christopher Bishop, dans un excellent ouvrage [Bis06], propose une lecture complètement bayésienne de l'apprentissage en mettant l'accent sur des espaces d'hypothèses considérés comme des distributions de probabilités et sur des principes inductifs spécifiques à ce point de vue : maximum de vraisemblance, etc. Pour leur part, Hastie, Tibshirani et Frieman [HTF09], sans nécessairement adopter strictement le point de vue bayésien, présentent l'apprentissage selon une perspective totalement statistique. Ils insistent ainsi particulièrement sur les tests en validation et généralement sur la manipulation

Copyright © 2010 Eyrolles.

de densités de probabilité. Enfin, un autre livre digne d'attention est celui de Simon Haykin [Hay08]. Cet ouvrage remarquable est un livre d'ingénieur, influencé par le point de vue de la théorie du contrôle et orienté vers les réseaux de neurones.

Comme nous l'avons déjà dit dans l'avant-propos, nous avons choisi de suivre un autre principe structurant. Nous présentons dans une première partie les concepts et principes fondamentaux qui permettent de comprendre et de justifier la plupart des méthodes d'apprentissage. En particulier nous nous sommes attachés à l'étude des conditions sous lesquelles un apprentissage est possible, ou impossible, et ceci indépendamment d'un algorithme particulier. Ceci nous permet de cerner les conditions nécessaires à un apprentissage, ainsi que de motiver l'utilisation de certains principes inductifs que l'on retrouve à la base de toutes les méthodes d'apprentissage. Le reste de l'ouvrage est dédié aux méthodes et algorithmes d'apprentissage ainsi qu'aux réalisations associées. Afin d'en organiser l'exposition, nous avons choisi de centrer notre attention sur le problème de la recherche d'une ou plusieurs hypothèse(s) dans l'espace d'hypothèses $\mathcal{H}$. Dans la section précédente, nous avons évoqué l'influence des connaissances préalables sur le processus de recherche et son efficacité. Plus l'espace $\mathcal{H}$ se trouve doté d'une structure forte, et plus son exploration peut être guidée, conduisant en général à une plus grande efficacité. L'ouvrage adopte cette ligne directrice en présentant les méthodes d'apprentissage en fonction de la structuration de $\mathcal{H}$, partant des espaces les mieux structurés, pour aller graduellement vers l'apprentissage dans des espaces « minimaux » pour lesquels il n'existe même plus d'espace d'hypothèses à proprement parler, mais seulement une notion de voisinage dans l'espace $\mathcal{X}$ des entrées et une mesure de performance. Les trois grandes parties présentant ces méthodes regroupent ainsi d'abord les méthodes d'*apprentissage par exploration* suivant les directions de recherche fournies par l'espace d'hypothèses, ensuite les méthodes d'*apprentissage par optimisation* lorsque $\mathcal{H}$ ne dispose plus que d'une notion de voisinage et d'une mesure de performance, et que sont donc utilisables essentiellement des techniques de gradient, finalement les méthodes d'*apprentissage par interpolation* qui sont les seules utilisables quand on ne connaît plus d'espace d'hypothèses *a priori*.

Plus la connaissance préalable est faible, et plus l'apprentissage requiert de données pour aboutir. On ne peut pas gagner sur tous les tableaux. En contrepartie, les méthodes développées pour les tâches dans lesquelles on dispose de peu d'informations préalables sont aussi celles qui sont d'usage le plus général, s'adaptant à tous les contextes. C'est pourquoi ces méthodes (par exemple les réseaux connexionnistes ou les algorithmes génétiques) sont les plus populaires, prêtes à être essayées sans grands efforts de réflexion *a priori*. Nous avons voulu souligner que ce calcul est parfois mauvais, et qu'il est souvent rentable de chercher à tirer parti de toutes les connaissances disponibles. Par ailleurs, il nous semble aussi que les tâches d'apprentissage essentiellement numériques qui ont fait florès ces dernières années vont probablement un jour céder le pas à des tâches d'apprentissage – comme la recherche de documents sur le réseau, leur analyse automatique, etc. – requiérant des espaces d'hypothèses beaucoup plus structurés et prenant en compte une énorme quantité de connaissances. C'est pourquoi nous avons réservé une place importante à ces méthodes, malgré leur usage encore modéré dans les applications actuelles.

Copyright © 2010 Eyrolles.

Notes pour aller plus loin

L'apprentissage artificiel est une discipline jeune, à l'instar de l'intelligence artificielle et de l'informatique, mais elle a déjà une histoire. Nous la brossons ici à grands traits, croyant qu'il

est toujours intéressant de connaître le passé d'une discipline, et que cela peut révéler, par les tensions mises à jour, les problèmes profonds, les grandes options, les écarts nécessaires.

Il serait bien sûr intéressant d'examiner l'étude de l'apprentissage artificiel dans une perspective plus large, tant historiquement, en remontant l'histoire de la pensée, que par la prise en compte des parrainages exercés par d'autres disciplines : philosophie, psychologie, biologie, logique, mathématique, etc. Cette étude qui reste à mener dépasse cependant de beaucoup le cadre de notre ouvrage et nous nous limiterons à quelques jalons.

Des principes préliminaires théoriques de l'apprentissage sont posés dès les premiers travaux en statistiques, dans les années 1920 et 1930, cherchant à déterminer comment inférer un modèle à partir de données, mais surtout comment valider une hypothèse par rapport à un jeu de données. Fisher en particulier étudie les propriétés des modèles linéaires et comment ils peuvent être inférés à partir d'un échantillon de données. À la même période, l'informatique naît avec les travaux de Gödel, Church puis surtout Turing en 1936, et les premières simulations informatiques deviennent possibles après la seconde guerre mondiale. À côté des réflexions théoriques et des débats conceptuels sur la cybernétique et le cognitivisme, dont nous avons parlé dans ce chapitre, des pionniers essaient de programmer des machines pour réaliser des tâches intelligentes, intégrant souvent de l'apprentissage. C'est particulièrement le cas des premières simulations de tortues ou souris cybernétiques que l'on place dans des labyrinthes en espérant les voir apprendre à s'en sortir de plus en plus vite. De son côté, Samuel chez IBM, dans les années 1959-1962, développe un programme pour jouer au jeu de dames américain qui apprend une fonction d'évaluation des positions lui permettant de devenir rapidement un très bon joueur.

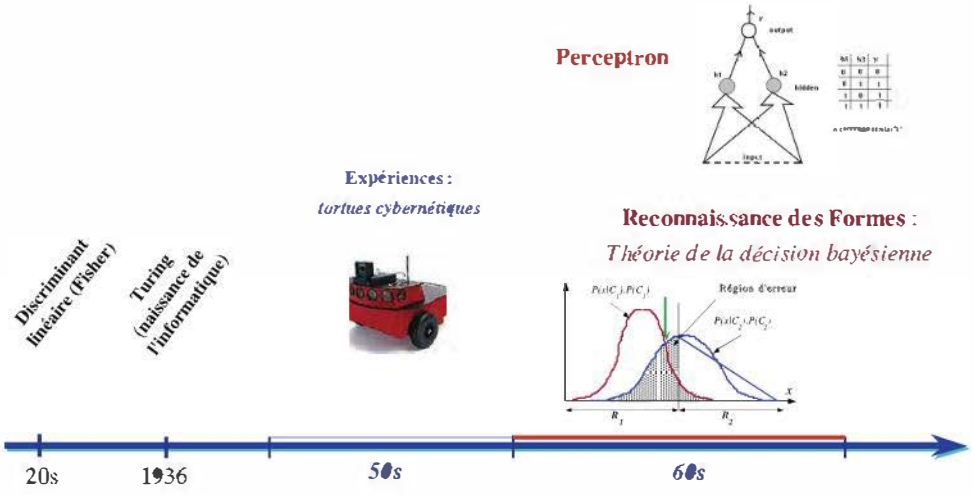


FIG. 1.13: *La première période de l'apprentissage artificiel.*

Dans les années 1960, l'apprentissage est marqué par deux courants. D'une part, un premier connexionnisme, qui sous la houlette de Rosenblatt père du perceptron, voit se développer des petits réseaux de neurones artificiels testés sur des tâches d'apprentissage supervisé de classes d'objets. D'autre part, des outils conceptuels sur la reconnaissance des formes se développent.

À la fin des années 1960, la publication du livre de Minsky et Papert [MP69] qui énonce les limites des perceptrons a pour effet d'arrêter pour une quinzaine d'années presque toutes les recherches dans ce domaine. De manière concomittante, l'accent mis en intelligence artificielle, dans les années 1970, sur les connaissances, leur représentation et l'utilisation de règles d'inférence sophistiquées (période des systèmes experts) encourage les travaux sur l'apprentissage dans des systèmes basés sur des représentations des connaissances structurées mettant en jeu des règles

Copyright © 2010 Eyrolles.

d'inférence complexes comme la généralisation, l'analogie, etc. C'est alors le triomphe de systèmes impressionnants réalisant des tâches d'apprentissage spécifiques en simulant plus ou moins des stratégies mises en jeu dans l'apprentissage humain. On retiendra en particulier le système ARCH de Winston en 1970 (voir chapitre 2) qui apprend à reconnaître des arches dans un monde de blocs à partir d'exemples et de contre-exemples ; le système AM de Lenat en 1976, qui découvre des conjectures dans le domaine de l'arithmétique par l'utilisation d'un jeu d'heuristiques elles-mêmes apprises dans le système EURISKO du même auteur en 1982, ou bien encore le système META-DENDRAL de Mitchell qui apprend des règles dans un système expert dédié à l'identification de molécules chimiques.

C'est aussi une période durant laquelle le dialogue est facile et fécond entre les psychologues et les praticiens de l'apprentissage artificiel, les hypothèses portant dans les deux communautés sur des concepts comme les mémoires à court terme et à long terme, le type procédural ou déclaratif des connaissances, etc. D'où aussi des systèmes comme ACT de Anderson testant des hypothèses générales sur l'apprentissage de concepts mathématiques dans l'éducation.

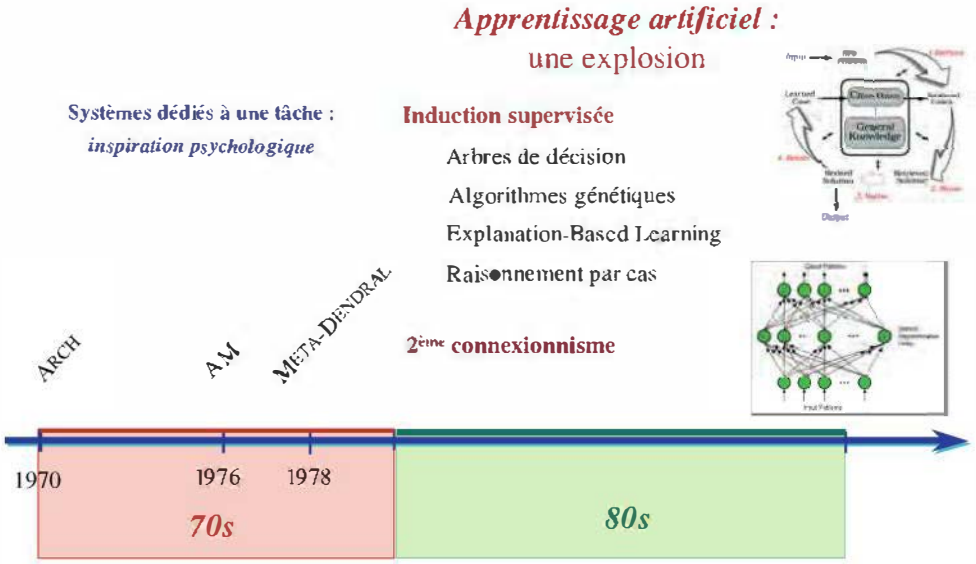


FIG. 1.14: *La deuxième période de l'apprentissage artificiel.*

Cependant, aussi spectaculaires soient-ils, ces systèmes présentent des faiblesses qui viennent de leur complexité. La première, la moins déterminante mais néanmoins influente, est qu'ils sont à la limite de ce qui est réalisable dans le cadre d'un travail de thèse, c'est-à-dire le *quantum* d'action dans l'institution scientifique. La deuxième est que leur réalisation implique nécessairement un grand nombre de choix, petits et grands, souvent implicites, et qui de ce fait ne permettent pas une réplication aisée des expériences, et surtout jettent le doute sur la portée générale et générique des principes mis en avant. C'est pourquoi les années 1980 ont vu progressivement se tarir les travaux portant sur de telles simulations à quelques brillantes exceptions près comme les systèmes ACT ou SOAR.

De plus, ces années ont vu une réémergence très puissante du connexionnisme en 1985, avec en particulier la découverte d'un nouvel algorithme d'apprentissage par descente de gradient pour les perceptrons multicouche (voir chapitre 10). Cela a profondément modifié l'étude de l'apprentissage artificiel en ouvrant grand la porte à tous les concepts et techniques mathématiques portant sur l'optimisation et sur les propriétés de convergence. Parallèlement à l'intrusion des mathématiques continues, d'autres mathématiciens se sont engouffrés (derrière Valiant en 1984 [Val84])

Copyright © 2010 Eyrolles.

dans la brèche ouverte par la notion d'espace des versions due à Mitchell (voir chapitre 4) et qui en gros permet d'envisager l'apprentissage comme la recherche, dans un espace d'hypothèses défini *a priori*, d'une hypothèse cohérente avec les données. D'un seul coup l'apprentissage était vu non plus comme la recherche d'algorithmes simulant une tâche d'apprentissage, mais comme un processus d'élimination d'hypothèses ne satisfaisant pas, ou mal, un critère d'optimisation. Il s'agissait alors dans ce cadre de chercher comment un échantillon de données tiré aléatoirement pouvait permettre d'identifier une bonne hypothèse dans un espace d'hypothèses donné. C'était extrêment déroutant, et comme le langage utilisé dans ces recherches était assez éloigné de celui des praticiens de l'apprentissage artificiel, ceux-ci continuèrent à développer des algorithmes plus simples mais plus généraux que ceux de la décennie précédente : arbres de décision (chapitre 13), algorithmes génétiques (chapitre 8), induction de programmes logiques (chapitre 5), etc.

Ce n'est que dans les années 1990, et surtout après 1995 et la parution d'un petit livre de Vapnik [Vap95], que la théorie statistique de l'apprentissage (chapitres 2 et 21) a véritablement influencé l'apprentissage artificiel en donnant un cadre théorique solide à des interrogations et à des constatations empiriques effectuées dans la pratique de l'apprentissage artificiel.

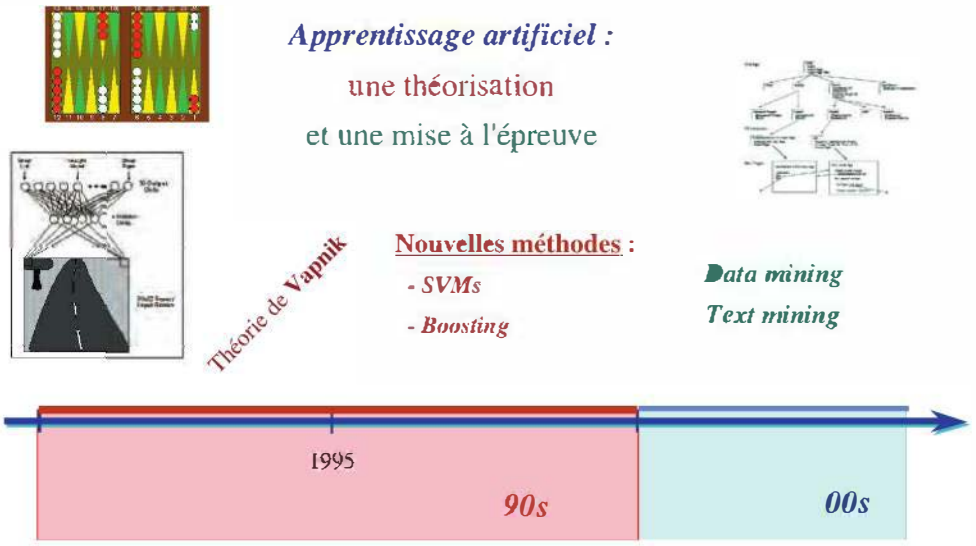


FIG. 1.15: *La troisième période de l'apprentissage artificiel.*

Le développement actuel de la discipline est dominé à la fois par un effort théorique vigoureux dans les directions ouvertes par Vapnik et les théoriciens de l'approche statistique, et par un redéploiement vers la mise à l'épreuve des techniques développées sur de grandes applications à finalité économique, comme la fouille de données, ou à finalité socio-économiques, comme la génomique.

Par ailleurs, à côté des tâches classiques d'induction, telles la classification ou la régression, de nouvelles applications sollicitent de nouvelles formes d'apprentissage. Ainsi, par exemple, les systèmes de recommandation qui se développent sur Internet bénéficieraient, et bénéficient, d'algorithmes capables d'apprendre à trier à partir d'exemples particuliers de préférences. C'est ce que l'on appelle le problème du « *ranking* » (voir chapitres 2 et 20). De même, la disponibilité de très grandes bases d'exemples, dont la plupart non étiquetés, par exemple les pages indexées sur Internet, a motivé une approche de l'apprentissage pouvant tirer parti des informations fournies par un tel échantillon. Il s'agit de l'*apprentissage semi-supervisé* (voir chapitre 19). Les données proviennent de plus en plus de mesures s'échelonnant sur le temps, ce que l'on appelle des *flots de données*. Là encore, de nouveaux algorithmes, de *data streaming* et d'*apprentissage en ligne*,

Copyright © 2010 Eyrolles.

doivent être mis au point. Nous citerons aussi des apprentissages à partir d'exemples composites, comme des molécules, qui ne peuvent pas être décrits par des vecteurs. L'*apprentissage multi-instance* (voir chapitre 21) est l'une des réponses à ce problème.

Il est indéniable que pour le moment l'apprentissage est ressenti comme nécessaire dans de très nombreux champs et que nous vivons un âge d'or pour cette discipline. Cela ne doit cependant pas faire oublier les immenses territoires laissés en friche (voir chapitre 20), ni la nécessité de renouer le dialogue avec les psychologues, les didacticiens, et plus généralement tous ceux qui travaillent sur l'apprentissage sous une forme ou une autre.

Les lecteurs intéressés par des articles généraux sur l'apprentissage peuvent se reporter à des articles parus dans des magazines scientifiques, dont : plusieurs numéros hors série de la revue Science & Vie : *Le cerveau et l'intelligence* déc. 1991, *À quoi sert le cerveau ?* juin 1996, *Le cerveau et la mémoire* mars 1998, *Les performances de la mémoire humaine* sept. 2000, *Le siècle des robots* juin 2009 ; des numéros hors série de la revue La Recherche : *L'intelligence artificielle* oct. 1985, *La mémoire* juil. 1994, *L'intelligence* déc. 1998, *La mémoire et l'oubli* juil. 2001, *Cerveau sans mémoire* 2003, *La mémoire* 2006, *L'intelligence* mars 2009, *Spécial Mémoire. Nouveaux regards sur le cerveau* juillet-août 2009 ; un numéro hors série de la revue Pour la Science : *La mémoire : le jardin de la pensée* avril 2001.

Une large littérature existe sur les fondements philosophiques de l'intelligence artificielle ou sur son histoire, dont une part non négligeable provient de travaux français. Nous citons ici des ouvrages faciles à se procurer, mais nous encourageons le lecteur à faire des recherches plus amples (voir [And92, BKL00, Cre97, Dup94, Eng96, Pin97]).

Voici une liste non exhaustive de revues spécialisées sur l'apprentissage artificiel :

- *Journal of Artificial Intelligence Research* (JAIR) accessible gratuitement sur Internet (http://www.jair.org/)
- *Journal of Machine Learning Research* (disponible gratuitement sur http://jmlr.csail.mit.edu/)
- *Machine Learning journal* (nécessite un abonnement : http://www.springer.com/computer/ai/journal/10994)
- *Data Mining and Knowledge Discovery journal*
- *Transactions on Knowledge and Data Engineering*
- *IEEE PAMI*

Voici aussi une liste de conférences complètement dédiées à l'apprentissage artificiel. Beaucoup d'autres dans le domaine de l'intelligence artificielle, de la reconnaissance des formes et de la fouille de données sont aussi pertinentes :

- *Conférence francophone d'Apprentissage* (CAP) : conférence annuelle francophone qui a pris la suite des Journées Françaises d'Apprentissage (JFA) depuis 1999.
- *European Conference on Machine Learning and on Principles of Knowledge Discovery in Databases* (ECML-PKDD) : conférence annuelle européenne (mais internationale).
- *International Conference on Machine Learning* (ICML) : conférence annuelle internationale (mais dominée par les Américains).
- *Neural Information Processing Systems* (NIPS) : conférence annuelle internationale dédiée à l'approche numérique et statistique de l'apprentissage et à la modélisation du système nerveux.
- *International conference on Knowledge Discovery in Databases* (KDD) : conférence annuelle

Copyright © 2010 Eyrolles.

internationale dédiée aux applications de l'apprentissage artificiel et de la fouille de données.

- *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD) : conférence annuelle internationale dédiée aux applications de l'apprentissage artificiel et de la fouille de données.
- *Uncertainty in Artificial Intelligence* (UAI) : conférence annuelle internationale dédiée principalement à l'approche statistique de l'intelligence artificielle et de l'apprentissage.

Une liste plus complète de ressources et d'appels à communication se trouve sur notre site : http://www.lri.fr/~antoine/Research/intro-cfp.html

Résumé

Il existe plusieurs types ou problèmes d'apprentissage qui sont définis par un certain nombre de caractéristiques dont l'espace des données, l'espace des hypothèses et le protocole régissant les interractions de l'apprenant avec son environnement. On distingue particulièrement l'apprentissage supervisé pour lequel un oracle fournit les réponses désirées, l'apprentissage non supervisé et l'apprentissage par renforcement.

L'étude de l'apprentissage tourne en particulier autour de deux questions :

- L'apprentissage est-il possible pour un problème et des données d'apprentissage particuliers ?
- L'apprentissage est-il réalisable efficacement ?

Les réponses à ces deux questions dépendent en grande partie de l'espace des hypothèses :

- Pour que l'apprentissage soit possible, il est nécessaire qu'il existe un biais d'apprentissage.
- L'efficacité de l'apprentissage dépend de la force de ce biais et de la structuration de l'espace des hypothèses.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 2

Première approche théorique de l'induction

Regardons passer un vol de cygnes en file indienne. Pouvons-nous prédire quel sera le prochain oiseau à passer ? Encore un cygne ? Une oie ? Le vol va-t-il s'arrêter ? Dans combien de temps passera un autre vol ? Que ce soit à l'aide d'expériences de pensée impliquant des volatiles ou plus classiquement des émeraudes (vertes jusqu'ici, mais bleues peut-être à partir de demain [1]), les philosophes ont cherché à comprendre l'induction, ce qui rend possible le passage de l'observation d'événements passés à la prédiction.

À ses débuts, l'apprentissage artificiel a effectué de l'induction comme M. Jourdain de la prose, sans s'arrêter à tous les problèmes profonds liés à l'induction comme forme de raisonnement. Après la réalisation de quelques systèmes pionniers et de quelques techniques impressionnantes, mais difficiles à maîtriser, l'étude de l'induction est devenue petit à petit plus rigoureuse.

En apprentissage artificiel, l'induction met en jeu d'abord un problème, donc des règles du jeu et une mesure de performance, ensuite un principe inductif qui spécifie ce qu'est l'hypothèse idéale étant donné un problème, et finalement un algorithme qui réalise au mieux le principe inductif, c'est-à-dire qui cherche effectivement l'optimum défini par le principe inductif. Ce chapitre est concerné par les deux premiers points. Une fois le problème d'apprentissage posé, plusieurs principes inductifs sont imaginables : choisir l'hypothèse qui s'accorde le mieux avec les données observées jusque-là, ou bien choisir l'hypothèse permettant la description la plus économique de ces données, ou d'autres encore. L'étude théorique, relativement récente, des conditions de validité de ces grands principes a conduit à la définition de principes inductifs plus sophistiqués qui sont à la base de nouvelles méthodes d'apprentissage.

[1] Il s'agit d'une métaphore classique, initialement introduite par le philosophe Goodman, pour discuter de l'induction. Voir le chapitre de J.G Ganascia dans [DKBM00].

Copyright © 2010 Eyrolles.

Sommaire

1 Poser un problème d'apprentissage 40
1.1 Définition du scénario d'apprentissage 40
1.1.1 Cas de l'apprentissage supervisé pour la classification 41
1.1.2 Cas de la régression 41
1.1.3 Cas de l'apprentissage de fonction de tri (*ranking*) 42
1.2 Formalisation des objectifs de l'apprentissage 43
1.2.1 Cas de l'apprentissage supervisé pour la classification et la régression 43
1.2.2 Cas de l'apprentissage de fonction de tri (*ranking*) 45
2 Approches bayésiennes et approche directe pour décider 46
2.1 Approches bayésiennes 46
2.2 Approche directe par calcul de fonction de décision 48
3 Le critère inductif 49
3.1 Trois grands principes inductifs 49
3.1.1 Principe de minimisation du risque empirique (MRE) 49
3.1.2 Apprentissage bayésien 50
3.1.3 Principe de compression maximale d'information 52
3.2 Les inférences bayésiennes 53
3.2.1 L'inférence bayésienne pure 54
3.2.2 La règle du maximum a posteriori (MAP) 55
3.2.3 La règle du maximum de vraisemblance 55
3.2.4 Discussion 56
3.3 Minimisation du risque empirique : exemples de traductions 57
3.3.1 Cas de l'apprentissage pour classer 57
3.3.2 Cas de la régression 58
3.3.3 Cas de l'estimation de densité 59
3.3.4 Cas de l'apprentissage de fonction de tri (*ranking*) 60
3.3.5 Cas de la prédiction en ligne 60
4 Analyse du principe de *minimisation du risque empirique* 60
4.1 La loi des grands nombres 63
4.2 Le compromis biais-variance 67
4.3 La consistance du MRE dans le cas de $\mathcal{H}$ fini 68
4.4 Le cas où $\mathcal{H}$ contient la fonction cible 70
4.5 Quelques leçons partielles 72
4.6 Vers un principe MRE contrôlé 73
5 Le lien entre le passé et le futur et le *no-free-lunch theorem* 73
5.1 Le *no-free-lunch* theorem : toutes les méthodes se valent ! 74
5.2 Leçons à tirer du *no-free-lunch* theorem 77
6 Notes historiques et bibliographiques 78

Copyright © 2010 Eyrolles.

TOUT LE MONDE sait distinguer un corbeau d'un canard. La couleur, le cri, la vitesse du vol, la silhouette, beaucoup d'attributs les séparent. Toute personne qui observe l'un de ces deux oiseaux peut lui donner son nom pratiquement sans erreur. Pourtant, cet observateur n'a certainement pas déjà vu *tous* les corbeaux ni *tous* les canards. Mais à partir d'observations en nombre limité, il a appris à les distinguer, c'est-à-dire à trouver des régularités permettant leur identification. Cette forme d'apprentissage, tirant des lois générales à partir d'observations particulières, s'appelle *induction* ou *généralisation*.

Il y a dans le paragraphe ci-dessus un autre exemple d'induction : il est écrit « *tout le monde sait ...* », ce qui est une généralisation (exagérée). Il faudrait d'abord fixer le cadre où cette loi est opératoire, la tempérer par des contraintes géographiques (« *En France, tout le monde...* »), zoologiques (il y a en France beaucoup d'espèces de canards et plusieurs de corbeaux), etc. Mais même dans un cadre plus précis, cette affirmation ne fait que généraliser des observations. Elle signifie en réalité « *presque tous les gens que j'ai observés dans cet exercice sont capables sous certaines conditions de distinguer un corbeau d'un canard* ». D'où la formulation raccourcie, qui énonce une loi extraite d'observations.

Induire : expliquer, prédire, faire simple

Si l'on se place d'un point de vue philosophique, l'induction est liée à plusieurs notions :

- La généralisation, c'est-à-dire le passage d'observations particulières à des classes d'événements ou à des lois s'appuie souvent sur une *recherche d'explications*. En effet, classiquement, une explication scientifique est définie comme une assignation causale. On parvient à expliquer un phénomène si on l'a relié de façon univoque à des antécédents à travers une ou plusieurs loi(s) de la nature. Ainsi un corbeau est différent d'un canard *parce que* la théorie de l'évolution des espèces selon un certain schéma idéalement déductif dicte que, dans nos régions tempérées, en cette période de l'évolution, peuvent coexister un certain nombre d'espèces présentant certaines caractéristiques spécifiques. Cela détermine des classes d'animaux possibles, dont celles que l'on nomme corbeaux et canards. Si tout va bien, cette théorie va expliquer pourquoi les volatiles peuvent exister dans certaines classes de poids, présenter certaines couleurs, etc. À partir de là, il devient possible de savoir comment distinguer des classes d'oiseaux.
- Une explication, qui consiste à remonter d'un phénomène connu à ses causes inconnues, est valide quand elle peut être retournée en un *outil de prédiction* permettant d'aller de causes connues à des phénomènes encore inconnus. L'induction est donc également liée à la *capacité de prédiction*. Cette prédiction n'est peut-être pas vraie à 100%, mais elle est fondée et généralement valide. Il est rare de rencontrer en France des canards noirs comme des corbeaux, et cela pourrait s'expliquer par la théorie de l'évolution.
- Finalement, à côté de leur pouvoir prédictif, les descriptions et les explications sont aussi jugées à l'aune de leur *simplicité*, de leur élégance et de leur fécondité par rapport à l'ensemble des connaissances. La théorie de l'évolution est-elle bien insérée dans les connaissances générales ? Est-elle performante dans d'autres contextes ? Plus prosaïquement, cette théorie permet-elle de prédire que l'on peut distinguer les canards des corbeaux à partir de leur couleur seulement (en France) ? Est-il besoin pour distinguer un canard d'un corbeau de mesurer la longueur des plumes de leurs ailes, alors que des critères simples suffisent.

L'induction est donc intrinsèquement liée aux concepts d'*explication*, de *prédiction* et d'*économie de description*. Nous trouverons la trace de ces liens tout au long de l'ouvrage.

Copyright © 2010 Eyrolles.

L'induction artificielle

Le point de vue décrit ci-dessus présuppose chez l'agent cognitif l'existence de tout un ensemble de connaissances sophistiquées. Ce genre de connaissances complexes (comme la théorie de l'évolution) ne se trouve pas encore chez les agents cognitifs artificiels. Du point de vue de l'ingénieur, le problème est d'inférer par des moyens automatiques une bonne règle de décision à partir d'un échantillon restreint de données sur le phénomène étudié. Cette règle de décision peut avoir deux buts, non contradictoires : soit permettre uniquement la *prédiction* sur une nouvelle observation (l'oiseau que je vois maintenant, est-ce un canard ou un corbeau ?), soit correspondre à la *découverte d'une théorie* générale du phénomène qui à la fois l'explique et permet de prédire ce qui se passera dans chaque cas particulier possible (il suffit de considérer la couleur pour savoir à quel type d'oiseau on a affaire : corbeau ou canard).

D'où les questions fondamentales suivantes :

- Qu'est-ce qui autorise à généraliser à partir d'un échantillon limité de données ?
- Comment réaliser cette extrapolation? Suivant quel principe ? Est-ce qu'une bonne explication des données disponibles est la promesse d'une bonne capacité de prédiction ?
- Quelles garanties peut-on avoir sur les performances d'une extrapolation ?

Ce chapitre a pour objectif de présenter les outils conceptuels et théoriques qui ont été développés pour répondre à ces questions.

1. Poser un problème d'apprentissage

Supposons que l'on ait un objectif du type :

- Réaliser un système capable d'apprendre quels sont les traitements les plus efficaces à partir de l'examen de bases de données médicales.
- Concevoir un centre de contrôle domotique optimisant les dépenses énergétiques d'une maison à partir de l'observation des habitudes de ses occupants.
- Concevoir un système de surveillance à domicile de personnes victimes de la maladie d'Alzheimer capable d'adapter ses missions et ses seuils d'alarme à l'évolution de la maladie.
- ...

Comment s'y prendre ?

La première étape consiste à préciser les objectifs de l'apprentissage et le scénario d'interaction entre le système apprenant et son environnement.

1.1 Définition du scénario d'apprentissage

Le scénario d'apprentissage spécifie :

- la forme des *entrées* (e.g. vecteurs d'attribut-valeur, ensembles de valeurs, séquences, descriptions semi-structurées (par exemple, une description XML), descriptions structurées) ;
- la forme des *sorties* (e.g. décision binaire (0/1), classification : une classe parmi un ensemble de classes, décision associée à un degré de certitude, sortie structurée (par exemple, un réseau sémantique)) ;
- le *protocole*, c'est-à-dire ce qui règle les interactions entre l'apprenant et l'environnement (voir section 4.2.2).

Afin de préciser la forme que doit prendre un scénario d'apprentissage, nous allons envisager plusieurs cas.

Copyright © 2010 Eyrolles.

1.1.1 Cas de l'apprentissage supervisé pour la classification

Dans l'apprentissage supervisé, l'apprenant reçoit des entrées constituées de couples (*entrée*, *étiquette*), notés $(\mathbf{x}_i, u_i)$ et appelés *exemples* d'apprentissage. Les entrées, ou formes d'apprentissage $\mathbf{x}_i$ sont prises dans un espace d'entrées $\mathcal{X}$, et les étiquettes u_i dans un espace de sortie $\mathcal{Y}$. Dans le cas où l'espace de sortie est restreint à deux valeurs : $\{0,1\}$, {vrai, faux} par exemple, on parle d'*apprentissage* supervisé *de concept* (*concept learning*). Si l'espace de sortie est constitué d'un ensemble fini d'étiquettes $\{\omega_k\}_{1 \leq k \leq K}$, on parle de *classification*.

La plupart des études existantes supposent que l'incertitude à propos des exemples rencontrés est modélisée par un processus aléatoire correspondant à une distribution de probabilité jointe $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$ sur l'espace produit $\mathcal{X} \times \mathcal{Y}$ [2].

Deux mécanismes à deux étapes peuvent être envisagés pour rendre compte de cette distribution sur les exemples.

1. Dans le premier (voir figure 2.1), on suppose une première étape dans laquelle les formes $\mathbf{x}$ sont engendrées selon une distribution $\mathsf{p}_{\mathcal{X}}$ définie sur $\mathcal{X}$. On suppose que dans une deuxième étape chaque forme $\mathbf{x}$ engendre une sortie y (ou u si il s'agit d'un exemple d'apprentissage) selon une distribution conditionnelle à chaque forme $\mathsf{p}_{\mathcal{Y}|\mathcal{X}}$.



FIG. 2.1: *Premier scénario de génération d'exemples.*

2. Le second scénario est dual du précédent. On y suppose que c'est d'abord la classe y d'un exemple qui est engendrée selon une distribution *a priori* $\mathsf{p}_{\mathcal{Y}}$, puis que la forme associée $\mathbf{x}$ est engendrée selon une distribution conditionnelle $\mathsf{p}_{\mathcal{X}|\mathcal{Y}}$ (voir figure 2.2).



FIG. 2.2: *Second scénario de génération d'exemples.*

La loi jointe sur les exemples peut ainsi être décomposée comme : $\mathsf{p}_{\mathcal{X}\mathcal{Y}} = \mathsf{p}_{\mathcal{Y}}\mathsf{p}_{\mathcal{X}|\mathcal{Y}} = \mathsf{p}_{\mathcal{X}}\mathsf{p}_{\mathcal{Y}|\mathcal{X}}$.

Dans ce cadre, lorsque l'apprenant agit comme un *classifieur*, il doit résoudre le problème suivant : étant donnée une réalisation d'un exemple $\mathbf{x}$, décider à quelle classe appartient $\mathbf{x}$. Une règle de classification ou de décision est donc une fonction $h : \mathcal{X} \rightarrow \mathcal{Y}$, qui détermine la classe, ou l'étiquette, $h(\mathbf{x})$ à laquelle devrait être assignée la forme d'entrée $\mathbf{x}$. En apprentissage artificiel, cette fonction, qui est effectivement réalisée grâce à un algorithme, est appelée *hypothèse*, d'où la notation h [3].

1.1.2 Cas de la régression

Dans le cas de la régression, la supervision est numérique. L'apprenant reçoit de l'expert des couples (*entrée*, *sortie*), notés $(\mathbf{x}_i, u_i)$, où u_i est un scalaire de $\mathbb{R}$ (dans le cas général, un vecteur de $\mathbb{R}^n$).

[2] Afin d'alléger les notations, nous utiliserons $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$ pour dénoter $\mathsf{p}_{\mathcal{X}\mathcal{Y}}(\mathbf{x}, y)$, et de même pour $\mathsf{p}_{\mathcal{X}}$ au lieu de $\mathsf{p}_{\mathcal{X}}(\mathbf{x})$ et $\mathsf{p}_{\mathcal{Y}}$ au lieu de $\mathsf{p}_{\mathcal{Y}}(y)$, sauf si le contexte nécessite de préciser.

[3] En statistique, la tradition est de nommer cette notion *modèle*.

Copyright © 2010 Eyrolles.

1.1.3 Cas de l'apprentissage de fonction de tri (*ranking*)

Le problème d'apprendre à trier des « objets » ou à les ordonner a stimulé un intérêt croissant ces dernières années. Ce problème apparaît en effet naturellement dans des tâches de recherche d'information (*information retrieval*) sur Internet par exemple, de recommandation (*collaborative filtering*) ou, en général, chaque fois que l'on veut ordonner les réponses possibles, comme c'est le cas aussi dans la prise de décision.

La formalisation de ce problème est loin d'être stabilisée tant les variations rencontrées dans les applications peuvent être importantes concernant la forme des exemples d'apprentissage, celle des sorties désirées, les mesures de performance à considérer, etc.

Le scénario le plus générique semble cependant être celui rencontré en recherche d'information. Le système de tri prend en entrée une liste de documents $D = \{d_1, d_2, \ldots, d_m\}$, d'une part, et une requête q, d'autre part. Le système produit en sortie la liste ordonnée, soit de tous les documents, soit des n meilleurs d'entre eux (voir figure 2.3).

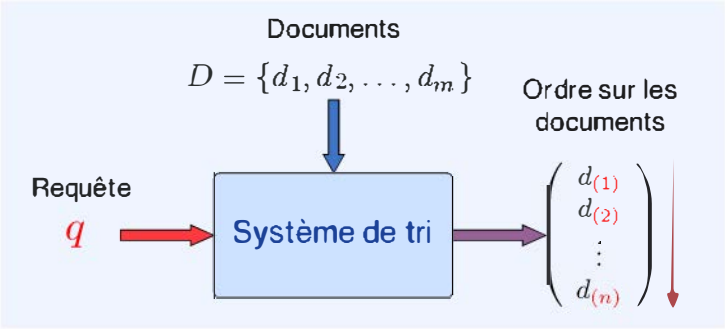


FIG. 2.3: *Tâche de tri de documents.*

Pour réaliser ce tri, le système met en œuvre une fonction qui peut soit attribuer une valeur $h(q, d)$ à chaque document d en fonction de la requête q, ce qui permet de classer les documents, soit plus directement établir une relation de préférence, notée $\succ$, sur les documents telle que $d_i \succ d_j$ si le document d_i est préféré au document d_j.

L'apprentissage s'effectue à l'aide d'un échantillon de requêtes $q_1, \ldots, q_m$ et des réponses associées, c'est-à-dire de listes ordonnées de documents. Dans le cas de la figure 2.4, on a supposé que l'apprentissage produit une fonction $h(q, d)$ permettant de calculer la valeur d'un document d pour une requête q. Une autre possibilité pourrait être que la fonction h calcule directement le rang du document d pour la requête q.

Ici encore, on suppose généralement que l'échantillon d'apprentissage est tiré aléatoirement suivant une distribution inconnue sur les requêtes et sur les documents.

Les scénarios exposés ci-dessus pour des problèmes d'apprentissage fréquents sont loin d'être exhaustifs. Ainsi, par exemple, le chapitre 16 décrit le scénario correspondant à l'apprentissage par renforcement pour des apprentissages s'étendant dans le temps. Des scénarios types, peu nombreux, ont été définis en apprentissage artificiel, et, à force d'être repris, ils ont fini par s'imposer comme des évidences. Il ne faut pourtant pas hésiter à les aborder avec un esprit critique et être prêt à définir un nouveau scénario si la tâche l'impose[4].

[4] Nous ne pouvons ici résister à rapporter une anecdote. En 1994, l'un d'entre nous discutait avec un éminent théoricien et défendait le besoin d'aborder des scénarios pouvant rendre compte des effets de séquence en apprentissage, c'est-à-dire du fait que l'ordre de présentation des exemples peut jouer un rôle. Ce chercheur balaya cette idée à ses yeux totalement iconoclaste. Puisque le scénario standard de données i.i.d. (indépen-

Copyright © 2010 Eyrolles.

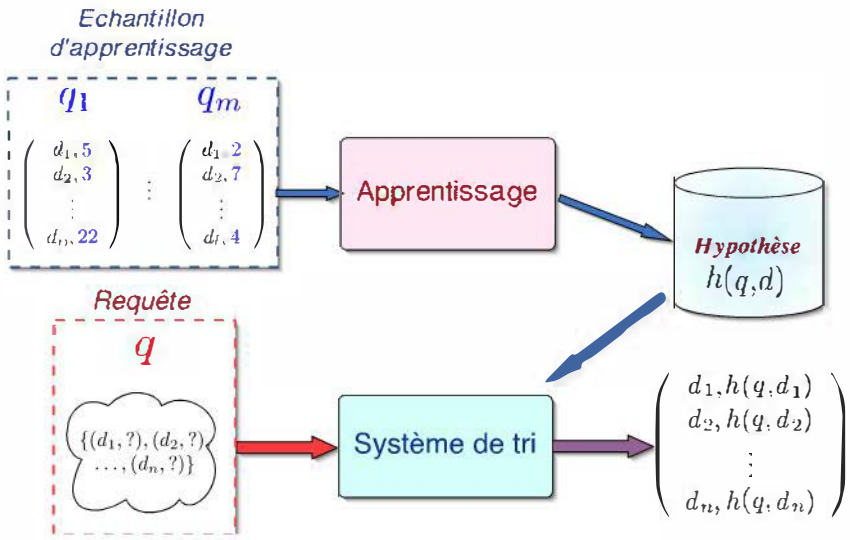


FIG. 2.4: *Scénario pour l'apprentissage de tri.*

Après avoir spécifié le scénario d'apprentissage, il faut définir l'objectif de l'apprentissage.

1.2 Formalisation des objectifs de l'apprentissage

À chaque problème d'apprentissage peut être associé un problème d'optimisation, c'est-à-dire un ou plusieurs critères mesurant les aspects que l'on cherche à améliorer.

Lorsque l'objectif principal est d'établir des prédictions et de prendre des décisions, l'accent est naturellement mis sur une mesure d'erreur sur les prédictions. Lorsque l'objectif est davantage de comprendre un phénomène dont on a observé un échantillon limité, la mesure de performance peut inclure des critères d'interprétabilité de la fonction de décision produite, à côté de sa performance en prédiction.

Nous allons examiner comment l'objectif de l'apprentissage a été formalisé dans le cadre de plusieurs tâches.

1.2.1 Cas de l'apprentissage supervisé pour la classification et la régression

Lorsque l'apprentissage a pour but la prédiction d'étiquettes à partir de la description de formes d'entrée, l'idéal serait naturellement que la fonction de décision apprise h coïncide avec la fonction cible f qui serait utilisée par la nature. À défaut de pouvoir atteindre un tel objectif, une mesure de performance raisonnable est la *probabilité d'erreur* de la fonction de décision apprise :

$$R_{\text{Réel}}(h) = \mathsf{p}_{\mathcal{X}\mathcal{Y}}\{h(\mathbf{x}) \neq y\} \tag{2.1}$$

c'est-à-dire la mesure, selon la distribution de probabilité $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$, de l'erreur commise par l'hypothèse apprise h.

damment et identiquement distribuées) s'appuyait sur des théorèmes de type MinMax, c'est-à-dire face au pire cas, il n'était nul besoin de faire appel à des formalisations plus fines. Quelques années plus tard, ce même théoricien devenait un ardent défenseur de nouveaux critères adaptés à l'apprentissage en ligne et clamait qu'il était évident qu'aucun jeu de données n'était réellement i.i.d. et que donc le scénario standard était limité, voire défectueux !

Copyright © 2010 Eyrolles.

---- EXEMPLE

Supposons que l'espace d'entrée $\mathcal{X}$ corresponde à une mesure de taille ($\mathcal{X} = \mathbb{R}$), et que l'espace de sortie soit à deux valeurs : $\mathcal{Y} = \{\texttt{Femme}, \texttt{Homme}\}$. Le monde est caractérisé par une certaine distribution de probabilité sur $\mathcal{X}$ et $\mathcal{Y}$: $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$. Par exemple, $\mathbf{p}_{\mathcal{X}}$ pour $x = 1.80m$ est assez forte alors qu'elle est faible pour $x \leq 1.40m$ (peu d'hommes ou de femmes de plus de 18 ans mesurent moins de 1.40m). Supposons aussi que $p(y = \texttt{Femme}|x = 1.80m) << p(y = \texttt{Homme}|x = 1.80m)$. Dans les zones de forte densité de probabilité de $\mathcal{X}$, il faut que la fonction de décision h soit précise (i.e. qu'elle prédise souvent $y = \texttt{Homme}$ pour x proche de 1.80m), car ces zones participent beaucoup à la mesure de performance (2.1). Il est moins important que cette fonction de décision se trompe dans les régions de $\mathcal{X}$ de faible densité, par exemple en-dessous de 1.40m.

Plus généralement, si des coûts différents sont associés à des types d'erreurs différents, cela est traduit par une *fonction de perte* (*loss function*) ℓ, définie comme suit :

$$\begin{aligned} \ell(h) : \mathcal{X} \times \mathcal{Y} &\rightarrow \mathbb{R}^+ \\ (\mathbf{x}, y) &\mapsto \ell(h(\mathbf{x}), y) \end{aligned} \tag{2.2}$$

---- EXEMPLE **(Fonction de perte asymétrique)**

La fonction de perte peut être asymétrique. Par exemple, en médecine, il peut être moins coûteux de diagnostiquer à tort une pathologie et de la traiter (pour rien), que de passer à côté de cette pathologie et de ne rien faire.

---- EXEMPLE **(Fonction de perte 0/1)**

Dans l'apprentissage de concept, c'est-à-dire lorsque la sortie peut prendre seulement deux valeurs {vrai, faux} interprétées comme *appartenant au concept* ou *n'appartenant pas au concept*, une fonction de perte très utilisée compte pour 1 chaque erreur de prédiction.

$$l(h(\mathbf{x}), y) = \begin{cases} 0 & \text{si } h(\mathbf{x}) = y \\ 1 & \text{si } h(\mathbf{x}) \neq y \end{cases} \tag{2.3}$$

---- EXEMPLE **(Fonction de perte quadratique)**

Pour les tâches de régression particulièrement la fonction d'erreur quadratique est généralement utilisée. Elle est en effet adaptée au cas où un bruit additif gaussien corrompt les données (voir chapitre 9).

$$l(h(\mathbf{x}), y) = (h(\mathbf{x}) - y)^2 \tag{2.4}$$

---- EXEMPLE **(Fonctions de perte distances)**

Plus généralement, les fonctions de distance peuvent être utilisées :

$$l(h(\mathbf{x}), y) = |h(\mathbf{x}) - y|^p \tag{2.5}$$

où $p > 0$ est un nombre fixé. Les valeurs $p = 1$ et $p = 2$ sont les plus usitées.

Copyright © 2010 Eyrolles.

Définition 2.1 (Risque réel)

La performance attendue de la fonction de décision h apprise est définie comme l'espérance du coût encourue si l'on utilise h dans le monde associé à la distribution de probabilité $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$. On appelle cette performance, le risque réel, *(la notation $L(h)$ est souvent également utilisée).*

$$R_{\text{Réel}}(h) = \mathbb{E}[\ell(h(\mathbf{x}), y)] = \int_{\mathbf{x}\in\mathcal{X}, y\in\mathcal{Y}} \ell(h(\mathbf{x}), y)\, \mathbf{p}_{\mathcal{X}\mathcal{Y}}\, d\mathbf{x}dy \tag{2.6}$$

Définition 2.2 (Fonction de décision optimale)

En supposant que la fonction de décision h soit prise à l'intérieur d'un ensemble de fonctions $\mathcal{H}$, l'hypothèse optimale, *notée h^*, pour un environnement spécifié par $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$ et par la fonction de perte ℓ est :*

$$h^* = \underset{h\in\mathcal{H}}{\text{ArgMin}}\, R_{\text{Réel}}(h) \tag{2.7}$$

Si l'on veut spécifier la meilleure décision y pour une entrée $\mathbf{x}$, on obtient la *règle de décision de Bayes*(1702-1761).

Définition 2.3 (Règle de décision optimale de Bayes)

La sortie optimale y^* *pour une entrée* $\mathbf{x}$ *est :*

$$y^* = \underset{y'\in\mathcal{Y}}{\text{ArgMin}} \int_{y\in\mathcal{Y}} \ell(y', y)\, \mathbf{p}_{\mathcal{Y}|\mathcal{X}}(y|\mathbf{x})\, dy \tag{2.8}$$

Il s'agit donc d'une sorte de centre de gravité de toutes les décisions $y \in \mathcal{Y}$ possibles minimisant la perte étant donnée l'entrée $\mathbf{x}$. Il faut bien noter que cette règle de décision optimale *n'appartient pas nécessairement à* $\mathcal{H}$. On notera son risque réel associé R^*.

1.2.2 Cas de l'apprentissage de fonction de tri (*ranking*)

Soit $\{\mathbf{x}, y\}$ une paire de variables aléatoires prise dans $\mathcal{X} \times \mathbb{R}$, où $\mathcal{X}$ est un espace énumérable de formes d'entrées (e.g. des documents). Et soit $\{\mathbf{x}', y'\}$ une autre paire de variables aléatoire suivant la même distribution que $\{\mathbf{x}, y\}$, et indépendante. Notons $Z = \frac{y-y'}{2}$.

Dans le problème de tri, on observe $\mathbf{x}$ et $\mathbf{x}'$, mais généralement pas leur étiquette (qui peut être par exemple une évaluation de leur pertinence). On dira que $\mathbf{x}$ est meilleur que $\mathbf{x}'$ si $y > y'$, c'est-à-dire si $Z > 0$.

Le but est d'ordonner $\mathbf{x}$ et $\mathbf{x}'$ de telle manière que la probabilité que la meilleure d'entre ces formes soit affectée d'une étiquette plus petite que l'autre soit aussi faible que possible. Formellement, une *règle de tri* est une fonction $r : \mathcal{X} \times \mathcal{X} \to \{-1, 1\}$. Si $r(\mathbf{x}, \mathbf{x}') = 1$, cela signifie que la règle met $\mathbf{x}$ avant $\mathbf{x}'$. La performance de la règle de tri est mesuré par le *risque de tri* :

$$L(r) = \mathbf{P}\{Z \cdot r(\mathbf{x},\mathbf{x}') < 0\} \tag{2.9}$$

c'est-à-dire par la probabilité que r trie deux formes d'entrée de manière erronée.

Copyright © 2010 Eyrolles.

Selon ce point de vue, le problème de tri devient équivalent à un problème de classification binaire dans lequel le signe de la variable aléatoire Z doit être estimé à partir de la paire $\{\mathbf{x}, \mathbf{x}'\}$. La règle de tri de risque minimal r^* peut alors être caractérisée de la manière suivante :

Soient :

$$\rho_+(\mathbf{x},\mathbf{x}') = \mathsf{P}\{Z > 0 \mid \mathbf{x},\mathbf{x}'\} \quad , \quad \rho_-(\mathbf{x},\mathbf{x}') = \mathsf{P}\{Z < 0 \mid \mathbf{x},\mathbf{x}'\}$$

Alors, la règle de risque minimal r^* est définie (en utilisant la fonction $\mathbb{I}_x$ qui prend la valeur 1 quand $x > 0$ et 0 autrement) par :

$$r^* = 2\,\mathbb{I}_{|\rho_+(\mathbf{x},\mathbf{x}') \geq \rho_-(\mathbf{x},\mathbf{x}')|} - 1$$

Notons $L^* = L(r^*) = \mathbb{E}\{\min(\rho_+(\mathbf{x},\mathbf{x}'),\rho_-(\mathbf{x},\mathbf{x}'))\}$. Alors pour toute règle de tri r, on a : $L^* \leq L(r)$.

Le problème d'apprentissage consiste à trouver une règle de tri de faible risque à partir de données d'apprentissage $\mathcal{S}_m = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$. Pour chaque règle candidate r, on peut évaluer son risque $L(r) = \mathsf{P}\{Z \cdot r(\mathbf{x},\mathbf{x}') < 0\}$. L'une des estimations les plus utilisées est la *U-statistique* :

$$L_m(r) = \frac{1}{m(m-1)} \sum_{i \neq j} \sum_{i \neq j} \mathbb{I}_{|Z_{i,j} \cdot r(\mathbf{x},\mathbf{x}') < 0|} \quad \text{où} \quad Z_{i,i} = \frac{y_i - y_j}{2}$$

Les U-statistiques ont été étudiées en profondeur. Une des inégalités classiques les concernant est celle d'Hoeffding selon laquelle pour tout $t > 0$, si $\sigma^2 = \mathrm{Var}(\mathbb{I}_{|Z_{i,j} \cdot r(\mathbf{x},\mathbf{x}') < 0|}) = L(r)(1 - L(r))$, alors :

$$\mathsf{P}\{|L_m(r) - L(r)|\} > t\} \leq 2\exp\left(-\frac{\lfloor (n/2) \rfloor t^2}{2\sigma^2 + 2t/3}\right) \tag{2.10}$$

où le symbole $\lfloor x \rfloor$ dénote l'entier immédiatement inférieur à x.

Un **cadre plus général** que le précédent considère le tri non plus de deux observations $\mathbf{x}$ et $\mathbf{x}'$ seulement, mais celui de m observations d'un seul coup. Dans ce cas, une règle de tri $r(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ fournit un classement sous la forme d'une permutation π de $\{1, \ldots, m\}$. Le but est alors que π coïncide ou soit aussi proche que possible de la permutation $\bar{\pi}$ pour laquelle $y_{\bar{\pi}(1)} \geq \ldots \geq y_{\bar{\pi}(m)}$. Étant donnée une fonction de perte l qui retourne un nombre compris dans $[0, 1]$ pour une paire de permutations en entrée, le risque de tri est défini comme :

$$L(r) = \mathbb{E}\, l(r(\mathbf{x}_1, \ldots, \mathbf{x}_m), \bar{\pi})$$

On a alors généralement recours à une U-statistique d'ordre m.

2. Approches bayésiennes et approche directe pour décider

Nous avons décrit le problème de l'apprentissage comme celui de la recherche d'une hypothèse ou fonction de décision performante en terme d'espérance de perte sur les événements à venir. Plusieurs approches permettent d'aborder ce problème.

2.1 Approches bayésiennes

Les approches bayésiennes décomposent le processus en deux étapes. D'abord, une étape d'*inférence* de modèles statistiques des données de type $\mathsf{p}_{Y|X}(\mathbf{x})$, puis une étape de *décision*

Copyright © 2010 Eyrolles.

s'appuyant sur ces probabilités *a posteriori* pour calculer la décision y^* optimale, selon l'équation (2.8). Il faut donc déterminer $\mathsf{p}_{\mathcal{Y}|\mathcal{X}}(y|\mathbf{x})$.

Il existe deux manières d'aborder ce problème. (Rappelons que nous étudions ici un problème de classification, c'est-à-dire que l'espace de sortie $\mathcal{Y}$ est fini, chaque élément $u_k \in \mathcal{Y}$ étant interprété comme l'étiquette d'une classe de formes).

1. La première consiste à utiliser la formule de Bayes :

$$\mathsf{p}_{\mathcal{Y}|\mathcal{X}}(y|\mathbf{x}) = \frac{\mathsf{p}_{\mathcal{X}|\mathcal{Y}}(\mathbf{x}|y)\,\mathsf{p}_{\mathcal{Y}}(y)}{\mathsf{p}_{\mathcal{X}}(\mathbf{x})} \tag{2.11}$$

Cela implique de calculer la distribution de probabilité conditionnelle $\mathsf{p}_{\mathcal{X}|\mathcal{Y}}(\mathbf{x}|u_k)$ pour chaque classe $u_k \in \mathcal{Y}$, et d'inférer les probabilités *a priori* des classes $\mathsf{p}_{\mathcal{Y}}(u_k)$. Le dénominateur peut se calculer comme :

$$\mathsf{p}_{\mathcal{X}}(\mathbf{x}) = \sum_{u_k \in \mathcal{Y}} \mathsf{p}_{\mathcal{X}|\mathcal{Y}}\,\mathsf{p}_{\mathcal{Y}}(u_k) \tag{2.12}$$

Cette approche est qualifiée de *générative* car la connaissance de la distribution sur les entrées $\mathsf{p}_{\mathcal{X}}$ et des distributions conditionnelles $\mathsf{p}_{\mathcal{Y}|\mathcal{X}}$ permet de générer des exemples synthétiques $(\mathbf{x}_i, u_i)$ selon la même distribution réelle $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$ si les distributions estimées sont correctes.

2. La deuxième méthode consiste à estimer directement les probabilités *a posteriori* $\mathsf{p}_{\mathcal{Y}|\mathcal{X}}(u_k|\mathbf{x})$, et retenir la meilleure classe grâce à l'équation (2.8). Cela ne permet plus de générer des exemples synthétiques, mais des informations sur la forme des classes y_k restent nécessaires. Bishop [Bis06] qualifie cette approche de *discriminante*.

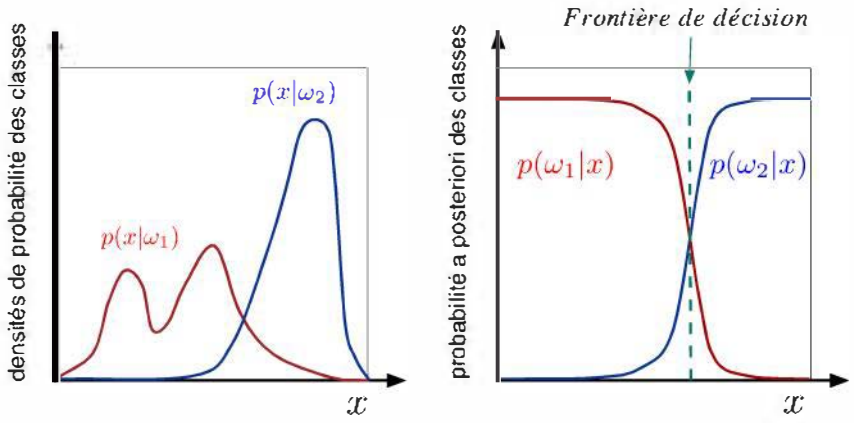


FIG. 2.5: *Exemple d'une tâche de classification à deux classes. À gauche, selon l'approche générative, on a estimé les probabilités conditionnelles* $\mathsf{p}_{\mathcal{X}|\mathcal{Y}}(\mathbf{x}|u_k)$. *À droite, selon l'approche discriminative, seules les probabilités a posteriori sont estimées. La connaissance des détails des distributions* $\mathsf{p}_{\mathcal{X}|\mathcal{Y}}(\mathbf{x}|u_k)$ *n'a pas d'effet sur la détermination du seuil de décision.*

La première approche, l'approche *générative*, est la plus exigeante en terme d'information à acquérir. Elle demande en particulier que soient estimées les distributions jointes sur $\mathcal{X}$ et sur chaque classe u_k. Lorsque la dimension de l'espace d'entrée $\mathcal{X}$ est élevée, cela requiert un très grand échantillon d'apprentissage. L'intérêt de connaître la distribution $\mathsf{p}_{\mathcal{X}}$ est de pouvoir

Copyright © 2010 Eyrolles.

détecter plus facilement les exemples étranges (*outliers*). En revanche, ce détour semble inutile quand l'objectif est de faire de la classification, qui peut utiliser directement les probabilités *a posteriori* $\mathrm{p}_{\mathcal{Y}|\mathcal{X}}(u_i|\mathbf{x})$. La figure 2.5 illustre l'inutilité de connaître en détail la structure des classes pour une tâche de classification.

Ce débat sur l'information nécessaire pour résoudre la tâche courante a été mené plus loin par Vapnik (~1935 ou ~1938, -), en particulier dans [Vap95] page 28. Pourquoi, en effet, chercher à estimer des distributions de probabilité qui requièrent l'estimation précise de nombreux paramètres quand ce que l'on cherche est une fonction de décision, c'est-à-dire une frontière dans l'espace d'entrée $\mathcal{X}$, qui, en général, implique beaucoup moins de paramètres?

2.2 Approche directe par calcul de fonction de décision

Dans l'approche directe pour la classification, il est pris acte que la seule chose nécessaire pour décider de la classe d'appartenance (ou plus généralement de l'étiquette) d'un exemple est une fonction $h(\mathbf{x})$ définie sur $\mathcal{X}$ et qui retourne la classe (ou l'étiquette) y de l'exemple $\mathbf{x}$ fourni.

L'avantage est que, *a priori* ou en fonction des caractéristiques du problème d'apprentissage, on peut décider quelle classe $\mathcal{H}$ de fonctions hypothèses doit être considérée. On pourra ainsi régler le nombre de paramètres que l'on est prêt à estimer en fonction de la richesse des informations fournies à l'apprenant. De plus, cette approche est généralement plus robuste que l'approche bayésienne, dans la mesure où une erreur de modélisation ou d'estimation de paramètres peut facilement conduire à une décision erronée, alors que la détermination directe d'une fonction de décision est moins sensible aux hypothèses formulées *a priori* sur le monde. Ainsi, par exemple, s'il a été supposé que les données sont générées par une distribution gaussienne, alors qu'elles le sont en réalité par une distribution à queue plus épaisse, la frontière de décision optimale peut se trouver nettement différente de celle obtenue en forçant l'adaptation d'une gaussienne aux données (voir figure 2.6).

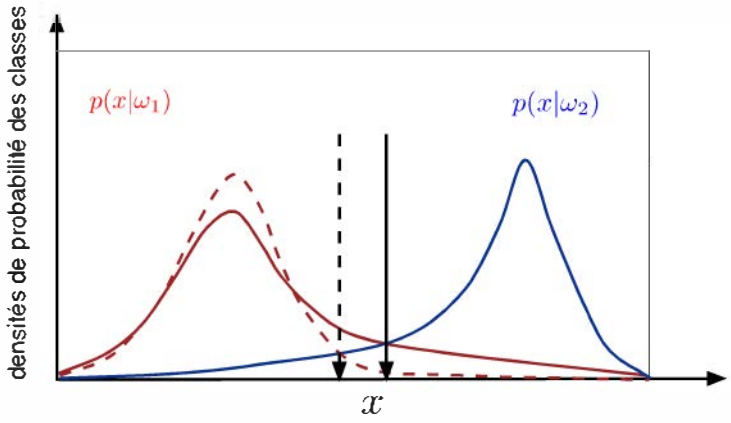


FIG. 2.6: *La frontière de décision estimée sur la base d'une distribution gaussienne de la classe ω_1 (en tirets) peut nettement différer de la frontière de décision optimale avec la vraie distribution des données (trait continu).*

La notion de probabilité ne joue plus aucun rôle dans cette approche, sauf, éventuellement, *a posteriori*, pour évaluer la significativité des hypothèses produites. Il n'est plus possible de générer des exemples synthétiques. D'un certain côté, on ne cherche plus à « comprendre » les données, mais le seul but est de prédire correctement. D'un autre côté, la connaissance des frontières de

Copyright © 2010 Eyrolles.

décision dans $\mathcal{X}$, et de leur expression dans $\mathcal{H}$, peut conduire à un autre type de description et donc de compréhension.

3. Le critère inductif

L'objectif de l'apprentissage est défini en terme de performance future attendue ou espérée. Dans ce cadre, on cherche une fonction de décision qui optimise cette performance. Mais il n'est évidemment pas possible d'optimiser directement le ou les critère(s) traduisant cette performance puisque celle-ci est définie sur des événements n'ayant pas encore eu lieu. Il faut donc avoir recours à un objectif de substitution, défini sur le passé, et dont on espère qu'en l'optimisant, on optimisera également le vrai objectif de l'apprentissage. Cet objectif de substitution est appelé *critère inductif*.

Mais avant d'examiner les critères inductifs usuels, qui définissent autant de problèmes d'optimisation, il est bon de prendre de la hauteur et d'examiner trois points de vue, d'ordre presque philosophique, sur la nature de l'induction. On parle alors de *principes inductifs*.

3.1 Trois grands principes inductifs

On peut identifier trois grandes classes de critères inductifs fondés sur des présupposés différents sur la nature de l'induction. Leur analyse montre cependant que, bien heureusement, ces critères entretiennent des relations étroites.

3.1.1 Principe de minimisation du risque empirique (MRE)

Dans le cas de l'apprentissage d'une règle de décision h, l'objectif est de trouver une règle h^* optimisant l'espérance de perte dont nous rappelons l'équation :

$$R_{\text{Réel}}(h) \;=\; L(h) \;=\; \mathbb{E}[\ell(h(\mathbf{x}), y)] \;=\; \int_{\mathbf{x}\in\mathcal{X}, y\in\mathcal{Y}} \ell(h(\mathbf{x}), y)\; \mathbf{p}_{\mathcal{X}\mathcal{Y}}\; d\mathbf{x}dy$$

Si la distribution de probabilité $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$ était connue, l'optimisation de cette quantité serait du ressort de méthodes d'intégration numérique (voir par exemple [RLPB08]) et de minimisation multi-dimensionnelle, voire de méthodes d'évolution simulée. Naturellement, elle ne l'est pas, sinon il n'y aurait pas de problème d'apprentissage.

Il faut donc soit l'estimer par la méthode générative, soit court-circuiter en partie cette estimation et estimer directement les densités de probabilité *a posteriori* $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}$, selon la méthode discriminative, soit la remplacer par une quantité empirique, on parle de *risque empirique*, tirée de l'échantillon d'apprentissage. Cette dernière approche est celle du *Principe de Minimisation du Risque Empirique (MRE)* (*Empirical Risk Minimization* ou ERM).

Formellement, cela consiste à remplacer la distribution jointe $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$ par :

$$\mathbf{p}_{\text{Emp}} \;=\; \frac{1}{m}\sum_{i=1}^{m} \delta_{\mathbf{x}_i}(\mathbf{x})\; \delta_{u_i}(y) \tag{2.13}$$

définie grâce à l'échantillon d'apprentissage $\mathcal{S}_m = \langle(\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), \ldots, (\mathbf{x}_m, u_m)\rangle \in \mathcal{Z}^m$. [5] La fonction $\delta_x(x')$ est égale à 1 si $x = x'$ et à 0 sinon. On se ramène donc à une densité de probabilité dont les seules « régions » non nulles coïncident avec les points d'apprentissage.

[5] Par souci de simplicité, $\mathcal{S}_m$ sera souvent noté $\mathcal{S}$, sauf lorsque l'on voudra mettre l'accent sur sa taille.

Copyright © 2010 Eyrolles.

Définition 2.4 (Risque empirique)

La performance mesurée sur l'échantillon d'apprentissage $\mathcal{S}_m = \langle (\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), \ldots, (\mathbf{x}_m, u_m) \rangle \in \mathcal{Z}^m$, *est définie par le* risque empirique *:*

$$R_{\text{Emp}}(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h(\mathbf{x}_i, u_i)) \tag{2.14}$$

Définition 2.5 (Principe de Minimisation du Risque empirique (MRE))

Le principe de minimisation du risque empirique *stipule de choisir l'hypothèse* h *qui minimise le risque empirique :*

$$h^*_{\mathcal{S}} = \underset{h \in \mathcal{H}}{\text{ArgMin}}\ R_{\text{Emp}}(h) \tag{2.15}$$

Par rapport aux autres approches essayant d'estimer des densités de probabilité, l'avantage de cette méthode est qu'elle conduit à l'optimisation d'une quantité facile à calculer. Par ailleurs, il peut sembler raisonnable de remplacer une intégrale par une somme sur des éléments tirés selon la même loi. Cependant, nous verrons (section 4) que, sans précaution, rien ne garantit qu'une hypothèse $h^*_{\mathcal{S}}$ minimisant le risque empirique R_{Emp} ait également un bon comportement par rapport au risque réel $R_{\text{Réel}}$, c'est-à-dire permette d'obtenir une bonne règle de décision.

3.1.2 Apprentissage bayésien

Fondamentalement, l'approche bayésienne se distingue de l'apprentissage direct d'une règle de décision par le fait qu'elle part d'une densité de probabilité supposée connue *a priori* $\mathsf{p}_{\mathcal{H}}$ sur l'espace des hypothèses $\mathcal{H}$ (aussi appelée *prior*) et qu'elle retourne une densité de probabilité *a posteriori* sur $\mathcal{H}$, $\mathsf{p}_{\mathcal{H}|\mathcal{Z}^m}$, après observation de l'échantillon d'apprentissage. Typiquement, il est attendu que la distribution de probabilité sur $\mathcal{H}$ présente des pics plus marqués après observation de l'échantillon d'apprentissage.

Soit $\mathcal{S}_m$ l'échantillon d'apprentissage de taille m, dans lequel on adoptera par souci de simplification la notation : $\mathbf{x}^m = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ et $u^m = \{u_1, \ldots, u_m\}$. On a ainsi, d'après la formule de Bayes :

$$\forall h \in \mathcal{H} : \quad \mathsf{p}_{\mathcal{H}|\mathcal{Z}^m = \mathcal{S}_m}(h) = \frac{\mathsf{p}_{\mathcal{Z}^m|\mathcal{H}=h}(\mathcal{S}_m)\, \mathsf{p}_{\mathcal{H}}(h)}{\mathbb{E}_{\mathcal{H}}\left[\mathsf{p}_{\mathcal{Z}^m|\mathcal{H}=h}(\mathcal{S}_m)\right]} = \frac{\overbrace{\mathsf{p}_{\mathcal{Y}^m|\mathcal{X}^m=\mathbf{x}^m,\mathcal{H}=h}(u^m)}^{\text{vraisemblance de } h}\ \overbrace{\mathsf{p}_{\mathcal{H}}(h)}^{\text{prior sur } h}}{\underbrace{\mathbb{E}_{\mathcal{H}}\left[\mathsf{p}_{\mathcal{Y}^m|\mathcal{X}^m=\mathbf{x}^m,\mathcal{H}=h}(u^m)\right]}_{\text{vraisemblance du modèle } \mathcal{H}}} \tag{2.16}$$

où on a utilisé le fait que $\mathsf{p}_{\mathcal{Z}^m|\mathcal{H}=h}(\mathcal{S}_m) = \mathsf{p}_{\mathcal{Y}^m|\mathcal{X}^m=\mathbf{x}^m,\mathcal{H}=h}(u^m)\, \mathsf{p}_{\mathcal{X}^m}(\mathbf{x}^m)$ puisque les hypothèses h influencent la génération des étiquettes $u^m \in \mathcal{Y}^m$ mais pas les formes $\mathbf{x}^m \in \mathcal{X}^m$.

L'équation dans son ensemble quantifie la probabilité *a posteriori* d'une hypothèse h après observation de l'échantillon de données $\mathcal{S}_m \in \mathcal{Z}^m$. La collection des formes présentes dans cet échantillon est notée $\mathbf{x}^m \in \mathcal{X}^m$ et la collection des étiquettes associées est notée $u^m \in \mathcal{Y}^m$. Notons que, dans le cas de la classification, on peut assimiler les hypothèses aux différentes classes possibles u_i, d'où les formules analogues de la section 2.1.

Copyright © 2010 Eyrolles.

Conformément au théorème de Bayes, cette probabilité *a posteriori* de h est proportionnelle à la probabilité *a priori* de h multipliée par la vraisemblance des données connaissant l'hypothèse h (première égalité ci-dessus).

Nous allons examiner tour à tour les trois termes intervenant dans cette expression. Leur sens est beaucoup plus simple que ce que peut suggérer leurs expressions formelles.

Vraisemblance de h

La vraisemblance de h exprime combien il est probable d'observer la séquence d'étiquettes u^m connaissant la séquence de formes x^m et la dépendance supposée h. Plus formellement, la vraisemblance de h est définie comme suit.

Définition 2.6 (Vraisemblance $\mathcal{L}(h, \mathrm{x})$)

Étant donnée un ensemble de densités de probabilité $\mathsf{p}_{\mathcal{Y}|\mathcal{X}=\mathrm{x},\mathcal{H}=h}$ *définies sur l'espace* $\mathcal{Y}$ *et une observation* $\mathbf{z} = (\mathrm{x}, y) \in \mathcal{Z}$, *on appelle* vraisemblance *de* h *la fonction* $\mathcal{L} : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$ *définie par :*

$$\mathcal{L}(h, \mathbf{z}) \stackrel{def}{=} \mathsf{p}_{\mathcal{Y}|\mathcal{X}=\mathrm{x},\mathcal{H}=h}(y) \tag{2.17}$$

c'est-à-dire la probabilité d'observer l'étiquette y *sous la densité de probabilité* $\mathsf{p}_{\mathcal{Y}|\mathcal{X}=\mathrm{x},\mathcal{H}=h}$

Par extension, on définit la vraisemblance d'une hypothèse étant donnée une séquence $\mathcal{L}(h, \mathbf{z}^m) = \mathsf{p}_{\mathcal{Y}^m|\mathcal{X}=\mathrm{x}^m,\mathcal{H}=h}$.

En tenant compte de l'indépendance supposée des exemples, cette vraisemblance s'exprime par :

$$\mathcal{L}(h, \mathbf{z}^m) = \mathsf{p}_{\mathcal{Y}^m|\mathcal{X}=\mathrm{x}^m,\mathcal{H}=h} = \prod_{i=1}^{m} \mathsf{p}_{\mathcal{Y}|\mathcal{X}=\mathrm{x}_i,\mathcal{H}=h}(u_i) \tag{2.18}$$

En passant par le logarithme[6] de cette expression, on obtient la *log-vraisemblance* qui est une somme.

$$\mathcal{LL}(h, \mathbf{z}^m) = \sum_{i=1}^{m} \log\left(\mathsf{p}_{\mathcal{Y}|\mathcal{X}=\mathrm{x}_i,\mathcal{H}=h}(u_i)\right) \tag{2.19}$$

Le prior ou probabilité *a priori* de h

La forme de la distribution $\mathsf{p}_{\mathcal{H}}$ est ce qui permet d'exprimer la connaissance *a priori* dans l'apprentissage bayésien. On prend souvent une distribution uniforme sur $\mathcal{X}$ pour exprimer l'ignorance complète avant observation de données[7].

Vraisemblance du modèle $\mathcal{H}$

Le dénominateur de l'équation (2.16) exprime la probabilité de la séquence d'étiquettes u^m étant données une séquence de formes x^m et toutes les hypothèses $h \in \mathcal{H}$ pondérées par leur probabilité *a priori* $\mathsf{p}_{\mathcal{H}}(h)$.

$$\mathbb{E}_{\mathcal{H}}\left[\mathsf{p}_{\mathcal{Y}^m|\mathcal{X}^m=\mathrm{x}^m,\mathcal{H}=h}(u^m)\right] = \int_{h\in\mathcal{H}} \mathsf{p}_{\mathcal{Y}^m|\mathcal{X}^m=\mathrm{x}^m,\mathcal{H}=h}(u^m)\, \mathsf{p}_{\mathcal{H}}(h)\, dh \tag{2.20}$$

Cette grandeur peut aussi être considérée comme une mesure de probabilité caractérisant l'espace des hypothèses $\mathcal{H}$ pour l'échantillon d'apprentissage $\mathcal{S}_m$. Cela signifie que pour chaque classe d'hypothèses $\mathcal{H}$, certaines séquences d'étiquettes y^m (pour la séquence de formes x^m) sont probables et d'autres moins. La classe d'hypothèses choisie *a priori* favorise donc certaines

[6] Qui est une fonction croissante monotone, et donc ne change pas l'argument donnant le maximum de la fonction.

[7] Qui est un cas particulier du principe d'entropie maximale défendu par Edwin Jaynes (1922-1998), voir [Jay03].

Copyright © 2010 Eyrolles.

classifications par rapport à d'autres. Cela conduit à considérer le problème de sélection de la classe d'hypothèses adéquate pour un certain problème. On parle aussi de *sélection de modèle*.

Un autre calcul de type bayésien, pour lequel nous reportons le lecteur à ([Her02], pp.78-79), montre que le meilleur modèle, ou classe d'hypothèses, est celui dont la vraisemblance est la plus grande après observation des données. Nous aurons l'occasion de revenir à plusieurs reprises sur ce problème fondamental pour l'apprentissage.

3.1.3 Principe de compression maximale d'information

À côté du principe de minimisation du risque empirique et de l'approche bayésienne sur l'apprentissage, un troisième point de vue considère que la meilleure description des données, conduisant à la plus petite erreur possible en généralisation, est celle qui minimise leur coût de description (souvent mesuré en bits). Une traduction de ce principe inductif est le principe de compression maximale de l'information ou encore de longueur minimale de description.

Définition 2.7 (Principe de longueur minimale de description, *MDLP*)

La meilleure théorie, ou hypothèse, ou le meilleur modèle, rendant compte d'un échantillon d'apprentissage minimise la somme de deux quantités :

1. *la longueur, mesurée en bits, de la* description de la théorie *;*
2. *la longueur, mesurée en bits, de la* description des données *lorsqu'elles sont* décrites à l'aide de la théorie.

Formellement, cela signifie que, selon ce principe, l'hypothèse optimale h^* vérifie :

$$h^* = \underset{h \in \mathcal{H}}{\mathrm{ArgMin}} \left\{ K(h) + K(\mathcal{S}_m | h) \right\} \tag{2.21}$$

où $K(h)$ mesure la longueur de description de h, et $K(\mathcal{S}_m|h)$ mesure la longueur de description des données $\mathcal{S}_m$ en utilisant l'hypothèse h (la théorie) pour les coder.

La parenté avec le principe bayésien est patente. En effet, cette dernière s'écrit (equation 2.16) :

$$\mathbf{p}_{\mathcal{H}|\mathcal{Z}^m=\mathcal{S}_m}(h) = \frac{\mathbf{p}_{\mathcal{Z}^m|\mathcal{H}=h}(\mathcal{S}_m)\, \mathbf{p}_{\mathcal{H}}(h)}{\mathbb{E}_{\mathcal{H}}\left[\mathbf{p}_{\mathcal{Z}^m|\mathcal{H}=h}(\mathcal{S}_m)\right]}$$

Soit en prenant l'opposé du logarithme de cette expression :

$$-\log \mathbf{p}_{\mathcal{H}|\mathcal{Z}^m=\mathcal{S}_m}(h) = -\log \mathbf{p}_{\mathcal{H}}(h) - \log \mathbf{p}_{\mathcal{Z}^m|\mathcal{H}=h}(\mathcal{S}_m) + \log \mathbb{E}_{\mathcal{H}}\left[\mathbf{p}_{\mathcal{Z}^m|\mathcal{H}=h}(\mathcal{S}_m)\right]$$

Comme le dernier terme à droite ne joue pas sur le choix de h, on a :

$$h^* = \underset{h \in \mathcal{H}}{\mathrm{ArgMin}} \left\{ -\log \mathbf{p}_{\mathcal{H}}(h) - \log \mathbf{p}_{\mathcal{Z}^m|\mathcal{H}=h}(\mathcal{S}_m) \right\}$$

Cette équation est formellement similaire à l'équation 21.20. Ce n'est pas une coïncidence. Nous renvoyons le lecteur au chapitre 21 et à l'annexe 11 pour une description plus détaillée de ce principe de compression maximale d'information et pour découvrir le lien profond qui le relie au point de vue bayésien.

Copyright © 2010 Eyrolles.

3.2 Les inférences bayésiennes

Comme il a été exposé, l'essence de l'approche bayésienne est de traiter la distribution sur l'espace $\mathcal{H}$ des hypothèses comme une variable aléatoire. Pour que cela ne soit pas un abus de langage, on suppose que l'espace des hypothèses peut être décrit par une famille paramétrée (par exemple par le vecteur θ) de distributions de probabilité. Le problème de l'inférence devient alors celui du calcul de la distribution *a posteriori* sur $\mathcal{H}$ à la lumière de l'information apportée par des données.

L'espoir est que les données permettent de concentrer la distribution de probabilité $\mathsf{p}(\theta)$ autour d'une valeur alors que cette distribution était plus « étalée » avant la prise en compte des données, reflétant notre ignorance préalable sur l'état de la nature (voir figure 2.7).

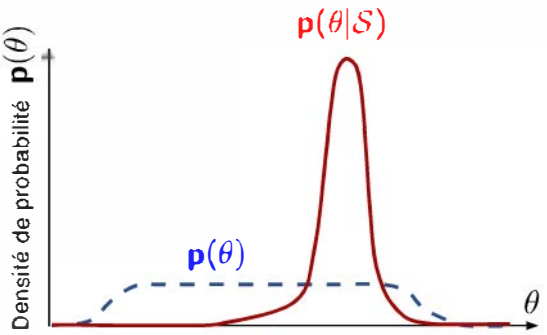


FIG. 2.7: *Révision de la distribution de probabilité sur les hypothèses par inférence bayésienne à partir d'un échantillon de données. Généralement, on observe une concentration de la distribution autour d'une valeur de θ, ce que l'on peut traduire par l'émergence d'un sous-ensemble d'hypothèses devenues beaucoup plus probables.*

Avant prise en compte des données, l'incertitude sur l'état du monde est traduite par la distribution *a priori* $\mathsf{p}(\theta)$. Par exemple, toutes les hypothèses ont même probabilité *a priori* $\mathsf{p}(\theta) = C^{\text{te}}$. Les données fournissent une information en modifiant la distribution sur θ grâce à la formule de révision des probabilités due au révérend Thomas Bayes (1702-1761).

Théorème 2.1 (Règle de Bayes)

$$\mathsf{p}(\theta|\mathcal{S}) = \frac{\mathsf{p}(\mathcal{S}|\theta)\,\mathsf{p}(\theta)}{\mathsf{p}(\mathcal{S})} = \frac{\mathsf{p}(\mathcal{S}|\theta)\,\mathsf{p}(\theta)}{\int \mathsf{p}(\mathcal{S}|\psi)\,\mathsf{p}(\psi)\,d\psi} \tag{2.22}$$

Pour un jeu de données $\mathcal{S}$ fixé, le dénominateur de l'équation 2.22 est constant (on y fait varier ψ, un vecteur de paramètres, sur toutes les valeurs possibles), d'où la formule de proportionnalité souvent utilisée :

$$\mathsf{p}(\theta|\mathcal{S}) \propto \mathsf{p}(\mathcal{S}|\theta)\,\mathsf{p}(\theta) \tag{2.23}$$

La distribution *a posteriori* de θ étant donné l'échantillon $\mathcal{S}$ est donc proportionnelle au produit de la distribution *a priori* $\mathsf{p}(\theta)$ et de la vraisemblance $\mathsf{p}(\mathcal{S}|\theta)$.

Lorsque la connaissance *a priori* sur l'état de la nature est faible, on considérera donc une distribution « étalée » $\mathsf{p}(\theta)$ (par exemple, une loi normale à grande variance). Plus l'échantillon de données sera important, plus son influence dominera dans la mise à jour de la distribution *a posteriori*, et moins sera influente la distribution *a priori*.

Copyright © 2010 Eyrolles.

Il est notable, par ailleurs, que les équations 2.22 et 2.23 permettent une mise à jour séquentielle de la distribution sur θ. Ainsi, si la distribution a été mise à jour après la prise en compte de $\mathcal{S}_1$, elle peut-être à nouveau modifiée en tenant compte de $\mathcal{S}_2$:

$$\mathbf{p}(\theta|\mathcal{S}_1, \mathcal{S}_2) \propto \mathbf{p}(\mathcal{S}_2|\theta)\,\mathbf{p}(\mathcal{S}_1|\theta)\,\mathbf{p}(\theta) \tag{2.24}$$

De plus, cette mise à jour séquentielle, pratique lorsque les données sont en grand nombre ou bien arrivent séquentiellement, fournit un résultat indépendant de l'ordre de prise en compte des sous-échantillons (en supposant ceux-ci indépendants conditionnellement étant donné le modèle sous-jacent $\mathbf{p}$).

Notons pour terminer que le dénominateur de l'équation 2.22, ou *distribution prédictive de* $\mathcal{S}$, $\mathbf{p}(\mathcal{S}) = \int \mathbf{p}(\mathcal{S}|\psi)\mathbf{p}(\psi)\,d\psi$ représente notre prédiction à propos de $\mathcal{S}$. Cette prédiction incorpore notre incertitude sur ψ grâce au prior $\mathbf{p}(\psi)$, et notre incertitude sur $\mathcal{S}$ quand ψ est connue, par $\mathbf{p}(\mathcal{S}|\psi)$. Cette distribution prédictive peut changer avec l'observation de nouvelles données. Si les données observées ont une faible probabilité selon la distribution prédictive, alors cela peut être le signe que la distribution $\mathbf{p}$ a été mal choisie.

Remarque (sur le choix de la distribution *a priori*)

La distribution *a priori* exprime à la fois notre incertitude sur le monde, plus la distribution est étalée, plus celle-ci est grande, et le type de modèle du monde que nous sommes prêts à considérer. Comme tel, ce choix est important. Il peut orienter le résultat obtenu et même conduire à des conclusions parfois gravement erronées si le type de modèle considéré ne permet pas d'exprimer le vrai modèle du monde.

Par ailleurs, les calculs de $\mathbf{p}(\theta|\mathcal{S})$ sont souvent difficiles et coûteux. Il est donc souhaitable de considérer des distributions de probabilité permettant des calculs simplifiés. C'est le cas notamment des *distributions conjuguées*. Celle-ci ont en effet la propriété que les distributions *a priori* et *a posteriori* appartiennent à la même famille de distributions. Les distributions exponentielles, beta ou normales, par exemple, ont cette propriété. Cela permet alors des mises à jour par adaptation des paramètres au lieu d'avoir à caractériser toute une distribution de manière complexe.

Jusqu'à récemment, le point de vue purement bayésien avait surtout une importance philosophique, mais était de peu de portée pratique. Cela est en train de changer rapidement avec l'arrivée de nouvelles techniques d'estimation stochastique fondées sur des échantillonnages aléatoires qui permettent d'estimer et d'étudier les distributions des paramètres à prédire. Ces méthodes incluent les *Markov Chain Monte-Carlo* (MCMC).

3.2.1 L'inférence bayésienne pure

L'inférence bayésienne pure consiste à spécifier la règle de décision optimale après observation des données. Elle ne vise pas, en revanche, à déterminer la meilleure hypothèse (ou la valeur optimale du paramètre θ). Dans ce cadre, la règle de décision prend en compte toute la connaissance sur l'incertitude liée à θ, c'est-à-dire toute la distribution *a posteriori* sur θ.

Par exemple, supposons que la tâche d'apprentissage consiste à prendre la décision $\mathbf{x}$, qui peut être de prédire la consommation électrique dans une heure, $\mathbf{x}$, à partir de $\mathcal{S}$, la connaissance de consommations électriques dans le passé. Au lieu de chercher d'abord le modèle h du monde le plus probable (ou le paramètre θ^* correspondant), l'inférence bayésienne pure consiste à calculer l'espérance de prédiction en prenant en compte toutes les prédictions possibles $\mathbf{p}(\mathbf{x}|\theta)$ pondérées par leur probabilité *a posteriori* $\mathbf{p}(\theta|\mathcal{S})$:

Copyright © 2010 Eyrolles.

Définition 2.8 (Règle d'inférence bayésienne)

La meilleure prédiction $\mathbf{x}$ *est celle qui maximise la probabilité :*

$$\begin{aligned} \mathbf{p}(\mathbf{x}|\mathcal{S}) &= \int \mathbf{p}(\mathbf{x}, \theta|\mathcal{S})\, d\theta \\ &= \int \mathbf{p}(\mathbf{x}|\theta)\ \mathbf{p}(\theta|\mathcal{S})\, d\theta \end{aligned} \tag{2.25}$$

la deuxième égalité provenant du fait que $\mathbf{x}$ est, par définition, conditionnellement indépendant des données $\mathcal{S}$ connaissant θ.

Cette formule, nécessitant le calcul effectif d'une intégrale, est en général très coûteuse à évaluer. C'est pourquoi les estimations ponctuelles (*pointwise estimation*), telles que la règle du *maximum de vraisemblance*, sont beaucoup plus employées. Cependant, l'approche bayésienne pure devient de plus en plus accessible alors que la puissance calcul disponible s'accroît.

3.2.2 La règle du maximum a posteriori (MAP)

La règle du *maximum a posteriori* (MAP) prend pour base la distribution *a posteriori* $\mathbf{p}(\theta|\mathcal{S})$ calculée par la règle de Bayes (équation 2.22). Mais, contrairement à l'inférence bayésienne, elle cherche d'abord une estimation du paramètre θ afin de prendre une décision.

Définition 2.9 (Principe du maximum a posteriori)

La valeur du paramètre θ *maximisant la probabilité a posteriori est choisie comme estimateur :*

$$\hat{\theta}_{MAP} = \underset{\theta}{\text{ArgMax}}\ \mathbf{p}(\theta|\mathcal{S}) \tag{2.26}$$

L'une des difficultés de cette méthode est qu'elle requiert la connaissance de la distribution *a priori* $\mathbf{p}(\theta)$. Plusieurs raisons militent pour ignorer ce terme. En particulier, on peut être tenté d'exprimer l'ignorance totale *a priori* sur le monde en posant que $\mathbf{p}(\theta)$ est uniforme (mais pourquoi uniforme selon l'échelle de θ et pas de θ^2 par exemple ?).

Si l'on décide d'ignorer effectivement ce terme, alors on obtient le principe du maximum de vraisemblance ci-dessous.

3.2.3 La règle du maximum de vraisemblance

Étant donné un échantillon $\mathcal{S} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$ de données (chaque $\mathbf{z}_i$ pouvant être une paire $(\mathbf{x}_i, u_i)$ dans le cas de l'apprentissage supervisé), indépendamment tirées selon une distribution paramétrée dont on suppose connue la forme $\mathbf{p}(\mathbf{x}|\theta)$. Le problème d'inférence consiste à utiliser l'information associée à $\mathcal{S}$ pour trouver une bonne estimation du paramètre inconnu θ.

La *fonction de vraisemblance* $\mathcal{L}(\theta|\mathcal{S})$ est la probabilité que les données $\mathcal{S}$ aient été produites par la distribution associée à la valeur θ : $\mathbf{p}(\cdot|\theta)$, c'est-à-dire $\mathbf{p}(\mathcal{S}|\theta)$. Ici, contrairement à l'approche bayésienne pure, ce sont donc les données qui sont supposées variables tandis que l'état du monde, représenté par θ, est supposé fixe.

Copyright © 2010 Eyrolles.

Si les données sont supposées indépendantes et identiquement distribuées (*i.i.d.*), alors :

$$\begin{aligned}\mathcal{L}(\theta|\mathcal{S}) &= \mathcal{L}(\theta|\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m) \\ &= \mathbf{p}(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m|\theta) = \prod_{i=1}^{m} \mathbf{p}(\mathbf{z}_i|\theta)\end{aligned} \tag{2.27}$$

La notion de vraisemblance des données est centrale pour de nombreuses techniques d'inférence statistique. Il faut noter que l'hypothèse de tirage i.i.d. n'est pas nécessaire pour définir une vraisemblance. Par exemple, si les données présentent une dépendance de Markov d'ordre 1 (telle que $\mathbf{z}_i$ dépend de $\mathbf{z}_{i-1}$), alors on pourrait définir la vraisemblance à l'aide de produits du type $\mathbf{p}(\mathbf{z}_i|\mathbf{z}_{i-1}, \theta)$.

Définition 2.10 (Principe du maximum de vraisemblance)

La valeur du paramètre θ pour laquelle la vraisemblance des données est maximale est appelée estimation du maximum de vraisemblance.

$$\hat{\theta}_{MLE} = \underset{\theta}{\text{ArgMax}}\, \mathcal{L}(\theta|\mathcal{S}) \tag{2.28}$$

En anglais, c'est le Maximum Likelihood Estimator *(MLE)*

Intuitivement, cette estimation correspond à la valeur du paramètre θ qui, en un sens, s'accorde le plus aux données observées.

En pratique, c'est la *log-vraisemblance* qui est généralement utilisée, en lieu et place de la vraisemblance, pour résoudre ce problème d'estimation car elle permet de remplacer une multiplication qui peut être délicate (avec des termes de valeurs éventuellement très proches de 0) par une addition.

$$\mathcal{LL}(\theta|\mathcal{S}) = \sum_{i=1}^{m} \log(\mathbf{p}(\mathbf{z}_i|\theta)) \tag{2.29}$$

et $\hat{\theta}_{\text{MLE}} = \text{ArgMax}_\theta\, \mathcal{LL}(\theta|\mathcal{S})$

Pour des problèmes simples, le calcul différentiel peut être utilisé pour résoudre cette équation. Pour des problèmes plus complexes, on cherche une solution à l'équation :

$$\frac{\partial \mathcal{L}(\theta|\mathcal{S})}{\partial \theta} = 0 \tag{2.30}$$

ou, de manière équivalente à : $\frac{\partial \mathcal{LL}(\theta|\mathcal{S})}{\partial \theta} = 0$

Lorsque les distributions manipulées ne sont pas simples (i.e. essentiellement la loi normale), il faut avoir recours à des techniques d'optimisation itératives, ou bien à des techniques d'expectation-maximisation (EM) pour certaines formes paramétriques de la fonction de vraisemblance (voir chapitre 15).

3.2.4 Discussion

Le point de vue bayésien consiste à expliciter l'incertitude sous la forme à la fois de l'incertitude sur le *modèle*, la famille de distribution $\mathbf{p}_\Theta$ choisie, et de l'incertitude sur la valeur des paramètres estimés θ (il s'agit souvent d'un vecteur et non d'un seul paramètre).

Dans l'approche du maximum de vraisemblance, une estimation ponctuelle du paramètre θ est souvent considérée comme l'objectif principal. Une analyse bayésienne complète (ou pure) s'attache, au contraire, à calculer toute la distribution *a posteriori* $\mathbf{p}(\theta|\mathcal{S})$ (et, éventuellement

Copyright © 2010 Eyrolles.

aussi sur la famille de distribution $\mathbf{p}_\bullet$ elle-même). La décision bayésienne consiste alors à calculer une moyenne pondérée par la probabilité de toutes les valeurs possibles du (ou des) paramètre(s) (voire des modèles envisagés). En principe, cette moyenne pondérée produit des prédictions plus précises que les estimations ponctuelles, telles que celle réalisée par le maximum de vraisemblance. En pratique, malheureusement, l'estimation de toute la distribution de probabilité est un problème difficile, particulièrement lorsque les données sont décrites en grande dimension. De plus, une prédiction utilisant une moyenne pondérée est moins susceptible d'être interprétable que lorsqu'une hypothèse unique est produite, comme c'est le cas pour les estimations ponctuelles retournant une valeur du paramètre θ.

Il est utile de noter que le principe du maximum de vraisemblance et l'apprentissage bayésien ne se prêtent pas aux mêmes méthodes de calcul. Le premier se traite comme un problème d'optimisation : il faut chercher le minimum d'une fonction d'erreur. En revanche, dans le second, l'essentiel du calcul implique une intégration sur des espaces de grandes dimensions. Dans ce dernier cas, les méthodes classiques d'intégration ne conviennent pas, et il faut se tourner vers des méthodes approchées, par exemple les méthodes de Monte-Carlo (voir le chapitre 3).

3.3 Minimisation du risque empirique : exemples de traductions

3.3.1 Cas de l'apprentissage pour classer

Dans le cas d'un problème d'apprentissage de concept, ou encore de classification binaire, la fonction de perte la plus simple est :

$$l(u_i, h(\mathbf{x}_i)) = \begin{cases} 0 & \text{si } u_i = h(\mathbf{x}_i) \\ 1 & \text{si } u_i \neq h(\mathbf{x}_i) \end{cases}$$

Le risque $R_{Réel}(h)$ mesure alors la probabilité de mauvaise classification.

Notons $m_{err.}(h)$ le nombre d'exemples de l'ensemble d'apprentissage $\mathcal{S}$ qui sont mal classés par une certaine hypothèse h choisie dans $\mathcal{H}$. Le risque empirique (ou taux d'erreur apparent) de h se mesure alors simplement par :

$$R_{Emp}(h) = \frac{m_{err.}(h)}{m}$$

Dans le **cas d'une classification à C classes** avec $C \neq 2$, on a souvent recours à la matrice de confusion.

Définition 2.1 (Matrice de confusion)

La matrice de confusion empirique $M_{emp}(i,j)$ *d'une règle de classification h est une matrice $C \times C$ dont l'élément générique donne le nombre d'exemples de l'ensemble d'apprentissage $\mathcal{S}$ de la classe i qui ont été classés dans la classe j.*

La somme des termes non diagonaux divisée par la taille m de l'ensemble d'apprentissage n'est autre que le risque empirique (ou erreur apparente) de la règle de classification.

— EXEMPLE **Matrice de confusion à deux classes** —

Soit une tâche de classification à deux classes, alors la matrice de confusion peut être écrite comme :

Estimé \ Réel	+	−
+	**VP**	FP
−	FN	**VN**

Copyright © 2010 Eyrolles.

où les éléments de la matrice peuvent être :

- Soit des *pourcentages* : **VP** correspond alors au taux de `Vrais positifs`, c'est-à-dire à la proportion d'exemples positifs qui sont classés comme positifs par la règle de classification. De même, **VN** correspond au taux de `Vrais Négatifs`, FP aux `Faux Positifs` et FN aux `Faux Négatifs`.
  On a **VP** + FN = 100 % (des exemples positifs) et FP + **VN** = 100 % (des négatifs).
  La règle de classification est parfaite sur les exemples si **VP** = **VN** = 100%.
- Soit des *nombres* : **VP** correspond alors au nombre de `Vrais positifs`, c'est-à-dire d'exemples positifs qui sont classés comme positifs par la règle de classification. De même, **VN** correspond au nombre `Vrais Négatifs`, FP aux `Faux Positifs` et FN aux `Faux Négatifs`.
  On a **VP** + FN = m^+ (nombre des exemples positifs) et FP + **VN** = m^-.
  L'erreur est alors : $\frac{FP+FN}{m}$.

La matrice de confusion se généralise directement au cas de plus de deux classes.

3.3.2 Cas de la régression

La régression consiste à estimer une fonction f à valeurs réelles, connaissant un échantillon fini de couples $(\mathbf{x}, u = f(\mathbf{x}))$ ou $(\mathbf{x}, u = f(\mathbf{x} + bruit))$.

La fonction f à estimer peut donc être considérée comme la somme d'une fonction déterministe et d'un signal d'erreur aléatoire de moyenne nulle (et le plus souvent considéré comme une gaussienne).

$$u = f(\mathbf{x}) + \epsilon \tag{2.31}$$

On peut aussi décrire ce phénomène en considérant que la fonction déterministe est la moyenne de la probabilité conditionnelle sur l'espace de sortie $\mathcal{U}$.

$$f(\mathbf{x}) = \int u\ p(u|\mathbf{x})\ du \tag{2.32}$$

L'espace des fonctions hypothèse $\mathcal{H}$ de l'apprenant peut ou non inclure l'espace des fonctions cible $\mathcal{F}$. Une fonction de perte usuelle pour la régression est la fonction erreur quadratique (L_2) :

$$L(u_i, h(\mathbf{x}_i)) = (u_i - h(\mathbf{x}_i))^2 \tag{2.33}$$

L'apprentissage consiste alors à trouver la fonction $h \in \mathcal{H}$ minimisant la fonctionnelle de risque :

$$R_{Réel}(h) = \int_{\mathcal{Z}=\mathcal{X}\times\mathcal{U}} (u - h(\mathbf{x}))^2 dF(\mathbf{x}, u) \tag{2.34}$$

sur la seule base de l'échantillon d'apprentissage. Cette fonctionnelle, le *risque réel*, mesure la précision des prédictions de l'apprenant.

Remarque

Sous l'hypothèse que le signal d'erreur est une gaussienne centrée en 0, ce risque peut aussi être écrit en fonction de la capacité de l'apprenant à approximer la fonction cible $f(\mathbf{x})$ (et non la sortie u), comme le montre le calcul suivant :

$$\begin{aligned} R_{Réel}(h) &= \int (u - f(\mathbf{x}) + f(\mathbf{x}) - h(\mathbf{x}))^2\ p(\mathbf{x}, u)\ d\mathbf{x}du \\ &= \int (u - f(\mathbf{x}))^2\ d\mathbf{x}du + \int (h(\mathbf{x}) - f(\mathbf{x}))^2\ p(\mathbf{x})\ d\mathbf{x} \\ &\quad + 2\int (u - f(\mathbf{x}))(f(\mathbf{x}) - h(\mathbf{x}))\ p(\mathbf{x}, u)\ d\mathbf{x}du \end{aligned} \tag{2.35}$$

Copyright © 2010 Eyrolles.

Sous l'hypothèse que le bruit est de moyenne nulle, le dernier terme dans la somme ci-dessus s'écrit :

$$\begin{aligned}\int (u - f(\mathrm{x}))(f(\mathrm{x}) - h(\mathrm{x}))\, p(\mathrm{x},u)\, d\mathrm{x}du &= \int (\epsilon(f(\mathrm{x}) - h(\mathrm{x}))\, p(u|\mathrm{x})\, p(\mathrm{x})\, d\mathrm{x}du \\ &= \int (f(\mathrm{x}) - h(\mathrm{x}))\left[\int \epsilon p(u|\mathrm{x})du\right] p(\mathrm{x})\, d\mathrm{x} \\ &= \int (f(\mathrm{x}) - h(\mathrm{x}))E - \epsilon(\epsilon|\mathrm{x})\, p(\mathrm{x})\, d\mathrm{x} \quad = \quad 0\end{aligned} \tag{2.36}$$

Le risque peut donc être récrit comme :

$$R_{Réel}(h) = \int (u - f(\mathrm{x}))^2\, p(\mathrm{x}, \mathrm{u})\, d\mathrm{x}du + \int (h(\mathrm{x}) - f(\mathrm{x}))^2\, p(\mathrm{x})\, d\mathrm{x} \tag{2.37}$$

Le premier terme ne dépend pas de la fonction d'approximation h et peut être écrit en terme de la variance sur le bruit :

$$\begin{aligned}\int (u - f(\mathrm{x}))^2 p(\mathrm{x}, u)\, d\mathrm{x}du &= \int \epsilon^2 p(u|\mathrm{x})\, p(\mathrm{x})\, d\mathrm{x}du \\ &= \int \left[\int \epsilon^2 p(u|\mathrm{x})du\right] p(\mathrm{x})\, d\mathrm{x} \\ &= \int E_\epsilon(\epsilon^2|\mathrm{x})\, p(\mathrm{x})\, d\mathrm{x}\end{aligned} \tag{2.38}$$

Et substituant (22.9) dans (22.8), on obtient :

$$R_{Réel}(h) = \int E_\epsilon(\epsilon^2|\mathrm{x})\, p(\mathrm{x})\, d\mathrm{x} + \int (h(\mathrm{x}) - f(\mathrm{x}))^2\, p(\mathrm{x})\, d\mathrm{x} \tag{2.39}$$

Le risque pour le problème de régression (sous l'hypothèse de fonction de perte L_2 et de bruit de moyenne nulle) présente donc une contribution exprimant la variance du bruit et une contribution exprimant la précision de la fonction d'approximation. Comme la variance du bruit est indépendante de la fonction d'approximation, la minimisation du second terme de (22.10) est équivalente à la minimisation du risque (22.5). Ainsi, chercher à obtenir le risque de prédiction optimal est équivalent à chercher l'approximation la plus précise de la fonction cible inconnue f.

3.3.3 Cas de l'estimation de densité

Un autre problème inductif important consiste à estimer une densité de probabilité dans l'espace d'entrée $\mathcal{X}$ à partir d'un échantillon de données $\{x_i\}_{1 \leq i \leq m}$. Dans ce cas, il n'y a pas nécessité de considérer un espace de sortie, et la sortie $h(\mathrm{x})$ de l'apprenant représente une densité sur $\mathcal{X}$.

La fonction de perte usuelle dans ce cas est la fonction :

$$l(h(\mathrm{x})) = -\ln h(\mathrm{x}) \tag{2.40}$$

donnant la fonctionnelle de risque :

$$R_{Réel}(h) = \int -\ln h(\mathrm{x})\, p(\mathrm{x}) d\mathrm{x} \tag{2.41}$$

Il est établi [DHS01] que la densité optimale h^* minimise cette fonctionnelle de risque. Par ailleurs, si la densité cible $f \notin \mathcal{H}$, alors on peut montrer que la solution h minimisant l'espérance de risque ou risque réel est caractérisable : c'est celle dont la *divergence de Kullback-Leibler* avec la vraie densité f est la plus faible (voir la définition de la divergence de Kullback-Leibler dans les chapitres 3, 20 et 21).

Copyright © 2010 Eyrolles.

3.3.4 Cas de l'apprentissage de fonction de tri (*ranking*)

Plusieurs mesures de performances ont été proposées, spécialement pour la tâche de recherche d'information (« *information retrieval* »).

- Le *Pair-Wise Error Count* représente le nombre de paires d'éléments qui sont mal classées par la règle de tri par rapport au nombre total de paires.
- Le *Mean Reciprocal Rank (MRR)* s'applique à la tâche de retourner des éléments « pertinents » et « non pertinents » pour chaque requête (e.g. des documents). Soit r_i le rang de l'élément jugé le plus pertinent pour une requête i, alors la mesure du rang réciproque pour cette requête est $1/r_i$. Le MRR est le rang réciproque moyen pour toutes les requêtes effectuées.
- Le *Winner Takes All (WTA)*. S'applique au même type de tâche. Si l'élément jugé le plus pertinent par la règle de tri est effectivement pertinente, le compte WTA pour cette requête vaut 0 ; sinon il prend pour valeur 1. Le WTA général est ici aussi la moyenne des WTA sur l'ensemble des requêtes effectuées.
- Le *Normalized Discounted Cumulative Gain measure (NDCG)* est une mesure cumulative de qualité de tri (et donc le coût correspondant vaut $1 - NDCG$). Pour une requête donnée i, le NDCG vaut : $\mathcal{N}_i = N_i \sum_{j=1}^{L} (2^{r(j)} - 1)/\log(1+j)$ où $r(j)$ est le rang donné par la fonction de tri à l'élément mis au rang j, et où une constante de normalisation N_i est choisie de telle manière qu'un ordre parfait donnerait $\mathcal{N}_i = 1$. L est le niveau de tri auquel le NCDG est calculé. Le score général est à nouveau obtenu par la moyenne sur l'ensemble des requêtes.

3.3.5 Cas de la prédiction en ligne

Lors de la prédiction en ligne, la tâche de l'apprenant est de faire une prédiction $h(\mathbf{x}_{t+i})$ (avec $i \geq 1$) à chaque pas de temps t à partir d'une mémoire des données passées et, éventuellement, des erreurs commises. On peut également être plus exigeant et lui demander d'expliciter l'hypothèse courante h_t qu'il utilise pour réaliser sa prédiction.

Une différence avec l'apprentissage hors ligne à partir d'un échantillon d'apprentissage est que la fonction cible peut changer avec le temps : on parle de *dérive de concept*. Dans ce cas, une approche intéressante est de pondérer les exemples passés pour donner un poids plus important aux exemples récents, sensés mieux représenter la fonction cible courante, et « oublier » progressivement les exemples antérieurs.

Le critère inductif peut alors prendre la forme :

$$R_{Emp}(h_T) = \underset{h \in \mathcal{H}}{\text{ArgMin}} \left\{ \frac{1}{\sum_{t<T} \lambda(t)} \sum_{t<T} \lambda(t) \ell\big(u_t, h_t(\mathbf{x}_t)\big) \right\} \tag{2.42}$$

avec $0 \leq \lambda(t) < 1$ et $\sum_{t<T} \lambda(t) < \infty$.

Nous renvoyons à [Cor09] pour une étude plus détaillée des multiples critères inductifs envisageables.

4. Analyse du principe de *minimisation du risque empirique*

L'objectif de cette section est d'étudier les propriétés d'un apprentissage guidé par le principe de minimisation du risque empirique. Nous nous limiterons dans ce chapitre introductif à certains

Copyright © 2010 Eyrolles.

cas simples qui suffiront à illustrer quelques principes essentiels régissant l'apprentissage supervisé. D'autres classes de situations et d'autres outils d'analyse seront étudiés au fur et à mesure de la présentation de méthodes d'apprentissage, et dans le chapitre 21 d'approfondissement.

Ré-examinons d'abord les composantes d'un problème d'apprentissage.

1. La **première** définit simplement un *algorithme d'apprentissage* comme un système de calcul d'une sortie à partir de la donnée de certaines entrées.

Définition 2.11 (Algorithme d'apprentissage supervisé)

Un algorithme d'apprentissage $\mathcal{A}$ *est un algorithme qui, prenant en entrée : un espace de description* $\mathcal{X}$, *un espace de sortie* $\mathcal{Y}$ *et un espace de fonctions hypothèses* $\mathcal{H}$ *définies de* $\mathcal{X}$ *sur* $\mathcal{Y}$, *ainsi qu'un échantillon d'apprentissage* $\mathcal{S}_m = (\langle \mathbf{x}_i, u_i \rangle)_{1 \leq i \leq m}$, *retourne une hypothèse* $h \in \mathcal{H}$ *:*

$$\mathcal{A} : \bigcup_{m=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^m \rightarrow \mathcal{H}$$

Notons que cette définition ne dit encore rien d'un quelconque critère de performance à poursuivre. Par ailleurs, il n'est pas non plus précisé si l'algorithme retourne toujours, de manière déterministe, la même hypothèse pour un échantillon ou une séquence d'apprentissage donné. Nous rencontrerons les deux types d'apprentissage, déterministe ou non, dans la suite de cet ouvrage. En revanche, en apprentissage artificiel, on considère généralement des hypothèses h déterministes.

Il faut ici rappeler la différence avec la règle de Bayes. Celle-ci (voir équation 2.8) ne fait pas référence à un espace d'hypothèses $\mathcal{H}$ donné.

2. La **deuxième composante** concerne *le critère de performance* visé par l'apprentissage. Nous avons vu qu'un objectif raisonnable est de chercher une hypothèse h minimisant l'espérance de perte, c'est-à-dire le *risque réel* :

$$\begin{aligned} h^\star &= \operatorname*{ArgMin}_{h \in \mathcal{H}} R_{\text{Réel}}(h) \\ &= \operatorname*{ArgMin}_{h \in \mathcal{H}} \int_{\mathbf{x} \in \mathcal{X}, y \in \mathcal{Y}} \ell(h(\mathbf{x}), y)\ \mathsf{p}_{\mathcal{X}\mathcal{Y}}\, d\mathbf{x} dy \end{aligned} \tag{2.43}$$

Ce critère de performance, qui est défini comme une espérance faisant intervenir une distribution d'événements inconnue $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$, ne peut être directement visé. L'algorithme d'apprentissage répond à un échantillon $\mathcal{S}_m = (\langle \mathbf{x}_i, u_i \rangle)_{1 \leq i \leq m}$ de données supposé représentatif de $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$. Il est donc raisonnable d'attendre d'un bon algorithme d'apprentissage qu'il retourne une hypothèse $h^\star_{\mathcal{S}} \in \mathcal{H}$ qui s'approche d'autant plus d'une hypothèse optimale $h^\star \in \mathcal{H}$, au sens du risque réel, que la taille de l'échantillon s'accroît. Cette demande devant *a priori* être vérifiée pour toute distribution $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$, on parle de *consistance universelle* de l'apprentissage.

Définition 2.12 (Consistance universelle)

Un algorithme d'apprentissage $\mathcal{A}$ *a la propriété de* consistance universelle *si :*

$$\forall\, \mathsf{p}_{\mathcal{X}\mathcal{Y}} : \quad R_{\text{Réel}}(\mathcal{A}(\mathcal{S}_m)) = R_{\text{Réel}}(h^\star_{\mathcal{S}_m}) \underset{m \rightarrow \infty}{\longrightarrow} R_{\text{Réel}}(h^\star) \tag{2.44}$$

L'existence d'apprentissages universellement consistants, démontrée par Stone [Sto77] en 1977,

Copyright © 2010 Eyrolles.

9,12 mm

est un grand accomplissement de la théorie de l'apprentissage[8]. Cela garantit en effet que, étant donné un échantillon d'apprentissage assez grand, le classifieur appris, dans le cas de la classification, sera presque aussi performant que le classifieur optimal de Bayes. Malheureusement, il ne s'agit que d'un résultat asymptotique. La convergence vers la règle optimale peut être arbitrairement lente, en fonction de la distribution des exemples, et, pour tout échantillon d'apprentissage de taille finie, la différence de performance entre le classificur optimal et celle du classificur appris peut être proche du maximum possible (e.g. de $1/2$ dans le cas de la classification binaire). Concevoir de bons systèmes d'apprentissage n'est donc pas une chose triviale.

Nous souhaiterions être capables d'estimer le risque réel associé au choix de l'hypothèse $h^\star_{\mathcal{S}}$ et de le comparer au risque optimal $R^\star$ associé à la règle de décision optimale de Bayes (voir équation (2.8)). Cependant, $R_{\text{Réel}}(h^\star_{\mathcal{S}})$ est une variable aléatoire, puisqu'il dépend de l'échantillon d'apprentissage, et il ne peut pas être calculé à partir des données puisqu'il dépend de la distribution des données $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$ qui est inconnue. C'est pourquoi les estimations de $R_{\text{Réel}}(h^\star_{\mathcal{S}})$ prennent généralement la forme de bornes en probabilité. C'est-à-dire que l'on cherchera à montrer que, très probablement, sauf si l'on a tiré un échantillon peu représentatif de $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$, l'algorithme retournera une hypothèse dont la performance est proche de la performance de la meilleure hypothèse possible $h^\star \in \mathcal{H}$. On parle souvent d'*apprentissage Probablement Approximativement Correct* (apprentissage PAC) (en anglais, *Probably Approximatively Correct*, ou PAC-Learning).

Définition 2.13 (Apprentissage PAC (Probablement Approximativement Correct))

Soit $\mathcal{H}$ une classe de fonctions de $\mathcal{X} \to \{0,1\}$. Un algorithme d'apprentissage au sens PAC *pour $\mathcal{H}$ est une fonction :*

$$\mathcal{A} : \bigcup_{m=1}^{\infty} \{\mathcal{X},\mathcal{Y}\}^m \to \mathcal{H}$$

qui associe une hypothèse $h \in \mathcal{H}$ à tout échantillon d'apprentissage $\mathcal{S}_m$ avec les propriétés suivantes :

$\forall \varepsilon \in [0,1], \forall \delta \in [0,1], \exists$ une taille d'échantillon minimale $m_0(\varepsilon,\delta)$, telle que si $m \geq m_0(\varepsilon,\delta)$, alors, pour toute distribution $\mathsf{p}_{\mathcal{X}\times\mathcal{Y}}$, si $\mathcal{S}_m$ est un échantillon d'apprentissage tiré i.i.d. (indépendamment et identiquement distribué) suivant $\mathsf{p}^m_{\mathcal{X}\mathcal{Y}}$, on a :

$$P\left\{R_{\text{Réel}}(h^\star_{\mathcal{S}}) \leq R_{\text{Réel}}(h^\star) + \varepsilon\right\} \geq 1 - \delta$$

On dit que $\mathcal{H}$ est apprenable au sens PAC *s'il existe un algorithme d'apprentissage pour $\mathcal{H}$.*

En d'autres termes, on demande que l'algorithme d'apprentissage réussisse à retourner une hypothèse $h^\star_{\mathcal{S}}$ pas beaucoup plus mauvaise que $h^\star$, l'hypothèse optimale, pour tout choix d'erreur d'approximation ε (soit encore : $R_{\text{Réel}}(h^\star_{\mathcal{S}}) - R_{\text{Réel}}(h^\star) \leq \varepsilon$) et pour tout paramètre de confiance δ. Ce dernier est nécessaire car on ne peut pas exclure *a priori* un « mauvais tirage » de l'échantillon d'apprentissage. Il faut donc admettre une probabilité, que l'on veut contrôler, qu'il puisse arriver que l'échantillon soit mauvais. Il est à attendre que plus on est exigeant sur l'erreur d'approximation ε et sur le paramètre de confiance δ, et plus l'apprentissage va requérir une taille croissante de l'échantillon d'apprentissage, lui permettant d'obtenir davantage d'information sur la fonction cible et de diminuer la probabilité d'un mauvais tirage.

Il est important de noter que l'apprentissage PAC ne fait aucune hypothèse sur la distribution $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$ des exemples, du moment qu'elle est fixe. Il doit fonctionner quelle que soit cette probabilité sous-jacente inconnue. On dit qu'il s'agit d'un cadre *contre toute distribution.*

[8] Stone a montré par une preuve élégante qu'un système d'apprentissage particulier, la classification par les k-plus-proches-voisins, est universellement consistant.

Copyright © 2010 Eyrolles.

3. Jusqu'ici, nous n'avons pas encore précisé *le critère inductif* utilisé par l'algorithme pour sélectionner une hypothèse sur la base de l'échantillon d'apprentissage disponible. C'est la **troisième composante** de l'apprentissage. Dans cette section, nous analysons le critère de minimisation du risque empirique (MRE) qui, rappelons-le, prescrit de choisir une hypothèse minimisant le coût sur les exemples d'apprentissage.

Parce que l'on ne connaît pas le risque réel associé à chaque hypothèse, on a recours à un critère inductif qui remplace le critère objectif par un problème d'optimisation de substitution s'appuyant sur l'échantillon d'apprentissage disponible. Dans le cas du *principe de minimisation du risque empirique* (MRE), on cherche une hypothèse minimisant le *risque empirique* :

$$\begin{aligned} h^*_{\mathcal{S}} &= \underset{h\in\mathcal{H}}{\text{ArgMin}}\ R_{\text{Emp}}(h) \\ &= \underset{h\in\mathcal{H}}{\text{ArgMin}}\ \frac{1}{m}\sum_{i=1}^{m}\ell(h(\mathbf{x}_i,u_i)) \end{aligned} \tag{2.45}$$

Ce principe inductif ne sera pertinent que si le risque empirique est corrélé avec le risque réel. Son analyse doit donc s'attacher à étudier la **corrélation entre les deux risques** et plus particulièrement la corrélation entre le risque réel encouru avec l'hypothèse sélectionnée à l'aide du principe *MRE* : $R_{\text{Réel}}(h^*_{\mathcal{S}})$ et le risque réel optimal : $R_{\text{Réel}}(h^*)$.

Le principe inductif de minimisation du risque empirique est fondé sur une loi des grands nombres qui établit que, pour une fonction h *fixée*, on a :

$$\lim_{m\to\infty} R_{\text{Emp}}(h) = R_{\text{Réel}}(h) \tag{2.46}$$

si l'échantillon d'apprentissage $\mathcal{S}_m$ résulte de m tirages indépendants et identiquement distribués selon la distribution sous-jacente $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$.

4.1 La loi des grands nombres

La loi des grands nombres est l'une des lois les plus importantes en statistiques. Dans sa forme la plus simple, elle énonce que :

Théorème 2.2 (Loi des grands nombres)

Sous des conditions faibles, la moyenne de variables aléatoires ξ_i tirées de manière indépendante d'une loi sous-jacente de probabilité $\mathbf{p}$ *converge vers la moyenne de la distribution lorsque la taille de l'échantillon tend vers l'infini.*

$$\frac{1}{m}\sum_{i=1}^{m}\xi_i \to \mathbb{E}(\xi) \qquad \textit{pour } m\to\infty \tag{2.47}$$

En supposant que ξ est distribuée selon $\mathbf{p}$.

Il est possible d'appliquer ce théorème au cas du risque empirique et du risque réel. En effet, le risque empirique est défini comme la moyenne des pertes sur les points de l'échantillon d'apprentissage, lui-même supposé tiré de manière i.i.d. (indépendamment et identiquement distribuée) selon $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$, tandis que le risque réel est la moyenne de cette perte sur l'ensemble de la distribution.

Copyright © 2010 Eyrolles.

D'après la loi des grands nombres, nous pouvons conclure que, pour *une* hypothèse h *fixée*, le risque empirique converge vers le risque réel lorsque la taille de l'échantillon d'apprentissage tend vers l'infini.

$$R_{\text{Emp}}(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(h(\mathbf{x}_i), u_i) \rightarrow \mathbb{E}(\ell(\mathbf{x}, y) \quad \text{pour } m \rightarrow \infty \tag{2.48}$$

Ici, la perte $\ell(\mathbf{x}, y)$ joue le rôle de variable aléatoire. Cela signifie donc que l'on peut approcher le risque réel par la mesure du risque empirique[9].

Une inégalité due à Chernoff, et généralisée par Hoeffding, précise la qualité de cette approximation.

Théorème 2.3 (Inégalité de Hoeffding)

Si les ξ_i sont des variables aléatoires, tirées indépendamment et selon une même distribution et prenant leur valeur dans l'intervalle $[a, b]$, alors :

$$P\left(\left|\frac{1}{m}\sum_{i=1}^{m} \xi_i - \mathbb{E}(\xi)\right| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2\,m\,\varepsilon^2}{(b-a)^2}\right) \tag{2.49}$$

Ce théorème énonce que la probabilité que la moyenne empirique dévie de plus de ε de l'espérance est bornée par une très petite quantité, $2\exp(-\frac{2\,m\,\varepsilon^2}{(b-a)^2})$, qui, de surcroît, décroît très rapidement avec m.

Appliquée au risque empirique et au risque réel, cette inégalité nous donne :

$$P(|R_{\text{Emp}}(h) - R_{\text{Réel}}(h)| \geq \varepsilon) \leq 2\exp(-\frac{2\,m\,\varepsilon^2}{(b-a)^2}) \tag{2.50}$$

si la fonction de perte ℓ est définie sur l'intervalle $[a, b]$.

On serait donc tenté de supposer qu'il suffit de sélectionner une hypothèse optimale au sens du risque empirique pour obtenir, *ipso facto* une hypothèse optimale au sens du risque réel, ce qui est notre objectif. Ce n'est malheureusement pas vrai. La loi des grands nombres évoquée, et qui s'applique pour une hypothèse donnée, ne tient plus lorsque le choix de l'hypothèse se fait *a posteriori* sur la base du résultat du risque empirique.

Un petit exemple peut aider à percevoir intuitivement la base du problème.

— EXEMPLE **Une classe trop nombreuse et trop diversifiée** —

Supposons que la tâche d'un recruteur soit d'évaluer des étudiants pour le compte d'un employeur. Pour cela, il utilise un test à choix multiple contenant m questions et, très naturellement, il décide de recommander l'étudiant qui aura obtenu le meilleur score.

Si le test est bien conçu, et en supposant qu'il soit représentatif de la future tâche, alors, effectivement, la note d'un étudiant donné devrait tendre, avec le nombre de questions, vers sa vraie performance future, chez l'employeur.

[9] Plus généralement, ce qui nous intéressera est la convergence en probabilité (selon le tirage de l'échantillon d'apprentissage) d'une suite de variables aléatoires (le risque empirique dépendant de $\mathcal{S}$). C'est l'objet du *théorème central limite* et de ses variantes. Essentiellement, ce théorème affirme que toute somme de variables aléatoires indépendantes et identiquement distribuées (les coûts associés à chaque exemple d'apprentissage) tend vers une variable aléatoire gaussienne.

Copyright © 2010 Eyrolles.

Si le recruteur teste simultanément 10 étudiants, il est fort probable que l'étudiant obtenant la meilleure note au test se révèle également le meilleur employé à l'avenir. Mais supposons maintenant que le recruteur ait la possibilité de tester 10 millions de personnes, parmi lesquelles, sans qu'il le sache, se sont glissés quelques millions de singes qui tapent aléatoirement leurs réponses sur l'ordinateur. Est-il encore possible d'assurer que l'« étudiant » ayant obtenu le meilleur score sera effectivement le meilleur employé possible ?

On sent bien que, dans ce cas où un très grand nombre de personnes ont passé le test, la performance du meilleur étudiant, *a posteriori*, n'a plus forcément un rapport étroit avec la performance future.

Cet exemple, dans lequel les étudiants jouent le rôle d'hypothèses testées sur un échantillon de m questions, montre que, si l'on ne contraint pas, d'une manière ou d'une autre, la richesse de l'espace des hypothèses, il ne peut plus y avoir de garantie que l'hypothèse qui semble la meilleure soit effectivement celle qui aura la performance optimale à l'avenir. Plus encore, le lien entre la performance mesurée empiriquement et la vraie performance à venir se relâche considérablement (un singe peut parfaitement réussir le test, par chance, et, probablement, ne pas être très performant à l'avenir).

Exemple Le test multiple

Le *test d'hypothèse classique* consiste à estimer la probabilité qu'une hypothèse donnée (e.g. cette pièce est truquée, ce gène est lié à telle activité biologique) soit vraie en partant de l'observation d'un ensemble d'expériences. Une manière standard d'opérer est de recourir à une hypothèse nulle exprimant la probabilité des événements possibles (e.g. la proportion de piles sur m tirages est $x\%$) si l'hypothèse alternative est vraie (e.g. pièce non truquée, gène indépendant).

Par exemple, en supposant une pièce non truquée, l'espérance du nombre moyen de « pile » est : $\mathbb{E}[\#piles] = m \cdot \mathsf{P}(pile) = m/2$ pour m lancés de pièces, et la variance est : $Var(\#piles) = \sqrt{\mathsf{P}(pile) \cdot (1 - \mathsf{P}(pile)) \cdot m} = \frac{\sqrt{m}}{2}$. Ainsi, pour $m = 100$, la probabilité pour qu'une pièce tombe plus de 60 fois sur pile ou moins de 40 fois est inférieure à 5% (en approchant la loi binomiale par une loi normale).

Plus précisément, la probabilité que la pièce tombe du côté pile P fois sur m tirages est donnée par la *loi binomiale* :

$$\text{prob}(P) = \frac{m!}{P!(m-P)!} \cdot \mathsf{P}^P \cdot (1 - \mathsf{P})^{m-P}$$

Dans le cas où $m \cdot \mathsf{P} \cdot (1 - \mathsf{P}) > 5$, la loi binomiale peut être estimée par la *loi normale* de moyenne $\mu = \mathbb{E}[P]$ et de variance $\sigma = \sigma[P]$:

$$prob(P) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{P-\mu}{\sigma}\right)^2}$$

Le problème de *test multiple* se pose lorsque l'on teste simultanément plusieurs hypothèses (e.g. on teste 100 pièces en les lançant chacune m fois, on teste 30 000 gènes pour détecter ceux qui sont liés à l'obésité). Dans ce cas, on cherche à détecter les hypothèses vraies (e.g. telle pièce est truquée) et à limiter le risque de *faux positif* (signaler une pièce comme étant truquée alors qu'elle ne l'est pas).

Il s'agit d'un problème méritant analyse car cette fois-ci la sélection des hypothèses « vraies » s'effectue *a posteriori*, après les expériences.

Par exemple, si j'ai 10 000 pièces non truquées que je lance chacune 100 fois en l'air, je trouverai en moyenne 500 pièces tombant plus de 60 fois sur pile ou moins de 40 fois, même

Copyright © 2010 Eyrolles.

si toutes les pièces sont non truquées. De fait, la probabilité qu'aucune pièce ne paraisse truquée est quasi nulle, à savoir $(0.95)^{10000}$.
À nouveau, le comportement d'une pièce sélectionnée comme truquée, n'est pas nécessairement représentatif de sa vraie nature, et il faut modifier les critères de sélection pour limiter le risque de faux positif (voir par exemple la notion de p-valeur et le test de Bonferroni, l'un des plus simples au sein d'une littérature devenant très abondante sur le sujet).

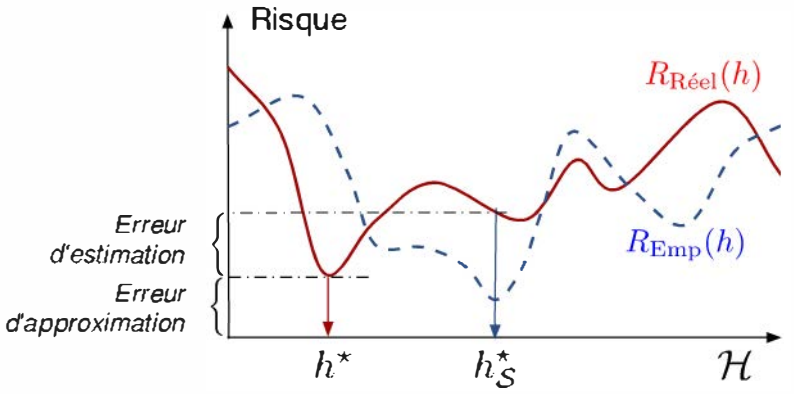


FIG. 2.8: *L'axe horizontal correspond à l'espace des fonctions considérées par l'apprenant. Les deux courbes représentent le risque réel, d'une part, et le risque empirique, d'autre part. Le risque réel est fixe, alors que la courbe du risque empirique varie en fonction de l'échantillon d'apprentissage. L'hypothèse la meilleure selon le risque empirique $h^\star_{\mathcal{S}}$ peut être très différente de l'hypothèse optimale $h^\star$. On a également figuré l'erreur d'approximation, due au choix de $\mathcal{H}$, et l'erreur d'estimation, correspondant à l'écart entre la vraie performance de $h^\star_{\mathcal{S}}$ et celle de $h^\star$.*

Vapnik [Vap95] définit la pertinence[10], souhaitable, du principe de minimisation du risque empirique comme étant une double convergence.

Définition 2.14 (Pertinence du principe *MRE*)

On dit que le principe MRE *est* pertinent *si le risque réel inconnu $R_{Réel}(h^\star_{\mathcal{S}})$ et le risque empirique $R_{Emp}(h^\star_{\mathcal{S}})$ convergent vers la* même *limite optimale $R_{Réel}(h^\star)$ lorsque la taille m de l'échantillon tend vers ∞ (voir la figure 2.9).*

$$\begin{aligned} R_{Réel}(h^\star_{\mathcal{S}}) &\underset{m\to\infty}{\longrightarrow} R_{Réel}(h^\star) \\ R_{Emp}(h^\star_{\mathcal{S}}) &\underset{m\to\infty}{\longrightarrow} R_{Réel}(h^\star) \end{aligned} \tag{2.51}$$

Notons bien que nous avons affaire ici à une *suite* de fonctions $h^\star_{\mathcal{S}_m}$ dépendante de la taille m

[10] « Consistance » est le terme généralement employé, repris directement de l'anglais. Dans le cas présent, ce terme n'évoque rien en français et fait un double emploi malheureux avec le terme d'apprenant « consistant » employé parfois pour décrire un apprenant qui cherche une hypothèse dont les prédictions sur tous les exemples de l'échantillon d'apprentissage sont en accord avec les réponses fournies par l'oracle. Le terme d'apprenant consistant est aussi employé pour décrire un apprenant dont l'erreur réelle tend vers l'erreur optimale de Bayes lorsque la taille de l'échantillon m tend vers l'infini (voir [DEV96]). Nous préférons introduire un nouveau terme, celui de *pertinence* qui traduit bien ce que l'on cherche : la validité du principe inductif *ERM*.

Copyright © 2010 Eyrolles.

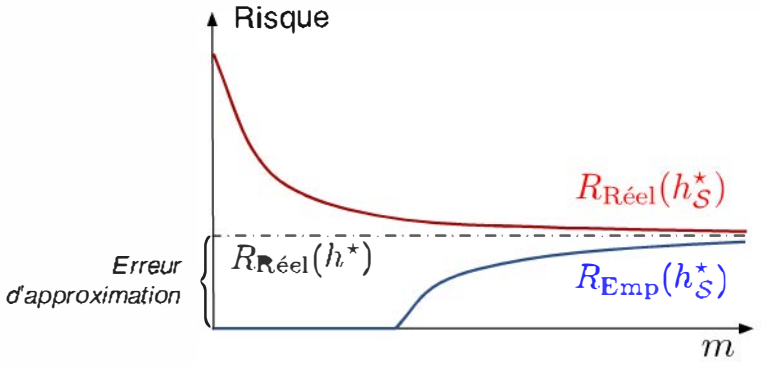


FIG. 2.9: Pertinence du principe *MRE*. *À cause du biais ou erreur d'approximation, le risque réel optimal atteint pour l'hypothèse $h^\star$ n'est généralement pas égal à zéro. Au fur et à mesure que la taille m de l'échantillon croît, le risque réel associé à l'hypothèse courante sélectionnée $\hat{h}_{\mathcal{S}_m}$ diminue et tend vers le risque optimal. Le risque empirique est inférieur au risque réel puisque $\hat{h}_{\mathcal{S}_m}$ est sélectionnée pour minimiser le premier et non le second. Souvent, il arrive que pour de petits échantillons de données, on trouve une hypothèse dans $\mathcal{H}$ dont le risque empirique est nul. Cela devient généralement impossible lorsque la taille m de l'échantillon croît.*

de l'échantillon d'apprentissage $\mathcal{S}_m$.

Dans le cas de cette double convergence, en effet, il est justifié d'utiliser le principe inductif *MRE* pour choisir une hypothèse à partir de l'observation d'un échantillon de données.

4.2 Le compromis biais-variance

Un algorithme d'induction sélectionne donc une hypothèse $h^\star_{\mathcal{S}}$ minimisant le critère inductif sur la base d'un échantillon d'apprentissage $\mathcal{S}$. On peut décomposer l'erreur commise par rapport au risque $R^\star$ de la décision optimale de Bayes (section 1.2.1) de la manière suivante :

$$R_{\text{Réel}}(h^\star_{\mathcal{S}}) - R^\star = \underbrace{\left[R_{\text{Réel}}(h^\star_{\mathcal{S}}) - R_{\text{Réel}}(h^\star)\right]}_{\text{Erreur d'estimation}} + \underbrace{\left[R_{\text{Réel}}(h^\star) - R^\star\right]}_{\text{Erreur d'approximation}} \tag{2.52}$$

Le second terme, appelé *erreur d'approximation*, correspond au biais introduit par le choix de l'espace d'hypothèses $\mathcal{H}$. Ce choix étant fait, on ne peut faire mieux que de trouver $h^\star$ la (ou une) meilleure hypothèse dans $\mathcal{H}$, au sens du risque réel. Le premier terme, appelé *erreur d'estimation*, est une quantité aléatoire qui dépend de l'échantillon d'apprentissage, et mesure la proximité entre l'hypothèse retenue par l'algorithme sur la base de l'échantillon $h^\star_{\mathcal{S}}$ et la meilleure hypothèse possible dans $\mathcal{H}$: $h^\star$ (voir figure 2.8 et 2.10). Dans le cadre d'une analyse statistique, les erreurs d'approximation et d'estimation sont respectivement appelées *biais* et *variance*[11].

L'objectif étant de diminuer l'écart entre le risque réel de l'hypothèse sélectionnée $h^\star_{\mathcal{S}}$ et le risque optimal $R^\star$, il semble naturel de chercher à minimiser à la fois les erreurs d'approximation et d'estimation. Malheureusement, ces deux erreurs varient généralement en sens inverse l'une de l'autre. En effet, afin de diminuer l'erreur d'approximation, il faudrait être mieux informé sur le bon choix de l'espace d'hypothèses $\mathcal{H}$. À défaut d'une telle connaissance *a priori*, on peut diminuer

[11] Plus précisément, le biais et la variance en statistique ont été définis à l'origine dans le cadre de la régression et pour la fonction de perte quadratique. Mais ces termes sont maintenant utilisés plus largement, et correspondent respectivement à l'erreur d'approximation et à l'erreur d'estimation.

Copyright © 2010 Eyrolles.

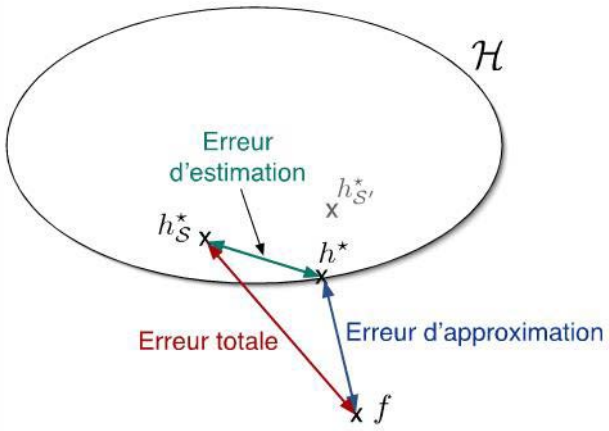


FIG. 2.10: Les sources d'erreur. *L'erreur commise en sélectionnant l'hypothèse $h^\star_S$ au lieu de la fonction optimale f (associée au risque $R^\star$) peut se décomposer en une erreur d'approximation due au choix d'un espace d'hypothèses $\mathcal{H}$, et une erreur d'estimation, due au caractère aléatoire de l'échantillon d'apprentissage $\mathcal{S}$. Ainsi, si l'échantillon avait été $\mathcal{S}'$ au lieu de $\mathcal{S}$, l'hypothèse sélectionnée aurait été $h^\star_{S'}$.*

l'erreur d'approximation en choisissant un espace d'hypothèses plus large. Cependant, cela a le plus souvent pour contrepartie d'augmenter l'erreur d'estimation, c'est-à-dire la variabilité de l'hypothèse $h^\star_S$ en fonction de l'échantillon $\mathcal{S}$. C'est le fameux **compromis biais-variance** des statisticiens.

L'estimation de l'erreur d'approximation, cruciale, est malheureusement difficile à réaliser car elle requiert une connaissance des régularités cibles, qui sont justement essentiellement inconnues. Il est établi que, pour des algorithmes consistants et capables d'ajuster l'espace d'hypothèses $\mathcal{H}$ avec l'échantillon d'apprentissage[12] la convergence vers 0 de l'erreur d'approximation peut être arbitrairement lente en l'absence de présupposés sur les régularités cibles. En revanche, les vitesses de convergence de l'erreur d'estimation peuvent être calculées même sans ces présupposés. Nous nous concentrons donc dans la suite **sur l'erreur d'estimation**.

4.3 La consistance du MRE dans le cas de $\mathcal{H}$ fini

Maintenant, supposons que l'on puisse borner la différence $|R_{\text{Réel}}(h) - R_{\text{Emp}}(h)|$ *pour toutes les hypothèses* de $\mathcal{H}$ simultanément :

$$\exists \varepsilon > 0, \forall h \in \mathcal{H} : \quad |R_{\text{Réel}}(h) - R_{\text{Emp}}(h)| \leq \varepsilon \tag{2.53}$$

soit encore :

$$\sup_{h \in \mathcal{H}} |R_{\text{Réel}}(h) - R_{\text{Emp}}(h)| \leq \varepsilon \tag{2.54}$$

où sup désigne le supremum (les lecteurs non familiers avec cette notion peuvent penser au maximum) (précisément $||\cdot||_\infty$ indique la norme supremum. ($\|x\|_\infty := \sup_{s\in[0,1]} x(s)$)).

[12] Comme c'est le cas d'algorithmes réalisant la minimisation du risque structurel ou utilisant des termes de régularisation (voir plus loin et le chapitre 21).

Copyright © 2010 Eyrolles.

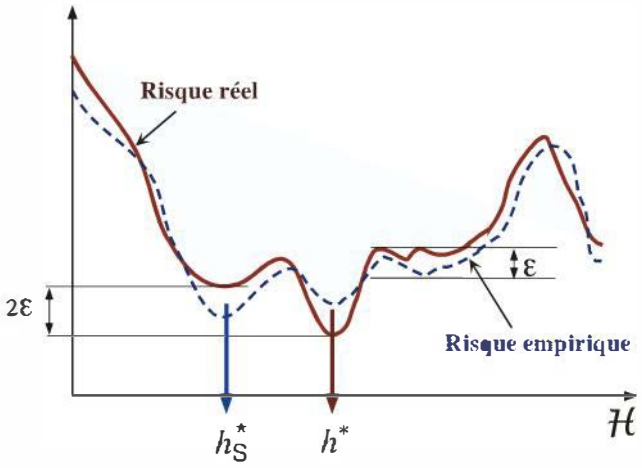


FIG. 2.11: *Si chaque hypothèse $h \in \mathcal{H}$ a un risque empirique proche de son risque réel (à moins de ε), alors minimiser le risque empirique (en application du principe inductif* MRE*) minimisera également approximativement le risque réel (à au plus 2ε).*

Alors, il est facile d'établir que $R_{\text{Réel}}(h^\star) - R_{\text{Réel}}(h_S^\star) < 2\,\varepsilon$ (voir figure 2.11). En effet :

$$\begin{aligned} R_{\text{Réel}}(h_S^\star) - R_{\text{Réel}}(h^\star) &= R_{\text{Réel}}(h_S^\star) - R_{\text{Emp}}(h_S^\star) + R_{\text{Emp}}(h_S^\star) - R_{\text{Réel}}(h^\star) \\ &\leq R_{\text{Réel}}(h_S^\star) - R_{\text{Emp}}(h_S^\star) + R_{\text{Emp}}(h^\star) - R_{\text{Réel}}(h^\star) \\ &\qquad \text{(Puisque par hypothèse } R_{\text{Emp}}(h_S^\star) < R_{\text{Emp}}(h^\star)) \\ &\leq 2\,\varepsilon \end{aligned}$$

Il s'agit là d'une condition suffisante, et elle est considérable puisqu'elle requiert la convergence du risque empirique sur le risque réel pour toutes les hypothèses simultanément (en fait pour un même échantillon $\mathcal{S}$), c'est-à-dire une *convergence uniforme*, et ce indépendamment de la distribution de probabilité sous-jacente $\mathbf{p}_{\mathcal{X}\cdot\mathcal{Y}}$.

On ne peut évidemment assurer l'inégalité 2.53 qu'en probabilité, puisqu'elle dépend du tirage de l'échantillon d'apprentissage. On cherche en fait les conditions permettant d'obtenir :

$$\forall\, 0 \leq \varepsilon, \forall \delta \leq 1 \;:\quad P^m(|R_{\text{Réel}}(h_S^\star) - R_{\text{Réel}}(h^\star)| \geq \varepsilon) < \delta \tag{2.55}$$

Ou encore les conditions de **convergence uniforme** sur l'espace des hypothèses $\mathcal{H}$:

$$P^m(\sup_{h \in \mathcal{H}} |R_{\text{Réel}}(h_S^\star) - R_{\text{Réel}}(h^\star)| \geq \varepsilon) \;\to\; 0 \quad \text{quand } m \to \infty \tag{2.56}$$

Pour cela, on borne la probabilité de l'événement correspondant à un tirage de $\mathcal{S}$ qui est tel que la différence entre la performance apparente $R_{\text{Emp}}(h^i)$ d'une hypothèse h^i au moins et sa vraie performance $R_{\text{Réel}}(h^i)$ soit plus grande que ε.

Soit l'ensemble des « mauvais » échantillons $\mathcal{S}$, ceux pour lesquels on se trompe en croyant que la performance empirique de la fonction $h^i \in \mathcal{H}$ est à moins de ε de sa performance vraie :

$$C_i \;=\; \{\{(\mathbf{x}_1, u_1), \ldots, (\mathbf{x}_m, u_m)\} : R_{\text{Réel}}(h^i) - R_{\text{Emp}}(h^i) > \varepsilon\} \tag{2.57}$$

D'après l'inégalité de Hoeffding, et pour une taille d'échantillon m assez grande, on a :

$$P^m[C_i] \;\leq\; \delta \tag{2.58}$$

Copyright © 2010 Eyrolles.

En considérant maintenant l'ensemble, supposé fini, des fonctions de $\mathcal{H}$, on a, d'après la borne sur l'union d'événements :

$$P^m[C_1 \cup \ldots \cup C_{|\mathcal{H}|}] \leq \sum_{i=1}^{|\mathcal{H}|} P^m[C_i] \tag{2.59}$$

C'est-à-dire que l'on est capable de borner la probabilité qu'il existe au moins une fonction h^i pour laquelle la performance empirique est trompeuse. Ainsi :

$$\begin{aligned} P^m[\exists h \in \mathcal{H} : R_{\text{Réel}}(h) - R_{\text{Emp}}(h) > \varepsilon] &\leq \sum_{i=1}^{|\mathcal{H}|} P^m[R_{\text{Réel}}(h^i) - R_{\text{Emp}}(h^i) > \varepsilon] \\ &\leq |\mathcal{H}| \exp(-2m\varepsilon^2) = \delta \end{aligned} \tag{2.60}$$

en supposant ici que la fonction de perte ℓ prend ses valeurs dans l'intervalle $[0, 1]$.

D'où l'on tire facilement que : $\varepsilon = \sqrt{\frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{2m}}$, c'est-à-dire que :

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m\left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \sqrt{\frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{2m}}\right] > 1 - \delta \tag{2.61}$$

On notera que la différence essentielle avec la borne de Hoeffding, valable pour une hypothèse fixée, réside dans le terme supplémentaire $|\mathcal{H}|$ à droite. C'est ce qui traduit que nous voulons que la borne tienne pour toutes les fonctions de $\mathcal{H}$ simultanément.

Par ailleurs, $\log|\mathcal{H}|$ est aussi le nombre de bits nécessaires pour spécifier une fonction h particulière dans $\mathcal{H}$. On retrouve là un lien intéressant, et non fortuit, avec la théorie de l'information et du codage (voir par exemple [vLBS04]).

Terminons en soulignant qu'un des résultats les plus extraordinaires de l'analyse de Vapnik et Chervonenkis est que la convergence uniforme dans $\mathcal{H}$ du risque empirique sur le risque réel est non seulement une condition suffisante, mais est aussi une condition nécessaire ! (Voir par exemple [VC71, DGL96, Men03]).

4.4 Le cas où $\mathcal{H}$ contient la fonction cible

Supposons maintenant que l'espace des hypothèses $\mathcal{H}$ est suffisamment bien informé pour contenir la fonction cible f. Cela signifie que l'erreur d'approximation est nulle. Il est alors possible de recourir à un autre calcul que celui de la borne de Hoeffding pour calculer la dépendance de l'écart entre risque empirique et risque réel en fonction de la taille m de l'échantillon d'apprentissage.

À tout instant, il existe au moins une hypothèse de risque empirique nul (ne serait-ce que la fonction cible f). Quelle est la probabilité qu'une hypothèse de risque réel $> \varepsilon$ paraisse bonne, c'est-à-dire soit de risque empirique nul ?

Nous supposerons ici que la fonction de perte l compte le nombre d'erreurs de classification[13] (voir l'annexe 1) :

$$l(h(\mathbf{x}_i), y_i) = \begin{cases} 0 & \text{si } y_i = h(\mathbf{x}_i) \\ 1 & \text{si } y_i \neq h(\mathbf{x}_i) \end{cases} \tag{2.62}$$

[13] Attention ! ! Même si ce que nous allons dire dans cette section peut se généraliser à d'autres fonctions de perte, le détail de la démonstration dépend de cette hypothèse, et ne peut donc se transposer sans précautions à d'autres fonctions de perte.

Copyright © 2010 Eyrolles.

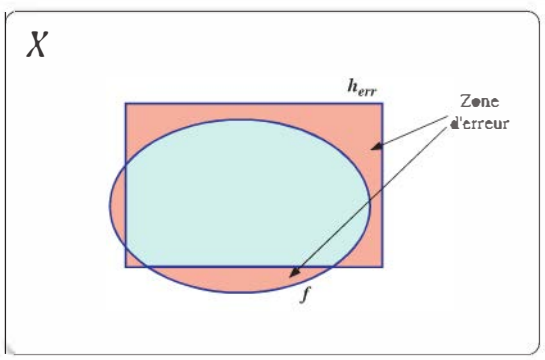


FIG. 2.12: *La zone d'erreur dans le cas de l'apprentissage d'un concept ou fonction binaire définie sur* $\mathcal{X}$.

Dans ce cas, le risque réel est égal à la probabilité qu'un exemple tombe dans la zone d'erreur entre la fonction cible f et la fonction hypothèse erronée h_{err} : $h_{err} \Delta f$ (le symbole Δ dénote la différence symétrique entre deux ensembles, voir figure 2.12).

$$R_{\text{Réel}}(h_{err}) = P_{\mathcal{X}}(h_{err} \Delta f) \geq \varepsilon \tag{2.63}$$

La probabilité qu'après l'observation d'un exemple on ne s'aperçoive pas que h_{err} est erronée est de $1 - \varepsilon$. Après l'observation d'un échantillon i.i.d. suivant la distribution $\mathcal{D}_{\mathcal{X}}$ de m exemples, la probabilité de « survie » de h_{err} vaut donc $(1 - \varepsilon)^m$.

En considérant maintenant l'ensemble $\mathcal{H}$ des hypothèses possibles, la probabilité que l'une d'entre elles « survive » après l'observation de $\mathcal{S}$ est bornée par : $|\mathcal{H}|(1 - \varepsilon)^m$ (on effectue ici une sommation car on a affaire à une union d'événements disjoints). On sait, par développement limité, que $|\mathcal{H}|(1 - \varepsilon)^m < |\mathcal{H}|e^{-\varepsilon m}$. En reprenant l'inéquation 2.55, il suffit donc d'avoir un échantillon de taille m telle que :

$$m \geq \frac{1}{\varepsilon} \ln \frac{|\mathcal{H}|}{\delta} \tag{2.64}$$

pour que l'erreur commise en choisissant l'hypothèse $h^*_{\mathcal{S}}$ minimisant le risque empirique soit bornée par ε avec une probabilité $> 1 - \delta$.

Ici, l'écart ε entre le risque réel et le risque empirique d'une hypothèse h varie en : $\varepsilon = \frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m}$, d'où :

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m} \right] > 1 - \delta \tag{2.65}$$

On note la différence avec l'équation (2.61) pour le cas général où $\mathcal{H} \neq \mathcal{F}$:

> Lorsque $\mathcal{H} = \mathcal{F}$, **la convergence est beaucoup plus rapide** puisqu'elle se fait en $\mathcal{O}(1/m)$ au lieu de $\mathcal{O}(\sqrt{1/m})$.

De fait, Vapnik [Vap82](théorème 6.8) a montré que l'exposant sur m varie continûment entre -1 et -1/2 de manière croissante en fonction de l'erreur minimale possible $\inf_{h \in \mathcal{H}} R_{\text{Réel}}(h)$. Cela peut s'expliquer de manière qualitative. Dans le cas où la fonction cible appartient à l'espace des hypothèses, cela signifie que la distribution des exemples dans $\mathcal{X} \times \mathcal{Y}$ peut être « représentée » par une fonction de $\mathcal{H}$. De ce fait, la variance sur les hypothèses $h^*_{\mathcal{S}}$ qui minimisent le risque empirique

Copyright © 2010 Eyrolles.

en fonction de l'échantillon d'apprentissage $\mathcal{S}$ est réduite. Or moins de données sont nécessaires pour approcher une variable aléatoire dont la variance est moindre. Il est donc avantageux de choisir un espace d'hypothèses $\mathcal{H}$ de faible erreur d'approximation.

Plus généralement, la vitesse de convergence est lente dans les cas où la région autour de la règle de décision optimale de Bayes est grande. Il faut en effet plus de temps, et d'informations, pour déterminer alors cette région.

4.5 Quelques leçons partielles

On retiendra des démonstrations précédentes trois idées :

1. D'abord que la cardinalité de $\mathcal{H}$, donc en un certain sens sa richesse, a un effet direct sur la borne d'erreur. Il est déjà apparent que le choix d'un ensemble $\mathcal{H}$ trop riche peut conduire à de mauvaises inductions.
2. Ensuite, le raisonnement utilisé dans la démonstration implique l'*ensemble* des fonctions hypothèse de $\mathcal{H}$. Nous verrons qu'une généralisation de ce raisonnement fait appel de même à un argument de *convergence uniforme*. Cette observation est très importante car elle indique que l'analyse est *de facto* une *analyse dans le pire cas*, s'appliquant en particulier à l'hypothèse pour laquelle la convergence est la plus mauvaise.
3. Finalement, l'idée directrice de la démonstration consiste à borner la probabilité qu'une zone d'erreur de poids $> \varepsilon$ ne soit pas atteinte par un exemple de l'échantillon d'apprentissage[14].

Sans avoir rendu compte de l'analyse beaucoup plus complète de Vapnik (décrite dans le chapitre 21), nous pouvons retenir à ce stade que le principe inductif de minimisation du risque empirique ne peut être appliqué sans précaution. Pour que la mesure du risque empirique soit corrélée avec le risque réel, il faut que l'espace d'hypothèses $\mathcal{H}$ dans lequel on choisit $h^*_{\mathcal{S}}$ ait de bonnes propriétés. De manière informelle, il faut que cet espace ne soit pas trop « riche » ou trop « souple », c'est-à-dire qu'on ne puisse pas y trouver des hypothèses s'accordant à n'importe quel jeu de données. On retrouve naturellement une idée déjà rencontrée avec le compromis biais-variance. Cela signifie que le principe *MRE* doit être modifié pour que la richesse de $\mathcal{H}$ soit également prise en compte lorsque l'on recherche la meilleure hypothèse. Toutes les techniques de contrôle de l'espace d'hypothèses visent à régler ce compromis.

Remarque (Comparaison avec l'analyse statistique classique)

L'analyse statistique classique des années 1960 et 1970 insistait sur la convergence des mesures de probabilité pour étudier les propriétés de l'apprentissage. De fait, il s'agissait d'étudier la distance dans un espace de modèles $\mathcal{F}$ (famille de distributions de probabilité avec un certain nombre de paramètres associés) entre la distribution apprise $\mathcal{A}(\mathcal{S})$ et la vraie distribution $f \in \mathcal{F}$. C'est ce que l'on peut appeler le *cadre de l'identification*.

Malheureusement, cette analyse bute sur la malédiction de la dimensionnalité. Une bonne approximation de la densité de probabilité nécessite un échantillon d'apprentissage dont la taille dépend exponentiellement de la dimension de l'espace des modèles. Elle suppose de plus que la famille de distributions de probabilité considérée *a priori* est correcte.

En revanche, l'analyse de convergence de risque, $R_{\text{Réel}}(\hat{h}_{\mathcal{S}})$ vers $R_{\text{Réel}}(h^*)$, que l'on pourrait appeler *cadre de l'imitation*, initiée par Vapnik et par les pionniers du cadre PAC, donne des bornes indépendantes de la dimension de l'espace $\mathcal{X}$ et de l'espace des hypothèses $\mathcal{H}$ (le nombre d'exemples peut ainsi même être inférieur au nombre de dimensions) et ne fait aucune hypothèse sur l'adéquation du choix initial de modèle comme nous allons le voir dans la suite.

[14] Une généralisation de cette idée a été utilisée dans le cas d'espaces de fonctions indicatrices de cardinalité infinie avec la notion de ε-réseau (ε-*net*)[Hau92].

Copyright © 2010 Eyrolles.

4.6 Vers un principe MRE contrôlé

L'analyse du principe de minimisation du risque empirique (MRE) montre qu'il ne peut pas être appliqué aveuglément, et qu'il faut contrôler l'espace des hypothèses considérées par l'algorithme d'apprentissage.

Cela nécessite :

1. de savoir *mesurer la richesse* ou encore la capacité de cet espace d'hypothèses ;
2. de savoir *modifier le critère MRE* pour prendre en compte cette richesse.

Plusieurs **mesures de richesse** ont été proposées, incluant des mesures :

- fondées sur le nombre de paramètres en jeu pour la définition des hypothèses (e.g. critères AIC ou BIC) ;
- résumant en un nombre la capacité à pouvoir trouver des hypothèses dans $\mathcal{H}$ rendant compte de n'importe quel échantillon d'apprentissage (e.g. dimension de Vapnik-Chervonenkis, complexité de Rademacher) ;
- caractérisant la diversité des fonctions de $\mathcal{H}$ (e.g. nombre de couverture) ;
- estimant la complexité algorithmique de description d'une hypothèse.

Les **approches pour modifier le MRE** sont elles-mêmes diverses :

- approche de *sélection automatique d'espace d'hypothèses* ou sélection automatique de modèles (e.g. Le SRM de Vapnik) ;
- approche de la *régularisation* qui ajoute un terme de pénalisation de l'hypothèse considérée dans le critère MRE ;
- approche dite de la félicité (*luckiness framework*) dans laquelle on tient compte de l'adéquation entre les hypothèses considérées et la distribution des exemples telle qu'elle est trahie par l'échantillon disponible (e.g. méthodes de maximisation de la marge) ;
- approches *prenant en compte l'algorithme d'apprentissage* et en particulier le sous-espace qu'il explore effectivement dans $\mathcal{H}$ (e.g. méthode par compression, par stabilité du risque empirique, par prise en compte de l'imperfection de l'optimisation, etc.) ;
- approches par *minimisation de la longueur de description* des données (e.g. MDLP) ;
- approches *bayésiennes* qui tendent à ajuster automatiquement la complexité des hypothèses sélectionnées.

Le problème du contrôle du principe MRE est encore l'objet de recherches actives. Afin de ne pas alourdir le présent chapitre qui vise d'abord à présenter les concepts de base et à faire comprendre les grands problèmes en jeu, **nous décrivons de manière plus détaillée ces approches et méthodes**, après la présentation de nombreuses méthodes d'apprentissage, **dans le chapitre 21**.

5. Le lien entre le passé et le futur et le *no-free-lunch theorem*

Nous avons présenté plusieurs principes inductifs et leurs variations nées de l'étude des conditions de leur validité. Ainsi, ont été passés en revue le principe *MRE* favorisant les hypothèses qui s'accordent le mieux aux données d'apprentissage, le principe bayésien stipulant (dans sa version maximum de vraisemblance) de choisir l'hypothèse dont il est le plus probable qu'elle soit à l'origine des données, le principe de compression d'information prescrivant de choisir le

Copyright © 2010 Eyrolles.

modèle du monde conduisant à sa description la plus compacte. Nous avons également vu que l'étude théorique de ces principes avait conduit à des principes inductifs plus sophistiqués dans lesquels la richesse de l'espace d'hypothèses est prise en compte. Les recherches récentes, portant en particulier sur les séparateurs à vastes marges (SVM) raffinent ces principes en prescrivant de prendre en compte aussi la distribution des exemples. Une question naturelle est alors de se demander lequel de ces principes inductifs est le meilleur ; lequel nous devrions choisir.

5.1 Le *no-free-lunch* theorem : toutes les méthodes se valent !

Le chapitre 1 a déjà apporté des éléments de réponse à cette question en insistant sur la nécessité d'un biais d'apprentissage pour permettre l'induction, c'est-à-dire d'hypothèses *a priori* sur le monde. Un théorème formalise et généralise cette idée : le *no-free-lunch theorem* dû à Wolpert (1992) [Wol92a]. Selon ce théorème, tous les principes inductifs, et tous les algorithmes d'apprentissage se valent. En l'absence de toute information sur le problème d'apprentissage autre que l'échantillon de données, aucune méthode n'est meilleure qu'une autre, y compris celle qui consiste à tirer une hypothèse au hasard. Exprimé d'une autre manière, ce théorème affirme qu'il n'y a *a priori* aucune corrélation entre l'échantillon de données $\mathcal{S}$ observé et les événements non encore observés. De ce fait, toute hypothèse sélectionnée sur la base de $\mathcal{S}$ n'a aucune raison d'être performante à l'avenir en dehors de $\mathcal{S}$. De manière plus abrupte, en dehors d'information supplémentaire sur le problème d'apprentissage, c'est-à-dire sur l'espace des fonctions cible, il n'est pas possible de réaliser autre chose que de l'apprentissage par cœur ! Aucune induction n'est possible, ou, du moins, plus légitime qu'une autre, par exemple de tirer les étiquettes des exemples non vus au hasard.

Avant d'examiner une expression plus formelle de ce théorème, essayons d'en saisir l'intuition. Soit l'espace $\mathcal{F}$ des fonctions cible. Soit $\mathcal{X}$ l'espace des entrées, et soit $\mathcal{U}$ l'espace des sorties. On suppose qu'un échantillon de formes $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ est tiré aléatoirement suivant une distribution $d_{\mathcal{X}}$ inconnue sur $\mathcal{X}$. Chacune de ces formes est étiquetée pour former l'échantillon $\mathcal{S} = \langle(\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), \ldots, (\mathbf{x}_m, u_m)\rangle$. On suppose ici que cet échantillon n'est pas bruité. Les étiquettes u_i ont donc été calculées grâce à l'application d'une fonction $f \in \mathcal{F}$. Le problème de l'induction est d'estimer laquelle sur la base de l'échantillon $\mathcal{S}$.

En l'absence d'informations supplémentaires sur $\mathcal{F}$, toutes les fonctions $f \in \mathcal{F}$ sont également possibles. Une fois fixé l'échantillon d'apprentissage, un certain nombre de ces fonctions sont éliminées car ne s'accordant pas aux données, mais toutes les autres fonctions restent candidates, et aucune prédiction n'est possible. C'est ce que nous avons vu dans le cas de fonctions binaires dans le chapitre 1. C'est également ce que montre la figure 21.1.

Si donc l'unique information dont nous disposons pour une tâche inductive est un échantillon d'apprentissage, alors seul un apprentissage par cœur de cet échantillon est possible, et aucune induction ne peut être effectuée avec quelque garantie que ce soit. En d'autres termes, et exprimé de manière peut-être plus brutale, il n'existe aucune corrélation *a priori* entre un échantillon d'apprentissage et les événements non vus. Plus formellement, notons $\mathsf{p}(h|\mathcal{S})$ la distribution des hypothèses dans $\mathcal{H}$ après la prise en compte de l'échantillon $\mathcal{S}$, c'est-à-dire après apprentissage. Si l'algorithme d'apprentissage est déterministe, fournissant une seule hypothèse, et toujours la même, pour un échantillon $\mathcal{S}$ donné, alors la distribution prend la forme d'un Dirac centré sur l'hypothèse choisie h. Si au contraire il s'agit d'un algorithme non déterministe, $\mathsf{p}(h|\mathcal{S})$ peut avoir une certaine extension. De la même manière, nous notons $\mathsf{p}(f|\mathcal{S})$ la distribution de probabilité des fonctions f de la nature étant donné l'échantillon d'apprentissage. L'expression de l'espérance

Copyright © 2010 Eyrolles.

de l'« écart » entre le résultat de l'apprentissage et la nature est alors :

$$E[R_{Réel}|\mathcal{S}] = \int_{h,f} \int_{\mathbf{x}\notin\mathcal{S}} p(\mathbf{x}) \, [1 - \delta(f(\mathbf{x}), h(\mathbf{x})] \, \mathbf{p}(h|\mathcal{S}) \, \mathbf{p}(f|\mathcal{S}) \tag{2.66}$$

où le symbole de Kronecker δ dénote la fonction nulle partout sauf là où ses arguments sont égaux, où elle vaut 1. Nous noterons ici que la somme ne fait intervenir que les formes x non vues en apprentissage, ce qui est différent de l'espérance de risque i.i.d. dans laquelle le tirage aléatoire des formes peut permettre le tirage de la même forme en apprentissage et en reconnaissance. Les deux expressions sont équivalentes dans le cas où l'échantillon $\mathcal{S}$ est de mesure nulle sur l'espace des entrées possibles $\mathcal{X}$. L'équation 2.66 exprime que l'espérance de risque réel étant donné un échantillon d'apprentissage $\mathcal{S}$ est liée à la somme de toutes les entrées possibles x pondérées par leur probabilité $\mathbf{p}(\mathbf{x})$, et à un « alignement » entre l'algorithme d'apprentissage caractérisé par $\mathbf{p}(h|\mathcal{S})$ et la vraie probabilité *a posteriori* de la nature $\mathbf{p}(f|\mathcal{S})$. De ce fait, en l'absence d'information *a priori* sur la distribution $\mathbf{p}(f|\mathcal{S})$, il est impossible de dire quoi que ce soit sur la performance en généralisation de l'algorithme d'apprentissage.

Si l'affirmation précédente n'a pas suffi à plonger le lecteur dans la consternation, le corollaire[15] suivant devrait achever de le faire. Nous noterons :

$$\mathbb{E}_k[R_{\text{Réel}}|f, m] = \int_{\mathbf{x}\notin\mathcal{S}} \mathbf{p}(\mathbf{x}) \, [1 - \delta(f(\mathbf{x}), h(\mathbf{x})] \, \mathbf{p}_k(h(\mathbf{x})|\mathcal{S})$$

l'espérance de risque associée à l'algorithme d'apprentissage $\mathcal{A}_k$ étant donné l'échantillon d'apprentissage $\mathcal{S}$, et la vraie fonction de la nature f.

Théorème 2.1 (No-free-lunch theorem (Wolpert, 1992))

Pour tout couple d'algorithmes d'apprentissage $\mathcal{A}_1$ et $\mathcal{A}_2$, caractérisés par leur distribution de probabilité a posteriori $\mathbf{p}_1(h|\mathcal{S})$ *et* $\mathbf{p}_2(h|\mathcal{S})$, *et pour toute distribution $d_\mathcal{X}$ des formes d'entrées* x *et tout nombre m d'exemples d'apprentissage, les propositions suivantes sont vraies :*

1. *En moyenne uniforme sur toutes les fonctions cible f dans $\mathcal{F}$:* $\mathbb{E}_1[R_{\text{Réel}}|f, m] - \mathbb{E}_2[R_{\text{Réel}}|f, m] = 0$.
2. *Pour tout échantillon d'apprentissage $\mathcal{S}$ donné, en moyenne uniforme sur toutes les fonctions cible f dans $\mathcal{F}$:* $\mathbb{E}_1[R_{\text{Réel}}|f, \mathcal{S}] - \mathbb{E}_2[R_{\text{Réel}}|f, \mathcal{S}] = 0$.
3. *En moyenne uniforme sur toutes les distributions possibles* $\mathbf{P}(f)$ *:* $\mathbb{E}_1[R_{\text{Réel}}|m] - \mathbb{E}_2[R_{\text{Réel}}|m] = 0$.
4. *Pour tout échantillon d'apprentissage $\mathcal{S}$ donné, en moyenne uniforme sur toutes les distributions possibles* $\mathbf{p}(f)$ *:* $\mathbb{E}_1[R_{\text{Réel}}|\mathcal{S}] - \mathbb{E}_2[R_{\text{Réel}}|\mathcal{S}] = 0$.

Pour une preuve de ce théorème, nous renvoyons le lecteur à [Wol92a]. De manière qualitative, le premier point de ce théorème exprime que quel que soit notre choix d'un « bon » algorithme d'apprentissage et d'un « mauvais » algorithme (par exemple un algorithme prédisant au hasard, ou bien une fonction constante sur $\mathcal{X}$), si toutes les fonctions cible f sont également probables, alors le « bon » algorithme aura la même performance en moyenne que le « mauvais ». Cela signifie aussi qu'il existe au moins une fonction cible pour laquelle la prédiction au hasard est meilleure que n'importe quelle autre stratégie de prédiction.

Le deuxième point du théorème affirme la même absence de supériorité d'un algorithme d'apprentissage sur tout autre algorithme, même quand l'échantillon d'apprentissage est connu. En

[15] Du latin *corollarium* : « petite couronne donnée comme gratification ».

Copyright © 2010 Eyrolles.

d'autres termes, celui-ci n'apporte pas plus d'informations à un algorithme plutôt qu'à un autre, fût-il à nouveau l'algorithme de prédiction au hasard. Les points trois et quatre ne font que renforcer ces résultats en affirmant l'égalité de tous les algorithmes, si l'on prend en compte des distributions non uniformes de fonctions cible, mais que l'on moyenne sur toutes ces distributions. Bien sûr, pour une distribution donnée, un algorithme va être meilleur que les autres, à savoir celui qui a la même distribution que $\mathsf{P}(f|\mathcal{S})$. Mais comment le deviner *a priori*?

Avant de discuter des leçons à tirer du *no-free-lunch theorem*, il est utile d'en illustrer la force à nouveau sur un exemple. Nous avons là en effet une sorte de loi de conservation (comme le dit Cullen Schaffer [SA94]). De même que pour chaque classe de problèmes pour laquelle un algorithme d'apprentissage est meilleur qu'un algorithme de prédiction au hasard, il existe une classe de problèmes pour laquelle cet algorithme est moins bon (voir figure 2.13). De même, pour chaque algorithme d'apprentissage, il existe des problèmes pour lesquels la courbe de performance en généralisation est ascendante et des problèmes pour lesquels cette courbe est descendante, c'est-à-dire pour lesquels plus l'algorithme apprend et plus il est mauvais en généralisation !

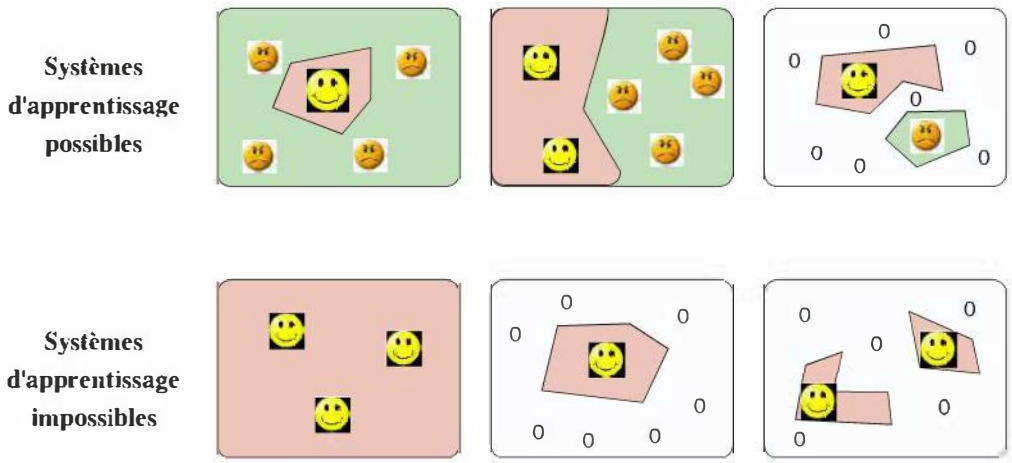


FIG. 2.13: *Le* no-free-lunch-theorem *prouve que pour chaque région de l'espace des problèmes pour laquelle un algorithme d'apprentissage a une performance supérieure au hasard (indiqué ici par un smiley « heureux »), il existe une région pour laquelle la performance est moins bonne que le hasard (indiqué ici par un smiley « triste »). Un « 0 » indique ici la performance d'un algorithme au hasard, donc la performance moyenne. Les trois figures du dessus correspondent à des situations possibles pour un algorithme d'apprentissage, tandis que les trois figures du dessous correspondent à des situations impossibles : celles d'un algorithme qui serait intrinsèquement supérieur à un algorithme au hasard quand on le considère sur l'ensemble des problèmes possibles (d'après [SA94]).*

—— EXEMPLE **Algorithme d'apprentissage de plus en plus mauvais** ——

Considérons l'algorithme de classification binaire majoritaire qui attribue à un nouveau point l'étiquette de la classe la plus représentée dans les exemples d'apprentissage de $\mathcal{S}$. Intuitivement, cet algorithme s'attend à ce que la classe la mieux représentée sur l'échantillon d'apprentissage soit de fait majoritaire. Cet algorithme simple peut-il n'être qu'équivalent à un algorithme tirant ses prédictions au hasard ? Sans en donner une preuve formelle, il est possible de s'en convaincre intuitivement. En effet, dans les problèmes pour lesquels une classe est nettement majoritaire, on peut s'attendre à ce que dans la plupart des cas l'algo-

Copyright © 2010 Eyrolles.

rithme majoritaire détecte correctement cette majorité dans l'échantillon d'apprentissage et soit de ce fait meilleur qu'une prédiction au hasard (de performance 1/2) sur les formes x non vues. Qu'en est-il alors pour les autres problèmes, ceux pour lesquels il n'existe pas de majorité nette, et qui d'après la loi binomiale sont de très loin les plus nombreux ? L'algorithme majoritaire n'est-il pas sur ceux-là équivalent à un algorithme au hasard, contredisant ainsi le *no-free-lunch theorem* ? Même si les deux classes sont également représentées sur $\mathcal{X}$, les variations d'échantillonnage feront que souvent l'une d'entre elles sera prévalente dans $\mathcal{S}$, entraînant une prédiction dans ce sens par l'algorithme majoritaire alors que, sur les exemples non vus, ce sera naturellement l'autre classe qui sera (un peu) mieux représentée. L'algorithme, sur ces problèmes, fera donc (un peu) moins bien que l'algorithme de prédiction au hasard. En prenant en compte tous les cas possibles, la performance globale de cet algorithme ne sera pas meilleure que celle de l'algorithme au hasard. Un raisonnement similaire montre que la courbe de généralisation de l'algorithme majoritaire peut être décroissante. Encore une fois, dans les cas où une classe est clairement majoritaire, l'algorithme majoritaire va avoir de plus en plus de chance de détecter correctement cette majorité avec des tailles d'échantillon croissantes (voir figure 2.14 (gauche)). Si en revanche les deux classes sont également représentées sur $\mathcal{X}$, alors la courbe va être décroissante (voir figure 2.14 (droite)). En effet, pour les petites tailles d'échantillon, la performance sera seulement légèrement inférieure à 1/2, puisque lorsque l'algorithme détectera une majorité dans son échantillon, ce sera l'autre classe qui sera de fait mieux représentée sur les exemples restants, mais de très peu. En revanche, plus l'échantillon d'apprentissage est important, plus le choix, forcément mauvais, de l'algorithme entraînera un mauvais taux de prédiction sur les exemples restants. À la limite, quand tous les exemples sauf un auront été vus par l'algorithme d'apprentissage, la prédiction sur le dernier sera forcément mauvaise (la classe prévalente sur $\mathcal{S}$ étant la classe opposée à celle de ce dernier), et la performance tombera à 0.

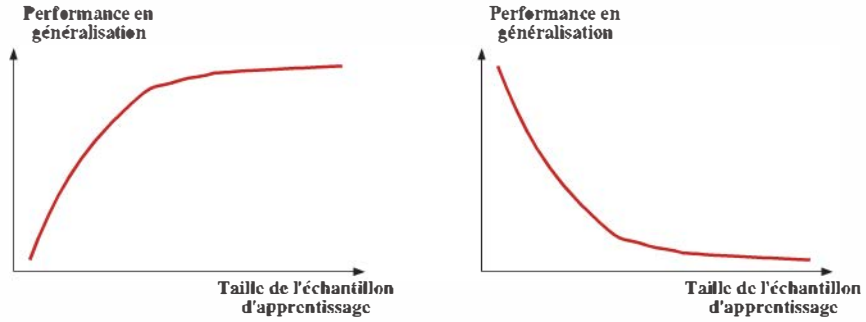


FIG. 2.14: *Le* no-free-lunch theorem *prouve que pour chaque région de l'espace des problèmes pour laquelle un algorithme d'apprentissage a une courbe en généralisation croissante (gauche), il existe une région pour laquelle il existe une courbe en généralisation décroissante, c'est-à-dire indiquant que plus l'algorithme apprend, moins il est performant sur les formes non vues (droite) (d'après [SA94].)*

5.2 Leçons à tirer du *no-free-lunch* theorem

Quelles leçons tirer de ce théorème? Faut-il jeter ce livre par terre et se maudire d'avoir consacré déjà tant de temps à étudier une science sans avenir? Le *no-free-lunch theorem* n'empêche pas de travailler sur les problèmes inductifs, il avertit simplement que la prudence est de rigueur. Plus précisément :

1. **Un algorithme d'apprentissage est forcément biaisé** vers une certaine classe de

Copyright © 2010 Eyrolles.

problèmes. C'est toujours en fonction de certains *a priori* sur les problèmes à résoudre qu'il faut concevoir et utiliser un algorithme d'apprentissage.

2. **Il n'est pas admissible de parler de la performance d'un algorithme sans préciser sur quelle classe de problèmes** il a été testé et pour quelle classe de problèmes il a été conçu.
3. **L'induction ne crée pas d'information.** Elle ne fait que transformer une information *a priori*, inscrite dans les biais de l'algorithme d'apprentissage, et qui est révélée par l'intermédiaire d'un échantillon d'apprentissage. Si l'information *a priori* est inadaptée à la situation rencontrée, le résultat sera également mauvais.

D'un certain côté, le *no-free-lunch theorem* est une nouvelle occasion de ne pas croire aux miracles. Il existe d'ailleurs d'autres versions de ce théorème pour des problèmes importants pour l'apprentissage :

1. Le *théorème du vilain petit canard* [Wat85] dit qu'il n'existe pas *a priori* de meilleur ensemble de descripteurs pour décrire des formes, et qu'en l'absence d'autres informations, il n'existe pas de meilleure notion de similarité entre formes. Toute similarité est dépendante de biais qui peuvent, ou non, être corrects pour l'application étudiée.
2. Le *no-free-lunch theorem pour les algorithmes d'optimisation* [Wol97] énonce qu'en moyenne sur tous les problèmes de recherche d'un *extremum* d'une fonction de coût, il n'existe pas d'algorithme de recherche qui soit intrinsèquement meilleur que tout autre algorithme de recherche. Cela signifie en particulier que les algorithmes de recherche par gradient, ou par recuit simulé ou par évolution simulée, tout aussi sophistiqués soient-ils, sont susceptibles d'être pires qu'une recherche au hasard sur certaines classes de problèmes.

6. Notes historiques et bibliographiques

Dire que l'apprentissage inductif est un problème d'optimisation qui conjugue un principe ou critère inductif à satisfaire au mieux et une méthode de recherche dans un espace d'hypothèse est presque devenu un dogme. Faire ressortir qu'il existe essentiellement trois types de principes inductifs : la *minimisation du risque empirique* (*MRE*), la *théorie bayésienne de la décision* qui se traduit souvent par un principe de maximum de vraisemblance, et le *principe de compression maximal de l'information*, ne suscite pas non plus de surprise. Pourtant, il a fallu beaucoup de temps pour que cette vision de l'apprentissage s'impose. Beaucoup aussi pour qu'il soit admis que ces principes, et en particulier le principe largement employé de minimisation du risque empirique, devaient être examinés pour voir s'ils conduisaient bien à la meilleure induction possible.

La théorie bayésienne de l'apprentissage s'est développée presque naturellement durant le XX^{e} siècle et en particulier depuis les années 1960. Elle conduit à la notion d'erreur bayésienne optimale, mais elle requiert une connaissance du modèle statistique sous-jacent. Nous recommandons à ce sujet la lecture des ouvrages de référence [Bis95, DHS01, Rip96, Web99].

De son côté, la théorie statistique de l'induction, dont la question centrale concerne les conditions de la convergence du risque empirique sur le risque réel, a été plus longue à émerger. Cette question ne semblait en effet pas centrale quand l'apprentissage cherchait d'abord à imiter des apprentissages naturels, dans le cadre de la biologie ou des sciences cognitives. Ce sont donc des mathématiciens essentiellement qui ont d'abord développé parallèlement et presque indépendamment les cadres *PAC* (Probablement Approximativement Correct) et VC (Vapnik-Chervonenkis).

Copyright © 2010 Eyrolles.

Le cadre *PAC* a été introduit par l'article très influent de Valiant en 1984 [Val84] dans lequel il étudiait ce qui se révèlerait un cas particulier de la convergence du risque empirique où l'espace d'hypothèses est celui de formules logiques et est supposé contenir le concept cible. Cela simplifiait considérablement les choses car, d'une part, le nombre d'hypothèses restait fini même s'il pouvait croître exponentiellement avec le nombre d'attributs et, d'autre part, on pouvait n'examiner que les hypothèses de risque empirique nul. Ce cadre incluait aussi un critère de complexité calculatoire sur l'apprentissage, imposant que la complexité reste polynomiale en un certain nombre de paramètres. Cependant, cet aspect du modèle *PAC* qui a permis de démontrer de nombreux théorèmes de non apprenabilité (en les ramenant à des problèmes de cryptographie) est pratiquement tombé en désuétude. Par ailleurs, afin de s'affranchir de la contrainte que le concept cible doive appartenir à l'espace d'hypothèses, un cadre généralisé a été proposé, appelé apprentissage agnostique. On n'en parle plus car il a été généralisé par l'approche de Vapnik.

En effet, pendant ce temps, en URSS (dans cette lointaine période d'avant la Russie), Vapnik et Chervonenkis, sous l'influence de Kolmogorov, étudiaient depuis les années 1960 le problème général de la convergence des moyennes empiriques vers leur espérance. Ils prouvèrent ainsi que la convergence des espérances de risque est équivalente à la convergence uniforme des fréquences vers des probabilités sur un domaine fini d'événements. C'est ce qui est appelé le théorème clé de la théorie statistique de l'apprentissage. Les premières bornes sur le risque réel en fonction du risque empirique furent prouvées pour la première fois par Vapnik et Chervonenkis en 1974. L'analyse montra que la convergence du risque empirique vers le risque réel fait intervenir une fonction de croissance de l'espace d'hypothèses. Comme cette fonction est très difficile à calculer, il est pratique de la caractériser par un nombre : la dimension de Vapnik-Chervonenkis. Les premiers travaux introduisant cette mesure sont ceux de Vapnik et Chervonenkis en 1971, et, indépendamment, de Sauer (1972) et de Shela (1972). L'introduction de la théorie de Vapnik et Chervonenkis s'est effectuée grâce à un article exceptionnel du « four germans gang »[16] [BEHW89] qui a eu un grand impact dans la communauté de la théorie de l'apprentissage (COLT : *Computational Learning Theory*).

L'analyse de Vapnik, largement popularisée par son livre de 1995 [Vap95], a fait prendre conscience à la communauté de l'apprentissage artificiel de l'importance cruciale de la définition et de la caractérisation de l'espace d'hypothèses. Depuis longtemps les praticiens savaient en effet qu'il leur fallait contrôler la complexité de leur modèle d'apprentissage pour ne pas être victime de surapprentissage, c'est-à-dire d'apprentissage par cœur sans généralisation. Depuis 1982, ils avaient admis, sous l'influence du papier de Mitchell [Mit82], qu'il fallait que l'espace d'hypothèses soit contraint par un biais. Cependant, c'est vraiment l'analyse de Vapnik qui a fourni un cadre conceptuel complet permettant de comprendre au moins heuristiquement le compromis entre risque empirique et capacité de l'espace d'hypothèses. Il faut cependant noter l'influence des papiers sur le compromis biais-variance [GBD92].

Pour toutes ces questions, nous reportons le lecteur aux ouvrages [CM98, Hay99, Vap95, KV94]. D'autres travaux sont plus techniques mais sont essentiels pour ceux qui veulent aller plus loin dans cette étude : [AB92, AB96, DGL96, Vid97]. Un ouvrage très intéressant sur des points de vue multiples de la théorie de l'apprentissage est [Wol95].

Le *no-free-lunch theorem* a des antécédents dans le « théorème du vilain petit canard » [Wat85] énoncé en 1963 à propos de la non-universalité de toute mesure de distance. Sa description et sa preuve sont dues à Wolpert [Wol92a] et [Wol95], de même que sa version pour les méthodes d'optimisation [Wol97]. Ce théorème a fait couler beaucoup d'encre dans les années 1990, mais il semble maintenant accepté par la communauté.

[16] Selon l'expression de Manfred Warmuth, l'un des quatre auteurs, et un théoricien éminent et inventif.

Copyright © 2010 Eyrolles.

Résumé

Ce chapitre a montré que l'induction peut être formalisée par un jeu entre une nature produisant des exemples étiquetés selon une fonction cible, et un apprenant cherchant à approcher cette fonction cible par une fonction hypothèse de manière à minimiser l'espérance de risque appelée risque réel. Pour ce faire, l'apprenant utilise un principe inductif lui dictant quelle hypothèse il doit choisir étant donnés les exemples d'apprentissage, et un algorithme de recherche effectif dans l'espace d'hypothèses. Ce chapitre s'est penché sur l'analyse des principes inductifs.

Il existe trois grands principes inductifs de base : le principe de *minimisation du risque empirique* qui dicte de choisir l'hypothèse qui minimise le risque sur l'échantillon d'apprentissage ; le *principe bayésien* qui stipule de choisir l'hypothèse minimisant l'espérance de risque, ce qui revient souvent à prendre l'hypothèse dont la vraisemblance est maximale étant donnés les exemples ; finalement le principe de *compression d'information* qui prescrit de choisir l'hypothèse permettant de transmettre l'information contenue dans les exemples d'apprentissage de la manière la plus économique. Les deux premiers ont été décrits et analysés plus en détail dans ce chapitre. En particulier, le principe *MRE* se prête naturellement à une analyse dans le pire cas, tandis que le principe bayésien, prenant en compte la distribution *a priori* des fonctions cible, conduit à une analyse en moyenne.

L'une des conséquences les plus remarquables de ces analyses est qu'elles soulignent l'importance cruciale de l'espace d'hypothèses considéré dans la confiance que l'on peut accorder aux inductions réalisées. Il faut un espace d'hypothèses suffisamment riche pour pouvoir approcher la fonction cible d'assez près, mais il ne faut pas qu'il le soit trop sous peine de conduire à des hypothèses apparemment bonnes sur les données d'apprentissage, mais mauvaises en réalité. La mise en évidence de ce compromis a amené à reconsidérer les principes inductifs pour en faire des principes inductifs avec contrôle et ajustement automatique de l'espace d'hypothèses.

Finalement, le *no-free-lunch theorem* rappelle qu'aucune méthode d'apprentissage n'est uniformément supérieure (ou inférieure) aux autres. Il faut à chaque fois spécifier l'espace de problèmes pour lequel une méthode est adaptée (ou non).

Copyright © 2010 Eyrolles.

Chapitre 3

L'environnement méthodologique de l'apprentissage

Ce chapitre a un double objectif. D'une part, il est centré sur l'utilisateur et décrit les différents points sur lesquels celui-ci peut jouer pour contrôler et ajuster l'apprentissage. D'autre part, il décrit les rouages généraux que l'on retrouve dans toutes les méthodes d'apprentissage.

D'un point de vue conceptuel, l'apprentissage se joue entre un espace de description des objets d'entrée et un espace d'hypothèses. Le choix d'un principe inductif permet d'évaluer, à partir des exemples, la qualité des hypothèses et de prescrire l'hypothèse théorique optimale. Pour qu'une méthode d'apprentissage soit effective, il faut spécifier un algorithme de recherche dans l'espace des hypothèses qui tentera d'identifier une hypothèse optimale ou du moins de s'en approcher.

Ce chapitre aborde les principaux ingrédients à la base d'une méthode d'apprentissage en partant du problème du choix de la représentation des entrées et de celle des hypothèses, puis en dressant un panorama des techniques de recherche et d'optimisation utilisables dans le contexte de l'apprentissage, enfin en examinant les méthodes d'évaluation et de validation des résultats obtenus à l'issue de l'apprentissage. Le problème du test et de la comparaison empirique des algorithmes est également discuté.

Ce chapitre se clôt en dressant une typologie des méthodes adaptées à chaque classe de problèmes.

Copyright © 2010 Eyrolles.

Sommaire

1 L'espace des données d'apprentissage 85
1.1 La représentation des objets de l'apprentissage 85
1.1.1 La nature des attributs 86
1.1.2 Représentations homogènes et représentations mixtes 88
1.2 Le prétraitement des données 88
1.3 Réduction de la dimension de l'espace d'entrée 89
1.3.1 Les grandes directions 90
1.3.2 La recherche des variables pertinentes 90
1.3.3 Techniques de sélection d'attributs 92
1.3.4 Filtres, méthodes symbiose et méthodes intégrées 93
1.3.5 Sélection vs. pondération 94
1.3.6 Détermination du nombre d'attributs pertinents 97
1.3.7 Pondération des attributs et apprentissage de tri ou *ranking* . 98
1.3.8 Références 99
1.3.9 L'extraction d'attributs 99
1.3.10 Le traitement du bruit dans les données 100
1.3.11 La discrétisation de données continues 100
1.3.12 La description des données imprécises 100
2 L'espace des hypothèses d'apprentissage 101
2.1 Le problème général de la représentation des connaissances 101
2.2 La classification 102
2.2.1 Définition 102
2.2.2 Classe, concept 103
2.2.3 Les fonctions séparatrices entre classes 103
2.3 La régression 104
2.4 Les distributions de probabilités 105
2.5 Les arbres de décision 105
2.6 Les hiérarchies de concepts 106
2.7 Les réseaux bayésiens et les modèles graphiques 107
2.8 Les chaînes de Markov et les modèles de Markov cachés 107
2.9 Les grammaires 108
2.10 Les formalismes logiques 109
2.11 Les règles d'association 110
3 Les protocoles d'apprentissage 112
3.1 Introduction 112
3.2 Batch vs. en ligne 112
3.3 Passif vs. actif 112
4 L'évaluation de l'apprentissage 113
4.1 L'évaluation *a priori* : critères théoriques 113
4.2 L'évaluation empirique *a posteriori* : généralités 114
4.3 L'estimation du risque réel d'une hypothèse 114
4.3.1 L'utilisation d'un échantillon de test 115
4.3.2 L'estimation par validation croisée 116
4.3.3 L'estimation par la méthode du *leave-one-out* 118
4.3.4 Quelques variantes de la méthode de validation croisée : *bootstrap, jackknife* 118

Copyright © 2010 Eyrolles.

4.4 Divers indices composites de performances 119
4.5 Le réglage des algorithmes par un ensemble de test 120
4.5.1 Estimation de risque : la courbe *ROC* 121
4.6 D'autres critères d'appréciation 124
5 La comparaison des méthodes d'apprentissage 124
5.1 La comparaison de deux algorithmes à partir d'un même jeu de données 125
5.1.1 La comparaison de deux algorithmes sur un même ensemble de test 125
5.1.2 Validation croisée à N plis avec t tests couplés 125
5.1.3 Les 5×2 cv t tests couplés 126
5.2 La comparaison de deux algorithmes à partir de jeux de données différents 126
5.2.1 Le test de rangs signés de Wilcoxon 127
5.3 La comparaison de multiples algorithmes 127
5.3.1 L'analyse de la variance : ANOVA 128
5.3.2 Le test de Friedman 128
5.4 Discussion 129
6 Autres problèmes pratiques 129
6.1 Classes très déséquilibrées 129
6.2 Matrice de coûts non symétrique 131
6.3 Données bruitées 131
6.4 Espace d'entrée de très grande dimension 131
6.5 Très gros volumes de données en grande dimension 132
6.6 Exemples non indépendants (non i.i.d.) 132
6.7 Apprentissage multi-objectif 133
6.8 Apprentissage multi-étiquette 133

EST-IL FACILE de définir un cygne ou une oie ? On pourrait penser que oui. Les naturalistes ont accumulé des connaissances sur cette question et les ont largement vulgarisées. Essayons donc avec notre dictionnaire usuel. Voici par exemple ce qui se trouve dans le *Petit Larousse*, édition 2000.

Cygne : oiseau palmipède *ansériforme* au long cou souple, migrateur.

Oie : oiseau palmipède massif au long cou et au bec large.

Ansériforme : oiseau, généralement palmipède, à l'allure de *canard*, mais dont certaines espèces sont des échassiers à bec crochu, tel que le *kamichi* et les *anatidés*. Les ansériformes forment un ordre.

Anatidé : oiseau palmipède au corps massif et au bec aplati, tel que le *canard*, l'*oie*, le *cygne*. Les anatidés forment une famille de l'ordre des ansériformes.

Canard : oiseau palmipède de la famille des anatidés, bon voilier et migrateur à l'état sauvage. Le canard cancane.

Kamichi : oiseau échassier des marais et des prairies humides de Patagonie, aux ailes armées de deux éperons. Longueur : 90 cm, genre Chauna, ordre des ansériformes, famille des anhimidés.

Copyright © 2010 Eyrolles.

Anhimidé ... (n'est pas une entrée dans ce dictionnaire.)

Bon... à moins que l'on soit dans un marais de Patagonie face à un échassier aux ailes armées, tout cela n'est pas très utile pour identifier un oiseau. Ces définitions circulaires masquent les données et les concepts sous des niveaux d'abstraction tellement différents qu'en pratique elles sont inopérantes. De plus, la variété des contextes (des biais d'apprentissage) est également importante : il faut au lecteur de grandes connaissances *a priori*, et très bien organisées. Par exemple la notion de *migrateur* est importante dans ces définitions et elle est supposée connue, alors que ce n'est pas une évidence à l'observation d'un oiseau... et à quoi peut bien servir de connaître le mot désignant le cri du canard pour caractériser cet animal ?

Alors, comment écrire un programme qui saurait apprendre à distinguer un cygne d'une oie? La réponse est qu'il faudra être plus modeste, c'est-à-dire soigneusement délimiter un cadre opérationnel par la définition de biais d'apprentissage. Rappelons l'exemple de l'avant-propos : l'univers est réduit à un lac sur lequel on impose que seulement deux espèces d'oiseaux puissent nager. Les observations aussi sont limitées à la taille et à la couleur. On ne cherche pas à définir la nature du cygne ou de l'oie de manière universelle : on n'a pour ambition que d'apprendre à les distinguer sous des conditions fixées de manière stricte.

Prenons maintenant sur un oiseau l'ensemble des attributs suivants :

- la taille ;
- le fait qu'il vole ou non ;
- son chant ;
- la couleur de son bec ;
- son *genre*.[1]

Ainsi la liste (`152 cm, vole, « couac », bec jaune, genre` *Anatidae*) nous indiquera, l'hiver dans nos régions, un cygne chanteur (*Cygnus Cygnus L.*) et la liste (`110 cm, ne vole pas, « krrr », bec noir, genre` *Aptedonytes*) se rapporte plutôt à un manchot, mais n'est pas assez complète pour que l'on sache de quelle espèce il s'agit.

Une autre question : est-il facile de définir une carte à jouer ? Mais oui. Il suffit de noter sa couleur et son rang, qui peuvent prendre respectivement leur valeurs dans les domaines $\{\spadesuit, \heartsuit, \diamondsuit, \clubsuit\}$ et $\{A, R, D, V, 10, 9, 8, 7, 6, 5, 4, 3, 2\}$. Cette définition est parfaite, puisque les cartes à jouer sont des objets dont le sens est par nature complètement décrit par ces deux caractéristiques. En revanche, aucun oiseau ne porte le nom de son espèce sur son plumage.

C'est que les noms donnés aux formes de la nature sont symboliques : ils sont une abstraction qui regroupe des individus selon des contrastes avec d'autres individus. Ces concepts ont été extraits d'une multitude de descripteurs souvent numériques, comme la taille, ou un peu plus abstraits comme la couleur, ou très élaborés, comme le fait d'être *migrateur* ou non. Autrement dit, les connaissances sur les individus doivent être symbolisées si l'on veut en tirer profit pour en extraire une définition opératoire.

Cette introduction n'a pour but que de rappeller que l'apprentissage artificiel doit évidemment se poser le problème de la symbolisation ou de la représentation des connaissances, qui est comme on le sait une des questions centrales de l'intelligence artificielle. Cette question est plus aisée à résoudre pour des données « artificielles » comme les cartes à jouer que pour des données naturelles, évidemment plus intéressantes.

[1] Dans la hiérarchie de Linné, cette variable est au-dessus de l'*espèce* et au-dessous de la *famille*, elle-même au-dessous de l'*ordre*.

Copyright © 2010 Eyrolles.

1. L'espace des données d'apprentissage

L'apprentissage s'appuie sur des données (des objets) qu'il faut représenter. Suivant le type de ces données, certaines représentations sont plus ou moins adaptées. Par ailleurs, toute description des données suppose déjà un prétraitement, ne serait-ce que dans le choix des attributs de description ou la manière de faire face à des données imparfaites.

1.1 La représentation des objets de l'apprentissage

Les connaissances sur les données elles-mêmes sont symbolisées grâce à un espace de représentation des données noté $\mathcal{X}$. C'est dans cet espace que s'effectue la description des objets. Dans l'exemple d'introduction de ce livre, les oies et les cygnes sont représentés par deux nombres : leur taille et leur niveau de gris. Par conséquent chaque objet (chaque oiseau) est représenté par deux valeurs numériques, ou par un point du plan, ou encore par un vecteur de $\mathbb{R}^2$. Dans l'exemple en tête de ce chapitre, un oiseau est représenté différemment : par cinq attributs de natures diverses.

Le premier exemple est très fréquent : la description d'un objet par d valeurs numériques, donc l'utilisation de $\mathcal{X} = \mathbb{R}^d$ comme espace de représentation, permet en effet d'utiliser des outils analytiques, géométriques, probabilistes, etc.

Il existe un autre cas courant : celui où les données sont représentées par un vecteur binaire. Ceci correspond au cas où l'on décrit les objets à l'aide d'une série de tests et où chaque objet est *VRAI* ou *FAUX* vis-à-vis de chaque test. Si l'on veut traiter des problèmes complexes, il faut naturellement un grand nombre de descripteurs binaires, parfois des centaines ou des milliers. Les propriétés de cet espace sont formalisées par la logique booléenne ou logique des propositions. La structure algébrique de ces espace est forte, mais les notions de continuité et de densité de probabilité sont non définies.

Définition 3.1 (Espace de représentation)

L'espace de représentation est noté $\mathcal{X}$ et ses éléments sont appelés données, instances *ou* objets. *Un* exemple $\mathbf{z}_i = (\mathbf{x}_i, u_i)$ *est un objet associé à sa supervision.*

Les éléments de $\mathcal{X}$ peuvent souvent être détaillés comme un ensemble de d attributs *ou* descripteurs *(*features *en anglais) :* $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_d\}$.

Notons que le terme « données » est vague, mais fait le lien avec l'apprentissage pour la fouille de données. Le terme « instance » est un anglicisme imprécis (souvent un objet, parfois un exemple). De même, en reconnaissance des formes, le terme « paramètre » est parfois employé, comme mauvaise traduction de *feature*.

Nous emploierons aussi la notion de distance sur l'ensemble des valeurs que peut prendre un attribut. Rappelons la définition de ce terme :

Définition 3.2 (Distance)

Une distance Δ *sur un espace $E \times E$ est une application de $E \times E$ dans $\mathbb{R}^+$ si et seulement si elle vérifie les propriétés :*

- $\Delta(x,y) = 0 \iff x = y$
- $\forall x,y \in \Sigma,\ \Delta(x,y) = \Delta(y,x)$ *(symétrie)*
- $\forall x,y,z \in \Sigma,\ \Delta(x,y) \leq \Delta(x,z) + \Delta(z,y)$ *(inégalité triangulaire)*

Copyright © 2010 Eyrolles.

L'inégalité triangulaire n'est pas toujours facile à définir dans les applications pratiques. Une application de $E \times E$ dans $\mathbb{R}^+$ qui vérifie au plus les deux premiers axiomes est parfois appelée *dissemblance*. Par abus de langage, le mot distance est souvent employé indifféremment pour ces deux concepts, en particulier en apprentissage. Nous serons par la suite aussi rigoureux que possible dans l'utilisation de ces deux termes.

Nous allons maintenant passer en revue les types d'attributs auxquels nous aurons affaire par la suite, les cas binaires et numériques étant les plus naturels et les plus simples.

1.1.1 La nature des attributs

Nous nous intéressons dans la suite de cet ouvrage aux attributs des types suivants :

Binaire

L'objet $\mathbf{x}$ est décrit par d attributs x_i dont chacun vaut 1 ou 0, autrement dit `vrai` ou `faux`.

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_d\} = \{0,1\}^d = \mathbb{B}^d$$

Dans le cas où les d attributs de $\mathbf{X}$ sont tous binaires, les données peuvent être représentées par une matrice binaire $(m \times d)$. Cette représentation a des interprétations mathématiques diverses : logique, algébrique (construction d'un treillis de Galois : chapitre 4, paragraphe 4), topologique (notion de distance), informatique (bases de données, voir chapitre 18), etc.

— EXEMPLE —

Pour quelques espèces d'animaux :

	Vole	*A des plumes*	*Pond des œufs*
oie	1	1	1
ornithorynque	0	0	1
rhinolophe	1	0	0
cygne	1	1	1

Nominal (ou catégoriel)

Par définition, un attribut de ce type appartient à un ensemble fini et non ordonné [2]. Par exemple la « couleur » $\{\spadesuit, \heartsuit, \diamondsuit, \clubsuit\}$ d'une carte à jouer est un attribut nominal dans la plupart des cas : d'une part elle ne peut prendre que quatre valeurs et d'autre part il n'y a pas d'ordre sur les couleurs. De même, une pièce au jeu d'échecs peut être de six formes différentes, mais, *grosso modo*, chacune peut s'emparer de chaque autre : elles n'ont pas d'ordre naturel de ce point de vue.

Dans certains cas, une distance ou une dissemblance peut se définir sur l'ensemble des valeurs que peut prendre un attribut nominal. Par exemple, l'ensemble des sons (ou *phonèmes*) de la langue française est un ensemble nominal : il n'est pas ordonné, mais on sait par exemple que le son `/a/` est plus proche du son `/in/` que du son `/k/`. Dans cet exemple, la propriété de l'inégalité triangulaire n'est pas vérifiée.

[2] Un attribut est *ordinal* quand il appartient à un ensemble ordonné, mais sur lequel on ne peut pas définir une distance, comme *couleur-médaille* $\in \{or, argent, bronze\}$. La confusion entre les termes « nominal » et « ordinal » est fréquente.

Copyright © 2010 Eyrolles.

Nominal arborescent

Il existe parfois une hiérarchie naturelle, mais pas un ordre total, sur les valeurs que peuvent prendre un attribut nominal. Par exemple, les groupes sanguins et facteurs rhésus sont au nombre de huit :

$$\{O+, O-, A+, A-, B+, B-, AB+, AB-\}$$

Du point de vue de la compatibilité pour la transfusion, $O+$ est « supérieur » à $A+$, $B+$ et $AB+$, puisque du sang $O+$ peut être tranfusé aux trois autres groupes et pas l'inverse. En revanche, du point de vue de cette relation d'ordre, on ne peut rien dire sur le couple $(A+, B+)$ ni sur le couple $(A+, A-)$.

Un autre exemple est celui de la couleur, donné en figure 3.1. Il sera utilisé au chapitre 4.

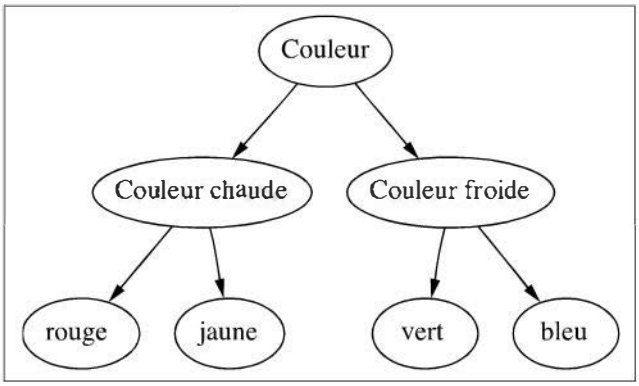


FIG. 3.1: *Une description arborescente possible pour l'attribut* Couleur.

Nominal totalement ordonné

Il est en réalité souvent possible de trouver une relation d'ordre sur un attribut nominal. La question est de savoir si elle est utile au problème ou non. Par exemple, si on s'intéresse à l'attribut couleur dans un catalogue de voitures, une relation d'ordre semble difficile à définir (le *bleu sprint* est-il supérieur ou inférieur à l'*orange calypso*?). En revanche, en astrophysique, la couleur est caractérisée par une longueur d'onde dans un certain intervalle : c'est un attribut numérique totalement ordonné, donc un intervalle de $\mathbb{R}$.

De même, dans certains jeux de cartes, les couleurs sont rangées dans un ordre décroissant : le ♠ l'emporte sur le ♥ qui l'emporte sur le ♢ qui l'emporte enfin sur le ♣.

Un attribut nominal totalement ordonné est assimilable à un intervalle de $\mathbb{R}$ ou de $\mathbb{N}$ et peut donc être muni d'une distance.

Séquenciel nominal

Un texte français est une séquence composée à partir d'un ensemble (un alphabet) d'une centaine de caractères : les cinquante-deux lettres minuscules et majuscules, l'intervalle (l'espace), quelques lettres accentuées, les signes de ponctuation, parfois des abréviations comme :-) ou €, etc. Évidemment, l'ordre de ces éléments nominaux est essentiel : la séquence « `Le commandant Cousteau` » et la séquence « `Tout commença dans l'eau` » sont différentes, bien que composées exactement des mêmes lettres[3].

On sait munir l'ensemble des valeurs que peut prendre un tel attribut d'une distance, en particulier quand l'ensemble des éléments qui composent la séquence (l'alphabet) est lui-même muni d'une distance.

[3] Dans ce cas précis, les espaces ne sont pas comptés, les accents ou cédilles non plus et les caractères minuscules et majuscules ne sont pas distingués.

Copyright © 2010 Eyrolles.

Séquenciel numérique

La cote boursière de tel ou tel titre est un exemple d'attribut séquenciel numérique : à chaque instant de temps significatif, une valeur numérique est donnée. On peut ainsi produire des séquences de plusieurs centaines de chiffres représentant l'évolution d'un cours sur une année.

Le cas de vecteurs d'attributs arrivant en séquence est typique des problèmes de traitement du signal, comme la parole : chaque centième de seconde est caractérisé après analyse spectrale par un élément de $\mathbb{R}^d$, d valant typiquement entre 10 et 20.

1.1.2 Représentations homogènes et représentations mixtes

L'espace de représentation $\mathcal{X}$ est souvent composé de d attributs de la même nature, généralement dans ce cas binaires ou numériques. Il existe aussi des espaces de représentation composés de plusieurs attributs séquenciels nominaux : par exemple dans les problèmes d'apprentissage de traducteurs, où l'on doit disposer de couples de phrases.

Dans les cas précédents, $\mathcal{X}$ est homogène : ses d attributs sont tous de même nature. Beaucoup de méthodes d'apprentissage ne peuvent s'appliquer que sur des données décrites dans un espace de représentation homogène.

Mais le cas le plus général est celui où l'espace de représentation $\mathcal{X} = \{x_1, \ldots, x_i, \ldots, x_d\}$ est mixte, autrement dit composé d'attributs de natures différentes. C'est le cas de la description d'un oiseau donnée ci-dessus pour un cygne chanteur : (`152 cm, vole, « couac », bec jaune,` genre *Anatidae*). Le premier attribut est numérique, le second est binaire, le troisième séquenciel et le dernier hiérarchique.

De même le diagnostic sur un patient entrant dans un hôpital porte sur une représentation non homogène de son état. Il pourra être décrit par exemple par les attributs suivants :

- vacciné contre la diphtérie, et si oui, depuis combien de temps ?
- température ;
- groupe sanguin ;
- description du type d'affection cutanée ;
- région et type de douleur ;
- ...

Peu de méthodes d'apprentissage sont capables d'apprendre un concept défini sur des attributs de natures diverses. La plupart du temps, on a recours à une description sous forme de combinaison booléenne de propriétés binaires extraites des attributs.

1.2 Le prétraitement des données

On effectue souvent un prétraitement des données avant de les utiliser dans l'algorithme d'apprentissage. Les différents problèmes à considérer incluent :

- *Le choix des attributs de description.* Nous avons vu lors du chapitre 1 à propos de la reconnaissance de caractères comment différents choix sont possibles et peuvent avoir une influence considérable sur la difficulté d'apprendre.
- *Le traitement du bruit.* Les données disponibles sont rarement décrites parfaitement. Souvent les défauts des instruments de mesure artificiels ou humains provoquent des erreurs. Plus grave, il arrive aussi dans le cas de l'apprentissage supervisé que les réponses de l'oracle elles-mêmes soient erronées. On qualifie ces types d'erreurs de *bruit de description* et de *bruit de classification.* Finalement, il est fréquent que les données ne soient pas décrites complètement, et qu'il y ait des *valeurs manquantes* à certains attributs. C'est le cas général pour les données médicales : seuls un certain nombre d'examens cliniques sont pratiqués sur

Copyright © 2010 Eyrolles.

chaque patient en fonction de sa pathologie, des contraintes de circonstance, etc. Ces valeurs manquantes posent souvent des problèmes difficiles à résoudre (voir par exemple le cas de l'apprentissage d'arbres de décision au chapitre 13).

- *Les données imprécises.* Les données peuvent faire l'objet de descriptions vagues : par exemple : « cet oiseau est gris ». Il faut néanmoins savoir représenter de tels attributs car ils apportent une certaine information. Il faut ensuite savoir les utiliser, en particulier pour les mettre en rapport avec les connaissances sur le monde et les hypothèses.

Examinons tour à tour plus précisément ces problèmes.

1.3 Réduction de la dimension de l'espace d'entrée

Plus l'apprentissage artificiel s'est attaqué à des problèmes réels, et plus l'importance des prétraitements s'est trouvée soulignée. Fournir des données qui soient aussi peu bruitées que possible, dans lesquelles les données aberrantes (*outliers*) sont retirées, éventuellement avec une distribution de probabilité rectifiée, est un préalable qui facilite l'apprentissage, et peut dans certains cas le rendre quasi trivial. Par ailleurs, un ensemble de descripteurs réduit peut également conduire à l'obtention de résultats d'apprentissage plus simples et plus aisés à interpréter.

Redécrire les données fait parfois partie du prétraitement. Cela peut aussi être constitutif de l'apprentissage. Parmi les codages possibles, l'un des plus simples consiste à sélectionner un sous-ensemble de descripteurs au sein des descripteurs d'origine. C'est ce que l'on appelle souvent la *sélection d'attributs*[4].

Le premier rôle de la réduction de dimension de l'espace de description des données est de **rendre l'induction possible** alors même que la faible taille de l'échantillon d'apprentissage ou la présence d'attributs non informatifs la rendent problématique. En effet, généralement, la capacité de l'espace d'hypothèses est fonction du nombre de dimensions de l'espace d'entrée. Et, comme nous l'avons vu au chapitre 2, plus cette capacité ou richesse est grande, et plus grand doit être le nombre d'exemples pour garantir en probabilité un lien entre risque empirique et risque réel. Par ailleurs, l'estimation des paramètres dans les méthodes paramétriques se dégrade exponentiellement avec le nombre d'attributs (la *malédiction ou le fléau de la dimensionnalité*, ou « *curse of dimensionality* » en anglais). Il est donc crucial de réduire au maximum la dimension de l'espace d'entrée, en conservant au maximum bien entendu l'information utile liée à l'échantillon d'apprentissage.

Lorsque les données disponibles sont en nombre suffisant par rapport au nombre de descripteurs, c'est-à-dire suffisant pour éviter le sur-apprentissage, la sélection d'attributs vise essentiellement à faciliter l'apprentissage en diminuant la complexité des calculs et/ou la taille des données à stocker. Mais, la sélection d'attributs peut également devenir l'objectif et non plus un outil ou une étape auxiliaire lorsque le problème consiste à **découvrir les facteurs clés** liés à un phénomène, comme c'est la cas par exemple dans l'analyse des facteurs de risque en médecine ou dans l'identification des gènes activés dans certaines conditions biologiques.

Un cas extrême est celui dans lequel les descripteurs sont bien plus nombreux que les exemples. Des applications aussi importantes que l'analyse du génome ou l'indexation de textes sur l'Internet ou des banques d'images ont brisé un tabou et stimulé de nouvelles recherches sur l'évaluation des attributs dans des contextes que les statisticiens s'interdisaient.

Il faut aussi noter que la dimensionnalité importante d'un espace de description est à l'origine de phénomènes qui tendent à nuire à la recherche de régularités dans les données. Comme Bellman l'avait souligné avec l'expression *malédiction de la dimensionnalité*, l'espace tend à devenir « vide » exponentiellement rapidement lorsque le nombre de dimensions de l'espace s'accroît.

[4] Dans la suite de cet ouvrage, nous utiliserons indifféremment les termes d'« attribut » ou de « descripteur ».

Copyright © 2010 Eyrolles.

Mais, outre la difficulté que cela entraîne pour l'estimation de densité qui est à la base de nombreuses techniques statistiques, cela a également des implications géométriques ennuyeuses. Ainsi, par exemple, en supposant que la densité des points soit uniforme dans l'espace, le volume d'un hypercube ou d'une hypersphère tend à se concentrer dans la « peau » de ce volume, c'est-à-dire à égale distance du centre. Cela se traduit par une *concentration des normes et des distances*. Ainsi, la norme de variables i.i.d. (identiquement et indépendamment distribuées) croît proportionnellement à $\sqrt{D}$ quand D est la dimension de l'espace, ce qui est normal, mais la variance reste approximativement constante. Donc, les vecteurs dont les composantes sont tirées aléatoirement tendent à avoir la même longueur. Cela signifie également que la distance entre deux vecteurs aléatoires tend vers une constante. La recherche de régularités par des méthodes géométriques, et généralement les approches par plus proches voisins, sont donc sérieusement menacées (voir aussi la section 2.8 du chapitre 18).

Il est clair que la réduction de dimension est un aspect important de l'analyse de données et de l'apprentissage.

1.3.1 Les grandes directions

La réduction de dimension cherche à découvrir :

- les variables non *pertinentes* ;
- les *corrélations* ou *dépendances* entre les variables (pertinentes).

La **pertinence** des variables est souvent fonction de la tâche à réaliser. Assez naturellement, c'est donc la performance sur cette tâche qui permet d'identifier les variables pertinentes, c'est-à-dire porteuses d'une information utile. L'essentiel de cette section porte sur ces techniques.

La recherche des **dépendances** entre variables ressort davantage de techniques non supervisées (voir chapitre 18). Il s'agit essentiellement de trouver une transformation des variables de base en une autre base de manière à conserver autant d'information que possible sur le signal d'origine, tout en diminuant la dimension de l'espace de description des données.

L'objectif général peut se décliner en deux sous-objectifs :

- *Identifier les variables latentes* qui « expliquent » la distribution de probabilité des données. Il s'agit d'une approche générative. On suppose que les variables observées résultent du mélange inconnu, linéaire ou non, de variables latentes.
- *Estimer la dimension intrinsèque* du nuage de points dans l'espace $\mathcal{X}$. Ainsi, les données peuvent en fait appartenir à un sous-espace de dimension très inférieure à D la dimension de $\mathcal{X}$, par exemple appartenir à un tore. On cherche alors le nombre minimal de degrés de liberté ou de paramètres indépendants permettant de reconstruire les données.
  Historiquement, l'une des premières méthodes proposées dans ce cadre est l'*analyse en composantes principales* (ACP) (voir la section 7.1).

1.3.2 La recherche des variables pertinentes

Généralement le choix des attributs vise à *diminuer le nombre des descripteurs* afin de faciliter l'apprentissage sans nuire à la qualité du résultat[5]. On distingue deux grandes approches :

- *La sélection d'attributs* consiste à éliminer les attributs les moins pertinents pour l'apprentissage. Le but est de diminuer la dimensionnalité du problème qui est à la fois une source

[5] Notons cependant que l'on peut imaginer au contraire de construire de nouveaux attributs qui viendront s'ajouter aux attributs initiaux. C'est ce qui est mis en jeu dans les techniques à base de fonctions noyaux, et en particulier dans la technique des *séparateurs à vastes marges* (SVM). Nous renvoyons le lecteur au chapitre 14 pour plus de détails.

Copyright © 2010 Eyrolles.

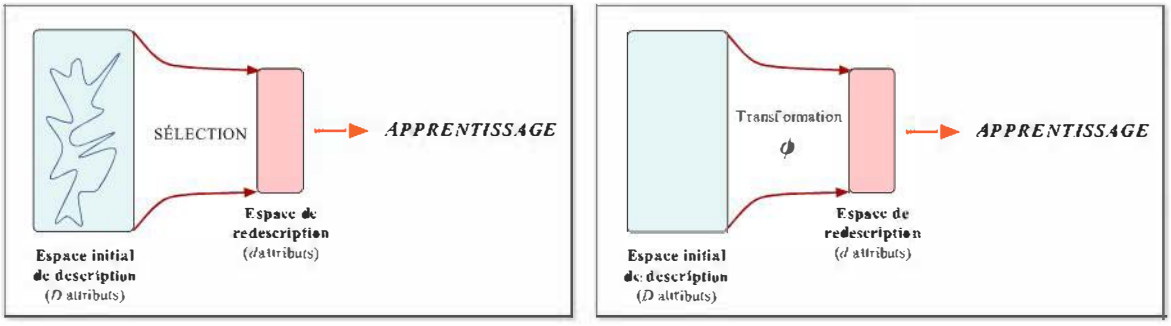


FIG. 3.2: *À gauche, la sélection d'attributs retient les d attributs les plus pertinents parmi les D attributs de l'espace d'entrées. À droite, la redescription transforme les attributs de l'espace d'entrée, ici par une fonction ϕ de combinaison des attributs initiaux, pour en construire d nouveaux.*

d'imprécision et un handicap calculatoire. Si on possède une description des données par un ensemble de D attributs, le problème est de chercher un sous-ensemble de d attributs qui préserve au mieux les informations nécessaires à l'algorithme d'apprentissage.

- *La redescription* réduit la dimensionnalité de l'espace d'entrée en appliquant des transformations, linéaires ou non, aux attributs initiaux.

Ces deux approches (voir la figure 3.2) sont fondées sur l'optimisation d'un certain critère J qui vise à estimer l'information portée par un sous-ensemble d'attributs. Dans le cas de la **sélection d'attributs**, ce critère s'applique à tout sous-ensemble d'attributs parmi l'ensemble des D attributs initiaux, et l'on cherche le sous-ensemble $\mathcal{X}_d$ de dimension $d \leq D$ optimisant J :

$$J(\mathcal{X}_d) = \max_{X \in \mathcal{X}_d} J(X)$$

Dans le cas de la **redescription**, le critère traduit la qualité des transformations possibles des D attributs initiaux, et l'on cherche la transformation $\phi^\star$ maximisant ce critère :

$$J(\phi^\star) = \max_{\phi \in \Phi} J(\phi(\mathcal{X}))$$

où Φ est l'ensemble des transformations potentielles.

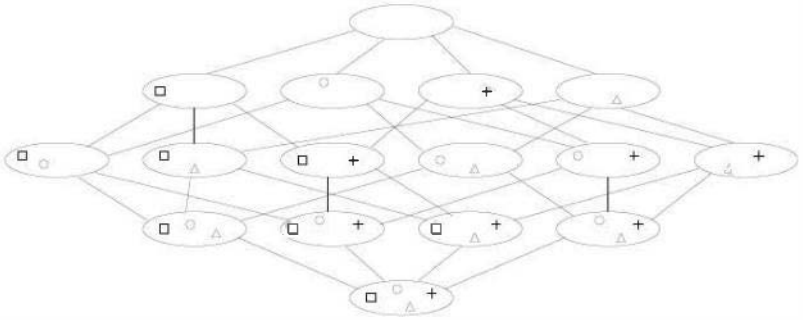


FIG. 3.3: *Treillis des sous-ensembles d'un ensemble à quatre éléments.*

La recherche du sous-ensemble optimal de descripteurs est un problème NP-difficile. S'il y a D attributs, il faut envisager de tester les 2^D sous-ensembles de descripteurs pour identifier le

Copyright © 2010 Eyrolles.

meilleur. Ces ensembles peuvent cependant être partiellement ordonnés dans un treillis grâce à la relation d'inclusion (voir la figure 3.3). De ce fait, différentes stratégies ont été proposées : parcours de l'ensemble vide vers l'ensemble total (*forward selection*, en anglais), ou parcours dans l'autre sens (*backward elimination*), ou un parcours aléatoire. Malheureusement, il n'est pas évident de disposer d'une mesure heuristique de qualité présentant des propriétés de monotonicité permettant un parcours bien guidé, comme le montre l'exemple suivant.

EXEMPLE **Non-monotonicité d'un critère de sélection**

Considérons le problème d'apprentissage de règle de classification sur un ensemble de cinq points en dimension $D = 3$ donné à la figure 3.4. Il est facile de voir que les deux classes (représentées par les symboles ● et ○) sont bien séparées, au moins sur cet ensemble d'apprentissage. Définissons un critère J, indépendant de tout algorithme, pour caractériser cette propriété. Admettons que si deux points de classes différentes sont très proches, une petite région autour d'eux va être « neutralisée », c'est-à-dire que tous les points d'apprentissage qui y sont situés seront ignorés. Le nombre de points restants est alors la valeur de J.
Puisque la séparation est parfaite en dimension 3, le critère vaut donc $J = 5$ au départ.

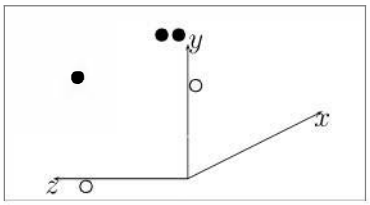


FIG. 3.4: *À trois dimensions, le critère vaut* 5.

Si on choisit $d = 2$, les figures 3.5 montrent les projections des données dans les trois sous-espaces possibles et la valeur correspondante de ce critère (les points « neutralisés » sont entourés d'un cercle hachuré). On constate que le meilleur sous-espace est (y,z), avec une valeur $J = 5$ pour le critère. Les sous-espaces (x, y) et (x, z) ont la valeur $J = 3$.

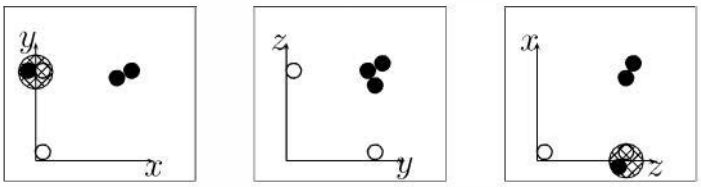


FIG. 3.5: *À deux dimensions, le meilleur sous-espace est* (y, z)*, avec une valeur* 5 *pour le critère.* (x, y) *et* (x, z) *lui donnent la valeur* 3.

Pour $d = 1$, les figures 3.6 montrent que le meilleur axe est x et que les deux plus mauvais sont y et z. Par conséquent, l'algorithme glouton qui consiste à choisir la coordonnée la plus efficace seule, puis le couple le plus efficace comprenant cette coordonnée, serait en échec sur cet exemple, puisque le couple de coordonnées le plus efficace est constitué des deux coordonnées les moins efficaces.

1.3.3 Techniques de sélection d'attributs

Les méthodes de sélection d'attributs ont pour but d'identifier les attributs qui sont utiles en vue d'une tâche de classification. Chaque exemple, ou forme d'entrée, est décrit par d attributs

Copyright © 2010 Eyrolles.

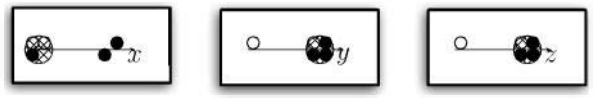


FIG. 3.6: *À une dimension, le meilleur sous-espace est l'axe x, avec une valeur 2 pour le critère. Les axes y et z ont la valeur 1.*

(e.g. gènes) et appartient à une classe (e.g. tumeur ou non tumeur). L'échantillon d'apprentissage fournit des exemples avec leur classe (supposée correcte). Le problème est de découvrir les attributs les plus informatifs pour la détermination de la classe des exemples d'apprentissage, et aussi pour les exemples à venir, encore inconnus. De plus, on peut chercher à déterminer un ensemble d'attributs minimal permettant de classer les exemples ou, au contraire, vouloir connaître tous les attributs corrélés à la classe des entrées, même s'ils sont redondants. Ce dernier cas est représentatif, par exemple, des objectifs de l'analyse du transcriptome.

Il faut noter que les attributs peuvent être informatifs à propos de la classe des exemples indépendamment les uns des autres (on parle de *corrélation linéaire*) ou en combinaison (il s'agit de *corrélations d'ordre supérieur*) (voir la figure 3.7). Il est évident que les corrélations d'ordre supérieur sont plus difficiles à découvrir que les corrélations linéaires, et exigent généralement plus de données d'apprentissage. Pour cette raison, les méthodes de sélection d'attributs sont le plus souvent orientées vers la découverte de corrélations linéaires entre les attributs (e.g. l'activité des gènes) et les classes.

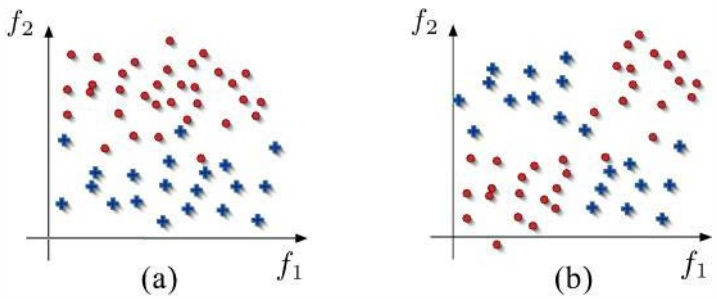


FIG. 3.7: Corrélations dans la sélection d'attributs. *En (a), l'attribut f_2 est clairement directement corrélé avec la classe des objets. En (b), en revanche, aucun des attributs n'apporte, indépendamment, d'information sur la classe des objets. Il faut prendre en compte les deux attributs ensemble (à travers une combinaison de type XOR) pour gagner une information significative sur les classes.*

1.3.4 Filtres, méthodes symbiose et méthodes intégrées

Il existe trois grandes classes de méthodes de sélection d'attributs : les *méthodes intégrées* (*embedded*, en anglais), les *méthodes symbiose* (*wrapper*) et les *méthodes de filtre* (*filter*) [BL97, GE03, KJ97].

1. Les *méthodes intégrées* consistent à utiliser directement le système d'apprentissage dans l'espoir qu'il découvrira automatiquement les descripteurs utiles pour la classification. Ainsi

Copyright © 2010 Eyrolles.

par exemple, un système d'induction d'arbre de décision (voir le chapitre 13) effectue une sélection automatique des descripteurs en choisissant ceux qui sont suffisants pour la construction de l'arbre. Malheureusement, ce type d'approche est condamné à produire des résultats peu fiables lorsque les données sont très rares par rapport au nombre d'attributs.

2. Les méthodes de type *symbiose* (comme un parasite et son hôte) évaluent les sous-ensembles d'attributs en fonction des performances des méthodes de classification qui les utilisent. Ainsi, étant donnée une méthode de classification (e.g. un perceptron multi-couche) et un ensemble d'attributs $\mathcal{F}$ (pour *features*, en anglais), la méthode symbiose explore l'espace des sous-ensembles de $\mathcal{F}$, utilisant la validation croisée pour comparer les performances des classifieurs entraînés sur chaque sous-ensemble. Intuitivement, les méthodes symbiose présentent l'avantage de sélectionner les sous-ensembles d'attributs pertinents qui permettent les meilleures performances en généralisation, ce qui est souvent le but final. Cependant, tandis qu'il a été souligné récemment que cette approche pouvait être biaisée et trop optimiste sur le vrai contenu informatif des attributs sélectionnés [XJK01], le principal inconvénient de ces méthodes est leur coût calculatoire attaché à l'exploration de l'espace des sous-ensembles de $\mathcal{F}$.

3. C'est pourquoi les *méthodes de filtre* conservent leur attrait. Elles sont utilisées dans une phase de prétraitement, indépendamment du choix de la méthode de classification. La plupart d'entre elles évalue chaque attribut indépendamment en mesurant la corrélation (selon une métrique à définir) de leurs valeurs sur les exemples avec la classe de ces exemples. En d'autres termes, ces méthodes évaluent l'information apportée par la connaissance de chaque attribut sur la classe des exemples. Sous certaines hypothèses d'indépendance et d'orthogonalité, les attributs ainsi estimés comme informatifs peuvent être optimaux par rapport à certains systèmes de classification. Un avantage important de cette approche est son faible coût calculatoire, puisqu'elle ne requière qu'un nombre d'évaluations linéaire en fonction du nombre d d'attributs, plus une opération de tri. Un inconvénient est lié au fait qu'il n'existe pas d'ensemble optimal de descripteurs indépendamment de l'algorithme d'apprentissage et de la taille de l'échantillon d'apprentissage. Ainsi, certains algorithmes sont très sensibles à la redondance dans les attributs (c'est-à-dire à leur corrélation) tandis que d'autres seront sensibles à la présence d'attributs non informatifs et bénéficieront au contraire de l'information redondante. De plus, les méthodes de filtre impliquent généralement d'utiliser un seuil permettant de séparer les attributs pertinents de ceux qui ne le sont pas. Or il n'est pas facile de déterminer ce seuil de manière non arbitraire (voir section 1.3.6).

1.3.5 Sélection vs. pondération

Il existe deux grandes approches pour la réduction de dimension. La première consiste à directement chercher à déterminer le sous-ensemble optimal de descripteurs. Cette approche, dite de *sélection*, (*subset search algorithms*, en anglais) est souvent associée aux méthodes intégrées ou aux méthodes symbiose. Cependant, comme il a déjà été noté, elle se heurte à la complexité de la recherche d'un bon sous-ensemble dans un ensemble d'éléments. La seconde approche, dite de *pondération* (*feature weighting*), consiste à évaluer le mérite de chaque descripteur indépendamment des autres, puis, dans un second temps, à déterminer les attributs à conserver, par exemple en fixant un seuil de pertinence. Elle est généralement associée aux méthodes de filtre.

1.3.5.1 Sélection. Parmi les méthodes de sélection, l'une des premières proposée a été l'algorithme FOCUS [AD91] qui utilise une recherche exhaustive en largeur d'abord pour identifier

Copyright © 2010 Eyrolles.

le sous-ensemble d'attributs de plus petit cardinal (nombre d'éléments) permettant une bonne classification des exemples d'apprentissage. Cet algorithme, de type symbiose, est évidemment inefficace en présence de nombreux attributs.

Technique de la couverture de Markov. Une autre technique, celle de la *couverture de Markov* (*Markov blanket*, en anglais), a fait l'objet de nombreux travaux récemment. Elle s'appuie sur une approche probabiliste.

Soit $\mathbf{F}$ le vecteur des attributs de description (correspondant à l'ensemble des attributs $\mathcal{F}$) prenant la valeur $\mathbf{f}$ pour un exemple donné. Et soit C la variable aléatoire associée à la classe, et qui prend la valeur ω pour l'exemple mentionné. Alors, $\mathsf{P}(C = \omega | \mathbf{F} = \mathbf{f})$ est la probabilité que la classe soit ω alors que la description de l'objet est donnée par le vecteur $\mathbf{f}$.

Soit un sous-ensemble $\mathcal{G} \in \mathcal{F}$ de descripteurs, dont le vecteur associé prend la valeur $\mathbf{f_G}$ (projection de $\mathbf{f}$ sur $\mathbf{G}$). Nous cherchons un sous-ensemble $\mathcal{G}$ tel que : $\mathsf{P}(C = \omega | \mathbf{F} = \mathbf{f_G})$ soit aussi proche que possible de $\mathsf{P}(C = \omega | \mathbf{F} = \mathbf{f})$.

La notion de proximité entre deux distributions de probabilité μ et σ définies sur une espace probabilisé $\mathcal{X}$ peut être évaluée par l'*entropie croisée* ou encore la *divergence de Kullback-Leibler*:

$$\mathrm{D}(\mu, \sigma) = \sum_{x \in \mathcal{X}} \mu(x) \log \frac{\mu(x)}{\sigma(x)} \tag{3.1}$$

Intuitivement, il s'agit de la « distance[6] » entre la vraie distribution de probabilité μ et la distribution estimée σ.

Nous voulons donc trouver un sous-ensemble $\mathcal{G}$ tel que :

$$J(\mathcal{G}) = \sum_{\mathbf{f}} \mathsf{P}(\mathbf{F} = \mathbf{f}) \, \mathrm{D}\big(\mathsf{P}(C|\mathbf{F} = \mathbf{f}), \mathsf{P}(C|\mathbf{F_G} = \mathbf{f_G})\big) \tag{3.2}$$

soit proche de 0.

Ce calcul requiert l'estimation des distributions conditionnelles $\mathsf{P}(C|\mathbf{F})$ et $\mathsf{P}(C|\mathbf{F_G})$, or nous ne disposons souvent que d'échantillons d'apprentissage trop limités pour obtenir une estimation correcte. Il est donc nécessaire de trouver une alternative. C'est là que peut intervenir la notion de *couverture de Markov*.

Brièvement, un réseau bayésien est un graphe orienté acyclique dans lequel les nœuds représentent des variables et les arcs des dépendances conditionnelles entre ces variables. Si un arc part d'un nœud X vers un nœud Y, cela correspond à une dépendance directe de la variable Y par rapport à la variable X et X est appelé *parent* de Y (Y étant un descendant de X) (voir le chapitre 11 pour plus de détails). La *couverture de Markov* d'un nœud A est l'ensemble $MB(A)$ des nœuds qui rendent A conditionnellement indépendants de tous les autres nœuds du graphe. Plus formellement, soient deux nœuds distincts A et B :

$$\mathsf{P}(A|MB(A), B) = \mathsf{P}(A|MB(A)) \tag{3.3}$$

On peut montrer que la couverture de Markov d'un nœud est égal à l'ensemble de ses nœuds parent, de ses fils et des parents de ses fils (voir figure 3.8).

La couverture de Markov d'une variable correspond donc à l'ensemble des variables dont la connaissance suffit à déterminer sa valeur. En particulier, si A correspond à la variable classe, sa couverture de Markov fournit un sous-ensemble d'attributs suffisants pour en déterminer la valeur. Un **algorithme de sélection d'attributs** repose sur le principe suivant :

[6] En toute rigueur, il ne s'agit pas d'une distance car $\mathrm{D}(\cdot, \cdot)$ n'est pas symétrique.

Copyright © 2010 Eyrolles.

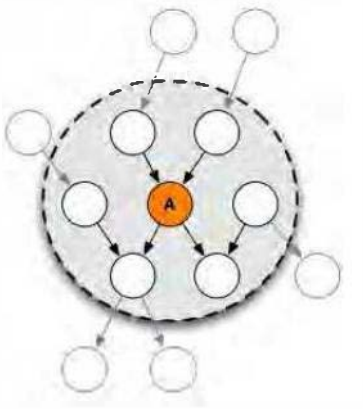


FIG. 3.8: *Couverture de Markov d'une variable A dans un réseau bayésien.*

1. Si on peut identifier une couverture de Markov pour l'attribut f_i, retirer f_i des attributs à considérer.
2. Recommencer jusqu'à ce que la couverture soit égale à l'ensemble des attributs restants.

Un théorème important de [KS96] assure que cette procédure est monotone et peut être appliquée dans n'importe quel ordre des variables. Plusieurs algorithmes ont été proposés selon ce principe. Cependant, cette approche est coûteuse en termes de calculs et nécessite des échantillons d'apprentissage importants.

Algorithme Winnow. L'algorithme Winnow, dans sa version d'origine [Lit88], s'applique à des concepts booléens linéairement séparables[7], par exemple des disjonctions.

Winnow, comme l'algorithme du perceptron (voir section 3.3 dans le chapitre 9), apprend un vecteur de poids sur les attributs de description des exemples. La différence réside dans la formule de mise à jour de ces poids, qui est multiplicative au lieu d'être additive. Cette méthode de *gradient exponentiel* qui est à la base du *boosting* et de méthodes d'ensemble (voir chapitre 17), donne à Winnow la propriété d'éliminer très rapidement les attributs non pertinents, en particulier quand ceux-ci sont nombreux.

1.3.5.2 Pondération. Les algorithmes de pondération calculent un poids pour chaque attribut indépendamment et les trient en fonction de ce poids. Plusieurs concepts de poids ont été proposés dont le point commun est de tenter d'estimer l'information apportée par la connaissance d'un attribut sur la classe de l'exemple étudié.

On notera en particulier les mesures suivantes proposées pour estimer l'apport d'information de chaque attribut.

- Mesures d'**information.** Ces mesures incluent la mesure d'entropie de variable, d'entropie conditionnelle et de gain d'information. ([YL04a, YL04b, JJ00]) ;
- Mesures **de distance**. ([RSK03, PN98]) ;
- Mesures **de dépendance**. ([Hal00, Mod93]) ;
- Mesures **de cohérence** (*consistency*). L'idée est de déterminer le nombre minimal d'attributs permettant de séparer les classes aussi bien que lorsque tous les attributs sont utilisés. Une *incohérence* est dite exister lorsque deux exemples ont la même valeur pour un attribut mais sont de classes différentes. ([AD94, DL03]) ;

[7] C'est-à-dire une fonction $f : \{0,1\}^n \to \{0,1\}$ telle qu'il existe un hyperplan de $\mathbb{R}^n$ séparant les deux classes.

Copyright © 2010 Eyrolles.

- Mesures de **précision** (*accuracy*). On utilise ici la précision de la méthode de classification en aval comme mesure de la qualité d'un jeu d'attributs. ([DL00, KJ97]) ;
- Mesures de **pertinence** (*relevance*). On dit qu'un attribut f_i est *fortement pertinent* si $\mathbf{P}(C|f_i, \mathcal{F}\setminus\{f_i\}) \neq \mathbf{P}(C|\mathcal{F}\setminus\{f_i\})$. L'attribut f_i est *faiblement pertinent* si $\mathbf{P}(C|f_i, \mathcal{F}\setminus\{f_i\}) = \mathbf{P}(C|\mathcal{F}\setminus\{f_i\})$ et $\exists \mathcal{S}_i \subset \mathcal{F}\setminus\{f_i\}$ tel que $\mathbf{P}(C|f_i, \mathcal{S}_i) \neq \mathbf{P}(C|\mathcal{S}_i)$. (Voir [JKP94]).

Voici deux exemples de systèmes par pondération des attributs.

EXEMPLE **Analyse de la variance (ANOVA)**

La méthode statistique de l'analyse de la variance peut être utilisée pour évaluer la corrélation de chaque attribut avec la classe. Le principe de la méthode repose sur la comparaison de la variance des valeurs prises par chaque attribut lorsque la classe des exemples est prise en compte et lorsqu'elle ne l'est pas. Si ces variances sont significativement différentes, cela indique que l'attribut apporte une information sur la classe. ANOVA est une méthode paramétrique, en ce qu'elle suppose que les valeurs des attributs suivent une distribution normale. Le F-test est employé pour évaluer la pertinence, c'est-à-dire le poids, de chaque attribut selon un test d'hypothèse qui examine les deux hypothèses :

- H_1 : les valeurs de l'attribut étudié diffèrent significativement selon la classe des exemples.
- H_0 : les valeurs des attributs ne diffèrent pas significativement.

EXEMPLE **La méthode RELIEF**

La méthode RELIEF [KR92, RSK03] évalue les attributs en fonction de leur apparente corrélation avec la classe. Le score de chaque attribut est fonction des variations de ses valeurs à l'intérieur de chaque classe par rapport à ses variations inter-classes. Cependant, par contraste avec les méthodes paramétriques, la méthode ne fait aucune hypothèse sur la distribution des valeurs des attributs. Par ailleurs, même si le score est calculé pour chaque attribut, son calcul repose sur une mesure de distance dans l'espace de tous les attributs qui tend à favoriser les attributs corrélés entre eux.

Cette méthode est facile à mettre en œuvre, ne repose pas sur des hypothèses statistiques hasardeuses et permet, dans une certaine mesure, d'identifier des corrélations non linéaires entre les attributs et la classe (voir figure 3.7 (b)). De plus, sa complexité calculatoire est remarquablement faible : en $\mathcal{O}(kmD)$ pour un échantillon de m exemples décrits par D attributs et k étant le nombre de voisins pris en compte dans l'algorithme. En revanche, cette méthode, comme toutes les méthodes de filtre linéaire en général, ne permet pas d'éliminer les attributs redondants. C'est une bonne chose quand le but est d'identifier tous les facteurs impliqués dans un processus (comme par exemple en génomique), une moins bonne chose quand le but est d'obtenir un bon classifieur.

1.3.6 Détermination du nombre d'attributs pertinents

Dans les méthodes de pondération particulièrement, se pose la question de déterminer un seuil de pertinence au-delà duquel on décide que les attributs ne valent pas la peine d'être conservés. Sauf à être arbitraire, il est difficile de fixer ce seuil directement. Plusieurs méthodes ont été proposées, parmi lesquelles :

- *Introduction d'attributs aléatoires.* L'idée est de mesurer le poids d'attributs artificiels pour lesquels on sait que les valeurs sont indépendantes de la classe. On ne retient alors que les attributs dont le poids est significativement supérieur à celui de ses attributs artificiels.
- *Comparaison au cas de classes aléatoires.* Le principe est de permuter aléatoirement les étiquettes des exemples, et de mesurer le poids des attributs dans cette condition aléatoire.

Copyright © 2010 Eyrolles.

On peut alors choisir de ne retenir que les attributs qui ont un poids supérieur au poids du meilleur attribut sous la condition aléatoire puisque la probabilité que ceux-ci soient bien évalués par hasard est faible. Ce critère est cependant en général trop restrictif, et on retient plutôt les attributs pour lesquels le rapport entre leur score avec les vraies étiquettes et sous la condition aléatoire est le plus grand.

- *Croisement de méthodes d'évaluation.* Une autre approche proposée récemment est d'utiliser plusieurs méthodes d'évaluation différentes (e.g. RELIEF et ANOVA) et d'utiliser l'intersection entre les meilleurs classés par les différentes approches pour décider du seuil (voir [CFM05]).

Remarque (Vers une théorie de l'apprentissage de classement des attributs ?)

Il est remarquable que, d'une certaine manière, apprendre à trier à partir d'exemples devrait être beaucoup plus simple que d'apprendre la fonction cible sous-jacente.

En effet, prenons le cas simple d'exemples dans $\mathcal{X}$ décrits par d attributs booléens et prenant leur étiquette dans $\mathcal{Y} = \{0, 1\}$. Le nombre de fonctions possibles de $\mathcal{X}$ dans $\mathcal{Y}$ est de 2^{2^d}, soit par exemple 2^{1024} dans le cas de $d = 10$ attributs.

En revanche, le nombre d'ordres possibles sur les d attributs est seulement de $d!$, soit $10! = 3\,628\,800$ pour 10 attributs (voir la figure 3.9 pour un exemple avec $d = 3$).

a1	a2	a3	XOR
0	0	0	-
0	0	1	+
0	1	0	+
0	1	1	-
1	0	0	-
1	0	1	+
1	1	0	+
1	1	1	-

$2^{2^3} = 2^8 = 256$
fonctions possibles

Mais seulement :
10 tris possibles
sur les attributs
(e.g. (a1,a2,a3))

FIG. 3.9: *Nombre de fonctions et de classements des attributs pour 3 attributs. À gauche, une table décrivant des exemples d'une fonction cible, qui, ici, est la fonction XOR.*

Pourtant, il n'existe pas pour le moment de théorie de l'apprentissage de classement des attributs. Une des difficultés provient de ce qu'il n'y a plus de notion de risque empirique mesurable ici. Il est évident qu'il y a là un domaine de recherche intéressant.

1.3.7 Pondération des attributs et apprentissage de tri ou *ranking*

Les méthodes de pondération des descripteurs permettent de trier ceux-ci par ordre croissant ou décroissant de pertinence (*feature relevance*, en anglais). Ces méthodes sont donc intéressantes à connaître dans le contexte plus général de l'*apprentissage de classement* ou *de tri* (*learning to rank* ou *ranking*).

L'apprentissage de classement est mis en œuvre lorsqu'il s'agit de produire en sortie un classement de réponses ordonnées par une fonction d'utilité. L'exemple le plus notable de cette tâche, et qui est à l'origine des travaux récents sur ce sujet, est celui de la présentation ordonnée des réponses à une requête d'un utilisateur sur la Toile. Dans ce problème, les données consistent en des exemples de requêtes et, pour chacune d'entre elles, un ensemble trié de réponses.

Copyright © 2010 Eyrolles.

Les recherches sur ce problème sont extrêmement actives et il n'est donc pas question d'en faire un état de l'art exhaustif et à jour ici (voir [ACH05] pour un ensemble d'articles). Deux approches rassemblent cependant une grande partie des travaux.

- La première pose le problème en terme d'un apprentissage de « régression ordonnée », c'est-à-dire pour lequel il s'agit d'apprendre à associer à une requête et une donnée (e.g. un document), un numéro dans un classement, ce qui peut aussi se faire en associant un intervalle dans $\mathbb{R}$. (voir [HGO00]).
- La seconde pose le problème en termes d'un apprentissage associant à une requête et une paire de données (e.g. documents), un ordre sur cette paire. Il est possible d'apprendre de la sorte n'importe quel classement, y compris des classements incohérents (e.g. $A \succ B$, $B \succ C$ et $C \succ A$). (Voir [BSR+05, CS02, FLSS04]).

1.3.8 Références

Sur la réduction de dimension, des références générales incluent [BL97, GE03, KJ97, Ng98, XJK01, YL03, LV07]. Des travaux s'adressent plus particulièrement à la sélection des gènes pertinents dans l'analyse du transcriptome, par exemple [KMC00].

1.3.9 L'extraction d'attributs

L'extraction d'attributs transforme l'espace d'entrée en remplaçant les attributs d'entrée par un ensemble plus petit correspondant si possible aux régularités sous-jacentes. On distingue souvent les approches par *transformations linéaires* de l'espace d'entrée de celles par *transformations non linéaires*. Parmi les premières, les plus usuelles sont :

- L'analyse en composantes principales, dont le but est d'identifier un petit ensemble de variables décrivant les données en minimisant la perte d'information. Cette dernière est mesurée par la variation dans l'échantillon de données, à travers une matrice de covariance ou de corrélation. Cette méthode ne prend pas en compte la classe des exemples : c'est une technique non supervisée.
- La méthode d'analyse en composantes principales communes prend au contraire en compte la classe des exemples et s'appuie sur une mesure du maximum de vraisemblance ou de l'écart aux moindres carrés.

Les méthodes d'extraction d'attributs par transformations non linéaires sont moins employées. Parmi elles figurent :

- La méthode des cartes auto-organisatrices de Kohonen, qui utilise une sorte de réseau connexionniste dans une approche non supervisée.
- Des méthodes issues des recherches sur les séparateurs à vastes marges (SVM). Nous renvoyons le lecteur intéressé à [SBE99], au chapitre 14[8].
- L'analyse en composantes indépendantes (*Independent Component Analysis*, ICA), qui est une technique récente connaissant un grand développement. Elle s'applique dans le cas où l'on suppose que les données proviennent de plusieurs sources indépendantes, combinées par une matrice de mélange. Tandis que l'analyse en composantes principales impose seulement une indépendance des données jusqu'à l'ordre deux (mais une orthogonalité des variables), l'analyse en composantes indépendantes suppose une indépendance statistique des sources, sans contrainte d'orthogonalité (voir la section 7.2 du chapitre 18, ou bien [HKO01a], ou encore le chapitre 10 de [Hay99] pour une introduction).

Le chapitre 6 de [CM98] est intéressant à consulter à ce sujet, de même que les chapitres 8 et 10 de [Hay99].

[8] L'analyse en composante principale par fonctions noyaux.

Copyright © 2010 Eyrolles.

1.3.10 Le traitement du bruit dans les données

Les bases de données dans lesquelles on essaie de découvrir des régularités sous-jacentes à l'aide de techniques d'apprentissage artificiel sont rarement parfaites, c'est-à-dire complètement et parfaitement décrites. Non seulement les données peuvent comporter des erreurs de description ou d'étiquetage, être imprécises, mais elles sont souvent inhomogènes, résultant de plusieurs sources rassemblées dans des contextes différents. Le plus souvent aussi, elles n'ont pas été constituées dans le but d'être analysées par une machine[9]. Il arrive aussi que des valeurs ne fournissent que des informations sur des contingences externes au problème étudié. Une banque a ainsi eu la surprise de découvrir récemment que plus de 75 % de ses clients étaient nés le 11 novembre 1911. Il était en effet plus rapide (et sans importance apparente) pour les opérateurs remplissant les fiches de saisir « 111111 ». Il faut également tenir compte de conventions implicites, telles que signaler une date manquante par « 9999 », ou un poids manquant par la valeur « -1 kg ». Sans précautions, il est facile d'obtenir des résultats erronés et, ce qui est pire, sans que personne ne s'en aperçoive.

Le traitement du bruit dans les données n'est pas un problème facile à résoudre, simplement parce qu'il n'est pas facile de distinguer ce qui est le résultat d'une erreur ou d'une variation non significative d'une observation authentique. Les méthodes usuelles reposent sur des tests statistiques du niveau de pertinence. Des outils de visualisation des données peuvent être précieux dans la détection d'anomalies. Cependant rien ne remplace l'avis éclairé d'un expert et la maîtrise des phénomènes à la source des données.

Il faut aussi noter que le bruit n'est pas toujours une mauvaise chose pour l'apprentissage. Au contraire, il peut arriver que l'on introduise volontairement du bruit dans les données afin de faciliter l'apprentissage de vraies généralisations au lieu d'apprendre par cœur les données sans en induire les régularités. L'introduction de bruit agit alors comme un facteur de régularisation (voir le chapitre 21 section 2.2).

1.3.11 La discrétisation de données continues

Certains algorithmes d'apprentissage, particulièrement les algorithmes symboliques, sont incapables de traiter directement des attributs à valeur continue. Il est nécessaire de les transformer en attributs à valeur discrète. Une autre raison pour discrétiser un attribut à valeur continue provient de ce que la distribution des valeurs peut ne pas être uniforme ou gaussienne, alors que la plupart des algorithmes en font la supposition (parfois implicite et méconnue de l'utilisateur). Il faut alors discrétiser en intervalles de distributions correspondants à des distributions uniformes ou gaussiennes.

Les méthodes de discrétisation sont nombreuses (par segmentation, par mesures d'entropie, etc.) et souvent dédiées à un contexte d'utilisation particulier. Nous renvoyons le lecteur aux publications sur le sujet, particulièrement dans le domaine de la fouille de données (*Data Mining*) : [HK01, WF99].

1.3.12 La description des données imprécises

L'une des méthodes les plus utilisées pour décrire des données imprécises est la logique floue. Nous renvoyons à [BM94] pour plus de détails.

[9] Il arrive parfois que les valeurs manquantes soient de fait plus informatives que les autres dans la mesure où elles révèlent l'interprétation du praticien (par exemple en médecine les champs manquants suffisent souvent à déterminer le diagnostic).

Copyright © 2010 Eyrolles.

2. L'espace des hypothèses d'apprentissage

Le chapitre 1 a souligné l'utilité de définir un espace d'hypothèses afin de ne pas avoir à représenter les concepts décrivant les données par des descriptions en extension, c'est-à-dire par des listes d'exemples. L'espace $\mathcal{H}$ des hypothèses, défini par le langage des hypothèses $\mathcal{L}_{\mathcal{H}}$, permet le recours à une description *en intension*, compacte et permettant d'établir naturellement des liens avec les autres connaissances disponibles. Le premier problème est de savoir représenter les connaissances, donc de trouver un langage approprié au contexte et à la tâche. Le second est de savoir comment mettre en relation des hypothèses et des données. C'est ce que l'on appelle souvent le problème de l'appariement (*matching*).

2.1 Le problème général de la représentation des connaissances

Les représentations des connaissances en intelligence artificielle ne se font pas en langage naturel, pour des raisons évidentes. On cherche plutôt des représentations à la fois expressives et concises, permettant d'exprimer tout ce que l'on désire de manière succincte, non ambiguë, indépendante du contexte et efficace, c'est-à-dire se prêtant naturellement aux raisonnements désirés. Plusieurs types de représentations ont été développés pour répondre à ces exigences. Il est intéressant de les comparer du point de vue de l'apprentissage.

1. Quels *types de régularités* ou de connaissances veut-on représenter ?
   - Des catégories ou classes ou concepts.
   - Des probabilités d'appartenance à une catégorie.
   - Des ontologies, c'est-à-dire des classes organisées hiérarchiquement.
   - Des règles d'association, des réflexes.
   - Des dépendances causales.
   - Des descriptions relationnelles.
   - Des évolutions temporelles.
   - ...

2. Quelles sont les *caractéristiques des entrées* disponibles ?
   - Entrées perceptives brutes ou déjà prétraitées.
   - Entrées discrètes ou continues.
   - Entrées bruitées ou non.
   - Entrées correspondant à des phénomènes déterministes ou non.
   - Entrées affectées d'incertitude.
   - Entrées affectées d'imprécision.
   - Entrées « plates », telles que des vecteurs d'attributs, ou structurées par des relations et une organisation, comme des graphes.

3. Quel *degré de transparence ou d'interprétabilité* souhaite-t-on dans les hypothèses produites par le système ?

   Ce dernier aspect est très important. Si l'on cherche seulement un système performant sur une tâche donnée, sans qu'il y ait nécessité d'interaction avec un « expert », une représentation opaque est acceptable. C'est par exemple le cas d'un système de reconnaissance de caractères ou d'identification de locuteurs sur la base d'un signal sonore. En revanche certaines applications exigent que l'utilisateur puisse examiner la connaissance produite par le système. C'est le cas d'un système de diagnostic médical et plus encore d'un système chargé de formuler des recommandations thérapeutiques. Mais cela peut aussi être utile

Copyright © 2010 Eyrolles.

lorsque l'expert peut aider le système à apprendre en lui transmettant des connaissances *a priori*. Encore faut-il qu'il soit alors possible de les traduire pour la machine. C'est généralement impossible avec une représentation « opaque » telle que la représentation utilisée dans les réseaux connexionnistes qui consiste en une matrice de nombres correspondant aux poids des connexions du réseau. C'est en revanche plus facile si la représentation utilise un formalisme logique.

Nous présentons maintenant les différents espaces d'hypothèses $\mathcal{H}$ que nous allons rencontrer par la suite. Ces espaces de représentation seront décrits avec plus de précision au fur et à mesure des chapitres à venir. Pour le moment, il est seulement question de faire un tour d'horizon des représentations utilisées en apprentissage artificiel. Il est d'ailleurs intéressant de noter que toutes les techniques de représentation des connaissances utilisées en intelligence artificielle ne sont pas citées ici : certaines d'entre elles ne se prêtent pas (encore ?) à l'apprentissage.

La table suivante présente d'abord les qualités des différentes représentations des hypothèses en fonction des critères cités ci-dessus.

	Fonctions séparatrices	Distributions de probabilités	Fonctions état → action	Arbres de décision	Hiérarchies de concepts	Réseaux bayésiens	Chaînes de Markov	Grammaires	Systèmes de règles
Concept	√	√	-	√	√	-	-	√	√
Classes multiples	√	√	-	√	√	-	-	-	√
Ontologies	-	-	-	√	√	-	-	-	√
Régression	-	√	√	√	-	-	-	-	√
Évolutions temporelles	-	√	√	-	-	-	√	√	-
Apprentissage non supervisé	√	√	√	√	√	-	-	-	-
Données continues	√	√	√	√	-	-	√	-	-
Connaissances relationnelles	-	-	√	-	√	√	-	√	√
Degré de certitude	-	√	√	-	-	√	√	√	√
Degré d'imprécision	-	√	√	-	-	√	-	-	-
Transparence, intelligibilité	-	-	-	√	√	√	-	√	√

2.2 La classification

2.2.1 Définition

L'apprentissage d'une règle de classification est l'un des thèmes de l'apprentissage artificiel le plus traité. Il y a plusieurs raisons à cela : d'abord, on sait l'aborder du point de vue des théories de l'apprentissage, la plupart du temps dans le cas de deux classes (mais on peut assez facilement généraliser à un nombre quelconque). Ensuite, un grand nombre de méthodes et d'algorithmes existent, en particulier dans le cas où l'espace de représentation est numérique. On est alors dans le domaine classique de la reconnaissance statistique des formes (*statistical pattern recognition*). Enfin, apprendre à classer est un problème central de l'intelligence, naturelle comme artificielle.

Copyright © 2010 Eyrolles.

Intuitivement, une règle de classification est un acte cognitif ou une procédure permettant d'affecter à un objet la famille à laquelle il appartient, autrement dit de le reconnaître. C'est ainsi qu'un enfant apprend à classer les animaux domestiques en « chiens » ou « chats », les plats en « salé » ou « sucré », etc. Par analogie, les ordinateurs de bureau qui reconnaissent l'écriture manuscrite ont appris (grâce à un programme d'apprentissage automatique) des règles pour distinguer les signes tracés ; d'autres programmes savent classer des sons, des signaux biomédicaux, etc. Toutes les procédures qui simulent des fonctions perceptives doivent évidemment posséder des capacités de généralisation, c'est-à-dire être munies de la faculté d'induction, sans quoi elles ne seraient capables de ne reconnaître que les exemples qui ont servi à les entraîner.

2.2.2 Classe, concept

Définition 3.1 (exemple)

Un exemple *est un couple* $(\mathbf{x}, u)$, *où* $\mathbf{x} \in \mathcal{X}$ *est la* description *ou la* représentation *de l'objet et* $u \in \mathcal{U}$ *représente la* supervision *de* $\mathbf{x}$. *Dans un problème de classification,* u *s'appelle la* classe *de* $\mathbf{x}$ *et appartient à un ensemble* $\mathcal{C} = \{\omega_1, \ldots, \omega_C\}$. C *désigne le nombre de classes possibles pour un objet.*

C doit être fini et en pratique petit pour que l'on puisse réellement parler de classification. Des exemples de classes sont : les sons du langage, l'alphabet, les espèces des oiseaux, un diagnostic médical, la présence ou l'absence d'une propriété pour un objet (par exemple qu'une carte à jouer soit un « honneur »), etc.

Dans le cas où $C = 2$, il est usuel de considérer que l'on fait l'apprentissage d'un *concept*, c'est-à-dire du partage de l'espace de représentation en deux parties, l'une où le concept est vérifié, l'autre où il est invalidé. Dans ce cas, on note[10] en général $\mathcal{C} = \{\texttt{vrai}, \texttt{faux}\}$ et on appelle *contre-exemples* les données classées faux (on garde le mot d'*exemples* pour les autres).

Il est à noter que le cas $C = 1$ est presque équivalent au précédent, puisqu'il s'agit d'apprendre aussi un concept, mais à partir seulement d'exemples ; en pratique, cependant, les algorithmes seront différents.

Par exemple, un enfant apprend sa langue maternelle avec un « algorithme » de généralisation où le rôle des contre-exemples est faible. En revanche, il classe les matières d'enseignement en celles qu'il aime et celles qu'il n'aime pas à partir d'une base d'apprentissage composée d'exemples des deux cas.

2.2.3 Les fonctions séparatrices entre classes

Au lieu d'essayer d'approcher directement la fonction de classification cible $f : \mathcal{X} \to \{\omega_1, \ldots, \omega_C\}$ par une règle de classification, il est souvent plus facile de transformer l'espace des classes en celui des fonctions séparatrices.

Définition 3.2 (fonction séparatrice)

Une fonction séparatrice, ou fonction de décision $\Upsilon_{ij} : \mathcal{H} \to \mathbb{R}$ *entre la classe* ω_i *et la classe* ω_j *est telle que* $\Upsilon_{ij}(\mathbf{x}) \geq 0$ *pour tous les objets* $\mathbf{x}$ *que la fonction cible affecte à la classe* ω_i *et* $\Upsilon_{ij}(\mathbf{x}) \leq 0$ *pour tous les objets qu'elle affecte à la classe* ω_j. *Si* $\Upsilon_{ij}(\mathbf{x}) = 0$, *l'affectation se fait arbitrairement.*

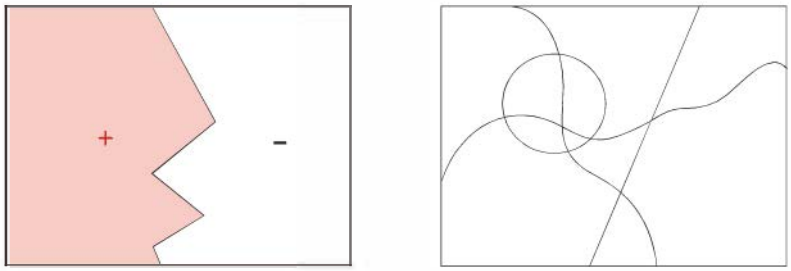
L'espace de l'apprentissage devient alors un ensemble d'hypothèses consitué de fonctions séparatrices. Ces fonctions peuvent être de natures extrêmement variées : par exemple des hyperplans

[10] Parfois $\mathcal{C} = \{+, -\}$, ou $\mathcal{C} = \{1, 0\}$, ou $\mathcal{C} = \{+1, -1\}$.

Copyright © 2010 Eyrolles.

(voir le chapitre 9), ou calculées à partir de réseaux connexionnistes multicouches (voir le chapitre 10) ou de densités de probabilités (voir le chapitre 15), etc.

Dans la définition ci-dessus, on ne considère que le signe de la fonction de décision pour décider de la région d'appartenance de l'entrée $\mathbf{x}$ (voir figure 3.10 (a)). On parle souvent dans ce cas de *fonctions séparatrices à seuil*. Dans le cas où il y a plus de deux classes, on peut combiner plusieurs fonctions de décision permettant ainsi une division de $\mathcal{X}$ en plusieurs régions (voir figure 3.10 (b)). On y reviendra au chapitre 9.



(a) (b)

FIG. 3.10: *Classification par fonctions séparatrices. En (a) la fonction séparatrice détermine deux classes suivant le signe de la sortie de la fonction. En (b), une classification pour plus de deux classes est obtenue par la combinaison de plusieurs fonctions séparatrices.*

Si en plus de son signe, on considère aussi la valeur de la sortie de la fonction de décision, il devient possible d'interpréter cette dernière comme une mesure de confiance dans la décision, selon l'idée naturelle que plus la forme d'entrée est « éloignée » de la frontière, plus son appartenance à la classe désignée est peu susceptible d'être remise en cause. Nous verrons que cette observation de bon sens est à l'origine d'un renouveau très fort pour l'utilisation de ces fonctions de décisions (voir les séparateurs à vastes marges dans le chapitre 9).

En dehors de leur simplicité conceptuelle et pratique évidente, les fonctions séparatrices permettent de mettre en œuvre naturellement un appariemment partiel entre entrée et hypothèse. En effet, les fonctions séparatrices peuvent se concevoir comme une sorte de produit scalaire défini sur $\mathcal{X} \times \mathcal{H}$. Ainsi, dans le cas du perceptron, déjà rencontré au cours du chapitre 2, la fonction de décision est définie par :

$$\mathbf{w}^T\mathbf{x} \begin{cases} \geq 0 \\ < 0 \end{cases} \implies \mathbf{x} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases} \tag{3.4}$$

en considérant le vecteur de description des observations augmenté $\mathbf{x}^T = (1, x_1, x_2, \ldots, x_d)$ et le vecteur poids $\mathbf{w}$ augmenté du seuil w_0 : $\mathbf{w}^T = (w_0, w_1, w_2, \ldots, w_d)$.

Cette faculté d'appariemment partiel dans lequel c'est l'« alignement » entre l'entrée et l'hypothèse qui décide de la classe de l'entrée est une propriété très intéressante qui n'est pas aussi facile à mettre en œuvre dans les formalismes logiques par exemple. C'est une des raisons de la popularité des fonctions de décision.

2.3 La régression

La régression concerne le cas où $\mathcal{H}$ est un ensemble de fonctions h à valeurs réelles. Une généralisation, la régression multidimensionnelle, est l'apprentissage d'une hypothèse $h : \mathcal{X} \rightarrow \mathbb{R}^n$. On cherche donc à apprendre une fonction à partir d'un ensemble de points et des valeurs

Copyright © 2010 Eyrolles.

que prend cette fonction sur ces points. Il n'y a pas de contre-exemples dans un tel problème d'apprentissage[11].

Il sera en particulier question de régression quand nous verrons l'apprentissage par renforcement, au chapitre 16.

2.4 Les distributions de probabilités

Au lieu de délimiter des frontières de décision sur l'espace $\mathcal{X}$, on peut y définir des distributions de probabilités. Chacune de ces distributions est associée à une classe et détermine la probablité qu'un objet $\mathbf{x} \in \mathcal{X}$ appartienne à cette classe (voir figure 3.11.). Pour qu'il soit aisé de manipuler ces distributions et que l'on puisse contrôler leur pouvoir de généralisation (cf. chapitre 2), elles sont généralement prises au sein de familles paramétrées de distributions, par exemple des fonctions gaussiennes. Nous y reviendrons au chapitre 15.

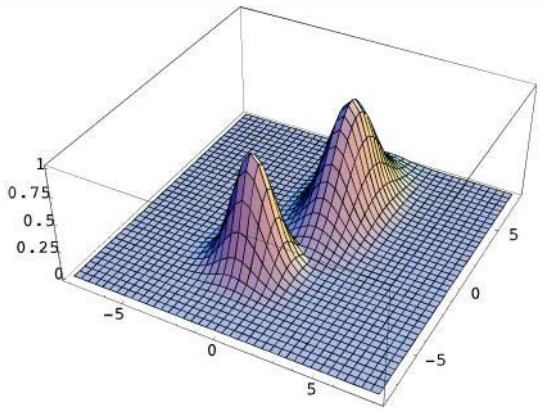


FIG. 3.11: *Un mélange de deux distributions de probabilités correspondant à deux classes d'objets.*

2.5 Les arbres de décision

Lorsque les exemples s'expriment comme des vecteurs d'attributs valeurs, et particulièrement quand ces attributs sont à valeurs discrètes, il est commode de décrire les concepts par des *arbres de décision* comme celui de la figure 3.12. Un arbre de décision prend la description d'un exemple en entrée et lui associe une classe. Chaque nœud de l'arbre correspond à une question portant sur un attribut[12].

De la sorte, en suivant une séquence de nœuds et de branches depuis la racine de l'arbre jusqu'à une feuille, on raffine progressivement la description des exemples concernés jusqu'à obtenir une description correspondant, si tout va bien, aux objets d'une classe. Chaque branche correspond à une conjonction de conditions sur les attributs décrivant les exemples.

Par exemple, dans la figure 3.12, une interprétation est la suivante : « Temps = couvert & température < 10 degrés » → Je reste chez moi)

L'ensemble des branches correspond ainsi à un ensemble de règles d'association décrivant les classes. Le langage défini par les arbres de décision est équivalent à la logique des propositions,

[11] On peut aussi voir la régression comme un problème de classification généralisé, dans lequel le nombre C de classes serait infini.

[12] Ces attributs peuvent aussi être numériques, comme on le verra au chapitre 13. Dans ce cas, on les compare à un seuil et chaque branche est associée à une valeur ou à un intervalle de valeurs possibles pour cet attribut.

Copyright © 2010 Eyrolles.

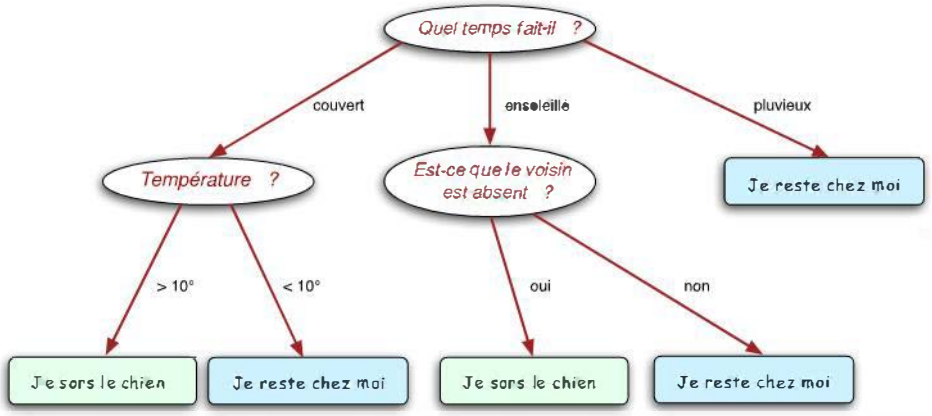


FIG. 3.12: *Un exemple d'arbre de décision. Chaque nœud ovale correspond à une question. Chaque flèche correspond à une réponse possible. Chaque feuille (nœud rectangulaire) est associée à une classe.*

chacun des tests étant une variable booléenne. Toute fonction booléenne peut être exprimée par un arbre de décision.

En revanche, un arbre de décision ne peut pas exprimer un concept *relationnel* comme :

$$\exists x \ \texttt{même-couleur}(x,y) \ \& \ \texttt{envergure}(x,e1) \ \& \ \texttt{envergure}(y,e2) \ \& \ \texttt{plus-petit}(e1,e2)$$

dont la signification est : « les oiseaux x de même couleur qu'un oiseau donné y mais d'envergure inférieure » : ce type de concept appartient à la logique des prédicats (voir le chapitre 5).

Par ailleurs, si certaines fonctions s'expriment de manière économique à l'aide d'arbres de décision, d'autres ne sont pas adaptées à cette représentation. Par exemple la *fonction parité* définie sur un vecteur booléen et qui retourne 1 si et seulement si un nombre pair d'attributs valent 1 s'exprime par un arbre très complexe.

Nous approfondirons cette manière de représenter les concepts au chapitre 13.

2.6 Les hiérarchies de concepts

Les arbres de décision introduisent l'idée de hiérarchie sur les attributs, mais pas sur les concepts. Les attributs placés plus près de la racine sont en quelque sorte plus importants que les attributs plus éloignés.

Il peut être souhaitable d'exprimer explicitement une hiérarchie dans le langage des concepts. C'est le cas par exemple pour les taxonomies de la classification biologique : le concept de `rapace` est situé en dessous du concept `oiseau` et plus haut que `faucon`. Il peut également être intéressant de disposer de relations d'héritage entre un concept et ses sous-concepts. De nombreux types de représentations de concepts peuvent donner lieu à des organisations hiérarchiques, ou ontologies pour peu que l'on explicite les liens de hiérarchie. Cependant de telles organisations sont alors souvent artificielles.

L'idéal est que l'apprenant lui-même construise la hiérarchie et soit prêt à la modifier si de nouvelles informations en indiquent l'utilité. Il existe peu de systèmes d'apprentissage aptes à de telles constructions. On les rencontre généralement en apprentissage non supervisé, quand c'est au système de découvrir des classes dans l'environnement. Le chapitre 18 fournit des précisions à ce sujet.

Copyright © 2010 Eyrolles.

2.7 Les réseaux bayésiens et les modèles graphiques

De nombreux types de dépendances peuvent être représentés à l'aide de structures probabilistes. Lorsque les dépendances et les indépendances entre les variables aléatoires sont explicitées dans un graphe, on utilise le terme de *modèles graphiques* (on trouve aussi les termes de « réseaux de croyance » (*belief networks*), « réseaux causaux » (*causal networks*), « diagrammes d'influence » (*influence diagrams*). La base des calculs effectués dans ces structures est la formule de révision des probabilités de Bayes et pour cette raison ils sont également appelés *réseaux bayésiens*.

La figure 3.13 montre un exemple d'un tel réseau. Les variables sont associées aux nœuds du réseau et les liens manquants entre les nœuds indiquent une certaine indépendance entre ces nœuds (les définitions précises seront données dans le chapitre 11). Les liens entre nœuds sont dirigés, pour indiquer des dépendances causales ou temporelles. Lorsque les liens sont symétriques ou non dirigés, on parle de champs de Markov aléatoires (*random Markov fields*).

Ici les deux variables FN et ZO jouent le rôle de variables causales, CP pouvant découler de FN et/ou de ZO, tandis que SA ne dépend, selon ce réseau, que de FN. Les variables FN et ZO sont affectées de leur probabilité *a priori*, tandis que les variables SA et CP sont associées à des matrices de probabilités conditionnelles indiquant leur dépendance sur les variables FN et ZO.

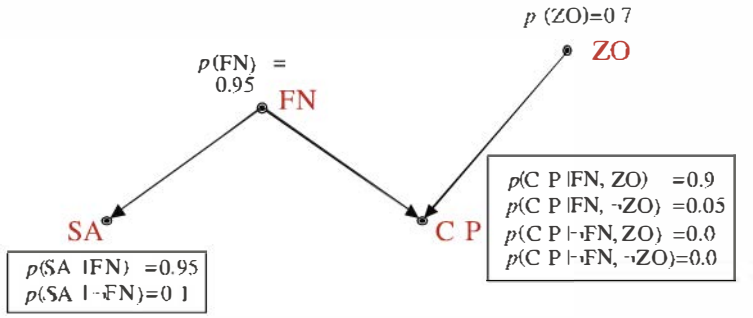


FIG. 3.13: *Exemple de réseau bayésien.*

L'apprentissage peut alors consister à trouver une structure de dépendances entre variables (le graphe) ou à estimer les probabilités conditionnelles définissant ces dépendances. Le chapitre 11 traite en particulier de l'apprentissage de réseaux bayésiens.

2.8 Les chaînes de Markov et les modèles de Markov cachés

Les chaînes de Markov, quant à elles, ne sont qu'un cas particulier de modèles graphiques tels qu'ils viennent d'être décrits. Elles représentent en effet les dépendances temporelles dans une séquence de variables $S_0, \ldots, S_t, \ldots$ Lorsque chaque variable ne dépend que de la variable précédente dans la séquence, on dit que l'on a une *chaîne de Markov* :

$$P(S_t|S_0, \ldots, S_{t-1}) = P(S_t|S_{t-1})$$

Intuitivement, cela peut être interprété comme le fait que le futur ne dépend que de l'état présent, ou du moins que celui-ci possède suffisamment d'informations pour qu'il ne soit pas utile de considérer le passé.

Le formalisme des chaînes de Markov est particulièrement adapté pour la représentation de séquences, qu'elles soient de nature temporelle, comme des cours boursiers ou un signal acoustique, de nature spatiale, comme une chaîne d'ADN, ou d'autres types de dépendances linéaires.

Copyright © 2010 Eyrolles.

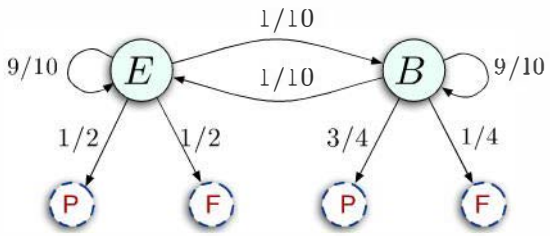


FIG. 3.14: *Une chaîne de Markov à états cachés. Les observations sont le côté de la pièce après lancer : `Pile` ou `Face`. Les états cachés correspondent à la vraie nature de la pièce : équilibrée (E) ou biaisée (B). La probabilité de rester dans le même état est de 9/10, et de 1/10 de changer d'état. Une pièce biaisée produit la face `Pile` pour 3/4 des lancers.*

Une généralisation des chaînes de Markov s'appelle les *modèles de Markov cachés* (*Hidden Markov Models* ou HMM). Formellement, un modèle de Markov caché (d'ordre un) est un modèle génératif de séquences défini par un ensemble d'états, un alphabet discret de symboles, une matrice de probabilités de transitions entre états et une matrice de probabilité d'émissions de chaque symbole de l'alphabet à partir de chaque état. Le système évolue aléatoirement d'un état à l'autre suivant les probabilités de transition en émettant des symboles de l'alphabet (voir par exemple la figure 3.14 pour l'« émission » de `pile` ou `face` par une pièce).

Seuls les symboles émis sont observables, et non les transitions entre états, qui sont internes au modèle. La séquence d'états est donc une séquence de variables cachées ou latentes expliquant les observations.

Trois types de questions au moins peuvent se poser lorsque l'on représente une séquence par un modèle de Markov : quelle est la probabilité d'observation de telle séquence étant donné tel modèle (question relative à la vraisemblance) ? Quelle est la séquence d'états la plus probable dans le modèle de Markov sachant que telle séquence de symboles a été observée (question relative au décodage) ? Finalement, en supposant que les paramètres de transition et d'émission ne soient pas parfaitement connus, comment leurs valeurs devraient être estimées ou révisées à la lumière des séquences de symboles observées (question relative à l'apprentissage) ?

Le chapitre 12 est consacré aux méthodes d'apprentissage adaptées à ce formalisme des modèles de Markov cachés.

2.9 Les grammaires

Quand on a affaire à des séquences d'éléments d'un ensemble nominal, souvent appelé un alphabet dans ce cas, le concept à apprendre doit séparer l'espace de toutes les séquences possibles en deux. C'est ce que fait une grammaire formelle : un compilateur de langage de programmation est un programme qui répond à la question : « est-ce que le programme que l'on vient de me soumettre est correct du point de vue de ma syntaxe ? ». Par exemple, un compilateur du langage `C` répond *VRAI* à la séquence :

```
#include <stdio.h> #include <math.h> int N;double x,res; main()
{N=0;while (N<21){fprintf(stdout, " %f     %f\n",n);N = N+1;}}
```

Il répond *FAUX* à celle-ci :

```
Je sais programmer en C.
```

Copyright © 2010 Eyrolles.

Naturellement, une grammaire du français répondrait exactement le contraire.

Les modèles grammaticaux que l'on sait apprendre ne sont pas en réalité aussi complexes. On verra au chapitre 7 comment on peut par inférence grammaticale généraliser des ensembles de séquences, en particulier sous la forme d'automates finis.

2.10 Les formalismes logiques

La logique des propositions

Si on dispose de l'ensemble des exemples et contre-exemples suivants, décrits par des attributs binaires :

Vole	*A des plumes*	*Pond des œufs*	*Oiseau*	Classe
VRAI	*VRAI*	*VRAI*	*VRAI*	oie
FAUX	*FAUX*	*VRAI*	*FAUX*	ornithorynque
VRAI	*FAUX*	*FAUX*	*FAUX*	rhinolophe
VRAI	*VRAI*	*VRAI*	*VRAI*	cygne

On peut par exemple induire le concept ci dessous :

$$h = [(Vole = VRAI) \wedge (A\ des\ plumes = VRAI) \Rightarrow (Oiseau = VRAI)]$$

Ce concept est écrit dans le langage de la logique des propositions, ou logique booléenne ou encore logique d'ordre 0. Il est *VRAI* pour tous les exemples. Mesurons sa valeur sur les objets suivants :

Vole	*A des plumes*	*Pond des œufs*	*Oiseau*	Classe	valeur de h
VRAI	*VRAI*	*VRAI*	*VRAI*	moineau	*VRAI*
FAUX	*VRAI*	*VRAI*	*VRAI*	autruche	*FAUX*

Le premier objet est représenté correctement par le concept, mais pas le second. À supposer que l'on veuille apprendre le concept « oiseau », la généralisation réalisée ne serait donc ici pas parfaite.

Ce langage de représentation des hypothèses est comme on le voit particulièrement adapté aux exemples représentés par des vecteurs binaires, ou dont la représentation naturelle peut facilement être transformée en vecteurs binaires[13]. Le chapitre 13 traitera en partie de l'apprentissage de ce type d'hypothèses, sous la forme particulière d'arbres de décision. Il en sera également question au chapitre sur l'espace des versions (chapitre 4).

La représentation par attribut-valeur

La logique des propositions peut s'étendre en remplaçant les valeurs binaires des attributs par des valeurs nominales ou hiérarchiques. Les exemples se représentent alors de la manière suivante :

Couleur	*Forme*	*Nombre de pieds*	Classe
rouge	hexagonale	3	tabouret
jaune	carrée	4	tabouret
vert	ronde	4	table
jaune	ovale	6	table

Le langage Attribut-valeur dans lequel on représente le concept appris est semblable à celui de la logique des propositions : on y utilise aussi des conjonctions et des disjonctions,

[13] Par exemple, un attribut continu peut être transformé en attribut binaire par comparaison à un seuil.

Copyright © 2010 Eyrolles.

mais sur des couples $(Attribut, valeur)$. Chaque attribut nominal ou hiérarchique (par exemple $Couleur$, voir la figure 3.1) prend sa valeur dans un ensemble de définition fini, éventuellement partiellement ordonné, comme $\{rouge, vert, jaune, bleu\}$, avec $Couleur-chaude = \{rouge, jaune\}$. Un concept **tabouret** appris dans le langage attribut-valeur pourrait être par exemple :

$$[Couleur = Couleurchaude] \wedge ([Forme = carrée] \vee [Forme = hexagonale])$$

L'intérêt des langages par attribut-valeur est celui des langages typés par rapport aux langages non typés : ils permettent un contrôle plus facile des inférences.

La logique des prédicats

Supposons que nous disposions des données ci-dessous, dont la signification formelle sera donnée au chapitre 5. Pour le moment, nous pouvons en rester à l'interprétation suivante : le numéro 1 ou 2 représente un individu. La relation $fille(1, 2) = VRAI$ s'interprète comme : *l'individu 2 est une fille de l'individu 1*. Les autres relations ont une signification naturelle.

$nom(1) = ève$ $\quad mère(1) = marie$ $\quad père(1) = max$ $\quad homme(1) = FAUX$
$nom(2) = max$ $\quad mère(2) = adèle$ $\quad père(2) = max$ $\quad homme(2) = VRAI$
$fille(1, 2) = VRAI$

D'autre part, le programme d'apprentissage dispose de connaissances *a priori*, comme :

$$\forall X \quad homme(X) = VRAI \Leftrightarrow femme(X) = FAUX$$

$$\forall X \quad homme(X) = FAUX \Leftrightarrow femme(X) = VRAI$$

À partir de ces exemples et de cette *théorie du domaine*, un programme de généralisation en *logique du premier ordre* peut apprendre le concept :

$$\forall X \quad \forall Y \quad (père(Y) = X) \wedge (femme(X) = VRAI) \Rightarrow (fille(X) = Y)$$

La différence avec l'apprentissage en logique des propositions est importante : cette formule est gouvernée par des quantificateurs $\forall$ (*quel que soit*) et $\exists$ (*il existe*) qui sont hors du langage booléen. La généralisation réalisée est par conséquent beaucoup plus radicale et profonde (elle est aussi plus difficile à réaliser). L'apprentissage de tels concepts s'appelle la programmation logique inductive. Il sera développé au chapitre 5.

Les logiques de description

Les logiques de description forment une famille de formalismes de représentation des connaissances dédiée principalement à la gestion automatique de la définition de concepts (ensemble d'individus) et de raisonnement sur ces concepts. Les concepts sont partiellement ordonnés dans une base de connaissances organisée en taxonomie par ce que l'on appelle une relation de subsomption. Basés sur des calculs de subsomption, les principaux mécanismes d'inférence déductive sont la *classification de concept*, qui consiste à insérer automatiquement un concept défini à la place la plus spécifique dans la taxonomie, et la « reconnaissance d'instances » qui consiste à trouver pour un individu donné tous les concepts dont il est instance (voir section 1.3 dans le chapitre 5). De nombreux systèmes de représentation ont été définis dans ce cadre, citons ici le système CLASSIC [BPS94].

2.11 Les règles d'association

Les règles d'association expriment des corrélations présentes dans les données. Un ensemble d'exemples étant fourni, chacun de ceux-ci étant décrit par des attributs (souvent appelés *items*

Copyright © 2010 Eyrolles.

en fouille de données), une règle d'association est une implication de la forme : $X \Longrightarrow Y$, où X et Y sont des attributs ou items.

On dit que cette règle a un *support* s si $s\%$ des exemples incluent tous les items présents dans X et dans Y, et une *confiance* c si $c\%$ des exemples qui contiennent les items de X contiennent aussi les items de Y. La figure 3.15 illustre ces concepts.

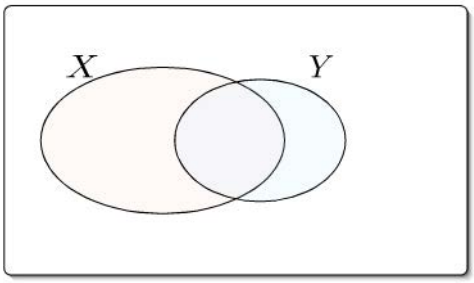


FIG. 3.15: *Support et confiance d'une règle d'association* $X \Longrightarrow Y$. *Le* support *correspond à la proportion d'exemples contenant à la fois les items de* X *et ceux de* Y *dans l'ensemble de tous les exemples. La* confiance *correspond à la proportion des items de* X *qui contiennent aussi les items de* Y.

Nous ne détaillerons pas les algorithmes de recherche de règles d'association ici. Mais, nous renvoyons au chapitre 18.

Remarque

Le but des algorithmes de découverte des règles d'association est de trouver toutes les règles d'association avec un *support* et une *confiance* supérieures à des seuils fixés par l'utilisateur. Des extensions permettant de prendre en compte des taxonomies, des associations quantitatives et des formes temporelles ont aussi été développées.

Les algorithmes de recherche de règles d'association ont généralement deux phases. D'abord, ils cherchent tous les ensembles d'items de support supérieur au seuil fixé (les *itemsets fréquents*, voir section 6.5). Ensuite, les règles peuvent être calculées directement à partir des itemsets fréquents.

Comme les données peuvent être en très grand nombre et impliquer d'énormes quantités de sous-ensembles d'items potentiels parmi lesquels il faut identifier les itemsets fréquents (*frequent item sets* ou FIS, en anglais), la première phase peut être très coûteuse en calculs. C'est pourquoi les recherches sur l'amélioration de son efficacité sont nombreuses. On peut distinguer deux types d'approches : celles qui essaient de réduire le nombre d'itemsets candidats qui ont besoin d'être comptés, et celles qui cherchent à rendre le comptage des candidats plus efficace.

Dans la première famille d'approches, l'identification de la propriété dite d'*antimonotonicité* selon laquelle tous les sous-ensembles des itemsets fréquents doivent aussi être fréquents permet un élagage très efficace. Des travaux ultérieurs se sont focalisés par exemple sur la recherche d'itemsets longs. En effet, pour des itemsets à n items, le nombre de sous-ensembles est de 2^n. Il est cependant possible d'identifier efficacement ces itemsets longs grâce à une technique de *look-ahead* : une fois qu'un itemset a été identifié comme fréquent, aucun de ses sous-ensembles n'a besoin d'être compté. On cherche donc à maximiser la probabilité que les itemsets comptés par *look-ahead* soient effectivement fréquents. Pour ce faire, une bonne heuristique consiste à biaiser la génération des candidats vers des items de grande fréquence puisque ceux-ci ont plus de chance d'être présents dans les itemsets fréquents longs.

Dans la seconde famille d'approches, des tables de hachage emboîtées peuvent être utilisées pour contrôler quels sont les itemsets candidats présents dans un exemple. Cette technique est efficace pour compter des itemsets courts, elle l'est moins pour les itemsets longs. Dans ce cas, on peut faire appel à des techniques de projection de bases de données qui permettent de réduire le nombre d'exemples à considérer.

Copyright © 2010 Eyrolles.

3. Les protocoles d'apprentissage

3.1 Introduction

On appelle *protocole* d'apprentissage les règles du jeu selon lesquelles s'opèrent les interactions entre l'apprenant et son environnement, qui peut comprendre un professeur et même d'autres apprenants, voire des acteurs malveillants. Nous en avons déjà parlé dans la section 4.2.2 du chapitre 1.

L'une des tâches de l'utilisateur est de choisir le protocole le plus adapté à son problème. Par exemple, il peut n'être intéressé que par des prédictions ou décisions et non pas par la formulation d'une hypothèse par l'apprenant. L'apprentissage est alors moins exigeant. Il peut être capable d'attendre que toutes les données soient disponibles avant de procéder à l'apprentissage (apprentissage batch) ou, au contraire, devoir fournir des réponses à n'importe quel moment (apprentissage en ligne ou *anytime learning*). Il se peut qu'un apprentissage actif soit envisageable, dans lequel l'apprenant a une certaine initiative sur les exemples à demander pour l'apprentissage.

3.2 Batch vs. en ligne

Le terme *flot de données* (*data streams*, en anglais) désigne des données arrivant au cours du temps, souvent de manière pratiquement continue. L'analyse de flots de données implique la plupart du temps le traitement à la volée des données sans qu'il soit possible de revenir sur ces données plus tard. C'est le cas de l'analyse de logs sur Internet ou de l'analyse du fonctionnement de réseaux d'ordinateurs. Souvent, l'utilisateur n'est pas alors seulement intéressé dans les relations entre les attributs, mais il cherche aussi à caractériser les évolutions de ces relations au cours du temps.

L'analyse de flots de données soulève d'intéressants problèmes, parmi lesquels le calcul d'agrégats et de résumés statistiques, ainsi que celui de l'*apprentissage incrémental* ou de l'apprentissage *en ligne*. Un défi majeur est celui de l'analyse d'énormes volumes de données, éventuellement hétérogènes, dans des environnements non-stationnaires. Se pose en particulier de manière cruciale la question de l'analyse de données qui ne vérifient pas l'hypothèse i.i.d. (données tirées aléatoirement indépendantes et identiquement distribuées).

Une extension de ce protocole concerne l'« *apprentissage au long cours* » (*long-life learning* ou *continuous learning*) dans lequel l'apprenant est supposé fonctionner sur une longue période de temps et donc faire face à des dérives de l'environnement, à la fois en termes de capteurs et d'informations disponibles, qu'en terme de modification du concept cible. Il est même envisageable que l'apprenant doive changer de tâche. Dans ce cas, il peut être intéressant de tirer parti de points communs entre les domaines et à utiliser de l'*apprentissage par transfert*.

Le chapitre 21 contient davantage d'informations sur ces différents types d'apprentissage.

3.3 Passif vs. actif

Ce qui précède dans l'ouvrage, et en particulier les analyses du chapitre 2, a semblé sanctionner l'idée que l'apprentissage était un processus passif devant les données, c'est-à-dire n'ayant pas d'initiative dans le choix des données d'apprentissage. Il est d'ailleurs exact que l'essentiel de l'analyse théorique de l'apprentissage est bâti sur l'hypothèse que les données sont indépendamment et identiquement distribuées (tirage i.i.d.), tant au cours de l'apprentissage, qu'après, en reconnaissance et décision.

Copyright © 2010 Eyrolles.

Pourtant, tous ceux qui ont joué au jeu de Mastermind[14] ou qui ont fait des expériences scientifiques, savent qu'il est généralement beaucoup moins coûteux, en termes d'expériences à réaliser, de sélectionner soigneusement les questions posées à la nature ou à l'adversaire. Il est donc tentant de doter les systèmes d'apprentissage d'une capacité à sélectionner ou à suggérer les expériences à réaliser. On parle alors d'**apprentissage actif**. L'espoir est évidemment alors de diminuer le coût du recueil de données (par exemple le coût d'expérimentation ou d'étiquetage par un expert). Deux problèmes sont alors à surmonter. D'une part, *sur quel critère sélectionner les données supposées les plus informatives*, d'autre part, *comment fonder ces nouveaux apprentissages pour lesquels l'hypothèse de tirage i.i.d. des données n'est plus respecté*.

Nous reportons le lecteur aux sections 4.2.2 du chapitre 1 et 1 du chapitre 21.

4. L'évaluation de l'apprentissage

Plutôt que de parler de l'apprentissage, nous devrions parler d'apprentissages au pluriel. Le problème de l'évaluation est donc également multiforme. Dans ce chapitre, nous nous concentrerons sur l'apprentissage inductif supervisé à partir d'un échantillon d'apprentissage.

La question est alors : étant donné un algorithme qui retourne des décisions ou bien une hypothèse permettant de calculer une décision pour n'importe quelle entrée de $\mathcal{X}$, comment évaluer la performance de cette règle de décision au sens large ?

Suffit-il par exemple de faire confiance au principe MRE et de se fonder sur la performance mesurée sur l'échantillon d'apprentissage ? Surtout pas, du moins sans précautions. En effet, non seulement la performance en apprentissage, ce que nous avons appelé *risque empirique* dans le chapitre 2, est intrinsèquement optimiste, mais en outre son comportement n'est pas forcément un bon indicateur de la vraie performance (le *risque réel*). Un phénomène classique, déjà présenté au chapitre 1 est schématisé sur la figure 3.16 dans lequel le risque empirique diminue au fur et à mesure que le système prend en compte davantage d'informations (soit par un accroissement du nombre d'exemples présentés, soit par une répétition des exemples d'apprentissage) tandis que le risque réel, d'abord décroissant, se met à augmenter après un certain stade. Ce phénomène est appelée *surapprentissage* ou *suradaptation* ou encore *surajustement* (*over-fitting*).

Le risque empirique ne peut donc à lui seul servir de base à l'estimation de la performance de l'apprentissage réalisé. Comment doit-on alors procéder?

4.1 L'évaluation *a priori* : critères théoriques

Une solution consiste à appliquer les résultats théoriques décrits dans les chapitres 2 et 21 qui fournissent des bornes en probabilité sur le risque réel en fonction du risque empirique. Ces bornes prennent la forme générale :

$$R_{\text{Réel}}(h) \;=\; R_{\text{Emp}}(h) \;+\; \Phi(\mathcal{H}, m)$$

où Φ est une fonction de la richesse de l'espace d'hypothèses $\mathcal{H}$ et m est la taille de l'échantillon d'apprentissage $\mathcal{S}$.

Si on peut obtenir ainsi des bornes asymptotiquement serrées en théorie , elles sont déterminées sous des hypothèses de « pire cas », c'est-à-dire en étant prêt à affronter toute distribution des données et tout concept cible. En pratique, les bornes calculées de cette manière sont généralement trop lâches et ne permettent pas d'estimer précisément la performance réelle. C'est

[14] Jeu dans lequel il faut deviner la position et la couleur d'une séquence de pions cachés par un adversaire.

Copyright © 2010 Eyrolles.

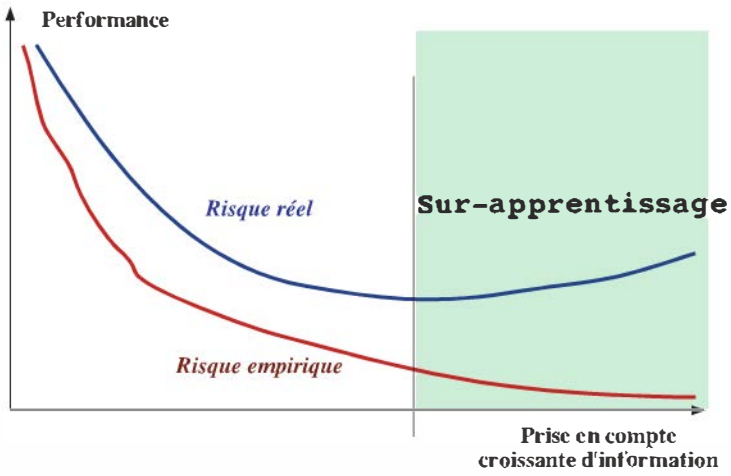


FIG. 3.16: Phénomène de sur-apprentissage. *Tandis que le risque empirique continue de diminuer au fur et à mesure de la prise en compte d'informations, le risque réel qui diminuait également dans un premier temps, commence à réaugmenter après un certain stade. Il n'y a alors plus de corrélation entre le risque empirique et le risque réel.*

pourquoi, sauf cas particuliers favorables, l'estimation de la performance en apprentissage s'opère généralement par des mesures empiriques.

4.2 L'évaluation empirique *a posteriori* : généralités

Quel que soit le principe inductif utilisé pour apprendre une règle de décision, l'algorithme qui cherche cette règle doit pouvoir l'évaluer, en regard du principe inductif, au sein d'un ensemble de possibilités. Ainsi, un algorithme qui s'appuie sur le principe MRE sélectionne une règle qui minimise le risque empirique. Tout algorithme utilise *in fine* une mesure de performance dont une composante est la fidélité aux données d'apprentissage. Cependant, tant la théorie, esquissée dans les chapitres 1 et 2 et développée dans le chapitre 21, que la pratique, montrent qu'il ne faut pas utiliser la performance directement et uniquement mesurée sur l'échantillon de données ayant servi à l'apprentissage pour estimer la vraie performance, c'est-à-dire le risque réel pour ce qui concerne l'apprentissage supervisé.

4.3 L'estimation du risque réel d'une hypothèse

Comme nous l'avons vu, le risque réel est souvent considéré comme le critère de performance le plus important d'un système d'apprentissage. Il est donc essentiel de pouvoir l'estimer le plus précisément possible, ce qui, en général, requiert l'utilisation des exemples d'apprentissage disponibles puisqu'ils sont supposés être notre (unique) source d'information sur la distribution vraie des données. Le problème souvent rencontré est que l'échantillon d'apprentissage est de taille limitée, pour ne pas dire réduite. Cela limite la précision des estimations possibles, mais, surtout, nécessite d'opérer des arbitrages entre la part de l'échantillon utilisée pour l'apprentissage et celle qui sert à l'estimation.

L'approche la plus simple consiste à utiliser toutes les données d'apprentissage, à la fois pour l'apprentissage et pour l'estimation de la performance. C'est ce que l'on appelle la *méthode de resubstitution*. Cette méthode est optimiste et conduit facilement à du sur-apprentissage.

C'est pour éviter ce biais que l'on opère une distinction en général entre les données utilisées

Copyright © 2010 Eyrolles.

pour l'apprentissage et les données utilisées pour le test. Il existe différentes méthodes pour opérer cette distinction, exposées dans la suite de cette section.

4.3.1 L'utilisation d'un échantillon de test

Présentation

La méthode la plus simple pour estimer la qualité objective d'une hypothèse d'apprentissage h est de couper l'ensemble des exemples en deux ensembles indépendants : le premier, noté $\mathcal{L}$, est utilisé pour l'apprentissage de h et le second, noté $\mathcal{T}$, sert à mesurer sa qualité. Ce second ensemble est appelé *échantillon (ou ensemble d'exemples) de test*. On a $\mathcal{S} = \mathcal{L} \cup \mathcal{T}$ et $\mathcal{L} \cap \mathcal{T} = \emptyset$ Comme nous allons le voir, la mesure des erreurs commises par h sur l'ensemble de test $\mathcal{T}$ est une estimation du risque réel d'erreur de h. Cette estimation se note :

$$\widehat{R}_{\text{Réel}}(h)$$

Examinons d'abord le cas particulier de l'apprentissage d'une règle de classification.

Le cas de la classification

Rappelons d'abord la définition d'une matrice de confusion (C est ici le nombre de classes) :

Définition 3.3 (Matrice de confusion)

La matrice de confusion $M(i,j)$ d'une règle de classification h est une matrice $C \times C$ dont l'élément générique donne le nombre d'exemples de l'ensemble de test $\mathcal{T}$ de la classe i qui ont été classés dans la classe j.

Dans le cas d'une classification binaire, la matrice de confusion est donc de la forme :

Classe réelle / *Classe estimée*	+ (P)	− (N)
+	**Vrais positifs** (VP)	Faux positifs (FP)
−	Faux négatifs (FN)	**Vrais négatifs** (VN)

Si toutes les erreurs sont considérées comme également graves, la somme des termes non diagonaux de M, divisée par la taille t de l'ensemble de test, est une estimation $\widehat{R}_{\text{Réel}}(h)$ sur $\mathcal{T}$ du risque réel de h.

$$\widehat{R}_{\text{Réel}}(h) = \frac{\Sigma_{i \neq j} M(i,j)}{t}$$

En notant t_{err} le nombre d'objets de l'ensemble de test mal classés, on a donc :

$$\widehat{R}_{\text{Réel}}(h) = \frac{t_{err}}{t}$$

Le point de vue de la statistique

En statistique, un problème central est celui d'estimer si les mesures effectuées doivent s'expliquer par une hypothèse d'intérêt (notée conventionnellement H_1), par exemple tel antibiotique est efficace, ou si elles peuvent s'expliquer par une *hypothèse nulle* (H_0), par exemple ne mettant pas en jeu l'effet des antibiotiques.

On obtient alors une matrice de confusion entre les hypothèses H_0 et H_1. Un test d'hypothèse revient alors à prendre une décision entre H_0 et H_1. Deux types d'erreurs sont alors distingués :

Copyright © 2010 Eyrolles.

- L'*erreur de première espèce* qui consiste à rejeter H_0 alors que H_0 est vraie. On note la probabilité de cette erreur α.
- L'*erreur de seconde espèce* qui consiste à accepter H_0 alors que H_1 est vraie. On note la probabilité de cette erreur β. La probabilité complémentaire, $1 - \beta$, est appelée *puissance du test.*

L'intervalle de confiance de l'estimation

Quelle confiance peut-on accorder à l'estimation $\widehat{R}_{\text{Réel}}(h)$? Peut-on la traduire numériquement ? La réponse à ces deux questions est donnée de manière simple par des considérations statistiques classiques. Si les échantillons aléatoires d'apprentissage et de test sont indépendants alors la précision de l'estimation ne dépend que du nombre t d'exemples de l'ensemble de test et de la valeur de $\widehat{R}_{\text{Réel}}(h)$.

Il est démontré à l'annexe 2 qu'une approximation suffisante dans le cas où t est assez grand (au delà de la centaine) donne l'*intervalle de confiance* de $\widehat{R}_{\text{Réel}}(h)$ à x % par la formule :

$$\left[\frac{t_{err}}{t} \pm \zeta(x) \sqrt{\frac{\frac{t_{err}}{t}\left(1 - \frac{t_{err}}{t}\right)}{t}} \right]$$

La fonction $\zeta(x)$ a en particulier les valeurs suivantes :

x	50 %	68 %	80 %	90 %	95 %	98 %	99 %
$\zeta(x)$	0.67	1.00	1.28	1.64	1.96	2.33	2.58

EXEMPLE Calcul d'intervalle de confiance

Par exemple, pour $t = 300$ et $t_{err} = 15$, on a $\widehat{R}_{\text{Réel}}(h) = 0.2$ et l'intervalle de confiance à 95 % de $\widehat{R}_{\text{Réel}}(h)$ vaut :

$$\left[0.2 \pm 1.96\sqrt{\frac{0.2(1 - 0.2)}{300}}\right] \approx [0.25, 0.15]$$

ce qui signifie que la probabilité que $R_{\text{Réel}}(h)$ soit dans cet intervalle est supérieure à 95 %. Si on avait obtenu la même proportion d'erreur sur un échantillon de test de taille 1000, cet intervalle aurait été réduit environ de moitié : [0.225,0.175].

L'estimation du taux d'erreur réel par une mesure sur un échantillon de test $\mathcal{T}$ indépendant de l'échantillon d'apprentissage $\mathcal{A}$, fournit une estimation non biaisée de $R_{\text{Réel}}(h)$ avec un intervalle de confiance contrôlable, ne dépendant que de la taille t de l'échantillon de test. Plus celle-ci est grande, plus l'intervalle de confiance est réduit et par conséquent plus le taux d'erreur empirique donne une indication du taux d'erreur réel.

Notons qu'il est absolument **indispensable d'accompagner les courbes d'apprentissage** relatives à un algorithme et un problème d'apprentissage **des intervalles de confiance correspondants.**

4.3.2 L'estimation par validation croisée

Lorsque la taille de l'échantillon de données étiquetées est faible, il faut faire face à un dilemme. Si l'échantillon retenu pour l'apprentissage est trop petit, le système peut ne pas atteindre le niveau de performance qui aurait été possible avec toutes les données. Il faudrait donc garder

Copyright © 2010 Eyrolles.

un maximum d'exemples pour l'apprentissage. D'un autre côté, si l'échantillon de test est trop réduit, il devient impossible d'estimer précisément la performance du système. Il existe un moyen de sortir de ce dilemme.

L'idée de la *validation croisée* à N plis (*N-fold cross-validation*) consiste à :

1. Diviser les données d'apprentissage $\mathcal{S}$ en N sous-échantillons de tailles égales.
2. Retenir l'un de ces échantillons, disons de numéro i, pour le test et apprendre sur les $N-1$ autres.
3. Mesurer le taux d'erreur empirique $\widehat{R}^i_{\text{Réel}}(h)$ sur l'échantillon i.
4. Recommencer N fois en faisant varier l'échantillon i de 1 à N.

L'erreur estimée finale est donnée par la moyenne des erreurs mesurées :

$$\widehat{R}_{\text{Réel}}(h) = \frac{1}{N}\sum_{i=1}^{N} \widehat{R}^i_{\text{Réel}}(h)$$

.

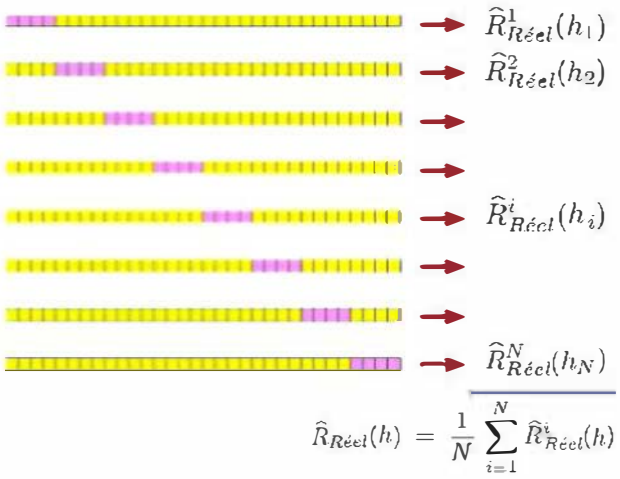


FIG. 3.17: *Validation croisée.*

On peut montrer que cette procédure (voir figure 3.17) fournit une estimation non biaisée du taux d'erreur réel. Il est courant de prendre pour N des valeurs comprises entre 5 et 10. De cette manière, on peut utiliser une grande partie des exemples pour l'apprentissage tout en obtenant une mesure précise du taux d'erreur réel. En contrepartie, il faut réaliser la procédure d'apprentissage N fois.

La question se pose cependant de savoir **quelle hypothèse apprise on doit finalement utiliser**. Il est en effet probable que chaque hypothèse apprise dépende de l'échantillon i utilisé pour l'apprentissage et que l'on obtienne donc N hypothèses différentes.

Deux approches sont alors souvent utilisées. L'idée générale est que la procédure de validation croisée sert seulement à estimer la performance que l'on peut atteindre en utilisant l'espace d'hypothèse $\mathcal{H}$ dans lequel on été prises les différentes hypothèses.

1. La **première** consiste à réapprendre une hypothèse finale en utilisant cette fois la totalité de l'échantillon d'apprentissage. On suppose que l'on obtient ainsi l'hypothèse la meilleure possible étant données toutes les informations disponibles, tout en ayant une estimation de la performance à en attendre.
2. La **deuxième** approche consiste à utiliser les N hypothèses h_i apprises et à les faire voter.

Copyright © 2010 Eyrolles.

Notons d'emblée que si les hypothèses apprises sont très différentes les unes des autres (en supposant que l'on puisse mesurer cette différence), c'est qu'il faut peut-être y voir une indication de l'inadéquation de l'espace des hypothèses $\mathcal{H}$. Cela semble en effet montrer une grande variance (en général associée à une grande dimension de Vapnik-Chervonenkis), et donc le risque d'un apprentissage sans valeur (voir la section 4.2 dans le chapitre 2).

4.3.3 L'estimation par la méthode du *leave-one-out*

Lorsque les données disponibles sont très peu nombreuses, il est possible de pousser à l'extrême la méthode de validation croisée en prenant pour N le nombre total d'exemples disponibles. Dans ce cas, on ne retient à chaque fois qu'un seul exemple pour le test, et on répète l'apprentissage N fois pour tous les autres exemples d'apprentissage (voir figure 3.18).

L'estimation par la méthode du *leave-one-out* est (presque) non biaisée. Si elle conduit aussi généralement à une moindre variance sur les hypothèses produites que la validation croisée (laissant de côté plus qu'un seul exemple pour le test), en revanche, la variance sur l'estimation de l'erreur, ou du risque réel, est plus élevée. De plus, elle tend à sous-estimer l'erreur si les données ne sont pas vraiment identiquement et indépendamment distribuées. Par ailleurs, si elle est simple à mettre en œuvre, elle est coûteuse en calculs [GE03].

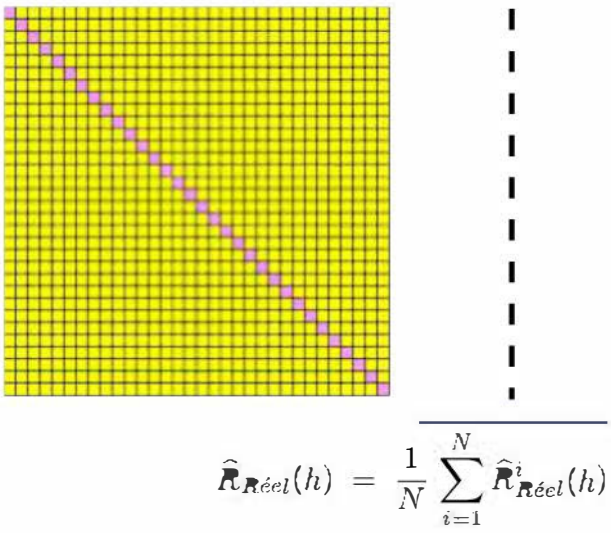


FIG. 3.18: *Estimation de la performance par méthode du leave-one-out.*

4.3.4 Quelques variantes de la méthode de validation croisée : *bootstrap*, *jackknife*

Ces techniques diffèrent des précédentes en ce qu'elles utilisent des tirages *avec remise* dans l'ensemble des exemples. Le procédé est le suivant : on tire aléatoirement un exemple, pour le placer dans un ensemble appelé *boostrap*[15]. Le procédé est répété n fois et l'apprentissage est alors effectué sur l'ensemble bootstrap. Un test est mené sur les exemples non présents dans cet ensemble, donnant une première valeur P_1 des erreurs du classificateur. Un autre test est réalisé sur l'ensemble complet des exemples, donnant la valeur P_2. L'ensemble de l'opération est répété K fois. Une certaine combinaison linéaire de la moyenne $\overline{P_1}$ des valeurs P_1 et de la moyenne $\overline{P_2}$

[15] On sait que le Baron de Munchausen savait s'élever dans les airs en tirant sur ses bottes. La méthode du même nom donne des résultats tout aussi étonnants (quoique ici justifiés théoriquement et pratiquement).

Copyright © 2010 Eyrolles.

des valeurs P_2 obtenues donne la valeur $\widehat{R}_{\text{Réel}}(h)$. La théorie |HTF01| propose la formule :

$$\widehat{R}_{\text{Réel}}(h) = 0.636 \times \overline{P_1} + 0.368 \times \overline{P_2}$$

en se basant sur le fait que l'espérance de la proportion des éléments non répétés dans l'ensemble de test est égale à 0.368. Pour de petits échantillons, la méthode bootstrap fournit une estimation remarquablement précise de $R_{\text{Réel}}(h)$. En contrepartie, elle demande une grande valeur de K (plusieurs centaines), c'est-à-dire un nombre élevé d'apprentissages de la règle de classification.

Il existe enfin une autre méthode proche mais plus complexe d'estimation appelée *jackknife*[16] qui vise à réduire le biais du taux d'erreur en resubstitution, lorsque des données sont utilisées à la fois pour l'apprentissage et pour le test. Nous renvoyons le lecteur intéressé à |Rip96| pp.72-73 ou à |Web99|. Ce sont également de bonnes références pour le problème de l'estimation de performance en général.

4.4 Divers indices composites de performances

La seule mesure du taux d'erreur peut ne pas être pertinente pour une tâche d'apprentissage donnée.

— EXEMPLE —

Supposons que nous considérions une tâche de discrimination entre deux classes, et qu'après apprentissage, on observe sur un ensemble de test constitué de 105 exemples positifs et 60 exemples négatifs, les performances suivantes :

SVM :

Estimé \ Réel	+	−
+	94	37
−	11	23

Bayésien naïf :

Estimé \ Réel	+	−
+	72	29
−	33	31

Apparemment, le système SVM (voir chapitre 14) est plus performant sur cette tâche, puisque son taux d'erreur est de : $\frac{11+37}{165} = 0.29$ au lieu de $\frac{29+33}{165} = 0.375$ pour le classifieur bayésien naïf (voir chapitre 2). Pourtant, ce critère d'erreur n'est peut-être pas celui qui nous intéresse en priorité.

Par exemple, on peut observer que la classification obtenue ici avec le SVM est biaisée vers la classe majoritaire puisque $94 + 37 = 131$ exemples sont classés +, au lieu de seulement $72 + 29 = 101$ pour le classifieur naïf, qui respecte donc beaucoup mieux les proportions des deux classes. De plus, il se peut que ce qui nous intéresse soit en fait, pour cette tâche, le taux de « vrais négatifs ». Dans ce cas, il faut choisir la classification retournée par le classifieur bayésien naïf avec 31 « vrais négatifs » au lieu de 23 seulement avec le SVM.

Afin d'aider à mesurer la performance pertinente en fonction des grandes tâches d'apprentissage, plusieurs indices composites calculés à partir de la matrice de confusion (répétée ci-dessous) ont été utilisés. Ils incluent :

- Le *taux de bonne prédiction* (*accuracy*) : précision $= \frac{VP+VN}{P+N}$
- Le *taux de vrais positifs* : taux_VP $= \frac{VP}{P}$ (aussi appelé *sensibilité* (*sensitivity*) en recherche médicale par exemple).
- Le *taux de vrais négatifs* : taux_VN $= \frac{VN}{N}$ (aussi appelé *spécificité* (*specificity*) en recherche médicale par exemple).

[16] Ou « couteau suisse » : un outil multifonction bien pratique.

Copyright © 2010 Eyrolles.

- Le *rappel* (*recall*) : $\frac{VP}{P}$
- Le *précision* (*precision*) : $\frac{VP}{VP+FP}$
- La *F_mesure* (*F-mesure*) qui est une moyenne harmonique du rappel et de la précision ($\beta > 0$) :

$$\frac{(1+\beta^2) \cdot rappel \cdot précision}{\beta^2 \cdot rappel + précision}$$

- On confond souvent la *F_mesure* avec la *F1_mesure* :

$$\frac{2 \cdot rappel \cdot précision}{rappel + précision}$$

Classe réelle / *Classe estimée*	+ (P)	− (N)
+	**Vrais positifs (VP)**	Faux positifs (FP)
−	Faux négatifs (FN)	**Vrais négatifs (VN)**

Le *rappel* et la *précision* sont deux mesures populaires en recherche d'information (*information retrieval*) car elles correspondent à deux souhaits importants. On cherche en effet à extraire le maximum des documents effectivement pertinents (*rappel*) tout en maximisant la proportion de documents pertinents dans les documents classés positivement (*précision*). Souvent, ces deux souhaits sont contradictoires.

4.5 Le réglage des algorithmes par un ensemble de test

Lorsque l'on cherche à résoudre un problème d'apprentissage, on cherche à se décider pour la meilleure méthode, ce qui implique :

- le choix du *principe inductif* ;
- le choix d'une *mesure de performance*, ce qui implique souvent celui d'une fonction de coût ;
- le choix d'un *algorithme d'apprentissage* ;
- le choix de l'*espace d'hypothèses*, qui dépend en partie du choix de l'algorithme ;
- le *réglage de tous les paramètres* contrôlant le fonctionnement de l'algorithme.

Généralement l'opérateur essaie plusieurs méthodes sur le problème d'apprentissage afin de déterminer celle qui semble la plus appropriée à la classe de problèmes concernée. Comment doit-il procéder ?

Il faut se méfier d'une approche qui semble naturelle. On pourrait en effet croire qu'il suffit de mesurer pour chaque méthode la performance empirique à l'aide de l'une des techniques décrites plus haut, dans la section 4.3, faisant appel d'une manière ou d'une autre à un échantillon de test. Cela serait cependant commettre la faute de faire dépendre le réglage de la méthode d'apprentissage (ou de ses méta-paramètres) d'un ensemble d'exemples, ici l'ensemble de validation, qui devient à la fois source de réglages et juge de paix pour l'estimation de performance. En procédant de la sorte, on risque d'adapter étroitement la méthode d'apprentissage à cet échantillon de validation. Cela est dangereux car il se peut que, comme dans le cas du phénomène de sur-apprentissage, à tant poursuivre ce but, on s'éloigne d'une diminution du risque réel. C'est pourquoi on prévoit, à côté de l'échantillon d'apprentissage et de l'échantillon de validation, un troisième échantillon indépendant des deux autres, l'*échantillon de test*[17], qui ne servira que pour l'estimation finale de la règle de décision finalement obtenue (voir figure 3.19).

[17] La terminologie est ici mal fixée dans la communauté : certains auteurs définissent « échantillon de test » et « échantillon de validation » à l'inverse. Il faut donc se méfier et vérifier que l'on utilise bien la même convention que son interlocuteur.

Copyright © 2010 Eyrolles.

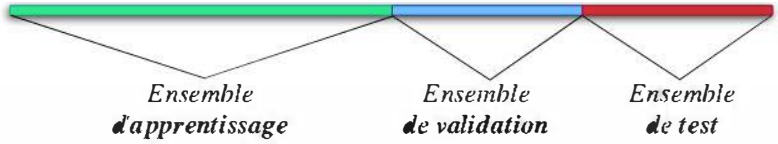


FIG. 3.19: *Échantillons utilisés pour l'apprentissage et son évaluation.*

Malheureusement, dans de nombreuses applications, le nombre d'exemples, c'est-à-dire d'observations pour lesquelles un expert a fourni une étiquette, est limité. Le plus souvent chaque nouvel exemple est coûteux à obtenir et il ne peut donc être question d'augmenter à volonté l'échantillon d'apprentissage, l'échantillon de validation et l'échantillon de test. Il arrive ainsi que l'on utilise une procédure plus complexe de double validation croisée imbriquée, dans laquelle les exemples sont tirés indépendamment pour constituer les trois ensembles.

4.5.1 Estimation de risque : la courbe *ROC*

Jusqu'ici nous avons essentiellement décrit des méthodes d'évaluation des performances ne prenant en compte qu'un nombre : l'estimation du risque réel. Cependant, dans un contexte de prise de décision, il peut être utile d'être plus fin dans l'évaluation des performances et de prendre en compte non seulement un taux d'erreur, mais aussi les taux de « faux positifs » et de « faux négatifs » (disponibles à partir de la matrice de confusion, voir au paragraphe 1.4). Souvent, en effet, le coût de mauvaise classification n'est pas symétrique et l'on peut préférer avoir un taux d'erreur un peu moins bon si cela permet de réduire le type d'erreur le plus coûteux (par exemple, il vaut mieux retirer à tort l'appendice (faux positif), plutôt que de ne pas détecter une appendicite (faux négatif)). La courbe *ROC* (de l'anglais *Receiver Operating Characteristic*) permet de régler ce compromis[18].

Supposons que la fonction de décision s'appuie sur une fonction h à valeur réelle définie sur l'espace des entrées $\mathcal{X}$. On peut alors établir un graphique pour chaque classe donnant la probabilité d'appartenir à cette classe en fonction de la valeur de sortie de la fonction h que l'on appelle ici critère de décision (voir la figure 3.20).

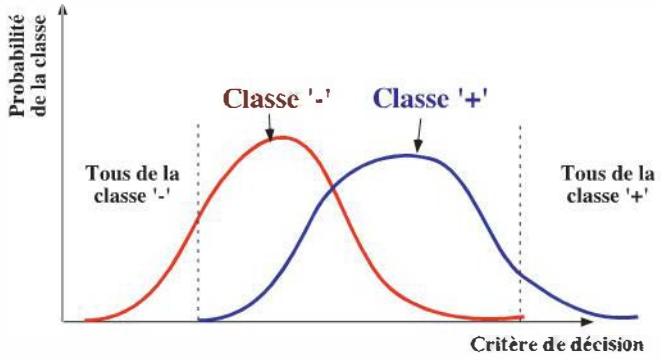


FIG. 3.20: *Courbes de probabilité correspondant aux classes '+' et '−'.*

Comme on le voit sur la figure 3.21, pour chaque seuil de décision, on peut calculer la proportion de « vrais positifs » (fraction des exemples de la classe + étiquetés comme + à partir du

[18] Ces courbes ont été utilisées pour la première fois lors de la deuxième guerre mondiale quand, dans l'usage des radars, on a voulu quantifier leur capacité à distinguer des interférences de nature aléatoire du signal indiquant réellement la présence d'aéronefs.

Copyright © 2010 Eyrolles.

seuil de décision) et de « faux positifs ».

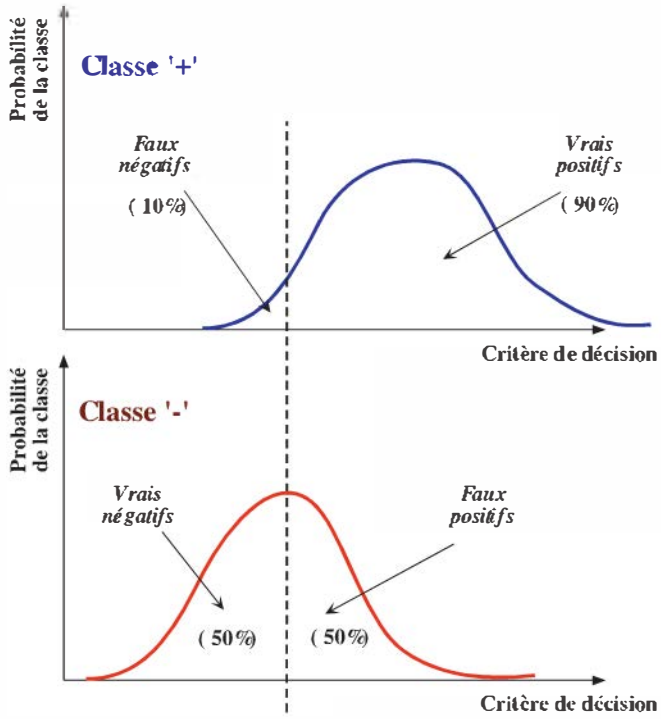


FIG. 3.21: *Seuil décidant pour chaque classe des « vrais positifs », « faux négatifs », « faux positifs » et « vrais négatifs ».*

On peut alors construire la courbe ROC : pour chaque seuil, on reporte la proportion de « vrais positifs » en fonction de celle des « faux positifs ». Il est clair qu'idéalement, si la fonction de décision permettait de séparer complètement les deux courbes de probabilité correspondant aux deux classes (apprentissage parfait), on pourrait obtenir 100 % de vrais positifs pour 100 % de faux positifs pour un seuil donné passant entre les deux courbes. En revanche, si les deux courbes de probabilité se superposent parfaitement (aucune discrimination), alors, pour chaque seuil de décision, il y a autant de « vrais positifs » que de « faux positifs ». La courbe ROC est alors la droite correspondant à la fonction identité. Le test a 50 % de chances de conduire au bon diagnostic. Plus la courbe s'incurve vers le haut, plus le test est pertinent (le rapport des « vrais positifs » sur les « faux positifs » augmente). La pertinence est mesurée par l'aire sous la courbe; elle augmente avec sa courbure (voir la figure 3.22).

Lorsque l'on a trouvé un système de classification suffisamment bon, il reste à choisir le seuil pour un diagnostic 'classe +' / 'classe −'. Le choix du seuil doit fournir une proportion de vrais positifs élevée sans entraîner une proportion inacceptable de faux positifs. Chaque point de la courbe représente un seuil particulier, allant du plus sévère : limitant le nombre de faux positifs au prix de nombreux exemples de la classe '+' non diagnostiqués (forte proportion de faux négatifs, c'est-à-dire faible proportion de vrais positifs), aux plus laxistes : augmentant le nombre de vrais positifs au prix de nombreux faux positifs (voir la figure 3.22). Le seuil optimal pour une application donnée dépend de facteurs tels que les coûts relatifs des faux positifs et faux négatifs, comme de celui de la prévalence de la classe '+'. Par exemple, un opérateur (de téléphonie ou de chaîne câblée) cherche à détecter des problèmes de résiliation : quels sont les abonnés susceptibles

Copyright © 2010 Eyrolles.

de le quitter ?[19]. Ces abonnés fuyants sont peu nombreux, mais très coûteux. On cherchera donc à essayer d'en détecter le maximum afin de tenter de les retenir, quitte à détecter aussi quelques faux. On utilisera alors un seuil « laxiste ».

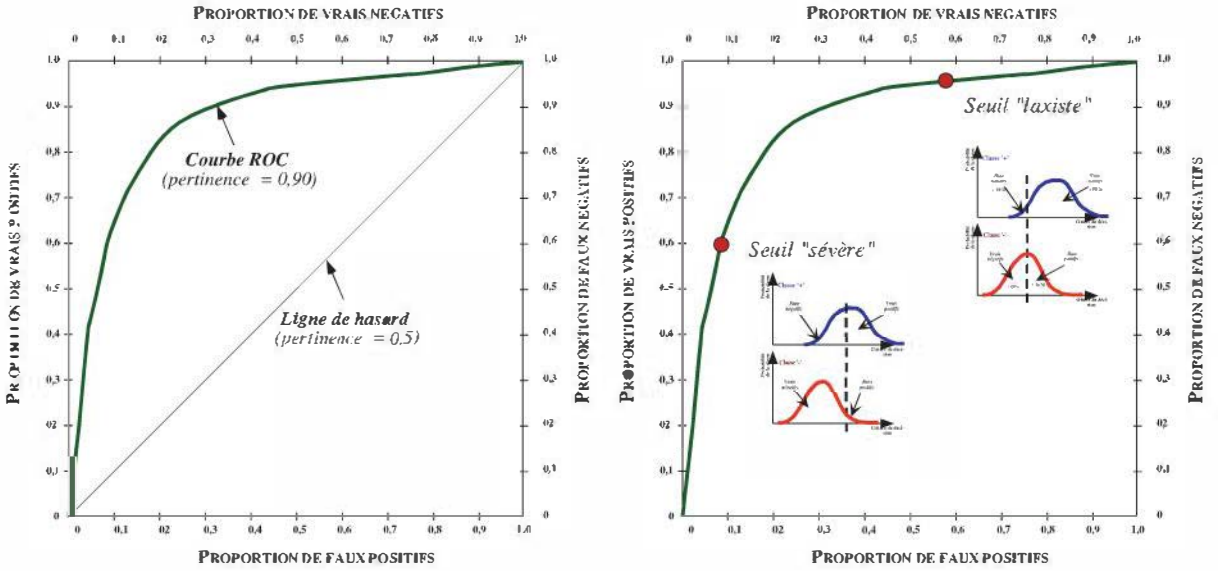


FIG. 3.22: *Une courbe ROC à gauche. Deux seuils sur cette courbe à droite.*

On peut comparer plusieurs systèmes d'apprentissage en comparant leurs courbes ROC. Si, pour une tâche donnée, un système a une courbe uniformément supérieure à celle de l'autre système, alors on peut conclure qu'il a un meilleur pouvoir discriminant. En revanche, il peut arriver que les courbes ROC correspondant aux deux systèmes se chevauchent (courbe 3.23). Dans ce cas, chaque système est supérieur à l'autre pour un intervalle de valeur du critère de décision, et le choix doit s'opérer sur la base des coûts relatifs des « faux positifs » et des « faux négatifs ».

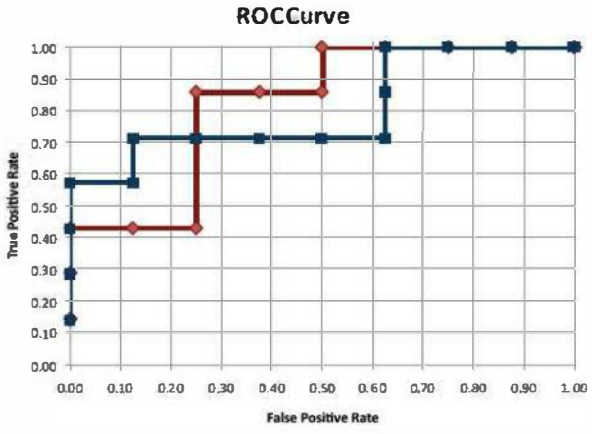


FIG. 3.23: *Courbes ROC correspondant à deux systèmes qui se croisent.*

[19] On les appelle les *churners* (de *churn* : baratte) dans le jargon du métier. On parle aussi d'attrition.

Copyright © 2010 Eyrolles.

4.6 D'autres critères d'appréciation

En plus des critères numériques, il existe un certain nombre de qualités qui permettent de distinguer une hypothèse parmi d'autres.

L'intelligibilité des résultats d'apprentissage

Dans le cas où l'hypothèse apprise est une grammaire ou un ensemble de règles logiques, par exemple, elle peut être munie d'une sémantique directement interprétable dans le langage de l'ingénieur ou de l'expert. Il est important dans ce cas qu'elle soit compréhensible. Cette faculté d'intelligibilité a déjà été évoquée dans le premier chapitre de ce livre : nous y avons fait remarquer que la discipline de l'extraction de connaissances dans les données faisait grand cas de cette intelligibilité et que parfois l'apprentissage d'un petit nombre de règles compréhensibles valait mieux qu'un fouillis de règles sophistiquées, même avec une performance objective supérieure.

La simplicité des hypothèses produites

Ce critère est relié au précédent. Il relève d'un argument rhétorique classique, le *rasoir d'Occam*, qui affirme qu'il ne sert à rien de multiplier les « entités » inutiles[20], autrement dit qu'une explication simple vaut mieux qu'une explication compliquée. Ce principe a été rationnalisé par certains chercheurs dans le domaine de la théorie de l'apprentissage [LV97].

5. La comparaison des méthodes d'apprentissage

On peut toujours utiliser différents algorithmes d'apprentissage sur une même tâche. Comment interpréter la différence de performance mesurée empiriquement entre deux algorithmes ? Plus concrètement, est-ce qu'un algorithme dont la performance en taux d'erreur de classification binaire vaut 0.17 est meilleur qu'un autre dont la performance mesurée est de 0.20?

La réponse n'est pas évidente, car la performance mesurée dépend à la fois des caractéristiques des tests empiriques effectués et des échantillons de tests utilisés. Départager deux systèmes sur une seule mesure est donc problématique. La même question se pose d'ailleurs pour deux hypothèses produites par le même algorithme à partir de conditions initiales différentes.

Toute une littérature porte sur ce problème et nous n'en esquissons que les grandes lignes dans ce qui suit. On ne saurait être trop attentif à ces questions lorsque l'on teste effectivement des systèmes, sous peine d'affirmer à peu près n'importe quoi.

Notons que le problème de décider si un algorithme ou une hypothèse est meilleure qu'une autre se place naturellement dans le *cadre du test d'hypothèses en statistique*, que nous avons déjà évoqué plus haut dans la section 4.3.1. Il n'est donc pas surprenant que la plupart des techniques de comparaison viennent du monde des statistiques.

Le principe général est de caractériser la distribution de probabilité sur les différences possibles de performance sous l'hypothèse qu'il n'y a pas en fait de différence entre les méthodes (hypothèse nulle : H_0). On compare alors la différence observée par rapport à cette courbe, et on décide si la probabilité que cette différence ait été observée sous H_0 est suffisamment petite pour que

[20] *Frustra fit per plura, quod fieri potest per pauciora*, classiquement traduit par : *It is vain to do with more what can be done with less.* Ou *Essentia non sunt multiplicanda praeter necessitatem. Entities should not be multiplied unnecessarily.* Guillaume d'Occam (1288-1348).

Copyright © 2010 Eyrolles.

l'on décide qu'il y a bien en fait une différence entre les algorithmes ou entre les hypothèses comparées.

Lors de la comparaison entre deux algorithmes, ou de deux hypothèses produites par le même algorithme à partir de conditions initiales différentes, il faut distinguer le cas où l'on se base sur des échantillons de test identiques ou différents. Nous allons traiter rapidement les deux cas.

5.1 La comparaison de deux algorithmes à partir d'un même jeu de données

5.1.1 La comparaison de deux algorithmes sur un même ensemble de test

Si les ensembles tests sur lesquels sont évalués les deux algorithmes sont les mêmes, alors les intervalles de confiance peuvent être beaucoup plus serrés que si on les évalue sur des ensembles de test différents dans la mesure où l'on élimine la variance due à la différence entre les échantillons de test.

Soient donc deux hypothèses de classification h_1 et h_2. Notons :

n_{00} = Nombre d'exemples de tests mal classés par h_1 et h_2
n_{01} = Nombre d'exemples de tests mal classés par h_1, mais pas par h_2
n_{10} = Nombre d'exemples de tests mal classés par h_2, mais pas par h_1
n_{11} = Nombre d'exemples de tests correctement classés par h_1 et h_2.

Posons comme hypothèse nulle que les deux classifieurs ont le même taux d'erreur en généralisation. Nous nous attendons alors à observer que $n_{01} = n_{10} = \frac{n_{01}+n_{10}}{2}$.

On peut alors avoir recours au test du χ^2 qui consiste à examiner l'écart quadratique entre la valeur attendue d'une variable aléatoire et la moyenne observée et à diviser cet écart par la valeur attendue.

$$\frac{\left(n_{01} - \frac{n_{01}+n_{10}}{2}\right)^2}{\frac{n_{01}+n_{10}}{2}} + \frac{\left(n_{10} - \frac{n_{01}+n_{10}}{2}\right)^2}{\frac{n_{01}+n_{10}}{2}} = \frac{(n_{01}+n_{10})^2}{n_{01}+n_{10}}$$

qui suit une loi du chi-2 à un degré de liberté.

Le *test de McNemar* est un test du chi-2 corrigé à un degré de liberté :

$$\frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} \sim \chi_1^2 \tag{3.5}$$

Il accepte l'hypothèse nulle, pas de différence entre les performances des deux classifieurs, au niveau α si cette valeur est inférieure ou égale à $\chi^2_{\alpha,1} \cdot \chi^2_{0.05,1} = 3.84$.

5.1.2 Validation croisée à N plis avec t tests couplés

L'idée de ce test est d'utiliser un mécanisme de validation croisée pour obtenir N couples d'ensembles d'apprentissage et de test : $(\mathcal{S}_i, \mathcal{T}_i)_{i=1,\ldots,N}$. Les fréquences d'erreur observées pour les deux classifieurs sur l'ensemble de test $\mathcal{T}_i$ sont notées : p_i^1 et p_i^2 respectivement.

Si les deux classifieurs ont des performances équivalentes, on s'attend à ce que les variables p_i^1 et p_i^2 aient mêmes moyennes, ou encore que la différence de leurs moyennes soient nulle.

Soit $p_i = p_i^1 - p_i^2$ cette différence. Étant donné que l'on suppose que les variables p_i^1 et p_i^2 suivent approximativement des lois normales, leur différence p_i suit aussi, approximativement, une loi normale de moyenne μ.

On définira donc l'hypothèse nulle H_0, correspondant à l'absence de différence entre les deux classifieurs, comme : $\mu = 0$.

Copyright © 2010 Eyrolles.

Soient la moyenne et la moyenne des écarts quadratiques :

$$m = \frac{\sum_{i=1}^{N} p_i}{N} \quad \text{et} \quad S^2 = \frac{\sum_{i=1}^{N} (p_i - m)^2}{N-1}$$

Sous l'hypothèse nulle, nous avons une statistique suivant une distribution t à $N-1$ degrés de liberté :

$$\frac{\sqrt{N}(m - 0)}{S} = \frac{\sqrt{N} \cdot m}{S} \sim t_{N-1} \tag{3.6}$$

Le test accepte l'hypothèse nulle, pas de différence entre les performances des deux classifieurs, au niveau α si cette valeur est dans l'intervalle $[-t_{\alpha/2,N-1}, t_{\alpha/2,N-1}]$. On a par exemple $t_{0.025,9} = 2.26$ et $t_{0.025,29} = 2.05$.

Notons que même si ce test fait l'hypothèse que les deux variables p_i^1 et p_i^2 suivent des lois normales, il est robuste aux écarts avec cette hypothèse. D'où l'utilisation très courante de ce test.

5.1.3 Les 5×2 cv t tests couplés

Dans son article de 1998, [Die98], Dietterich remarque que l'inconvénient de la validation croisée à N plis avec tests couplés quand N est grand est qu'il y a un fort recoupement des ensembles d'apprentissage ce qui peut conduire à une sous estimation de la variance.

Le test qu'il propose utilise cinq réplications d'une validation croisée à deux plis. À chaque réplication, le jeu de données est divisé en deux parties égales. On note $p_i^{(j)}$ la différence entre les taux d'erreur des deux classifieurs pour le pli $j \in \{1, 2\}$ et la réplication $i \in \{1, \ldots, 5\}$. Pour chaque réplication i, la moyenne des taux d'erreur est $\bar{p}_i = (p_i^{(1)} + p_i^{(2)})/2$, et la variance estimée est $s_i^2 = (p_i^{(1)} - \bar{p}_i)^2 + (p_i^{(2)} - \bar{p}_i)^2$.

Soit l'hypothèse nulle H_0 selon laquelle les deux algorithmes ont la même performance. La différence $p_i^{(j)}$ de deux variables identiquement distribuées, en ignorant ici qu'elles ne sont pas indépendantes, peut être considérée comme suivant une loi normale de moyenne 0 et de variance inconnue σ^2. Dans ce cas, le rapport s_i^2/σ^2 suit une distribution chi-2 (χ^2) à un degré de liberté, et leur somme suit une distribution du χ^2 à cinq degrés de liberté :

$$M = \frac{\sum_{i=1}^{5} s_i^2}{\sigma^2} \sim \chi_5^2 \quad \text{et} \quad t = \frac{p_1^{(1)}}{\sqrt{M/5}} = \frac{p_1^{(1)}}{\sqrt{\sum_{i=1}^{5} s_i^5/5}} \sim t_5 \tag{3.7}$$

ce qui donne une t-statistique à 5 degrés de liberté.

Le test 5×2 cv t tests couplés accepte l'hypothèse nulle, pas de différence entre les performances des deux classifieurs, au niveau α si cette valeur est dans l'intervalle $[-t_{\alpha/2.5}, t_{\alpha/2.5}] \cdot t_{0.025,5} = 2.57$.

On peut aussi utiliser ce test pour décider si un algorithme est supérieur à un autre, pour une tâche donnée, dans une version à un côté (*one-sided*). Ainsi, on décidera que l'algorithme 1 est inférieur à l'algorithme 2 en comparant les hypothèses : $H_0 : \mu \leq 0$ et $H_1 : \mu > 0$.

On calcule les mêmes statistiques que précédemment (équation 3.7) et l'hypothèse nulle est acceptée si la valeur calculée est inférieure à : $t_{\alpha,5} \cdot t_{0.05,5} = 2.02$.

Alpaydin, [Alp99], a amélioré ce test pour le rendre plus robuste et plus performant.

5.2 La comparaison de deux algorithmes à partir de jeux de données différents

Il existe une différence fondamentale entre les tests effectués à partir d'un jeu de données unique et les tests portant sur plusieurs jeux de données. Les tests sur un seul jeu de données

Copyright © 2010 Eyrolles.

impliquent généralement l'estimation de la performance moyenne et de sa variance sur des étapes répétées d'apprentissage et de test avec des tirages aléatoires à partir du jeu de données. Comme ces ensembles, d'apprentissage comme de test, ne sont pas indépendants, un grand soin doit être apporté afin d'éviter des estimations biaisées.

Il est cependant fréquent que l'on veuille comparer des algorithmes sur des jeux de données différents. Dans ce cas, la variance des résultats est en partie attribuable à la différence des jeux de données. Demsar dans une étude très intéressante, [Dem06], avertit que le t-test n'est plus une méthode appropriée, et recommande le test de rangs signés de Wilcoxon.

5.2.1 Le test de rangs signés de Wilcoxon

De nombreux tests statistiques s'appuient sur des *a priori* sur la forme de la distribution des mesures effectuées. Par exemple, le t-test étudié plus haut suppose que les mesures suivent une loi normale. Il peut cependant arriver que nous nous sentions inconfortables à l'idée d'émettre de telles suppositions. Dans ce cas, il faut avoir recours aux *tests non paramétriques*. Les tests basés sur la comparaison des rangs font partie de ces tests.

L'idée est ici de remplacer les valeurs numériques des taux d'erreur observés sur les classifieurs comparés et les jeux de test par les rangs des performances mesurées. Si les classifieurs sont de performance comparable, on devrait obtenir des rangs à peu près bien mélangés. En revanche, si un classifieur est significativement supérieur à un autre, alors les rangs correspondant aux taux d'erreur qu'il obtient sur les jeux de données devraient être biaisés vers les premiers rangs, c'est-à-dire les taux d'erreur les plus faibles. De même, si un classifieur a une variance de ses résultats supérieure à la variance de l'autre classifieur, il devrait obtenir à la fois plutôt des rangs élevés et des rangs bas. Il existe des tests statistiques spécifiques sur les rangs moyens ou sur la distribution des rangs permettant de calculer le degré auquel on peut croire, ou non, à la supériorité d'un algorithme sur l'autre.

Les test de rangs signés (*signed rank tests*) s'appuient sur un autre type de classement. Ici, on classe l'écart entre les performances des deux classifieurs testés sur les différents jeux de test, et pour chaque écart $|R_{\text{Emp}}(A) - R_{\text{Emp}}(B)|$ (noté $|\tau_A - \tau_B|$ sur la figure 3.24), on retient l'algorithme gagnant. Ainsi, si l'algorithme A est le gagnant pour le plus grand écart, alors il a gagné pour le rang 1, et ainsi de suite.

S'il n'y avait pas de différence significative de performance entre les deux algorithmes, on devrait obtenir des rangs moyens à peu près égaux pour les deux algorithmes. Soit N le nombre de jeux de tests, G_A (resp. G_B) la somme des rangs sur lesquels l'algorithme A (resp. B) a gagné, et $T = \min(G_A, G_B)$. Pour $N \geq 25$, la statistique

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}$$

est distribuée à peu près selon une loi normale. Avec $\alpha = 0.05$, l'hypothèse nulle, les deux algorithmes ont des performances équivalentes, peut être rejetée si $z < -1.96$.

5.3 La comparaison de multiples algorithmes

Lorsque l'on compare entre eux davantage que deux algorithmes, on se retrouve dans la situation de tests multiples, ce qui demande un soin particulier, comme devraient le savoir les praticiens de l'apprentissage, conscients du problème du lien entre mesure du risque empirique sur l'apparente meilleure hypothèse et son risque réel. Pourtant, de nombreuses publications et études sont entachées d'erreurs méthodologiques.

Copyright © 2010 Eyrolles.

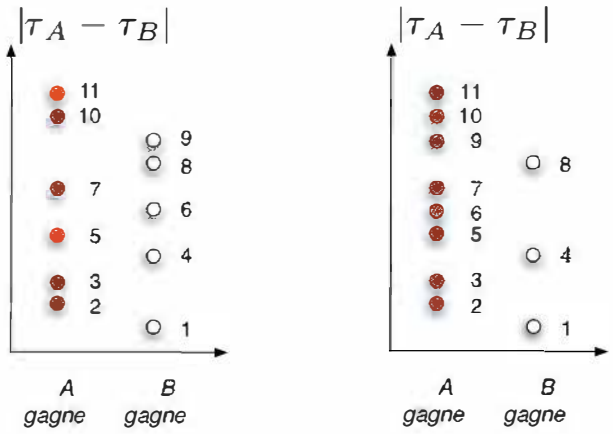


FIG. 3.24: *Pour 11 jeux de données, on note (à gauche) que l'algorithme A a gagné sur les rangs 2, 3, 5, 7, 10 et 11, et l'algorithme B sur les rangs 1, 4, 6, 8 et 9. On a donc* $G_A = 37$ *et* $G_B = 29$. *À droite, on obtient* $G_A = 53$ *et* $G_B = 13$. *La valeur critique pour 11 jeux de test est 10. Donc, on ne peut pas conclure à gauche :* $|G_A - G_B| < 10$ *alors que l'on peut décider que A est meilleur que B à droite* ($G_A - G_B = 40$).

Le problème est d'éviter de commettre au moins une erreur de type 1, consistant à rejeter l'hypothèse H_0 (e.g. l'algorithme n'est pas meilleur que les autres) alors qu'elle est vraie (de fait, il n'y a pas de supériorité significative).

Les statistiques offrent des procédures appropriées pour tester la signification de la différence entre moyennes multiples. Les deux plus connues sont le test ANOVA (*ANalysis Of VAriance*) et le test non paramétrique de Friedman.

5.3.1 L'analyse de la variance : ANOVA

L'analyse ANOVA divise la variabilité totale entre les résultats observés pour les différents algorithmes entre une variabilité entre les algorithmes, une variabilité entre les jeux de données, et une variabilité résiduelle (erreur). Si la variabilité entre algorithmes est estimée comme significativement plus grande que la variabilité résiduelle, alors l'hypothèse nulle peut être rejetée et on peut conclure qu'il y a une différence entre les algorithmes. Nous renvoyons le lecteur à tout ouvrage général sur les tests statistiques pour plus de détails.

La limite principale du test ANOVA est qu'il suppose que les échantillons de données sont issus d'une distribution normale, ce qui n'a aucune raison d'être dans le cas de mesures de taux d'erreur. C'est pourquoi on préfère le test de Friedman.

5.3.2 Le test de Friedman

Dans le test de Friedman, on classe les K algorithmes pour chaque jeu de test séparément. Le meilleur algorithme obtenant le rang 1, le deuxième le rang 2, etc. En cas d'égalité, le rang moyen est assigné aux algorithmes égaux. Soit r_i^j le rang du j-ème algorithme sur le jeu de données j parmi N. Le test de Friedman compare les rangs moyens $R_j = \frac{1}{N} \sum_i r_i^j$ des algorithmes.

Sous l'hypothèse nulle selon laquelle tous les algorithmes sont équivalents, les rangs de chaque algorithme devraient obéir à la statistique de Friedman :

Copyright © 2010 Eyrolles.

$$\chi_F^2 = \frac{12N}{K(K+1)}\left[\sum_j R_j^2 - \frac{K(K+1)^2}{4}\right]$$

avec $K-1$ degrés de liberté, quand N et K sont suffisamment grands ($N > 10$ et $K > 5$).

Iman et Davenport, [ID80], ont montré que cette statistique était trop prudente et ont calculé une statistique plus précise :

$$F_F = \frac{(N_1)\chi_F^2}{N(K-1) - \chi_F^2}$$

distribuée selon la F-distribution avec $K-1$ et $(K-1)(N-1)$ degrés de liberté.

5.4 Discussion

Les procédures de comparaison discutées ne prennent en compte que les taux d'erreurs. Il est clair que si deux classifieurs ont la même performance, mais que l'un est plus complexe que l'autre, il vaut mieux retenir le plus simple des deux.

Par ailleurs, peu d'études prennent en compte des coûts de mauvaise classification dissymétriques. Il y a là un champ de recherche qui reste à explorer.

6. Autres problèmes pratiques

6.1 Classes très déséquilibrées

Il existe de nombreuses applications pour lesquelles les classes d'objets sont déséquilibrées (*imbalanced data sets*, en anglais). Par exemple, dans le cas de l'identification de pollutions maritimes à partir d'images satellites, la fraction d'images montrant un bateau en train de dégazer en mer est heureusement très faible par rapport à l'ensemble des images récoltées, de l'ordre de moins de une pour dix mille [KHM98]. De même, dans des applications de détection de bioactivité, au sein de toutes les molécules documentées dans les chimiothèques, très peu de molécules sont bioactives par rapport à une cible pharmacologique. En conséquence, si l'on entraînait sans précaution un système sur les données brutes, le système obtiendrait un score excellent en se contentant de prédire indistinctement la classe majoritaire.

Cette prédiction serait correcte, et le déséquilibre des classes sans importance, si le système était ensuite appelé à travailler dans les mêmes conditions, par exemple si les images satellites sont parcourues aléatoirement pour une surveillance en continu. Mais supposons qu'une information de pollution parvienne au centre de surveillance : les images satellites examinées seront celles des bateaux dans la zone concernée. Ce n'est plus un bateau sur dix mille qu'il faut identifier, mais un sur dix. L'apprentissage doit donc être fait à partir d'un ensemble d'images où il y a dix bateaux innocents pour un seul coupable.

Prenons un autre exemple : la lecture automatique des numéros des plaques minéralogiques à partir des photos flash des bornes radars. Pour simplifier, intéressons-nous seulement à l'identification du dernier chiffre, en supposant le véhicule français (5 pour un véhicule du Doubs, 3 pour un de la Gironde, etc.). Admettons que, pour chaque chiffre, il y a autant de véhicules qui circulent en France (ce n'est sans doute pas vrai, mais si on avait les chiffres précis, on pourrait en tenir compte). On a donc un problème de classification : à partir de l'image du dernier chiffre, lui attribuer une classe entre 0 et 9. L'ensemble d'apprentissage peut se constituer avec

Copyright © 2010 Eyrolles.

un nombre identique d'images supervisées par la même classe, puisque ce que nous appellerons au chapitre 15 la *probabilité a priori* des classes vaut un dixième pour chaque classe. Les classes sont donc parfaitement équilibrées en phase d'apprentissage et de test.

Mais quand le radar est installé à la sortie de Besançon, il prendra une proportion plus importante de photos de plaques se terminant par 5, mais il n'en tiendra pas compte (à tort) dans sa décision et il aura tendance à verbaliser à tort trop de plaques ne se terminant pas par 5. C'est un problème du même type que le précédent : la décision est biaisée par rapport à l'apprentissage, car les probabilités a *priori* des classes ont changé entre les deux phases.

Si on connaît les probabilités a *priori* des classes en exploitation, et qu'elles sont différentes de celles de l'ensemble d'apprentissage, il est donc nécessaire de rééquilibrer cet ensemble d'apprentissage. Pour ce faire, on dispose des approches suivantes (on suppose pour simplifier qu'il n'y a que deux classes) :

- **Ré-équilibrage des classes**. Cela peut être effectué soit en retirant des exemples de la classe majoritaire (par exemple par un tirage aléatoire), soit en augmentant artificiellement la classe minoritaire. La première solution a l'inconvénient d'éliminer des données qui peuvent être porteuses d'une information utile, spécialement lorsque la taille de l'échantillon d'apprentissage est petite par rapport à la dimension de l'espace des entrées. C'est pourquoi il est intéressant de considérer la seconde solution. Celle-ci se décline en plusieurs options :
  - *Création d'exemples virtuels.* Une technique intéressante est de créer des exemples virtuels proches des exemples par un procédé de bruitage, par exemple par une génération de distribution gaussienne autour des exemples. En contrôlant la variance de cette distribution, on provoque de fait une sorte de lissage ou de régularisation des hypothèses produites (qui ont ainsi tendance à moins épouser les exemples des classes minoritaires).
  - *Augmentation du poids des exemples.* Lorsque l'algorithme d'apprentissage prend en compte le poids des exemples (dans certains cas de réseaux connexionnistes ou dans les approches bayésiennes par exemple), il est envisageable d'augmenter le poids des exemples des classes minoritaires de manière à rééquilibrer le poids global des classes.

- **Modification de la fonction de performance**. Sans modifier l'échantillon d'apprentissage, il est possible de modifier le critère de performance que le système cherche à optimiser. Deux approches en particulier sont utilisées :
  - *Modification de la matrice de coûts.* Une possibilité est de pénaliser beaucoup la mauvaise classification des exemples des classes minoritaires par rapport à la mauvaise classification des exemples des classes majoritaires (voir la section 6.2).
  - *Modification du critère de performance.* Dans un esprit proche de la solution précédente, il est possible de modifier la formule du taux d'erreur pour tenir compte des fréquences respectives des différentes classes.
  - Utilisation d'une technique de « *déplacement co-varié* » (*covariate-shift*) qui modifie le critère empirique pour biaiser l'apprentissage vers la région de $\mathcal{X}$ dans laquelle devraient se trouver la majorité des exemples à venir. (voir |QCSSL09|).

Il faut remarquer que cette modification de la fonction de performance produit un résultat qui n'est pas facile à relier avec exactitude au changement de la valeur des probabilités *a priori* des classes entre l'apprentissage et la décision. Aucune de ces techniques n'est de plus uniformément supérieure aux autres. Il faut donc en examiner les avantages et inconvénients pour chaque application, en testant l'hypothèse apprise en situation de fonctionnement, si possible. La littérature sur ce problème est abondante (voir par exemple |Dom99, DH00, Elk01, JS02, KHM98, SLD02|).

Copyright © 2010 Eyrolles.

6.2 Matrice de coûts non symétrique

Un autre cas de dissymétrie entre les classes se produit quand une mauvaise classification n'a pas les mêmes effets ou le même coût. Un exemple simple est le dépistage primaire d'une maladie : il vaut mieux du point de vue de la santé publique réaliser un test « pessimiste », qui envoie vers des examens complémentaires une proportion importante de patients sains que de ne pas faire d'examen complémentaire sur des patients vraiment malades. En conséquence, le coût de mauvaise décision n'est pas le même pour les deux classes. Ce cas peut se traiter en se déplaçant sur la courbe ROC, comme expliqué au paragraphe 4.5.1. La théorie bayésienne de la décision (chapitre 15) peut intégrer explicitement des coûts de décision non symétriques. Il faut noter que dans la plupart des méthodes d'apprentissage de concept, le choix de coûts d'erreur égaux est implicitement fait.

6.3 Données bruitées

Les données peuvent être bruitées de plusieurs manières. Le *bruit de mesure* modifie la valeur réelle des attributs sans changer la classe. Il est souvent dû à des erreurs de saisie ou des imprécisions pendant la capture des données. Il n'y a pas grand-chose à faire pour le réduire. Le *bruit de classe* est une erreur de l'expert, qui se trompe (soit par mégarde, soit par incompétence) dans l'attribution d'une étiquette.

Il est parfois possible de détecter et d'éliminer les erreurs dues au bruit de classe quand une donnée de la classe ω_1 est loin (au sens d'une certaine distance) des autres données de la même classe et entourée de données de la classe ω_2. Les techniques d'édition que l'on verra avec les méthodes des plus proches voisins, au chapitre 15 sont conçues pour détecter ces données mal étiquetées et pour enlever ces points isolés (ou *outliers*) que l'on peut supposer victimes d'un bruit de classe. Mais le problème est très délicat à régler dans le cas général, puisque rien n'assure que deux classes doivent être bien séparées. Le véritable problème se situe dans les zones de confusion entre classes, où une méthode d'édition donne un résultat aléatoire.

Dans certains cas, on peut émettre des hypothèses sur la probabilité P_{ij} qu'une donnée de vraie classe ω_i soit étiquetée par la classe ω_j et en tirer profit dans l'exploitation des données. Ces travaux restent pour le moment du domaine de la recherche (voir par exemple [Mag07]).

6.4 Espace d'entrée de très grande dimension

Lorsque les données sont décrites à l'aide de très nombreux descripteurs, comme c'est par exemple le cas en analyse d'images, en génomique ou en recherche d'information sur le Web, plusieurs problèmes potentiels se posent :

Le problème le plus important est que les algorithmes de recherche de régularités dans des espaces de grande dimension ne peuvent plus distinguer facilement les « vraies » régularités des coïncidences fortuites. Par exemple, dans le cas de la classification à l'aide de données issues de puces à ADN, il existe statistiquement presque toujours au moins un gène (parmi des milliers) dont la mesure de l'activité permet de classer parfaitement les quelques dizaines d'exemples d'apprentissage, sans que ce gène soit véritablement lié au phénomène biologique étudié. Cela est dû au fait que la *capacité* d'un espace d'hypothèses prenant en compte de très nombreux attributs est très grande et qu'il est alors nécessaire de disposer soit d'un échantillon d'apprentissage de taille suffisante (c'est-à-dire énorme), soit de connaissances *a priori* permettant de limiter effectivement cette capacité. Une approche souvent utilisée dans ce cas est de réduire la dimension de l'espace d'entrée en recourant à des méthodes telles que celles décrites dans la section 1.3.

Copyright © 2010 Eyrolles.

6.5 Très gros volumes de données en grande dimension

Alors qu'un nombre limité de données décrites dans un espace de grande dimension pose le problème de la découverte fiable de structures dans ces données, disposer d'un nombre important de données en grande dimension pose de surcroît des problèmes calculatoires.

Prenons le cas par exemple d'un site de e-commerce disposant d'un catalogue de $d = 10^5$ articles, que nous nommerons *items*, et devant satisfaire une moyenne de $N = 10^6$ transactions par semaine. L'analyse des données correspondantes pose des défis à la fois calculatoires et statistiques. Par exemple, le simple calcul d'une matrice de corrélation par paire requière $\mathcal{O}(Nd^2)$ opérations et une taille mémoire de l'ordre de $\mathcal{O}(d^2)$, soit 10^{16} et 10^{10} respectivement. Des ordres de grandeur clairement trop importants pour être considérés à la légère.

Heureusement, ce genre de données est fréquemment du type *clairsemé* (*sparse*), c'est-à-dire que chaque transaction, par exemple, n'implique typiquement qu'une dizaine d'items. Cela signifie que seulement 10/50 000 ou encore 0,02 % des éléments de la matrice de transactions $N \times d$ sont non nuls.

Il est donc intéressant d'essayer de tirer parti de cette propriété. C'est ce qui est fait couramment en *fouille de données* (*data mining*) avec la recherche des sous-ensembles I d'items (appelés *itemsets* en anglais, mais aussi, fréquemment, en français) présents dans la matrice de transactions. Les itemsets fréquents peuvent aussi être considérés comme des contraintes sur l'ensemble de tous les modèles probabilistes d'un certain ordre pour les données.

Il existe plusieurs variantes d'algorithmes efficaces pour chercher les *itemsets fréquents*, c'est-à-dire dont la fréquence f_I est supérieure à un certain seuil, dans des matrices clairsemées. Nous renvoyons au chapitre 18.

Remarque

Le traitement des gigantesques bases de données ou quantités de données (au-dessus du tera-octets ou tera-bytes (tera pour 10^{12})) reste encore essentiellement du ressort de la recherche. Outre des progrès impératifs sur l'efficacité des algorithmes[21] et sur la répartition de leur exécution sur des grilles de calcul, la solution de ce problème passe par le développement de techniques d'échantillonnage sans biais, de calculs d'estimations statistiques à la volée, et d'algorithmes d'apprentissage incrémental.

6.6 Exemples non indépendants (non i.i.d.)

Parce que l'analyse théorique de l'apprentissage repose essentiellement sur des méthodes statistiques et que le lampadaire de la statistique éclaire essentiellement le terrain dans lequel les événements ou données sont supposés indépendants et identiquement distribués, on a longtemps eu recours et abusé de l'hypothèse i.i.d.

Cependant, il est courant que les données ne vérifient pas cette propriété. Les patients qui viennent voir un médecin ne sont pas nécessairement indépendants les uns des autres, ce qui revient à dire que la connaissance de patients déjà vus peut fournir des informations sur le prochain patient. De même, supposons qu'une caméra soit disposée dans une rue de Paris pour faire un apprentissage sur le type de véhicules présents. Si les prises de vue sont effectuées avec un écart suffisant, alors il est probable qu'elles seront i.i.d., en revanche, avec des prises de vues rapprochées, le système pourra être victime d'effets de séquences, par exemple lorsqu'un un cortège officiel passe.

Les données non indépendantes posent des problèmes intéressants. D'une part, elles apportent moins d'information sur les régularités sous-jacentes $\mathbf{p}_{\mathcal{X}\times\mathcal{Y}}$ puisqu'elles sont en partie redon-

[21] Il est de plus en plus souvent dit qu'un « bon » algorithme d'apprentissage sur m données doit être en complexité au plus $\mathcal{O}(m)$.

Copyright © 2010 Eyrolles.

dantes. D'autre part, les dépendances entre exemples peuvent indiquer des tendances temporelles intéressantes à détecter et à identifier. On peut également les exploiter comme dans le *tracking* qui consiste à apprendre une dépendance locale $\mathbf{p}_{\mathcal{X}_t \times \mathcal{Y}}$ sur un sous-espace $\mathcal{X}_t$ à chaque instant et à prédire à peu près $\mathcal{X}_{t+1}$. Ainsi, il est possible d'apprendre des modèles plus simples, car locaux, et d'obtenir une performance générale aussi bonne, voire meilleure qu'un apprentissage sur l'ensemble des données, avec un apprentissage en ligne beaucoup moins coûteux (voir [SKS07]).

Il existe plusieurs analyses théoriques pour rendre compte des performances à attendre dans ce type de situation. Pour le moment, ces analyses reposent sur l'idée de détecter une granularité suffisante dans les données pour que les « grumeaux » correspondants puissent être considérés indépendants. Les bornes sur l'erreur en généralisation deviennent alors nettement moins bonnes que sous l'hypothèse d'indépendance. Ces études peuvent certainement être améliorées.

6.7 Apprentissage multi-objectif

L'apprentissage est fondamentalement une activité multi-objectifs. Par exemple, on veut à la fois identifier une bonne règle de décision, qui ne soit pas trop coûteuse à évaluer, qui soit éventuellement facile à interpréter, et qui s'articule aisément avec les connaissances antérieures. Pourtant, la plupart du temps, soit un seul objectif est explicitement retenu, comme la performance en généralisation, soit les différents objectifs sont agrégés dans un seul nombre. Cela est essentiellement dû au fait qu'une grande partie des algorithmes classiques ne peuvent optimiser qu'un critère de performance scalaire.

Des expériences ont cependant été décrites dans lesquelles le fait de poursuivre simultanément plusieurs buts pouvait améliorer la performance sur certains des sous-buts, avec les mêmes données d'apprentissage. Par exemple, un réseau de neurones, auquel était demandé de prédire la classe de formes en entrée, voyait ses performances augmentées pour cette tâche si on lui demandait simultanément de prédire aussi la couleur des formes (c'est-à-dire que la couleur était demandée en sortie au lieu d'être fournie en entrée pour le réseau à un seul objectif. De ce fait, le réseau ne bénéficiait pas de davantage d'information en cours d'apprentissage, mais on l'obligeait à établir des relations entre classe et couleur, ce qui, dans ce cas, avantageait l'apprentissage). Ces phénomènes observés de manière empiriques restent à analyser en profondeur.

Des progrès ont été réalisés récemment sur l'optimisation multi-critères grâce au concept de *front de Pareto*. Ces progrès sont liés en partie à l'utilisation de techniques évolutionnaires (voir chapitre 8) et de méthodes d'optimisation stochastiques (voir par exemple [YS08]).

6.8 Apprentissage multi-étiquette

On parle d'apprentissage multi-étiquette quand la tâche d'apprentissage est la prédiction d'une ou de plusieurs étiquette(s) pour chaque entrée, et où les données d'apprentissage sont souvent étiquetées avec plusieurs valeurs. L'ensemble d'apprentissage prend alors la forme : $\mathcal{S} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_m, Y_m)\}$ où les Y_i sont des ensembles de valeurs prises dans l'espace de sortie : $Y_i \in \mathcal{Y}$.

Par exemple, dans le contexte du tri, on peut vouloir pour une entrée donnée que le système renvoie une liste d'entrées à classer avant et une liste d'entrée à classer après. Il peut aussi s'agir d'associer des étiquettes ou *tag* à des documents sur Internet, ou de catégoriser des phrases, des courriels ou des molécules dans des classes non mutuellement exclusives.

Ce genre de tâche, qui n'est étudiée que depuis peu de temps, requiert des métriques d'évaluation particulières, ainsi que des algorithmes dédiés (voir par exemple le tutoriel donné à ECML-PKDD-09, disponible sur :
http://www.ecmlpkdd2009.net/wp-content/uploads/2009/08/learning-from-multi-label-data.pdf).

Copyright © 2010 Eyrolles.

Résumé

Ce chapitre a abordé la représentation des connaissances pour les objets et les concepts, les méthodes de base pour la recherche d'une hypothèse d'apprentissage et les techniques de la mesure de qualité d'une hypothèse. Les points suivants ont été traités :

- Les objets d'apprentissage sont décrits par des attributs qui peuvent être numériques, symboliques, séquentielles, mixtes.
- Avant toute phase d'apprentissage, on peut chercher à nettoyer les données, en particulier en éliminant les attributs inutiles.
- Les hypothèses d'apprentissage doivent aussi être représentées. Un grand nombre de possibilités existent : classes, valeurs numériques, arbres, modèles bayésiens et markoviens, grammaires, formules logiques. Toutes ces représentations ne sont pas compatibles avec tous les types de données.
- Une fois l'espace des hypothèses défini, il faut pouvoir l'explorer. Les méthodes de gradient, discret et numérique sont des techniques de base utilisées dans beaucoup d'algorithmes d'apprentissage pour la recherche d'une bonne solution.
- Il faut aussi savoir mesurer la qualité d'une hypothèse produite par un algorithme : les techniques par ensemble de test et ensemble de validation ainsi que les méthodes par validation croisée sont à la base de cette mesure.
- Il est aussi essentiel de savoir comparer deux algorithmes d'apprentissage sur des jeux de données identiques ou non, pour savoir lequel est le plus efficace dans cette situation particulière.
- L'utilisateur doit savoir définir le protocole d'apprentissage approprié à son problème.

Copyright © 2010 Eyrolles.

Deuxième partie

Apprentissage par exploration

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 4

Induction et relation d'ordre : l'espace des versions

La première partie de ce livre a été dévolue aux concepts fondamentaux de l'apprentissage artificiel. Ce chapitre est le premier de ceux destinés à présenter des techniques d'apprentissage. Il présente une méthode générale d'induction supervisée de concepts à partir d'exemples et de contre-exemples.

L'apprentissage de concept a longtemps été envisagé uniquement comme une recherche empirique d'une hypothèse cohérente avec l'échantillon de données. La notion d'espace des versions a profondément renouvelé cette manière de voir. De fait, l'espace des versions n'est rien d'autre que l'ensemble de toutes les hypothèses, prises dans un ensemble $\mathcal{H}$ donné a priori, *qui sont cohérentes avec les données d'apprentissage. Cet ensemble est en général trop grand pour être manipulé tel quel, mais il se trouve que si une relation de généralité entre hypothèses peut être définie sur $\mathcal{H}$, alors il devient possible de définir deux ensembles finis S et G qui ont la propriété de définir implicitement tous les concepts cohérents, c'est-à-dire l'espace des versions. L'apprentissage revient alors à construire et à adapter ces deux ensembles frontière : c'est ce que fait l'algorithme d'élimination des candidats.*

Ce chapitre constitue un pivot entre les chapitres conceptuels et les chapitres plus appliqués. Il nous permettra d'examiner un exemple clair d'articulation entre la notion d'espace des hypothèses cohérentes discutées dans le chapitre 2 et les principes des méthodes d'exploration de l'espace des hypothèses exposés dans le chapitre 3.

Copyright © 2010 Eyrolles.

Sommaire

1 Les concepts de base 140
1.1 La description des attributs, la description des concepts 140
1.2 Les sélecteurs 141
1.3 La relation de généralité entre les hypothèses 142
1.4 La relation entre un objet et un concept 143
2 La structuration de l'espace des hypothèses 144
2.1 Préliminaires 144
2.2 Un ordre partiel sur l'espace des hypothèses 146
2.3 Quelques opérateurs de spécialisation et de généralisation 147
2.4 Quelques propriétés utiles d'un espace structuré par une relation d'ordre partiel 149
3 La construction de l'espace des versions 152
3.1 L'algorithme d'élimination des candidats 152
3.2 Un exemple d'application : le système LEX 155
4 La représentation des connaissances par un treillis de Galois 156
4.1 La construction de la structure 156
4.2 L'utilisation pour l'apprentissage 158

VOICI UN PETIT problème d'apprentissage. J'ai devant moi quatre oiseaux, deux canards et deux manchots. Les attributs suivants sont suffisants pour les décrire : la *Forme* de leur bec, leur *Taille*, leur *Envergure* et la *Couleur* de leur cou. Le premier nous indique si le bec est *Aplati* ou non, les deux suivants se mesurent en centimètres et le dernier peut prendre les valeurs *Roux*, *Orange*, *Gris* ou *Noir*. Ces oiseaux sont étiquetés soit + (les canards), soit − (les manchots) et je veux apprendre un concept cohérent avec les exemples, une formule qui explique tous les canards et rejette tous les manchots.

Je me donne un langage de représentation pour les concepts : je vais les écrire comme une conjonction de certaines propriétés sur les attributs. Par exemple, la formule logique suivante :

$$[Aplati = VRAI] \wedge Taille \in [30, 50] \wedge Envergure \in]-\infty, +\infty[\wedge [Couleur = CouleurChaude]$$

est un concept qui représente l'ensemble des oiseaux dont le bec est de la forme *Aplati*, dont la taille est comprise entre 30 cm et 50 cm, dont l'envergure est indifférente et dont la couleur du cou est soit *Roux*, soit *Orange*. Ce qui peut s'écrire, dans une syntaxe plus légère, comme :

$$(VRAI, [30, 50], ?, CouleurChaude)$$

Je me trouve devant ces quatre oiseaux que je vais examiner les uns après les autres. Ils sont représentés par le tableau suivant :

	Aplati	*Taille*	*Envergure*	*Couleur*	*Classe*
$e_1 =$	*VRAI*	30	49	Roux	+ (canard)
$e_2 =$	*FAUX*	70	32	Gris	− (manchot)
$e_3 =$	*VRAI*	40	46	Orange	+ (canard)
$e_2 =$	*FAUX*	60	33	Orange	− (manchot)

Copyright © 2010 Eyrolles.

Je commence par le premier exemple, un canard. Comment écrire, dans le langage des concepts, une hypothèse de généralisation compatible pour le moment avec lui seul? Pourquoi pas justement :

$$(VRAI, [30, 50], ?, CouleurChaude)$$

Mais si je veux être prudent, je peux aussi produire le concept qui se contente de mémoriser ce premier exemple :

$$(VRAI, [30, 30], [49, 49], Roux)$$

Je dispose également d'une solution radicalement inverse, induire le concept universel :

$$(VRAI \vee FAUX) \wedge]-\infty, +\infty[\wedge]-\infty, +\infty[\wedge Couleur = (?, ?, ?, ?)$$

qui signifie que j'accepte à partir de cet exemple unique de considérer comme cohérents tous les oiseaux que peut décrire mon langage des hypothèses.

Si je considère maintenant le second exemple, il est clair que cette dernière possibilité doit être éliminée puisqu'elle couvre désormais un contre-exemple. Je peux la « couper au ras » de ce contre-exemple[1] selon l'un des attributs. Il y a six solutions, parmi lesquelles :

$$v_1 = (?,]-\infty, 49], ?, ?)$$

$$v'_1 = (?, ?, ?, CouleurChaude)$$

Je peux aussi conserver ma politique prudente et produire :

$$v_2 = (VRAI, [30,30], [49,49], Roux)$$

qui est toujours cohérent vis-à-vis de mes deux exemples. J'aurais aussi pu générer, entre autres :

$$v_3 = (?,]-\infty, 31], ?, ?)$$
$$v_4 = (VRAI, [0,35], [46, +\infty[, Roux)$$

Il est intuitif de vérifier que le concept v_1 est plus général que v_4 et v_2, et que de même v_4 est plus général que v_2. En revanche, bien que v_3 soit apparemment un concept très vaste, on ne peut pas dire qu'il soit plus général que v_4 : il ne contient pas l'objet $[VRAI, 33, 50, Roux]$ que v_4 contient.

Cet exercice sera terminé plus loin ; la notion intuitive de concept « plus général » qu'un autre sera formalisée. Pour le moment, retenons quelques principes qui gouvernent cette façon de procéder :

- Une formule du langage des concepts décrit un ensemble d'objets (ici, d'oiseaux).
- Les concepts sont reliés entre eux par une relation de généralité, qui reflète l'inclusion des ensembles d'objets qu'ils représentent.
- Le nombre de concepts cohérents avec les exemples est très grand (ou infini, si la taille des objets n'est pas mesurée en nombres entiers).
- Les exemples sont introduits les uns après les autres et à chaque fois l'ensemble courant d'hypothèses se modifie pour qu'elles restent cohérentes avec les exemples déjà examinés.

[1] C'est-à-dire exclure ce contre-exemple et le minimum de ce que le langage des hypothèses impose d'exclure avec lui.

Copyright © 2010 Eyrolles.

Ce chapitre a pour objet de formaliser ces notions, en particulier grâce à l'introduction d'une relation d'ordre dans l'espace des concepts. Il va aussi montrer qu'il est possible de calculer, en traitant séquentiellement les exemples, deux ensembles finis S et G de concepts à partir desquels on peut déduire tous ceux qui acceptent les exemples et refusent les contre-exemples.

Notations utiles pour le chapitre

$\mathcal{H}$	L'ensemble de toutes les hypothèses, choisi par l'apprenant
$\mathcal{L}_{\mathcal{X}}$	Le langage de description des exemples
$\mathcal{L}_{\mathcal{H}}$	Le langage de description des hypothèses
$couverture(h)$	L'ensemble des exemples couverts par l'hypothèse h
$\preceq$	...est plus spécifique que ...
$\succeq$	...est plus général que ...
$\top$	L'élément maximal de $\mathcal{H}$
$\bot$	L'élément minimal de $\mathcal{H}$
hps	L'hypothèse la plus spécifique cohérente avec un ensemble d'exemples
hpg	L'hypothèse la plus générale cohérente avec un ensemble d'exemples
$\vee$	La disjonction logique
$\wedge$	La conjonction logique
$\overset{gen}{\Rightarrow}$	Un opérateur de généralisation
$\overset{spe}{\Rightarrow}$	Un opérateur de spécialisation

1. Les concepts de base

1.1 La description des attributs, la description des concepts

Nous disposons d'un ensemble d'apprentissage de m exemples $\mathcal{S} = \langle (\mathbf{x}_i, u_i), i = 1, m \rangle$, avec u_i valant soit $VRAI$ ou $+$ (exemple du concept à apprendre), soit $FAUX$ ou $-$ (contre-exemple). L'ensemble des exemples positifs est noté $\mathcal{S}_+$, celui des exemples négatifs $\mathcal{S}_-$. Les valeurs $\mathbf{x}$ sont prises dans un espace de représentation[2] $\mathcal{X}$, qui est une combinaison d'attributs de nature diverse. Nous considérons ici que les attributs peuvent être :

- *Binaires* : le bec d'un oiseau est *Aplati* ou non.
- *Numériques* comme l'*Envergure*.
- *Nominaux* comme la *Couleur*. En général, un attribut nominal est simplement une énumération de valeurs possibles comme $Couleur = \{Rouge, Orange, Gris, Noir\}$. Cet ensemble de valeurs possibles s'appelle un domaine. Mais une hiérarchie peut exister sur ces valeurs, comme dans notre exemple d'introduction :
  $CouleurChaude = \{Rouge, Orange\}$,
  $CouleurFroide = \{Gris, Noir\}$ et
  $Couleur = \{CouleurChaude, CouleurFroide\}$.
  On sait dans ce cas que l'attribut est appelé *arborescent* (voir le chapitre 3 et la figure 4.1).

[2] L'espace de représentation est noté LI (pour *Language of Instances*) dans la terminologie originale de l'espace des versions.

Copyright © 2010 Eyrolles.

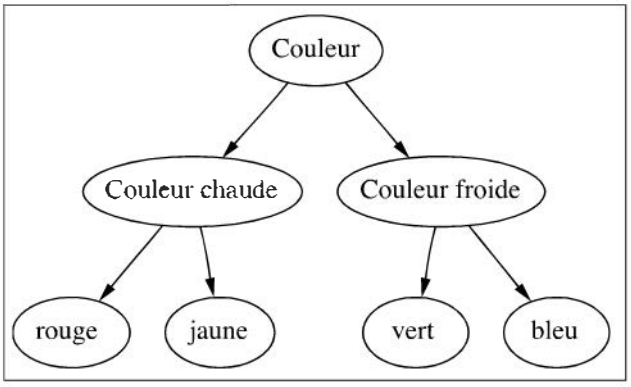


FIG. 4.1: *L'attribut Couleur est arborescent.*

On cherche à apprendre des hypothèses, ou concepts, qui sont des éléments écrits dans $\mathcal{L}_{\mathcal{H}}$, le langage des hypothèses[3] (voir le chapitre 1). Le langage $\mathcal{L}_{\mathcal{H}}$ peut être défini de façons variées, mais nous pouvons déjà donner un exemple : le langage de la logique des propositions sur des sélecteurs, qui est apparu informellement dans l'exemple d'introduction.

1.2 Les sélecteurs

Il est fréquent dans l'apprentissage par l'espace de versions que le langage $\mathcal{L}_{\mathcal{H}}$ choisi pour représenter les concepts soit, comme dans l'exemple liminaire, une partie de la logique des propositions. Plus précisément, il s'agit souvent d'une conjonction de propriétés binaires sur les attributs des exemples. On appelle *sélecteurs* ces propriétés, que l'on peut définir ainsi :

Définition 4.1

Un sélecteur est une application agissant sur un seul attribut de l'espace de représentation des données, à valeurs dans $\{VRAI, FAUX\}$.

Selon la nature des attributs, un sélecteur prend des formes différentes :

- Si l'attribut est *binaire*, le sélecteur s'écrit $(attribut = VRAI)$ ou $(attribut = FAUX)$; sa valeur $VRAI$ ou $FAUX$ se déduit directement de celle de l'attribut.
- Si l'attribut est *nominal* de domaine D, le sélecteur s'écrit $(attribut \in D')$, avec $D' \subset D$. Il est $VRAI$ si l'attribut prend une valeur de D', $FAUX$ sinon.
- Si l'attribut est *arborescent*, le sélecteur s'écrit $(attribut = V)$, où V est une valeur attachée à un nœud de l'arbre. Il est $VRAI$ si la valeur de l'attribut est un nœud compris au sens large entre V et une feuille de l'arbre.
- Si l'attribut est *numérique*, le sélecteur est défini par un intervalle de $\mathbb{R}$. Il est $VRAI$ si la valeur est incluse dans cet intervalle, bornes comprises.

— EXEMPLE —

Pour revenir à notre exemple de départ, le concept :

$$[Aplati = VRAI] \wedge Hauteur \in [30;50] \wedge Largeur \in]-\infty;+\infty[\wedge [Couleur = CouleurChaude]$$

qui s'écrit de manière simplifiée :

$$(VRAI, [30;50], ?, CouleurChaude)$$

[3] Dans le vocabulaire classique de l'espace des versions, le langage des hypothèses est souvent noté LC (pour *Language of Generalizations*).

Copyright © 2010 Eyrolles.

est composé d'une conjonction de quatre sélecteurs, un pour chaque attribut. L'attribut *Aplati* est binaire, *Taille* et *Envergure* sont numériques et *Couleur* est arborescent, comme le montre la figure 4.1. Sur l'objet :

Bec Aplati	*Taille*	*Envergure*	*Couleur du cou*
VRAI	60	46	*noir*

Le premier sélecteur est $VRAI$, le second est $FAUX$, le troisième est $VRAI$ et le quatrième est $FAUX$.

1.3 La relation de généralité entre les hypothèses

Nous avons vu dans la section 5, que l'induction supervisée pouvait être considérée comme un jeu entre l'espace des exemples et l'espace des hypothèses. Le processus d'apprentissage teste les hypothèses candidates de $\mathcal{H}$ sur les exemples d'apprentissage dans $\mathcal{X}$. Les informations ainsi glanées servent à déterminer d'autres hypothèses candidates, et ainsi de suite jusqu'au critère d'arrêt.

Nous supposons ici que les exemples d'apprentissage sont considérés séquentiellement, donc qu'à l'étape t l'apprenant a fabriqué une hypothèse candidate h_t cohérente avec l'échantillon d'apprentissage partiel $\mathcal{S}_t$. Un nouvel exemple d'apprentissage $\mathbf{z}_{t+1} = (\mathbf{x}_{t+1}, u_{t+1})$, avec $u_{t+1} \in \{VRAI, FAUX\}$ devient alors disponible. Il y a deux possibilités : soit il est correctement classé par l'hypothèse courante h_t, donc $h_t(\mathbf{x}_{t+1}) = u_{t+1}$, auquel cas il n'y a pas de raison de modifier h_t et l'on a simplement $h_{t+1} = h_t$, soit $\mathbf{x}_{t+1}$ n'est pas correctement classé par h_t. Deux cas sont alors possibles.

- L'*exemple est de classe négative*, c'est un contre-exemple du concept cible, et il est incorrectement classé comme positif par h_{t+1}. Cela signifie que la partie de $\mathcal{X}$ « couverte » par h_{t+1} est trop grande, au moins en ce qui concerne le point $\mathbf{x}_{t+1}$, (donc que le concept courant n'est plus correct). Il faut donc la réduire, c'est-à-dire chercher une sous-partie excluant $\mathbf{x}_{t+1}$ mais couvrant encore tous les exemples positifs de $\mathcal{S}_t$.
- Au contraire, $\mathbf{x}_{t+1}$ est de *classe positive*, et il est incorrectement classé comme négatif par h_{t+1}. Dans ce cas, cela signifie que la partie de $\mathcal{X}$ « couverte » par h_{t+1} est trop petite, au moins en ce qui concerne le point $\mathbf{x}_{t+1}$ (donc que le concept courant n'est plus complet). Il faut donc l'augmenter, c'est-à-dire chercher une sur-partie incluant $\mathbf{x}_{t+1}$ mais ne couvrant aucun des exemples négatifs de $\mathcal{S}_t$ (voir la figure 4.2).

Dans les deux cas, il est patent que l'hypothèse courante doit être modifiée en fonction des relations d'inclusion dans $\mathcal{X}$. Il faut donc trouver une relation entre les hypothèses dans $\mathcal{H}$ qui respecte la relation d'inclusion dans $\mathcal{X}$. On parle de *relation de généralité* entre les hypothèses.

Plus formellement, nous dirons qu'une hypothèse h_1 est plus spécifique ou encore moins générale qu'une hypothèse h_2 si et seulement si l'ensemble des exemples couverts par h_1 est inclus dans l'ensemble des exemples couverts par h_2.

Définition 4.2 (Couverture d'une hypothèse)

La couverture d'une hypothèse $h \in \mathcal{H}$, notée couverture(h), est l'ensemble des exemples de $\mathcal{X}$ que décrit h. On dit que h couvre *les éléments de couverture(h).*

Définition 4.3 (Relation de généralité dans $\mathcal{H}$)

Une hypothèse h_1 est plus spécifique *(ou* moins générale*) qu'une hypothèse h_2, ce qui se note $h_1 \preceq h_2$, si et seulement si couverture(h_1) $\subseteq$ couverture(h_2).*

Copyright © 2010 Eyrolles.

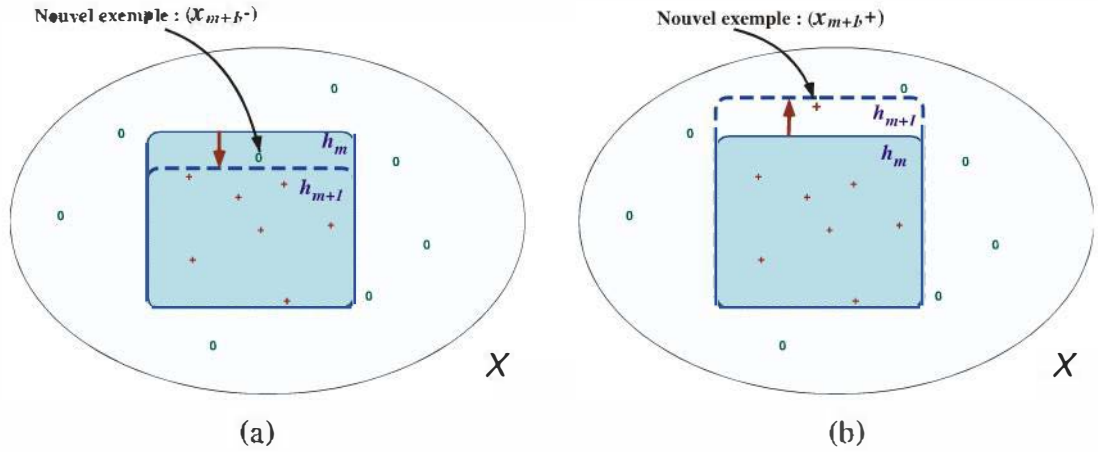


FIG. 4.2: *Lorsqu'un nouvel exemple est mal classé par l'hypothèse courante, il faut modifier celle-ci soit en la réduisant au sens de l'inclusion (a) afin d'exclure le nouvel exemple s'il est négatif, soit en l'augmentant (b) s'il est positif.*

Remarque : Cette relation est à prendre au sens large : toute hypothèse est plus spécifique qu'elle-même.

Les relations $\preceq$ et $\succeq$ sur $\mathcal{H}$ sont illustrées dans les figures 4.3 et 4.4.

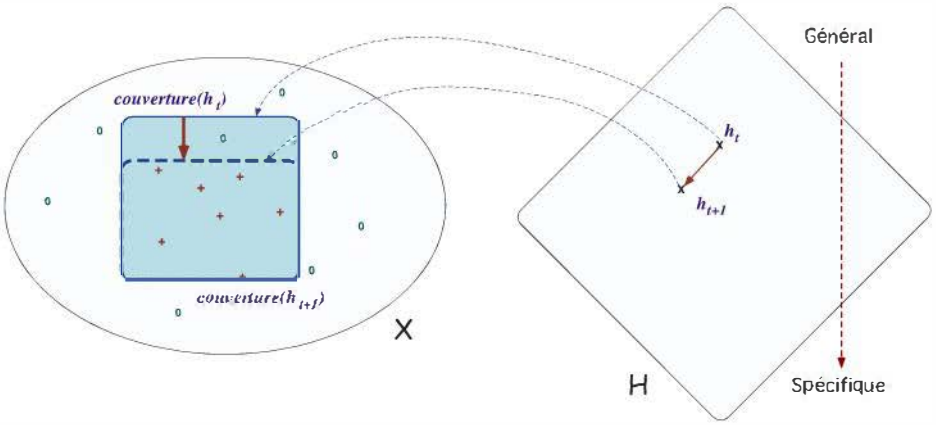


FIG. 4.3: *La relation d'inclusion dans $\mathcal{X}$ induit la relation de généralisation dans $\mathcal{H}$. Ici, $h_{t+1} \preceq h_t$.*

1.4 La relation entre un objet et un concept

Nous nous situons dans ce chapitre dans le cadre du principe inductif MRE, qui a été décrit au chapitre 2. Cela signifie que chaque hypothèse sera évaluée en fonction de sa performance mesurée sur l'échantillon d'apprentissage, c'est-à-dire de son *risque empirique*. La méthode de base de l'espace des versions traite de plus le cas du risque empirique nul, c'est-à-dire qu'elle recherche

Copyright © 2010 Eyrolles.

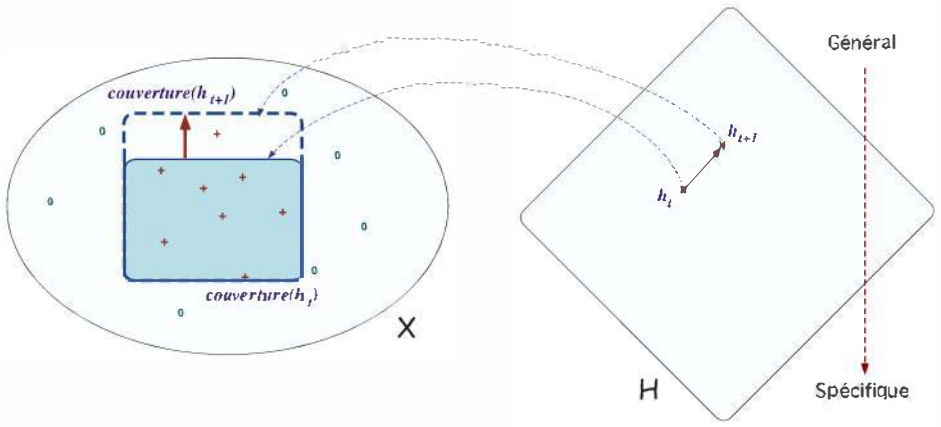


FIG. 4.4: *La relation d'inclusion dans $\mathcal{X}$ induit la relation de généralisation dans $\mathcal{H}$. Ici, $h_{t+1} \succeq h_t$.*

les hypothèses cohérentes[4], s'accordant parfaitement aux données, c'est-à-dire ne faisant aucune erreur sur les exemples d'apprentissage.

Définition 4.4 (Hypothèse cohérente, correcte, complète)

Lorsque le risque empirique associé à une hypothèse est nul, on dit que l'hypothèse est cohérente. *Cela signifie :*

1. *Que tous les exemples positifs de l'échantillon d'apprentissage sont correctement étiquetés par l'hypothèse : on dit aussi que l'hypothèse « couvre » tous les exemples positifs. L'hypothèse est alors dite* complète.
2. *Que tous les exemples négatifs sont correctement classés par l'hypothèse, c'est-à-dire rejetés comme ne faisant pas partie du concept. On dit alors que l'hypothèse ne couvre pas les exemples négatifs : elle est dite* correcte.

La figure 4.5 illustre ces notions. Si l'on suppose que les exemples étiquetés sont issus d'un concept cible, un risque empirique nul signifie que, sur l'échantillon d'apprentissage au moins, le concept cible et l'hypothèse considérée coïncident.

2. La structuration de l'espace des hypothèses

2.1 Préliminaires

Dans sa version de base, la méthode de l'espace des versions a pour but de résoudre le problème suivant :

Trouver tous les éléments de $\mathcal{H}$ cohérents avec les exemples.

[4] Nous évitons ici l'anglicisme « consistant » qui est souvent employé. Il se confond souvent avec la « consistance » de la méthode *MRE*, pour laquelle nous employons le mot « pertinence » (voir le chapitre 2).

Copyright © 2010 Eyrolles.

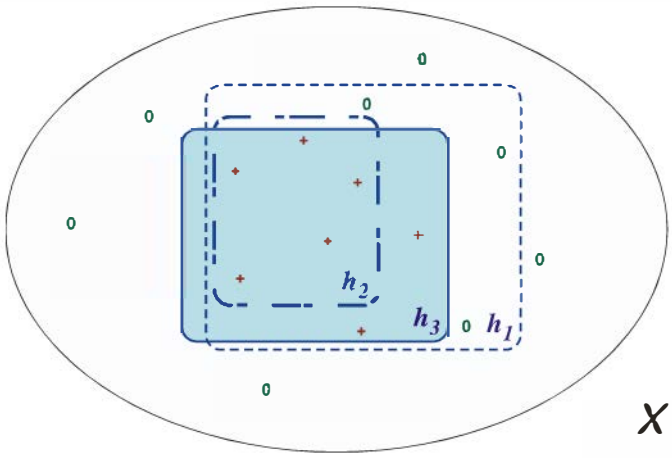


FIG. 4.5: *La notion de couverture des exemples (figurés par des +) et des contre-exemples (figurés par des 0) par les hypothèses. Les hypothèses sont ici figurées par le sous-espace des exemples qu'elles couvrent dans l'espace $\mathcal{X}$. Les hypothèses h_1, h_2 et h_3 sont respectivement* complète *mais* incorrecte, correcte *mais* incomplète, *et* complète *et* correcte, *c'est-à-dire* cohérente.

Comme on le verra, la relation de généralité sur $\mathcal{H}$ permet d'éviter une énumeration complète de ses individus, tout en gardant la possibilité de vérifier pour tout concept s'il est cohérent ou non avec les données d'apprentissage.

Il est à noter que cette méthode de base exclut le bruit dans les données d'apprentissage, c'est-à-dire qu'elle ne sait pas traiter le cas où deux objets identiques sont l'un étiqueté positif, l'autre étiqueté négatif.

Il est commode de supposer que le langage de généralisation inclut le langage de représentation des exemples : ainsi, chaque exemple peut être considéré comme une hypothèse ne couvrant que cet exemple, qui s'écrit comme l'exemple lui-même. C'est ce que l'on appelle l'*astuce de la représentation unique* (*single representation trick*). Ce n'est pas tout à fait le cas du problème d'apprentissage donné en introduction, puisque l'exemple :

	Bec Aplati	*Taille*	*Envergure*	*Couleur*	*Classe*
$e_1 =$	*VRAI*	30	49	roux	+

s'écrit dans le langage des concepts de manière un peu différente :

$$(VRAI, [30, 30], [49, 49], roux)$$

Mais la correspondance est assez directe pour que l'on puisse ignorer la fonction de transformation. Il existe des cas où cette transformation est plus complexe[5].

Pour simplifier, nous supposerons dans la suite sans réelle perte de généralité que

$$\mathcal{L}_{\mathcal{X}} \subset \mathcal{L}_{\mathcal{H}} \tag{4.1}$$

[5] C'est le cas de l'inférence de grammaires régulières que nous verrons au chapitre 7. Dans ce problème, un exemple est représenté par une séquence de lettres et son apprentissage par cœur par un automate qui ne reconnaît que cette séquence. C'est encore plus net en programmation logique inductive (chapitre 5), où la vérification qu'un concept couvre un exemple se fait par l'éxécution d'un programme Prolog, ce qui est loin d'être trivial.

Copyright © 2010 Eyrolles.

$\mathcal{L}_{\mathcal{X}}$ est donc identifié à un sous-ensemble de $\mathcal{L}_{\mathcal{H}}$, ce que l'on peut exprimer d'une autre façon : un exemple est le concept qui généralise le moins l'exemple en question.

2.2 Un ordre partiel sur l'espace des hypothèses

La relation d'inclusion définie sur $\mathcal{X}$ induit une relation de généralité sur $\mathcal{H}$ qui est une relation d'ordre partiel. La figure 4.6 illustre cette notion. Cette relation est partielle et non pas totale, ce qui signifie que deux éléments quelconques dans l'espace considéré peuvent ne pas être liés par cette relation.

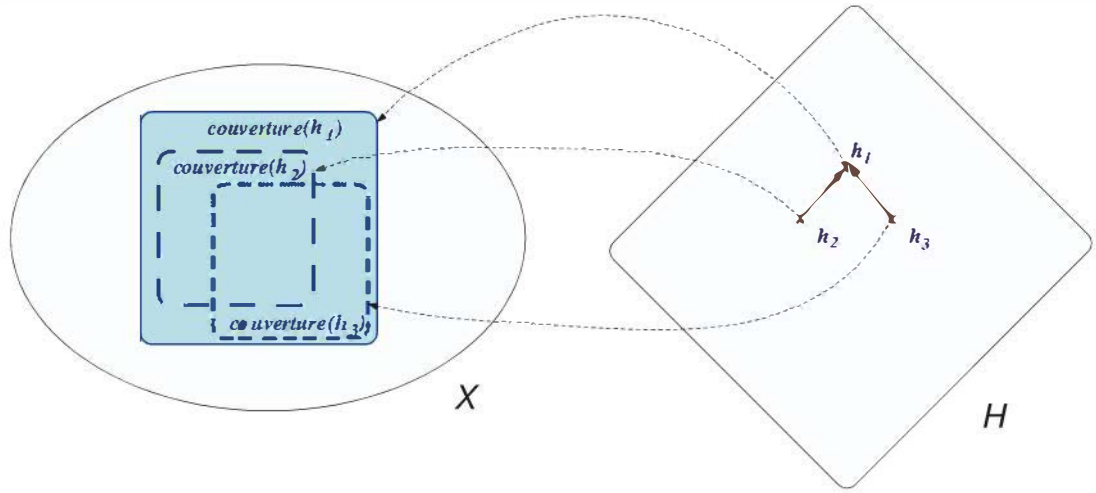


FIG. 4.6: *La relation d'inclusion dans $\mathcal{X}$ induit la relation de généralisation dans $\mathcal{H}$. Il s'agit d'une relation d'ordre partielle : ici, les hypothèses h_2 et h_3 sont incomparables entre elles, mais elles sont toutes les deux plus spécifiques que h_1.*

Une relation d'ordre partiel induit une structure de *treillis* sur $\mathcal{H}$. Cela signifie que pour tout couple d'hypothèses h_i et h_j, il existe au moins une hypothèse qui soit plus générale que chacune d'entre elles et qu'il n'est pas possible de la spécifier sans perdre cette propriété. L'ensemble de ces hypothèses est appelé le *généralisé maximalement spécifique* de h_i et h_j et noté $gms(h_i, h_j)$. De même, il existe un ensemble d'hypothèses plus spécifiques que h_i et h_j qu'il n'est pas possible de généraliser sans perdre cette propriété. On appelle cet ensemble le *spécialisé maximalement général* et on le note $smg(h_i, h_j)$.

Par une extension facile au cas de plus de deux hypothèses, on peut définir de même un ensemble $gms(h_i, h_j, h_k, \ldots)$ et un ensemble $smg(h_i, h_j, h_k, \ldots)$.

Finalement, nous supposons[6] qu'il existe dans $\mathcal{H}$ une hypothèse plus générale que toutes les autres (ou élément maximal) notée $\top$ et une hypothèse plus spécifique que toutes les autres (ou élément minimal) notée $\bot$ (voir la figure 4.7).

Les exemples ainsi que plusieurs figures du chapitre 1 et de ce chapitre montrent clairement que la relation d'inclusion est fondamentale pour le problème de l'induction. En effet, une hypothèse incorrecte (donc couvrant indûment des exemples négatifs) devra être spécialisée pour que sa couverture exclue ces exemples, alors qu'une hypothèse incomplète (ne couvrant pas tous les exemples positifs connus) devra être généralisée pour que ces exemples deviennent éléments de sa couverture. Il est donc naturel que le processus d'induction soit guidé par ces relations d'inclusion.

[6] C'est en général le cas en pratique.

Copyright © 2010 Eyrolles.

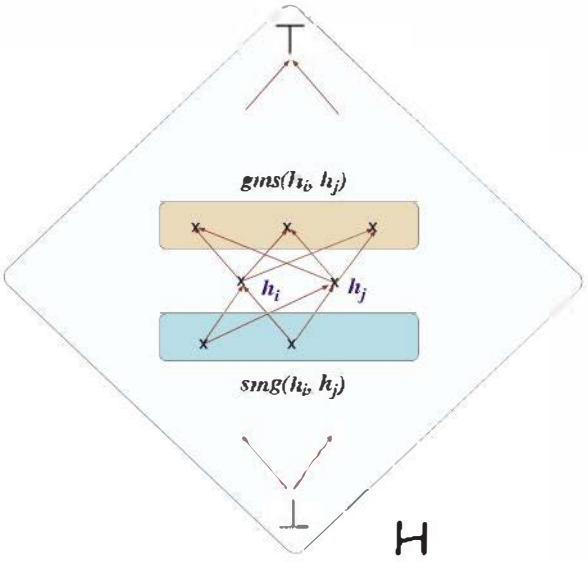


FIG. 4.7: *Une vision schématique et partielle du treillis de généralisation sur $\mathcal{H}$ induit par la relation d'inclusion dans $\mathcal{X}$. Chaque flèche indique la relation de généralité (notée $\preceq$ dans le texte). L'élément le plus spécifique du treillis est $\perp$ et le plus général est $\top$.*

Nous avons également déjà souligné dans le chapitre 1 que l'induction nécessite une mise à jour des hypothèses directement dans $\mathcal{H}$. Comme l'espace $\mathcal{H}$ des hypothèses est défini par son langage de description $\mathcal{L}_{\mathcal{H}}$, cela signifie qu'il faut trouver comment associer à la relation d'inclusion dans $\mathcal{X}$ des opérations syntaxiques sur $\mathcal{L}_{\mathcal{H}}$ correspondant à la relation de généralité. Trouver des équivalences aux relations d'inclusion dans $\mathcal{X}$ revient donc à trouver des opérateurs dans le langage $\mathcal{L}_{\mathcal{H}}$ qui permettent de modifier une hypothèse h_m en une nouvelle hypothèse h_{m+1} inclue dans la première ou l'incluant[7], c'est-à-dire plus spécifique ou plus générale.

2.3 Quelques opérateurs de spécialisation et de généralisation

Le problème de la recherche d'opérateurs syntaxiques de généralisation ou de spécialisation sera également débattu dans le chapitre 5 portant sur la programmation logique inductive, en particulier parce que la solution n'est pas évidente lorsque l'on utilise des représentations en logique des prédicats dite aussi logique d'ordre 1. Il est en revanche facile d'exhiber des exemples d'opérateurs satisfaisants dans le cas de représentations fondées sur la logique des propositions et la représentation attribut-valeur (voir le chapitre 13). À titre d'illustration, nous présentons quelques-uns de ces opérateurs de généralisation.

L'opération de généralisation est notée $\overset{gen}{\Rightarrow}$. Une formule du type $A \wedge (B = v_1) \rightarrow \mathcal{C}$ signifie qu'un objet couvert par le concept $\mathcal{C}$ est décrit par la conjonction d'un sélecteur sur les attributs A (ici avec A binaire) et B (ici avec B nominal ordonné), ce dernier valant v_1 pour B. On constatera que les généralisations proposées ne sont évidemment pas des opérations logiquement valides.

[7] On tire profit ici de la confusion assumée entre la notion de concept et celle de partie de $\mathcal{X}$ (voir l'équation 4.1) pour parler de concept inclus dans un autre, alors que la relation d'inclusion n'a, à proprement parler, de sens que pour les catégories qui sont définies sur $\mathcal{X}$.

Copyright © 2010 Eyrolles.

Opérateur de clôture d'intervalle

$$\left.\begin{array}{l} A \wedge (B = v_1) \rightarrow \mathcal{C} \\ A \wedge (B = v_2) \rightarrow \mathcal{C} \end{array}\right\} \overset{gen}{\Rightarrow} A \wedge (B \in [v_1, v_2]) \rightarrow \mathcal{C}$$

Par exemple :

$$\left.\begin{array}{l} Bec\ Aplati \wedge (Envergure = 50) \rightarrow \text{canard} \\ Bec\ Aplati \wedge (Envergure = 55) \rightarrow \text{canard} \end{array}\right\}$$
$$\overset{gen}{\Rightarrow} Bec\ Aplati \wedge (Envergure \in [50, 55]) \rightarrow \text{canard}$$

Opérateur de l'ascension dans l'arbre de hiérarchie

Pour généraliser une description incluant un attribut arborescent, il suffit de le remplacer par l'un de ses ascendants dans l'arbre :

$$\left.\begin{array}{l} A \wedge (B = n_1) \rightarrow \mathcal{C} \\ A \wedge (B = n_2) \rightarrow \mathcal{C} \end{array}\right\} \overset{gen}{\Rightarrow} A \wedge (B = n) \rightarrow \mathcal{C}$$

où n est le plus petit nœud ascendant commun aux nœuds n_1 et n_2. Par exemple :

$$\left.\begin{array}{l} Bec\ Aplati \wedge (Couleur = roux) \rightarrow \text{canard} \\ Bec\ Aplati \wedge (Couleur = orange) \rightarrow \text{canard} \end{array}\right\}$$
$$\overset{gen}{\Rightarrow} Bec\ Aplati \wedge (Couleur = Couleur\ chaude) \rightarrow \text{canard}$$

Opérateur d'abandon de conjonction

$$A \wedge B \rightarrow \mathcal{C} \overset{gen}{\Rightarrow} A \rightarrow \mathcal{C}$$

Par exemple : $Bec\ Aplati \wedge (Couleur = roux) \rightarrow \text{canard} \overset{gen}{\Rightarrow} Bec\ Aplati \rightarrow \text{canard}$

Opérateur d'ajout d'alternative

$$A \rightarrow \mathcal{C} \overset{gen}{\Rightarrow} A \vee B \rightarrow \mathcal{C}$$

Par exemple :

$$Bec\ Aplati \rightarrow \text{canard} \overset{gen}{\Rightarrow} Bec\ Aplati \vee (Couleur = orange) \rightarrow \text{canard}$$

Opérateur de changement de conjonction en disjonction

$$A \wedge B \rightarrow \mathcal{C} \overset{gen}{\Rightarrow} A \vee B \rightarrow \mathcal{C}$$

Par exemple :

$$Bec\ Aplati \wedge (Couleur = orange) \rightarrow \text{canard} \overset{gen}{\Rightarrow} Bec\ Aplati \vee (Couleur = orange) \rightarrow \text{canard}$$

Chaque opérateur de généralisation permet de transformer l'expression d'une hypothèse en l'expression d'une hypothèse plus générale, couvrant davantage d'éléments de l'espace des exemples $\mathcal{X}$. Il est possible de renverser chaque opérateur de généralisation pour obtenir des *opérateurs de spécialisation* qui transforment une hypothèse en une hypothèse moins générale, ou plus spécifique, couvrant moins d'éléments de $\mathcal{X}$.

Copyright © 2010 Eyrolles.

Ainsi l'opérateur de généralisation par montée dans la hiérarchie des descripteurs peut fournir un opérateur de spécialisation par descente dans la hiérarchie :

$$A \wedge (B = n) \rightarrow C \overset{spe}{\Rightarrow} A \wedge (B = n_1) \rightarrow C$$

où n est un nœud ascendant du descripteur n_1.

Bec Aplati $\wedge$ (*Couleur* = *Couleur chaude*) $\rightarrow$ canard

$\overset{spe}{\Rightarrow}$ *Bec Aplati* $\wedge$ (*Couleur* = *roux*) $\rightarrow$ canard

Pour résumer, si un espace d'hypothèses $\mathcal{H}$ est défini par un langage $\mathcal{L}_\mathcal{H}$ qui admet des opérateurs de spécialisation et de généralisation, alors il est muni d'une structure d'ordre partiel associée à la relation d'inclusion sur l'espace des exemples $\mathcal{X}$. De ce fait, elle est particulièrement pertinente pour la tâche d'induction. Ainsi, la recherche d'une hypothèse cohérente avec les exemples d'apprentissage peut être guidée par ces opérateurs et être par conséquent beaucoup plus efficace qu'une recherche par gradient (chapitre 3). En examinant quelques propriétés de l'espace $\mathcal{H}$ ainsi muni d'une relation d'ordre, nous allons voir que l'avantage peut être plus important encore.

2.4 Quelques propriétés utiles d'un espace structuré par une relation d'ordre partiel

Rappelons d'abord qu'il n'y a généralement pas bijection entre l'espace des hypothèses $\mathcal{H}$ et l'espace des exemples $\mathcal{X}$: c'est ce qu'implique l'existence d'un biais de représentation, c'est-à-dire les limites de l'expressivité du langage $\mathcal{L}_\mathcal{H}$.

De ce fait, il est important de s'assurer que certaines propriétés sont vérifiées. Il en est deux qui nous concernent spécialement. La première a trait à la convexité de $\mathcal{H}$: est-on certain que l'application des opérateurs de spécialisation/généralisation sur des expressions de $\mathcal{L}_\mathcal{H}$ produit toujours des expressions valides qui ont une contrepartie dans $\mathcal{X}$? En d'autres termes, ne risque-t-on pas, en jouant avec ces opérateurs, de produire des « hypothèses » qui n'auraient pas de sens? La deuxième propriété est duale de la précédente : peut-il y avoir des hypothèses dans $\mathcal{H}$ qui sont de fait plus générales ou plus spécifiques qu'une autre hypothèse de $\mathcal{H}$, mais qu'on ne puisse pas obtenir à partir de celle-ci par une séquence d'opérateurs de spécialisation/généralisation ? Si l'une ou l'autre de ces deux propriétés s'avérait non vérifiée (voir figure 4.8), alors l'exploration de $\mathcal{H}$ par l'application des opérateurs pourrait conduire à des résultats aberrants : soit des hypothèses sans signification dans $\mathcal{X}$, soit au contraire la non production d'hypothèses pertinentes de $\mathcal{H}$. Heureusement, ces deux propriétés peuvent être obtenues. Plus formellement :

Définition 4.5 (Ensemble convexe pour la généralisation)

Un ensemble E dont les éléments sont représentables dans un langage $\mathcal{L}_\mathcal{H}$ est convexe *si et seulement si :*

$$\textit{Pour tous } h_1, h_2, h_3 \textit{ tels que } h_1, h_3 \in E, \textit{ et } h_1 \preceq h_2 \preceq h_3 \textit{ alors } h_2 \in E$$

Propriété 4.1 (Théorème de convexité [Hir90])

L'ensemble des hypothèses $\mathcal{H}$ défini par un langage $\mathcal{L}_\mathcal{H}$ sur lequel sont définis des opérateurs de spécialisation/généralisation est convexe.

Copyright © 2010 Eyrolles.

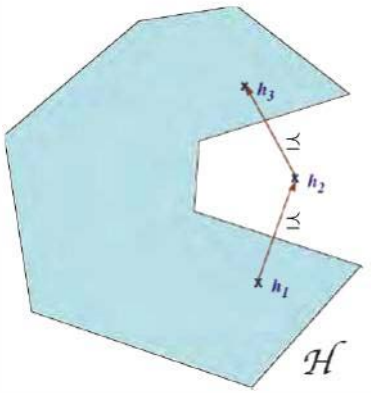


FIG. 4.8: *Un exemple d'ensemble non convexe pour la relation de généralité.*

Définition 4.6 (Ensemble borné pour la généralisation)

Un ensemble C d'hypothèses décrits par un langage $\mathcal{L}_\mathcal{H}$ est borné *si et seulement si pour tout h dans C il existe une hypothèse g maximalement générale dans C et une hypothèse s maximalement spécifique dans C telles que $s \preceq h \preceq g$.*

Il n'est pas possible de garantir la propriété d'être borné pour tout ensemble décrit sur un langage $\mathcal{L}_\mathcal{H}$ muni d'opérateurs de spécialisation/généralisation. C'est donc une contrainte qu'il faut à chaque fois vérifier ou imposer si besoin. Cette contrainte est en général vérifiée sur les langages d'ordre 0 (logique des propositions), elle demande par contre des soins particuliers en logique d'ordre 1 (logique des prédicats).

Si les deux propriétés précédentes sont vraies, alors une troisième propriété, fondamentale pour la suite, en découle.

Définition 4.7 (S : Les hypothèses cohérentes maximalement spécifiques)

L'ensemble des hypothèses de $\mathcal{H}$ couvrant les exemples positifs et excluant les exemples négatifs, et telles qu'il ne soit pas possible de les spécialiser sans perdre ces propriétés, est appelé le S-set. *Nous le noterons S dans la suite.*

Définition 4.8 (G : Les hypothèses cohérentes maximalement générales)

L'ensemble des hypothèses de $\mathcal{H}$ couvrant les exemples positifs et excluant les exemples négatifs, et telles qu'il ne soit pas possible de les généraliser sans perdre ces propriétés, est appelé le G-set. *Nous le noterons G dans la suite.*

Théorème 4.1 (Représentation de l'espace des versions par S et G [Hir90])

Si un ensemble d'hypothèses est convexe et borné, alors il peut être représenté par sa borne inférieure S et sa borne supérieure G.

Définition 4.9 (Espace des versions)

*L'ensemble de toutes les hypothèses cohérentes avec les exemples d'apprentissage est appelé l'*espace des versions.

Le théorème (4.1) prouve que l'espace des versions peut être représenté de manière économique par ses bornes S et G.

Copyright © 2010 Eyrolles.

Cette propriété a trois corollaires essentiels :

1. À tout instant, l'ensemble des hypothèses cohérentes avec un ensemble d'exemples d'apprentissage est représentable par une *borne inférieure* (un ensemble d'hypothèses S) et une *borne supérieure* (un ensemble d'hypothèses G) : toute hypothèse comprise entre un élément de S et un élément de G est cohérente.
2. Un algorithme d'*apprentissage* peut opérer *en calculant ces deux bornes*, et donc en calculant l'ensemble des hypothèses cohérentes. Il s'agit là d'une idée novatrice par rapport aux algorithmes d'apprentissage recherchant *une* hypothèse cohérente par modification incrémentale d'une hypothèse initiale.
3. Il n'existe donc pas de concept cohérent moins spécifique qu'un élément de G ou plus spécifique qu'un élément de S.

En considérant un exemple comme une hypothèse, grâce à la relation $\mathcal{L}_{\mathcal{X}} \subset \mathcal{L}_{\mathcal{H}}$ (équation 4.1), nous pouvons donc remarquer que S est un sous-ensemble du *gms* de l'ensemble des exemples positifs (on ne peut rien dire d'aussi simple sur G). Nous allons voir l'application de cette idée dans l'algorithme d'élimination des candidats proposé par T. Mitchell [Mit82].

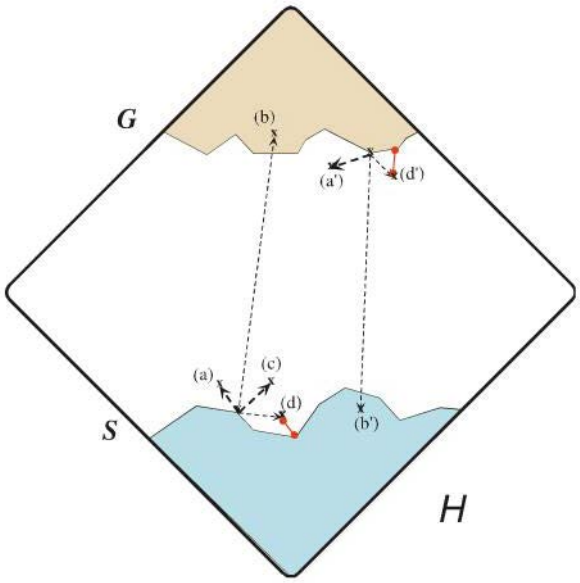


FIG. 4.9: *Cette figure schématise les différents cas possibles lors de la mise à jour des ensembles S et G par l'algorithme d'élimination des candidats. Les cas* (a), (b), (c) *et* (d) *correspondent à la mise à jour de S pour tenir compte d'un exemple positif. On suppose ici qu'un élément de S ne couvre pas ce nouvel exemple et doit être généralisé. Les flèches en pointillés illustrent le cas où quatre directions de généralisations seraient possibles. La direction* (b) *doit être éliminée car elle correspond à une surgénéralisation : l'hypothèse produite est en effet plus générale qu'une hypothèse de G et doit donc couvrir des exemples négatifs. L'hypothèse* (d) *doit également être écartée car elle est plus générale qu'une autre hypothèse de S qui est cohérente avec les exemples. Il reste donc les hypothèses* (a) *et* (c) *qui remplaceront l'ancienne hypothèse dans S. Les cas* (a'), (b') *et* (d') *illustrent des cas duaux dans le cas de la mise à jour de l'ensemble G pour tenir compte d'un nouvel exemple négatif.*

Copyright © 2010 Eyrolles.

3. La construction de l'espace des versions

Nous allons voir maintenant comment construire S et G à partir des exemples, et nous vérifierons que tout élément de l'espace des versions est plus spécifique qu'un certain élément de G et moins spécifique qu'un certain élément de S.

3.1 L'algorithme d'élimination des candidats

L'apprentissage par l'espace des versions est associé à un algorithme de construction des solutions, appelé l'*élimination des candidats* (algorithme 1).

Il procède de manière itérative, exemple par exemple, en mettant à jour S et G. Sa convergence est assurée par un théorème (non démontré ici) qui prouve qu'un seul examen de chaque exemple suffit et que l'ordre de présentation des exemples n'influe pas sur le résultat de l'algorithme.

Algorithme 1 : Algorithme d'élimination des candidats.

Résultat : Initialiser G comme l'hypothèse la plus générale de $\mathcal{H}$
Initialiser S comme l'hypothèse la moins générale de $\mathcal{H}$
pour chaque *exemple* x **faire**
  si x *est un exemple positif* **alors**
    Enlever de G toutes les hypothèses qui ne couvrent pas x
    pour chaque *hypothèse* s *de* S *qui ne couvre pas* x **faire**
      Enlever s de S
      `Généraliser(`s,x,S`)`
      c'est-à-dire : ajouter à S toutes les généralisations minimales h de s telles que :
        • h couvre x et
        • il existe dans G un élément plus général que h
      Enlever de S toute hypothèse plus générale qu'une autre hypothèse de S
    fin
  sinon
    `/* x est un exemple négatif */`
    Enlever de S toutes les hypothèses qui couvrent x
    pour chaque *hypothèse* g *de* G *qui couvre* x **faire**
      Enlever g de G
      `Spécialiser(`g,x,G`)`
      c'est-à-dire : ajouter à G toutes les spécialisations maximales h de g telles que :
        • h ne couvre pas x et
        • il existe dans S un élément plus spécifique que h
      Enlever de G toute hypothèse plus spécifique qu'une autre hypothèse de G
    fin
  fin si
fin

Cet algorithme gère deux procédures `Generaliser(s,x,G)` et `Specialiser(g,x,S)`, qui seront utilisées pour remplacer dans G (respescivement S) un concept devenant trop spécifique (respectivement trop général) par un ou plusieurs autres concepts permettant de respecter les contraintes de consistance. Ces procédures se définissent grâce aux notions de spécialisation minimale et de généralisation minimale. La figure 4.9 illustre les différents cas qui peuvent se rencontrer lors de la mise à jour des bornes S et G par l'algorithme d'élimination des candidats.

Copyright © 2010 Eyrolles.

Il est donc démontré que cet algorithme itératif remplit le but fixé : il permet de trouver S et G. À partir de là il autorise la caractérisation de tous les concepts cohérents avec les exemples.

— EXEMPLE **Les oiseaux** —

Généralisation et spécialisation minimale

Dans l'exemple des canards et des manchots, les opérations de spécialisation et de généralisation ont une définition naturelle. Soit le concept :

$$v_1 = (Bec\ Aplati = ?\ ,\ Taille \in [0, 60],\ Envergure \in [33, +\infty[,\ Couleur = Couleur\ chaude)$$

et le contre-exemple (on ne répète pas les noms des attributs) :

$$e_4 = (FAUX, 60, 38, orange)$$

Il y a quatre possibilités de spécialisation minimale de v_1 vis-à-vis de e_4 :

$$\begin{array}{r} (\ VRAI\ \ , [0, 60], [33, +\infty[, Couleur\ chaude) \\ (\ ?\ \ , [0, 59], [33, +\infty[, Couleur\ chaude) \\ (\ ?\ \ , [0, 60], [39, +\infty[, Couleur\ chaude) \\ (\ ?\ \ , [0, 60], [33, +\infty[, roux) \end{array}$$

Pour le concept :

$$v_4 = (VRAI, [0, 59], [46, +\infty[, roux)$$

et l'exemple :

$$(FAUX, 60, 47, orange)$$

on obtient la généralisation minimale :

$$(?, [0, 60], [47, +\infty[, Couleur\ chaude)$$

Déroulement de l'algorithme

Pour simplifier, nous allons éliminer le troisième attribut, l'*Envergure*. Les données d'apprentissage sont donc les suivantes :

	Bec Aplati	*Taille*	*Couleur*	*Classe*
$e_1 =$	$VRAI$	30	roux	+
$e_2 =$	$FAUX$	70	gris	−
$e_3 =$	$VRAI$	40	orange	+
$e_4 =$	$FAUX$	60	orange	−

DEBUT

Initialisation

$G = (?, ?, ?)$

$S = \emptyset$

Lecture de $e_1 = ((VRAI, 30, roux), +)$

On généralise minimalement S pour couvrir e_1 :

$S = \{(VRAI, [30.30], roux)\}$

G est inchangé : $G = (?, ?, ?)$

Lecture de $e_2 = ((FAUX, 70, gris), -)$

S est inchangé : $S = \{(VRAI, [30, 30], roux)\}$

Copyright © 2010 Eyrolles.

Il y a quatre spécialisations minimales de G pour rejeter e_2 :

$$(VRAI, ?, ?)$$
$$(?, [0, 69], ?)$$
$$(?, [71, +\infty[, ?)$$
$$(?, ?, Couleur\ chaude)$$

On ajoute à G les trois hypothèses pour lesquelles il existe dans S un élément plus spécifique.

$$G = \{ (VRAI, ?, ?), (?, [0, 69], ?) (?, ?, Couleur\ chaude) \}$$

Lecture de $e_3 = ((VRAI, 40, orange), +)$

On généralise minimalement S pour couvrir e_3

$S = \{(VRAI, [30,40], Couleur\ chaude)\}$

G est inchangé.

Lecture de $e_4 = ((FAUX, 60, orange), -)$

On essaie de spécialiser chaque élément de G par rapport à e_4.

Le premier élément ne couvre pas e_4.

Les deux suivants donnent chacun 4 concepts possibles :

Spécialisations minimales de $(?, [0,69], ?)$ pour rejeter $(FAUX, 60, orange)$:

$$(VRAI, [0, 69], ?)$$
$$(?, [0, 59], ?)$$
$$(?, [61, 69], ?)$$
$$(?, ?, Couleur\ froide)$$

Spécialisations minimales de $(?, ?, Couleur\ chaude)$ pour rejeter $(FAUX, 60, orange)$:

$$(VRAI, ?, Couleur\ chaude)$$
$$(?, [0, 59], Couleur\ chaude)$$
$$(?, [61, +\infty[, Couleur\ chaude)$$
$$(?, ?, roux)$$

On ajoute à G les spécialisations pour lesquelles il existe dans S un élément plus spécifique. Finalement, en ne conservant que le concept le plus général quand il en existe deux en relation de généralité, G comporte les deux éléments suivants :

$$G = \{ (VRAI, ?, ?), (?, [0, 59], ?) \}$$

S est inchangé : $S = \{(VRAI, [30,40], Couleur\ chaude)\}$

FIN

Tout concept plus général que le seul élément de S et plus spécifique qu'un des deux éléments de G est solution. Voici trois concepts valides :

$$(VRAI, [30, 55], ?)$$
$$(VRAI, [0, 59], Couleur\ chaude)$$
$$(?, [0, 42], ?)$$

Copyright © 2010 Eyrolles.

3.2 Un exemple d'application : le système LEX

Le système LEX a été développé par Tom Mitchell en 1983 pour apprendre automatiquement les conditions d'utilisation des opérateurs mathématiques permettant de calculer la primitive d'expressions mathématiques. C'est exactement ce à quoi les élèves des classes préparatoires aux grandes écoles passent beaucoup de temps. Au début, on leur donne un ensemble de « recettes » que l'on appelle ici des opérateurs OP. Par exemple (r et c sont ici de constantes) :

Sortie des constantes de multiplication	OP_1 : $\int r\, f(x)\, dx = r \int f(x)\, dx$
Règle d'intégration par parties	OP_2 : $\int u\, dv = uv - \int v\, du$
Règle de l'identité	OP_3 : $1 \times f(x) = f(x)$
L'intégrale d'une somme est la somme des intégrales	OP_4 : $\int [f_1(x) + f_2(x)]\, dx = \int f_1(x)\, dx + \int f_2(x)\, dx$
Primitive de la fonction $sin(x)$	OP_5 : $\int sin(x)\, dx = -cos(x) + c$
Primitive de la fonction $cos(x)$	OP_6 : $\int cos(x)\, dx = sin(x) + c$
Règle d'intégration des fonctions puissance	OP_7 : $\int x^n\, dx = \frac{x^{n+1}}{n+1} + c$

Au début de leur apprentissage, les élèves connaissent ces règles mais ne savent pas exactement dans quel contexte il est judicieux d'appliquer chacune d'entre elles. De ce fait, ils mettent beaucoup de temps à résoudre les problèmes car ils se perdent dans de nombreuses impasses. Au fur et à mesure de leur entraînement, ils apprennent à appliquer ces règles juste au moment où elles font progresser vers la solution. C'est ce type d'apprentissage qu'essaie de simuler le système LEX.

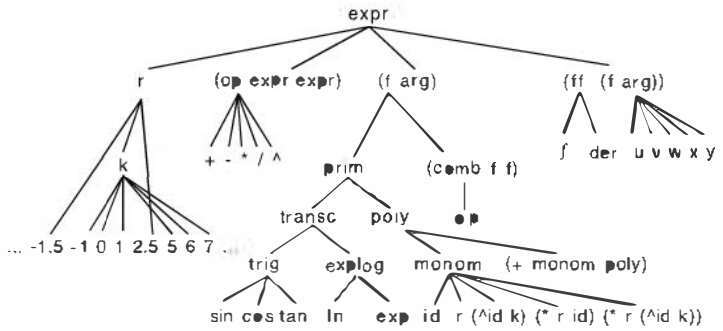


FIG. 4.10: *Une taxonomie des fonctions de base telle que celle employée par le système* LEX.

Au début de son apprentissage, le système connaît un ensemble d'opérateurs d'intégration symboliques tels que ceux donnés plus haut. Il est également doté d'une taxonomie sur les concepts de fonctions mathématiques (voir figure 4.10). Le système suit alors un cycle qui est schématisé sur la figure 4.11. À chaque cycle, un module fournit un exercice à résoudre. Un système de résolution de problème tente alors de trouver une solution en enchaînant des opérateurs d'intégration. Cela fournit un arbre de résolution qui peut aboutir à une solution ou à un échec (par exemple si le système ne trouve pas de solution avec les ressources calcul allouées). Chaque utilisation des opérateurs selon une branche ayant mené à un succès fournit un exemple positif de l'emploi de cet opérateur (en fait, il faut aussi examiner si la solution est optimale). Inversement, chaque utilisation d'un opérateur le long d'une branche ayant mené à un échec correspond à une utilisation erronée de cet opérateur et fournit donc un exemple négatif de contexte d'utilisation. Ces exemples et contre-exemples sont alors fournis à un système d'apprentissage utilisant l'algo-

Copyright © 2010 Eyrolles.

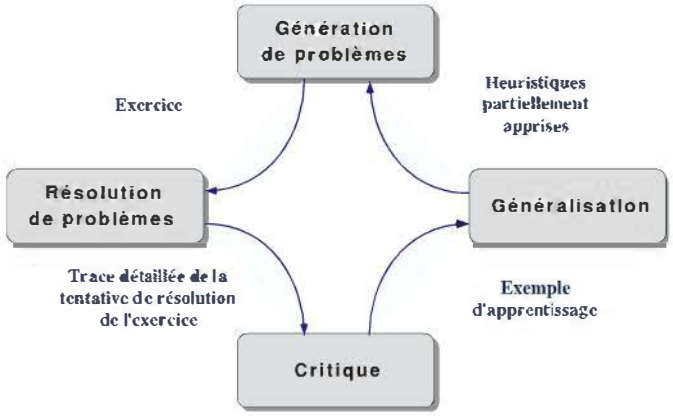


FIG. 4.11: *L'architexture générale du système* LEX *avec le cycle d'utilisation.*

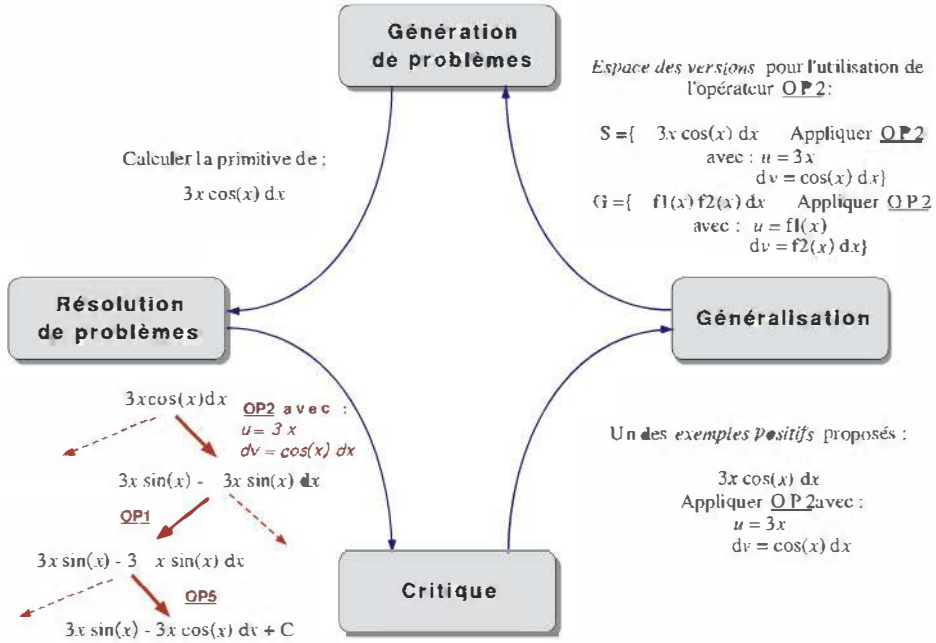


FIG. 4.12: *Un exemple de cycle d'apprentissage dans le système* LEX.

rithme d'élimination des candidats qui calcule ainsi les bornes S et G définissant les contextes dans lesquels il est approprié d'utiliser l'opérateur considéré (voir la figure 4.12 pour un exemple de cycle).

4. La représentation des connaissances par un treillis de Galois

4.1 La construction de la structure

Nous nous plaçons maintenant dans une situation techniquement un peu différente de celle traitée par l'espace des versions, mais proche dans son esprit. Nous supposons que le langage de

Copyright © 2010 Eyrolles.

représentation est purement binaire, c'est-à-dire que chaque exemple ou contre-exemple se décrit par un ensemble de réponses $VRAI$ ou $FAUX$ à des tests qu'on lui pose. Par exemple :

x_1 *vole*
x_2 *a des plumes*
x_3 *pond des œufs*
x_4 *mammifère*
x_5 *nage sous l'eau*

Sur cette représentation, un ensemble d'objets $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$ peut être décrit par le tableau suivant :

	x_1	x_2	x_3	x_4	x_5	commentaire
s_1	1	1	1	0	0	oie
s_2	0	0	1	1	1	ornithorynque
s_3	1	0	0	1	0	rhinolophe
s_4	1	1	1	0	0	cygne

L'idée de la représentation par treillis de Galois est de ne pas garder les exemples sous forme de matrice de $VRAI$ et $FAUX$, ou ici de 0 et de 1, mais de les transformer en une représentation ordonnée, comme sur la figure 4.13.

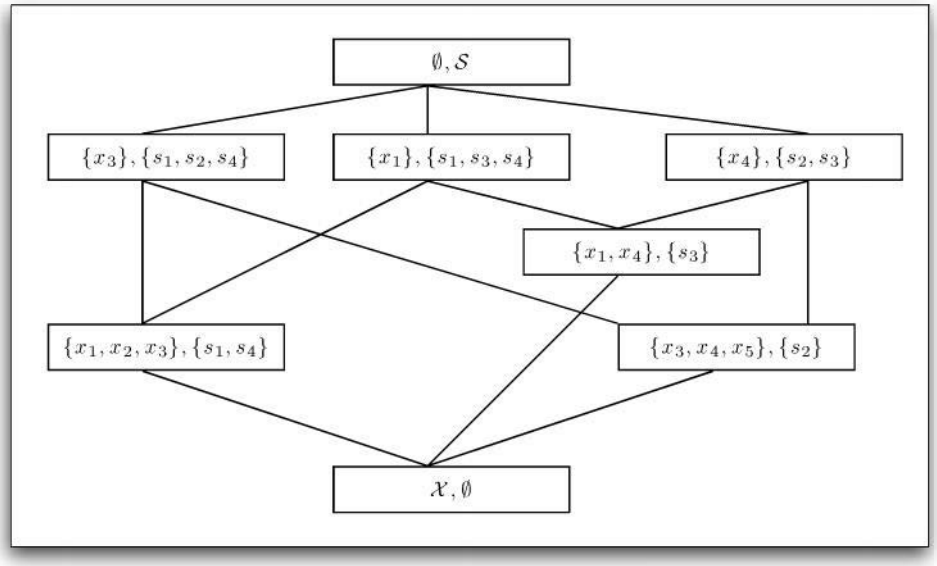


FIG. 4.13: *Un treillis de Galois.*

Que signifie un tel diagramme? Le niveau supérieur de la figure, composée d'une seule case, correspond à l'absence d'attribut ; le niveau en dessous, composé de trois cases, exprime la relation des exemples avec un attribut parmi les quatre ; le niveau suivant exprime la relation des exemples avec deux attributs, etc. On devrait selon ce principe s'attendre à trouver dans le second niveau les cinq cases de la figure 4.14.

Mais on ne représente que celles qui sont indispensables : comme l'ensemble $\{s_1, s_4\}$ est strictement inclus dans l'ensemble $\{s_1, s_3, s_4\}$, la variable x_1 rend l'écriture de x_2 inutile à ce niveau. De même, x_5 est rendue inutile par x_4 ou par x_3.

Copyright © 2010 Eyrolles.

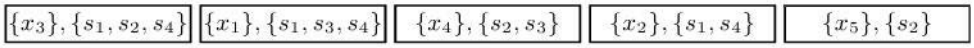


FIG. 4.14: *La première ligne complète.*

Pour tracer un trait représentant la relation d'ordre entre les éléments de niveaux différents dans le treillis de Galois, il faut que les deux composantes représentant les attributs et les exemples soient en double relation d'inclusion stricte, mais en sens inverse. Par exemple, la case $\boxed{\{x_1\},\{s_1,s_3,s_4\}}$ est en relation avec la case $\boxed{\{x_1,x_4\},\{s_3\}}$ puisque $\{x_1\} \subset \{x_1,x_4\}$ et $\{s_1,s_3,s_4\} \supset \{s_3\}$.

Cette structure résume parfaitement les relations de généralité des attributs vis-à-vis des objets et, symétriquement, celles des objets vis-à-vis des attributs. Il est évidemment possible de reconstituer le tableau des données d'apprentissage à partir de celle-ci. On peut aussi démontrer que le treillis de Galois construit sur une matrice binaire est unique. Divers algorithmes, soit prenant en compte tous les exemples à la fois, soit incrémentaux, ont été proposés pour réaliser cette construction. Il est important de remarquer que la taille du treillis peut être exponentielle en fonction du nombre d'attributs.

4.2 L'utilisation pour l'apprentissage

En quoi cette relation d'ordre particulière dans le langage des exemples peut-elle être utile à l'apprentissage ? Par la structuration qu'elle dégage des données, qui permet de les explorer de manière organisée. Si les données sont partagées en exemples positifs et négatifs et que l'on cherche à élaborer un concept correct le plus général possible, une technique est de travailler de manière ascendante dans la structure du treillis.

Sur notre exemple, supposons que les objets s_2 et s_3 soient les exemples et s_1 et s_4 les contre-exemples. La remontée dans le treillis se fait en suivant la relation d'ordre, successivement par les cases $\boxed{\mathcal{X},\emptyset}$ $\boxed{\{x_3,x_4,x_5\},\{s_2\}}$ $\boxed{\{x_4\},\{s_2,s_3\}}$. Il n'est pas possible d'aller plus loin car les objets s_1 et s_4 seraient couverts. Dans ce cas précis, le concept est compatible avec les données et il est facile de vérifier que l'attribut x_4 *mammifère* forme à lui tout seul ce concept.

Dans des cas plus complexes, le test d'un concept s'effectue de la même façon, en contrôlant sa cohérence au fur et à mesure de la progression dans le treillis. Il est possible que celle-ci soit mise en échec si les exemples sont bruités, par exemple si l'un est à la fois négatif et positif. Dans ce cas, la structure permet de trouver le concept correct le plus général, le concept complet le moins général, ou un compromis entre les deux[8].

La plupart du temps, le treillis n'est pas construit avant l'apprentissage, mais en parallèle avec la création du concept. On ne développe que la partie nécessaire au fur et à mesure. Cette technique permet d'éviter une explosion combinatoire, mais oblige à opérer des choix sur lesquels on ne pourra pas toujours revenir.

Nous n'avons pas précisé quel espace d'hypothèses ni quelle méthode d'apprentissage utiliser, car tous les deux sont à la disposition de l'utilisateur : la structure en treillis de Galois induit une technique cohérente d'exploration des données quel que soit le type du concept cherché[9] et quel que soit l'algorithme d'apprentissage proprement dit. Par exemple, les méthodes d'apprentissage par plus proches voisins (chapitre 15) sont utilisées dans ce cadre [NN97].

[8] Rappelons qu'un concept est *correct* s'il ne couvre aucun exemple négatif, *complet* s'il couvre tous les exemples positifs, *compatible ou cohérent* s'il est correct et complet.

[9] Souvent écrit sous la forme d'une formule en logique des propositions.

Copyright © 2010 Eyrolles.

Notes historiques et sources bibliographiques

La thèse de Tom Mitchell en 1978, *Version spaces : an approach to concept learning*, a marqué un tournant fondamental dans l'approche de l'apprentissage artificiel. Jusque-là en effet, mis à part les travaux portant sur la reconnaissance des formes, l'intelligence artificielle était essentiellement guidée par l'étude de la cognition naturelle, humaine en particulier. Dans cette optique, les systèmes d'apprentissage développés représentaient des tentatives de simuler la cognition humaine sur des tâches particulières. Ainsi, le système ARCH de Patrick Winston (1970) [Win70], simulait l'apprentissage d'un concept (celui d'arche) à partir d'exemples positifs et négatifs d'arches. Le système AM de Doug Lenat [Len78] simulait le raisonnement d'un mathématicien en train d'aborder la théorie des nombres et de produire des conjectures dans ce domaine. Dans tous les cas, il était supposé que le concept cible (ou la connaissance cible) était connaissable par un agent humain, et donc par le système apprenant censé le simuler. D'une part, cela allait de pair avec des approches théoriques de l'apprentissage portant sur l'*identification* exacte du concept cible et non sur une approximation. D'autre part, cela conduisait à considérer des algorithmes explorant l'espace des hypothèses possibles en adaptant et en modifiant progressivement une hypothèse unique, de même qu'apparemment un agent humain raisonne sur la base de la meilleure hypothèse courante et l'adapte si nécessaire. Les idées contenues dans la thèse de Tom Mitchell ont profondément bouleversé ce point de vue.

D'abord, l'idée d'espace des versions, l'ensemble de toutes les hypothèses cohérentes avec les données d'apprentissage, met soudain à distance les systèmes artificiels et leurs contreparties naturelles. Cela autorise à étudier des algorithmes d'apprentissage nouveaux et sans nécessairement de plausibilité psychologique. L'algorithme d'élimination des candidats en est un exemple. Ensuite, il devient naturel de s'interroger sur l'espace des hypothèses même et sur sa capacité à contenir le concept cible. Cela a conduit Mitchell à souligner l'inévitabilité d'un biais pour apprendre. Comme nous l'avons déjà amplement discuté dans les chapitres 1 et 2, la possibilité de l'induction est complètement dépendante de la richesse de l'espace des hypothèses. Avant la thèse de Mitchell, les chercheurs en intelligence artificielle examinaient en quoi la représentation des connaissances choisie était ou non favorable à des raisonnements pertinents pour le domaine considéré (c'est en particulier toute l'essence des recherches de Lenat). En revanche il n'était pas question de s'interroger sur la possibilité, encore moins la nécessité, d'avoir un espace d'hypothèses limité. La réalisation progressive de ce dernier point s'accompagne de l'essor des travaux portant sur des théories de l'apprentissage comme techniques d'approximation et non plus comme identification d'un concept cible. Le développement concomittant du connexionnisme dans les années quatre-vingts joue alors un rôle de catalyseur en permettant l'intrusion des mathématiques du continu, et donc des outils de l'analyse mathématique, comme l'optimisation et la convergence, dans l'étude de l'apprentissage.

Pour terminer par une note philosophique, il est remarquable que la vision de l'apprentissage comme sélection de bonnes hypothèses au sein d'un ensemble d'hypothèses possibles donné *a priori* s'accorde à la vision actuelle de la biologie. La théorie de l'évolution de Charles Darwin, celle de Pierre Changeux et de ses collègues (qui voient l'apprentissage comme élimination de connexions dans le cerveau), et la théorie de Noam Chomsky sur l'apprentissage de la langue naturelle comme spécialisation d'une grammaire universelle définissant l'enveloppe des langues possibles, toutes ces démarches théoriques vont à l'unisson. L'approche actuelle de l'apprentissage artificiel, considérant l'apprentissage comme la sélection des hypothèses les plus performantes par rapport aux observations, s'est finalement jointe à ce mouvement. L'avenir nous dira la destinée de cet étonnant exemple multidisciplinaire de pensée unique.

Copyright © 2010 Eyrolles.

Nous donnons dans ce qui suit quelques indications bibliographiques aux lecteurs intéressés par les recherches et les développements portant sur les espaces des versions.

Les origines de l'apprentissage par généralisation et spécialisation remontent au moins à Winston [Win75] ; la formalisation et la résolution du problème par l'espace des versions sont une des étapes majeures de la naissance de la discipline. L'appellation « espace des versions » provient du fait que Tom Mitchell étudiait toutes les versions des classifieurs cohérents avec les données, ces classificurs étant, dans son cas, pris dans un espace d'expressions logiques. L'essentiel de la théorie et de l'algorithmique a été produit par Mitchell [Mit82]. La présentation qui en est donnée ici est en partie reprise du texte de Jacques Nicolas dans [Nic93]. Les concepts introduits et l'algorithme d'élimination des candidats peuvent être trouvés dans presque tous les livres d'intelligence artificielle. Nous recommandons particulièrement celui de Mitchell [Mit97].

L'algorithme d'élimination des candidats a fait l'objet de l'examen critique de David Haussler [Hau88] qui a souligné en particulier que la taille de la borne G de l'espace des versions pouvait croître exponentiellement avec le nombre d'exemples négatifs. Cette observation relativise évidemment l'avantage apporté par la considération de ces bornes. Cependant, l'examen de la preuve de Haussler laisse supposer que le phénomène de croissance exponentielle ne peut se produire que pour des échantillons de données très particuliers, et présentés dans un ordre très défavorable. Des chercheurs comme Haym Hirsh [Hir90] ou Benjamin Smith et Paul Rosenbloom [SR90] ont proposé des heuristiques pour améliorer l'ordre de présentation des exemples. Une limite plus sérieuse de l'approche de Mitchell concerne les données bruitées, c'est-à-dire mal décrites ou mal classées. L'insistance sur la stricte cohérence des hypothèses de l'espace des versions avec les exemples condamne généralement l'algorithme original à ne pas pouvoir trouver d'hypothèse s'accordant aux données. Des propositions ont donc été formulées visant à relâcher l'exigence de stricte cohérence. Hirsh [Hir90, Hir92] a ainsi présenté un algorithme d'apprentissage dans lequel on forme un espace des versions pour chaque exemple positif, puis on en établit l'intersection. Il montre que cette idée permet de traiter des données bruitées. Michèle Sebag [Seb94a, Seb94b] a poussé cette idée plus loin avec l'approche de *disjunctive version spaces* qui marie une approche de généralisation en représentation attribut-valeur avec la technique de l'espace des versions.

L'apprentissage grâce à la structure en treillis de Galois a été en particulier étudié par [Gan93, Wil92a], [LS98]. La théorie de ces espaces est développée dans [Bir67]. L'apprentissage par plus proche voisins en liaison avec la construction de cette structure est étudiée dans [NN97].

Résumé

- La méthode de l'espace des versions vise à définir tous les concepts cohérents avec un ensemble d'exemples.
- Comme le nombre de ces concepts peut être infini, on s'intéresse à une définition de leur ensemble en intension.
- Celle-ci est définie par deux ensembles finis S et G et une relation d'ordre sur les concepts.
- La recherche d'un concept particulier s'effectue, comme dans les treillis de Galois, en exploitant la structure algébrique des solutions potentielles.
- La méthode de l'espace des versions est d'une grande importance historique et méthodologique. Son intérêt pratique a été prouvé, par exemple en inférence grammaticale.

Copyright © 2010 Eyrolles.

Chapitre 5

La programmation logique inductive

La programmation logique inductive (PLI) réalise l'apprentissage de formules de la logique des prédicats à partir d'exemples et de contre-exemples. L'enjeu est de construire des expressions logiques comportant des variables liées les unes aux autres. Par exemple, à partir de la description des liens de parenté dans quelques familles (« Jean est le père de Pierre », « Paul est le père de Jean », « Paul est le grand-père de Pierre »...), un programme de PLI doit être capable de trouver une formule du type « Pour tous les x et z tels que z est le grand-père de y, il existe x tel que x est le père de y et y est le père de z ».

Ce type d'apprentissage est difficile à réaliser. On se limite la plupart du temps à un sous-ensemble de la logique des prédicats qu'on appelle « programme logique », en référence aux langages de programmation du type Prolog qui travaillent directement dans ce langage. La PLI a deux caractéristiques fortes : d'abord, le langage de représentation des hypothèses est très bien connu mathématiquement et algorithmiquement. La notion de généralisation peut donc être introduite en cohérence avec les outils de la logique, comme la démonstration automatique. Ensuite, du fait de la richesse de ce langage, la combinatoire de l'apprentissage est très grande : il s'agit d'explorer un espace immense en faisant constamment des choix qu'il sera difficile de remettre en question. C'est pourquoi il est important en PLI de bien décrire les biais d'apprentissage qui limitent cette exploration.

Comme le langage de description des concepts est riche, la PLI peut s'appliquer à un grand nombre de situations. En pratique, les algorithmes permettent d'apprendre des concepts opératoires dans des domaines aussi variés que le traitement de la langue naturelle, la chimie, le dessin industriel, la fouille de données, etc.

Copyright © 2010 Eyrolles.

Sommaire

1 La programmation logique inductive : le cadre général 165
1.1 Complexité de l'induction et expressivité du langage d'hypothèses . . . 165
1.2 La relation de couverture en logique du premier ordre 166
1.3 La subsomption en logique du premier ordre 168
1.3.1 La θ-subsomption 168
1.3.2 L'implication 169
1.3.3 La subsomption des *théories* 169
1.3.4 La subsomption relative à une théorie 170
1.4 Un résumé des relations de subsomption possibles 170
2 La logique des prédicats et les programmes logiques : terminologie 171
2.1 La syntaxe de la logique des prédicats 171
2.1.1 Le langage de la logique des prédicats 172
2.1.2 Le langage des clauses et des programmes logiques 172
2.2 Système de preuve pour les langages de clauses 173
2.2.1 La substitution 174
2.2.2 L'unification 174
2.2.3 L'unificateur le plus général 174
2.2.4 La résolution 174
2.2.5 Résolution et interprétation d'un programme Prolog 175
3 La structuration de l'espace des hypothèses en logique des prédicats 175
3.1 Le calcul de la *lgg* pour la θ-subsomption 175
3.2 Le calcul de *rlgg* pour la θ-subsomption relative 177
3.3 Le calcul de *lgg* pour la résolution inverse 179
4 L'exploration de l'espace des hypothèses 182
4.1 Le squelette des algorithmes de PLI 183
4.1.1 Stratégies de recherche 184
4.1.2 L'élagage de l'espace de recherche 184
4.2 Les biais de recherche dans l'espace d'hypothèses 184
4.2.1 Les biais syntaxiques 185
4.2.2 Les biais sémantiques 185
5 Deux exemples de systèmes de PLI 186
5.1 Un système empirique descendant : FOIL 186
5.2 Un système empirique ascendant : PROGOL 189
6 La probabilisation de la PLI 190
7 Les domaines d'application de la PLI 191
8 Les chantiers de la PLI 193
8.1 Une transition de phase rédhibitoire? 193
8.2 Quelques problèmes ouverts en PLI 195

L'ORNITHOLOGIE s'apprend vite : le débutant que nous avons rencontré dans l'avant-propos de ce livre sait maintenant reconnaître nombre d'espèces : des rapaces, comme l'épervier et le faucon pèlerin ou des palmipèdes, comme la sarcelle et la bernache. Il rencontre à nouveau l'expert, qui se montre satisfait de ses progrès (mais conscient de ses limites) et qui lui propose un nouvel exercice : apprendre à reconnaître

Copyright © 2010 Eyrolles.

le sexe de l'animal (sans dissection) sur des espèces où seule la taille peut éventuellement aider. L'ensemble d'apprentissage proposé par l'expert donne pour mesure l'envergure en centimètres. Il est le suivant :

	espèce	envergure	sexe
x_1	épervier	60	mâle
x_2	épervier	80	femelle
x_3	pèlerin	90	mâle
x_4	pèlerin	110	femelle
x_5	sarcelle	70	mâle
x_6	sarcelle	70	femelle

Le débutant s'aperçoit rapidement qu'il ne sait pas trouver un concept pour distinguer les mâles des femelles. C'est très simple à voir : les exemples x_5 et x_6 sont décrits par les mêmes attributs et sont associés à des étiquettes contradictoires. Il fait part de cette réflexion à l'expert. « D'accord », lui répond ce dernier, « je sais que vous connaissez la méthode de l'espace des versions et que vous essayez de trouver un concept écrit dans un certain langage des hypothèses. Vous avez choisi pour ce langage la logique des propositions sur des sélecteurs portant sur des attributs. N'est-ce pas ? C'est en effet impossible d'apprendre quoi que ce soit dans ce langage sur ces données. Je vais vous initier à une autre façon de faire de l'apprentissage. »

L'expert propose alors de reformuler les données sous la forme de prédicats, ce qui consiste par exemple à écrire l'attribut « envergure » pour l'exemple x_5 de la manière suivante :

$$envergure(x_5, 70) = VRAI$$

Dans ce formalisme, le premier exemple devient :

$$(epervier(x_1) = VRAI) \wedge (envergure(x_1,60) = VRAI) \wedge (male(x_1) = VRAI)$$

ou plus simplement :

$$epervier(x_1) \wedge envergure(x_1, 60) \wedge male(x_1)$$

L'ensemble des exemples est maintenant l'union (le OU ou $\vee$) d'expressions de ce type, c'est-à-dire une formule logique formée de la disjonction des exemples, chacun écrit comme une conjonction ET ($\wedge$) de prédicats sur ses attributs et sa supervision.

« Et alors ? Est-ce que la contradiction entre les exemples x_5 et x_6 a disparu pour autant ? » demande le débutant ? « Non », répond l'expert, « vous avez raison, ce n'est qu'une réécriture pour l'instant. Mais je vais vous donner un peu de matériel supplémentaire. »

Il propose alors d'utiliser deux nouveaux prédicats *env-sup*(x, y) et *meme-esp*(x, y) avec la signification suivante : *env-sup*(x, y) est $VRAI$ quand l'envergure de l'exemple noté x est strictement supérieure à celle de l'exemple noté y et *meme-esp*(x, y) est $VRAI$ si l'exemple x est de la même espèce que l'exemple y. Les exemples deviennent alors, en notant pour raccourcir :

epervier(x) par $E(x)$

pelerin(x) par $P(x)$

sarcelle(x) par $S(x)$

envergure$(x, 60)$ par $e(x, 60)$, etc.

Copyright © 2010 Eyrolles.

	$E(\mathbf{x})$	$P(\mathbf{x})$	$S(\mathbf{x})$	$B(\mathbf{x})$	$e(\mathbf{x}, 60)$	$e(\mathbf{x}, 70)$
$\mathbf{x}_1$	*VRAI*	*FAUX*	*FAUX*	*FAUX*	*VRAI*	*FAUX*
$\mathbf{x}_2$	*VRAI*	*FAUX*	*FAUX*	*FAUX*	*FAUX*	*FAUX*
$\mathbf{x}_3$	*FAUX*	*VRAI*	*FAUX*	*FAUX*	*FAUX*	*FAUX*
$\mathbf{x}_4$	*FAUX*	*VRAI*	*FAUX*	*FAUX*	*FAUX*	*FAUX*
$\mathbf{x}_5$	*FAUX*	*FAUX*	*VRAI*	*FAUX*	*FAUX*	*VRAI*
$\mathbf{x}_6$	*FAUX*	*FAUX*	*VRAI*	*FAUX*	*FAUX*	*VRAI*

	$e(\mathbf{x}, 80)$	$e(\mathbf{x}, 90)$	$e(\mathbf{x}, 100)$	$e(\mathbf{x}, 110)$	$male(\mathbf{x})$	$fem(\mathbf{x})$
$\mathbf{x}_1$	*FAUX*	*FAUX*	*FAUX*	*FAUX*	*VRAI*	*FAUX*
$\mathbf{x}_2$	*VRAI*	*FAUX*	*FAUX*	*FAUX*	*FAUX*	*VRAI*
$\mathbf{x}_3$	*FAUX*	*VRAI*	*FAUX*	*FAUX*	*VRAI*	*FAUX*
$\mathbf{x}_4$	*FAUX*	*FAUX*	*FAUX*	*VRAI*	*FAUX*	*VRAI*
$\mathbf{x}_5$	*FAUX*	*FAUX*	*FAUX*	*FAUX*	*VRAI*	*FAUX*
$\mathbf{x}_6$	*FAUX*	*FAUX*	*FAUX*	*FAUX*	*FAUX*	*VRAI*

Il faut également écrire la table de vérité des prédicats relationnels *env-sup* et *meme-esp*. La seconde est évidente et la première débute ainsi :

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
$\mathbf{x}_1$	*FAUX*	*FAUX*	*FAUX*	*FAUX*	*FAUX*	*FAUX*
$\mathbf{x}_2$	*VRAI*	*FAUX*	*FAUX*	*FAUX*	*VRAI*	*VRAI*
$\mathbf{x}_3$	*VRAI*	*VRAI*	*FAUX*	*FAUX*	*VRAI*	*VRAI*
...	...	...	...	...	...	...

Quoiqu'un peu impressionné par la quantité plus importante de données, le débutant tente l'apprentissage d'un concept dans le même langage. Il obtient par exemple :

$$\begin{aligned} male &= male(\mathbf{x}_1) \vee male(\mathbf{x}_3) \vee male(\mathbf{x}_5) \\ &= [E(\mathbf{x}_1) \wedge e(\mathbf{x}_1, 60) \wedge] \vee [P(\mathbf{x}_3) \wedge e(\mathbf{x}_3, 90)] \vee [S(\mathbf{x}_5) \wedge e(\mathbf{x}_5, 70)] \end{aligned}$$

Mais cela ne le satisfait pas, car il n'y a aucune généralisation. Il essaie donc d'utiliser la connaissance supplémentaire fournie par l'expert et obtient ceci, après avoir essayé un grand nombre de formules :

$$\begin{aligned} & env\text{-}sup(\mathbf{x}_2, \mathbf{x}_1) \wedge E(\mathbf{x}_1) \wedge E(\mathbf{x}_2) \wedge male(\mathbf{x}_1) \wedge fem(\mathbf{x}_2) \\ \wedge \quad & env\text{-}sup(\mathbf{x}_4, \mathbf{x}_3) \wedge P(\mathbf{x}_3) \wedge P(\mathbf{x}_4) \wedge male(\mathbf{x}_3) \wedge fem(\mathbf{x}_4) \\ \wedge \quad & meme\text{-}esp(\mathbf{x}_1, \mathbf{x}_2) \wedge meme\text{-}esp(\mathbf{x}_3, \mathbf{x}_4) \end{aligned}$$

« Bon, dit l'expert, nous y sommes presque. Nous savons tous les deux que les éperviers et les faucons sont des rapaces, mais pas les sarcelles. Pouvez-vous utiliser cette information ? » « Voyons cela », répond le débutant :

$$\begin{aligned} & env\text{-}sup(\mathbf{x}_2, \mathbf{x}_1) \wedge rapace(\mathbf{x}_1) \wedge rapace(\mathbf{x}_2) \wedge male(\mathbf{x}_1) \wedge fem(\mathbf{x}_2) \\ \wedge \quad & env\text{-}sup(\mathbf{x}_4, \mathbf{x}_3) \wedge rapace(\mathbf{x}_3) \wedge rapace(\mathbf{x}_4) \wedge male(\mathbf{x}_3) \wedge fem(\mathbf{x}_4) \\ \wedge \quad & meme\text{-}esp(\mathbf{x}_1, \mathbf{x}_2) \wedge meme\text{-}esp(\mathbf{x}_3, \mathbf{x}_4) \end{aligned}$$

Copyright © 2010 Eyrolles.

« Cette fois, c'est presque bon. Je vous donne une dernière clé. Que dites-vous de la formule suivante ? »

$$env\text{-}sup(\mathbf{X}, \mathbf{Y}) \wedge rapace(\mathbf{X}) \wedge rapace(\mathbf{Y}) \wedge meme\text{-}esp(\mathbf{X}, \mathbf{Y}) \wedge male(\mathbf{Y}) \wedge fem(\mathbf{X})$$

« Elle signifie que, pour toutes les espèces de rapaces (et non pour les autres oiseaux), la femelle d'une espèce est d'envergure supérieure au mâle » répond le débutant « mais je n'aurais pas su la trouver : je ne connaissais ni les prédicats relationnels ni les expressions avec des variables ».

Notations utiles pour le chapitre

$\models$	Relation d'implication sémantique (théorie des modèles)
$\vdash$	Relation d'implication logique (théorie de la preuve)
$\models_T$	Relation d'implication relative à la théorie T
$\wedge$	ET : conjonction
$\vee$	OU : disjonction
$\neg$	Négation
$h_1 \succcurlyeq h_2$	L'hypothèse h_1 est plus générale que (ou subsume) h_2

1. La programmation logique inductive : le cadre général

1.1 Complexité de l'induction et expressivité du langage d'hypothèses

Le chapitre 4 a montré comment on peut formaliser le problème de l'induction de concept, c'est-à-dire de l'apprentissage d'une fonction indicatrice, à valeur dans $\{0, 1\}$. L'idée essentielle est de considérer l'ensemble de toutes les hypothèses cohérentes[1] avec l'échantillon d'apprentissage $\mathcal{S} = \{(\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), \ldots (\mathbf{x}_m, u_m)\}$. Cet espace est appelé l'espace des versions. Sa mise à jour s'effectue incrémentalement à chaque fois qu'un nouvel exemple d'apprentissage devient disponible. Elle passe par la détermination de la plus petite généralisation (respectivement spécialisation) d'une hypothèse et du nouvel exemple lors de l'adaptation du S-set (respectivement du G-set). Nous avons vu dans le chapitre 4 que les concepts de généralisation et de spécialisation se définissent par référence à la relation d'inclusion entre les *extensions* des concepts, c'est-à-dire l'ensemble des objets de $\mathcal{X}$ qu'ils couvrent. Un concept est dit plus spécifique qu'un autre si son extension est incluse dans l'extension de l'autre. On obtient ainsi une relation d'ordre partiel qui est exploitée dans l'algorithme d'élimination des candidats pour mettre à jour les deux bornes de l'espace des versions : le S-set et le G-set.

Cette approche de l'apprentissage soulève deux problèmes. Le premier est que la notion de couverture d'un exemple par une hypothèse n'est pas toujours aussi simple que le chapitre 4 peut le laisser penser. Nous y revenons dans la section suivante. Le deuxième problème est que les régularités observées sur les parties de $\mathcal{X}$ ne se transportent pas complètement dans l'espace des concepts $\mathcal{H}$ défini par le langage $\mathcal{L}_\mathcal{H}$ d'expression des hypothèses. Ainsi, l'ensemble des parties de

[1] C'est-à-dire, rappelons-le, couvrant tous les exemples positifs et excluant tous les exemples négatifs de cet échantillon.

Copyright © 2010 Eyrolles.

$\mathcal{X}$ forme une algèbre, ce qui signifie qu'il existe des opérations bien définies pour calculer le plus petit ensemble d'exemples contenant deux ensembles d'exemples, de même que pour calculer le plus grand ensemble d'exemples contenus à la fois dans deux ensembles d'exemples. Ce sont les opérations classiques d'union et d'intersection. Malheureusement, les opérations correspondantes sur l'espace des concepts $\mathcal{H}$: la *plus petite généralisation* (*least general generalization, lgg*) et la *spécialisation maximale* (*most general specialization, mgs*) ne sont pas en général définies de manière unique. Cela provient du fait que tout ensemble d'exemples ne correspond pas forcément à un concept, de même que toute expression dans le langage des hypothèses $\mathcal{L}_\mathcal{H}$ n'a pas nécessairement une contrepartie dans $\mathcal{X}$. Nous avons en effet vu la nécessité de l'existence d'un biais de langage, limitant la richesse de $\mathcal{H}$. De ce fait, la complexité de l'apprentissage de concept dépend fortement de $\mathcal{L}_\mathcal{H}$, le langage d'expression des hypothèses.

En fonction du langage $\mathcal{L}_\mathcal{H}$, il peut être possible, ou impossible, de définir une relation $\succcurlyeq$ sur $\mathcal{L}_\mathcal{H} \times \mathcal{L}_\mathcal{H}$ de généralité intensionnelle (dans $\mathcal{H}$), appelée *subsomption*, qui coïncide avec la relation d'inclusion dans $\mathcal{X}$. Si $h_1 \succcurlyeq h_2$ implique que h_1 est aussi générale que h_2, la relation de subsomption est dite *saine* (*sound*) ; si h_1 est aussi générale que h_2 implique que $h_1 \succcurlyeq h_2$, alors la relation de subsomption est dite *complète* (*complete*). Lorsque la relation de subsomption est saine, on peut démontrer que l'espace des versions correspondant à un ensemble d'exemples est convexe par rapport à la relation $\succcurlyeq$. Grâce à cela, on peut alors représenter l'espace des versions par une borne inférieure, le S-set, et par une borne supérieure, le G-set (voir le chapitre 1).

La complexité de l'induction supervisée dépend donc du langage d'expression des hypothèses $\mathcal{H}$. Dans le chapitre 4, nous avons essentiellement fait référence à des langages d'hypothèses en attributs-valeurs. Ceux-ci sont souvent insuffisants pour décrire des domaines dans lesquels il est nécessaire de pouvoir décrire des relations (comme dans le domaine des arches décrits dans le chapitre 2 ou dans l'exemple introductif avec les relations de comparaison d'envergure). C'est pourquoi on est tenté d'utiliser la logique des prédicats ou logique du premier ordre qui permet d'exprimer des concepts relationnels. Mais est-il alors raisonnable de vouloir pratiquer de l'induction avec un tel langage d'expression des hypothèses? Nous examinons le prix à payer dans la suite.

1.2 La relation de couverture en logique du premier ordre

L'objectif de la programmation logique inductive (PLI) est de construire des programmes logiques à partir d'exemples supervisés. Schématiquement, un programme logique est un ensemble de règles de la forme *prémisses* $\rightarrow$ *conclusion*. Selon que l'on autorise l'usage de la négation pour formuler les prémisses ou non, on parle de programmes normaux ou de programmes définis. Étant donné leur avantage en terme de pouvoir expressif, nous étudierons dans ce chapitre l'apprentissage des programmes normaux. Par ailleurs, deux grandes familles d'approches sont considérées en programmation logique inductive :

1. L'*apprentissage empirique* dans lequel on dispose de nombreux exemples et contre-exemples pour apprendre un nouveau concept (i.e. un programme). On cherche alors à apprendre une définition qui permette d'expliquer (couvrir) tous les exemples positifs connus mais aucun contre-exemple.
2. L'*apprentissage interactif*, dans lequel on cherche à adapter une description (ou théorie) du domaine en fonction de quelques nouveaux exemples et contre-exemples. On parle aussi dans ce cas de *révision de connaissances*.

Le deuxième type d'apprentissage implique en général des raisonnements et des mécanismes de généralisation beaucoup plus complexes, et malheureusement moins maîtrisés que dans le

Copyright © 2010 Eyrolles.

premier. Pour des raisons de place, nous nous limitons donc dans ce chapitre à l'étude de l'apprentissage empirique.

Dans le langage de description des hypothèses par attributs-valeurs, une expression peut prendre la forme :

$$(Taille = grande) \wedge (Couleur = rouge) \wedge (Forme = carré)$$

dénotant le concept « grands carrés rouges ». La même expression sert à la fois à dénoter un exemple (un certain « grand carré rouge ») et un ensemble d'exemples (tous les grands carrés rouges). La logique par attributs-valeurs n'est pas capable de faire la distinction entre les exemples et les hypothèses. Cela révèle un manque de pouvoir expressif, mais permet l'astuce de la représentation unique (*single representation trick*), ce qui signifie en pratique que le test de couverture d'un exemple par une hypothèse est le même que le test de subsomption entre deux hypothèses. Ce test est aisé dans le cas de la logique des attributs-valeurs et de la logique des propositions en général.

Prenons le cas d'expressions conjonctives en logique attributs-valeurs. Le test de subsomption entre deux expressions revient à contrôler que chaque paire attribut-valeur qui apparaît dans l'expression la plus générale apparaît aussi dans l'autre. Le processus de généralisation d'une expression consiste à monter dans la hiérarchie définie par la relation de subsomption. La généralisation revient donc à abandonner un ou plusieurs termes de la conjonction. Par exemple, si le concept :

$$(Taille = moyenne) \wedge (Couleur = rouge) \wedge (Forme = cercle)$$

doit être généralisé pour couvrir l'exemple :

$$(Taille = petite) \wedge (Couleur = rouge) \wedge (Forme = cercle) \wedge (Poids = lourd)$$

il suffit de laisser tomber $(Taille = moyenne)$ pour obtenir le concept adéquat :

$$(Couleur = rouge) \wedge (Forme = cercle)$$

Il est à noter que la généralisation obtenue est unique et minimale (*lgg*).

De la même manière, spécialiser une expression revient à ajouter une paire attribut-valeur dans la conjonction. Par exemple, si le concept :

$$(Couleur = rouge) \wedge (Forme = cercle)$$

ne doit pas couvrir l'exemple négatif :

$$(Taille = grande) \wedge (Couleur = rouge) \wedge (Forme = cercle)$$

il suffit d'ajouter au concept une paire d'attributs-valeurs absente de l'exemple négatif pour le spécialiser assez. Il faut noter qu'en revanche il n'y a pas ici de spécialisation maximale unique.

En logique des prédicats, il est nécessaire de reconsidérer les notions de couverture, de subsomption, et par voie de conséquence les opérations de généralisation et de spécialisation.

Considérons à nouveau des concepts conjonctifs, mais cette fois exprimés en logique du premier ordre. Par exemple, en utilisant la syntaxe des programmes Prolog[2], le `concept1` à apprendre pourrait se représenter[3] par :

[2] Dans cette syntaxe, le symbole `:-` signifie l'implication du membre gauche par le membre droit et la virgule dans le membre droit est la conjonction.

[3] Soit, en notation logique traditionnelle : `rouge(X)` $\wedge$ `cercle(X)` $\rightarrow$ `concept1(X)`.

Copyright © 2010 Eyrolles.

```
concept1(X) :- rouge(X), cercle(X).
```

Un concept peut être défini par plusieurs clauses, par exemple :

```
concept2(X) :- rouge(X), carré(X).
concept2(X) :- vert(X), cercle(X).
```

ce qui signifie que le concept correspond aux objets qui sont soit des carrés rouges, soit des cercles verts.

Les exemples sont décrits par des *atomes clos* (sans variable), par exemple :

```
petit(petitcerclerouge).
rouge(petitcerclerouge).
cercle(petitcerclerouge).
```

Ici, `petitcerclerouge` est un exemple de `concept1` car `concept1(petitcerclerouge)` peut être prouvé. D'une manière générale, étant donnée une conjonction d'atomes clos `Desc(Exemple)` décrivant `Exemple`, un concept `Concept(X) :- Conditions(X)` classe `Exemple` positivement si :

$T \wedge$ `Desc(Exemple)` $\wedge$ `(Concept(X) :- Conditions(X))` $\models$ `Concept(Exemple)`.

On suppose donc de manière générale que l'on dispose d'une connaissance initiale T, ou *théorie du domaine,* mise sous forme de programme logique. Pour tester si un exemple est couvert par un concept, on ajoute sa description dans la théorie du domaine, et on chercher à prouver[4] `Concept(Exemple)` à l'aide de la définition du concept. Si la preuve échoue, on interprète cet échec comme une classification négative (négation par l'échec). Dans ce cas, il faudra modifier la définition du concept. Pour cela il faut examiner la notion de subsomption en logique du premier ordre.

1.3 La subsomption en logique du premier ordre

Il s'agit maintenant de définir la relation de généralité ou subsomption entre clauses. Par exemple, on s'attend à ce que la clause :

```
concept3(X) :- rouge(X), cercle(X).
```

subsume, donc soit plus générale, que la clause :

```
concept3(X) :- petit(X), rouge(X), cercle(X).
```

En effet, on constate[5] que l'extension correspondant à la première définition inclut l'extension correspondant à la seconde définition. Il se trouve d'ailleurs que la seconde clause contient les mêmes littéraux que la première et qu'elle est donc, selon notre définition de la section précédente, plus spécifique. Cependant, ce cas n'épuise pas la notion de subsomption de deux clauses. Prenons par exemple les deux clauses suivantes :

```
concept4(X) :- carré(X), triangle(Y), mêmecouleur(X,Y).
concept4(X) :- carré(X), triangle(t), mêmecouleur(X,t).
```

La première clause décrit l'ensemble des carrés de même couleur que les triangles existants. La seconde clause décrit l'ensemble des carrés ayant la même couleur qu'un triangle particulier `t`. Il est évident que le deuxième ensemble est inclus dans le premier et donc que la première clause subsume la seconde. Il faut donc rendre compte aussi de ce cas de subsomption.

1.3.1 La θ-subsomption

En combinant les deux cas précédents, on arrive ce qu'on appelle la θ-subsomption, qui se définit informellement ainsi :

[4] Le symbole $\models$ correspond à la notion sémantique de l'implication. Nous reviendrons sur ce formalisme dans la section suivante.

[5] Nous restons volontairement informels dans cette première exposition des notions de couverture et de subsomption.

Copyright © 2010 Eyrolles.

Clause1 subsume Clause2 s'il existe une substitution θ applicable à Clause1 et telle que tous les littéraux dans la clause ainsi obtenue apparaissent dans Clause2.

Il est à noter que si Clause1 θ-subsume Clause2, alors on a aussi Clause1 $\models$ Clause2. En revanche, la réciproque n'est pas toujours vraie, comme le montre l'exemple suivant :

```
list([V|W]) :- list(W).
list([X,Y|Z]) :- list(Z).
```

À partir de la liste vide, la première clause construit des listes de n'importe quelle longueur, tandis que la seconde construit des listes de longueur paire. Toutes les listes construites par la seconde clause peuvent aussi l'être par la première, qui est donc plus générale. Pourtant, il n'y a pas de substitution applicable à la première clause et permettant d'obtenir la seconde (une telle substitution devrait appliquer W à la fois sur [Y|Z] et sur Z, ce qui est impossible). La θ-subsomption est donc plus faible que l'implication. Elle a en outre des limitations rédhibitoires si l'on veut induire des clauses récursives. Soit en effet, les clauses : p(f(f(a))) :- p(a) et p(f(b)) :- P(b). Si l'on cherche la plus petite généralisation par rapport à la θ-subsomption, on trouve la clause : p(f(Y)) :- p(X), tandis que la clause p(f(X)) :- p(X), plus satisfaisante, ne peut être trouvée. Le problème est que la θ-subsomption ne peut pas prendre en compte les clauses qui peuvent être résolues avec elles-mêmes.

1.3.2 L'implication

On pourrait envisager d'utiliser l'implication pour définir la subsomption entre clauses :

Clause1 subsume Clause2 si Clause1 $\models$ Clause2

Cela introduit cependant deux problèmes. Le premier est qu'il s'agit d'une définition sémantique (s'appuyant sur la théorie des modèles) et qu'il reste donc à préciser la procédure effective de preuve de subsomption ainsi que la procédure permettant de généraliser une clause. Le second problème est que la plus petite généralisation (*lgg*) n'est pas toujours unique si la subsomption est définie comme l'implication logique. Soit par exemple les deux clauses :

```
list([A,B|C]) :- list(C).
list([P,Q,R|S]) :- list(S).
```

Selon l'implication logique, ces clauses ont deux *lgg* :
list([X|Y]) :- list(Y) et list([X,Y|Z]) :- list(V).
Selon la θ-subsomption, seule cette dernière est une *lgg*. Il est à noter que la première *lgg* est en réalité plus plausible.

1.3.3 La subsomption des *théories*

Jusque-là, nous avons seulement considéré la subsomption entre deux clauses. Dans la plupart des cas intéressants, nous devons cependant prendre en compte des ensembles de clauses décrivant des théories[6] sur le monde. Il faut donc définir aussi la subsomption entre théories. Par exemple, soit la théorie :

```
concept5(X) :- petit(X), triangle(X).
polygone(X) :- triangle(X).
```

Elle est impliquée logiquement par la théorie suivante :

```
concept5(X) :- polygone(X).
polygone(X) :- triangle(X).
```

puisque tout modèle de la seconde est un modèle de la première théorie. Pourtant, la clause :

[6] En programmation logique, une théorie est simplement définie comme un ensemble de clauses.

Copyright © 2010 Eyrolles.

```
concept5(X) :- petit(X), triangle(X).
```

n'est pas logiquement impliquée par la clause :

```
concept5(X) :- polygone(X).
```

La subsomption entre théories ne peut donc pas être réduite à la subsomption entre clauses.

1.3.4 La subsomption relative à une théorie

Nous devons avoir recours dans ce cas à la notion de *subsomption relative à une théorie* entre deux clauses. Par définition : `Clause1` subsume `Clause2` relativement à la théorie T si $T \wedge$ `Clause1` $\models$ `Clause2`, ce que nous notons : `Clause1` $\models_T$ `Clause2`.

Par exemple, supposons que T contienne la clause :

```
polygone(X) :- triangle(X).
```

Nous avons alors :

$T \wedge$ `concept5(X) :- polygone(X)` $\models$ `concept5(X) :- petit(X), triangle(X).`

De cette manière, nous obtenons en effet que :

```
concept5(X) :- polygone(X)
```

subsume :

```
concept5(X) :- petit(X), triangle(X)
```

relativement à T.

1.4 Un résumé des relations de subsomption possibles

Pour résumer, les trois types de relations de subsomption utilisés en programmation logique inductive sont la θ-subsomption, l'implication logique et la subsomption relative à une théorie du domaine. De ces trois types de subsomptions, la première est la plus aisée à réaliser (et elle est déjà NP-complète !). En particulier, la θ-subsomption est décidable, tandis que l'implication logique ne l'est pas[7], même pour des clauses de Horn. De même, la subsomption relative est plus sévère que l'implication : les deux sont indécidables, mais les procédures de preuve pour l'implication ne nécessitent pas la prise en compte de la théorie T, contrairement aux procédures pour la subsomption relative qui doivent prendre en compte toutes les dérivations possibles à partir de $T \wedge$ `Clause`.

En pratique, lors de l'apprentissage à partir d'exemples représentés par des clauses, en présence ou non d'une théorie du domaine, il est essentiel de pouvoir déterminer la plus petite généralisation (relative dans le cas de la subsomption relative) ainsi que leur spécialisation maximale. L'existence de ces généralisations ou spécialisations dépend à la fois du type de subsomption considéré et du langage utilisé : logique des clauses ou logique réduite aux clauses de Horn. Le tableau suivant fournit un résumé des résultats connus sur les six cas possibles (+ correspondant à une réponse positive et − à une réponse négative).

	Clauses de Horn		Clauses générales	
Type de subsomption	*lgg*	*mgs*	*lgg*	*mgs*
θ-subsomption	+	+	+	+
Implication ($\models$)	−	−	+ si sans fonction	+
Implication relative ($\models_T$)	−	−	−	+

FIG. 5.1 : Existence de *lgg* et de *mgs*.

[7] Church en 1932 a montré qu'il ne peut exister d'algorithme pouvant décider en un temps fini si une inférence est logiquement valide ou non en logique des prédicats standard.

Copyright © 2010 Eyrolles.

Ce tableau illustre une fois de plus le compromis existant entre l'expressivité d'un langage et les raisonnements qui peuvent être réalisés. Il est par ailleurs important de noter que l'utilisation d'un langage causal sans symbole de fonction conduit à une structure de treillis. C'est pourquoi la plupart des travaux en PLI se placent dans ce cadre.

Avant d'aborder les moyens effectifs d'exploration de l'espace des hypothèses, il est nécessaire de rappeller plus formellement quelques concepts de base en logique.

2. La logique des prédicats et les programmes logiques : terminologie

Ce paragraphe définit de manière plus rigoureuse ce qu'est une formule de la logique des prédicats, autrement dit quels sont les concepts que nous cherchons à apprendre. Ces concepts sont construits à partir de symboles primitifs (les variables, les connecteurs, les quantificateurs, les prédicats, les fonctions et les parenthèses), en respectant une syntaxe stricte. Cette syntaxe permet, par l'application de certaines règles, de réaliser des démonstrations dans ce système formel, c'est-à-dire de déduire des théorèmes à partir d'axiomes. Finalement, une sémantique doit être proposée pour permettre une interprétation hors de ce système formel qui n'a pas de signification en soi [NS93, GN88, Tau94].

2.1 La syntaxe de la logique des prédicats

Les formules logiques sont écrites dans un langage construit à partir des *symboles primitifs* suivants :

Variable $X, Y \ldots$ Une variable prend ses valeurs sur un *domaine*. Des exemples de domaines sont : l'ensemble des nombres entiers, l'ensemble des clients d'une compagnie d'assurances.

Constante $a, b, \ldots, jerome, laure, \ldots, 1, 2, \ldots, VRAI, FAUX, \ldots$ Un ensemble de constantes forme le domaine d'une variable. Une *instanciation* d'une variable est une constante de son domaine.

Connecteur $\neg, \vee, \wedge, \longrightarrow, \longleftrightarrow$

Quantificateur $\forall, \exists$

Prédicat $P, Q, \ldots, sont-maries, \ldots$

Un prédicat possède une *arité* a (le nombre d'arguments sur lequel il porte) qui doit valoir au moins 1. On note, si besoin est, le prédicat de manière plus complète par : P/a. Un prédicat est une *relation* entre plusieurs domaines de variables, autrement dit une application de l'ensemble de ces domaines dans $\{VRAI, FAUX\}$. Par exemple, pour le prédicat $sont-maries/2 : sont-maries(jerome, laure) = VRAI$.

Fonction $f, g, \ldots, age-ainé \ldots$

Une fonction diffère d'un prédicat par la nature de son résultat, qui peut appartenir à n'importe quel domaine. Par exemple, pour la fonction $age-ainé/2 : age-aine(jerome, X)$ a pour valeurs l'age de l'aîné(e) de $jerome$ (leur mère n'est pas ici précisée). Une fonction d'arité 0 n'est autre qu'une constante.

Parenthèses (,).

Copyright © 2010 Eyrolles.

2.1.1 Le langage de la logique des prédicats

Terme Un *terme* est défini récursivement comme

- soit une constante ;
- soit une variable ;
- soit une fonction appliquée à des termes, c'est-à-dire une expression de la forme : $f(t_1, \ldots, t_m)$ où f est une fonction d'arité m et les t_i sont des termes.

Un exemple de terme : $sont - maries(pere(laure), mauricette)$.

Littéral Un *littéral* est un prédicat appliqué à des termes, éventuellement précédé du symbole $\neg$. C'est donc une expression de la forme : $p(t_1, \ldots, t_m)$ ou $\neg p(t_1, \ldots, t_m)$, où p est un symbole de prédicat et les t_i sont des termes.

Un exemple de littéral : $plus - grand - que(age(pere(X)), age(mauricette))$.

Atome Un littéral positif (c'est-à-dire non précédé du symbole $\neg$) est appelé un *atome*. Un littéral qui ne contient pas de variable est dit *clos* ou *complètement instancié*. Par exemple : $age(pere(jerome))$ est un atome clos.

Formule Une *formule* est définie récursivement :

- Un littéral est une formule.
- Si α et β sont des formules alors $(\alpha \vee \beta)$, $(\alpha \wedge \beta)$, $(\alpha \longrightarrow \beta)$, $(\alpha \longleftrightarrow \beta)$ et $(\neg\alpha)$ sont des formules.
- Si v est une variable et α est une formule, alors $((\exists v)\alpha)$ et $((\forall v)\alpha)$ sont des formules.

Sous-formule Une sous-formule est une suite de symboles d'une formule, mais qui est elle-même une formule.

Variable libre et liée Une variable X est dite *liée* dans une formule ϕ s'il existe dans ϕ une sous-formule commençant par $(\forall X)$ ou par $(\exists X)$. Sinon, X est dite *libre*.

2.1.2 Le langage des clauses et des programmes logiques

Clause Une *clause* est une formule de type particulier : c'est une disjonction finie de littéraux dont toutes les variables sont quantifiées universellement, c'est-à-dire commandées par $\forall$. On simplifie en général l'écriture d'une clause en supprimant les quantificateurs $\forall$: toute variable doit donc être interprétée comme étant universellement quantifiée.

Clause de Horn Une clause *de Horn* est une clause qui a soit zéro soit un seul littéral positif.

Clause définie Une clause *définie* est une clause de Horn qui a exactement un littéral positif. Une clause définie s'écrit donc, quand on a supprimé les quantificateurs universels :

$$A \vee \neg B_1 \vee \ldots \vee \neg B_m$$

On transforme cette notation en :

$$B_1 \wedge \ldots \wedge B_m \longrightarrow A$$

puis, en introduisant un nouveau connecteur et en changeant la notation de la conjonction :

$$A \longleftarrow B_1, \ \ldots \ , B_m$$

Le connecteur $\longleftarrow$ signifie l'implication du membre de gauche par le membre droit. Cette notation est employée dans certaines versions du langage Prolog[8]. Dans la suite de ce chapitre, les clauses seront notées de cette manière.

[8] La notation que nous employons est intermédiaire entre la notation logique et la notation la plus classique en Prolog, qui a été utilisée plus haut. Selon cette dernière, une clause s'écrit : A :- $B_1, \ \ldots \ , B_m$.

Copyright © 2010 Eyrolles.

Tête et corps de clause définie A, le seul littéral positif d'une clause définie est appelé *tête* de la clause et la conjonction B_1 , ... , B_m est le *corps* de la clause.

Une clause *unitaire* est une clause définie qui n'est composée que de son unique littéral positif A ; elle est donc notée $A \longleftarrow$

Clause but Une *clause but* est une clause de Horn qui n'a aucun littéral positif; elle est donc notée : $\longleftarrow B_1, \ \ldots \ , B_m$

Programme logique défini Un programme logique défini, ou pour simplifier un *programme logique*, est un ensemble de clauses définies.

— Exemple —

$$
\begin{array}{l}
enfant(X,Y) \longleftarrow fille(X,Y) \\
enfant(X,Y) \longleftarrow fils(X,Y) \\
fils(laurent, gaston) \longleftarrow \\
fils(julien, laurent) \longleftarrow \\
fille(laure, gaston) \longleftarrow \\
fils(jerome, laure) \longleftarrow \\
grand-parent(gaston, julien) \longleftarrow \\
grand-parent(gaston, jerome) \longleftarrow \\
grand-parent(gaston, julien) \longleftarrow \\
grand-parent(gaston, jerome) \longleftarrow \\
grand-parent(X,Y) \longleftarrow enfant(Z,X), enfant(Y,Z)
\end{array}
$$

Programme Prolog Un programme Prolog est un programme logique défini. Une requête est une clause but.

— Exemple —

```
                  <-- enfant(Z,T)          % une clause but
enfant(X,Y)       <-- fille(X,Y)           % une clause
enfant(X,Y)       <-- fils(X,Y)
                  fils(laurent,gaston) <-- % un fait
                  fils(julien,laurent) <--
                  fille(laure,gaston) <--
                  fils(jerome,laure) <--
```

2.2 Système de preuve pour les langages de clauses

Étant donné un programme logique $\mathcal{P}$, le but est de mener des *raisonnements* à partir de ce programme afin de savoir, par exemple, quels faits sont $VRAI$ étant donné $\mathcal{P}$.

En pratique, la preuve dans un programme Prolog se fait en utilisant la règle d'inférence logique dite de *modus ponens*, qui s'énonce informellement : Si (α) et $(\alpha$ implique $\beta)$ sont $VRAI$, alors (β) est $VRAI$, et se note classiquement :

$$\frac{\alpha \quad \wedge \quad (\alpha \longrightarrow \beta)}{\beta}$$

L'algorithme de *résolution*, formalisé par Robinson [Rob65], est employé pour cette démonstration. Nous allons le présenter après avoir introduit les notions de substitution et d'unification.

Copyright © 2010 Eyrolles.

2.2.1 La substitution

Une *substitution* est une liste finie de paires X_i/t_i, où X_i est une variable et t_i un terme. Si σ est la substitution $\{X_1/t_1, \ldots X_i/t_i, \ldots X_n/t_n\}$, l'ensemble des variables $\{X_1, \ldots X_n\}$ est noté $dom(\sigma)$.

Une substitution σ s'applique à une formule F en remplaçant chaque occurrence des variables de $dom(\sigma)$ par le terme correspondant, le résultat étant noté $F\sigma$.

2.2.2 L'unification

Si (s_i, t_i) est une paire de termes, un *unificateur* est une substitution σ telle que pour tout i : $s_i\sigma = t_i\sigma$.

Si un tel unificateur existe, on dit que les termes (s_i, t_i) peuvent s'unifier.

L'unificateur de deux littéraux $p(s_1, \ldots s_m)$ et $p(t_1, \ldots t_m)$ est un unificateur de l'ensemble des paires de termes $\{(s_i, t_i)\}$.

2.2.3 L'unificateur le plus général

L'*unificateur le plus général* (*upg*) est une substitution σ telle que pour tout unificateur θ, il existe une substitution γ telle que $\sigma\gamma = \theta$.

Il est démontré que l'*upg* de deux clauses est unique à un renommage de variables près.

2.2.4 La résolution

La résolution d'un programme logique consiste en une suite d'étapes, chacune construisant une nouvelle clause à partir de deux. Dans le cas d'un programme Prolog, dont les clauses sont d'une forme particulière (clauses définies et requêtes ne contenant que des littéraux), on peut appliquer une méthode adaptée : la *SLD-résolution.*

Une étape de SLD-résolution[9] est définie comme suit. Soit deux clauses Prolog :

$$C_1 : \quad H_1 \leftarrow A \ , \ a$$
$$C_2 : \quad H_2 \leftarrow b$$

où H_1, a et b sont des conjonctions (éventuellement vides) d'atomes. A est un littéral quelconque du corps de C_1, et a est le reste du corps de C_1.

On dit que C_1 peut être résolue avec C_2 si H_2 et A s'unifient avec un *upg* θ (soit, $A\theta = H_2\theta$). Le résultat de l'étape de résolution est la clause suivante, appelée *résolvante* :

$$Res(C_1, C_2) : \quad H_1\theta \leftarrow b\theta \ , \ a\theta$$

Par cette définition, on impose donc que le littéral résolu s'unifie avec la tête de C_2 et avec un littéral du corps de C_1.

Sur la figure 5.2, on a : $C_1 = fille(X, Y) \leftarrow parent(Y, X)$ et $C_2 = parent(claire, marie)$.

La clause C_1 peut se récrire sous la forme $C_1 = fille(X, Y) \vee \neg parent(Y, X)$. On peut alors appliquer la règle de résolution en choisissant :

- $L_1 = \neg parent(Y, X)$
- $L_2 = parent(claire, marie)$
- $\theta = \{Y/claire, X/marie\}$

La clause résolvante C est alors l'union de :

- $(C_1 - \{L_1\})\theta = fille(marie, claire)$
- $(C_2 - \{L_2\})\theta = \emptyset$

[9] SLD pour : *résolution linéaire avec fonction de sélection pour clauses définies.*

Copyright © 2010 Eyrolles.

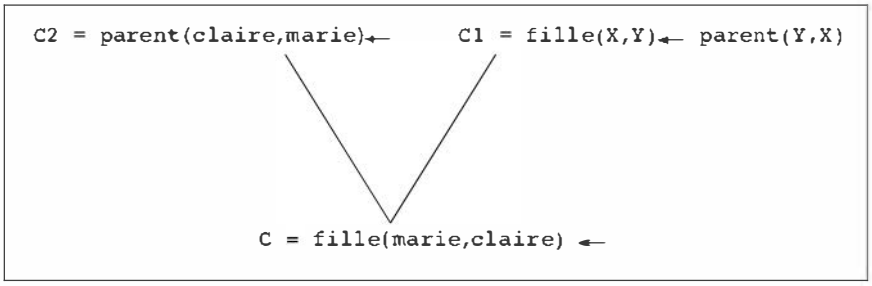


FIG. 5.2: *Une étape de la résolution.*

2.2.5 Résolution et interprétation d'un programme Prolog

Prolog est un langage conçu pour fournir une partie utile et efficace de la technique de démonstration par réfutation. Précisément, Prolog est un système basé sur un démonstrateur de théorème utilisant une forme particulière de résolution : la résolution linéaire avec fonction de sélection pour clauses définies (résolution SLD). Cette stratégie restreint le choix des clauses à chaque étape de résolution, ainsi que le choix du littéral qui est utilisé. Cela correspond à une recherche en profondeur cherchant d'abord à satisfaire chaque sous-but avant de passer au suivant. Si cette stratégie de recherche est systématique et (relativement) efficace, elle ne garantit malheureusement pas la terminaison des démonstrations. Prolog est en ce sens un démonstrateur sans garantie de complétude : des théorèmes vrais peuvent ne pas être démontrés parce que le démonstrateur tombe dans une branche infinie.

Dans le formalisme de Prolog, la connaissance, ou théorie, est transcrite sous forme de faits et de règles considérés comme des axiomes. Les requêtes sont exprimées comme des théorèmes dont on demande au démonstrateur de prouver leur validité dans la théorie.

3. La structuration de l'espace des hypothèses en logique des prédicats

Muni de la relation de couverture et d'une des relations de subsomption définies dans la section 1.2, il est possible d'envisager l'induction de programmes logiques comme une exploration de l'espace des hypothèses, guidée par les relations de subsomption. C'est ainsi par exemple que pour généraliser deux programmes, on cherchera leur plus petite généralisation suivant la relation de subsomption considérée, en faisant l'hypothèse qu'en généralisant minimalement on limite les risques de surgénéralisation. Le raisonnement est le même pour la recherche de spécialisations. Nous allons maintenant examiner les moyens effectifs de calculer des plus petites généralisations et des spécialisations maximales en fonction des relations de subsomption utilisées.

3.1 Le calcul de la *lgg* pour la θ-subsomption

Définition 5.1 (θ-subsomption)

On dit qu'une clause c_1 θ-subsume une clause c_2 si et seulement si il existe une substitution θ telle que $c_1\theta \subseteq c_2$. c_1 est alors une généralisation de c_2 par θ-subsomption.

Copyright © 2010 Eyrolles.

Par exemple, la clause :

$$pere(X,Y) \leftarrow parent(X,Y), etalon(X)$$

θ-subsume la clause :

$$pere(neral, aziza) \leftarrow parent(neral, aziza), etalon(neral), jument(aziza)$$

avec $\theta = \{(X = neral), (Y = aziza)\}$

Plotkin a introduit la notion de *moindre généralisé* ou encore de *plus petit généralisé* [Plo70], un opérateur de généralisation qui permet de généraliser deux clauses. Le calcul de la généralisation la moins générale de deux clauses est défini par les règles suivantes :

- La généralisation la moins générale de deux termes t_1 et t_2, notée $lgg(t_1, t_2)$ (*least general generalization*), est une variable si :
  - au moins un des deux termes est une variable ;
  - un des deux termes est une constante, et $t_1 \neq t_2$;
  - t_1 et t_2 sont deux termes fonctionnels construits sur des symboles de fonction différents.
- Si $t_1 = f(c_1, \ldots, c_n)$ et $t_2 = f(d_1, \ldots, d_n)$, alors

  $$lgg(t_1, t_2) = f(lgg(c_1, d_1), \ldots, lgg(c_n, d_n))$$

  Pour appliquer cette règle, il faut prendre soin de vérifier que si θ_1 et θ_2 sont les deux substitutions telles que $lgg(t_1, t_2)\theta_i = t_i$ $(i \in \{1, 2\})$, alors il ne doit pas exister deux variables distinctes X et Y telles que $X\theta_1 = Y\theta_1$ et $X\theta_2 = Y\theta_2$.
  Par exemple, la généralisation la moins générale de $f(a, b, a)$ et de $f(b, a, b)$ est $f(X, Y, X)$ et non $f(X, Y, Z)$.
- La lgg de deux littéraux construits sur le même symbole de prédicat est donnée par :

  $$lgg(p(t_1, \ldots, t_n), p(s_1, \ldots, s_n)) = p(lgg(t_1, s_1), \ldots, lgg(t_n, s_n))$$

- Enfin, la lgg de deux clauses $l_0 \leftarrow l_1, \ldots, l_n$ et $m_0 \leftarrow m_1, \ldots, m_n$ est une clause qui a pour tête $lgg(l_0, m_0)$, et pour corps l'ensemble des $lgg(l_i, m_i)$ pour tout couple (l_i, m_i) de littéraux de même signe et de même prédicat.

EXEMPLE **La *lgg***

Pour illustrer la *lgg*, considérons les deux scènes constituées d'objets géométriques, représentées sur la figure 5.3 [MB96].

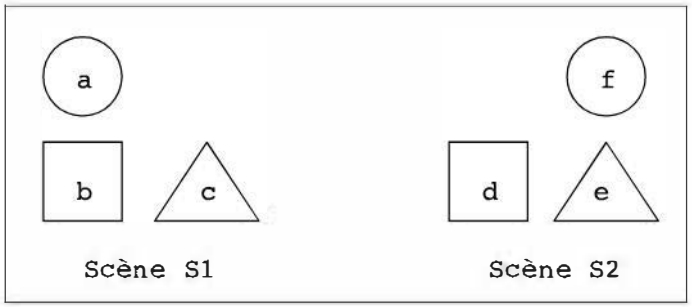


FIG. 5.3: *La généralisation la moins générale.*

Copyright © 2010 Eyrolles.

La première scène représente trois objets a, b, et c. L'objet a est un cercle, b est un carré et c est un triangle. Les objets sont placés de telle façon que a est au-dessus de b, et que b est situé à gauche de c. La deuxième scène représente trois objets d, e, et f décrits de manière similaire. Les deux scènes S_1 et S_2 sont décrites par les deux clauses suivantes qui décrivent le placement de trois figures géométriques :

$$
\begin{array}{lll}
scene(s1) & \leftarrow & \\
 & sur(s1, a, b), & \\
 & a\_gauche(s1, b, c), & \\
 & cercle(a). & \\
 & carre(b). & \\
 & triangle(c). &
\end{array}
\qquad
\begin{array}{lll}
scene(s2) & \leftarrow & \\
 & sur(s2, f, e), & \\
 & a\_gauche(s2, d, e), & \\
 & cercle(f). & \\
 & carre(d). & \\
 & triangle(e). &
\end{array}
$$

La *lgg* des clauses représentant les deux scènes de la figure 5.3 est la clause suivante :

$$
\begin{array}{l}
scene(S) \leftarrow \\
\quad sur(S, A, B), a\_gauche(S, C, D), \\
\quad cercle(A), carre(C), triangle(D).
\end{array}
$$

Cette clause généralise les deux clauses précédentes et traduit le fait que dans les deux scènes, l'objet circulaire se trouve au-dessus d'un autre objet, et que l'objet carré est à gauche de l'objet triangulaire.

3.2 Le calcul de *rlgg* pour la θ-subsomption relative

On définit la plus petite généralisation relative (*rlgg*), (*relative least general generalization*) de deux clauses en utilisant la θ-subsomption relative de manière tout à fait analogue.

Définition 5.2 (θ-subsomption relative)

Soit P un programme logique. Une clause c_1 θ-subsume une clause c_2 relativement à P si et seulement si il existe une substitution θ telle que $P \models c_1\theta \rightarrow c_2$.

Définition 5.3 (*rlgg* de deux exemples)

Soient deux exemples e_1 et e_2 qui sont des atomes liés. Soit une théorie du domaine $T = a_1 \wedge \ldots \wedge a_n$ où les a_i sont également des atomes liés. La rlgg *de e_1 et e_2 est donnée par :*

$$rlgg(e_1, e_2) = lgg(e_1 \longleftarrow T, e_2 \longleftarrow T)$$

avec :

$$
\begin{array}{rcl}
e_i \longleftarrow T & = & \neg T \vee e_i \\
 & = & \neg(a_1 \wedge \ldots \wedge a_n) \vee e_i \\
 & = & \neg a_1 \vee \ldots \vee \neg a_n \vee e_i
\end{array}
$$

EXEMPLE **Le tri rapide**

Nous illustrons le calcul de la *rlgg* de deux clauses en considérant l'exemple du tri rapide (*quick sort*) [MF90]. La théorie du domaine, qui traduit la connaissance initiale sur le problème est

Copyright © 2010 Eyrolles.

constituée d'instances de littéraux construits sur les prédicats $partition/4$ et $append/3$:

$partition(1, [], [], []) \leftarrow\!-$
$partition(2, [4, 3, 1, 0], [1, 0], [4, 3]) \leftarrow\!-$
...
$append([], [1], [1]) \leftarrow\!-$
$append([0, 1], [2, 3, 4], [0, 1, 2, 3, 4]) \leftarrow\!-$
...

On dispose d'un certain nombre d'exemples de tris réalisés à l'aide du prédicat $qsort/2$:

$qsort([], []) \leftarrow\!-$
$qsort([1, 0], [0, 1]) \leftarrow\!-$
$qsort([4, 3], [3, 4]) \leftarrow\!-$
...

On dispose aussi de deux exemples positifs e_1 et e_2 :

$$e_1 = qsort([1], [1])$$
$$e_2 = qsort([2, 4, 3, 1, 0], [0, 1, 2, 3, 4])$$

La *rlgg* de e_1 et e_2 est donnée par la définition 5.3. :

$$rlgg(e_1, e_2) = lgg(e_1 \leftarrow T, e_2 \leftarrow T)$$

où T est la conjonction de tous les atomes liés de la théorie du domaine et des exemples. D'où :

$e_1 \leftarrow T = qsort([1], [1]) \leftarrow$
$append([], [1], [1]), append([0, 1], [2, 3, 4], [0, 1, 2, 3, 4]), \ldots,$
$partition(1, [], [], []), partition(2, [4, 3, 1, 0], [1, 0], [4, 3]), \ldots,$
$qsort([], []), qsort([1, 0], [0, 1]), qsort([4, 3], [3, 4])$
...

et $e_2 \leftarrow T = qsort([2, 4, 3, 1, 0], [0, 1, 2, 3, 4]) \leftarrow$
$append([], [1], [1]), append([0, 1], [2, 3, 4], [0, 1, 2, 3, 4]), \ldots,$
$partition(1, [], [], []), partition(2, [4, 3, 1, 0], [1, 0], [4, 3]), \ldots,$
$qsort([], [])$
$qsort([1, 0], [0, 1]), qsort([4, 3], [3, 4])$
...

La *rlgg* des atomes e_1 et e_2 est donc définie par :

$rlgg(e_1, e_2) = qsort([A|B], [C|D]) \leftarrow$
$append(E, [A|F], [C|D]), append([], [1], [1]),$
$append(G, [H|J], [J|K]), append([0, 1], [2, 3, 4], [0, 1, 2, 3, 4]),$
$partition(A, B, L, M), partition(1, [], [], []),$
$partition(H, N, O, P), partition(2, [4, 3, 1, 0], [1, 0], [4, 3]),$
$qsort(L, E), qsort([], []),$
$qsort(O, G), qsort([1, 0], [0, 1]),$
$qsort(M, F), qsort(P, I), qsort([4, 3], [3, 4]),$
...

Copyright © 2010 Eyrolles.

La construction de la *rlgg* pose un certain nombre de problèmes. En effet, comme nous l'avons déjà évoqué, les atomes de la théorie du domaine T utilisés dans le calcul de la *rlgg* doivent être des atomes liés. Cela impose une définition en extension de la théorie du domaine, ce qui n'est pas concevable pour la majorité des problèmes à résoudre. Pour résoudre ce problème, [Bun88] suggère de calculer cette définition en extension à partir du plus petit modèle de Herbrand de la connaissance initiale exprimée en intension. Cette méthode n'est pas entièrement satisfaisante, puisqu'elle risque d'omettre un certain nombre d'instances de la théorie du domaine et fausser le résultat de l'algorithme d'apprentissage. Plotkin, quant à lui, propose une méthode pour supprimer les littéraux logiquement redondants [Plo71a] [Plo71b]. Malheureusement, la détection des littéraux redondants est coûteuse puisqu'elle nécessite la mise en place de techniques de preuve de théorème. De plus, les clauses débarassées de leurs littéraux redondants peuvent encore contenir un grand nombre de littéraux.

3.3 Le calcul de *lgg* pour la résolution inverse

Plutôt que de définir la subsomption par référence à la sémantique, Muggleton a proposé de la définir à partir de la technique de preuve, qui en logique des prédicats, est le principe de résolution de Robinson utilisé dans Prolog. Dans ce cadre, on dira qu'une clause subsume une autre clause si elle permet la déduction de celle-ci par SLD-résolution. Plus formellement :

Définition 5.1 (SLD-subsomption)

Soit $\mathcal{P}$ un programme logique ; une clause C est plus générale qu'une clause D au sens de la SLD-subsomption relativement à $\mathcal{P}$ ssi $C, \mathcal{P} \vdash_{SLD} D$.

En dehors du fait qu'elle prend en compte la théorie du domaine sous la forme du programme $\mathcal{P}$, l'un des intérêts de cette définition est qu'elle induit, comme la θ-subsomption, des opérations syntaxiques sur les programmes qui sont les fondements de la technique d'apprentissage par inversion de la résolution. Par ailleurs, la justesse de la SLD-résolution garantit que ce qui dérive d'une théorie par SLD-résolution est une conséquence logique de cette théorie. En bref, si $C, \mathcal{P} \vdash_{SLD} D$, alors $C, \mathcal{P} \models D$, ce qui fait de la SLD-subsomption un cas particulier « constructif » de l'implication relative à une théorie.

On peut alors dériver analytiquement l'inversion de la résolution à partir de la règle de résolution exprimée par l'équation 5.1. Tout d'abord, θ peut toujours s'écrire sous la forme d'une composition de substitutions θ_1 et θ_2, avec θ_i contenant les variables de C_i. L'équation s'écrit donc :

$$C = (C_1 - \{L_1\})\theta_1 \cup (C_2 - \{L_2\})\theta_2 \tag{5.1}$$

On restreint l'inversion de résolution à l'inférence de clauses C_2 qui ne contiennent aucun littéral en commun avec C_1.

$$C - (C_1 - \{L_1\})\theta_1 = (C_2 - \{L_2\})\theta_2$$

Or $L_1\theta_1 = \neg L_2\theta_2$ donc $L_2 = \neg L_1\theta_1\theta_2^{-1}$, on obtient ainsi :

$$C_2 = (C - (C_1 - \{L_1\})\theta_1)\theta_2^{-1} \cup \{\neg L_1\theta_1\theta_2^{-1}\} \tag{5.2}$$

On note le non-déterminisme de cet opérateur, notamment concernant le choix de la clause C_1, et des substitutions θ_1 et θ_2.

Copyright © 2010 Eyrolles.

Dans le cadre de la PLI, S. Muggleton et W. Buntine ont donc eu l'idée d'inverser la résolution classique utilisée en programmation logique. Dans [Mug87], S. Muggleton introduit quatre règles de résolution inverse. La notation $\frac{A}{B}$ s'interprète comme : on peut déduire A de B.

Absorbtion (opérateur V)

$$\frac{q \longleftarrow A \quad p \longleftarrow A, B}{q \longleftarrow A \quad p \longleftarrow q, B}$$

Identification (opérateur V)

$$\frac{p \longleftarrow A, B \quad p \longleftarrow A, q}{q \longleftarrow B \quad p \longleftarrow A, q}$$

Intra construction (opérateur W)

$$\frac{p \longleftarrow A, B \quad p \longleftarrow A, C}{q \longleftarrow B \quad p \longleftarrow A, q \quad q \longleftarrow C}$$

Inter construction (opérateur W)

$$\frac{p \longleftarrow A, B \quad q \longleftarrow A, C}{p \longleftarrow r, B \quad r \longleftarrow A \quad q \longleftarrow r, C}$$

Dans ces règles, les lettres minuscules représentent des atomes et les lettres majuscules des conjonctions d'atomes. Les règles d'absorbtion et d'identification inversent une étape de résolution. La figure 5.4 montre comment l'opérateur V inverse une étape de la résolution.

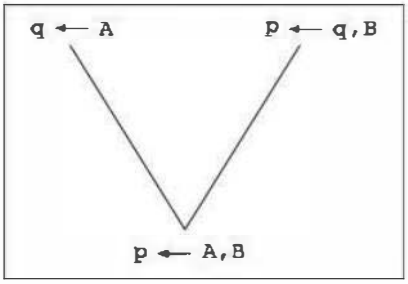


FIG. 5.4: *L'opérateur V d'absorbtion.*

La figure 5.5 illustre par un exemple le comportement de l'opérateur V en montrant plusieurs étapes de résolution inversée. La théorie du domaine est ici constituée des clauses b_1 et b_2 et on dispose d'une observation e_1. L'opérateur V permet, à partir de l'exemple e_1 et de la clause b_2, d'induire la clause c_1. La clause c_1 permet ensuite, avec la clause b_1, d'induire la clause c_2.

Les opérateurs d'intra construction et d'inter construction, appelés aussi opérateurs W (figure 5.6) résultent de la combinaison de deux opérateurs V qui représentent chacun une étape inverse de résolution.

Ces règles d'inférence présentent la particularité d'introduire un nouveau prédicat qui n'apparaissait pas dans les préconditions des règles. Par exemple, l'opérateur W de la figure 5.6 introduit le nouveau prédicat q. On parle alors d'*invention de prédicat*, utilisée par exemple dans le système CIGOL [MB88b]. L'invention de prédicat est bien illustrée par l'exemple de la figure 5.7.

En effet, dans cet exemple, on dispose des deux clauses :

$$min(X, [s(X)||Z]) \longleftarrow min(X, Z)$$
$$min(X, [s(s(X))||Z]) \longleftarrow min(X, Z)$$

Copyright © 2010 Eyrolles.

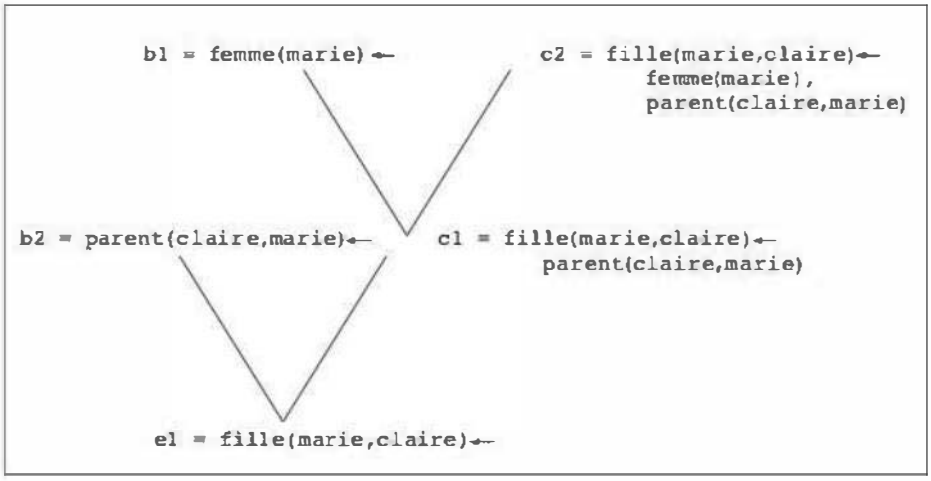


FIG. 5.5: *La résolution inverse.*

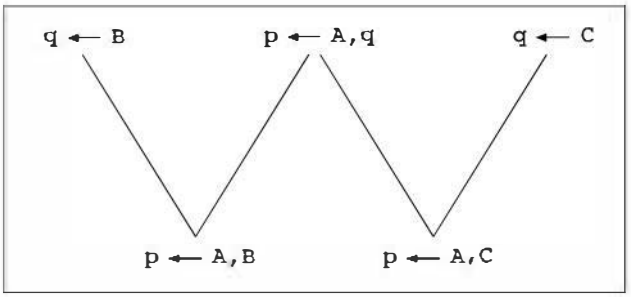


FIG. 5.6: *L'opérateur W d'intra-construction.*

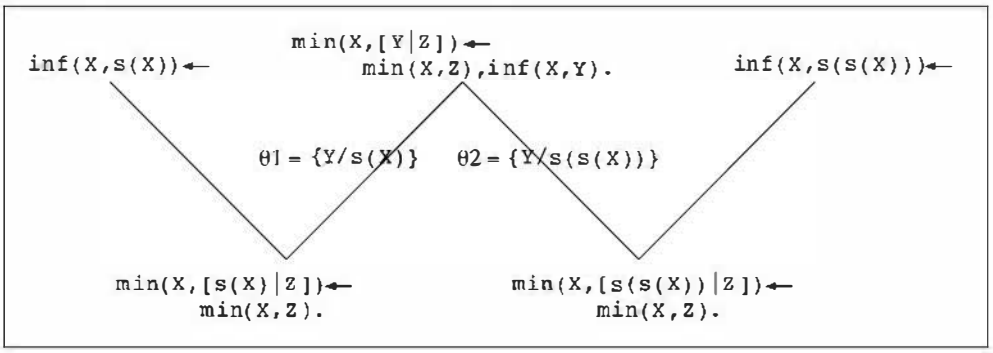


FIG. 5.7: *L'intra construction.*

et l'application de l'opérateur W permet d'induire les trois clauses :

$$inf(X, s(X)) \leftarrow\!-$$
$$min(X, [Y||Z]) \leftarrow\!- min(X, Z), inf(X, Y)$$
$$inf(X, s(s(X))) \leftarrow\!-$$

dans lesquelles on constate que le prédicat $inf/2$ a été inventé.

L'inversion de la résolution est une opération non déterministe. Typiquement, à chaque étape

Copyright © 2010 Eyrolles.

de l'inversion de la résolution, plusieurs généralisations d'une clause peuvent être réalisées, en fonction du choix de la clause avec laquelle elle est résolue, et de la substitution inverse employée. Pour surmonter ce problème de non-déterminisme, Muggleton a proposé, dans un cadre unifiant la plus petite généralisation relative (*rlgg*) et la résolution inverse, que ce soit l'inversion de résolution la plus spécifique qui soit choisie dans le processus de généralisation, en utilisant la substitution inverse la plus spécifique à chaque pas de l'inversion de résolution.

4. L'exploration de l'espace des hypothèses

La notion de couverture d'un exemple par une hypothèse, la relation de subsomption entre hypothèses et les opérateurs permettant de construire les plus petites généralisations (ou spécialisations) autorisent à envisager une approche de l'apprentissage par la détermination de l'espace des versions, comme cela a été montré dans la chapitre 4. Cependant, en raison de la taille de l'espace des versions en programmation logique inductive, les programmes d'induction ne cherchent pas à construire l'espace des versions, même à l'aide des bornes que sont le S-set et le G-set. Ils cherchent « seulement » à trouver une solution dans l'espace des versions. Pour ce faire, ils parcourent l'espace des hypothèses en suivant les directions de généralisation et de spécialisation, en essayant d'élaguer au maximum l'espace des versions et en se guidant par des biais heuristiques tentant d'accélérer la recherche.

Les programmes de PLI s'imposent en général de respecter certaines contraintes connues sous le nom de *sémantique normale* [MD94]. Elles correspondent à une reformulation des conditions de cohérence (complétude et correction) d'une hypothèse avec les exemples positifs et négatifs.

Étant donnés une *théorie du domaine* T, c'est-à-dire un ensemble de clauses qui reflètent la connaissance *a priori* sur le domaine du problème, un ensemble d'exemples E, (avec l'ensemble $E = E^+ \cup E^-$ constitué d'exemples positifs et négatifs), la programmation logique inductive cherche à induire une hypothèse H, composée d'un ensemble de clauses, telle que les quatre conditions suivantes soient respectées :

Définition 5.2 (Sémantique normale)

- *satisfiabilité* a priori : $T \wedge E^- \not\models \square$
- *satisfiabilité* a posteriori : $T \wedge H \wedge E^- \not\models \square$
- *nécessité* a priori : $T \not\models E^+$
- *condition* a posteriori *(complétude)* : $T \wedge H \models E^+$

La condition de satisfiabilité *a priori* permet de s'assurer que les exemples négatifs ne peuvent pas être déduits de la connaissance initiale.

La condition de consistance permet de vérifier qu'à partir de l'hypothèse induite et de la théorie du domaine, on ne peut pas prouver d'exemple négatif.

Il est également nécessaire que les exemples positifs ne puissent pas être déduits de la connaissance T (nécessité *a priori*).

L'hypothèse induite doit quant à elle permettre, avec la connaissance initiale, de prouver les exemples positifs (condition *a posteriori*).

En fait, dans la plupart des systèmes de PLI, la théorie (ou connaissance initiale) T du domaine

Copyright © 2010 Eyrolles.

et l'hypothèse H à induire sont exprimées sous forme de clauses définies. Dans ce cas, on parle de sémantique définie[10]. Seules les quatre conditions suivantes doivent alors être remplies :

Définition 5.3 (Sémantique définie)

- $\forall e \in E^-, e$ *est faux dans* $\mathcal{M}(T)$
- $\forall e \in E^-, e$ *est faux dans* $\mathcal{M}(T \wedge H)$
- $\exists e \in E^+$, *tel que* e *est faux dans* $\mathcal{M}(T)$
- $\forall e \in E^+, e$ *est vrai dans* $\mathcal{M}(T \wedge H)$

Pour simplifier et pour couvrir les cas pratiques, la majorité des systèmes de programmation logique inductive est basée sur le cas particulier de la sémantique définie, dans lequel tous les exemples sont des faits liés.

4.1 Le squelette des algorithmes de PLI

L'algorithme générique de la PLI (algorithme 2) utilise deux fonctions : la fonction **effacer** influence la stratégie de recherche qui peut alors être effectuée soit en largeur d'abord, soit en profondeur d'abord, ou selon une autre stratégie ; la fonction **choisir** détermine les règles d'inférence à appliquer à l'hypothèse H. Ces règles d'inférence peuvent être soit inductives, soit déductives.

Algorithme 2 : Algorithme générique de PLI

```
début
    Initialiser H
    tant que la condition d'arrêt de H n'est pas remplie faire
        effacer un élément h de H
        choisir des règles d'inférence r_1, ..., r_k à appliquer à h
        Appliquer les règles r_1, ..., r_k à h pour obtenir h_1, ..., h_n
        Ajouter h_1, ..., h_n à H
        élaguer H
    fin tant que
fin
```

Définition 5.4 (Règle d'inférence déductive r)

Elle fait correspondre une conjonction de clauses S à une conjonction de clauses G telle que $G \models S$. Dans ce cas, r est une règle de spécialisation.

Définition 5.5 (Règle d'inférence inductive r)

Elle fait correspondre une conjonction de clauses G à une conjonction de clauses S telle que $G \models S$. Dans ce cas, r est une règle de généralisation.

[10] Si une théorie T est constituée de clauses définies, elle possède une fermeture syntaxique (on ne peut rien déduire de plus) appelée le plus petit modèle de Herbrand dans lequel toute formule logique est soit vraie, soit fausse.

Copyright © 2010 Eyrolles.

4.1.1 Stratégies de recherche

La stratégie de recherche est un choix fondamental pour le concepteur d'un système de PLI :

- Les systèmes CIGOL [MB88b], CLINT [De 92], et GOLEM [MF90] sont représentatifs de la classe des *systèmes ascendants*. Ils considèrent les exemples et la théorie du domaine et généralisent itérativement l'hypothèse recherchée en appliquant des règles d'inférence inductives.
- Les systèmes FOIL [Qui90], [QC95] MOBAL [KW92], PROGOL [Mug95], et CLAUDIEN [DB93], [DVD96], [DD97] sont quant à eux représentatifs de la classe des *systèmes descendants*. Ils considèrent d'abord l'hypothèse la plus générale qu'ils cherchent à spécialiser itérativement en appliquant des règles d'inférence déductives.

4.1.2 L'élagage de l'espace de recherche

La fonction **élaguer** détermine quelles hypothèses doivent être enlevées de l'ensemble H. La généralisation et la spécialisation sont à la base de deux cas d'élagage de l'ensemble des hypothèses. Cet élagage est le même que celui pratiqué dans la construction de l'espace des versions par l'algorithme d'élimination des candidats.

- Si $B \wedge H$ n'implique pas logiquement un exemple positif e^+, c'est-à-dire $B \wedge H \not\models e^+$, alors aucune spécialisation de H ne pourra impliquer e^+, et toutes les spécialisations de H peuvent donc être élaguées de l'espace de recherche.
- Si un exemple négatif e^- est couvert, c'est-à-dire $B \wedge H \wedge e^- \models \square$, alors toutes les généralisations de H peuvent être élaguées puisqu'elles sont également inconsistantes avec $B \wedge E$.

L'exemple de la figure 5.8 illustre le premier cas.

Supposons que la base de connaissances contienne les faits :

$$pere(gaston, francois) \longleftarrow$$
$$pere(gaston, laurent) \longleftarrow$$
$$pere(laurent, julien) \longleftarrow$$

et que l'ensemble des exemples positifs contienne l'exemple :

$$oncle(francois, julien) \longleftarrow$$

on peut élaguer toutes les clauses plus spécifiques que la clause $oncle(A, B) \longleftarrow pere(A, B)$ (zone hachurée de la figure 5.8).

Mais l'espace de recherche n'est pas élagué assez efficacement par cette propriété, qui rappelons-le, découle directement de la relation de subsomption entre hypothèses. Il faut donc définir des biais supplémentaires limitant davantage l'espace de recherche.

4.2 Les biais de recherche dans l'espace d'hypothèses

L'espace de recherche peut être très vaste, voire infini. L'implantation effective d'un système de PLI nécessite donc l'utilisation de biais déclaratifs. Les biais imposent des contraintes ou des restrictions sur les hypothèses que l'algorithme peut considérer et définissent donc l'espace de recherche. Le rôle du biais est double :

- réduire la taille de l'espace de recherche pour diminuer le temps de réponse du système ;
- garantir une certaine qualité de l'apprentissage en interdisant au système de considérer certaines hypothèses inutiles.

On distingue deux classes de biais déclaratifs : les biais syntaxiques, aussi appelés biais de langage, et les biais sémantiques.

Copyright © 2010 Eyrolles.

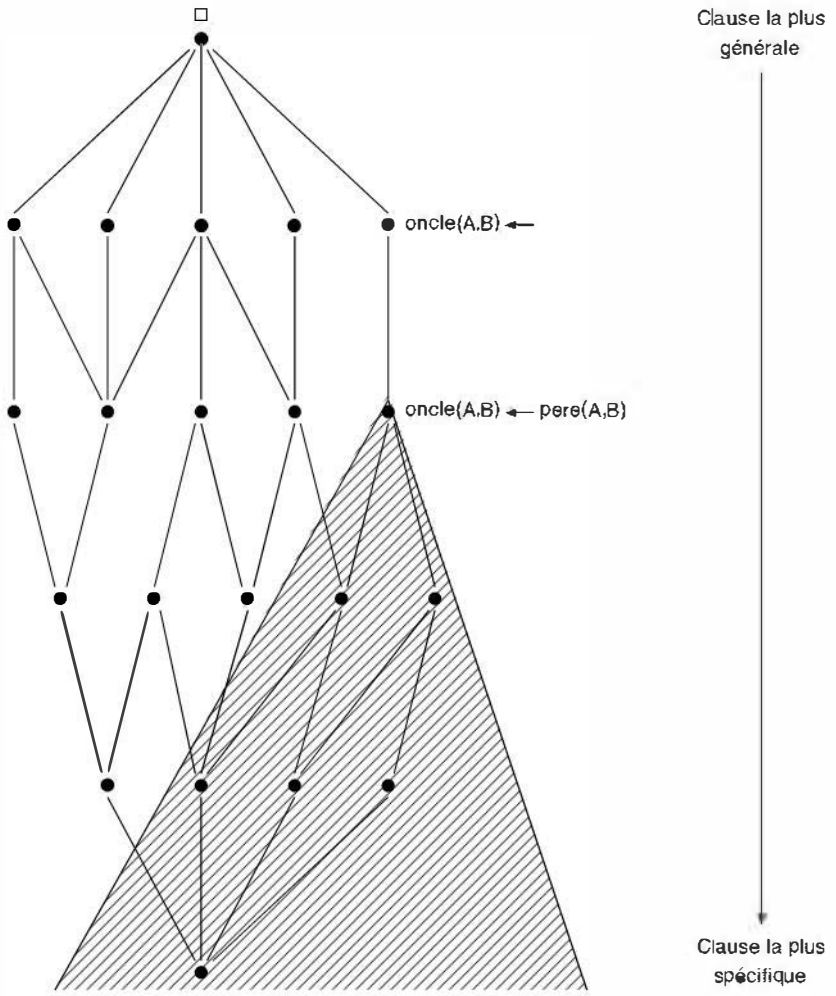


FIG. 5.8: *L'élagage de l'espace de recherche par la relation de subsomption.*

4.2.1 Les biais syntaxiques

Un biais syntaxique permet de définir l'ensemble des hypothèses envisageables en spécifiant explicitement leur syntaxe. Le nombre des hypothèses à considérer peut en particulier être réduit par les méthodes suivantes :

- limiter le nombre de littéraux des clauses induites ;
- limiter le nombre de variables apparaissant dans une clause ;
- limiter la profondeur de la récursion dans les termes ;
- n'autoriser que les clauses *range-restricted*, c'est-à-dire dont l'ensemble des variables de la tête est inclus dans l'ensemble des variables du corps.

4.2.2 Les biais sémantiques

Si les biais syntaxiques restreignent l'espace de recherche en imposant la syntaxe des hypothèses, les biais sémantiques imposent des restrictions sur le sens des hypothèses. La définition de *types* et de *modes* est utilisée depuis de nombreuses années en programmation logique afin

Copyright © 2010 Eyrolles.

de permettre une meilleure analyse statique des programmes et ainsi d'augmenter leur efficacité. Depuis leur introduction dans le système MIS [Sha83], il est devenu courant d'utiliser également les modes et les types en PLI. Ils constituent alors une façon de définir des biais sémantiques. En effet, si un littéral contient une variable L de type t, toutes les autres occurences de cette variable L dans la même clause devront également être de type t. Ceci permet d'élaguer de l'espace de recherche des clauses incorrectement typées. De même, la déclaration de modes permet de restreindre la taille de l'espace de recherche. Par exemple, l'utilisation du prédicat $concat/3$ associée à la déclaration de mode suivante :

$$concat(+L1, +L2, -L3)$$

où + signifie argument en entrée (devant être instanciés), et − argument en sortie, permet de rejeter de l'espace de recherche toutes les clauses utilisant le prédicat *concat* sur des variables ne respectant pas ce mode, comme par exemple :

$$concat(L1, L2, [a, b, c])$$

Voici quelques façons d'utiliser les modes sur les variables des prédicats :

- Toutes les variables en entrée dans la tête de la clause doivent se retrouver dans le corps de clause.
- Les variables en sortie dans la tête doivent être présentes dans le corps.
- On peut aussi forcer toutes les variables en sortie dans le corps soit à servir de résultat intermédiaire (et donc d'être en entrée dans un autre prédicat), soit à servir à définir la sortie du prédicat de la tête (et donc d'apparaître en sortie dans la tête).
- Il est également envisageable d'interdire qu'une même variable apparaisse plusieurs fois en sortie de différents prédicats dans le corps de la clause.

Le système PROGOL [Mug95, Rob97] permet par exemple de déclarer des modes sur les variables des littéraux afin de restreindre le nombre potentiel de clauses dans l'espace de recherche. De plus, le système distingue les prédicats pouvant apparaître en tête de clause de ceux admis dans le corps. Les littéraux qui peuvent apparaître en tête sont déclarés avec la directive *modeh*. Les littéraux autorisés dans les corps de clauses sont eux déclarés grâce à la directive *modeb*. Par exemple, la déclaration

$$modeh(1, plus(+int, +int, -int))$$

spécifie que la tête des clauses de l'espace de recherche sera constituée du prédicat $plus/3$ avec trois arguments de type entier, les deux premiers arguments étant en mode entrée et le dernier en mode sortie.

5. Deux exemples de systèmes de PLI

5.1 Un système empirique descendant : FOIL

Le système FOIL[11] [Qui90] a été développé par J.R. Quinlan et a ensuite inspiré d'autres systèmes parmi lesquels FOCL [PK92], FOIDL [MC95], MFOIL [Dze93], ICN et MULT_ICN [MV95]. FOIL cherche à induire un programme logique qui couvre tous les exemples positifs

[11] Le système FOIL est accessible par ftp à l'adresse ftp.cs.su.oz.au/pub/foil6.sh

Copyright © 2010 Eyrolles.

du concept à apprendre, et aucun exemple négatif. Chaque clause apprise est construite par spécialisation successive de la clause la plus générale par ajout d'un nouveau littéral. Le littéral ajouté est choisi en calculant un *gain d'information* de façon similaire à la construction d'un arbre de décision (voir le chapitre 13). L'utilisation de FOIL impose une représentation en extension de la théorie du domaine. FOIL permet de définir des biais syntaxiques (limitation du nombre de variables apparaissant dans chaque clause, taux minimal de couverture des clauses, etc.). Il est également possible de préciser les modes des arguments des prédicats.

L'algorithme

Algorithme 3 : FOIL

début
  $P \longleftarrow \emptyset$
  $Pos \longleftarrow$ exemples positifs
  tant que *Pos est non vide* **faire**
    $Neg \longleftarrow$ exemples négatifs
    $C = q(X_1, \ldots, X_n)$
    tant que *Neg est non vide* **faire**
      Ajouter le littéral de meilleur gain au corps de C
      Retirer de Neg les exemples négatifs non couverts par C
    fin tant que
    Ajouter la clause apprise C à P
    Retirer de Pos les exemples couverts par C
  fin tant que
  Retourner le programme appris P
fin

L'algorithme 3 est la base du système FOIL. La boucle la plus externe permet de construire des clauses tant que tous les exemples ne sont pas couverts. La boucle interne construit une clause en ajoutant un à un des littéraux qui ont le gain le plus élevé. Pour la construction de chaque clause à l'étape i, FOIL gère deux ensembles de tuples T_i^+ et T_i^-. Chaque élément de ces ensembles est une instance liée de la clause en construction correspondant à un exemple couvert. Les tuples de T_i^+ correspondent à des exemples positifs, et ceux de T_i^- à des exemples négatifs. À chaque étape, les ensembles T_i^+ et T_i^- sont calculés à partir des ensembles T_{i-1}^+ et T_{i-1}^- de l'étape précédente. La fonction de gain utilisée pour le choix du littéral L à ajouter lors du passage à une nouvelle étape est calculée à partir du nombre d'éléments des différents ensembles de tuples :

$$gain(L) = n_i^+ \left(log_2 \frac{\|T_i^+\|}{\|T_i^+\| + \|T_i^-\|} - log_2 \frac{\|T_{i+1}^+\|}{\|T_{i+1}^+\| + \|T_{i+1}^-\|} \right)$$

où n_i^+ est le nombre d'exemples positifs couverts par la clause en construction, et les T_{i+1} sont les ensembles de tuples considérés lorsqu'on a ajouté le littéral L.

L'un des problèmes majeurs de FOIL intervient lors de la construction d'une clause. En effet, à chaque ajout d'un littéral dans la clause, il se peut que le nombre de littéraux candidats soit très grand. Or pour chacun d'entre eux, FOIL doit calculer les ensembles de tuples positifs et négatifs pour être en mesure d'évaluer le gain. Ceci peut être préjudiciable à l'efficacité du système.

Du fait de sa méthode d'apprentissage, FOIL permet de traiter des données bruitées. En effet, il peut considérer qu'une certaine partie des exemples est bruitée, et arrêter la construction de

Copyright © 2010 Eyrolles.

la clause lorsqu'un certain pourcentage d'exemples est couvert.

— EXEMPLE **Une illustration du fonctionnement** ———————————————

Nous illustrons l'utilisation du système FOIL par l'apprentissage d'une définition du concept *oncle*. On dispose de la connaissance suivante sur les liens de parenté au sein d'une famille.

```
/* valeurs possibles pour les arguments des predicats */
P: alfred,michel,remi,franck,charles,paul.

oncle(P,P)
/* exemples positifs pour le predicat oncle */
remi,paul            remi,franck          michel,charles
;
/* exemples negatifs pour le predicat oncle */
alfred,alfred        alfred,michel        alfred,remi
alfred,franck        alfred,charles       alfred,paul
michel,alfred        michel,michel        michel,remi
michel,franck        michel,paul          remi,alfred
remi,michel          remi,charles         remi,remi
franck,paul          franck,alfred        franck,michel
franck,remi          franck,franck        franck,charles
charles,paul         charles,alfred       charles,michel
charles,remi         charles,franck       charles,charles
paul,paul            paul,alfred          paul,michel
paul,remi            paul,franck          paul,charles
.

*pere(P,P)
/* exemples positifs pour le predicat pere */
michel,paul          alfred,michel        alfred,remi
michel,franck        remi,charles
;
/* exemples negatifs pour le predicat pere */
alfred,alfred        alfred,franck        alfred,charles
alfred,paul          michel,alfred        michel,michel
michel,remi          michel,charles       remi,alfred
remi,michel          remi,remi            remi,franck
remi,paul            franck,alfred        franck,michel
franck,remi          franck,franck        franck,charles
franck,paul          charles,alfred       charles,michel
charles,remi         charles,franck       charles,charles
charles,paul         paul,alfred          paul,michel
paul,remi            paul,franck          paul,charles
paul,paul
```

FOIL apprend le prédicat `oncle/2` défini par la clause :

```
oncle(A,B) :- pere(C,A), pere(D,B), pere(C,D), A<>D.
```

Copyright © 2010 Eyrolles.

5.2 Un système empirique ascendant : PROGOL

PROGOL [12] est un système de PLI basé sur l'implication inverse inventée par Muggleton, [Mug95], son concepteur. PROGOL opère par généralisation ascendante en partant de chaque clause la plus spécifique couvrant chaque exemple considéré. La clause finale est calculée grâce à un algorithme A^* intégrant une mesure de compression que PROGOL cherche à maximiser. PROGOL implémente une procédure en quatre étapes. **Sélection d'un exemple** à généraliser ; **construction de la clause la plus spécifique** impliquant l'exemple ; **recherche** d'une clause généralisant la clause de départ ; **élimination** des exemples redondants après généralisation (voir l'algorithme 4). Cet algorithme n'est pas complet, il ne découvre pas toutes les généralisations possibles.

Des déclarations de modes et de types permettent de définir des biais sémantiques. Il est nécessaire de spécifier le littéral à employer en tête de clause (en utilisant la directive *modeh*), et les littéraux qu'il est possible d'utiliser dans le corps des clauses (en utilisant la directive *modeb*).

PROGOL permet de définir la théorie du domaine en intension sous la forme de clauses Prolog. Ceci est un apport significatif par rapport à FOIL avec lequel il était nécessaire de définir la connaissance initiale en extension. Les exemples positifs sont exprimés par des clauses définies, et les exemples négatifs sont des négations de faits instanciés. [Rob97] expose les différents paramètres du système.

Algorithme 4 : PROGOL

début
  tant que *il reste des exemples positifs* **faire**
    pour chaque *exemple positif e* **faire**
      Construire la clause c_1 la plus spécifique qui implique l'exemple e
      Trouver une clause c_2 plus générale que c_1 (au sens de la θ-subsomption) telle que la mesure de compression soit maximale
      Retirer tous les exemples couverts par la clause c_2
    fin
  fin tant que
fin

EXEMPLE **Une illustration du fonctionnement**

Considérons par exemple, l'apprentissage du tri rapide par PROGOL. Etant donné le programme suivant :

```
% Declaration des modes pour le litteral en tete de clause
:- modeh(1,qsort([+int|+ilist],-ilist))?

% Declaration des modes pour les litteraux en corps de clause
:- modeb(1,qsort(+ilist,-ilist))?
:- modeb(1,part(+int,+ilist,-ilist,-ilist))?
:- modeb(1,append(+ilist,[+int|+ilist],-ilist))?

% Types
ilist([]).
ilist([Head|Tail]) :- int(Head), ilist(Tail).
```

[12] Une version du système PROGOL est accessible par ftp à l'adresse suivante : http ://www.comlab.ox.ac.uk/activities/machinelearning/PProgol/ppman.html

Copyright © 2010 Eyrolles.

```
% Theorie du domaine definie en intension
part(X,[],[],[]).
part(X,[X|Tail],List1,List2) :- part(X,Tail,List1,List2).
part(X,[Head|Tail],[Head|Tail1],List2) :-
            Head < X, part(X,Tail,Tail1,List2).
part(X,[Head|Tail],List1,[Head|Tail2]) :-
            Head > X, part(X,Tail,List1,Tail2).

append([],List,List).
append([Head|Tail],List1,[Head|List2]) :- append(Tail,List1,List2).
```

Exemples positifs	Exemples négatifs
`qsort([],[]).`	`:- qsort([0,2,1],[0,2,1]).`
`qsort([3,2,1],[1,2,3]).`	`:- qsort([0,2,1],[0,1]).`
`qsort([X],[X]).`	`:- qsort([1,0,2],[2,0,1]).`
`qsort([X,Y],[X,Y]) :- X < Y.`	`:- qsort([1,0,2],[2,1,0]).`
`qsort([Y,X],[X,Y]) :- X < Y.`	`:- qsort([1,2,0],[1,0,2]).`
`qsort([X,Y,Z],[X,Y,Z]) :- X<Y, Y<Z.`	`:- qsort([0,2,1],[2,1,0]).`
`qsort([X,Z,Y],[X,Y,Z]) :- X<Y, Y<Z.`	`:- qsort([2,1,0],[2,1,0]).`
`qsort([X,Z,Y],[X,Y,Z]) :- X<Y, Y<Z.`	`:- qsort([2,0,1],[2,1,0]).`
`qsort([Y,X,Z],[X,Y,Z]) :- X<Y, Y<Z.`	`:- qsort([2,1],[1]).`
`qsort([Y,Z,X],[X,Y,Z]) :- X<Y, Y<Z.`	`:- qsort([1],[2]).`
`qsort([Z,X,Y],[X,Y,Z]) :- X<Y, Y<Z.`	`:- qsort([0,1,2],[1,0,2]).`
	`:- qsort([0,1],[1,0,1]).`

Le système PROGOL induit une définition du prédicat *qsort*/2 sous la forme des clauses suivantes :

```
qsort([],[]).
qsort([A|B],C) :- qsort(B,D), part(A,D,E,F), append(E,[A|F],C).
```

Sur cet exemple, la stratégie de parcours de l'espace de recherche par PROGOL est bien mise en évidence. La clause la plus spécifique est d'abord calculée, puis PROGOL part de la clause la plus générale et la spécialise en ajoutant des littéraux. Pour chaque clause examinée, PROGOL compte le nombre d'exemples positifs et négatifs couverts et effectue une mesure de compression qui lui permet de sélectionner la clause à retenir.

6. La probabilisation de la PLI

Ce chapitre est centré sur les représentations logiques et sur l'apprentissage d'hypothèses exprimées en logique à partir d'exemples. L'une des limites de la logique classique est son incapacité à représenter l'incertitude et à permettre le raisonnement incertain. Le calcul des probabilités offre de son côté un cadre rigoureux pour le raisonnement incertain, mais il est limité aux concepts propositionnels. Depuis plusieurs années, il existe donc des efforts de recherche pour essayer de marier au mieux les possibilités des deux approches, et dépasser leurs limites.

L'une des premières questions est de définir la notion de « couverture » d'un exemple par une hypothèse. Dans le cadre de la logique, nous avons vu que cette notion dépend d'une procédure de preuve qui renvoie un booléen vrai ou faux. Dans le cadre du calcul probabiliste, la notion

Copyright © 2010 Eyrolles.

de couverture devient probabiliste, renvoyant une valeur de probabilité. On peut harmoniser les deux notions en posant que les valeurs `vrai` et `faux` correspondent respectivement à des valeurs de probabilité 1 et 0.

Dans le cadre probabiliste, les hypothèses peuvent prendre plusieurs formes selon la représentation utilisée : *réseau bayésien* (voir chapitre 11), *modèle de Markov* (chapitre 12), *grammaire probabiliste* (chapitre 7), etc.

Une fois le cadre fixé, il est possible de calculer, selon le formalisme choisi, la probabilité qu'un événement, un exemple, soit issu de l'hypothèse considérée. Comme nous l'avons vu au chapitre 2, l'apprentissage dans ces modèles consiste fondamentalement à identifier l'hypothèse la plus probable en tenant compte à la fois de la distribution de probabilité *a priori* sur l'espace des hypothèses et de la vraisemblance des données selon chaque hypothèse.

Afin de construire une logique du premier ordre probabilisée, il faut :

- savoir annoter les clauses avec une valeur de probabilité ;
- rendre la notion de couverture probabiliste.

Plusieurs choix sont possibles pour cela, ce qui entraîne une certaine variété de cadres logiques probabilistes. Les approches proposées peuvent essentiellement s'organiser autour de deux points de vue :

- Une distribution de probabilité est définie sur les mondes possibles ou interprétations. C'est le cas des *programmes logiques bayésiens* et des *réseaux de Markov logiques*.
- Une distribution de probabilité est définie sur les preuves. C'est le cas des *programmes logiques stochastiques*, de PRISM et de ICL.

Comme dans les approches probabilistes non logiques, l'apprentissage se décompose en l'apprentissage de la structure, ce qui correspond aux clauses, et en l'apprentissage des paramètres.

Le lecteur trouvera des descriptions détaillées de ces approches dans [RFKT08, Rae08, GT07].

7. Les domaines d'application de la PLI

Découpe en éléments finis

Le découpage en éléments finis (voir figure 5.9) est une technique très utilisée par les ingénieurs qui cherchent à analyser des contraintes sur des structures physiques [DM92]. En effet, les contraintes qui s'exercent sur la structure peuvent être exprimées par des équations différentielles, mais il n'est généralement pas possible de résoudre de telles équations à l'aide d'un ordinateur en un temps acceptable. Pour résoudre ce problème, les ingénieurs découpent la structure en un nombre fini d'éléments, et utilisent des approximations linéaires pour calculer les contraintes sur chaque partie élémentaire de la structure.

Dans la pratique, il n'existe pas de méthode générale permettant de réaliser un maillage acceptable pour une pièce. GOLEM a été utilisé dans ce contexte en vue d'induire un modèle de la méthode utilisée par les experts pour découper les structures. Il faut noter que les experts ne sont pas capables de modéliser leur méthode de découpage et que GOLEM est ici utilisé pour extraire cette expertise.

La densité du maillage dépend des propriétés géométriques de l'objet, des forces et des contraintes qui agissent dessus, et des relations entre ses différents composants. Pour traiter ce problème avec la PLI, une structure est représentée par un ensemble d'arêtes, des propriétés sur les arêtes, et des relations entre les arêtes. Chaque objet est ainsi décrit en spécifiant :

Copyright © 2010 Eyrolles.

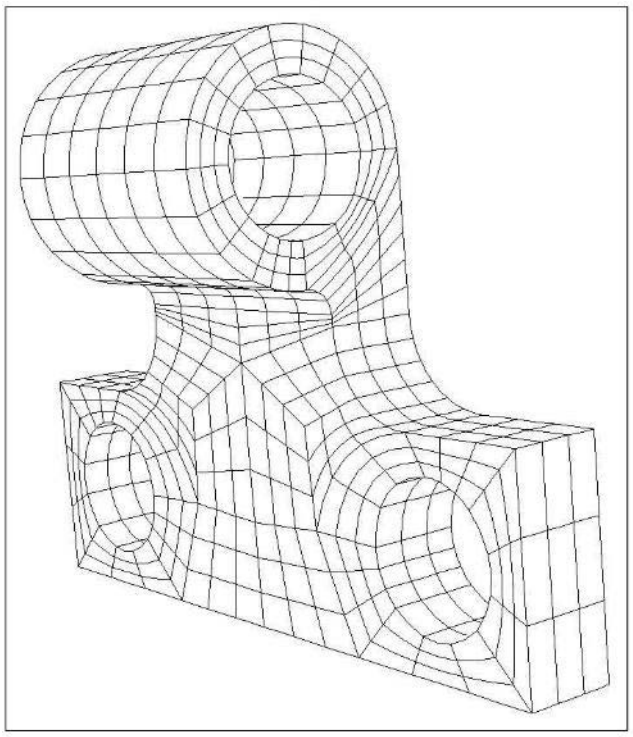


FIG. 5.9: *Une pièce découpée en parties élémentaires.*

- Le type de ses arêtes : longue importante, importante, courte importante, pas importante, circuit, demi-circuit...
- Les différentes conditions existants aux zones communes à plusieurs arêtes : libre, fixée sur un côté, fixée sur deux côtés, complètement fixée.
- Ls contraintes de charge qui s'exercent sur chaque arête : pas de charge, chargée sur un côté, chargée sur les deux côtés, chargée de manière continue.
- La représentation géométrique de l'objet. Les auteurs s'attachent à la notion d'arêtes voisines, d'arêtes opposées et d'arêtes identiques. On spécifie que les arêtes sont opposées si elles jouent un rôle symétrique dans la pièce. Certaines arêtes sont non seulement opposées, mais elles ont de plus la même forme et la même longueur. On précise alors qu'elles sont identiques.

Pour chaque arête a_i d'une structure s, on dispose d'un fait $mesh(a_i, n)$ qui signifie que cette arête est découpée en n éléments finis.

Le prédicat à apprendre est donc $mesh(Arete, N)$, où $Arete$ est une arête et N est le nombre de segments qui constitueront cette arête. Avec des descriptions de trois objets, GOLEM induit cinquante-six règles différentes. Ces règles ne sont pas toutes pertinentes, mais GOLEM induit des règles de la forme :

```
mesh(A, 1) :-
        not_important(A),
        not_loaded(A).
```

qui signifie qu'une arête A est découpée en un élément si elle n'est pas importante et n'a pas de contrainte.

Copyright © 2010 Eyrolles.

Une autre règle induite est :

```
mesh(A, B) :-
        cont_loaded(A),
        same(A,C),
        cont_loaded(C),
        mesh(C, B).
```

qui met en évidence que si deux arêtes sont identiques et qu'elles sont contraintes de manière continue, leur découpage se fait également en un nombre identique d'éléments.

Il est intéressant de noter que selon les experts du domaine, les règles fournies par le système mettent en évidence des dépendances qui leur étaient jusqu'alors inconnues [BM95b].

Une liste d'applications

L'ILP est utilisée dans de nombreux domaines pour l'apprentissage de concepts relationnels (voir par exemple la page http ://www.doc.ic.ac.uk/ shm/applications.html). En voici une liste partielle :

Sciences de la vie : structure 3-D des protéines, découverte de neuropeptides, analyse de tests de laboratoire, diagnostic précoce de rhumatisme, mutagénèse.

Environnement : développement des algues, données de qualité de l'eau.

Langage naturel : règles d'accord des verbes au passé en anglais, apprentissage d'analyseurs syntaxiques, définition des groupes nominaux, catégorisation de texte, analyse de groupes syntaxiques, prédiction des césures, etc.

Autres domaines : bases de données géographiques, métallurgie, analyse des accidents de la circulation, marketing et commerce.

Pour ne prendre que l'exemple du langage naturel, les techniques d'apprentissage semblent donc être tout à fait adaptées pour inférer un analyseur morphologique, syntaxique ou sémantique d'un langage. De plus, Prolog est un langage considéré comme bien adapté au traitement du langage naturel, ce qui fait également de la programmation logique inductive une méthode appropriée pour cette tâche.

Les premiers travaux dans le domaine ont été réalisés par J. Zelle et R. Mooney [ZM93] [Zel95] [Moo96]. Ils se poursuivent en particulier dans les domaines cités ci-dessus.

8. Les chantiers de la PLI

8.1 Une transition de phase rédhibitoire ?

L'attrait majeur de la PLI est le recours à un langage des hypothèses expressif (plus que la plupart des autres langages de généralisation) et commode, puisqu'il permet la représentation aussi bien des exemples que des hypothèses ou que de la théorie du domaine. La logique des prédicats est de plus un domaine bien connu grâce aux travaux des logiciens. Comme nous l'avons exposé, utiliser la PLI nécessite cependant de choisir avec soin les relations d'ordre et les opérateurs d'exploration. Imposer des limites pour obtenir des ordres bien fondés et des systèmes d'induction contrôlés introduit des difficultés, certes à prendre au sérieux, mais semble-t-il non rédhibitoires. Cependant, des travaux récents mettent en cause la possiblité même de l'induction

Copyright © 2010 Eyrolles.

supervisée lorsque le langage d'hypothèses est trop expressif. Nous allons examiner ce nouveau problème.

Ces travaux sont fondés sur des considérations complètement indépendantes de celles que nous avons développées dans le chapitre 2 qui dérivent de l'analyse statistique de l'apprentissage. À l'origine, il s'agissait de d'examiner la complexité moyenne de la résolution de problèmes de satisfaction de contraintes, plutôt que d'examiner leur complexité dans le pire cas. Ainsi, le problème 3-SAT (problème de satisfiabilité) consistant à chercher une conjonction de clauses de 3 termes (e.g. $(x_1 \vee \bar{x_5} \vee x_{11}) \wedge (x3 \vee x_5 \vee x_8)$) vraie pour l'assignation des variables à des valeurs de vérité, est connu pour être NP-complet. Cependant, l'examen plus fin de la difficulté des problèmes montre une transition brutale, en fonction de certains paramètres, entre une région où il est quasiment toujours possible de trouver un telle 3-DNF et une région où c'est quasiment toujours impossible, avec une région intermédiaire extrêmement étroite correspondant à un pic de complexité de l'algorithme de recherche. Il se trouve qu'il est possible d'établir une relation entre ces problèmes de satisfiabilité et le problème de la recherche d'hypothèses de risque empirique nul, c'est-à-dire couvrant tous les exemples positifs et aucun exemple négatif.

En effet, l'apprentissage supervisé implique une exploration de l'espace des hypothèses guidée par la valeur du risque empirique (éventuellement régularisé). La plupart des stratégies de contrôle de cette exploration utilisent une technique de gradient afin de modifier la ou les hypothèse(s) courante(s) dans la direction du gradient le plus favorable. En fin de compte, il faut donc déterminer si chaque exemple (positif ou négatif) est couvert par chaque hypothèse candidate examinée. Cela ressemble à un problème de satisfiabilité.

En particulier, il est intéressant d'étudier le taux de couverture des hypothèses, c'est-à-dire la proportion d'exemples tirés au hasard qu'elles couvrent. Ainsi, si le taux est de 100 % (ou 1), c'est que l'hypothèse couvre tous les exemples. Idéalement, ce taux de couverture peut prendre toutes les valeurs entre 0 et 1, de manière graduelle. Cela permet d'envisager une exploration guidée par le gradient du risque empirique si ce taux de couverture présente une certaine régularité dans l'espace des hypothèses (i.e. deux hypothèses proches au sens des opérateurs permettant de passer de l'une à l'autre ont des taux de couverture proches).

C'est ce paysage du taux de couverture en fonction des hypothèses qu'ont étudié Giordana et Saitta [GS00] dans le cas d'espaces d'hypothèses composés de clauses de Horn. Par exemple, en faisant varier le nombre de variables, le nombre de littéraux et le nombre d'atomes clos (et en répétant ces expériences un grand nombre de fois), ils ont obtenus des courbes telles que celle de la figure 5.10 qui rappellent des phénomènes de *transition de phase* en physique. De manière surprenante, le taux de couverture mesuré sur les hypothèses y passe brutalement de 0 à 1 sans presque de transition lorsque l'on va progressivement vers des hypothèses plus générales. Ces courbes sont potentiellement de grande portée. Pourquoi ?

Concrètement, imaginons un algorithme d'apprentissage ascendant, opérant par généralisation progressive et prudente. Lorsqu'une hypothèse courante ne couvre pas tous les exemples positifs connus, le système tente de la généraliser un peu afin d'en couvrir davantage, tout en essayant de ne pas couvrir les exemples négatifs. Le nombre de généralisations possibles est en général très grand et le système se guide en choisissant celles qui conduisent à des hypothèses de meilleur risque empirique. Or que se passe-t-il selon les courbes de taux de couverture observées ? Lorsque l'on prend une hypothèse trop spécifique, elle ne couvre pratiquement aucun exemple, positif ou négatif, son taux de couverture est nul. Le problème, c'est que quand on la généralise, son taux de couverture reste nul. Le risque empirique reste donc constant et il est impossible de se guider dans l'espace de recherche. Ce n'est que lorsque les hypothèses considérées ont été suffisamment généralisées qu'elles se trouvent dans la région de la transition de phase (la « falaise ») que leur taux de couverture varie et que l'on peut enfin comparer les hypothèses entre elles. Avant, il est

Copyright © 2010 Eyrolles.

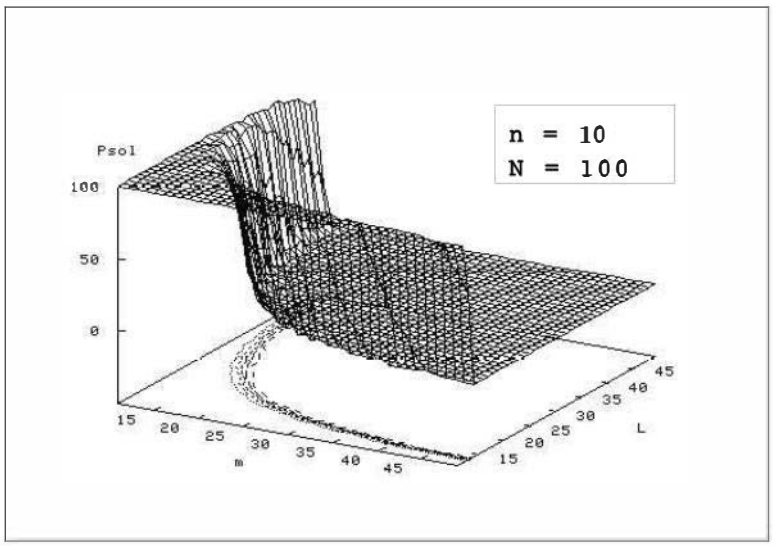


FIG. 5.10: *Une courbe tridimensionnelle de la probabilité de couverture d'un exemple tiré aléatoirement par une hypothèse h en fonction de* m, *le nombre de littéraux dans h et de* L, *le nombre d'atomes clos (ou constantes).* m *et* N *correspondent respectivement au nombre de variables (10 dans cette expérience) et au nombre de tuples dans les relations (100 ici). Les hypothèses les plus générales se situent à gauche sur cette figure, correspondant à des hypothèses peu contraintes (*m *et* L *petits).*

impossible au système de se guider. La recherche devient donc aléatoire. Le même phénomène se produit pour un système descendant qui spécialise progressivement des hypothèses intialement trop générales.

L'induction supervisée par généralisation ou spécialisation progressive semble donc selon cette analyse très difficile, sauf pour des problèmes « jouets ». C'est d'ailleurs ce que semblent confirmer empiriquement des expériences complémentaires de Giordana et Saitta (voir [GSC10]). Il y a là un problème fondamental qui relève encore de la recherche théorique et expérimentale. Il touche en tout cas les systèmes utilisant la logique des prédicats comme langage de description des hypothèses. On ne sait pas encore dans quelle mesure il se produit sur d'autres représentations des connaissances, par exemple en inférence grammaticale (voir le chapitre suivant).

8.2 Quelques problèmes ouverts en PLI

L'extension de la PLI repose en particulier sur la résolution des problèmes suivants :

- la prise en compte de données numériques ;
- l'utilisation d'autres logiques (confirmatoires, modales ...) ;
- l'apprentissage effectif de règles récursives ;
- la prise en compte de l'incertain.

Ces limitations motivent une activité de recherche continue dans ce domaine, dont rend compte en particulier la conférence internationale annuelle ILP (Inductive Logic Programming).

Copyright © 2010 Eyrolles.

Notes historiques et sources bibliographiques

Quoique les fondations de la logique remontent (facilement) à Aristoste, c'est seulement durant le $XIX^{ème}$ et le $XX^{ème}$ siècles que des théoriciens tels que Boole et Frege l'ont formalisée rigoureusement dans un cadre mathématique. Au $XX^{ème}$ siècle, une école de philosophes connus sous le nom de positivistes logiques a promu l'idée que la logique était le socle ultime non seulement des mathématiques, mais de toute la science. Selon ce point de vue, toute affirmation mathématique peut être exprimée dans le cadre de la logique des prédicats, et tous les raisonnements scientifiques valides sont basés sur des dérivations logiques à partir d'un ensemble d'axiomes. Le positivisme logique trouva un support puissant dans les travaux de Gödel dans les années 1930 qui ont montré qu'un ensemble réduit de règles de dérivation était suffisant (complet) pour dériver toutes les conséquences de formules en logique des prédicats. Plus tard, Robinson, en 1965, démontra que l'inférence déductive en logique des prédicats pouvait être réalisée à l'aide d'une unique règle d'inférence, la résolution. Pour appliquer le principe de résolution, il faut mettre les formules logiques sous la forme de clauses logiques. La découverte de Robinson a été d'une importance fondamentale pour l'automatisation de de la déduction. Colmerauer et Kowalski furent les pionners au début des années 1970 du développement de Prolog, un langage de programmation logique. Dans ce langage, toutes les formules logiques sont décrites sous la forme de clauses de Horn.

C'est Plotkin qui dans sa thèse en 1971 [Plo71a] a posé les fondations de ce qui allait devenir le champ de la programmation logique inductive. Plotkin ne se limitait pas à la logique des clauses de Horn, ce qui n'est pas surprenant puisque la programmation logique et le langage Prolog n'existaient pas encore. Ses principales contributions concernent l'introduction de la subsomption relative comme relation de généralité entre clauses, et un algorithme de calcul du moindre généralisé d'un ensemble de clauses sans théorie du domaine, ce généralisé étant la borne inférieure (unique en l'absence de théorie du domaine) de l'espace des généralisations d'un ensemble de clauses. Plotkin a démontré que malheureusement, il n'existe pas en général de moindre généralisé fini de deux clauses en présence d'une théorie du domaine. Cela restreignait évidemment la portée de son algorithme et c'est pourquoi Shapiro [Sha83] étudia une autre approche, portant sur l'induction descendante de clauses de Horn, du général au spécifique, au lieu d'ascendante comme pour Plotkin. De plus, Shapiro étudia le débogage algorithmique de programmes Prolog par identification automatique de la clause défectueuse. Ces systèmes d'induction étaient cependant très inefficaces et limités à de tout petits problèmes.

Sammut et Banerji, (voir [Sam93]) ont développé un système appelé MARVIN qui réalise un apprentissage incrémental au sens où le système considère les exemples à la suite des autres et apprend, ce faisant, des concepts qui sont utilisés pour traiter les exemples suivants. La connaissance préalable ou théorie du domaine (*background knowledge*) s'accroît donc au cours du temps. L'apprentissage se fait par généralisation progressive en utilisant une substitution qui remplace les types de variables par leur sur-type dans le langage de description des concepts. Il s'agit du seul opérateur de généralisation utilisé. Le système a été testé sur un ensemble de problèmes variés. Plus tard, Muggleton et Buntine démontrèrent que l'opérateur de généralisation de Banerji était un cas particulier de l'inversion d'un pas dans la preuve par résolution. Un ensemble de contraintes plus générales a été fourni et les deux opérateurs en 'V' ont été proposés. L'autre paire d'opérateurs en 'W', également basés sur l'inversion de la résolution, est également issue de leurs travaux. D'autres travaux ont aussi porté sur l'invention de prédicats, mais pour le moment ils sont restés confinés à la logique des propositions et non à la logique des prédicats.

Quinlan, [Qui90], a proposé le système FOIL qui induit des clauses de Horn de manière très

Copyright © 2010 Eyrolles.

efficace. La méthode utilise une heuristique de recherche descendante, par spécialisation progressive, guidée par un critère de gain d'information (voir le chapitre 13). L'un des inconvénients de la méthode est qu'elle est gloutonne et donc susceptible de tomber dans des optima locaux mauvais, voire d'errer sans direction comme le montre le phénomène de transition de phase (évoqué dans la section 7).

Ce chapitre est en partie extrait des notes de cours de Marc Bernard, de l'université de Saint-Etienne [Ber00], qui nous a aimablement autorisé à les utiliser.

Résumé

- La programmation logique inductive (PLI) est essentiellement tournée vers l'apprentissage supervisé de concepts ou de théories (exprimés sous forme de programmes logiques). L'apprentissage y est guidé par une relation de généralité dans l'espace des hypothèses en cherchant des hypothèses cohérentes avec les exemples d'apprentissage.
- Contrairement à la plupart des techniques d'apprentissage qui sont concernées par des connaissances en logique attribut-valeur, la PLI s'occupe de l'apprentissage de règles exprimées en logique des prédicats. Cela implique des choix et des précautions pour la définition d'une relation de généralité entre hypothèses. Les relations principalement étudiées sont la θ-subsomption dont la traduction opérationnelle est la SLD-subsomption qui débouche sur des opérateurs de généralisation par inversion de la résolution.
- Face à la complexité des espaces d'exemples et d'hypothèses en logique des prédicats, il est nécessaire d'utiliser des biais pour contrôler la recherche dans l'espace d'hypothèses.
- Les systèmes d'apprentissage de concepts en PLI se divisent entre les systèmes descendants, opérant par spécialisations successives afin de ne pas couvrir les exemples négatifs, et les systèmes ascendants opérant par généralisations successives à partir des exemples positifs.
- Il reste beaucoup à faire pour étendre le champ d'application de ces systèmes, en particulier lorsque les données sont numériques et bruitées. Par ailleurs, il faudra aussi résoudre le problème posé par le phénomène de transition de phase en logique des prédicats.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 6

Transfert de connaissances et apprentissage par analogie

L'apprentissage implique une exploration de l'espace des hypothèses pour rendre compte des données connues. En dehors même des problèmes soulevés dans les chapitres théoriques sur la légitimité de tout critère inductif, cette exploration est généralement intrinsèquement très coûteuse et doit donc être guidée. Lorsqu'une relation de généralité peut être définie sur l'espace des hypothèses, la recherche d'une ou plusieurs hypothèse(s) correcte(s) est considérablement aidée. Cependant, même l'utilisation d'une relation de généralité dans $\mathcal{H}$ peut être insuffisante pour pouvoir résoudre effectivement un problème d'apprentissage. L'espace des hypothèses est en effet généralement extrêmement complexe et riche, et même une recherche informée peut se révéler inefficace. On aimerait alors pouvoir débarrasser le langage de description des hypothèses de tous les descripteurs et relations non pertinents pour la tâche d'apprentissage. On aimerait également pouvoir hiérarchiser la recherche et ne prendre en compte que graduellement les détails de l'univers étudié. De même, il serait pratique de pouvoir bénéficier des explorations passées dans $\mathcal{H}$ à propos de problèmes analogues, voire de pouvoir transcrire des explorations menées dans d'autres espaces d'hypothèses relatifs à d'autres environnements ou univers de connaissances. Finalement, une question essentielle est celle de la construction d'un nouveau domaine de connaissances. Est-il possible d'envisager de le réaliser en s'aidant d'autres domaines ? Est-il envisageable de procéder autrement ? Quels mécanismes pourraient être mis en jeu ?

Ce chapitre étudie donc la modification et la construction des espaces d'hypothèses. Les approches pour résoudre ces problèmes sont encore à un stade exploratoire. Elles sont très inspirées de l'étude de la cognition humaine. Il est hors de doute que l'importance du sujet et l'état encore préliminaire de nos connaissances justifieront de nombreuses recherches à l'avenir. C'est cette conviction qui nous a fait écrire ce bref chapitre, même si les travaux actuels sont rares, et les applications encore davantage. Enfin, nous donnons une description d'un cas très particulier de l'apprentissage par transfert de connaissances, l'apprentissage par proportion analogique.

Copyright © 2010 Eyrolles.

Sommaire

1 L'apprentissage en présence de théorie 200
2 L'apprentissage par examen de preuve (*EBL*) 201
2.1 Le principe de l'*EBL* 201
2.2 Une illustration de l'apprentissage *EBL* 202
2.3 Discussion sur l'apprentissage de concept à partir d'explications 205
2.4 L'apprentissage de connaissances de contrôle à partir d'explications . . . 206
2.5 Bilan sur l'apprentissage à partir d'explications 207
3 Abstraction et reformulation des connaissances 208
4 Changement de repère, raisonnement par analogie et RàPC 210
4.1 Raisonnement par analogie 210
4.2 Raisonnement à partir de cas (RàPC) 211
5 L'apprentissage par proportion analogique 213
5.1 Définition et axiomes 213
5.2 Une introduction à l'algèbre de la proportion analogique 214
5.3 Utilisation de la proportion analogique en apprentissage 215
5.3.1 Un exemple 216
5.3.2 Dissemblance analogique et apprentissage 216
6 Bilan 217

1. L'apprentissage en présence de théorie

Jusqu'à présent, nous avons décrit l'apprentissage comme la recherche d'une hypothèse dans $\mathcal{H}$ en bonne adéquation, selon une mesure de risque, avec les données expérimentales décrites dans $\mathcal{X}$. L'apprentissage peut cependant prendre d'autres visages. Par exemple, un apprenti joueur d'échec humain sera susceptible d'apprendre le concept de « fourchette » (une menace portant simultanément sur plusieurs pièces) à partir de très peu d'exemples, parfois même à partir d'un seul. Comment est-il alors possible de généraliser cette expérience très limitée à toutes les situations de fourchettes, ou, au moins, à une grande partie d'entre elles ? L'apprentissage peut aussi porter non pas sur la détermination d'une bonne hypothèse mais sur l'efficacité de la recherche de bonnes hypothèses. Il se peut aussi que l'apprenant possède des descriptions de concepts qui soient correctes, mais qui ne soient pas vraiment opérationnelles dans le monde[1].

Le point commun à ces types d'apprentissages est l'intervention d'une connaissance importante sur le domaine d'application, connaissance que l'on appelle « théorie du domaine »[2]. C'est cette

[1] Un exemple, sans doute extrême, est celui d'un système de diagnostic de panne d'ordinateur capable de caractériser les pannes à partir des équations de Shrödinger associées aux atomes de l'appareil. Cette connaissance même correcte, ne serait cependant sans doute pas d'une grande utilité pratique et la recherche d'une description plus opérationnelle pourrait légitimement être considérée comme un apprentissage à part entière.

[2] Cette « théorie » peut prendre l'aspect très formel de programmes en logique du premier ordre, voir le chapitre 5,

Copyright © 2010 Eyrolles.

connaissance qui permet de pouvoir tirer parti de très peu d'exemples, grâce à des raisonnements enrichissant l'expérience. C'est aussi cette connaissance qui peut éventuellement être modifiée dans l'apprentissage.

2. L'apprentissage par examen de preuve (EBL)

En schématisant, le problème étudié dans l'apprentissage par examen de preuve (*Explanation-Based Learning : EBL*) est de calculer la réponse associée à une situation ou à une forme $\mathbf{x} \in \mathcal{X}$. Il se peut que la recherche de cette réponse soit impossible pour le système, ou du moins qu'elle soit très coûteuse. L'apprentissage consiste alors à rendre la recherche plus facile la prochaine fois, d'où parfois l'emploi du terme de *speed-up learning*, et ceci non seulement pour la situation $\mathbf{x}$, mais aussi pour des situations semblables. Il faut bien sûr définir la notion de similarité. Dans le cas de l'EBL, elle est obtenue à partir de l'utilisation de la théorie du domaine.

2.1 Le principe de l'EBL

On suppose que le système est soumis au problème $\mathbf{x} \in \mathcal{X}$, qui peut aussi bien être de trouver une étiquette dans une tâche de classification, comme de trouver un plan d'action. Le système cherche alors une solution $h(\mathbf{x})$ à ce problème. Si la recherche de cette solution est difficile, voire irréalisable, il peut être intéressant pour le système d'examiner une solution finalement obtenue, par le système ou grâce à un autre système ou un professeur, pour voir dans quelles conditions cette solution (éventuellement généralisée) pourrait être réutilisée dans une situation $\mathbf{x}'$ semblable. Conceptuellement, il s'agit d'identifier une région de l'espace des problèmes $\mathcal{X}$ pour laquelle la solution h, ou une solution voisine, est adéquate. La figure 6.1 illustre ce principe. D'une certaine manière, l'apprentissage EBL peut être considéré comme une technique d'extrapolation dans l'espace $\mathcal{X}$ guidée par une théorie du domaine. La partie *Apprentissage par approximation et interpolation* (en particulier le chapitre 15) de cet ouvrage explore d'autres techniques d'extrapolation fondées sur des mesures numériques et des propriétés de lissage. Peu d'auteurs ont examiné le lien entre les deux approches (une exception est [DF97]).

La solution généralisée peut prendre trois formes :

1. Celle d'une *solution toute faite*, appropriée pour une classe de problèmes dans $\mathcal{X}$, ou bien, comme c'est souvent le cas, pour la reconnaissance de concept. Il s'agit alors d'apprendre une définition opérationnelle de concepts.
2. Celle de *macro-opérateurs*, c'est-à-dire d'étapes de solution. Cela peut permettre d'accélérer la recherche d'une solution en diminuant la profondeur de recherche nécessaire.
3. Celle de *règles de contrôle* ou *heuristiques*, permettant de guider la recherche d'une solution dans l'arbre des possibilités.

Évidemment, ces trois types de connaissances apprises ne sont pas exclusifs l'un de l'autre, mais dans la pratique, les concepteurs de systèmes d'apprentissage par EBL ont privilégié l'une ou l'autre de ces possibilités. Pour des fins d'illustration dans la suite de cette section, nous avons choisi de montrer seulement comment l'apprentissage peut déboucher sur la première voie.

comme la forme de réseaux sémantiques souvent liés à des types d'inférence non déductifs : propagation de marqueurs, par exemple.

Copyright © 2010 Eyrolles.

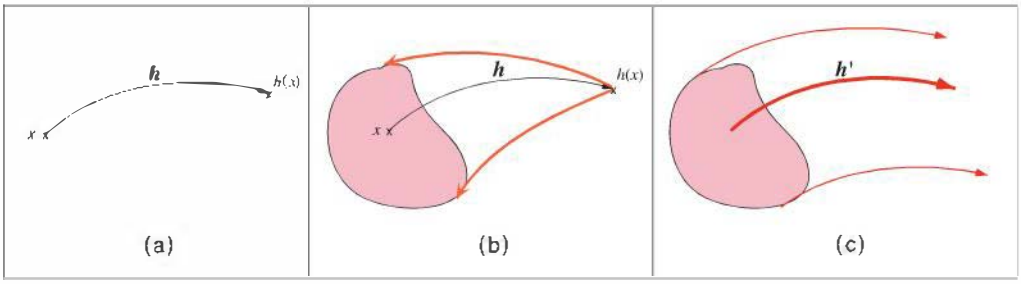


FIG. 6.1: (a) *À partir d'une solution h, trouvée ou fournie, pour le problème x, l'apprentissage EBL cherche sous quelles conditions la solution h est applicable. (b) Ces conditions suffisantes sont obtenues par un mécanisme de régression à travers la théorie $\mathcal{T}$ du domaine. (c) Cela permet de trouver à la fois une généralisation de la solution initiale h et une région de l'espace des problèmes $\mathcal{X}$ pour laquelle cette solution généralisée est applicable.*

2.2 Une illustration de l'apprentissage *EBL*

Nous montrons dans cette section comment l'apprentissage *EBL* permet d'apprendre une définition opérationnelle et générale d'un concept à partir d'un exemple positif unique. Pour cela nous reprenons l'exemple devenu célèbre de l'apprentissage du concept d'objets empilables, initialement exposé dans [MKKC86] qui est aussi l'article fondateur de l'*EBL*.

Soit un monde d'objets caractérisés par les attributs : *couleur, volume, propriétaire, matériau, type* et *densité*. Il existe de plus une relation possible entre les objets : la relation *sur* traduisant qu'un objet est sur un autre. La tâche consiste à apprendre le concept d'empilement d'un objet sur un autre. Plus précisément, il s'agit d'apprendre une définition du prédicat :

```
empilable(Objet1,Objet2)
```

vrai lorsque l'on peut empiler l'`Objet1` sur l'`Objet2`. Nous supposons que l'apprenant dispose préalablement d'une *théorie du domaine* $\mathcal{T}$ décrite sous la forme de règles logiques que nous décrirons ici en utilisant, comme dans le chapitre 5, la syntaxe Prolog :

(T1) : `poids(X,W) :- volume(X,V), densité(X,D), W is V*D.`

(T2) : `poids(X,50) :- est_un(X,table).`

(T3) : `plus_léger(X,Y) :- poids(X,W1), poids(Y,W2), W1 < W2.`

On suppose de plus que l'apprenant cherche une définition opérationnelle (c'est-à-dire facilement évaluable) du *concept but* :

(C1) : `empilable(X,Y) :- plus_léger(X,Y).`

(C2) : `empilable(X,Y) :- not(fragile(Y)).`

Plus formellement, le *critère d'opérationnalité* impose que le concept soit exprimé à l'aide de prédicats utilisés pour décrire l'exemple (*volume, densité, couleur,...*) et de prédicats facilement évaluables (comme le prédicats <).

Finalement, l'apprenant est pourvu de l'*exemple positif* décrit par les faits suivants :

```
sur(obj1,obj2).
est_un(obj1,boite).
est_un(obj2,table).
couleur(obj1,rouge).
couleur(obj2,bleu).
```

Copyright © 2010 Eyrolles.

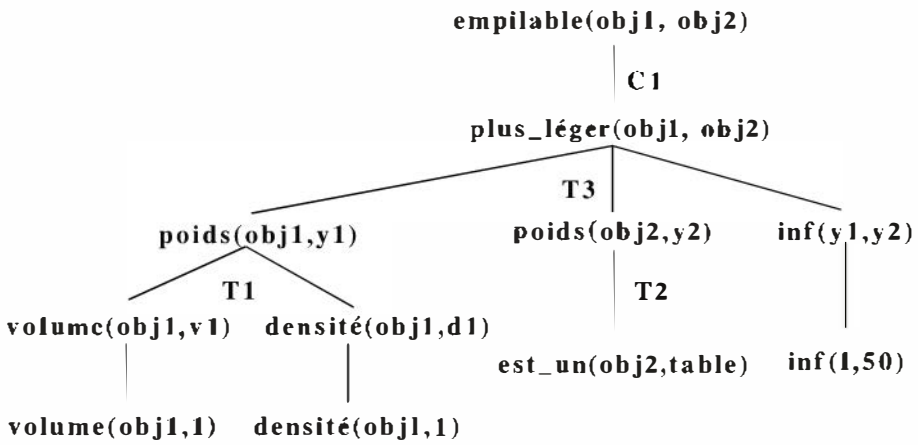


FIG. 6.2: *Un arbre de preuve obtenu dans la théorie du domaine $\mathcal{T}$ montrant que l'exemple satisfait bien le concept cible.*

```
volume(obj1,1).
volume(obj2,0.1).
propriétaire(obj1,frederic).
propriétaire(obj2,marc).
densité(obj1,0.3).
matériau(obj1,carton).
matériau(obj2,bois).
```

L'apprentissage EBL consiste d'abord à examiner une preuve, exprimée à l'aide de la théorie du domaine, que l'exemple fourni est bien un exemple positif du concept cible, puis à généraliser cette preuve pour en extraire un ensemble de contraintes suffisant à garantir la validité de la preuve générale.

Le **premier pas** nécessite que la théorie du domaine soit suffisante pour qu'il soit possible de trouver une preuve, aussi appelée explication (d'où l'appellation EBL), montrant que l'exemple satisfait au concept cible. Chaque branche de la preuve peut éventuellement être réduite pour se terminer sur une expression vérifiant le critère d'opérationnalité. Dans le cas présent, l'explication de l'exemple est donnée sous forme graphique dans la figure 6.2.

Le **second pas** consiste à analyser l'arbre de preuve trouvé pour identifier les conditions de son applicabilité afin de découvrir l'ensemble des situations dans lesquelles il pourrait s'appliquer. Pour le cas exposé ici, la figure 6.3 décrit la généralisation de la preuve donnée dans la figure 6.2.

L'exemple positif initial, le seul connu pour le concept cible, a été généralisé de plusieurs points de vue. D'abord, il était décrit par une collection d'attributs dont beaucoup, n'intervenant pas dans la preuve, se révèlent non pertinents et peuvent donc être éliminés. Ensuite, de nombreuses constantes ont été remplacées par des variables. Notons cependant que ce n'est pas systématique. La constante numérique 50, par exemple, demeure dans la preuve généralisée car elle n'a pas été introduite par une instanciation aux feuilles de l'arbre de preuve mais par une règle de la théorie. Finalement, la généralisation va plus loin que cette simple variabilisation grâce à la *procédure de régression* d'une formule à travers une règle proposée par Waldinger [Wal77]. Cette procédure calcule les conditions suffisantes d'une conclusion C en fonction d'une preuve P, c'est-à-dire un ensemble d'assertions A tel que A entraîne C grâce à P. Dans le cas de la régression d'une conclusion à travers un arbre de preuve, la procédure fonctionne itérativement à travers les étapes

Copyright © 2010 Eyrolles.

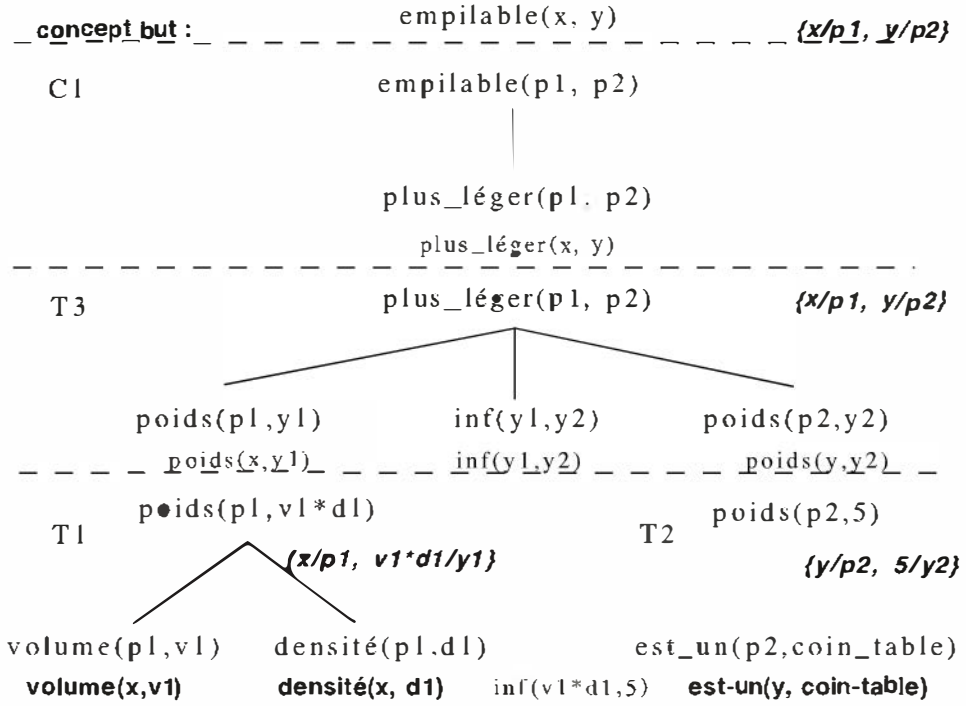


FIG. 6.3: *L'arbre de preuve généralisé est obtenu par régression du concept cible dans l'arbre de preuve en calculant à chaque étape (dénotée par les lignes en pointillés) les littéraux les plus généraux permettant cette étape (indiqués en gras). À la fin de ce processus, la conjonction des littéraux obtenus aux feuilles de l'arbre généralisé fournit la définition opérationnelle et généralisée du concept cible.*

de la preuve, calculant à chaque fois les conditions suffisantes de la conclusion de l'étape, en fonction de la règle de dérivation employée pour cette étape. La procédure se termine lorsqu'elle a parcouru toutes les étapes de la preuve initiale pour atteindre les feuilles de l'arbre de preuve.

Un exemple de l'application de cette procédure est donné par la figure 6.3. Les éléments de la preuve initiale trouvée pour l'exemple positif du concept `empilable(Obj1,Obj2)` sont représentés en fonte normale. La frontière de chaque étape de régression est indiquée par les lignes en pointillés, avec à chaque fois en gras les conditions suffisantes calculées. Par exemple, le concept cible général est `empilable(X,Y)`. On calcule sa régression à travers la règle `empilable(X,Y) :- plus_léger(X,Y)`, ce qui produit la précondition `plus_léger(X,Y)`. On continue alors en calculant la régression de cette expression à travers (T3) le pas suivant de la preuve, ce qui conduit à l'ensemble de préconditions {`poids(X,Y1)`, `inf(Y1,Y2)`, `poids(Y,Y2)`}. Étape par étape, on atteint ainsi les feuilles de l'arbre de preuve, produisant la définition générale suivante :

```
empilable(X,Y) :- volume(X,VX), densité(X,DX),
                  est_un(Y,table), inf(VX,DX,50).
```

Cette procédure a été implantée en particulier dans l'algorithme Prolog-EBG [KCC87]. Lorsqu'une théorie correcte et complète lui est fournie, elle produit une hypothèse (ensemble de règles Prolog) elle-même correcte et couvrant les exemples positifs connus des concepts à apprendre. Ces règles sont des conditions suffisantes décrivant les concepts cible en fonction de la théorie du domaine.

Copyright © 2010 Eyrolles.

2.3 Discussion sur l'apprentissage de concept à partir d'explications

Plusieurs problèmes méritent d'être soulignés.

- Il est important de remarquer que l'algorithme rapidement décrit ci-dessus produit seulement une définition suffisante (et non nécessaire) du concept cible. Ceci est dû au fait que la preuve est guidée par les exemples fournis qui n'épuisent pas forcément toutes les possibilités de preuve du concept. Par exemple, dans l'apprentissage du concept `empilable(X,Y)`, la théorie propose deux définitions, l'une se rapportant au fait que `X` est plus léger que `Y`, l'autre au fait que `Y` n'est pas fragile. L'exemple positif fourni ne se rapporte qu'à la première définition, et les conditions trouvées par régression à travers la preuve relative à cet exemple ne concernent donc que la première définition possible du concept. C'est bien pourquoi on parle d'apprentissage à partir d'explications. Si l'on apprend bien une généralisation de l'exemple positif, il s'agit cependant d'une spécialisation de la définition initiale du concept cible.

  Certains commentateurs de l'apprentissage EBL ont fait remarquer que l'exemple positif fourni est, d'un certain côté, redondant et inutile. On pourrait très bien apprendre une définition complète du concept cible par régresssion selon tous les arbres de preuve possibles en arrêtant les preuves lorsque le critère d'opérationnalité est satisfait comme cela se fait en programmation logique par évaluation partielle [HB88]. L'intérêt des exemples n'est cependant pas négligeable. Ils permettent en effet d'orienter la recherche de preuve dans un espace de possibilités généralement immense. De plus, si ces exemples sont bien choisis, ils sont représentatifs de l'environnement et guident donc l'apprentissage vers les seules régions de $\mathcal{X}$ pertinentes. Ainsi, tant l'apprentissage qu'ensuite la reconnaissance d'un concept sont accélérés par élimination des cas de preuves inutiles (c'est d'ailleurs l'une des raisons pour laquelle les Anglo-Saxons parlent de *speed-up learning*).

- L'exemple d'apprentissage ci-dessus se traitait à l'aide d'une théorie du domaine à la fois correcte, complète (toute formule vraie dans l'interprétation intentionnelle de la théorie doit être déductible de la théorie), et adéquate, c'est-à-dire décrite dans des termes pouvant conduire à une reformulation utile des concepts. Il est évident que la qualité des descriptions apprises par EBL dépend grandement des propriétés de la théorie du domaine. C'est là l'une des plus importantes limitations pour la mise en pratique de la méthode car il est rare que l'on dispose d'une théorie parfaite avant apprentissage. On peut caractériser les problèmes en trois classes :

  1. Le cas d'une *théorie incomplète*. Cela peut peut avoir pour conséquence l'impossibilité de trouver une preuve pour un exemple positif, ou l'obtention d'une preuve incomplète, voire de multiples preuves mutuellement incohérentes.
  2. Le cas d'une *théorie incorrecte*. Dans ce cas, il peut y avoir découverte de preuve incorporant des contradictions avec le monde réel, c'est-à-dire apprentissage de mauvais concepts.
  3. Finalement, il y a le cas de *théorie impossible à traiter*, interdisant de trouver des preuves de l'exemple fourni.

  Ces trois classes de problèmes ont fait l'objet de travaux, surtout la première. C'est ainsi qu'on a cherché à étendre la notion de preuve à des preuves « plausibles » exprimées dans des extensions de la logique (incertaines ou floues). On a également cherché à compenser l'insuffisance de la théorie du domaine par l'utilisation simultanée de plusieurs exemples.

- S'il est intéressant de savoir utiliser des exemples positifs d'un concept à l'aide d'une théorie du domaine, il peut être tout aussi tentant d'utiliser des exemples négatifs. Par exemple, on

Copyright © 2010 Eyrolles.

pourrait vouloir apprendre à partir d'une tentative infructueuse de mater le roi adverse dans une situation pourtant apparemment favorable. Deux approches sont envisageables dans ce cas. La première consiste à traiter l'exemple négatif comme un exemple positif, le généraliser et utiliser la négation du concept appris pour caractériser les situations nécessaires au succès. La seconde part de l'échec d'une preuve d'un exemple négatif pour analyser comment il faudrait la modifier pour parvenir à un succès et ainsi caractériser ce qui sépare les situations positives des situations négatives. L'apprentissage à partir d'échecs (*learning by failure*) a été particulièrement étudié dans le contexte de l'apprentissage de connaissances de contrôle puisqu'il y est crucial de savoir quand il faut éviter d'avoir recours à un opérateur.

2.4 L'apprentissage de connaissances de contrôle à partir d'explications

L'apprentissage de concept à partir d'explications requiert donc une théorie du domaine aussi correcte et complète que possible. Un domaine dans lequel il est naturel de supposer une telle théorie est celui de la résolution de problèmes. Dans ce contexte en effet, l'enjeu de l'apprentissage est souvent de rendre efficace un résolveur de problème capable en principe de résoudre n'importe quel problème, mais souvent inutilisable en pratique à cause de la complexité de la recherche d'une solution. L'apprentissage peut alors prendre trois formes :

1. L'*apprentissage des conditions* dans lesquelles il est intéressant d'envisager l'utilisation d'un opérateur.
2. L'*apprentissage de macro-opérateurs* qui sont souvent utiles dans un environnement donné et qu'il est donc intéressant de connaître.
3. L'*apprentissage d'heuristiques de contrôle* permettant de trier par ordre d'intérêt décroissant les opérateurs envisageables dans un état donné.

Pour apprendre à partir d'explications *les conditions d'application d'un opérateur*, il suffit de le considérer comme un concept dont on cherche à caractériser le domaine d'application. Pour obtenir des exemples positifs d'application d'opérateurs, on examine les séquences d'opérateurs correspondant à des solutions de problèmes. Cela fournit pour chaque opérateur utilisé un ou des état(s) dans le(s)quel(s) il a été employé, et donc un ou plusieurs exemple(s) positif(s) de son application. Un exemple de cette approche figure dans le chapitre 4 avec le système LEX de Mitchell. On y voyait à l'œuvre un apprentissage par élimination des candidats dans l'espace des versions ; une variante appelée LEX2 a été développée, qui utilise un apprentissage à partir d'explications.

La deuxième approche consiste à chercher à améliorer l'efficacité d'un système de résolution de problème ou de planification en *définissant des macro-opérateurs* correspondant à des sous-séquences généralisées de solutions. Il faut d'une part sélectionner les séquences utiles : si elles sont trop spécifiques, elles serviront rarement et ralentiront inutilement le processus de choix à chaque nœud de la recherche. Inversement, si elles correspondent à des sous-séquences courtes, donc généralement d'un domaine d'application plus vaste, elles seront moins utiles car ne correspondant pas à des « grands pas » dans la solution. Nous reviendrons sur ce problème. D'autre part, il faut apprendre leur domaine d'application, ce qui s'effectue souvent par régression comme dans l'apprentissage de concept.

La troisième approche est d'*apprendre des heuristiques de contrôle* guidant le système lorsqu'il cherche une solution à un problème. Un système opérant de cette manière tout en apprenant des macro-opérateurs est le système SOAR [LRN86]. Il utilise pour cela un mécanisme appelé *chunking* qui produit des « chunks » (littéralement « gros morceaux ») ou macro-opérateurs. SOAR est un système général de résolution de problème opérant par décomposition de problèmes en sous-problèmes. Grâce à la nature récursive de ce procédé, des hiérarchies de sous-buts sont

Copyright © 2010 Eyrolles.

produites. À chaque fois qu'une impasse est rencontrée dans la résolution d'un problème, un sous-but est engendré et le système cherche à le résoudre par une méthode de recherche faible telle que *générer et tester*[3]. La solution trouvée pour sortir de l'impasse est alors vue comme une explication de la démarche à suivre pour sortir à l'avenir d'impasses similaires. Le mécanisme de chunking entre en jeu lorsqu'un sous-but a été résolu. Un résumé de la procédure de résolution du sous-but est enregistré sous forme de règles de production. La partie action de chaque règle est fondée sur les résultats du sous-but ; la partie condition ou antécédent exprime les aspects pertinents qui déterminent le succès de l'emploi de la procédure. SOAR a été testé dans de très nombreux problèmes et est proposé comme modèle cognitif de résolution de problème [New90].

PRODIGY [Min88, Min90] est un autre système intéressant d'apprentissage d'heuristiques de contrôle. Il s'agit d'un système de planification fonctionnant en utilisant une théorie générale sur la planification, une théorie du domaine et une description des opérateurs à la STRIPS[4] avec des listes de préconditions, d'ajouts et de retraits exprimées avec des prédicats sur le monde. Comme SOAR, PRODIGY utilise une résolution de problèmes par décomposition en sous-problèmes. Après avoir produit un plan d'action, le système analyse ses bons choix et ses impasses et cherche à les expliquer en termes de sa théorie du domaine. Il est ainsi capable de produire des règles de contrôle telles que :

```
if   (and (current-node node)
          (candidate-goal node (on X Y))
          (candidate-goal node (on Y Z)))
then (prefer goal (on Y Z) to (on X Y))
```

Ce genre de règle est obtenu après l'analyse d'un échec lorsque le robot a essayé de placer d'abord la sous-pile d'objets supérieure avant une sous-pile inférieure. Ceci n'est évidemment qu'un petit exemple des capacités de PRODIGY.

Comme SOAR, PRODIGY a été testé intensivement. Il en est ressorti que l'apprentissage de macro-opérateurs et de règles de choix des opérateurs n'était pas nécessairement avantageux. En effet, chaque nouvelle règle, chaque nouvel opérateur ajoute un choix à l'ensemble des choix possibles à chaque étape de résolution. Le coût ainsi encouru peut dépasser l'avantage retiré. C'est ce que Minton, le concepteur principal de PRODIGY, a appelé le *problème de l'utilité*. C'est pourquoi PRODIGY incorpore un module d'évaluation de l'utilités de chaque règle et opérateur, prenant en compte l'économie éventuellement réalisée et le coût en terme de temps de sélection et d'application. Seuls les règles et opérateurs dont l'utilité mesurée est jugée assez élevée sont conservés. De cette manière, les gains empiriquement mesurés sur un ensemble de tâches sont de l'ordre de 20 % à 110 % avec une moyenne autour de 40 %, ce qui sans être négligeable, est toutefois étonnament faible quand on pense que le fait d'avoir un facteur de branchement (nombre de choix disponibles à chaque nœud) réduit (grâce aux règles de contrôle) devrait permettre un gain exponentiel. Il s'agit là d'un problème qui est encore du domaine de la recherche.

2.5 Bilan sur l'apprentissage à partir d'explications

Dans ce survol de l'apprentissage à partir d'explications, nous avons laissé de côté certains problèmes tels celui de l'apprentissage dans le cas de preuves récursives ou dans le cas de preuves non déductives comme cela se rencontre dans des systèmes à base de schémas. Nous renvoyons

[3] Méthodes consistant à engendrer les différentes possibilités à chaque point de choix puis à choisir en fonction d'une fonction d'évaluation. La célèbre méthode de recherche A^* en est un exemple.

[4] STRIPS (Stanford, 1971)[FN71, FN72] est le grand ancêtre de la plupart des systèmes de planification. Il a en particulier fixé la manière de représenter les opérateurs.

Copyright © 2010 Eyrolles.

le lecteur intéressé à l'abondante littérature portant sur l'apprentissage *EBL*. Il faut surtout retenir que ces méthodes d'apprentissage permettent l'obtention de généralisations justifiées, à partir d'un petit nombre d'exemples, ce qui les distingue radicalement des méthodes inductives classiques. Si, les travaux sur cette approche ont été peu nombreux depuis plus d'une décennie, nous pensons cependant, comme Gerald Dejong l'un de leurs promoteurs, qu'il ne peut s'agir que d'une éclipse passagère. Certes il est difficile de disposer d'une théorie du domaine forte, mais il existe cependant des domaines où de telles théories préexistent, comme la planification, le contrôle ou le diagnostic. Dans ces cas là, et en particulier quand chaque exemple est coûteux à obtenir, l'apprentissage à partir d'explications peut être très intéressant. Plus généralement, on ne voit pas comment l'emploi exclusif de techniques sans usage de théorie du domaine pourrait couvrir l'ensemble des problèmes d'apprentissage. Les techniques telles que l'*EBLEBL* ont donc leurs beaux jours devant elles.

De fait, la technique *EBL* a trouvé récemment un champ d'application important dans le domaine des problèmes de satisfaction de contraintes (SAT). Dans ces problèmes, la question est de décider si une formule logique sous forme CNF (« *Conjunctive Normal Form* »), par exemple $(x_1 \vee \bar{x_7} \vee x_9) \wedge (\bar{x_2} \vee x_5 \vee \bar{x_6}) \wedge \ldots)$ est satisfiable : on peut trouver une assignation des variables booléennes x_i telle que la formule est vraie ou non. En effet, les algorithmes complets de recherche de satisfiabilité sont tous, à ce jour, des variations de l'algorithme de Davis et Putnam (1960) qui réalise une exploration systématique de l'espace de recherche en testant en chaque nœud une assignation possible d'une variable avec retour arrière lorsqu'une branche ne peut conduire à une solution. La majorité des recherches ont porté sur la conception d'heuristiques permettant une exploration plus efficace de l'arbre de recherche, sans que cela change notablement la puissance des algorithmes. Cependant, des travaux récents ont fait appel à une technique dite d'*apprentissage de clause,* issue des techniques d'*EBL*, et qui consiste à chercher une explication pour les échecs (points de retour). L'apprentissage produit alors une clause de conflit qui est ajoutée à l'ensemble des clauses et qui permet d'accélérer la recherche ultérieure. Les méthodes développées ainsi ont considérablement augmenté la puissance des algorithmes de recherche, permettant de résoudre des problèmes inaccessibles jusque-là (voir [BKS04]).

3. Abstraction et reformulation des connaissances

Les capacités d'inférence et de raisonnement d'un agent dépendent de la manière dont est exprimée sa connaissance et dont celle-ci peut interagir avec sa perception de l'environnement. Un pouvoir expressif trop limité de la représentation utilisée peut empêcher certains raisonnements. C'est une des raisons pour lesquelles il faut des millions d'exemples en apprentisage par renforcement pour apprendre le concept de « fourchette » aux échecs, quand quelques exemples suffisent avec une représentation en logique des prédicats et un apprentissage à partir d'explications. D'un autre côté, trop d'informations disponibles peuvent conduire à des raisonnements incapables de trouver une solution en un temps raisonnable. Il est donc crucial pour un agent intelligent de savoir contrôler l'information qu'il prend en compte pour résoudre une tâche. Il serait extrêmement utile de disposer de systèmes capables d'apprendre ce contrôle. L'état de l'art en apprentissage artificiel est encore loin de pouvoir répondre à ce problème, mais on réfléchit depuis longtemps sur les changements de représentation et les reformulations des connaissances en intelligence artificielle. Nous en évoquons rapidement quelques aspects ici, renvoyant en particulier à la synthèse très informée de Jean-Daniel Zucker [Zuc01].

Copyright © 2010 Eyrolles.

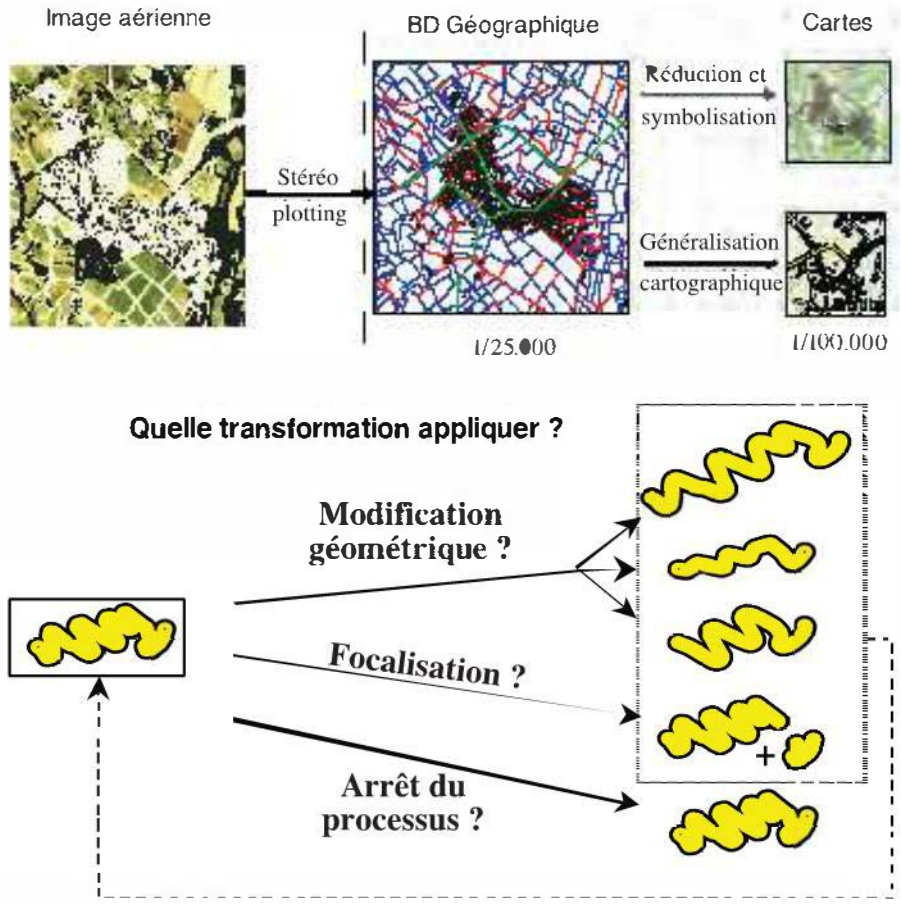


FIG. 6.4: *En haut est schématisé le travail du cartographe : comment traduire une photo aérienne en carte ? En bas, on voit que pour traduire le tracé d'une route par exemple, il peut faire appel à plusieurs opérateurs applicables en séquence. L'une des tâches d'apprentissage consiste à apprendre quand appliquer chacun de ces opérateurs. (Tiré de [Zuc01], p.21).*

L'idée générale des changements de représentation est de trouver une représentation permettant une résolution simple de la tâche courante. Depuis les travaux pionniers d'Amarel en 1968 [Ama68], on sait qu'une bonne représentation peut rendre triviale la recherche d'une solution. J.-D. Zucker distingue trois types de changements de représentation :

- l'*abstraction* : changement de représentation, dans un même formalisme, qui simplifie la résolution du problème courant en cachant des détails et en préservant des propriétés désirables ;
- la *reformulation* : modification du formalisme de représentation tout en conservant les mêmes informations ;
- l'*approximation* : simplification des inférences, dans la même représentation, afin de simplifier la résolution.

L'abstraction est souvent considérée comme un changement de granularité ou de niveau de détail d'une représentation. Cela néglige le fait qu'il n'y a pas, généralement, de hiérarchie toute faite de niveaux de détails. Il s'agit de « construire » une description de haut niveau pertinente pour le contexte et la tâche. Pour cela plusieurs voies sont possibles. La plus étudiée en intelligence artificielle consiste à considérer le processus d'abstraction comme celui d'une projection d'une

Copyright © 2010 Eyrolles.

représentation dans une autre *modulo* une relation d'équivalence, de telle manière que ne soient distingués dans la nouvelle représentation que les éléments qui doivent l'être pour la résolution du problème courant. Ainsi se trouvent éliminés les « détails » superflus. Une autre approche est de considérer l'abstraction comme un problème pour lequel des opérateurs de changement de représentation sont disponibles et dont on cherche une solution sous forme de séquence de tels opérateurs. La figure 6.6, tirée de [Zuc01] illustre ce point de vue en schématisant la tâche du géographe qui doit représenter sur une carte les éléments d'un paysage photographié par satellite. La carte finalement produite résulte du choix de nombreux opérateurs. On peut alors envisager d'apprendre automatiquement les conditions d'application de chaque opérateur, de manière similaire à ce qui a été discuté dans la section 2 sur l'*EBL*.

L'étude de l'abstraction est certainement étroitement liée à celle de l'apprentissage. Il s'agit là d'un vaste champ de recherche qui reste encore largement à explorer.

4. Changement de repère, raisonnement par analogie et RàPC

4.1 Raisonnement par analogie

L'induction suppose connu un échantillon $\mathcal{S} = \langle(\mathbf{x}_i, f(\mathbf{x}_i))\rangle_{i=1..m}$ à partir duquel on induit une fonction h aussi proche de f que possible permettant la prédiction d'une réponse en tout point de l'espace $\mathcal{X}$. Le *raisonnement par analogie* suppose beaucoup moins : de la connaissance d'un couple $(\mathbf{x}, f(\mathbf{x}))$, il cherche à prédire la réponse en un point $\mathbf{x}' \neq \mathbf{x}$ (voir figure 6.5). Évidemment, en l'absence de davantage de données, cela n'est possible que si l'on connaît des propriétés utiles sur l'espace $\mathcal{X} \times \mathcal{F}$.

Dans sa version la plus fruste, le raisonnement par analogie opère comme la méthode par plus proches voisins : lorsqu'il est difficile, voire impossible, de trouver une solution pour le problème $\mathbf{x} \in \mathcal{X}$, il peut être intéressant d'emprunter la solution connue d'un problème $\mathbf{x}'$ « proche » de $\mathbf{x}$. Toute la question devient alors de caractériser la notion de proximité. Lorsque, de plus, le problème $\mathbf{x}'$ dont on s'inspire n'appartient pas au même domaine de connaissance, on parlera souvent de similarité. Cette notion est dépendante de la tâche : deux situations peuvent être soit très similaires, soit très différentes en fonction du contexte. Un caillou et mon iPhone peuvent tous les deux servir de presse-papiers pour empêcher des feuilles de s'envoler pendant que je rédige cet ouvrage la fenêtre ouverte. Pour cette tâche, ils sont similaires. Je me garderai cependant bien d'utiliser l'un et l'autre indifféremment lorsque je veux téléphoner à mon coauteur ou faire des ricochets sur la rivière toute proche.

Nous verrons au chapitre 18 comment se traduit la notion de proximité dans un espace décrit par des attributs numériques. Vapnik [Vap95] parle de *transduction* lorsque l'on cherche à prédire la valeur en un point $\mathbf{x}'$ connu à l'avance à partir d'un échantillon d'exemples $\mathcal{S} = \langle(\mathbf{x}_i, u_i)\rangle_{1 \leq i \leq m}$. Selon lui, il n'est pas alors nécessaire de passer par l'induction d'une fonction h définie sur tout l'espace d'entrée $\mathcal{X}$. Il suppose cependant l'existence d'une telle fonction s'appliquant uniformément dans $\mathcal{X}$. Cette vision ne prend pas en compte la possibilité qu'il est inutile de chercher une fonction unique s'appliquant dans tout l'espace des situations, mais qu'il puisse y avoir pour chaque situation une fonction définie localement, qu'il faut « déformer » (minimalement) pour pouvoir l'appliquer en un autre point[5]. Lorsqu'une ou des théorie(s) de domaine permet(tent) de redécrire les situations, la notion de similarité doit faire intervenir des propriétés des redes-

[5] Les mathématiciens seront tentés d'y reconnaître la notion de covariance sur des variétés différentielles.

Copyright © 2010 Eyrolles.

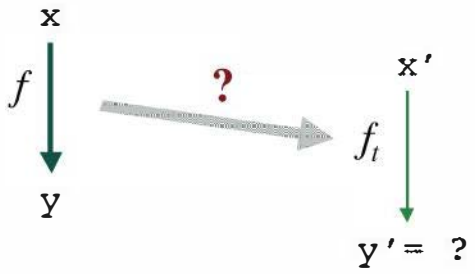


FIG. 6.5: *Schématiquement, l'analogie consiste à trouver la réponse à une question* x' *connaissant un couple* $(x, f(x))$ *(ou plus précisément* (x, y)*, sans généralement connaître* f*). La fonction* f_t *produisant la réponse en* x' *peut être différente de* f *en* x.

criptions possibles et des théories qu'elles mettent en jeu. On parle alors vraiment d'analogies. Celles-ci présentent certaines particularités notables.

- Si l'analogie entre deux situations fait souvent intervenir la recherche de « points communs » entre elles, elle n'est cependant pas équivalente à un processus de généralisation. On peut plutôt l'envisager comme la recherche du canal de transmission le plus économique entre une description de situation et une autre (voir [Cor96]). Cette recherche met en avant des primitives ou des structures de description potentiellement intéressantes parmi toutes les primitives et structures envisageables. En ceci, elle aide déjà à la solution d'un problème donné par la focalisation sur les descriptions pertinentes.
- Contrairement à une distance, la notion de proximité mise en jeu dans l'analogie n'est pas symétrique, et, corollairement, ce qui est transporté d'une situation à l'autre n'est pas forcément la même chose dans un sens et dans l'autre.

Les modèles de raisonnement par analogie ont surtout été étudiés dans le cadre de recherches sur la cognition humaine. D'autres types de raisonnements permettant le transfert de connaissances entre domaines ont été également abordés, par exemple le *blending* [FT98] ou l'*effet tunnel cognitif* [CTC00]. Il reste encore beaucoup à faire pour comprendre pleinement ces types de raisonnements mêlant divers types de représentations, de connaissances et de d'inférences. Nous recommandons à ce sujet la lecture des ouvrages de Hofstadter, dont [Hof95].

4.2 Raisonnement à partir de cas (RàPC)

Le raisonnement à partir de cas (RàPC) (*Case-Based Reasoning, CBR*) est tourné vers la résolution de problèmes par l'utilisation d'une base de cas. À côté de son inspiration venant des sciences cognitives (nos inférences et nos raisonnements sont souvent très guidés par nos expériences similaires antérieures), une motivation pour la modélisation informatique de ce type de raisonnement vient d'une réaction aux difficultés rencontrées par les systèmes experts à la fin des années quatre-vingts. Il s'est avéré en effet difficile d'« extraire » des règles de raisonnement des experts pour tout un nombre de raisons maintenant très bien documentées. En revanche, il semblait que les experts fournissaient aisément des cas tirés de leur expérience et illustratifs de leur expertise. La tentation était forte, dès lors, de construire une mémoire de cas et de l'exploiter en présence d'un nouveau problème.

Dans cette optique, la résolution du nouveau problème nécessite une étape de *remémoration* afin d'identifier un *cas source* (scénario ou schéma) le plus similaire possible, puis une étape d'*adaptation* destinée à modifier la solution source juste assez pour résoudre le *cas cible*. D'un certain côté, il s'agit d'un raisonnement rappelant à la fois l'EBL (on cherche à résoudre un

Copyright © 2010 Eyrolles.

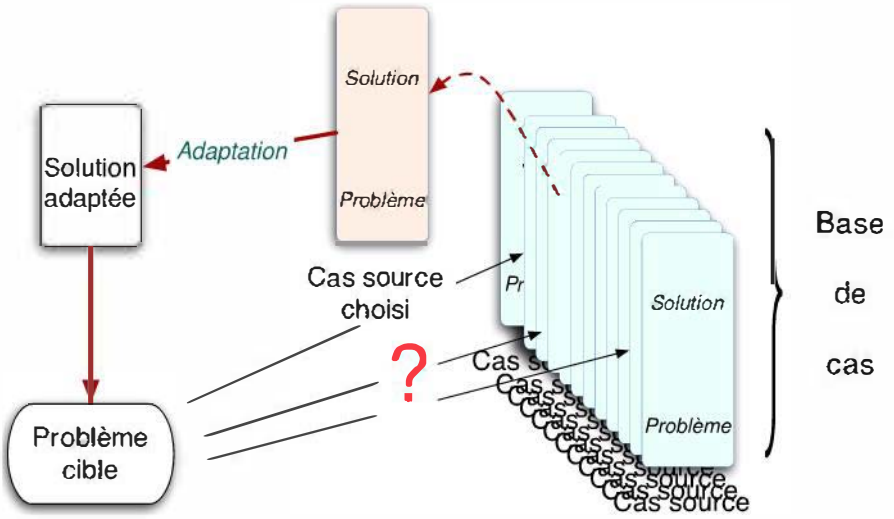


FIG. 6.6: *Étapes du raisonnement à partir de cas.*

problème en adaptant une connaissance antérieure), et le raisonnement par analogie (on s'appuie sur un cas source pour en adapter la solution au cas cible). Cependant, le RàPC se distingue de ces deux types de raisonnement par le fait qu'il ne s'appuie pas sur une preuve comme l'EBL, mais sur un cas source, et qu'il doit identifier celui-ci dans la base de cas, alors que la source est généralement supposée fournie dans le raisonnement par analogie. De plus, le raisonnement par cas implique une troisième étape qui est celle de la *modification de la base de cas* après résolution, soit par stockage du nouveau cas, soit par modification de la base de cas existante.

Pour être performant, le RàPC doit utiliser une *mesure de similarité* appropriée entre cas source potentiels et cas cible ou bien une *indexation* de la mémoire efficace. Dans les deux cas, cela implique de trouver une description adéquate du cas cible. Le plus simple est de le décrire sous la forme d'une collection d'attributs. Cependant, la similarité ou l'indexation requièrent à tout le moins une pondération des différents descripteurs. Ainsi, lors de l'évaluation du prix d'une voiture d'occasion, le kilométrage joue un rôle important alors que la couleur est secondaire. Mais s'il s'agit de voitures de collection, il se peut que la couleur soit prépondérante. La description est donc dépendante du problème à résoudre. Par ailleurs, il peut être insuffisant de ne s'intéresser qu'à la similarité entre le cas cible et les cas source potentiels. Il est en effet possible que la « solution » d'un cas source apparemment proche ne soit pas adaptable au cas cible. Idéalement, il faudrait donc tenir compte également du coût de l'adaptation dans la similarité.

Une fois qu'un cas cible a été résolu, ou non, il est utile de tirer bénéfice de cette expérience. Cela peut prendre *a priori* plusieurs formes :

- l'adaptation de la *mesure de similarité* ;
- l'amélioration de la *méthode d'adaptation ou de résolution* ;
- la modification de *la base de cas*, soit par mémorisation du cas cible résolu, qui devient ainsi un nouveau cas source potentiel, soit par modification de la structure de la base de cas : modification des cas, changement de la hiérarchie, de l'indexation, etc.

Á défaut de pouvoir rentrer dans les détails dans le cadre de cet ouvrage, nous renvoyons aux références déjà anciennes mais encore fondamentales [RS89, Kol93, AP94], ainsi qu'aux conférences annuelles francophones sur le raisonnement à partir de cas (par exemple http://liris.cnrs.fr/~rapc/mediawiki/index.php/Rapc2009, pour 2009).

Copyright © 2010 Eyrolles.

5. L'apprentissage par proportion analogique

5.1 Définition et axiomes

Une forme particulière d'analogie, déjà explorée par Aristote [Hes65, GHK01], est la *proportion analogique*. Il s'agit d'une relation entre quatre objets du même univers qui s'énonce ainsi :

« A *est à* B *comme* C *est à* D »

et se note

$$A : B :: C : D$$

Selon la nature des objets, la signification de cette relation varie. Prenons quatre exemples.

- Dans les nombres binaires, ou les valeurs de vérité logiques, on a par exemple : $0 : 1 :: 0 : 1$ mais pas $1 : 1 :: 0 : 1$ ni $0 : 1 :: 1 : 0$.
- Dans les nombres réels, la relation $\frac{A}{B} = \frac{C}{D}$ exprime une proportion analogique. Sous cette interprétation, on peut écrire : $6 : 3 :: 22 : 11$.
- Une proportion analogique sur la sémantique de mots français est par exemple :
  « une jument *est à* un poulain *comme* une vache *est à* un veau ».
- La relation de proportion analogique suivante est morphologique :
  « abcd *est à* ecd *comme* abfab *est à* efab ».
  Elle ne porte en effet que sur la séquences des lettres. Sa signification est la suivante :

  > Pour transformer abcd en ecd, exactement comme pour transformer abfab en efab, il faut remplacer en début de séquence ab par e.

Il n'y pas de définition universelle de la proportion analogique entre quatre objets appartenant à un ensemble X, puisque les relations « est à » et « ce que » dépendent de la nature de X. Cependant, selon la signification habituelle du mot « analogie » en philosophie et en linguistique, trois axiomes fondamentaux sont généralement requis [Lepage03] :

Définition 6.1 (Proportion analogique)

Une proportion analogique sur un ensemble X est une relation sur X^4, c'est-à-dire un sous-ensemble $\mathcal{A} \subset X^4$ qui respecte les trois axiomes ci-après. Pour tout quadruplet (A, B, C, D) de $\mathcal{A}$, les quatre éléments A, B, C et D sont dits en proportion analogique *et l'on note : « la proportion analogique $A : B :: C : D$ est vraie », ou simplement $A : B :: C : D$, qui se lit « A est à B comme C est à D ».*

Les deux premiers axiomes sont :

$$\text{Symétrie de la relation « est à » :} \quad A : B :: C : D \;\Leftrightarrow\; C : D :: A : B$$
$$\text{échange des moyens :} \quad A : B :: C : D \;\Leftrightarrow\; A : C :: B : D$$

*Le troisième axiome (*déterminisme*) exige que l'une des deux implications suivantes soit vraie (l'autre étant une conséquence) :*

$$A : A :: B : x \;\Rightarrow\; x = B$$
$$A : B :: A : x \;\Rightarrow\; x = B$$

Au regard de ces axiomes, il s'avère que cinq autres formulations sont équivalentes à $A : B :: C : D$:

Copyright © 2010 Eyrolles.

$$B : A :: D : C \qquad D : B :: C : A \qquad C : A :: D : B$$
$$D : C :: B : A \qquad \text{et} \qquad B : D :: A : C$$

Donc, au total, huit proportions analogiques sont axiomatiquement équivalentes. En conséquence, il y a seulement trois analogies non équivalentes possibles entre quatre objets, ayant les formes canoniques :

$$A : B :: C : D \qquad B : A :: C : D \qquad C : B :: A : D$$

Dans la suite de ce chapitre, nous utiliserons parfois par abus de langage le terme *analogie* à la place de *proportion analogique*.

Une fois la définition posée, il est intéressant de la prolonger par la notion suivante : *résoudre une équation analogique*, c'est-à-dire trouver le quatrième terme d'une proportion analogique quand on connaît les trois premiers termes.

Définition 6.2 (Équation analogique)

D est solution de l'équation analogique en x : $A : B :: C : x$ *ssi* $A : B :: C : D$.

Selon la nature des objets et la définition de l'analogie, une équation analogique peut ne pas avoir de solution, avoir une seule solution ou plusieurs solutions. Par exemple, sous l'interprétation de la proportion analogique par multiplication, 9 est la seule solution dans $\mathbb{R}$ de l'équation $4 : 2 :: 18 : x$. Mais quelles sont les solutions de $aabbb : ab :: aaabbbb : x$, et pourquoi ?

5.2 Une introduction à l'algèbre de la proportion analogique

Pour unifier la notion de proportion analogique afin de l'appliquer à différents espaces X, Yvon et Stroppa ont proposé en particulier dans [SY06] une définition générale qui vérifie les axiomes et aide à construire la résolution des équations analogiques. Présentons-la par l'exemple.

Dans $\mathbb{R}$, si l'on considère la multiplication comme l'opération de base pour l'analogie, nous savons déjà que

$$A : B :: C : D \quad \Leftrightarrow \quad \frac{A}{B} = \frac{C}{D}$$

Pour analyser la seconde relation, il est naturel de décomposer les quatre nombres en facteurs premiers. Par exemple : $\frac{6}{10} = \frac{21}{35}$ peut s'écrire, en notant explicitement la multiplication, comme : $\frac{2\times3}{2\times5} = \frac{7\times3}{7\times5}$. Autrement dit, les nombres $u = 6$, $v = 10$, $w = 21$ et $x = 35$ sont en proportion analogique parce qu'il existe quatre facteurs $f_1 = 2$, $f_2 = 7$, $f_3 = 3$ et $f_4 = 5$ tels que :

$$u = f_1 \times f_3,\ v = f_1 \times f_4,\ w = f_2 \times f_3,\ x = f_2 \times f_4$$

Pouvons-nous transposer cette factorisation croisée dans un autre univers ? Prenons l'ensemble Σ^* des séquences sur l'alphabet $\Sigma = \{a, b, c, d\}$, muni de l'opération non commutative de concaténation (notée explicitement « . »). L'analogie morphologique (pour dire plus simplement, la règle de conjugaison) *marchons* : *marcher* :: *chantons* : *chanter* peut aussi se factoriser comme $f_1.f_3 : f_1.f_4 :: f_2.f_3 : f_3.f_4$, avec $f_1 = march$, $f_2 = chant$, $f_3 = ons$, $f_4 = er$.

Pour poursuivre, examinons l'analogie numérique

$$18 : 63 :: 30 : 105$$

et l'analogie morphologique.

déridés : *ridons* :: *démarchés* : *marchons*

Copyright © 2010 Eyrolles.

Nous pouvons les factoriser de la manière suivante :

$$\begin{array}{rcccccccccc} 18 & = & 2 & \times & 3 & \times & 2 & \times & 1 & \times & 3 \\ 63 & = & 1 & \times & 3 & \times & 1 & \times & 7 & \times & 3 \\ 30 & = & 2 & \times & 5 & \times & 2 & \times & 1 & \times & 3 \\ 105 & = & 1 & \times & 5 & \times & 1 & \times & 7 & \times & 3 \end{array}$$

$$\begin{array}{rcccccccccc} \textit{déridés} & = & \textit{dé} & . & \textit{rid} & . & \textit{é} & . & \epsilon & . & \textit{s} \\ \textit{ridons} & = & \epsilon & . & \textit{rid} & . & \epsilon & . & \textit{on} & . & \textit{s} \\ \textit{démarchés} & = & \textit{dé} & . & \textit{march} & . & \textit{é} & . & \epsilon & . & \textit{s} \\ \textit{marchons} & = & \epsilon & . & \textit{march} & . & \epsilon & . & \textit{on} & . & \textit{s} \end{array}$$

Nous constatons que dans les deux cas, chaque quadruplet de facteurs de rang i, que l'on lit dans une colonne, est soit (f_i, f_i, g_i, g_i), soit (f_i, g_i, f_i, g_i). Un facteur est éventuellement l'élément neutre de l'ensemble considéré (1 pour la multiplication dans $\mathbb{R}$ et ϵ pour la concaténation dans Σ^*).

Nous ne développerons pas plus loin cette analyse algébrique ici. Yvon et Stroppa ont montré que, dans une grande variété de cas, une proportion analogique respectant les axiomes présentés plus haut peut être analysée par ce concept de quadruple factorisation, respectant la propriété évoquée sur les quatre facteurs. Leur étude traite des séquences, des valeurs binaires (interprétées comme des valeurs de vérité de la logique des propositions), des ensembles finis, des nombres réels (avec la multiplication ou l'addition), des arbres, etc. Un des avantages de cette approche est qu'elle permet dans tous les cas de donner des algorithmes fondés pour la résolution des équations analogiques.

Notons qu'on peut, dans le même esprit, définir par factorisation une relation d'analogie entre quatre distributions de probabilités gaussiennes, quatre transformations linéaires, quatre permutations, etc. L'exploration de la notion de proportion analogique n'est pas encore achevée.

Cette exploration a au moins deux utilités déjà en partie opérationnelles. La première est de permettre d'engendrer de manière rationnelle des objets nouveaux, comme des formes phonétiques dans |YVON99| ou des caractères manuscrits dans |BMD07|. Le principe est le suivant : M. A. est bilingue et enregistre des phrases en anglais et en japonais. M. B. ne parle que le japonais et enregistre les mêmes phrases. On peut fabriquer par résolution d'équation analogique les phrases que M. B aurait prononcées en anglais s'il avait su parler cette langue.

La seconde nous intéresse plus directement pour cet ouvrage : comment employer la notion de proportion analogique dans le cadre de l'apprentissage supervisé ?

5.3 Utilisation de la proportion analogique en apprentissage

Soit $\mathcal{S}$ un ensemble d'apprentissage $\mathcal{S} = \{(\mathbf{x}, u(\mathbf{x}))\}$, où x est la description d'un exemple ($\mathbf{x}$ peut être un vecteur binaire de $\mathbb{B}^d$, par exemple) et $u(\mathbf{x})$ son étiquette dans un ensemble fini. Ayant la description $\mathbf{y}$ d'un nouvel exemple, nous voudrions assigner une étiquette $u(\mathbf{y})$ à ce dernier à partir de la seule connaissance de $\mathcal{S}$.

Dans sa version de base, l'apprentissage par proportion analogique consiste à rechercher dans $\mathcal{S}$ un triplet $(\mathbf{x}^\star, \mathbf{z}^\star, \mathbf{t}^\star)$ tel que « $\mathbf{x}^\star$ *est à* $\mathbf{z}^\star$ *comme* $\mathbf{t}^\star$ *est à* $\mathbf{y}^\star$ » et prédire pour $\mathbf{y}$ l'étiquette $\hat{u}(\mathbf{y})$ qui est une solution de l'équation « $u(\mathbf{x}^\star)$ *est à* $u(\mathbf{z}^\star)$ *comme* $u(\mathbf{t}^\star)$ *est à* $\hat{u}(\mathbf{y})$ ». Si plus d'un tel triplet est trouvé, une procédure de vote entre classes peut être utilisée. Il est important de noter que $\mathbf{y}$ et $u(\mathbf{y})$ appartiennent à deux domaines différents : par exemple, dans le cas simple d'un apprentissage de règle de classification, $\mathbf{y}$ peut être un vecteur de $\mathbb{R}^n$ alors que u est une étiquette de classe.

Copyright © 2010 Eyrolles.

5.3.1 Un exemple

—— EXEMPLE ——

Soit *matou* l'objet à classer dans l'une des classes suivantes : *Félin* ou *Ruminant*. Supposons qu'il n'y ait que trois exemples dans l'ensemble d'apprentissage : *veau*, *taureau* et *chaton*. Les animaux sont décrits par quatre attributs binaires (1 pour $VRAI$ et 0 pour $FAUX$) qui sont *a des griffes (AG)*, *est un adulte (EA)*, *est un mâle (EM)*, *boit du lait (BL)*.

Animaux	*AG*	*EA*	*EM*	*BL*	classe
veau	0	0	0	1	*Ruminant*
taureau	0	1	1	0	*Ruminant*
chaton	1	0	0	1	*Félin*
matou	1	1	1	0	?

Pour chaque attribut, nous remarquons qu'il y a une relation d'analogie entre les trois objets de l'ensemble d'apprentissage et l'objet *matou*, car les quatre attributs binaires sont dans une des configurations analogiques suivantes :

0 est à 0 ce que 1 est à 1 (AG)
0 est à 1 ce que 0 est à 1 (EA et EM)
1 est à 0 ce que 1 est à 0 (BL)

Ainsi, les quatre objets dans cet ordre sont en analogie puisque tous leurs attributs binaires sont en analogie :

veau **est à** *taureau* **ce que** *chaton* **est à** *matou*

La classe de *matou* est alors la solution de l'équation analogique sur les classes :

Ruminant **est à** *Ruminant* **ce que** *Félin* **est à** *classe(matou)*

Ce qui nous donne : *classe(matou)* = *Félin*.

5.3.2 Dissemblance analogique et apprentissage

Pour assouplir la relation analogique entre quatre objets et aller plus loin dans l'application à l'apprentissage, il est intéressant de définir une proportion analogique *approximative*, que pourrait traduire l'expression linguistique A est à B *à peu près ce que* C est à D. Par exemple, 8 est à 2 *à peu près ce que* 21 est à 5, ou jument est à poulain *à peu près ce que* taureau est à veau.

Pour rester cohérent avec les définitions précédentes, il faut quantifier le terme « à peu près » par une certaine valeur réelle positive, égale à 0 quand l'analogie est exacte et croissante au fur et à mesure que les quatre objets sont de moins en moins en proportion (voir [MBD08] pour les détails). Il faut aussi que cette valeur, que l'on appelle « dissemblance analogique » (DA), ait des bonnes propriétés vis-à-vis de l'analogie. Il est souhaitable qu'elle soit symétrique, qu'elle reste inchangée lors de permutation des termes moyens de l'analogie. Si possible, il faut aussi qu'elle vérifie, comme une distance, une certaine inégalité triangulaire. Ces propriétés permettent alors de réaliser un algorithme d'apprentissage dont le principe est le suivant :

1. extraire d'un ensemble d'apprentissage tous les triplets d'objets qui sont à dissemblance analogique faible avec l'objet de classe inconnue (disons qu'ils sont en nombre n) ;
2. résoudre les n équations analogiques sur les classes de ces triplets ;
3. choisir la classe majoritaire parmi les n classes solutions.

En réalité, il se peut que le second point fournisse moins de n classes. Une équation comme

« ω_1 *est à* ω_2 *comme* ω_1 *est à* $\mathbf{x}$ »

Copyright © 2010 Eyrolles.

se résout facilement par $\mathbf{x} = \omega_2$, mais que faire de l'équation suivante ?

$$\text{« } \omega_1 \textit{ est à } \omega_2 \textit{ comme } \omega_3 \textit{ est à } \mathbf{x} \text{ »}$$

En l'absence d'information sur la nature des classes, on peut considérer qu'elle n'a pas de solution.

On peut voir cette technique comme une extension de la règle de classification par les k plus proches voisins (voir le chapitre 15), qui cherche dans $\mathcal{S}$ les k descriptions $\mathbf{x}^*$ qui minimisent une certaine distance à $\mathbf{y}$ et assigne à $\mathbf{y}$ l'étiquette majoritaire parmi les k.

6. Bilan

Cet ouvrage fait une place réduite aux apprentissages mettant en œuvre beaucoup de connaissances ou des théories du domaine. Ce n'est pas parce que les auteurs les considèrent comme négligeables. Au contraire, notre conviction est que l'apprentissage artificiel ne pourra faire l'économie de l'étude des apprentissages utilisant des connaissances. Cependant, pour des raisons à la fois historiques (en particulier l'irruption du connexionnisme dans les années quatre-vingts), scientifiques (il est tellement plus facile d'écrire des théorèmes dans des espaces numériques), institutionnelles (le *quantum* d'action en science est souvent celui d'un travail de thèse, or les systèmes à base de connaissances demandent des développements à beaucoup plus long terme) et économiques (il y a déjà fort à faire avec l'exploration préliminaire de bases de connaissances « nues » telles que les bases de données génomiques ou les bases de données des entreprises), l'étude des méthodes telles que l'apprentissage à partir d'explications, l'abstraction, l'analogie, etc. a fortement marqué le pas ces dernières années. On peut espérer que dans dix ans un ouvrage sur l'apprentissage y consacrera une bien plus large place.

Résumé

Dès que l'on aborde des domaines complexes, il faut faciliter les raisonnements et l'apprentissage :

- en contrôlant l'expression de l'espace des hypothèses, par exemple en réalisant des abstractions ;
- en apprenant des connaissances permettant une exploration plus efficace des hypothèses, par exemple à l'aide de macro-opérateurs ou d'heuristiques de contrôle que rend possible l'apprentissage à partir d'explications ;
- en facilitant le transfert de connaissances et de solutions entre domaines, par exemple en utilisant des raisonnements comme l'analogie.

Toutes ces techniques requièrent une ou des théories du domaine fortes. C'est de là qu'elles tirent une grande puissance en permettant l'apprentissage à partir de peu de données. C'est là aussi la source des difficultés de leur application.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 7

L'inférence grammaticale

Ce chapitre traite de l'apprentissage inductif de grammaires et d'automates à partir d'exemples. Le problème est d'extraire, à partir d'exemples (et éventuellement de contre-exemples), une grammaire capable d'engendrer un langage, c'est-à-dire un ensemble de séquences, en général infini.

On se place ici dans le cas d'un apprentissage de concept : l'ensemble d'exemples est donc formé de deux parties, un échantillon positif, sous-ensemble fini du langage de la grammaire que l'on cherche à apprendre, et un échantillon négatif (éventuellement vide), ensemble fini de séquences n'appartenant pas au langage.

La décision de l'appartenance d'une phrase inconnue à une grammaire est réalisée par un algorithme d'analyse syntaxique, dont le rôle est de rendre une réponse binaire à la question : telle séquence de symboles appartient-elle au langage engendré par telle grammaire ?

L'inférence grammaticale a été étudiée depuis le développement (simultané) de la théorie des grammaires formelles et de l'intelligence artificielle. Cette discipline offre un grand intérêt théorique aux chercheurs en apprentissage. De plus, elle est l'objet de nombreuses applications. Nous consacrerons principalement ce chapitre à l'inférence régulière, c'est-à-dire à l'apprentissage des automates finis. C'est en effet dans ce cadre qu'ont été menés la plupart des travaux jusqu'à ce jour. Nous exposerons ensuite quelques extensions, en particulier aux grammaires algébriques.

Copyright © 2010 Eyrolles.

Sommaire

1 Définitions et notations 225
1.1 Langages, grammaires, automates et partitions 225
1.2 Échantillons d'un langage et automates associés 231
2 Les protocoles de l'inférence : quelques résultats théoriques 233
2.1 La spécification d'un problème d'inférence grammaticale 233
2.2 L'identification à la limite d'une grammaire 234
2.3 Deux propriétés de l'identification à la limite. 235
2.4 Autres protocoles pour l'inférence de grammaires. 235
2.5 L'inférence grammaticale et l'apprentissage *PAC* 236
2.5.1 Apprentissage *PAC* 236
2.5.2 Comparaison de l'apprentissage *PAC* et de l'identification à la limite 237
2.6 Résultats *PAC* pour les langages réguliers 238
2.7 Apprentissage *PACS* : *PAC* Simple 238
2.8 Apprentissage *PAC* avec distributions bienveillantes 239
3 L'espace de recherche de l'inférence régulière 240
3.1 Le point de la situation 240
3.2 Deux propriétés fondamentales 240
3.3 La taille de l'espace de recherche 241
4 L'inférence régulière sans échantillon négatif 241
4.1 Une méthode caractérisable : l'inférence de langages k-réversibles . . . 242
4.2 Une méthode heuristique : l'algorithme ECGI 243
5 L'inférence régulière sous contrôle d'un échantillon négatif 246
5.1 L'ensemble frontière 246
5.2 Le lien avec l'espace des versions 247
5.3 Les algorithmes RIG et BRIC 247
5.4 L'algorithme RPNI 247
5.4.1 Une exploration efficace du treillis 247
5.5 Variantes et extensions 250
6 L'inférence de grammaires algébriques 250
6.1 Présentation 250
6.2 L'apprentissage à partir d'échantillons structurés 251
6.3 L'inférence de grammaires linéaires équilibrées 253
6.4 Une méthode par compression : SEQUITUR 253
6.5 Une méthode par exploration : l'algorithme GRIDS. 254
6.6 Une méthode empirique. 255
7 L'inférence d'automates probabilistes 257
7.1 Les automates probabilistes 257
7.2 Automates probabilistes et distributions de probabilités sur Σ^* 258
7.3 Automates probabilistes déterministes 258
7.4 Inférence d'automates probabilistes 259
8 Quelques approches complémentaires 260
8.1 Le point de vue connexionniste 260
8.2 L'inférence de systèmes de réécriture 260

Copyright © 2010 Eyrolles.

Le zoo vient d'acquérir deux cygnes, un noir et un blanc. Chaque jour, le gardien transporte de sa cabane à leur cage six rations de farines animales, trois pour chacun. Comme il a déjà à s'occuppper des ornithorynques et des rhinolophes, il décide de simplifier sa tâche : plutôt que de transporter les rations, il va essayer d'apprendre aux cygnes à venir se servir eux-mêmes. Qui plus est, cela fera une attraction dans l'établissement : à une heure dite, la porte s'ouvrira et les deux animaux quitteront leur cage pour se diriger vers leur repas. Avant de proposer le numéro, il faut dresser les oiseaux. Le gardien, qui a de bonnes lectures, décide de les conditionner de la manière suivante. Un jour sur quatre, il laisse l'un des oiseaux dans sa cage sans rien lui donner et emmène l'autre à sa cabane, où ce dernier a la bonne surprise de trouver non pas trois, mais cinq rations de pâtée. Un jour sur quatre, il fait la même chose en échangeant les oiseaux. Un jour sur quatre, il emmène les deux qui ont alors droit à deux rations et demi chacun. Le reste du temps, il leur apporte trois rations chacun dans leur cage, comme d'habitude. Quelques semaines de conditionnement, seul ou à deux, et l'expérience est prête. La porte s'ouvre. Que vont faire les cygnes ? Le tableau suivant montre le résumé de la situation : le cygne noir est noté CN et le blanc CB.

CN et CB se déplacent tous les deux	CN et CB ont chacun 2.5 rations
CN se déplace et CB ne se déplace pas	CN a 5 rations et CB n'a rien
CN ne se déplace pas et CB se déplace	CN n'a rien et CB a 5 rations
CN et CB ne se déplacent ni l'un ni l'autre	CN et CB ont chacun 3 rations

Chacun des cygnes a maintenant appris que s'il reste sur place, il aura peut-être trois rations, peut-être aucune, selon ce que fera l'autre. Il sait aussi que s'il fait seul l'effort de se déplacer, cinq rations seront peut-être pour lui. Cependant, si son camarade fait aussi le voyage, il n'y en aura que deux et demi pour chacun. La difficulté vient de ce qu'aucun des deux cygnes ne sait ce que va faire l'autre. Faut-il attendre et espérer que l'autre animal a fait de même pour recevoir les trois rations ? Faut-il essayer d'en avoir cinq, au risque que l'autre cherche aussi à manger plus, ce qui pénalise les deux ?

Le gardien a de bonnes raisons d'être optimiste. Chaque jour, pense-t-il, chaque cygne va raisonner comme suit : « je dois me déplacer, pour deux raisons. La première est que cela m'assure de manger au moins deux rations et demi. La seconde est que l'autre cygne va probablement penser comme moi et se déplacer aussi : il ne faut donc pas que je reste dans ma cage ». C'est en effet ce que l'on observe la plupart du temps : confrontés à des situations de ce type[1], deux sujets choisissent la sécurité. Le paradoxe apparent de la situation vient de ce que s'ils restaient tous deux dans leur cage, chacun aurait plus à manger. Mais comme ils n'ont pas la possibilité de se mettre d'accord, le résultat n'est pas finalement le meilleur pour eux.

Maintenant, nous allons voir comment des cygnes doués de capacités à vrai dire rares dans cette espèce peuvent mieux profiter de la situation. Nous allons émettre deux hypothèses. La première est que CB possède une stratégie qui permet chaque jour de prendre une décision en fonction de ce qui s'est passé les jours précédents. La seconde est que CN connaît la famille de la stratégie que CB applique, mais qu'il ne sait pas exactement laquelle. Il va donc l'apprendre au fil des jours. Une fois qu'il l'aura induite du comportement de son adversaire, il lui sera facile de la déjouer et de manger plus. C'est le gardien qui va être surpris.

Commençons par CB pour présenter dans quelle famille de stratégies il se place. Le mieux est de représenter cela graphiquement. La figure 7.1 présente l'exemple de la stratégie naturelle qui

[1] Ce problème est souvent présenté par une autre métaphore : *le dilemme des prisonniers [Axe97, Del92, Gir00]*.

Copyright © 2010 Eyrolles.

consiste à se déplacer (sortir de la cage) chaque jour quelle que soit l'action effectuée la veille par CN. C'est sur cette analyse que compte le gardien. Chaque jour, la décision consiste pour CB à se souvenir de ce que CN a fait la veille et à agir en conséquence. Pour cela, CB, qui s'est installé la veille dans un certain « état » (un cercle du diagramme) emprunte au jour j l'arc qui porte l'action effectuée la veille par son adversaire et se retrouve dans un nouvel « état ». Comme dans cette stratégie CB sort tous les jours, le diagramme n'a qu'un état. Chaque jour, CB s'y trouve, et quelle que soit l'action de CN la veille, CB va emprunter un des deux arcs pour prendre sa décision : évidemment celle de sortir.

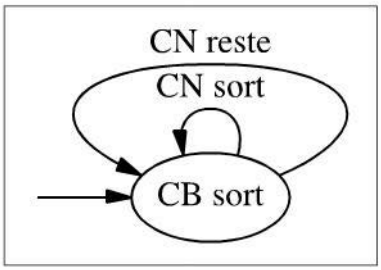


FIG. 7.1: *La stratégie triviale pour le cygne blanc CB : je sors tous les jours, quelle qu'ait été l'action de mon adversaire la veille.*

Mais le cygne blanc peut aussi vraiment tenir compte de ce que son partenaire CN a fait les jours précédents. En réalité, c'est la seule information observable[2]. Par exemple, il peut décider de rester tous les jours dans sa cage, sauf si la veille CN est sorti et de ne pas sortir deux jours de suite. L'hypothèse de CB est ici est que si CN est sorti un jour, il va rester tranquille le lendemain. Cette stratégie est décrite par la figure 7.2 et le tableau ci-dessous.

Ce qui s'est passé au jour $j-1$	La décision de CB au jour j
CB est resté et CN est sorti	CB sort
CB est resté et CN est resté	CB reste
CB est sorti et CN est sorti	CB reste
CB est sorti et CN est resté	CB reste

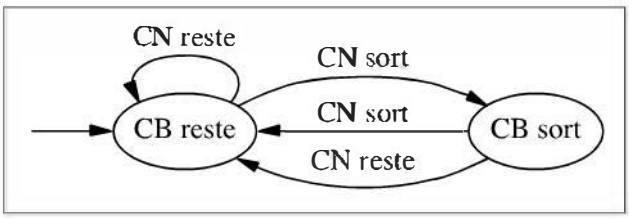


FIG. 7.2: *Une stratégie un peu plus complexe pour le cygne blanc CB : je reste tous les jours dans ma cage, sauf si mon adversaire est sorti hier. Dans ce cas, je pense qu'il va rester aujourd'hui et donc je sors.*

D'une manière générale, nous supposons donc que la stratégie du cygne blanc s'écrit sous la forme graphique dont nous avons déjà donné deux exemples. Ce genre de modèle s'appelle un

[2] Il s'agit ici du *dilemme itéré du prisonnier*.

Copyright © 2010 Eyrolles.

automate fini. Chaque automate fini, donc chaque stratégie, correspond à la description pour le cygne blanc du concept « aujourd'hui, je sors » et de sa négation « aujourd'hui, je reste », en fonction des couples d'actions commises par les deux animaux dans les jours précédents.

Supposons que la stratégie utilisée par CB soit celle de la figure 7.3. Comment CB utilise-t-il ce dessin pour agir en fonction de ce qu'a fait l'adversaire les jours précédents? Appelons « états » les trois nœuds du graphe : l'un est en forme de cercle, l'autre de carré et le dernier de losange. Dans chacun est inscrite la décision à prendre. Le premier jour, le cygne blanc place sa stratégie dans l'état cercle. Il va donc sortir vers la cabane du gardien et observer ce que fait le cygne noir. Si celui-ci se déplace, la stratégie du cygne blanc reste dans le même état : le lendemain, il se déplacera encore. Par contre, si son adversaire reste dans la cage, la stratégie passera dans l'état rectangle, ce qui impose que le lendemain, le cygne blanc restera dans sa cage.

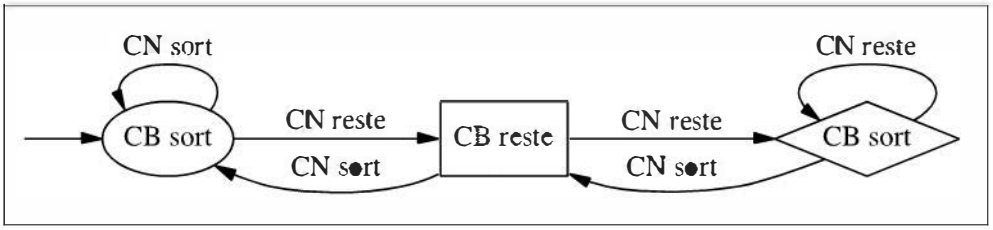


FIG. 7.3: *La stratégie du cygne blanc.*

jour	action de CB	état correspondant	action de CN
1	sortir	○	sortir
2	sortir	○	rester
3	rester	□	sortir
4	sortir	○	sortir
5	sortir	○	rester
6	rester	□	rester
7	sortir	◇	rester
8	sortir	◆	sortir
9	rester	□	

TAB. 7.1: *La séquence des actions journalières de CB selon sa stratégie, qui ne dépend que de ce que CN a fait les jours précédents.*

Sur une durée un peu plus longue, la séquence donnée dans le tableau 7.1 est conforme à la stratégie du cygne blanc. En effet, le neuvième jour, la séquence des actions effectuées par le cygne noir les huit jours précédents conduit le cygne blanc (en suivant les flèches marquées dans le graphe) dans l'état en forme de rectangle, qui est marqué CB reste. D'où la décision prise par CB : rester. Le lendemain, neuvième jour, quelle que soit l'action du cygne noir le huitième, le cygne blanc restera dans sa cage : il sera soit dans l'état en forme de cercle, soit dans celui en forme de losange, mais la décision sera la même.

Nous avons admis que le cygne blanc possédait une stratégie dans la famille de celles que l'on peut représenter par un automate fini, comme les trois présentées ci-dessus. On a vu que cela permet d'utiliser les actions passées de l'adversaire. Plaçons-nous maintenant du point de vue du cygne noir. Il n'a pas de stratégie *a priori*, mais il possède des facultés d'apprentissage qui vont

Copyright © 2010 Eyrolles.

lui permettre de déjouer celle de l'adversaire. Et comment ? D'abord, il va apprendre quelle est la stratégie du cygne blanc, quel automate il utilise, en observant ses réactions quotidiennes. Il n'aura plus ensuite qu'à prendre la meilleure tactique, déduite de celle de l'adversaire.

Le premier point est le sujet de ce chapitre, l'inférence grammaticale : est-il possible d'apprendre un concept représenté sous la forme d'un automate fini quand on connaît des exemples de séquences conformes au concept et d'autres non couvertes par ce concept ? La réponse est globalement positive, comme on le verra. Dans ce cas, l'ensemble d'apprentissage est constitué des huit exemples et contre-exemples donnés ci-dessous dans le tableau 7.2.

Séquence des actions de CN	Action de CB
(premier jour)	sortir
s	sortir
s r s	sortir
s r s s	sortir
s r	rester
s r s s r	rester
s r s s r r s	rester
s r s s r r r s	rester
s s r r s s s s	rester

TAB. 7.2: *L'ensemble des séquences d'apprentissage qui permet à CN d'apprendre la stratégie de CB. On a abrégé « CN sort » en* s *et « CN reste » en* r.

Le second point provient de la théorie des jeux (le cygne noir est particulièrement savant). Il se trouve que, si on a la connaissance d'une stratégie adverse sous forme d'automate fini, on peut en déduire la meilleure contre-stratégie. Sans détailler le cas général, la meilleure réponse du cygne noir à la stratégie du cygne blanc est ici la suivante : rester dans la cage deux fois, puis alterner : rester, sortir, rester, sortir... un jour sur deux. Un calcul simple montre que la quantité de rations que le cygne noir avale par jour tend rapidement vers 4, la moyenne entre 3 et 5. L'apprentissage de la stratégie de l'adversaire par inférence grammaticale lui a donc été profitable, puisque s'il avait quitté sa cage chaque jour, la stratégie de CB lui aurait dicté de faire de même et CN n'aurait eu que deux rations et demi par jour.

Le plan de ce chapitre

Nous présentons à la section 1 les notions de base qui nous permettent de définir l'espace des solutions possibles de l'inférence grammaticale régulière. Nous introduisons aussi la notion de *complétude structurelle* d'un échantillon relativement à un automate fini, un biais d'apprentissage qui assure d'une certaine manière que cet échantillon est représentatif de l'automate.

L'inférence grammaticale est l'un des domaines de l'apprentissage automatique symbolique pour lequel la caractérisation et l'évaluation d'une méthode ont été le plus formalisées. Nous présentons en particulier le critère d'*identification exacte* d'un langage, défini par Gold [Gol67, Gol78].

Nous consacrons la section 3 à une description formelle de l'inférence régulière. Elle est vue comme un problème de *recherche explicite* dans l'espace des hypothèses de généralisation de l'échantillon positif. Compte tenu d'une limitation raisonnable des possibilités de généralisation, l'espace de recherche à explorer est un ensemble d'automates structuré construit sur l'échantilon positif.

Copyright © 2010 Eyrolles.

De nombreux algorithmes d'inférence ont été proposés. Certains sont *constructifs* ou *caractérisables* au sens où ils sont capables d'identifier *avec certitude* n'importe quel langage appartenant à une *classe définie* (un sous-ensemble strict de langages réguliers défini de manière constructive). D'autres algorithmes sont qualifiés ici d'*heuristiques,* soit parce qu'ils n'identifient pas avec certitude n'importe quel langage d'une classe définie, soit parce qu'il n'existe pas de caractérisation de la classe de langages qu'ils peuvent identifier. Ces algorithmes utilisent un *biais d'apprentissage* heuristique pour guider la généralisation effectuée à partir des données.

Certains algorithmes d'inférence régulière n'utilisent que des échantillons positifs. La plupart de ces algorithmes sont décrits dans les articles de synthèse [BF72, Ang82, Mic90, Gre94, dlH05]. Nous présentons à la section 4 quelques algorithmes constructifs ou heuristiques qui utilisent ce type de données.

Si on dispose également d'un échantillon négatif, l'inférence peut être plus précise et on peut en particulier chercher à construire *le plus petit automate compatible (cohérent)* avec les échantillons positifs et négatifs. Nous étudions le cadre formel de ce problème à la section 5. Nous définirons pour cela la notion d'*ensemble frontière* (section 5.1). Nous présenterons ensuite quelques méthodes d'inférence.

La section suivante traitera plus brièvement du problème de l'inférence de grammaires algébriques, un problème pour le moment moins bien maîtrisé. On y verra quelques exemples d'algorithmes heuristiques. On traitera ensuite de l'apprentissage d'une classe de grammaires particulières, les automates probabilistes, qui ne sont pas sans rapports avec les modèles de Markov cachés (voir le chapitre 12). La dernière section abordera certaines extensions et problèmes ouverts de ce domaine, qui est à l'heure actuelle une véritable mine théorique et pratique pour les chercheurs.

Notations utiles pour le chapitre

Σ	Un alphabet (ensemble fini)
ϵ	Le mot vide
Σ^*	L'ensemble de tous les mots sur Σ
Σ^+	Σ^* sauf le mot vide ϵ
Σ^n	Tous les mots sur Σ de longueur inférieure ou égale à n

1. Définitions et notations

1.1 Langages, grammaires, automates et partitions

Notions de base

Définition 7.1

Soit Σ un ensemble fini, appelé alphabet *et les* lettres *$a, b, c, \ldots$ les éléments de Σ. On note u, v, w, x des éléments de Σ^*, c'est-à-dire des* chaînes *ou* phrases *(on dit aussi* mots *ou séquences) de longueur dénombrable sur Σ. On note ϵ la* chaîne vide *de longueur nulle et $|u|$ la longueur de la chaîne u.*

Définition 7.2

u est un préfixe *de v s'il existe w tel que $uw = v$.*

Copyright © 2010 Eyrolles.

Définition 7.3

Un langage L est un sous-ensemble quelconque de Σ^*. Les éléments de L sont des séquences de lettres de Σ, donc des chaînes.

Grammaires

Une *grammaire* est un objet mathématique auquel est associé un processus algorithmique permettant d'engendrer un langage.

Définition 7.4

Une grammaire *est un quadruplet $G = (N, \Sigma, P, S)$ dans lequel :*

- *N est un alphabet composant l'ensemble des* symboles non terminaux *de G.*
- *Σ est l'*alphabet terminal *de G, disjoint de N. On note : $V = N \cup \Sigma$.*
- *$R \subseteq (V^*N^+V^* \times V^*)$ est un ensemble fini de* règles de production.
- *$S \in N$ est l'*axiome *de G.*

Une règle de production s'écrit $\alpha \longrightarrow \beta$, avec $\beta \in V^*$ et $\alpha \in V^*N^+V^*$, ce qui signifie que α comporte au moins un symbole non terminal.

Génération d'un langage par une grammaire

Soit $u \in V^+$ et $v \in V^*$. On dit que u *se réécrit en* v selon la grammaire G, ou que la grammaire G *dérive* v de u *en une étape* si et seulement si on peut écrire u et v sous la forme :

- $u = xu'y$ (avec éventuellement $x = \epsilon$ ou $y = \epsilon$)
- $v = xv'y$
- *avec :* $(u' \longrightarrow v') \in P$

La grammaire G *dérive* v *en k étapes* de u si et seulement s'il existe $k \geq 1$ et une suite $(v_0 \dots v_k)$ de mots de V^+ tels que :

- $u = v_0$
- $v = v_k$
- v_i se récrit en v_{i+1} pour $0 \leq i \leq k-1$

La grammaire G *dérive* v de u s'il existe un entier k tel que u dérive v en k étapes. Pour $k \geq 1$, la séquence $u, \dots, v_i, \dots v$ s'appelle une *dérivation* de v par u.

Définition 7.5

On dit qu'un mot $v \in \Sigma^*$ est engendré *par la grammaire G quand il peut se dériver à partir de l'axiome S de G.*

Le langage engendré par la grammaire G *est l'ensemble des mots de Σ^* engendrés par G. On le note $L(G)$.*

Deux exemples

Quand il n'y a pas d'ambiguïté, on simplifie la notation d'une grammaire en ne représentant que ses règles et en mettant toutes celles de même partie gauche sur la même ligne.

EXEMPLE **Exemple 1**

La grammaire définie par $N = \{S\}$, $\Sigma = \{a, b, c\}$ et $P = \{(S \longrightarrow aSb), (S \longrightarrow \epsilon)\}$ peut ainsi s'écrire :

$$S \longrightarrow aSb \mid \epsilon$$

Copyright © 2010 Eyrolles.

Elle engendre le langage $\{a^n b^n \mid n \geq 0\}$. En effet, sur un exemple, son axiome S dérive le mot $aaabbb$ en quatre étapes : trois applications de la règle $S \longrightarrow aSb$ et une de la règle $S \longrightarrow \epsilon$, soit :

$$S \ , aSb, aaSbb, aaaSbbb, aaabbb$$

EXEMPLE **Exemple 2**

La grammaire définie par $N = \{1,2,3,4\}$, $\Sigma = \{a,b\}$ et $P = \{(1 \longrightarrow a2),(1 \longrightarrow b3),(1 \longrightarrow \epsilon),(2 \longrightarrow a1),(2 \longrightarrow b4),(3 \longrightarrow a4),(3 \longrightarrow b1),(4 \longrightarrow a3),(4 \longrightarrow b2)\}$ est exactement équivalente à l'automate fini de la figure 7.4. Les phrases qu'elle accepte sont composées d'un nombre pair de a et pair de b. Elle s'écrit plus simplement[3] :

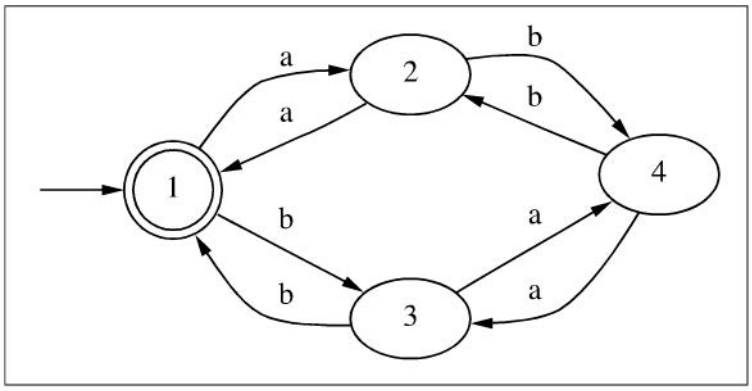


FIG. 7.4: *Un automate fini acceptant les phrases composées d'un nombre pair de a et pair de b.*

$$1 \longrightarrow a2 \mid b3 \mid \epsilon \quad 2 \longrightarrow a1 \mid b4$$
$$3 \longrightarrow a4 \mid b1 \quad 4 \longrightarrow a3 \mid b2$$

Deux types de grammaires

Selon la forme de leur règles $\alpha \longrightarrow \beta$, on distingue les grammaires suivantes :

Type 2 : grammaires algébriques[4] : $A \longrightarrow \beta$, avec $A \in N$ et $\beta \in V^*$

Type 3 : grammaires régulières : $A \longrightarrow wB$ ou $A \longrightarrow w$, avec $w \in \Sigma^*$, $A \in N$ et $B \in N$

Un langage pouvant être engendré par une grammaire régulière (resp. algébrique) est appelé *langage régulier* (resp. *langage algébrique*). Un résultat classique de la théorie des langages est le suivant [AU72] :

Théorème 7.1

Tout langage régulier peut être engendré par un automate fini. Tout automate fini engendre un langage régulier.

Les automates finis

Comme l'assure le théorème précédent, les automates finis sont équivalents aux grammaires régulières. Ils sont d'un emploi beaucoup plus facile. Nous allons maintenant les définir rigoureusement.

[3] L'axiome n'est pas noté ici S, mais 1.

[4] Les grammaires algébriques sont aussi appelées *context-free* ou *hors-contexte*.

Copyright © 2010 Eyrolles.

Définition 7.6 (Automates finis)

Un automate fini *est un quintuplet* $(Q, \Sigma, \delta, q_0, F)$ *où* Q *est un ensemble fini d'états,* Σ *est un alphabet fini,* δ *est une fonction de transition, c'est-à-dire une application de* $Q \times \Sigma \rightarrow 2^Q$, $Q_0 \in Q$ *est le sous-ensemble des* états initiaux *et* $F \in Q$ *est le sous-ensemble des états finaux ou d'acceptation.*

Définition 7.7 (Automates déterministes)

Si, pour tout q *de* Q *et tout* a *de* Σ, $\delta(q, a)$ *contient au plus un élément (respectivement exactement un élément) et si* Q_0 *ne possède qu'un élément* q_0, *l'automate* A *est dit* déterministe *(respectivement* complet*). Par la suite, nous utiliserons l'abréviation AFD pour « automate fini déterministe » et AFN pour un « automate fini non déterministe ».*

L'automate fini de la figure 7.4 est un AFD. Un autre exemple d'automate fini est représenté à la figure 7.5. Il comporte cinq états, $Q = \{0, 1, 2, 3, 4\}$. Il est défini sur l'alphabet à deux lettres, $\Sigma = \{a, b\}$. Les états initiaux sont 0 et 5, $Q_0 = \{0, 5\}$ et les états 3 et 4 sont finaux, $F = \{3, 4\}$.

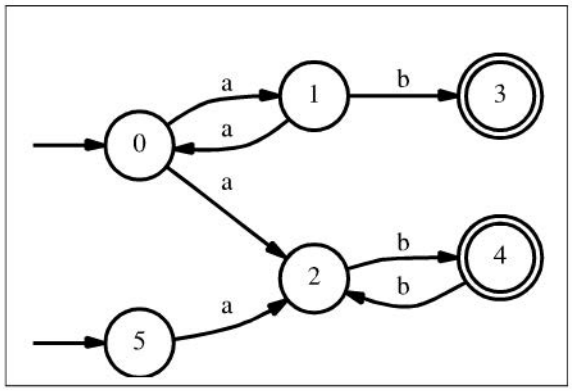


FIG. 7.5: *Une représentation de l'automate* A_1.

Il s'agit ici d'un AFN (automate non déterministe) car il y a deux arcs étiquetés a à partir de l'état 0 et deux états initiaux. En d'autres termes, $\delta(0, a) = \{1, 2\}$.

Langage accepté par un automate fini

Définition 7.8 (Acceptation d'une chaîne par un automate)

Une acceptation *d'une chaîne* $u = a_1 \ldots a_l$ *par un automate* A, *éventuellement non déterministe, définit une séquence, éventuellement non unique, de* $l + 1$ *états* $(q^0, \ldots, q^l)$ *telle que* $q^0 \in Q_0$, $q^l \in F$ *et* $q^{i+1} \in \delta(q^i, a_{i+1})$, *pour* $0 \leq i \leq l - 1$. *Les* $l + 1$ *états sont dits* atteints *pour cette acceptation et l'état* q^l *est* utilisé comme état d'acceptation. *De façon similaire, les* l *transitions, c'est-à-dire des éléments de* δ, *sont dites* exercées *par cette acceptation.*

Par exemple, la séquence des états $(0, 1, 3)$ correspond à une acceptation de la chaîne aab dans l'automate A_1.

Définition 7.9 (Langage accepté)

Le langage $L(A)$ *accepté par un automate* A *est l'ensemble des chaînes acceptées par* A.

Copyright © 2010 Eyrolles.

Théorème 7.2

Le langage L est accepté par un automate A si et seulement si il peut être engendré par une grammaire régulière. De plus, un langage régulier peut être représenté par une expression régulière *[AU72].*

On peut remarquer que l'automate de la figure 7.5 n'est pas *complet*, au sens où la fonction δ n'est pas définie partout : par exemple, on ne connaît pas $\delta(5, b)$, puisqu'aucune transition portant la lettre b ne part de l'état 5. On peut compléter un automate en lui ajoutant un *état-puits*, non final, où mènent toutes les transitions non définies et qui boucle sur lui-même pour toutes les lettres de l'alphabet.

Un état-puits est un exemple d'état non *co-accessible*, ce qui signifie qu'on ne peut accéder à aucun état final à partir de lui. De même, certains états peuvent être non *accessibles*, au sens où on ne peut pas y accéder à partir d'un état initial. Les état à la fois accessibles et co-accessibles sont dits *utiles*. On peut éliminer les états inutiles d'un automate sans changer le langage qu'il accepte. Un automate sans états inutiles est dit *émondé* [Sak03].

Définition 7.10 (Automate canonique)

Pour tout langage régulier L, il existe un AFD noté $A(L)$ qui engendre L et possède un nombre minimal d'états. $A(L)$ est généralement appelé automate déterministe minimal *ou* automate canonique *de L. On démontre que $A(L)$ est unique [AU72]. Par la suite, nous parlerons indifféremment de l'automate canonique ou de l'automate minimal déterministe de L.*

Par exemple, l'automate canonique représenté à la figure 7.6 accepte le langage composé des phrases commençant par un nombre impair de a, suivi d'un nombre impair de b [5]. Il s'agit du langage accepté également par l'automate A_1 de la figure 7.5. Il n'existe pas d'automate déterministe comportant moins d'états et acceptant ce langage. Cet automate est émondé, mais il serait facile de le compléter en lui ajoutant un état-puits.

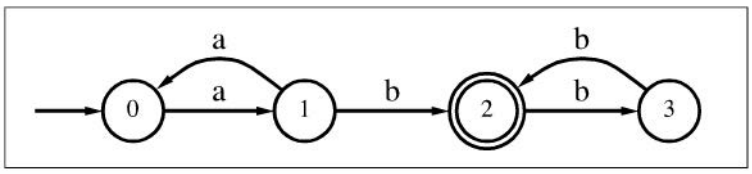


FIG. 7.6: *Automate canonique du langage défini par l'expression régulière $L = a(aa)^*b(bb)^*$.*

Automates dérivés

Nous définissons maintenant une relation d'ordre partiel sur l'ensemble des automates, qui permettra un apprentissage par généralisation dans l'esprit de la méthode de l'espace des versions.

Définition 7.11 (Partition et partition plus fine qu'une autre)

Pour tout ensemble S, une partition π est un ensemble de sous-ensembles de S, non vides et disjoints deux à deux, dont l'union est S. Si s désigne un élément de S, $B(s, \pi)$ désigne l'unique élément, ou bloc, *de π comprenant s. Une partition π_i* raffine, *ou* est plus fine que, *une partition π_j ssi tout bloc de π_j est un bloc de π_i ou est l'union de plusieurs blocs de π_i.*

[5] ce qui correspond à l'expression régulière $L = a(aa)^*b(bb)^*$.

Copyright © 2010 Eyrolles.

Définition 7.12 (Automate quotient)

Si $A = (Q, \Sigma, \delta, q_0, F)$ *est un automate, l'automate* $A/\pi = (Q', \Sigma, \delta', B(q_0, \pi), F')$ dérivé de A relativement à la partition π *de* Q, *aussi appelé* l'automate quotient A/π, *est défini comme suit :*

$Q' = Q/\pi = \{B(q, \pi) | q \in Q\}$,

$F' = \{B \in Q' | B \cap F \neq \phi\}$,

$\delta' : Q' \times \Sigma \rightarrow 2^{Q'} : \forall B, B' \in Q', \forall a \in \Sigma, B' \in \delta'(B, a)$ *ssi* $\exists q, q' \in Q, q \in B, q' \in B'$ *et* $q' \in \delta(q, a)$

Nous dirons que les états de Q *appartenant au même bloc* B *de la partition* π *sont* fusionnés.

Reprenons l'automate A_1, représenté à la figure 7.5 et définissons la partition de son ensemble d'états, $\pi_2 = \{\{0,1,5\}, \{2\}, \{3,4\}\}$. L'automate quotient A_1/π_2, obtenu en fusionnant tous les états appartenant à un même bloc de π_2, est représenté à la figure 7.7.

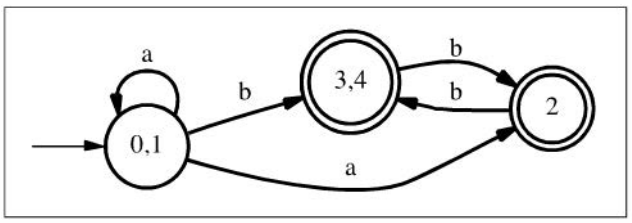


FIG. 7.7: *L'automate quotient* A_1/π_2.

Une propriété fondamentale de l'opération de fusion est la suivante :

Propriété 7.1

Si un automate A/π_j *dérive d'un automate* A/π_i, *alors le langage accepté par* A/π_i *est inclus dans celui accepté par* A/π_j.

Par conséquent, en partant d'un automate A, on peut construire tous les automates dérivés de A en énumérant les partitions des états de A. Il existe sur cet ensemble une relation d'ordre partiel qui est cohérente avec l'inclusion des langages qu'acceptent ces automates. On reconnaît ici la relation de généralité de l'espace des versions (voir le chapitre 4).

EXEMPLE

En reprenant l'exemple A_1 de la figure 7.5, on a vu que le choix de la partition $\pi_2 = \{\{0,1,5\}, \{2\}, \{3,4\}\}$ permet de dériver l'automate quotient $A_2 = A_1/\pi_2$, représenté à la figure 7.7. On sait donc que le langage reconnu par A_2 inclut celui reconnu par A_1. Prenons maintenant la partition $\pi_3 = \{\{0,1,2,5\}, \{3,4\}\}$. Elle permet de dériver un automate A_3, représenté à la figure 7.8.

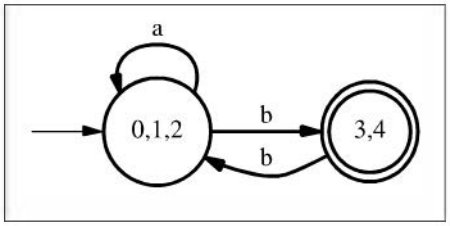


FIG. 7.8: *L'automate quotient* A_1/π_3.

La partition π_3 est *moins fine* que π_2, puisque ses blocs sont construits comme une union de

Copyright © 2010 Eyrolles.

blocs de π_2 ; on peut donc assurer que l'automate A_3 reconnaît un langage qui inclut celui reconnu par A_2. En revanche, si on construit l'automate A_4 (figure 7.9) par dérivation de A_1 selon la partition $\pi_4 = \{\{0,5\},\{1,3\},\{2,4\}\}$, qui n'est ni moins fine ni plus fine que π_2, on ne peut rien dire sur l'inclusion éventuelle des langages reconnus par A_4 et A_2. Par exemple, la phrase abb est reconnue par A_4 et pas par A_2, alors que la phrase b est reconnue par A_2 et pas par A_4.

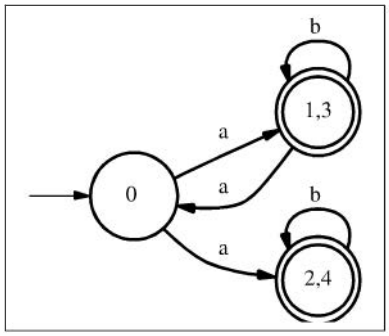


FIG. 7.9: *L'automate quotient A_1/π_4.*

Propriété 7.2

L'ensemble des automates dérivés d'un automate A, qui est partiellement ordonné par la relation de dérivation, est un treillis. *Les automates A et UA, l'automate universel, en sont respectivement les éléments minimal et maximal. On note ce treillis $Lat(A)$.*

On retrouve donc ici la structure en treillis de l'espace des hypothèses, associée à une relation d'ordre de généralité, comme définie au chapitre 4.

1.2 Échantillons d'un langage et automates associés

En inférence grammaticale, les notations I_+ et I_- sont classiques. Nous les employons donc dans ce chapitre à la place de $\mathcal{S}_+$ et $\mathcal{S}_-$

Définition 7.13

Nous désignons par I_+ un sous-ensemble fini, appelé échantillon positif, *d'un langage L quelconque. Nous désignons par I_- un sous-ensemble fini, appelé* échantillon négatif, *du langage complémentaire $\Sigma^* - L$. I_+ et I_- sont des sous-ensembles finis et disjoints de Σ^*.*

Complétude structurelle

On définit maintenant un biais d'apprentissage : on ne cherchera à inférer que des automates pour lesquels l'échantillon positif est d'une certaine manière représentatif (en termes techniques, *structurellement complet*). Ce biais est conforme à un principe d'économie de type « rasoir d'Occam » ou principe MDL : il est inutile de rechercher des automates trop complexes.

Définition 7.14

Un échantillon I_+ est structurellement complet *relativement à un automate A acceptant L, s'il existe une acceptation $\mathcal{AC}(I_+, A)$ de I_+ telle que :*

- *Toute transition de A soit exercée.*
- *Tout élément de F (l'ensemble des états finaux de A) soit utilisé comme état d'acceptation.*

Copyright © 2010 Eyrolles.

Par exemple, l'échantillon $I_+ = \{aaab, ab, abbbbb\}$ est structurellement complet relativement à l'automate A_1.

L'automate canonique maximal, l'arbre accepteur des préfixes et l'automate universel

On a maintenant besoin de représenter l'échantillon d'apprentissage dans l'univers des concepts, c'est-à-dire de définir des automates qui ne reconnaissent que cet échantillon.

Définition 7.15 (Automate canonique maximal)

On désigne par $MCA(I_+) = (Q, \Sigma, \delta, q_0, F)$ *l'*automate maximal canonique relatif à I_+ *[Mic80].*

L'abréviation **MCA** pour *Maximal Canonical Automaton* provient de la terminologie anglaise. Le qualificatif *canonique* se rapporte ici à un échantillon. Le MCA ne doit pas être confondu avec l'automate canonique d'un langage (voir déf. 7.10).

Par construction, $L(MCA(I_+)) = I_+$ et $MCA(I_+)$ est le plus grand automate (l'automate ayant le plus grand nombre d'états) pour lequel I_+ est structurellement complet. $MCA(I_+)$ est généralement non déterministe.

Sa construction est facile à observer sur un exemple. L'automate représenté à la figure 7.10 est l'automate maximal[6] canonique relatif à l'échantillon $I_+ = \{a, ab, bab\}$.

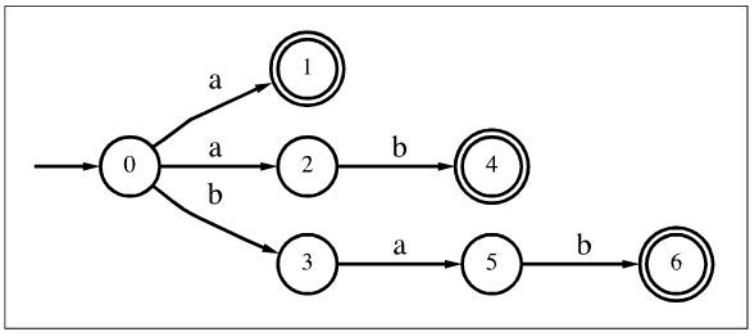


FIG. 7.10: *L'automate maximal canonique de l'échantillon* $\{a, ab, bab\}$.

Définition 7.16 (Arbre accepteur des préfixes)

Nous désignons par $PTA(I_+)$ *l'*arbre accepteur des préfixes *de* I_+ *[Ang82]. Il s'agit de l'automate quotient* $MCA(I_+)/\pi_{I_+}$ *où la partition* π_{I_+} *est définie comme suit :*

$$B(q, \pi_{I_+}) = B(q', \pi_{I_+}) \text{ ssi } Pr(q) = Pr(q').$$

L'abréviation PTA pour *Prefix Tree Acceptor* provient de la terminologie anglaise.

En d'autres termes, $PTA(I_+)$ peut être obtenu à partir du $MCA(I_+)$ en fusionnant les états partageant les mêmes préfixes. $PTA(I_+)$ est déterministe.

À titre d'exemple, l'automate représenté à la figure 7.11 est l'arbre accepteur des préfixes relatif à l'échantillon $I_+ = \{a, ab, bab\}$.

[6] Rigoureusement parlant, on pourrait construire un automate canonique ayant encore plus d'états en « éclatant » l'état initial en trois états distincts.

Copyright © 2010 Eyrolles.

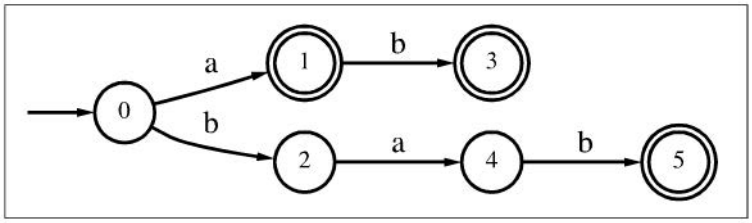


FIG. 7.11: *L'arbre accepteur des préfixes $PTA(I_+)$, avec $I_+ = \{a, ab, bab\}$.*

Définition 7.17 (Automate universel)

Nous désignons par UA l'automate universel. *Il accepte toutes les chaînes définies sur l'alphabet Σ, c'est-à-dire $L(UA) = \Sigma^*$. Il s'agit donc du plus petit automate pour lequel tout échantillon de Σ^* est structurellement complet.*

L'abréviation UA pour *Universal Automaton* provient de la terminologie anglaise.

L'automate universel défini sur l'alphabet $\Sigma = \{a,b\}$ est représenté à la figure 7.12.

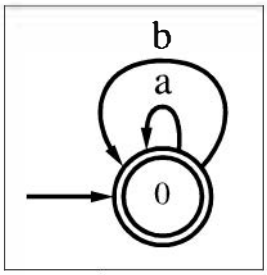


FIG. 7.12: *L'automate universel sur l'alphabet $\Sigma = \{a, b\}$.*

2. Les protocoles de l'inférence : quelques résultats théoriques

2.1 La spécification d'un problème d'inférence grammaticale

Si l'on veut définir un cadre aussi général et productif que possible pour l'inférence grammaticale, comme d'ailleurs pour tout problème d'apprentissage inductif, cinq points doivent être spécifiés :

- La *classe de grammaires* à inférer : grammaires *régulières*[7] ou l'une de ses sous-classes, ou grammaires hors-contexte. Comme on le verra, quelques autres familles de grammaires ont aussi été étudiées.
- L'*espace des hypothèses*, ou le *langage des concepts*, c'est-à-dire l'ensemble des descriptions tel que chaque grammaire de la classe considérée possède au moins une description dans cet espace. Pour les grammaires régulières, il s'agit en général de l'espace des *automates*. Pour

[7] Rappelons qu'une grammaire régulière, équivalente à un automate fini, engendre un langage régulier (voir section 1).

Copyright © 2010 Eyrolles.

les grammaires hors-contexte, l'espace de représentation est en général directement celui de leurs règles.

- Un *ensemble d'exemples,* pour chaque grammaire à inférer et un *protocole* de présentation de ces exemples. Ce terme désigne la façon dont les exemples sont présentés à l'algorithme d'apprentissage : tous à la fois ou un par un. Il se peut aussi que la présentation des exemples soit plus informative, quand on dispose d'un *oracle* (voir le paragraphe 2.4).
- La *classe des méthodes d'inférence* en considération. Il s'agit en particulier du choix entre méthodes constructives ou heuristiques (voir section 1).
- Les *critères* d'une inférence réussie, définis en section 2.2.

Une *méthode d'inférence* est alors un processus calculable qui lit des exemples conformément à un protocole et propose des solutions dans l'espace des hypothèses.

2.2 L'identification à la limite d'une grammaire

Pour illustrer les notions ci-dessus et présenter un cadre théorique de référence, nous présentons dans ce paragraphe un protocole de présentation et un critère de réussite définis conjointement : l'*identification à la limite*, proposée par Gold [Gol67]. De nombreux résultats découlent de ce critère, tant sur la possibilité d'identifier ou non un langage que sur la complexité calculatoire de ce processus d'identification [Gol78, Pit89]. En outre, ces travaux constituent une justification théorique de la nécessité d'utiliser un échantillon négatif.

Définition 7.18 (présentation positive)

Une présentation positive *d'un langage L est une séquence infinie de phrases de L comportant au moins une occurrence de chaque élément de L. Les éléments d'une* présentation positive *sont appelés des* exemples.

Définition 7.19 (présentation négative)

Une présentation négative *d'un langage L est une séquence infinie d'éléments de $\Sigma^* - L$ comportant au moins une occurrence de chaque élément de $\Sigma^* - L$. Les éléments d'une présentation négative sont appelés* exemples négatifs *ou* contre-exemples.

Définition 7.20 (présentation complète)

Une présentation complète d'un langage L est une séquence infinie de paires (x, d), de $\Sigma^* \times \{0, 1\}$ telle que chaque x de L apparaît dans au moins une paire. Pour chaque (x, d), $d = 1$ si et seulement si x appartient à L, $d = 0$ sinon.

Le protocole d'identification à la limite est défini comme suit. Soit M une méthode d'inférence et $S = (x_1, d_1).(x_2, d_2)\ldots$ une présentation complète d'exemples d'un langage L. Nous notons S_i le sous-ensemble fini des i premiers éléments de S selon un certain protocole. La méthode d'inférence doit proposer une solution dans l'espace d'hypothèses après chaque nouvel exemple lu. Soit $h(S_i)$, l'hypothèse proposée par la méthode M après avoir lu les i premiers exemples présentés.

Copyright © 2010 Eyrolles.

Définition 7.21

La méthode d'inférence M identifie à la limite pour la classe de langages $\mathcal{L}$ pour une classe de présentation $\mathcal{P}$ si pour tout L de $\mathcal{L}$ et tout S de $\mathcal{P}$, il existe un indice fini j, tel que :

- $h(S_i) = h(S_j)$, *pour tout* $i \geq j$.
- $L(h(S_j)) = L$.

La première condition impose que la méthode converge à partir d'un indice fini, la seconde qu'elle converge vers une représentation unique de la solution correcte.

Définition 7.22

Une méthode d'inférence M identifie à la limite une classe $\mathcal{C}$ de langages, si M identifie à la limite n'importe quel langage L de la classe $\mathcal{C}$ pour n'importe quelle présentation d'exemples de L.

Ce protocole s'intéresse donc à l'*identification exacte* d'un langage.

2.3 Deux propriétés de l'identification à la limite.

Propriété 7.3

La classe des langages réguliers[8] *n'est pas identifiable à la limite à partir d'une présentation positive.*

Propriété 7.4

La classe des langages réguliers et celle des langages algébriques[9] *sont identifiables à la limite par présentation complète.*

2.4 Autres protocoles pour l'inférence de grammaires.

La richesse théorique de l'apprentissage des grammaires formelles se traduit en particulier par la variété des protocoles proposés et les résultats démontrés dans ces protocoles. Citons par exemple la technique *par oracle*, dont une variante est la suivante : à chaque étape, l'algorithme d'inférence calcule une grammaire hypothèse et la propose à l'oracle. Celui-ci répond « oui » si l'hypothèse est la bonne, et sinon, donne éventuellement une correction à l'hypothèse proposée, ce qui permet à l'algorithme de calculer une nouvelle hypothèse. L'algorithme peut aussi proposer une séquence et l'oracle lui répond simplement si oui ou non elle appartient au langage à identifier. Il a été démontré que dans ce cadre, l'identification des automates déterministes est possible avec un nombre de questions à l'oracle polynomial en fonction de la taille de l'automate à trouver et de la taille du plus long contre-exemple.

D'un point de vue pratique, on doit en général admettre que les protocoles de l'identification à la limite ou de l'intervention d'un oracle sont inutilisables. On se contente d'observer des exemples et de calculer une grammaire qui les généralise. La validité de l'inférence peut être bien sûr vérifiée par un ensemble de test.

Il existe par exemple un serveur Internet nommé `gowachin` [LP97] qui fonctionne ainsi : l'auteur d'un algorithme d'inférence régulière demande à ce serveur une session de test. Quelques paramètres sont précisés : nombre d'états de l'automate canonique, taille de l'alphabet, « densité » de l'échantillon. L'automate est fabriqué aléatoirement par le serveur, ainsi que deux échantillons

[8] Le résultat de Gold est plus général, mais cette version suffira ici.
[9] Ici encore, Gold a un résultat plus puissant.

Copyright © 2010 Eyrolles.

d'apprentissage et de test. Le premier est donné à l'utilisateur, qui fait tourner son algorithme et renvoie une grammaire en réponse. Le serveur l'utilise pour analyser l'échantillon de test et retourne la matrice de confusion (voir au chapitre 3 le paragraphe 4). Il est à noter que la validité statistique de ce test n'est pas assurée ici, car on ne peut pas savoir si les séquences engendrées aléatoirement sont indépendantes.

On connaît un théorème (décevant) sur la recherche de l'automate fini « optimal », l'automate fini déterministe minimal $A(L)$, que nous avons aussi appelé *automate canonique* d'un langage L(7.10). Ce problème est connu sous le nom de *recherche du plus petit automate compatible*[10].

Théorème 7.3

Étant donné un échantillon $I_+ \cup I_-$ d'exemples positifs et négatifs, trouver le plus petit automate compatible avec I est un problème NP-difficile.

C'est-à-dire qu'il n'existe vraisemblablement pas d'algorithme qui puisse résoudre ce problème en un temps polynomial en fonction du nombre d'états de $A(L)$. Mais cela n'empêche pas de chercher des heuristiques qui permettent de trouver en un temps raisonnable un automate de bonne qualité (au sens du protocole gowachin, par exemple).

2.5 L'inférence grammaticale et l'apprentissage *PAC*

Nous avons vu dans la section précédente les résultats négatifs concernant l'identification à la limite proposée par Gold. Nous allons présenter d'autres résultats dans le cadre *PAC* (voir le chapitre 2). Bien que très contraignant[11], ce cadre est intéressant car il possède des versions affaiblies dans lesquelles l'approximation des langages réguliers est possible polynomialement par présentation complète.

2.5.1 Apprentissage *PAC*

Le cadre *PAC* original proposé par Valiant [Val84] a subi de nombreuses adaptations et extensions depuis sa création [BEHW89, Hau88, KV89, Pit89, KV89]. Nous présentons ici la version proposée par Pitt [Pit89] qui est adaptée à l'inférence de langages réguliers (plus particulièrement, dans cette version, Pitt considère que les langages réguliers sont représentés par des automates finis déterministes (DFA)).

Nous rappellons ici une description informelle du modèle *PAC*, puis nous en donnons une version adaptée à l'inférence grammaticale. Rappelons que ce concept a été décrit au chapitre 2.

Soit L un langage défini sur Σ^*. Soit h un concept appris sur L. On dit que h est correct[12] pour L si, pour tout mot u de Σ^*, h arrive à dire si u appartient ou non à L. On dira que h est approximativement correct si la probabilité qu'il classe mal un mot est « petite ».

Un algorithme d'apprentissage est correct (respectivement approximativement correct) si pour tout langage L défini sur Σ^*, le concept qu'il apprend est correct (respectivement approximativement correct) pour L.

On dit qu'un algorithme d'apprentissage est probablement approximativement correct si la probabilité qu'il apprenne un concept approximativement correct est grande.

[10] *Compatible* signifie simplement que toutes les phrases de I_+ sont acceptées (donc que l'automate est correct et complet (voir au chapitre 3) et toutes celles de I_- sont rejetées. C'est un synonyme de « cohérent ».

[11] Ce cadre est contraignant dans le sens où il impose des résultats quelle que soit la distribution de probabilités selon laquelle les exemples sont distribués.

[12] Dans les autres chapitres, en particulier 2 3 4, le mot « cohérent » a été employé. Nous gardons ici provisoirement « correct » pour faire la transition avec l'apprentissage *PAC*.

Copyright © 2010 Eyrolles.

Valiant a formalisé cette notion intuitive en créant le critère *PAC* : *probablement approximativement correct*. Cette première version ne prend pas en compte la complexité temporelle et spatiale nécessaire à l'inférence. Lorsque le temps d'apprentissage est ignoré, ce critère est trop permissif et considère que l'apprentissage d'algorithmes triviaux est satisfaisant. Nous allons donner maintenant la version formelle de l'apprentissage *PAC* polynomial (aussi appelée poly-*PAC*) comme l'a défini Pitt [Pit89].

Définition 7.23 (Identification *PAC* [Pit89])

- *Soit M un DFA de taille n.*
- *Soient ϵ et δ deux réels strictement positifs.*
- *Soit une distribution de probabilités $\mathbf{P}_D$ définie sur les chaînes de Σ^* de taille au plus p.*
- *Soit un algorithme (éventuellement aléatoire) A qui prend en entrée un ensemble de mots tirés conformément à la distribution D et étiquetés correctement par M.*
- *Soit l le nombre d'exemples obtenus par A.*

A PAC-identifie les DFA si, quel que soit M, quel que soit D, quel que soit m, A produit un DFA M' en temps polynomial en n, p, l, $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $|\Sigma|$ tel que :

$$\mathbf{P}\Big(\sum_{x \in L(M) \ominus L(M')} \mathbf{P}_D(x) < \epsilon\Big) > 1 - \delta$$

On notera que le critère classiquement utilisé pour mesurer la taille d'un DFA est le nombre de ses états. Par ailleurs, le symbole $\ominus$ désigne la différence symétrique de deux langages et $|\Sigma|$ est la taille de l'alphabet.

On est ici dans le cas de l'apprentissage d'un concept sur un ensemble fini, puisque la taille des phrases est bornée par p. Un résultat supplémentaire sur les automates finis permet de considérer ce cas comme raisonnable dans un contexte d'approximation : pour tout $\gamma > 0$, il existe un entier p tel que la probabilité qu'une chaîne de longueur supérieure à p soit tirée est inférieure à γ.

2.5.2 Comparaison de l'apprentissage *PAC* et de l'identification à la limite

Quelques points de comparaison entre ces deux méthodes peuvent être soulignés :

- L'identification à la limite impose de retrouver exactement une représentation du langage cible. Le cadre *PAC* pour sa part se contente d'une approximation de la cible.
- L'identification à la limite ne pose aucune contrainte sur la complexité calculatoire des algorithmes alors que l'approximation *PAC* demande la polynomialité de l'inférence en fonction de la précision, de la tolérance imposée au résultat ainsi que de la taille de la cible et celle de Σ, et d'une borne sur la plus grande chaîne.
- L'identification à la limite ne garantit d'aucune manière la vitesse de convergence alors que l'approximation *PAC* impose que la précision de l'inférence soit dépendante de la quantité de données.
- Enfin, le critère de Valiant amène la notion de distribution de probabilités des exemples qui n'existait pas dans le critère de Gold. Il impose cependant que l'approximation soit garantie quelle que soit la distribution D alors que dans le cadre de l'identification à la limite, l'identification doit se faire pour toute présentation.

Copyright © 2010 Eyrolles.

2.6 Résultats *PAC* pour les langages réguliers

Il existe nombre de résultats négatifs concernant l'identification PAC. Celui qui va nous intéresser principalement a été démontré par Kearns et Valiant [KV89] : ils ont prouvé que la *PAC*-identification polynomiale, indépendamment de la représentation[13] de la classe des automates finis déterministes acycliques de taille est un problème NP-difficile.

Pitt et Warmutt ont montré que le problème qui consiste à retrouver un automate polynomialement plus grand que le plus petit DFA compatible avec les données est NP-difficile. Ils montrent de plus que la taille du vocabulaire prend part à la difficulté de l'inférence [PW93] :

Théorème 7.4

Le problème suivant est NP-difficile : étant donné une constante k et des exemples positifs et négatifs définis sur un alphabet Σ^*, de taille $|\Sigma| = 2$, existe-t-il un DFA comportant au plus n^k états compatible avec les données, avec n la taille du plus petit automate compatible.

Si l'on considère que la taille de l'alphabet n'est pas donnée mais fait partie du problème, on a [PW93] :

Théorème 7.5

Quelle que soit la constante $\epsilon > 0$, il n'existe pas d'algorithme polynomial permettant de déterminer s'il existe un automate compatible avec les données de taille $n^{(1-\epsilon)\log\log n}$.

Les principales variations du cadre *PAC* portent sur des restrictions sur la classe des distributions des exemples. Le principe est de donner à l'algorithme de « bons » exemples. Ce peut être réalisé soit en définissant un professeur qui choisit les exemples en fonction de la cible, soit en imposant des contraintes sur la distribution de probabilités : l'algorithme est alors presque assuré de travailler avec des données caractéristiques de la cible. Deux idées qui vont dans ce sens ont été proposées par Denis et *al.* [DDG96] d'une part et par Denis et Gilleron [DG97] d'autre part. La première approche se conforme au principe qui stipule que les exemples les plus simples doivent être préférés. Le deuxième point de vue propose que l'enseignant choisisse des exemples afin de faciliter l'apprentissage. Nous allons présenter maintenant ces deux approches.

2.7 Apprentissage *PACS* : *PAC* Simple

Le modèle *PACS* (*PAC* Simple) proposé par Denis et *al.* [DDG96] a pour but de donner à l'algorithme d'apprentissage des exemples qui sont en adéquation avec la cible. Ils définissent donc un professeur qui construit un ensemble d'apprentissage en fonction de la cible. Un autre professeur « adversaire » ou arbitre ajoute ensuite des exemples de son choix à l'ensemble d'apprentissage afin d'éviter une collusion complète entre l'enseignant et l'apprenant. Le système demeure cependant d'une certaine manière potentiellement collusif. Dans ce modèle, étant donnée une cible r, les exemples donnés à l'algorithme d'apprentissage, (c'est-à-dire les exemples simples) sont ceux qui ont une complexité de Kolmogorov étant donné r faible. Plus formellement, si on note $K(\alpha|r)$ la complexité de Kolmogorov[14] du mot α étant donnée r, l'ensemble d'apprentissage défini pour un DFA r sera : $S^r_{simple}\{\alpha \in \Sigma^* : K(\alpha|r) < \mu \log(|r|)\}$, où μ est une constante.

[13] L'apprentissage indépendant de la représentation consiste à laisser à l'algorithme le choix de la représentation : *AFN*, *AFD*, grammaire, etc.

[14] Pour une étude de la complexité de Kolmogorov, le lecteur pourra consulter [LV97].

Copyright © 2010 Eyrolles.

Résultats

Denis et *al.* [DDG96] ont montré que dans ce modèle les langages k-réversibles (voir au paragraphe 4.1) étaient identifiables.

Par ailleurs, Parekh et Honavar [PH97] ont montré que les DFA de taille N, avec N connu et fixé, étaient probablement exactement identifiables dans le modèle *PACS*. Ce résultat est plus fort que la *PACS*-apprenabilité au sens où l'identification de l'automate cible peut se faire avec une grande probabilité. Plus précisément, ils ont montré que l'échantillon caractéristique nécessaire à l'identification à la limite de l'algorithme RPNI[15] était présent avec une probabilité supérieure à $1 - \delta$, lorsque l'on se place dans le cadre *PACS* ayant pour critère de confiance δ.

La faiblesse principale de ce cadre de travail est qu'il suppose la possibilité du calcul de la complexité de Kolmogorov (voir chapitre 21). C'est donc un cadre purement théorique. Sa deuxième faiblesse est la difficulté d'interdire une collusion entre oracle et apprenant, c'est-à-dire qu'un codage de l'hypothèse peut être dissimulé dans les exemples. Conscients de ce problème, certains des auteurs se sont tournés vers une approche plus réaliste en pratique.

2.8 Apprentissage *PAC* avec distributions bienveillantes

Denis et Gilleron ont proposé [DG97] une adaptation de cadre *PAC* : l'apprentissage *PAC* avec des distributions bienveillantes.

Cette approche affaiblit le cadre *PAC* en opérant une restriction sur l'ensemble des distributions de probabilités selon lesquelles les exemples sont tirés. La classe de distributions bienveillantes est définie en fonction de la cible que l'algorithme doit identifier.

Informellement, un *exemple utile* est un exemple dont la taille est polynomiale en la taille de la cible. Une distribution bienveillante est une distribution telle que tous les *exemples utiles* ont une probabilité non nulle. Par ailleurs, on peut supposer que plus les exemples auront une probabilité faible, plus l'inférence sera difficile à réaliser. C'est pourquoi les auteurs ajoutent un paramètre de complexité au modèle *PAC* : l'algorithme d'inférence devra être polynomial en l'inverse de plus petite probabilité des exemples rencontrés.

Résultats

Denis et Gilleron montrent [DG97] d'une part que la classe des DFA définis sur Σ^n, n fixé et connu, est *PAC*-identifiable sous distributions bienveillantes.

La preuve est basée sur la propriété d'identification à la limite de l'algorithme RPNI [OG92a]. Elle consiste à construire un échantillon d'apprentissage à partir d'un enseignant et à montrer qu'un tel ensemble d'apprentissage contient probablement l'échantillon caractéristique.

Par ces deux approches, on appréhende maintenant mieux la difficulté de définir la notion d'apprentissage d'un automate. Le problème consiste à poser des contraintes sur les exemples que l'on donne à l'apprenant et à travers lui, sur la distribution de probabilités sur les exemples.

Si on ne fait aucun présupposé sur la pertinence des exemples (cadre *PAC* classique), le cadre est trop contraignant et des algorithmes qui infèrent correctement en pratique n'apprennent pas dans ce cadre. Si on propose de donner des exemples pertinents (cadre *PACS* et cadre *PAC* sous distributions bienveillantes) on ne peut éviter le problème de la collusion.

[15] Il sera question en détail de cet algorithme au paragraphe 5.4.

Copyright © 2010 Eyrolles.

3. L'espace de recherche de l'inférence régulière

3.1 Le point de la situation

Nous avons défini l'inférence régulière comme la recherche d'un automate A inconnu à partir duquel un échantillon positif I_+ est supposé avoir été généré. Étant donnée l'hypothèse supplémentaire de complétude structurelle de I_+ relativement à l'automate A, nous allons maintenant préciser la définition de ce problème comme une exploration dans un ensemble structuré construit sur I_+.

Nous retournons ainsi à un protocole de présentation tout à fait classique : l'algorithme d'apprentissage dispose de l'ensemble des exemples positifs et négatifs et tout renseignement supplémentaire par un oracle est interdit.

Nous avons vu au paragraphe 1.1 de ce chapitre un certain nombre de définitions et de notions formelles. Rappelons-les rapidement avant de les utiliser pour construire cet ensemble structuré.

- À partir d'un échantillon positif I_+, on peut construire deux automates particuliers qui n'acceptent que lui : $MCA(I_+)$ et $PTA(I_+)$. Le second est un automate dérivé du premier.
- À partir d'un automate A quelconque, on peut construire d'autres automates par l'opération de dérivation : chaque état d'un automate dérivé de A est la fusion d'un certain nombre d'états de A.
- Un automate dérivé de A reconnaît un langage qui inclut $L(A)$.
- L'ensemble des automates dérivés d'un automate A forme le treillis $Lat(A)$.
- Un échantillon est structurellement complet par rapport à un automate s'il est représentatif du langage qu'il accepte.

Nous disposons maintenant de tout le matériel technique pour définir l'ensemble de toutes les solutions d'un problème d'inférence régulière.

3.2 Deux propriétés fondamentales

Un premier théorème nous assure que, sous le biais de la complétude structurelle, l'ensemble des hypothèses compatibles avec l'échantillon est exactement le treillis construit sur $MCA(I_+)$.

Théorème 7.6

Soit I_+ un échantillon positif d'un langage quelconque régulier L et soit A n'importe quel automate acceptant exactement L. Si I_+ est structurellement complet relativement à A alors A appartient à $Lat(MCA(I_+))$. Réciproquement, si un automate A appartient à $Lat(MCA(I_+))$ alors I_+ est structurellement complet relativement à A.

Le second théorème assure que l'on peut réduire l'espace de recherche si on cherche l'automate canonique d'un langage.

Théorème 7.7

Soit I_+ un échantillon positif d'un quelconque langage régulier L et soit $A(L)$ l'automate canonique acceptant L. Si I_+ est structurellement complet relativement à $A(L)$ alors $A(L)$ appartient à $Lat(PTA(I_+))$.

De plus, nous pouvons énoncer une propriété d'inclusion entre ces deux treillis :

Copyright © 2010 Eyrolles.

Propriété 7.5

$Lat(PTA(I_+)) \subseteq Lat(MCA(I_+))$

Cette propriété découle directement de la définition 7.16 du $PTA(I_+)$ qui est un automate quotient du $MCA(I_+)$. De plus, comme le treillis $Lat(PTA(I_+))$ est généralement strictement inclus dans le treillis $Lat(MCA(I_+))$, rechercher une solution dans $Lat(PTA(I_+))$ au lieu de $Lat(MCA(I_+))$ permet de considérer un espace de recherche plus restreint.

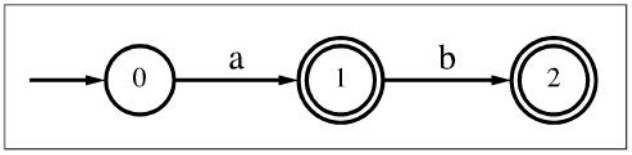


FIG. 7.13: *Un automate pour lequel l'échantillon $\{ab\}$ n'est pas structurellement complet.*

En guise d'illustration, considérons l'exemple suivant. Conformément à la définition 7.14, l'échantillon $I_+ = \{ab\}$ n'est pas structurellement complet relativement à l'automate A de la figure 7.13, où q_1 et q_2 sont des états d'acceptation. L'état 1, qui est final, n'est l'état d'acceptation d'aucune phrase de I_+. Pour cette raison, l'automate A ne peut pas être dérivé du $MCA(I_+)$ et évidemment pas non plus du $PTA(I_+)$. Il n'est pas possible de définir une partition π de l'ensemble des états Q_{MCA} telle que A corresponde à $MCA(I_+)/\pi$ avec $q_1 \in F_{MCA/\pi}$. Au contraire, l'échantillon $\{ab, a\}$ est structurellement complet relativement à l'automate A qui peut donc être dérivé du MCA, ou du PTA associés à cet échantillon.

3.3 La taille de l'espace de recherche

Grâce aux résultats présentés à la section 3.2, nous savons que si nous disposons d'un échantillon positif I_+ d'un langage inconnu L, avec I_+ structurellement complet relativement à un automate inconnu A acceptant exactement L, alors nous pouvons dériver A pour une certaine partition π de l'ensemble des états du $MCA(I_+)$. Nous pouvons dès lors considérer l'inférence régulière comme un problème de recherche de la partition π.

Mais l'exploration de cet espace n'est pas un problème facile : sa taille est le nombre de partitions $|\mathcal{P}(N)|$ d'un ensemble à N éléments où N est ici le nombre d'états de l'élément nul du treillis, c'est-à-dire $PTA(I_+)$ ou $MCA(I_+)$. Ce nombre croît plus qu'exponentiellement en fonction de N. À titre d'exemple, $|\mathcal{P}(10)| = 10^5$, $|\mathcal{P}(20)| = 5 \times 10^{13}$, $|\mathcal{P}(30)| = 8.5 \times 10^{23}$...

4. L'inférence régulière sans échantillon négatif

Étant donné un échantillon positif I_+ d'un langage régulier L, nous avons à définir le critère qui guidera la recherche de l'automate A acceptant I_+. En particulier, le $MCA(I_+)$ et le $PTA(I_+)$ acceptent tous deux l'échantillon positif. Cependant, ils n'acceptent aucune autre phrase du langage L et, donc, ne généralisent pas l'information contenue dans les données d'apprentissage.

Nous savons, par la propriété 7.1, que n'importe quel automate dérivé du $MCA(I_+)$ accepte un langage incluant I_+. En ce sens, n'importe quel automate appartenant à $Lat(MCA(I_+))$ constitue une généralisation possible de l'échantillon positif. Conformément au résultat de Gold présenté à la section 2.2, nous savons qu'il n'est pas possible d'identifier la classe entière des

Copyright © 2010 Eyrolles.

langages réguliers à partir d'un échantillon positif seulement. En particulier, aucun exemple positif ne peut éviter une surgénéralisation éventuelle de l'inférence. En d'autres termes, si une solution proposée par un algorithme d'inférence correspond à un langage incluant strictement le langage à identifier, aucun exemple positif ne peut contredire cette solution.

Pour éviter le risque de surgénéralisation, deux approches sont classiquement utilisées. La première possibilité consiste à rechercher la solution dans une sous-classe particulière des langages réguliers ; il s'agit alors d'une méthode *caractérisable*. La seconde possibilité consiste à utiliser une information *a priori* pour guider la généralisation recherchée, le propre des méthodes *heuristiques*.

4.1 Une méthode caractérisable : l'inférence de langages k-réversibles

Angluin a proposé une méthode d'inférence qui identifie à la limite à partir d'une présentation positive la classe des langages *k-réversibles*, pour k quelconque mais fixé *a priori* [Ang82]. Nous détaillons ci-après les principales notions nécessaires à la définition de cette méthode d'inférence.

Définition 7.24 (Automate inverse)

L'automate inverse $A^r = (Q, \Sigma, \delta^r, F, I)$ *d'un automate* $A = (Q, \Sigma, \delta, I, F)$ *est défini comme suit :*

$$\forall q \in Q, \quad \forall a \in \Sigma, \quad \delta^r(q, a) = \{q' \in Q | q \in \delta(q', a)\}.$$

Par conséquent, l'automate inverse A^r *est obtenu en inversant les états d'entrée et de sortie de l'automate original* A*, et en inversant également le sens des transitions de* A*. Nous présentons à la figure 7.14 un exemple d'automate et son inverse.*

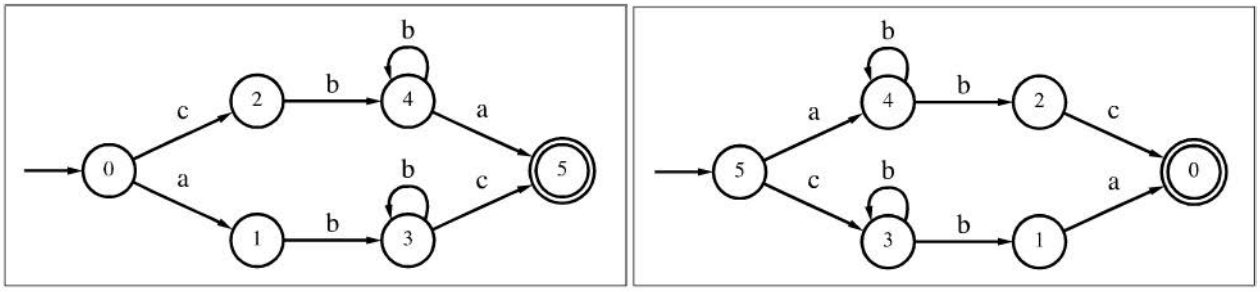


FIG. 7.14: *L'automate A et son inverse* A^r.

Il est à noter que nous avons défini, à la section 1, les automates finis comme comprenant un seul état d'entrée et un ensemble d'états de sortie. Ces automates permettent de représenter un langage régulier quelconque. La généralisation aux automates comportant un ensemble d'états d'entrée est triviale.

Définition 7.25

Un automate $A = (Q, \Sigma, \delta, q_0, F)$ *est* déterministe avec anticipation k *si :*

$$\forall q \in Q, \forall x \in \Sigma^*, |x| > k, \quad |\delta^*(x, q)| \leq 1.$$

où δ^* *désigne l'extension classique à* $\Sigma^* \times Q \to 2^Q$*, de la fonction de transition* δ*.*

Par conséquent, un automate est déterministe avec anticipation k si, lors de l'acceptation d'une chaîne quelconque, il existe au plus un état auquel cette acceptation peut mener en anticipant de k lettres par rapport à la position courante dans la chaîne. Un AFD est déterministe avec anticipation 0*. L'automate* A^r *de la figure 7.14 est déterministe avec anticipation* 1*.*

Copyright © 2010 Eyrolles.

Définition 7.26

Un automate A est k-réversible *s'il est déterministe et si son inverse A^r est déterministe avec anticipation k.*

Définition 7.27

Un langage L est k-réversible *s'il existe un automate k-réversible qui l'accepte.*

L'algorithme k-RI[16] consiste à partir de l'accepteur des préfixes de l'échantillon positif puis à fusionner des états, c'est-à-dire à définir un nouvel automate quotient, tant que la condition de k-réversibilité est violée. Angluin [Ang82] a démontré que cet algorithme permettait d'identifier la classes des langages k-réversibles et qu'il produisait, pour I_+ et k fixé, l'automate canonique du plus petit langage k-réversible incluant I_+ . Nous présentons à la figure 7.15 un exemple d'exécution de l'algorithme k-RI avec $I_+ = \{ab, bb, aab, abb\}$ et $k = 1$.

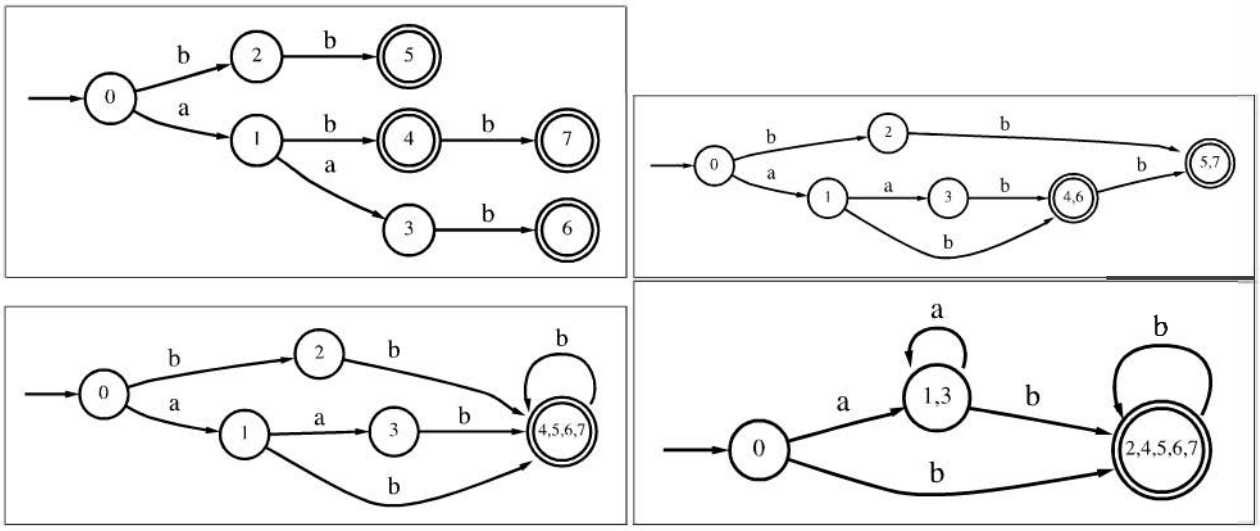


FIG. 7.15: *Exemple d'exécution de l'algorithme k-RI, avec* $I_+ = \{ab, bb, aab, abb\}$.

La fonction *k-réversible* (A/π) renvoie $VRAI$ si l'automate quotient A/π est k-réversible, elle renvoie $FAUX$ sinon. La fonction *non réversible* $(A/\pi, \pi)$ renvoie deux blocs distincts de π, tels que la condition de k-réversibilité ne soit pas vérifiée pour les deux états associés dans A/π. La fonction *dériver* (A, π') renvoie l'automate quotient de A conformément à la partition π'.

Rappelons qu'il est prouvé que cette méthode identifie à la limite la classe des langages k-réversibles pour k donné.

4.2 Une méthode heuristique : l'algorithme ECGI

L'algorithme ECGI est une méthode d'inférence heuristique conçue pour extraire l'information sur la longueur de sous-chaînes des éléments de I_+ et de la concaténation de ces sous-chaînes. Cet algorithme est incrémental, c'est-à-dire qu'il mesure une « distance » entre un nouvel exemple et le modèle existant et adapte le modèle conformément. Le langage accepté par la grammaire inférée inclut I_+ ainsi que d'autres chaînes obtenues par concaténation de sous-chaînes des éléments de I_+.

[16] L'abréviation k-RI provient de la terminologie anglaise pour *k-Reversible Inference*.

Copyright © 2010 Eyrolles.

Algorithme 5 : Algorithme k-RI

Données : k, l'ordre du modèle, I_+, l'échantillon positif
Résultat : A_k, un automate canonique acceptant le plus petit langage k-réversible incluant I_+

début
  /* N désigne le nombre d'états de $PTA(I_+)$ */
  $\pi \leftarrow \{\{0\},\{1\},\ldots,\{N-1\}\}$
  /* Un bloc par état du $PTA(I_+)$ */
  $A \leftarrow PTA(I_+)$
  tant que $\neg$(k-réversible (A/π)) **faire**
    $(B_1, B_2) \leftarrow$*non réversible* $(A/\pi, \pi)$
    $\pi \leftarrow \pi \backslash \{B_1, B_2\} U \{B_1 U B_2\}$
    /* Fusion du bloc B_1 et du bloc B_2 */
  fin tant que
  A/π
fin

Nous reprenons la description de l'algorithme ECGI tel qu'il a été formellement présenté dans [RV88, RPV89]. La représentation utilisée est celle des grammaires régulières. Nous illustrons également son fonctionnement à l'aide de la représentation équivalente en automates.

Propriété 7.6

L'algorithme ECGI produit une grammaire $G = (N, \Sigma, P, S)$ régulière, non déterministe, sans cycle [17] *et qui vérifie la condition suivante :*

$\forall A, B, C \in N, \quad \forall b, a \in \Sigma \quad$ *si* $(B \rightarrow aA) \in P$ *et* $(C \rightarrow bA) \in P$ *alors* $b = a$.

En d'autres termes, la même lettre de l'alphabet terminal est associée avec toutes les productions ayant le même non-terminal en partie droite. Cela permet d'associer les terminaux aux états plutôt qu'aux arcs de l'automate équivalent.

Définition 7.28

À toute production dans P, sont associées les règles d'erreurs *suivantes :*

Insertion de a : $A \rightarrow aA, \forall (A \rightarrow bB) \in P, \forall a \in \Sigma$
Substitution de b par a : $A \rightarrow aB, \forall (A \rightarrow bB) \in P, \forall a \in \Sigma$
$A \rightarrow a, \forall (A \rightarrow b) \in P, \forall a \in \Sigma$
Suppression de b : $A \rightarrow B, \forall (A \rightarrow bB) \in P, \forall a \in \Sigma$
$A \rightarrow \epsilon, \forall (A \rightarrow b) \in P, \forall a \in \Sigma$

Définition 7.29

La grammaire étendue $G' = (N', \Sigma, P', S)$ *de G est la grammaire obtenue en ajoutant les règles d'erreurs à P.*

Définition 7.30

La dérivation corrective optimale *de $\beta \in \Sigma^*$ est la dérivation de β, conformément à la grammaire G' et qui utilise un nombre minimal de règles d'erreurs.*

[17] Une grammaire est sans cycle si quel que soit le non-terminal A, aucune phrase engendrée à partir de A n'est de la forme xA, avec $x \in \Sigma^+$. En pratique, cela revient à dire que l'automate correspondant ne comporte pas de boucle et donc que le langage engendré est fini.

Copyright © 2010 Eyrolles.

L'algorithme ECGI construit d'abord la grammaire canonique acceptant la première chaîne de I_+. Ensuite, pour chaque nouvel exemple β, la dérivation corrective optimale est calculée et la grammaire est étendue conformément à cette dérivation[18]. Elle produit la séquence optimale de règles correctes (celles déjà présentes dans la grammaire) et de règles d'erreurs pour l'acceptation de β. Nous présentons à la figure 7.16 un exemple d'exécution de l'algorithme ECGI prenant en entrée l'échantillon $I_+\{aabb, abbb, abbab, bbb\}$ et en représentant chaque grammaire par un automate associé. Les états et transitions en pointillés correspondent à l'extension de la grammaire à chaque étape.

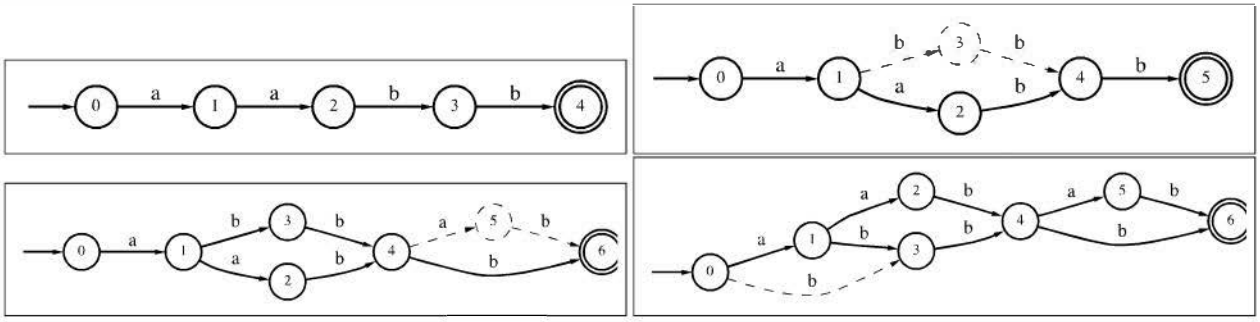


FIG. 7.16: *Exemple d'exécution de l'algorithme ECGI, avec* $I_+ = \{aabb, abbb, abbab, bbb\}$.

Algorithme 6 : Algorithme ECGI

Données : I_+
Résultat : Une grammaire G compatible avec I_+ et vérifiant la propriété 7.6
début
  $x \leftarrow I_+^1$; $n \leftarrow |x|$
  $N \leftarrow \{A_0, \ldots, A_{n-1}\}$; $\Sigma \leftarrow \{a_1, \ldots, a_n\}$
  $P \leftarrow \{(A_i \rightarrow a_i A_i), i = 1, \ldots, n-1\} \cup \{A_{n-1} \rightarrow a_n\}$
  $S \leftarrow A_0$; $G_1 \leftarrow (N, \Sigma, P, S)$
  pour $i = 2$ *à* $|I_+|$ **faire**
    $G \leftarrow G_{i-1}$; $x \leftarrow I_+^i$; $P' \leftarrow$ *dériv_optim* (x, G_{i-1})
    pour $j = 1$ *à* P' **faire**
      $G \leftarrow$ *étendre_gram* (G, p_j)
    fin pour
    $G_i \leftarrow G$
  fin pour
  Retourner G
fin

La fonction *dériv_optim* (x, G_{i-1}) renvoie la dérivation corrective optimale de la chaîne x conformément à la grammaire G_{i-1}. La fonction *étendre_gram* (G, p_j) renvoie la grammaire étendue conformément à la règle d'erreur p_j.

Supposons que nous disposions désormais d'un échantillon négatif I_- du langage inconnu L. Dans ce cas, nous savons que nous pouvons, en principe, identifier à la limite au moins n'importe quel langage régulier (voir section 2.2). L'inférence peut être alors considérée comme

[18] Cette dérivation est obtenue par une procédure d'analyse corrective qui constitue une application de l'algorithme de programmation dynamique (voir le chapitre 3 pour les généralités et le chapitre 12 pour une application particulière de la programmation dynamique : l'algorithme de Viterbi).

Copyright © 2010 Eyrolles.

la découverte d'un automate A compatible avec les échantillons positifs et négatifs, c'est-à-dire tel que $I_+ \subseteq L(A)$ et $I_- \cap L(A) = \phi$. Il existe un grand nombre d'automates compatibles appartenant à $Lat(MCA(I_+))$. $MCA(I_+)$ satisfait ces conditions mais ne généralise pas l'échantillon positif. Par conséquent, nous pouvons chercher une solution plus générale *sous le contrôle* de l'échantillon négatif.

Si nous choisissons la simplicité de l'automate inféré comme critère de généralité et que nous restreignons la recherche à des automates déterministes, la solution cherchée est alors l'AFD compatible et comportant le nombre minimal d'états. Il s'agit du problème du *plus petit AFD compatible*, déjà évoqué plus haut (paragraphe 2.3). Nous avons vu qu'il n'existe pas d'algorithme polynomial résolvant ce problème dans tous les cas. Cependant, on peut trouver un algorithme polynomial qui permet, en général, de trouver une solution quasi exacte. Nous détaillons cet algorithme à la section 5.4. Deux autres algorithmes seront abordés ensuite. Afin de comparer ces trois algorithmes et d'étudier le problème du plus petit AFD compatible, nous poursuivons la caractérisation de l'espace de recherche introduite à la section 3.

5. L'inférence régulière sous contrôle d'un échantillon négatif

5.1 L'ensemble frontière

La notion d'ensemble frontière permet de définir un sous-espace intéressant de solutions à un problème d'inférence avec contre-exemples.

Définition 7.31

Une antichaîne $\overline{AS}$ *dans un treillis d'automates est un ensemble d'automates tel qu'aucun élément de* $\overline{AS}$ *n'est relié par la relation de dérivation avec un autre élément de* $\overline{AS}$.

Définition 7.32

Un automate A est à une profondeur maximale *dans un treillis d'automates s'il n'existe pas, dans ce treillis, d'automate A', différent de A, tel que A' puisse être dérivé de A et tel que $L(A') \cap I_- = \phi$.*

Définition 7.33

*L'*ensemble frontière $BS_{MCA}(I_+, I_-)$ *est l'antichaîne dont chaque élément est à une profondeur maximale dans $Lat(MCA(I_+))$.*

Propriété 7.7

L'ensemble frontière $BS_{PTA}(I_+, I_-)$ contient l'automate canonique $A(L)$ de tout langage régulier L dont I_+ est un échantillon positif et I_- un échantillon négatif.

Par conséquent, l'ensemble frontière du treillis construit sur $MCA(I_+)$ est l'ensemble des automates les plus généraux compatibles avec l'échantillon positif et l'échantillon négatif. D'autre part, le problème du plus petit AFD compatible se ramène à découvrir le plus petit AFD appartenant à l'ensemble frontière du treillis construit sur $PTA(I_+)$.

Copyright © 2010 Eyrolles.

5.2 Le lien avec l'espace des versions

Compte tenu de sa construction à partir de I_+ puis de son élagage par I_-, le treillis des solutions est un cas particulier de l'espace des versions décrit au chapitre 4. On peut considérer simplement que le $PTA(I_+)$ est l'unique généralisation la plus spécifique, donc que $S = \{PTA(I_+)\}$. En revanche, il existe plusieurs solutions plus générales, qui ne sont autres que les éléments de l'ensemble frontière : $G = \mathcal{BS}_{MCA}(I_+, I_-)$. Tout élément du treillis qui peut se transformer en élément de BS par fusion d'états est une solution. L'inférence grammaticale ajoute un biais pour trouver une des solutions de l'espace des versions : chercher un automate sur BS, donc tel que toute fusion de deux de ses états conduise à l'acceptation d'exemples négatifs. C'est donc une heuristique de simplicité qui est adoptée, conformément au principe du rasoir d'Occam (voir au chapitre 3 le paragraphe 4.6) et au principe MDL.

5.3 Les algorithmes RIG et BRIG

Étant donnés des échantillons positif I_+ et négatif I_-, il est facile, par un algorithme nommé RIG[19] [MdG94], de construire complètement $\mathcal{BS}_{MCA}(I_+, I_-)$. Cet algorithme procède par énumération des automates dérivés du $MCA(I_+)$, c'est-à-dire des partitions dans $Lat(MCA(I_+))$. Il s'agit d'une énumération en largeur à partir de l'élément nul du treillis des partitions, le $MCA(I_+)$. Cette énumération ne conserve que les automates compatibles à chaque profondeur dans le treillis. Par la propriété 7.1 d'inclusion des langages, nous pouvons effectuer un élagage par héritage, qui consiste à éliminer tous les automates à la profondeur $i+1$ qui dérivent d'au moins un automate non compatible à la profondeur i. Ensuite, les automates restant à la profondeur $i+1$ subissent un *élagage direct* qui consiste à éliminer les automates non compatibles à cette profondeur. Finalement, l'ensemble frontière est obtenu en mémorisant tous les automates compatibles et qui ne possèdent aucun dérivé compatible. La solution proposée par l'algorithme RIG est, par exemple, le premier automate déterministe rencontré à la profondeur maximale de l'ensemble frontière, mais l'algorithme RIG peut également fournir comme solution tout l'ensemble frontière (comme le ferait l'algorithme de l'élimination des candidats dans la méthode de l'espace des versions).

La complexité de RIG étant non polynomiale[20], une version heuristique, nommé BRIG, a été proposée [MdG94]. Elle consiste à ne considérer qu'une petite proportion des partitions, générée aléatoirement. Cette sélection aléatoire permet de garantir, en pratique, une complexité polynomiale et de construire un sous-ensemble du BS. L'algorithme BRIG est clairement de nature heuristique car il n'existe pas de caractérisation de la classe de langages qu'il identifie (son caractère aléatoire implique que différentes exécutions partant des mêmes données ne conduisent pas nécessairement au même résultat).

5.4 L'algorithme RPNI

5.4.1 Une exploration efficace du treillis

L'algorithme RPNI [OG92b] effectue une recherche en profondeur dans $Lat(PTA(I_+))$ et trouve un optimum local au problème du plus petit AFD compatible. Nous savons que l'identification d'un langage régulier L par exploration du treillis des partitions du $PTA(I_+)$ n'est

[19] L'abréviation RIG provient de la terminologie anglaise pour *Regular Inference of Grammars*.

[20] Dans le pire des cas, aucun élagage par I_- n'est effectué. Dans ce cas, l'algorithme RIG explore tout le treillis des partitions. Malgré un élagage effectif par l'échantillon négatif, il a été observé expérimentalement que la complexité de RIG reste non polynomiale en fonction de N, le nombre d'états du $MCA(I_+)$ [MdG94].

Copyright © 2010 Eyrolles.

envisageable que si l'on suppose la complétude structurelle de I_+ relativement à l'automate canonique $A(L)$. Sous cette hypothèse, l'algorithme RPNI est particulièrement efficace.

Par construction du $PTA(I_+)$ (voir définition 7.16), chacun de ses états correspond à un préfixe unique et les préfixes peuvent être triés par ordre lexicographique[21]. Cet ordre s'applique donc également aux états du $PTA(I_+)$. L'algorithme RPNI procède en $N-1$ étapes où N est le nombre d'états du $PTA(I_+)$. La partition à l'étape i est obtenue en fusionnant les deux premiers blocs, par ordre lexicographique, de la partition à l'étape $i-1$ et qui, de plus, donne lieu à un automate quotient compatible.

Algorithme 7 : Algorithme RPNI

Données : I_+, I_-
Résultat : Une partition du $PTA(I_+)$ correspondant à un AFD compatible avec $I_+ et I_-$
début
  /* N désigne le nombre d'états de $PTA(I_+)$ */
  $\pi \leftarrow \{\{0\},\{1\},\ldots,\{N-1\}\}$
  $A \leftarrow PTA(I_+)$
  pour $i = 1$ à $N-1$ **faire**
    pour $j = 0$ à $i-1$ **faire**
      $\pi' \leftarrow \pi \backslash \{B_j, B_i\} U \{B_i U B_j\}$
      /* Fusion du bloc B_i et du bloc B_j */
      $\pi'' \leftarrow$ *déterm_fusion* (A/π')
      si A/π'' correct(I_-) **alors**
        $\pi \leftarrow \pi''$
      fin si
    fin pour
  fin pour
  Retourner $A \leftarrow A/\pi$
fin

L'automate quotient A/π' peut être non déterministe. La fonction *déterm_fusion* (A/π') réalise la *fusion pour déterminisation* en renvoyant la partition π'' obtenue en fusionnant récursivement tous les blocs de π' qui créent le non-déterminisme[22]. Si A/π' est déterministe, la partition π'' est égale à la partition π'.

Si l'automate déterministe ainsi obtenu est correct pour I_-, c'est-à-dire s'il n'en accepte aucun échantillon, alors $RPNI$ est relancé à partir de cette solution provisoire. Pour conserver l'ordre lexicographique, quand un état est fabriqué par fusion et correspond donc à un bloc d'états du PTA, il prend alors le rang de l'état du PTA de rang le plus faible dans ce bloc.

La solution proposée par l'algorithme RPNI est un automate déterministe appartenant au $BS_{PTA}(I_+, I_-)$. Par la propriété 5.1, nous savons qu'il s'agit de l'automate canonique pour le langage qu'il accepte. Cependant, il ne s'agit du plus petit AFD compatible que si les données d'apprentissage satisfont une condition supplémentaire, c'est-à-dire contiennent un échantillon

[21] L'ordre lexicographique sur les chaines de Σ^*, correspond à un ordre par longueur des chaînes et, pour une longueur donnée, à l'ordre alphabétique. Par exemple, pour l'alphabet $\{a, b\}$, les premières chaînes dans l'ordre lexicographique sont : $\epsilon, a, b, aa, ab, ba, bb, aaa, aab \ldots$

[22] L'opération de fusion pour déterminisation d'un AFN ne doit pas être confondue avec l'algorithme classique de déterminisation d'un AFN, qui a pour objet de produire un AFD acceptant le même langage [AU72]. Conformément à la propriété 7.1, l'AFD obtenu par fusion des états d'un AFN accepte, en général, un surlangage de l'AFN dont il provient.

Copyright © 2010 Eyrolles.

dit *caractéristique*, formellement défini dans [OG92b]. En d'autres termes, lorsque les données d'apprentissage sont suffisamment représentatives, la découverte de l'automate canonique du langage à identifier est garantie. De plus, cet automate est également la solution du problème du plus petit AFD compatible, dans ce cas particulier. Les auteurs ont démontré que la taille d'un échantillon caractéristique propre à cet algorithme est $\mathcal{O}(n^2)$, où n est le nombre d'états de l'automate cible [OG92b]. La complexité calculatoire de l'algorithme RPNI, dans sa dernière version publiée, est $\mathcal{O}((|I_+| + |I_-|) \cdot |I_+|^2)$.

Il est de plus démontré [TB73] que si l'échantillon d'apprentissage contient toutes les chaînes de longueur inférieures à $2k-1$ où k est le nombre des états de l'automate cible alors, l'identification est garantie. Mais cette propriété est fine : si l'ensemble d'apprentissage contient toutes les chaînes sauf une partie infime de l'ensemble caractéristique, alors, l'identification n'est plus garantie [Ang78].

—— EXEMPLE **Fonctionnement de RPNI** ——

Partons des ensembles $I_+ = \{\epsilon, ab, aaa, aabaa, aaaba\}$ et $I_- = \{aa, baa, aaab\}$. L'automate $PTA(I_+)$ est représenté sur la figure 7.17, en haut. Il possède dix états.

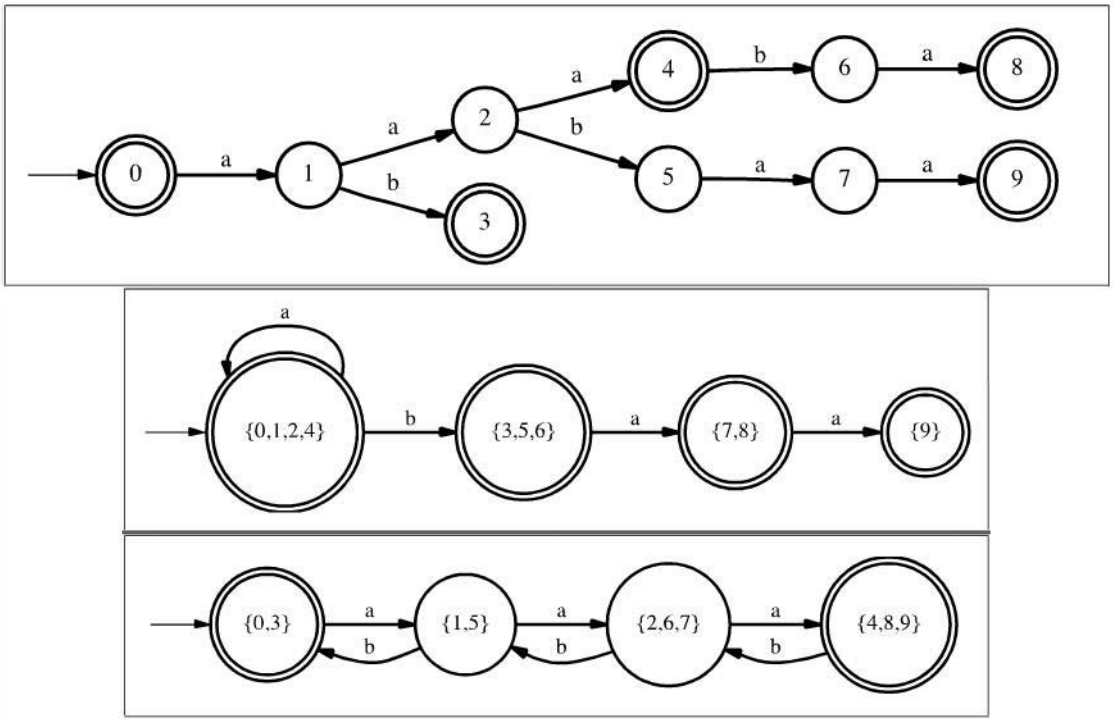


FIG. 7.17: *Exemple d'exécution de l'algorithme RPNI. En haut, $PTA(I_+)$, avec $I_+ = \{\epsilon, ab, aaa, aabaa, aaaba\}$. Au centre, le résultat de la fusion des états 0 et 1 suivie de la procédure de fusion pour déterminisation. Cet automate n'est pas une solution puisqu'il accepte au moins une phrase (ici, les trois) de $I_- = \{aa, baa, aaab\}$. En bas, l'automate obtenu comme solution.*

- L'algorithme *RPNI* commence par fusionner deux états. En l'absence d'autre indication les états 0 et 1 sont choisis. Ceci conduit à un indéterminisme, puisque l'état $\{0,1\}$ mène désormais à lui-même et à l'état 2 par une transition a. La procédure de fusion pour déterminisation est donc appellée.

Copyright © 2010 Eyrolles.

- Elle fusionne d'abord l'état $\{0, 1\}$ avec l'état 2. Mais ceci amène deux autres indéterminismes : $\{0, 1, 2\}$ mène à 4 et à lui-même par a, mais aussi à 5 et 3 par b.
- Elle poursuit en fusionnant $\{0, 1, 2\}$ avec 4 et en fusionnant 5 avec 3. Il reste un nouvel indéterminisme : $\{0, 1, 2, 4\}$ mène par b à $\{3, 5\}$ et 6.
- Elle fusionne derechef ces deux derniers états, ce qui mène à un automate déterministe (figure 7.17, au milieu).

Cet automate accepte la phrase aa dans I_-. On doit donc remettre en question la fusion initiale des états 0 et 1.

- L'étape suivante est de fusionner les états 0 et 2, puis de procéder à la fusion par déterminisation. Ici aussi, l'automate obtenu accepte une phrase de I_-.
- En revanche, l'étape suivante, la fusion de 0 et 3, mène à un automate cohérent avec I_+ et I_-, dont on sait qu'il est sur le BS. C'est cette solution qui est retenue (figure 7.17, en bas).

Le langage reconnu peut s'exprimer comme « l'ensemble des phrases dont le nombre de a moins le nombre de b vaut 0 *modulo* 3 ». L'échantillon a été choisi comme l'un des plus petits possibles qui soit caractéristique de ce langage, ce qui explique qu'on puisse l'inférer avec si peu d'information.

5.5 Variantes et extensions

La plupart des algorithmes efficaces d'inférence régulière sont actuellement des variations sur $RPNI$. La faiblesse de celui-ci est en effet d'imposer un ordre arbitraire aux essais successifs des fusions d'états du PTA (ceux-ci sont rangés dans l'ordre lexicographique du préfixe correspondant dans le PTA). Les algorithmes plus efficaces fusionnent de préférence les deux états les plus « prometteurs » avant d'appliquer la fusion pour déterminisation. Ce terme est évidemment empirique, mais il existe des heuristiques expérimentalement efficaces pour le définir. On trouvera les détails dans [LPP98].

Un certain nombre d'autres possibilités ont été développées, comme le travail dans un espace d'automates non déterministes, ou l'inférence par fission (en partant de l'automate universel).

Citons également ici, pour anticiper sur le chapitre 8 un algorithme heuristique, nommé GIG[23], qui optimise par un *algorithme génétique* la recherche d'une solution optimale, c'est-à-dire à la profondeur maximale. Il fait évoluer dans $Lat(PTA(I_+))$ une population d'automates dérivés du $PTA(I_+)$, par le biais d'opérateurs spécifiques de mutation et de croisement. Une technique analogue est aussi évoquée dans ce même chapitre 8 : les *stratégies d'évolution*, qui gèrent des populations d'automates.

6. L'inférence de grammaires algébriques

6.1 Présentation

Comme on l'a vu aux paragraphes précédents, l'inférence des grammaires régulières est un problème bien formalisé, possédant (si l'on admet le biais de la complétude structurelle) un cadre algorithmique clair. Ce n'est pas le cas de l'apprentissage des grammaires algébriques. On peut donner deux raisons à cet état de fait :

[23] L'abréviation **GIG** provient de la terminologie anglaise pour *Grammatical Inference by Genetic search*.

Copyright © 2010 Eyrolles.

- Les grammaires régulières sont un « tout petit » sous-ensemble des grammaires algébriques. L'espace de recherche d'un concept à partir d'exemples est donc beaucoup plus grand pour ces dernières.
- On ne dispose pas pour les grammaires algébriques d'une relation d'ordre qui permettrait de parcourir un treillis fini de solutions et de pratiquer un élagage systématique.

Ce second point est relié au fait que la représentation en automates finis, presque systématiquement utilisée pour l'inférence régulière, n'a pas d'équivalent commode pour les grammaires algébriques. Les algorithmes qui ont été proposés visent donc pour la plupart à contourner ces problèmes, soit en définissant des sous-classes de grammaires algébriques où l'on pourra appliquer des algorithmes du type de ceux utilisés pour les grammaires régulières, soit en enrichissant la présentation des exemples par un oracle. Nous présenterons brièvement une méthode de chaque type. La bibliographie de ce chapitre renvoie à des références plus complètes sur ce sujet difficile, qui a beaucoup progressé récemment. Un état des lieux théorique et pratique à lire en priorité est l'article de de la Higuera [dlHO06].

6.2 L'apprentissage à partir d'échantillons structurés

Cette famille de méthodes [CR71, Sak90], suppose que l'échantillon d'apprentissage (positif seulement) est composé non seulement de phrases, mais également pour chacune d'elles d'indications sur la façon dont la phrase a été engendrée par la grammaire à trouver. Cette hypothèse forte permet de mettre en œuvre un algorithme qui produit une grammaire compatible avec l'échantillon, en introduisant un nombre minimal de concepts ; en conséquence, l'espace de recherche est implicitement restreint et les grammaires trouvées appartiennent à une stricte sous-famille des langages algébriques.

Plus précisément, l'échantillon d'apprentissage est constitué de phrases structurées par un parenthésage représentant leur *arbre de dérivation* [AU72]. Par exemple, la grammaire G :

1. $S \rightarrow a$,
2. $S \rightarrow a + S$

engendre la phrase : $x = a + a + a$ par l'arbre de dérivation de la figure 7.18. Sous forme

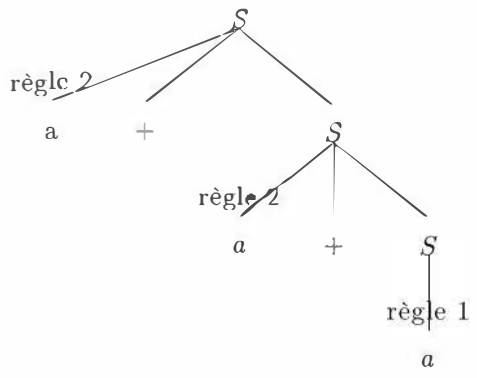


FIG. 7.18: *L'arbre de dérivation de la phrase $a + a + a$ par la grammaire G.*

structurée, cette phrase s'écrit : $x = (a + (a + (a)))$

Cette contrainte sur l'échantillon revient à supposer que chaque partie droite des règles de la grammaire qui a engendré cet échantillon est encadrée par une paire de parenthèses, et donc qu'elle appartient à un sous-ensemble des grammaires algébriques ; ce sous-ensemble est strict.

Copyright © 2010 Eyrolles.

On connaît des résultats théorique intéressants sur des variantes de l'identification à la limite de langages algébriques dans ce cadre : à partir d'échantillons positif et négatif structurés comme d'échantillon positif structuré et d'échantillon négatif classique. Leur exposé sort du cadre de cet ouvrage ; le lecteur intéressé peut se reporter à l'article de synthèse déjà cité [dlHO06].

L'algorithme que nous présentons [Sak90] infère des grammaires algébriques *réversibles*, dont la définition formelle est la suivante :

Définition 7.34

Une grammaire algébrique $G = (V, \Sigma, P, S)$ est réversible quand, pour tous non-terminaux A, B et C et toutes séquences α, β et γ de V^* :

- *s'il y a dans P deux règles $A \longrightarrow \alpha$ et $B \longrightarrow \alpha$, alors $A = B$, et*
- *s'il y a dans dans P deux règles $A \longrightarrow \alpha B \beta$ et $A \longrightarrow \alpha C \beta$, alors $B = C$*

Les grammaires réversibles possèdent une propriété très forte : tout langage algébrique peut être engendré par une grammaire réversible. Du point de vue de l'inférence grammaticale, apprendre un concept sous la forme d'une grammaire réversible ne limite donc pas les possibilités en terme de langage appris.

D'autre part, le fait d'utiliser des échantillons structurés permet de disposer d'une autre propriété intéressante du point de vue de l'apprentissage :

Propriété 7.8

La classe des langages algébriques est identifiable à la limite à partir d'échantillons structurés positifs, si l'on utilise une grammaire réversible pour représenter ces langages.

Nous présentons maintenant l'algorithme que Sakakibara a utilisé pour démontrer cette propriété, qui est opérationnel en pratique. Son principe est de construire une grammaire qui ne génère que l'échantillon, puis de généraliser la grammaire en éliminant les règles qui violent la condition de réversibilité. Sa convergence théorique est assurée, et bien qu'on ne connaisse pas de démonstration au fait qu'il pourrait exister un échantillon caractéristique de taille polynomiale, comme dans le cas de l'algorithme RPNI, cet algorithme converge en pratique sur des échantillons de ce type.

Donnons son déroulement sur un exemple.

— Exemple —

Soit :

$$I_+ = \{((ab)(c)), ((a(ab)b)(c(c))), ((ab)(c(c)))\}$$

La grammaire suivante est d'abord produite : elle ne génère que l'échantillon.

$$\begin{array}{ll} S \longrightarrow AB & B \longrightarrow c \\ A \longrightarrow ab & C \longrightarrow aC'b \\ S \longrightarrow CD & D \longrightarrow cD' \\ C' \longrightarrow ab & S \longrightarrow EF \\ D' \longrightarrow c & F \longrightarrow cF' \\ E \longrightarrow ab & F' \longrightarrow c \end{array}$$

On va procéder à une généralisation par fusion de non-terminaux. Pour préserver la nature réversible de la grammaire inférée, les non-terminaux B, D' et F' sont fusionnés en B, et A, C' et E sont fusionnés en A, ce qui amène la grammaire suivante :

$$\begin{array}{ll} S \longrightarrow AB & A \longrightarrow ab \\ B \longrightarrow c & S \longrightarrow CD \\ C \longrightarrow aAb & D \longrightarrow b \\ S \longrightarrow AF & F \longrightarrow cB \end{array}$$

Copyright © 2010 Eyrolles.

De nouveau, il faut fusionner trois non-terminaux : D et F et B en B, et dans la grammaire résultante, encore fusionner A et C en A. Finalement, on obtient :

$$\begin{array}{ll} S \longrightarrow AB & A \longrightarrow ab \\ A \longrightarrow aAb & B \longrightarrow c \\ B \longrightarrow cB & \end{array}$$

Cette grammaire reconnaît le langage $\{a^m b^n c^n \mid m, n \geq 1\}$.

6.3 L'inférence de grammaires linéaires équilibrées

Citons maintenant un algorithme qui concerne une sous-classe des grammaires algébriques.

Définition 7.35

Une grammaire linéaire équilibrée *est caractérisée par des règles de la forme : $A \rightarrow uBv \mid w$ avec : $u, v, w \in \Sigma^*, A, B \in N$ et : $\mid u \mid = \mid v \mid$;*

Un langage linéaire équilibré *est tel qu'il existe une grammaire linéaire équilibrée qui l'engendre. La famille de ces langages algébriques est strictement supérieure à celle des langages réguliers.*

Takada [Tak88] utilise la notion suivante :

Définition 7.36

Un langage est dit **engendré par une grammaire $G = (V, \Sigma, S, P)$ sous le contrôle d'un sous-ensemble $C \in P^*$** *quand toutes ses phrases se dérivent de S par une suite de règles de P formant un élément de C.*

Le sous-ensemble C peut être vu comme un langage sur P^* : il peut donc être régulier, algébrique, etc.

La méthode construit d'abord une certaine grammaire G_L, linéairement équilibrée universelle (autrement dit telle que $L(G_L) = X^*$), et utilise la propriété suivante :

Propriété 7.1

Pour tout langage linéaire équilibré L, il existe un unique sous- ensemble régulier $C \in P^*$ tel que L soit engendré par G_L sous le contrôle de C.

Le problème de l'inférence d'une grammaire linéairement équilibrée peut alors se ramener à celui de la construction d'un ensemble de contrôle éventuellement infini, mais régulier ; Takada propose un algorithme général d'inférence de ces grammaires dont la partie inductive peut être composée à partir de n'importe quel algorithme d'inférence régulière.

6.4 Une méthode par compression : SEQUITUR

L'idée est ici de définir des opérateurs sur la famille des grammaires hors-contexte qui transforme une grammaire en une autre, sans augmenter le langage associé. Il s'agit donc au départ plus de compression que de généralisation, mais cet algorithme calcule (apprend ?) effectivement une grammaire à partir de données. Le point de départ est la compression d'un texte, vu comme une suite de mots, donc la création d'une grammaire qui génère une seule séquence. Le principe est de remplacer les sous-séquences qui arrivent plus d'une fois par un nouvel élément non terminal.

Copyright © 2010 Eyrolles.

Par exemple, partant de la phrase *abcdbcabcdbcc*, l'algorithme SEQUITUR produit [LS00a] d'abord :

$S \longrightarrow abcdbcabcdbcc$

Il repère ensuite que la phrase s'écrit $(abcdbc)^2c$ et en déduit une forme plus compacte :

$S \longrightarrow AAc$

$A \longrightarrow abcdbc$

Il travaille enfin sur la partie droite de la seconde règle pour trouver :

$S \longrightarrow AAc$

$A \longrightarrow aBdB$

$B \longrightarrow bc$

L'intérêt de cette technique est sa rapidité (linéaire en la taille du texte traité). Cet algorithme s'est révélé très efficace pour la compression de séquences de nature variée : textes, musique, etc.

Son extension à une généralisation (donc à un apprentissage) a été considérée par un certain nombre de chercheurs. En effet, puisqu'il extrait une grammaire d'une phrase, il transforme cette dernière en un exemple structuré et permet d'envisager d'appliquer un algorithme du type de celui présenté au paragraphe 6.2. Les expériences dans cette voie se sont révélées plutôt décevantes [CE05].

6.5 Une méthode par exploration : l'algorithme GRIDS.

L'algorithme GRIDS [LS00a] utilise une combinaison de deux principes d'apprentissage : l'induction est menée à l'aide du principe de compression de l'information, sa validité est testée grâce au principe ERM. L'idée est de partir de la grammaire qui engendre les exemples d'apprentissage, et seulement ceux-là. Ensuite, des non-terminaux sont créés quand deux parties droites de règle partagent une même sous-séquence de symboles. Des non-terminaux sont fusionnés si leurs règles de production sont suffisamment proches. La table 7.3 explicite un exemple de création de non-terminal et un exemple de fusion de deux non-terminaux.

Création	Fusion
	$A \rightarrow aB$
$A \rightarrow abc$	$A \rightarrow aC$
$A \rightarrow abbc$	$B \rightarrow bc$
	$C \rightarrow bB$
$\Downarrow$	$\Downarrow$
$A \rightarrow aB$	$A \rightarrow aD$
$A \rightarrow abB$	$D \rightarrow bc$
$B \rightarrow bc$	$D \rightarrow bD$

TAB. 7.3: *Exemples d'actions que réalise l'algorithme* GRIDS. *Remarquons que la fusion des non-terminaux B et C entraîne une généralisation : le langage engendré est infini, car le nouveau non-terminal D est récursif, alors que ce n'était pas le cas avant la fusion.*

Le fonctionnement de l'algorithme est assez proche de celui décrit pour les algorithmes génétiques (voir le chapitre 8) : dans un premier temps il considère toutes les créations possibles de

Copyright © 2010 Eyrolles.

non-terminaux. Ensuite, il fait de même avec les grammaires résultantes pour les fusions de non-terminaux. Il utilise alors une fonction d'évaluation lui permettant de sélectionner la meilleure grammaire et de tester si cette grammaire obtient une évaluation supérieure à celle qu'il avait sélectionnée précédemment. Si c'est le cas, l'algorithme recommence le cycle des deux étapes création-fusion. Sinon, il retourne la grammaire précédente.

La fonction d'évaluation est l'implémentation du principe de compression de l'information. Cette fonction est une métrique rendant compte de la « simplicité » de la grammaire : elle prend en considération la taille de la grammaire (le nombre de non-terminaux, la taille des parties droites des règles, etc.) et la longueur des dérivations des mots de l'échantillon d'apprentissage dans cette grammaire. Ainsi, une grammaire est meilleure qu'une autre si le nombre de bits nécessaires pour réaliser le codage de cette grammaire et celui des mots de l'ensemble d'apprentissage dans cette grammaire est plus petit que le nombre de bits utilisés par le codage de la seconde grammaire (et des exemples d'apprentissage dans cette grammaire). Cette fonction permet d'éviter que soit toujours inférée la grammaire universelle.

Initialement, la grammaire n'engendre que les exemples d'apprentissage, et chaque étape ne fait que *généraliser* le langage engendré (le nouveau langage inclut l'ancien).

Cette approche a été reprise par d'autres chercheurs pour l'adapter à des problèmes particuliers. On peut citer par exemple l'algorithme E-GRIDS [PPK+04] qui combine cette approche avec un algorithme génétique, ou encore l'algorithme BOISDALE [SF04]. Il est à noter que l'un des plus anciens algorithmes d'inférence grammaticale, celui de Cook et Rosenfeld [CR74], fonctionnait déjà sur un principe analogue.

6.6 Une méthode empirique.

La méthode présentée dans ce paragraphe s'est révélée la plus efficace sur la compétition OMPHALOS [Cla06a] d'inférence de grammaires algébriques. Elle produit des grammaires particulières, dites NTS (*Non Terminally Separated*), dont l'intérêt en théorie des langages a été analysé en dehors de la question même de leur apprentissage [Cla06b]. La notion de base est celle de constituant d'une phrase produite par une grammaire algébrique :

Définition 7.37 (Constituant)

Soit $G = \langle \Sigma, V, P, S \rangle$ une grammaire algébrique et $L = \mathcal{L}(G)$ son langage engendré. Une occurrence d'un facteur u dans un mot lur du langage L est un constituant de L s'il existe $N \in V$ tel que $S \stackrel{*}{\Rightarrow} lNr$ et $N \stackrel{*}{\Rightarrow} u$.

Autrement dit, un constituant est un morceau de phrase (un facteur) produit par un non-terminal. Naturellement, en général, un facteur peut apparaître dans plusieurs phrases, tantôt comme constituant, tantôt sans cette propriété. Prenons par exemple la grammaire suivante :

$$S \longrightarrow aAa \mid bAb \mid c$$

Elle engendre les palindromes[24] sur $\Sigma\{a, b\}$ par exemple : $x = baabbaabbaab$. On peut remarquer que le facteur $baab$, qui apparaît trois fois dans x, n'est qu'une fois un constituant. Une grammaire NTS est définie par la propriété suivante :

[24] C'est-à-dire des mots w qui se lisent également de gauche à droite et de droite à gauche.

Copyright © 2010 Eyrolles.

Définition 7.38 (Grammaire NTS)

Une grammaire hors contexte $G = \langle \Sigma, V, P, S \rangle$ est NTS (Non Terminally Separated) si pour tout $N_1, N_2 \in V$ et mots $u, l, r \in \Sigma^*$ tels que $N_1 \stackrel{*}{\Rightarrow} lur$ et $N_2 \stackrel{*}{\Rightarrow} u$ il existe une dérivation $N_1 \stackrel{*}{\Rightarrow} lN_2r$.

Autrement dit, un facteur qui apparaît une fois comme constituant produit par N doit être aussi produit par N dans toutes ses autres occurences.

La grammaire précédente légèrement modifiée devient NTS :

$$S \longrightarrow aAa \mid bAb \mid c$$

Ses constituants sont du type général $wc\overline{w}$, par exemple *bacab*.

L'apprentissage à partir d'exemple d'une grammaire NTS consistera à rechercher les constituants dans les exemples (en s'aidant des contre-exemples) : puisque les constituants sont reliés aux éléments non terminaux, la grammaire s'en déduira.

Pour cela, il faut dans un premier temps construire l'ensemble des facteurs des exemples positifs contenus dans l'échantillon d'apprentissage. Le présupposé est que tous les constituants ont une fréquence d'apparition importante, ce qui permet d'éliminer un certains nombre de facteurs. Mais cela ne suffit pas : on trouve aussi souvent, voir plus fréquemment, les facteurs des constituants que les constituants eux-mêmes. C'est pourquoi dans un deuxième temps il faut sélectionner parmi les facteurs restants ceux qui ont le plus de chance d'être des constituants. Cela se fait en calculant un critère $MI(w)$, qui est d'autant plus élevé que le facteur w apparaît dans des contextes différents, donné par la formule suivante :

$$MI(w) = \sum_{l \in \Sigma'} \sum_{r \in \Sigma'} \frac{f(lwr)}{f(w)} log \frac{f(lwr)f(w)}{f(lw)f(wr)}$$

où $f(x)$ est la fréquence du facteur x dans les données, Σ' est l'alphabet du langage Σ auquel est ajouté un marqueur de début et de fin de mot (pour ne pas pénaliser les facteurs apparaissant à ces endroits dans les exemples). La méthode consiste donc à sélectionner les facteurs dont l'information mutuelle est la plus élevée.

En supposant que les étapes précédentes aient permis de trouver les constituants du langage, il reste à inférer une grammaire. Cela est réalisé à l'aide d'un *graphe de substitution*. Les nœuds de ce graphe sont constitués par les facteurs précédemment sélectionnés. Il y a un arc entre deux nœuds si les facteurs correspondants apparaissent au moins une fois dans le même contexte. A chaque *composante connexe* du graphe, c'est-à-dire à chaque ensemble de sommets directement ou indirectement reliés entre eux, est affectée un non-terminal dont les règles de production génèrent chaque noeud de la composante (tous ces constituants ont été générés par le même non terminal). On remplace ensuite toutes les occurrences des facteurs du graphe par le non terminal correspondant et on recommence le processus de sélection sur les formes sententielles ainsi obtenues. L'algorithme s'arrête quand la grammaire inférée est capable d'engendrer l'échantillon d'apprentissage.

Bien entendu cette méthode est purement heuristique : la fréquence et le MI minimaux pour qu'un facteur soit sélectionné sont des paramètres de l'algorithme qu'il faut patiemment faire évoluer pour trouver les optimaux. L'algorithme n'a d'autre part une chance d'aboutir que si la grammaire est NTS. D'autres heuristiques, utilisant par exemple les exemples négatifs quand ils étaient disponibles, ont été utilisées pour améliorer cet algorithme.

Copyright © 2010 Eyrolles.

La méthode EMILE [AV02] est apparentée à celle que nous venons de décrire, en ce qu'elle cherche les facteurs qui se retrouvent dans des contextes différents. Elle vise l'apprentissage des structures syntaxiques de la langue naturelle et a donné lieu à un certain nombre d'expériences sur l'apprentissage à partir de textes.

7. L'inférence d'automates probabilistes

Nous avons traité jusqu'ici des algorithmes d'apprentissage de concept sous la forme d'une grammaire. Pour toute phrase sur Σ^*, on sait décider si elle appartient ou non au langage engendré par une grammaire régulière ou algébrique. L'espace de représentation est donc partagé en deux par un concept grammaire. Une extension de l'inférence grammaticale consiste à étudier l'apprentissage de *grammaires probabilisées*.

7.1 Les automates probabilistes

Prenons d'abord le cas des automates finis non déterministes. Une extension naturelle consiste à associer à chaque transition une valeur comprise entre zéro et un et à chaque état deux valeurs comprises entre zéro et un. L'interprétation est la suivante. Si l'on est dans un état q, la première valeur $I(q)$ qui lui est associée est la probabilité de le considérer comme initial, la seconde $F(q)$ celle de le considérer comme final. De cet état partent des transitions : par exemple si $q' \in \delta(q, a)$, alors il existe une transition entre q et q' par la lettre a. La valeur $\mathsf{P}(q, a, q')$ associée à cette transition est la probabilité que cette transition soit empruntée. S'il n'existe pas de transition entre q et q' par la lettre a, alors $\mathsf{P}(q, a, q') = 0$. Les fonctions I, F et P doivent vérifier sur l'ensemble Q des états et l'alphabet Σ :

$$\sum_{q \in Q} I(q) = 1$$

$$\forall q \in Q,\ F(q) + \sum_{a \in \Sigma,\ q' \in \delta(q,a)} \mathsf{P}(q, a, q') = 1$$

Par exemple, la figure 7.19 représente un automate fini probabiliste à trois états. L'état S et l'état A ont chacun une probabilité 0.5 d'être initiaux, l'état F une probabilité de 1 d'être final. Les probabilités des transitions sont indiquées sur les arcs du graphe.

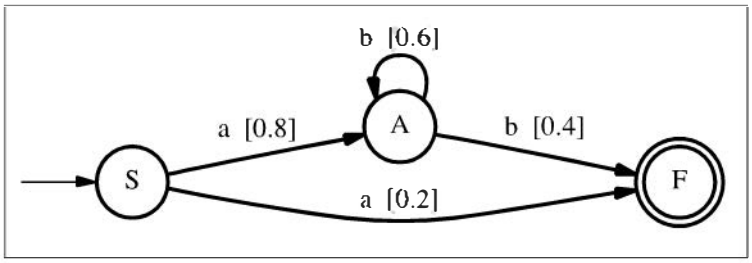


FIG. 7.19: *Un automate fini probabiliste.*

La probabilité d'une phrase d'être engendrée par un automate probabiliste se calcule comme la somme, sur toutes les façons possibles que cette phrase a d'être acceptée par l'automate, du

Copyright © 2010 Eyrolles.

produit des probabilités des règles utilisées multiplié celui des probabilités de l'état initial et de l'état final. Plus formellement :

Définition 7.39

Un chemin *dans un automate probabiliste est une suite d'états $q_1, \ldots, q_t$ et de transitions qui les relient, avec $t \geq 1$.*

À tout chemin correspond une phrase $x = x_1 \ldots x_{t-1}$ de longueur $t-1$ sur Σ^* (si $t = 1$, la phrase est ϵ).

La probabilité d'acceptation de la phrase x associée à un chemin est le produit $I(q) \times \mathbf{P}(q_1, x_1, q_2) \times \ldots \times \mathbf{P}(q_{t-1}, x_{t-1}, q_t) \times F(q_t)$ (si $t = 1$, cette valeur se ramène à $I(q_1) \times F(q_1)$).

Une phrase peut être acceptée par différents chemins.

Un chemin qui accepte x est valide *si la probabilité associée est non nulle. Pour la phrase $x = x_1 \ldots x_{t-1}$, notons $\theta(x)$ l'ensemble des chemins valides qui acceptent x.*

La probabilité d'acceptation de x par l'automate probabiliste est la somme sur θ de toutes les probabilités associées à un chemin de θ.

Par exemple, la phrase *abbb* peut être acceptée par l'automate de la figure 7.19 par la succession de transitions $S \xrightarrow{0.8} A \xrightarrow{0.6} A \xrightarrow{0.6} A \xrightarrow{0.3} F$ avec une probabilité de $0.5 \times 0.8 \times 0.6 \times 0.6 \times 0.6 \times 0.1 = 0.0432$, ou par la succession $S \xrightarrow{0.8} A \xrightarrow{0.6} A \xrightarrow{0.6} A \xrightarrow{0.3} A$ avec une probabilité de $0.5 \times 0.8 \times 0.6 \times 0.6 \times 0.3 \times 0.9 = 0144$. On la notera : *abbb*[0.00576]. De même, la phrase *bb*, qui peut être acceptée de deux façons, a la probabilité $0.5 \times 0.6 \times 0.6 \times 0.1 + 0.5 \times 0.6 \times 0.3 \times 0.9 = 0.099$ (*bb*[0.099] est acceptée par la suite d'états A, A et A ou par la suite d'états A, A et F).

7.2 Automates probabilistes et distributions de probabilités sur Σ^*

Un automate probabiliste A attribue donc à chaque phrase x de Σ^* une valeur $\mathbf{P}_A(x)$ entre 0 et 1. Cela induit naturellement la définition suivante :

Définition 7.40

Deux automates probabilistes sont dits équivalents s'ils attribuent les mêmes valeurs à toutes les phrases de Σ^*.

Mais il est important de noter qu'un automate probabiliste ne définit effectivement une distribution de probabilités sur Σ^*, avec la propriété $\sum_{x \in \Sigma^*} \mathbf{P}_A(x) = 1$, que si une certaine condition est remplie :

Propriété 7.9

Un état de A est dit utile *s'il apparaît dans au moins un chemin valide de A. Si tous les états de A sont utiles, alors A définit une distribution de probabilités qui somme à 1 sur Σ^*. On dit alors que A est* consistant.

Il est également important de savoir que toutes les distributions de probabilités sur Σ^* ne peuvent pas être définies par un automate probabiliste [dlHTV+05a].

7.3 Automates probabilistes déterministes

Nous avons traité jusqu'ici d'automates probabilistes en général *non déterministes*, comme celui de l'exemple : il possède en effet deux transitions portant la même lettre au départ de S, (comme de A, d'ailleurs). Il est naturel de se demander si à tout automate probabiliste non

Copyright © 2010 Eyrolles.

déterministe correspond un automate probabiliste déterministe équivalent ; mais il faut d'abord définir exactement ce dernier concept.

Définition 7.41

Un automate probabiliste est déterministe *quand de chaque état part au plus une transition portant une lettre donnée et quand la fonction I vaut 0 partout sauf sur un seul état.*

La réponse à la question précédente est alors négative [dlHTV+05a] ; autrement dit, étant donnée un automate probabiliste non-déterministe définissant une certaine distribution de probabilités sur Σ^*, on ne peut pas toujours trouver un automate probabiliste déterministe définissant la même distribution de probabilités sur Σ^*.

7.4 Inférence d'automates probabilistes

Un *échantillon d'inférence stochastique* est constitué d'un ensemble de phrases muni de fréquences d'apparition. On peut donc aussi le considérer comme un *multi-ensemble* de séquences. Par exemple, le multi-ensemble :

$$\{a, ab, ab, ab, abb, abb, abbbb\}$$

peut se récrire :

$$\{a[\frac{1}{7}], ab[\frac{3}{7}], abb[\frac{2}{7}], abbbb[\frac{1}{7}]\}$$

Le problème d'inférence posé dans cette section consiste donc à apprendre un automate fini stochastique à partir d'un tel échantillon d'inférence.

Si on suppose connaître les *composants structurels* d'un automate probabiliste, c'est-à-dire son alphabet, ses états et ses transitions, la question se ramène à l'estimation les fonctions I, F et $\mathbf{P}$ à partir de l'échantillon d'apprentissage. On peut remarquer alors que le problème semble très proche de celui de l'apprentissage de modèles de Markov cachés (HMM), traité au chapitre 12. Ce n'est pas qu'une apparence : il a été démontré que, sous certaines contraintes techniques dont le détail n'a pas d'importance ici, un automate probabiliste peut être transformé en un HMM de taille équivalente, et vice-versa [dlHTV+05a, dlHTV+05b, DDE05]. Nous renvoyons donc à ce chapitre pour l'examen de ce type d'apprentissage et nous travaillons donc maintenant sous l'hypothèse que les composants strucurels de l'automate à apprendre sont également inconnus.

Le détail des méthodes proposées ne sera pas développé. Nous donnons le principe commun à plusieurs d'entre elles [CO94a, Car97a, JP98, TD], qui est une généralisation de la technique RPNI (sans exemples négatifs) expliquée au paragraphe 5.4.

- On construit le PTA stochastique, qui génère au maximum de vraisemblance l'échantillon d'apprentissage stochastique.
- Le processus de généralisation est la fusion d'états.
- Quand deux états sont fusionnés, il faut recalculer les probabilités associées à ses transitions de sortie de manière à garantir la consistance du langage accepté.
- Le choix des deux états à fusionner se fait par exemple en cherchant à maximiser la ressemblance entre la distribution de probabilités sur l'échantillon et celle donnée sur Σ^* par l'automate obtenu ou sur un test statistique.
- Comme il n'y a pas de contre-exemple, l'arrêt du processus doit être réalisé par un critère empirique, qui établit un compromis entre la complexité de l'automate inféré et son adéquation avec l'échantillon.

Copyright © 2010 Eyrolles.

Notons qu'il existe une extension stochastique de l'algorithme ECGI qui permet d'inférer une grammaire régulière stochastique par une procédure incrémentale en une seule passe sur l'échantillon positif [CM99]. En effet, les probabilités associées aux productions de la grammaire peuvent être mises à jour par comptage de la fréquence d'utilisation des règles lors de l'analyse corrective de chaque nouvelle chaîne.

8. Quelques approches complémentaires

8.1 Le point de vue connexionniste

Terminons ce chapitre par un aperçu sur un certain nombre de travaux qui portent sur l'utilisation de *réseaux connexionnistes* en inférence grammaticale, habituellement à partir d'échantillons positif et négatif. Le principe de cet apprentissage ne correspond pas au cadre classique de l'inférence tel que nous l'avons présenté. Cependant une équivalence théorique entre certains réseaux connexionnistes et les automates finis déterministes a été démontrée.

L'idée est d'utiliser l'apprentissage par réseaux connexionnistes (voir le chapitre 10) ayant pour entrées des séquences et non plus des vecteurs de $\mathbb{R}^d$. Il faut pour cela utiliser des architectures plus complexes que celles des réseaux multicouches. Parmi les nombreuses variantes, les réseaux connexionnistes récurrents, et les réseaux d'ordre 2 ont été utilisés pour approcher le comportement d'automates finis. Des extensions de ces modèles ont été proposées afin d'approximer les *automates à piles* qui sont équivalents aux grammaires algébriques. Une bonne référence sur ces sujets est [SG98].

8.2 L'inférence de systèmes de réécriture

On peut se demander si l'apprentissage de systèmes de production de langages, comme les grammaires, ne peut pas être tenté en visant d'autres « machines » à générer des phrases. Les grammaires à réécriture, en particulier celles de la hiérarchie de Chomsky, bien que possédant de multiples avantages, ne sont peut-être pas foncièrement adaptées à l'apprentissage (du moins à partir des grammaires algébriques, puisque pour les automates finis, on a vu qu'il existait des algorithmes efficaces). Un certain nombre de propositions ont été formulées dans ce sens, dont celle d'apprendre des *systèmes délimités de réécriture de mots* (SDRM).

Sans entrer dans le formalisme complet, disons qu'un SRDM est, comme une grammaire formelle, un ensemble fini de règles de réécriture qui transforment un mot en un autre. La différence vient du fait que la syntaxe des règles n'est pas contrainte de la même façon. Prenons par exemple le système de réécriture suivant :

$$\frac{\begin{array}{rcl} aacbb & \vdash & acb \\ \$acb£ & \vdash & \$c£ \end{array}}{c}$$

Le fonctionnement consiste à partir d'une phrase, par exemple $aaacbbb$, de la marquer à gauche et à droite $\$aaacbbb£$, puis de tenter de la transformer en la phrase $\$c£$ (celle écrite sous la barre, rallongée des marqueurs) par l'application successive de règles écrites au-dessus de la barre. Ici, comme :

$\$a\underline{aacbb}b£ \vdash \$\underline{aacbb}£ \vdash \underline{\$acb£} \vdash \$c£$

on dit que $aaabbb$ appartient au langage engendré par le SRDM.

Il est facile de constater que, dans cet exemple, le langage est $\{a^n c b^n \mid n \geq 1\}$ (il est algébrique).

Copyright © 2010 Eyrolles.

Des algorithmes d'apprentissage ont été conçus pour certaines restrictions des SRDM, comme la méthode LARS [EdlHJ06]. Des propriétés de convergence ont été démontrées. Un des intérêts est que les grammaires apprises se placent en dehors de la hiérarchie classique : à partir d'exemples positifs et négatifs, certains langages réguliers ne sont pas apprennables, mais certains langages algébriques le sont.

Notes historiques et sources bibliographiques

L'inférence grammmaticale est désormais à la convergence de plusieurs domaines qui se sont développés plus ou moins indépendamment :

- l'algorithmique d'apprentissage pour la *reconnaissance structurelle des formes* [Fu74] ;
- la théorie de l'apprentissage [Pit89] ;
- les sciences cognitives.

Les algorithmes actuels sont très puissants. *RPNI* et ses dérivés sont capables d'apprendre des automates de plusieurs centaines de milliers d'états à partir d'échantillons de très grande taille. Ceci permet de représenter des concepts aussi complexes que des modèles de la langue écrite. Le développement des algorithmes d'inférence stochastiques a parallèlement permis l'apprentissage de concepts qui ressemblent à celui dont l'échantillon provient. Les applications sont désormais d'une grande variété et d'une grande efficacité. Citons en vrac comme domaines d'application :

- les bioséquences [AM97, BC97] ;
- la structure des textes électroniques [YLF99, Fer01, AMN94] ;
- le traitement du langage naturel [Vid94a] ;
- la traduction automatique [CGV94, ABC+01] ;
- la reconnaissance de la parole ([CM99] ;
- l'étude des styles musicaux [CAVR98] ;
- la reconnaissance des caractères manuscrits [LVA+94] ;
- la compression de données [NMWM94].

L'article [dlH05] présente un remarquable guide de lecture de la bibliographie de l'inférence grammaticale, en insistant particulièrement sur les applications. Le livre qui fera référence en inférence grammaticale est en cours de parution [dlH10].

Les conférences bisannuelles ICGI (*International Colloquium on Grammatical Inference*) [CO94b, MdlH96, HS98, dO00, AFvZ02, PS04, SKS+06, CCM08, Sem10] regroupent les travaux théoriques, algorithmiques et pratiques des spécialistes. Les revues *Machine Learning Journal*, *IEEE Transaction on PAMI*, *JAIR*, *Theoretical Computer Science*, *Pattern Recognition*, etc. publient régulièrement des articles (parfois dans des numéros spéciaux) consacrés à cette discipline.

La théorie de l'inférence grammaticale régulière est particulièrement bien décrite dans [Pit89]. L'analyse algébrique de l'espace de recherche a été décrite dans [DMV94]. Les performances des algorithmes sont détaillés dans [LPP98]. Les articles [Sak97], [Lee96] et surtout désormais [dlHO06] décrivent l'inférence de grammaires algébriques, tandis que [dlHTV+05a, dlHTV+05b] est complet sur les automates probabilistes. Une partie du texte de ce chapitre est repris de [DM98] et de [Dup96].

Copyright © 2010 Eyrolles.

On trouvera dans les articles bibliographiques des références à des techniques du même type qui permettent l'apprentissage de transducteurs (des automates finis qui transforment des séquences en d'autres séquences), à des grammaires d'arbres, à des grammaires de formes bidimensionnelles, etc.

Le site internet : `http ://eurise.univ-st-etienne.fr/gi/` est très complet sur l'inférence grammaticale. Il renvoie en particulier à des bibliographies thématiques et à des textes de présentation.

On peut aussi citer, dans un registre voisin, les travaux de D. Osherson, ([OSW86]) qui s'intéressent à l'apprentissage de fonctions récursives.

Résumé

- L'inférence grammaticale s'intéresse à l'apprentissage de concept à partir d'exemples positifs et négatifs de séquences. Sa théorie est riche, ses algorithmes et ses applications sont variés.
- Souvent, les concepts appris sont des automates finis, éventuellement probabilistes : ce sont des modèles simples, mais très utiles. L'apprentissage de grammaires algébriques, plus complexes, se développe rapidement.
- La théorie et les algorithmes de l'inférence grammaticale sont des thèmes riches et actifs en apprentissage.
- Des extensions existent pour les automates stochastiques, certains types de transducteurs et des modèles grammaticaux plus complexes, comme les grammaires d'arbres.

Copyright © 2010 Eyrolles.

Chapitre 8

Apprentissage par évolution simulée

La recherche d'une bonne hypothèse est généralement très difficile car les espaces d'hypothèses sont de taille considérable, souvent infinie, et que la fonction d'évaluation y est généralement irrégulière. C'est pourquoi il est avantageux de disposer soit d'une structure forte sur $\mathcal{H}$, comme c'est le cas lorsque l'on peut utiliser une relation de généralité entre hypothèses, soit de connaissances a priori *sur le domaine permettant de contraindre la recherche. À défaut de quoi, on est condamné à recourir à des méthodes d'exploration plus faibles, telles les méthodes de gradient plus ou moins sophistiquées (comme le recuit simulé).*

Les algorithmes évolutionnaires occupent une place intermédiaire. Le principe est de mener une recherche dans l'espace $\mathcal{H}$ en utilisant une population d'hypothèses que l'on fait évoluer selon les recettes de l'évolution naturelle : en utilisant des opérateurs génétiques de croisement et de mutation, et en sélectionnant les meilleurs individus pour engendrer la génération suivante d'hypothèses. D'un côté, on peut considérer qu'il s'agit là d'une sorte de méthode de gradient stochastique utilisant une population plutôt qu'une seule hypothèse. De l'autre, on peut aussi voir ces algorithmes comme des méthodes d'exploration guidées car elles découplent l'espace d'hypothèses en l'espace d'hypothèses proprement dit et un espace génotypique dans lequel des opérateurs d'évolution guident la recherche de meilleures hypothèses.

C'est en raison de ces deux visages que nous avons placé l'exposé de ces méthodes à la charnière de la partie sur l'apprentissage par exploration (guidée) et de la partie sur l'apprentissage par optimisation (utilisant des méthodes de gradient). Cette dualité les rend très attrayantes et importantes à connaître. Elles sont d'un large emploi dans les applications.

Copyright © 2010 Eyrolles.

Sommaire

1 Trois espaces au lieu de deux 265
2 Un modèle formel simplifié de l'évolution 268
2.1 Le jeu entre $\mathcal{H}$ et $\mathcal{G}$ 268
2.2 L'apprentissage comme processus d'évolution d'une population 268
3 Les algorithmes génétiques 269
3.1 La représentation dans l'espace génotypique 270
3.2 L'algorithme générique 270
3.3 L'initialisation de la population 270
3.4 Les opérateurs 271
3.4.1 Les opérateurs de croisement 271
3.4.2 Les opérateurs de mutation 272
3.4.3 Débat : le rôle des opérateurs 272
3.5 La sélection 273
3.5.1 L'étape de sélection 274
3.5.2 L'étape de remplacement 276
3.6 Le théorème des schémas : une explication de la puissance des AGs ? . . 276
4 Les stratégies d'évolution 281
5 La programmation génétique 281
5.1 La représentation des programmes 284
5.1.1 L'ensemble des terminaux et des fonctions 284
5.1.2 Le choix de l'ensemble des terminaux et des fonctions 285
5.1.3 Phénotypes et génotypes en programmation génétique 285
5.1.4 Initialisation d'une population de programmes 285
5.2 Les opérateurs génétiques sur les programmes 285
5.3 Évaluation et sélection 287
5.4 Le fonctionnement 287
5.5 Illustrations 287
5.5.1 Induction d'une séparatrice entre deux classes 287
5.5.2 Induction d'un programme de contrôle de robot simulé 288
5.6 Une analyse de la programmation génétique 290
6 La coévolution 291
6.1 Un exemple d'écologie : les systèmes de classeurs 293

Copyright © 2010 Eyrolles.

LES CANARDS ne sont pas tous sauvages. Certains sont domestiqués et modifiés à des fins alimentaires ou esthétiques. Pour ce faire, les éleveurs « créent » des races par des manipulations génétiques naturelles, essentiellement par des croisements visant à améliorer tel ou tel critère sur l'oiseau : sa qualité gustative, la quantité de sa ponte ou certains aspects de son apparence. Voici deux exemples de ce que l'on peut lire dans une encyclopédie récente des animaux domestiques[1] :

[1] A. Raveneau : *Inventaire des animaux domestiques en France (bestiaux, volailles, animaux familiers et de rapport)*. Nathan, 1993.

Canard Kaki Campbell. Cette race doit son nom à une éleveuse anglaise du Gloucestershire, Mrs Campbell, qui la créa en 1901 à partir de canard sauvage, de Coureur indien brun et blanc et peut-être d'Orpington. On attribue à une éleveuse, Mme Flamencourt, les premières sélections françaises dans les années 1920.(...) La cane est utilisée en croisement avec des mâles Pékin pour la production de canards dits Nantais.

Canard Huppé. Mutation du canard fermier. Les sujets les mieux huppés ont été sélectionnés par des éleveurs séduits par cette singularité afin de créer un standard.

On voit donc que l'homme peut utiliser avec succès les principes darwiniens de la sélection pour optimiser sa production domestique. Il est moins certain que l'on puisse considérer l'évolution naturelle comme une optimisation, non pas parce que les espèces existantes sont ratées, mais parce le critère que la nature chercherait à optimiser est difficile à formuler et peut-être inexistant. Restons donc sur l'idée que, étant donné un animal cible, il est parfois possible de fabriquer un animal proche de cette cible, en utilisant des croisements et en profitant des mutations naturelles.

En quoi cette constatation peut-elle être utile à l'apprentissage ? D'abord, comme on l'a vu, l'apprentissage artificiel est étroitement relié à l'optimisation, soit sous la forme de l'exploration d'un espace structuré, soit par l'utilisation de propriétés analytiques. Ensuite, puisqu'il est possible de modifier avec une grande rapidité (quelques dizaines de générations) le code génétique d'animaux complexes dans une direction choisie, on peut se demander si une modélisation informatique de ce code et la programmation d'une technique singeant l'évolution naturelle ne seraient pas une bonne technique d'optimisation.

Cette question a été posée dès les années 1960 et la réponse a été positive sous plusieurs angles. D'abord, les algorithmes dits génétiques, fondés sur l'idée précédente, ont connu un grand succès dans l'optimisation de fonctions compliquées et peu régulières, au prix cependant d'un temps important de calcul. Plus intéressant pour l'apprentissage, de plus en plus indépendamment de la métaphore biologique, des techniques d'évolution de code, au sens informatique cette fois, ont été développées. C'est ce que l'on appelle la programmation génétique. Le but de ce chapitre est de présenter un point de vue unifié sur ces techniques et d'en montrer la puissance et l'intérêt pour l'apprentissage artificiel.

Notations utiles pour ce chapitre

P_{mut}	Probabilité de mutation d'un bit
P_{crois}	Probabilité de croisement
$P_{\mathcal{H}}(t)$	Population d'hypothèses dans $\mathcal{H}$ à l'instant t.
$P_{\mathcal{G}}(t)$	Population de génomes dans $\mathcal{G}$ à l'instant t.
N	Taille de la population $P_{\mathcal{G}}$
ϕ	La fonction de performance (*fitness function*)
t	Le nombre d'individus participant à un tournoi
L	La longueur des chaîne de bits des génomes de $\mathcal{G}$
$n(s,t)$	Nombre de génomes représentants le schéma s dans la population $P_{\mathcal{G}}(t)$

1. Trois espaces au lieu de deux

Jusqu'à présent, à l'exception du chapitre 6, l'apprentissage a été abordé comme un jeu entre deux espaces :

Copyright © 2010 Eyrolles.

- l'*espace des exemples* $\mathcal{X}$, correspondant au monde à décrire et à interpréter. ;
- l'*espace des hypothèses* $\mathcal{H}$, correspondant à l'espace des descriptions du monde ou encore des modèles ou théories.

Le but de l'apprentissage est de trouver une, ou plusieurs, bonne(s) description(s) du monde $\mathcal{X}$ dans $\mathcal{H}$. De ce fait, l'espace des hypothèses est chargé de trois missions :

1. Il doit *réaliser un biais adéquat*, c'est-à-dire pouvoir décrire les modèles du monde performants dans l'application visée, tout en étant suffisamment limité dans son expressivité pour que l'induction soit possible et peu coûteuse en terme de taille d'échantillon de données. Ce prérequis a été largement analysé dans le chapitre 2.
2. Il doit aussi correspondre à un *langage d'expression des hypothèses* $\mathcal{L}_{\mathcal{H}}$ *aussi intelligible que possible* s'il est désirable qu'un expert humain puisse interpréter ce qui est appris, y compris pour orienter le système apprenant. C'est pourquoi, quand c'est possible, on préférera souvent un apprentissage de règles plutôt qu'un apprentissage par réseau connexionniste, dont le résultat est plus difficile à interpréter.
3. Finalement, il doit *posséder une topologie favorisant une exploration efficace*, c'est-à-dire telle qu'elle permette d'atteindre une hypothèse ou une combinaison d'hypothèses quasi optimale avec un coût d'exploration limité. En général, l'exploration se fait par modification des hypothèses courantes par l'emploi d'opérateurs syntaxiques simples opérant sur l'expression des hypothèses dans le langage $\mathcal{L}_{\mathcal{H}}$. Le problème est de trouver des opérateurs permettant d'atteindre n'importe quel point de $\mathcal{H}$ à partir de n'importe quel autre point (propriété de complétude) et conduisant naturellement aux régions de $\mathcal{H}$ les plus prometteuses.

Ces trois missions ne sont pas forcément faciles à remplir simultanément. Il n'est généralement pas aisé de trouver un langage $\mathcal{L}_{\mathcal{H}}$ à la fois limité du point de vue du biais inductif, mais expressif et intelligible, tout en possédant des propriétés syntaxiques favorisant l'exploration. Pourquoi alors ne pas découpler ces objectifs ? C'est ce que réalisent les algorithmes d'évolution simulée.

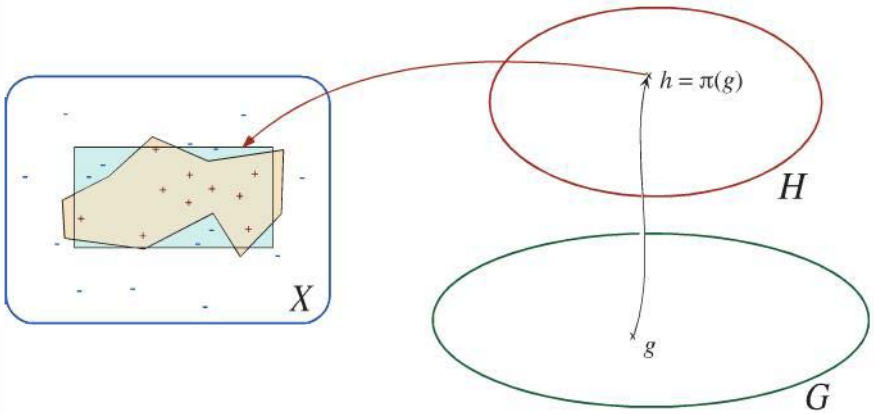


FIG. 8.1: *L'induction est ici réalisée par un jeu entre trois espaces : d'abord l'espace des exemples* $\mathcal{X}$, *ensuite l'espace des hypothèses* $\mathcal{H}$ *dans lequel est cherchée une description adéquate de* $\mathcal{X}$, *et enfin l'espace « génotypique »* $\mathcal{G}$ *dans lequel s'effectuent les opérations permettant l'exploration de* $\mathcal{H}$.

L'idée est la suivante : à l'espace des hypothèses $\mathcal{H}$ est associé un autre espace appelé, pour des raisons de similarité avec les mécanismes de l'évolution naturelle, espace « génotypique » et

Copyright © 2010 Eyrolles.

noté $\mathcal{G}$. Lorsqu'il faut changer d'hypothèse h_t dans $\mathcal{H}$, on prend l'expression génotypique associée g_t dans $\mathcal{G}$ à laquelle on fait subir des opérateurs de modification pour obtenir de nouvelles expressions g_{t+1} correspondant à de nouvelles hypothèses h_{t+1} qui sont alors testées pour mesurer leur performance dans le monde $\mathcal{X}$. Ce cycle d'évaluation des hypothèses dans $\mathcal{H}$ et de modification *via* $\mathcal{G}$ se déroule jusqu'à ce qu'une bonne hypothèse soit trouvée ou qu'il soit décidé de consacrer les ressources calculatoires à d'autres tâches, y compris éventuellement à une autre passe d'apprentissage en partant de conditions initiales différentes (voir la figure 8.1).

Le découplage des missions dévolues à l'espace des hypothèses avec un espace spécial dédié à la mission d'exploration est une idée intéressante. Le problème est d'identifier les espaces $\mathcal{G}$ appropriés. Plusieurs écoles se sont développées en concurrence depuis les années 1960, chacune affirmant avoir identifié le bon espace $\mathcal{G}$ avec ses opérateurs associés. Se distinguent essentiellement :

- Les *algorithmes génétiques* développés par John Holland à partir des années 1960 et analysés en profondeur en 1975 [Hol75]. Holland a ainsi proposé une analyse théorique, le « théorème des schémas », montrant selon lui la supériorité d'un espace génotypique décrit dans un langage de chaînes de bits. Les références à la métaphore de l'évolution naturelle sont les plus explicites dans ce travail et le principe général de ces algorithmes a servi de standard pour les autres approches.
- Les *stratégies d'évolution* développées par Rechenberg et Schwefel en 1965 pour la résolution de problèmes d'optimisation de structures mécaniques. Cette école préconise l'utilisation de $\mathbb{R}$ pour l'espace génotypique.
- La *programmation évolutive*, développée par Fogel à partir de 1966, étudie l'exploration d'un espace d'automates à états finis (FSA) pour décrire certains types de fonctions et de programmes.
- La *programmation génétique*, défendue avec force par Koza depuis 1992, promeut l'usage d'un espace d'arbres décrivant des programmes dans un langage adéquat (par exemple des expressions du langage Lisp).

Si les applications et les idées de développement sont foisonnantes pour l'ensemble de ces approches, les arguments théoriques expliquant leur fonctionnement et démontrant la supériorité de l'une d'entre elles sur les autres sont encore au stade exploratoire. Beaucoup reste donc à faire dans ce domaine.

Après une description générique de ces approches, ce chapitre étudie brièvement tour à tour les quatres familles de méthodes décrites ci-dessus. Une idée intéressante qui s'impose naturellement avec ces approches d'*évolution simulée* est celle de populations d'hypothèses se développant en symbiose. Ce phénomène est appelé « coévolution ». Ce chapitre se termine par sa description et par les perspectives ouvertes vers le développement d'« écologies » d'hypothèses, c'est-à-dire des théories (au sens de fragments de connaissance organisés), plutôt que sur la recherche d'une hypothèse unique.

Afin de souligner l'idée essentielle, selon nous, du passage par un espace dédié à la mission d'exploration, nous appellerons ces familles de techniques « algorithmes opérant sur le génotype » ou encore AGs . Nous devons avertir le lecteur que cela est original et qu'en général on parle plutôt d'algorithmes d'évolution simulée (ES), tandis que l'acronyme AG est réservé à l'approche des algorithmes génétiques.

Copyright © 2010 Eyrolles.

2. Un modèle formel simplifié de l'évolution

Cette section introduit les concepts fondamentaux communs aux algorithmes opérants sur le génotype ainsi que l'algorithme d'apprentissage général.

2.1 Le jeu entre $\mathcal{H}$ et $\mathcal{G}$

Nous considérons désormais que l'espace des hypothèses $\mathcal{H}$ est couplé à un espace génotypique $\mathcal{G}$. Quelle est la nature de ce couplage ? Notons π la relation liant $\mathcal{G}$ à $\mathcal{H}$. En général, il ne s'agit pas d'une bijection qui associe à chaque h un unique g. S'il y a en effet toujours surjection : chaque hypothèse h est l'image par π d'au moins un élément $g \in \mathcal{G}$ ($\mathcal{H}$ étant le plus souvent défini par référence à $\mathcal{G}$: $\mathcal{H} = \pi(\mathcal{G})$), il n'y a pas nécessairement injection : plusieurs éléments g pouvant être projetés sur une même hypothèse h. On parlera donc à raison de dualité entre $\mathcal{G}$ et $\mathcal{H}$, mais ce sera par abus de langage que l'on parlera de couple (g, h).

Dans cette dualité ($\mathcal{G}$, $\mathcal{H}$), la répartition des rôles est la suivante :

- La *mesure de performance* s'effectue dans $\mathcal{H}$: on mesure la valeur d'une hypothèse donnée en la confrontant, par l'intermédiaire d'une fonction de risque, appelée ici mesure de performance (*fitness function*), aux données d'apprentissage fournies dans $\mathcal{X}$. On notera η cette fonction de performance : $\eta : \mathcal{H} \times \mathcal{X} \rightarrow \mathbb{R}$.
- La *variation* concommittante à l'apprentissage s'effectue dans $\mathcal{G}$. En réponse à la mesure de performance attachée à $\pi(g_t) = h_t$, le système calcule un ou plusieurs nouveaux éléments g_{t+1} correspondant à une ou plusieurs nouvelles hypothèse(s).

2.2 L'apprentissage comme processus d'évolution d'une population

Ce processus d'apprentissage par jeu entre un espace $\mathcal{H}$ d'individus se mesurant à l'environnement et un espace $\mathcal{G}$ responsable du calcul de nouveaux individus a été en grande partie influencé par la métaphore de l'évolution des espèces telle que décrite par Darwin en 1859 [Dar59]. C'est pourquoi on qualifie souvent l'espace $\mathcal{H}$ de phénotypique : celui où s'expriment les traits des individus dont le génotype est décrit dans $\mathcal{G}$, l'espace génotypique. Par ailleurs, les méthodes d'apprentissage issues de cette métaphore utilisent souvent une population d'individus pour réaliser l'apprentissage plutôt qu'un individu unique. Dans ce cas, on considère une population $P_{\mathcal{G}}(t)$ d'éléments de $\mathcal{G}$ à l'instant t qui, en fonction des performances des hypothèses de la population $P_{\mathcal{H}}(t)$ associée, détermine une nouvelle population $P_{\mathcal{G}}(t+1)$ à l'étape suivante. On a alors le processus schématisé dans la figure 8.2 et dont les étapes du cycle général sont les suivantes :

Étape 1 **Expression** : une population $P_{\mathcal{G}}(t) \in \mathcal{G}$ est projetée dans l'espace phénotypique $\mathcal{H}$ par l'application π : $P_{\mathcal{H}}(t) = \pi(P_{\mathcal{G}}(t))$.

Étape 2 **Mesure de la performance** : la population d'hypothèses est évaluée, en général par une évaluation individuelle de chaque hypothèse, grâce à la fonction ϕ qui utilise les données d'apprentissage. Cela évalue implicitement la population $P_{\mathcal{G}}(t)$.

Étape 3 **Sélection d'une opération de variation de $P_{\mathcal{G}}(t)$** : en fonction de l'évaluation de la population $P_{\mathcal{H}}(t)$, le système sélectionne l'opération de variation qui va transformer la population $P_{\mathcal{G}}(t)$.

Étape 4 **Variation** : l'opération de variation sélectionnée à l'étape précédente est appliquée à $P_{\mathcal{G}}(t)$ pour obtenir la nouvelle population $P_{\mathcal{G}}(t+1)$.

Copyright © 2010 Eyrolles.

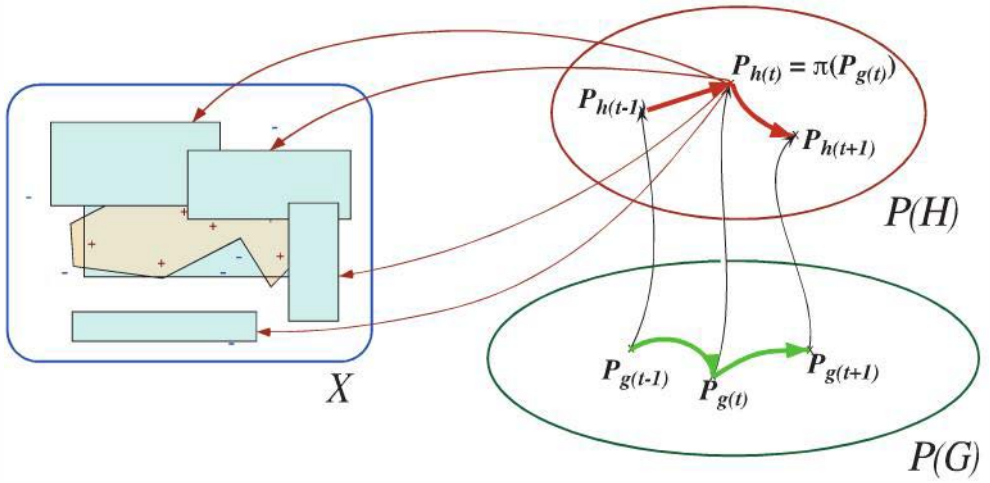


FIG. 8.2: *Le processus général d'apprentissage par l'utilisation d'un espace génotypique. Nous avons noté ici $\mathcal{P}(\mathcal{H})$ et $\mathcal{P}(\mathcal{G})$ des espaces de populations d'hypothèses et de génomes. De ce fait, dans ce schéma, il faut comprendre $P_{h(t)}$ et $P_{g(t)}$ respectivement comme une population d'hypothèses à l'instant t et une population de génomes à l'instant t. Ces populations évoluent en cours de recherche.*

Ce cycle est répété jusqu'à ce qu'une population $P_{\mathcal{H}}(t)$ obtienne une évaluation jugée satisfaisante (par exemple la meilleure hypothèse $h \in P_{\mathcal{H}}(t)$ ne change plus ou très peu du point de vue de sa performance), ou jusqu'à ce qu'une borne de ressources calculatoires soit atteinte.

Nous allons illustrer ce processus général en étudiant le cas des algorithmes génétiques.

3. Les algorithmes génétiques

Les algorithmes génétiques ont été popularisés par John Holland en 1975 grâce à son livre *Adaptation in natural and artificial systems* (réédité en 1992) [Hol75]. Dans ce travail, Holland cherche à utiliser la métaphore de l'adaptation naturelle des espèces par variation et sélection dans le but de réaliser automatiquement une adaptation optimale à l'environnement. À cet effet, Holland met en avant quatre principes :

1. la *dualité* entre un espace génotypique chargé des opérations de variation d'une génération à la suivante et un espace phénotypique dans lequel se produit la sélection ;
2. un *codage* des éléments de $\mathcal{G}$ *par un alphabet discret*, à l'instar de l'alphabet à quatre lettres de l'ADN (Holland milite pour un alphabet binaire pour des raisons théoriques qui seront étudiées à la section 3.6) ;
3. une variation entre générations assurée par des *opérateurs génétiques*, en particulier *sexués* ;
4. le concept de *primitives* (*building blocks*) dans le code génétique qui se combinent pour assurer des interactions complexes nécessaires à la construction d'hypothèses sophistiquées.

Nous avons décrit l'idée fondamentale de dualité dans l'introduction de ce chapitre. Tournons-nous maintenant vers le problème du codage utilisant un alphabet binaire.

Copyright © 2010 Eyrolles.

3.1 La représentation dans l'espace génotypique

Les algorithmes génétiques standards (car il y a des mutants, évidemment) utilisent une représentation dans l'espace des génotypes $\mathcal{G}$, ou encore des génomes, par des chaînes de bits de longueur fixe. Par exemple, chaque hypothèse de $\mathcal{H}$ pourrait être associée à un génome de longueur 12 bits, permettant ainsi de représenter 2^{12} individus. Du point de vue pratique, l'intérêt de ce type de représentation est qu'il se prête à des opérateurs de modification simples (voir section 3.4). Le théorème des schémas (voir section 3.6) revendique une justification théorique de l'emploi de représentations binaires.

La question est de savoir comment choisir la représentation adéquate pour le problème étudié. Cela implique un choix délicat de la taille de la représentation : plus celle-ci est grande, plus le nombre d'hypothèses représentables est élevé, mais plus aussi l'exploration de $\mathcal{G}$ et donc de $\mathcal{H}$ peut être coûteuse. Généralement, le choix du codage par des bits est lié à celui des opérateurs de modification des génomes, dans la mesure où il faut éviter qu'un opérateur puisse, en « cassant » les primitives du codage, engendrer des génomes ne correspondant à aucune hypothèse valide.

3.2 L'algorithme générique

L'algorithme génétique standard règle l'évolution d'une population de génomes sous la pression sélective des performances des hypothèses correspondantes dans $\mathcal{H}$. Il est dans ses grandes lignes décrit par les étapes de l'algorithme 8.

Algorithme 8 : Algorithme génétique standard.

début

1. **Initialisation** de la population $P_{\mathcal{G}}(0)$.

répéter

2. **Évaluation** de la population courante $P_{\mathcal{G}}(t)$ par un calcul de la performance de chaque hypothèse h associée à chaque génome $g \in P_{\mathcal{G}}(t)$.
3. **Sélection** des μ meilleurs génomes de $P_{\mathcal{G}}(t)$ qui vont devenir les « procréateurs » de la génération suivante $P_{\mathcal{G}}(t+1)$.
4. **Modification** des procréateurs par des opérateurs génétiques et obtention de λ « descendants ».
5. **Remplacement** de la population courante par les individus résultant des opérations génétiques de l'étape 4.

jusqu'à *critère d'arrêt satisfait*

fin

Nous allons aborder ces différentes étapes une par une.

3.3 L'initialisation de la population

La plupart des algorithmes opérant sur le génotype organisent l'exploration de l'espace $\mathcal{G}$ (donc indirectement de $\mathcal{H}$) en s'appuyant sur une population d'individus qui évoluent en essayant de maximiser la performance mesurée dans $\mathcal{H}$. Deux questions découlent de cette technique : d'abord, quelle est la *taille optimale de la population* pour favoriser l'exploration efficace de $\mathcal{G}$, ensuite, *comment initialiser* les individus de *la première génération* $P_{\mathcal{G}}(0)$?

Aucune de ces deux questions n'a de réponse théorique bien établie, et le choix de ces deux paramètres est donc souvent le résultat de jugements plus ou moins éclairés sur la nature du problème étudié.

Copyright © 2010 Eyrolles.

La taille de la population de génomes, généralement fixe au cours du déroulement de l'algorithme et notée N, est fréquemment de l'ordre de plusieurs centaines d'individus dans les algorithmes génétiques. En effet, les justifications théoriques de ces algorithmes indiquent qu'il faut des populations très grandes (idéalement infinies) pour que l'algorithme fonctionne optimalement. Cela est aussi souhaitable pour conserver une bonne diversité à la population, permettant ainsi d'échapper aux optima locaux. Inversement, les considérations pratiques de coût calculatoire militent pour l'emploi de populations de taille limitée, sous peine de ne pas pouvoir répéter suffisamment les étapes nécessaires à l'évolution de la population.

Par ailleurs, le choix de la population initiale s'effectue souvent par un tirage aléatoire sur chaque bit des chaînes constituant les individus de $\mathcal{G}$. Cependant, si des connaissances *a priori* sont disponibles sur le problème, il est possible de biaiser la population initiale en sélectionnant des individus représentant des hypothèses prometteuses. Il faut cependant se méfier de ne pas pour autant trop amoindrir la diversité génétique nécessaire à l'exploration par évolution simulée.

3.4 Les opérateurs

Dans le cadre des algorithmes opérant sur le génotype, on appelle *opérateurs* des applications définies sur $\mathcal{G}$ permettant le renouvellement de la population à chaque génération. La variété des opérateurs imaginables est infinie. C'est à la fois un des charmes des travaux exploratoires dans ce domaine, mais aussi une source d'agacement car bien souvent seul le concepteur d'un nouvel opérateur en voit clairement l'intérêt ! John Holland a proposé de suivre les recettes de l'évolution naturelle en adoptant les deux principaux opérateurs connus vers le milieu du XX^{e} siècle[2] : la reproduction sexuée impliquant un *croisement* entre génomes, et la *mutation* par erreur de recopie ou par altération due à un agent extérieur (par exemple des rayons cosmiques).

3.4.1 Les opérateurs de croisement

Un opérateur de croisement est une application de : $\mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G} \times \mathcal{G}$. Il prend donc deux génomes et produit deux nouveaux génomes. De manière analogue au croisement (*cross-over*) de la génétique naturelle, l'idée essentielle est de combiner des caractéristiques des parents dans l'espoir d'obtenir des descendants bénéficiant du meilleur des deux parents. Dans le cas d'une représentation par chaîne de bits de longueur fixe, le croisement consiste à sélectionner des sous-chaînes de chaque génome et à les intervertir. On distingue le croisement à un point et le croisement multipoint.

- Dans **le croisement à un point**, on tire d'abord au hasard un point entre deux bits de la représentation par chaîne, puis en prenant les deux génomes parents, on échange leurs sous-chaînes à partir de ce point (voir figure 8.3 (a)).
- Dans le **croisement multipoint**, on tire d'abord au hasard plusieurs points entre bits de la représentation par chaîne, puis en prenant les deux génomes parents, on échange les sous-chaînes correspondantes (voir figure 8.3 (b)).
- Dans le cas extrême, les sous-chaînes échangées sont réduites à un bit. Chaque bit a alors une certaine probabilité, souvent proche de 0.5, d'être échangé avec le bit correspondant de l'autre génome. On parle alors de **croisement uniforme** (voir figure 8.3 (c)).

[2] Depuis, les recherches sur la génétique ont dévoilé d'autres opérateurs qui semblent guider l'évolution et permettre par exemple une adaptation plus rapide quand des changements de milieu qui ont déjà été rencontrés dans le passé surviennent (voir par exemple le livre de Christopher Wills *La sagesse des gènes* et le livre de science fiction *L'échelle de Darwin* (*Darwin's radio*) de Greg Bear pour un roman basé sur ces idées.

Copyright © 2010 Eyrolles.

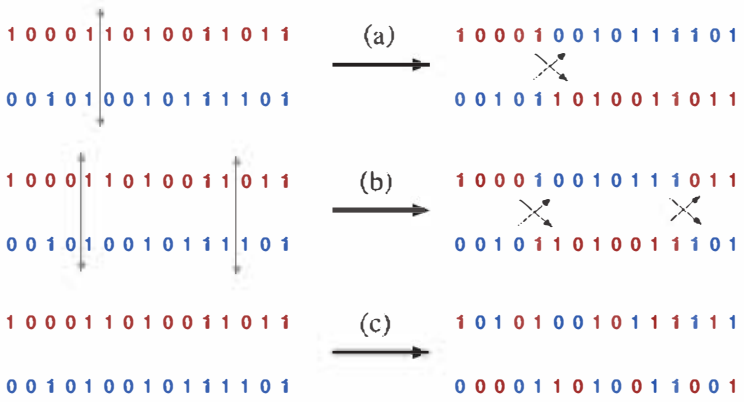


FIG. 8.3: *En (a) un exemple de croisement à un point. En (b) un exemple de croisement multi-point. En (c) un exemple de croisement uniforme.*

3.4.2 Les opérateurs de mutation

Un opérateur de mutation est une application de : $\mathcal{G} \rightarrow \mathcal{G}$. Il prend en entrée un génome (une chaîne de bits) et retourne une autre chaîne de bits. On parle de « mutation » pour indiquer qu'à l'instar des mutations génétiques rencontrées dans la nature, la transformation n'affecte que quelques bits de la chaîne. Un opérateur de mutation fonctionne en changeant aléatoirement, suivant une probabilité p_{mut}, chaque bit de la chaîne passée en argument dont la valeur est inversée (voir figure 8.4). En général la probabilité p_{mut} est faible, juste suffisante pour assurer la mutation de quelques génomes dans la population $P_{\mathcal{G}}(t)$. Mais la proportion idéale de mutations est sujette à débats.

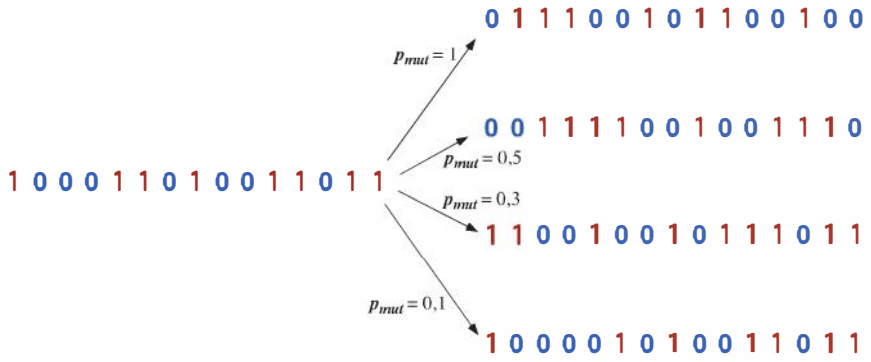


FIG. 8.4: *Un exemple d'opération de mutation sur une chaîne de bits.*

3.4.3 Débat : le rôle des opérateurs

Une manière d'envisager les algorithmes génétiques est de les considérer comme des algorithmes d'optimisation par recherche dans un espace de possibilités. De tels algorithmes font face au dilemme classique « exploitation contre exploration ». L'*exploitation* utilise l'information acquise, par exemple sous la forme d'un gradient local, pour modifier l'estimation courante. Ce genre de stratégie conduit le plus souvent à des optima locaux. L'*exploration* teste des solutions ailleurs que dans le voisinage immédiat de la solution courante, parfois par tirage aléatoire. Cette seconde stratégie permet d'explorer plus largement l'espace de recherche et d'échapper ainsi aux optima

Copyright © 2010 Eyrolles.

locaux. En revanche, en n'exploitant pas l'information acquise, elle est entièrement soumise aux aléas du tirage et ressemble à une marche aléatoire. Certaines méthodes de recherche, comme le *recuit simulé*, implémentent un compromis entre l'exploitation et l'exploration (voir aussi certaines méthodes d'apprentissage par renforcement (chapitre 16). C'est également le cas des algorithmes génétiques par l'usage des opérateurs de croisement et de mutation.

La recherche en génétique laisse supposer que l'échange de matériel génétique entre individus joue un grand rôle dans l'évolution des organismes avancés. Le principe en serait que ce *croisement* permet une recombinaison de primitives déjà testées (blocs d'ADN ou séquences de bits) ouvrant la possiblité de combinaisons encore plus performantes entre ces primitives. Par combinaisons successives de blocs de plus en plus grands, le croisement permettrait ainsi l'évolution vers des organismes de plus en plus performants tendant à être retenus par le processus de sélection naturelle. Il s'agit là d'une sorte d'exploitation des informations contenues dans la population courante de génomes. En revanche, cet opérateur, combiné avec le processus de sélection qui élimine des individus, tend à appauvrir le stock de primitives disponibles. Il est donc nécessaire de disposer d'un autre opérateur introduisant des nouveautés dans ce stock de primitives. C'est le rôle de l'opérateur de *mutation* qui, par la modification aléatoire de bits, conserve un taux de renouvellement, donc de diversité, des génomes. La mutation joue ainsi un rôle d'exploration.

La proportion de ces deux opérateurs dans l'étape de modification d'une génération $P_{\mathcal{G}}(t)$ pour passer à la suivante est sujette à controverse. Pour John Holland, l'opération de croisement est fondamentale puisqu'elle permettrait l'émergence progressive de primitives de plus en plus intéressantes dans le contexte de l'environnement courant. L'opération de mutation ne serait là que pour conserver une source de diversité nécessaire. En revanche, les promoteurs des stratégies d'évolution ont le discours inverse, minimisant le rôle du croisement pour se concentrer sur celui de la mutation, ce qui revient à implémenter une sorte d'algorithme de gradient stochastique. Qui a raison ?

Il est utile de se reporter au *no-free-lunch theorem* discuté dans le chapitre 2. Ce théorème affirmant qu'aucune méthode inductive n'est meilleure qu'une autre indépendamment d'une classe de problèmes donnée a une contrepartie pour ce qui concerne les méthodes d'optimisation par recherche dans un espace de solutions (voir [Wol97]). Ici aussi, aucune méthode d'optimisation n'est meilleure en moyenne sur tous les problèmes d'optimisation possibles. Il en résulte que chaque méthode est adaptée à certaines classes de problèmes et inopérante pour d'autres classes de problèmes. La controverse entre partisans d'une forte proportion d'opérations de croisement et ceux favorisant l'usage de la mutation est donc stérile. Ce qui est intéressant est d'essayer d'identifier les types de problèmes pour lesquels chaque méthode est la plus performante. Dans l'état actuel des connaissances, il semble que l'usage important de l'opération de croisement soit favorable lorsque la fonction à optimiser est relativement continue. À ceux que cette caractérisation très qualitative ne satisfait pas, nous conseillons de consulter les travaux portant sur la notion de « rugosité » du paysage de la fonction à optimiser (voir par exemple [Ang98]).

Pour terminer, on peut noter que les techniques les plus récentes utilisent une adaptation dynamique de la proportion de chaque opérateur en fonction des informations glanées lors de l'exploration de l'espace de recherche. Cela facilite aussi l'adaptation dans des environnements changeant avec le temps.

3.5 La sélection

Les opérateurs étudiés dans la section précédente fonctionnent dans l'espace $\mathcal{G}$ des génotypes. La sélection, qui consiste à calculer la nouvelle population d'hypothèses, s'effectue dans l'espace $\mathcal{H}$ sous la pression de l'environnement décrit dans l'espace des entrées $\mathcal{X}$. On peut y distinguer

Copyright © 2010 Eyrolles.

trois phases : une phase d'*évaluation* dans laquelle chaque hypothèse $h \in \mathcal{H}$ est évaluée (par exemple en mesurant son risque empirique), une phase de *sélection* proprement dite où l'on ne retient qu'une partie de la population courante d'hypothèses, et une phase de *remplacement* qui remplace la population courante $P_{\mathcal{H}}(t)$ par une nouvelle population $P_{\mathcal{H}}(t+1)$.

Il est tentant d'évacuer rapidement *la phase d'évaluation* des hypothèses. Après tout il n'y a rien là de nouveau : il s'agit seulement d'évaluer la qualité ou la performance de chaque hypothèse dans le contexte des données d'apprentissage. Dans le contexte des algorithmes opérant sur le génotype, on parle de *fonction d'évaluation* ou de *fonction de performance* quand on n'adopte pas directement le terme anglais de *fitness function*. Nous voulons cependant souligner l'importance de cette étape dans le contexte des algorithmes opérant sur le génotype. En effet, ces algorithmes impliquent généralement l'évolution de populations d'hypothèses, ce qui signifie que, le plus souvent, de très nombreuses hypothèses doivent être testées. Si l'évaluation d'une hypothèse est coûteuse, comme c'est fréquemment le cas (par exemple dans la recherche d'une structure mécanique devant remplir un certain cahier des charges, ou dans la recherche d'un programme de commande de robot), alors l'évaluation d'une population d'hypothèses peut être, et de loin, l'étape la plus exigeante en ressources calculatoires. Dans ce cas, on peut utiliser, si c'est possible et avec circonspection, une approximation de la fonction d'évaluation pour accélérer le calcul, surtout pour les premières générations. Une fois que toutes les hypothèses de la population courante ont été évaluées, il faut décider quelles hypothèses méritent attention et peuvent servir pour le calcul de la génération suivante. C'est le rôle de l'étape de sélection.

3.5.1 L'étape de sélection

Une fois que la qualité des individus est déterminée grâce à la *fonction de performance* ϕ, il reste à décider quels individus seront sélectionnés pour le calcul de la génération suivante par l'utilisation d'opérateurs génétiques, et quels individus seront conservés ou au contraire éliminés. C'est ce que réalise l'étape de sélection.

D'un certain côté, l'étape de sélection joue un rôle de méthode d'exploitation puisqu'elle utilise l'information rendue disponible par l'évaluation de la population courante des hypothèses pour décider où doit porter l'attention dans la génération suivante.

Conformément à la théorie darwinnienne de l'évolution, la sélection doit tendre à favoriser les meilleurs individus d'une population pour en faire les « parents » de la génération suivante, c'est-à-dire ceux qui transmettront du matériel génétique. Orientant l'exploration de $\mathcal{G}$, ils contrôleront aussi l'exploration de $\mathcal{H}$, avec l'espoir de la conduire dans les régions les plus intéressantes. Il faut cependant régler avec soin la pression sélective, c'est-à-dire la tendance à ne conserver que les meilleurs individus. Si en effet celle-ci est trop élevée, le risque est grand d'une perte de diversité dans la population d'hypothèses, ce qui conduit à une convergence prématurée. Inversement, une pression sélective insuffisante ne conduit pas à une convergence. Plusieurs techniques ont été proposées pour régler ce compromis. Nous en présentons ici trois parmi les plus classiques.

1. **La sélection proportionnelle à la performance** mesurée. Chaque hypothèse $h \in P_{\mathcal{H}}(t)$ est évaluée par la fonction de performance ϕ (phi comme *fitness*). La performance proportionnelle de chaque hypothèse est alors définie comme :

$$\phi_{prop}(h_i) = \frac{\phi(h_i)}{\sum_{h_j \in P_{\mathcal{H}}(t)} \phi(h_j)}$$

Selon la sélection proportionnelle à la performance, la probabilité qu'un individu soit sélectionné pour participer à la production de la génération suivante est égale à sa performance proportionnelle $\phi_{prop}(h)$. Par analogie avec la roulette des casinos, on parle souvent de *sélection par roulette* (voir figure 8.5).

Copyright © 2010 Eyrolles.

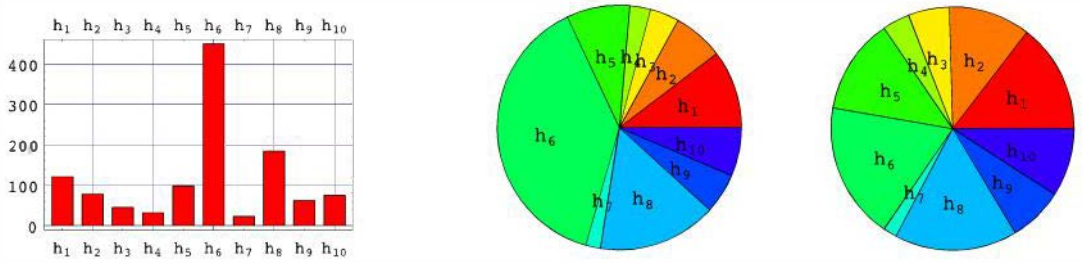


FIG. 8.5: *On considère ici une population de dix hypothèses dont la performance est figurée dans le graphique de gauche. Dans la sélection par roulette (premier diagramme circulaire), à chaque individu est associé un rayon de la roulette dont l'angle est proportionnel à sa performance absolue. Le tirage au hasard selon la roulette donne une espérance de tirage de chaque individu proportionnelle à sa performance absolue. Dans la sélection par roulette proportionnelle au rang (second diagramme circulaire), à chaque individu est associé un rayon de la roulette dont l'angle est proportionnel à son rang dans la population, le meilleur individu étant crédité du plus grand nombre, le second de ce nombre moins un, le dernier de un. Le tirage au hasard selon la roulette donne une espérance de tirage de chaque individu proportionnelle à ce nombre.*

2. La **sélection proportionnelle au rang.** Un inconvénient de la sélection proportionnelle à la performance est que si un ou plusieurs « superindividu(s) » apparaisse(nt), de performance très supérieure aux autres, ils risquent de complètement monopoliser le pool génétique dont sera issu la prochaine génération. Il arrive alors fréquemment que la population stagne dans un optimum local. Pour éviter ce risque de perte de diversité prématurée, on utilise souvent la sélection proportionnelle au rang. Dans ce cas, les individus sont ordonnés dans l'ordre de leurs performances respectives, et la probabilité de participer à la construction de la prochaine génération est déterminée par le rang et non directement par la performance. Par exemple, on utilise une roulette dans laquelle chaque individu est associé à une part proportionnelle à $\frac{rang(h_i)}{N(N-1)/2}$, où N est la taille de la population. L'effet obtenu est celui d'une pression sélective adoucie lorsque la variance des performances est grande, tandis qu'elle est au contraire accentuée si la variance est faible (voir figure 8.5).

3. La **sélection par tournoi.** Les méthodes de sélection décrites ci-dessus exigent l'évaluation de toutes les hypothèses de la population. Une autre méthode peut éviter cela qui de plus se prête bien à une parallélisation des calculs. Supposons que l'on ait besoin de sélectionner μ individus pour engendrer la génération suivante, l'idée est alors d'organiser μ tournois de $t \leq N$ individus chacun. On retient alors le meilleur individu de chaque tournoi. La taille k de ces tournois est un paramètre permettant de contrôler la pression sélective : plus k est grand, plus celle-ci est grande. Dans la pratique, on utilise souvent des valeurs de k comprises entre 2 et 10. Un autre atout de cette méthode est qu'elle est peu sensible aux erreurs de la fonction de performance ϕ.

Une fois que les individus les plus aptes ont été sélectionnés, ils sont utilisés comme entrées des opérateurs génétiques afin de calculer des individus neufs. Supposons que μ individus aient ainsi été retenus : un cas typique est d'utiliser environ $\mu/2$ individus pour des croisements, $\mu/20$ pour de la mutation et le reste pour une simple recopie. Bien sûr ces proportions peuvent varier,

Copyright © 2010 Eyrolles.

et même considérablement. Ainsi dans les *stratégies d'évolution*, le croisement n'est pas employé, tandis que les μ individus sélectionnés sont utilisés pour l'opération de mutation.

3.5.2 L'étape de remplacement

Après que la sélection a retenu μ parents et que ceux-ci ont produit λ descendants, il faut décider de quels individus sera composée la population suivante, sachant qu'en général la taille N de la population reste fixe. Plusieurs variantes sont possibles suivant que des individus de la population précédente sont admis à perdurer dans la nouvelle population ou non.

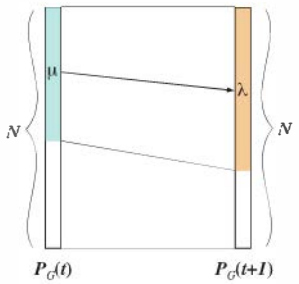


FIG. 8.6: *Le schéma général du remplacement d'une population par la suivante. μ parents sont sélectionnés dans la population courante pour engendrer λ individus qui figureront dans la population suivante. Le reste des individus nécessaires à la complétion de la population est obtenu par des méthodes variées.*

Une méthode extrême consiste à remplacer entièrement la génération précédente par les descendants calculés. On parle alors de *remplacement* (μ, λ) où les μ individus de la génération suivante (ici $\mu = N$) sont les meilleurs des λ individus obtenus par les opérateurs génétiques. L'inconvénient de cette méthode est qu'elle risque de perdre des individus excellents d'une génération et de les remplacer par des individus plus médiocres. C'est pourquoi une autre méthode est plus souvent employée, le *remplacement* $(\mu + \lambda)$, qui consiste à tirer les N nouveaux individus parmi les μ parents sélectionnés plus les λ descendants calculés.

Une alternative à ces méthodes impliquant le remplacement d'une génération par la suivante est la technique de *génération par remplacement local* (*steady-state*). Le principe consiste à sélectionner une sous-population de la population totale puis de déterminer par roulette ou par tournoi un ou des parents dans cette sous-population pour engendrer les remplaçants de cette sous-population, en respectant une population totale constante. Chaque génération n'implique donc que le remplacement de certains individus et non le remplacement de toute la population. L'intérêt de cette technique est de se prêter aisément à une implémentation asynchrone et décentralisée.

3.6 Le théorème des schémas : une explication de la puissance des AGs ?

Les algorithmes opérant sur le génotype découplent le problème de l'expression des hypothèses de celui de son exploration en déléguant celui-ci à un autre espace doté d'opérateurs propres : l'espace des génotypes. Ces algorithmes ont également une source d'inspiration dans les modèles de l'évolution. Depuis leur conception, les algorithmes opérant sur le génotype ont été testés sur de très nombreux domaines d'application et revendiquent des succès parfois spectaculaires sur des problèmes réputés difficiles. Il est temps de s'interroger sur la source de la puissance de ces algorithmes, ce qui devrait permettre aussi, dans l'esprit du *no-free-lunch theorem* (voir 5.1), de mieux cerner les familles de problèmes pour lesquels ils sont les plus adaptés.

Copyright © 2010 Eyrolles.

Depuis quelques années, plusieurs travaux de recherche importants ont été dédiés à l'étude théorique du fonctionnement des algorithmes opérant sur le génotype (voir par exemple [Cer95, Kal99, Rud97, Vos99] ...). Ces travaux utilisent principalement des modélisations par chaînes de Markov, par martingales, ou des concepts tels que la rugosité de l'espace de recherche. Nul doute qu'ils défrichent un terrain qui se révélera fécond et qu'un spécialiste des algorithmes opérant sur le génotype doit connaître. Cependant, il nous semble intéressant de consacrer l'espace limité de ce chapitre à une analyse théorique déjà ancienne : le *théorème des schémas* dû à Holland en 1975. En effet, même si elle a été depuis l'objet de nombreuses études critiques et semble dépassée sur certains points, elle est encore la source de profondes interrogations.

Le point de départ de cette étude théorique réside dans un résultat de la théorie de Markov, à savoir que le nombre de tests nécessaires à l'obtention d'un optimum global par une recherche aléatoire croît exponentiellement avec la dimension L de l'espace de recherche quelle que soit la fonction de performance cible. Qu'est-ce qui fait alors la différence avec des algorithmes utilisant le génotype ? Qu'est-ce qui peut expliquer leur puissance apparente ?

L'une des motivations de John Holland était de montrer pourquoi l'opérateur de croisement est essentiel dans le fonctionnement des algorithmes génétiques. L'idée fondamentale est que les génotypes associés à des hypothèses contiennent dans leur code des *primitives de construction* (*building blocks*). Dans le cadre des algorithmes génétiques, une primitive peut être n'importe quelle sous-séquence de chaîne de bits qui se présente dans la population. Mais cette notion s'étend aussi aux algorithmes utilisant d'autres types de représentations : des sous-arbres dans le cas de la programmation génétique ou des portions d'automates dans le cas de la programmation évolutive. L'hypothèse est que la possession dans leur code génétique de bonnes primitives augmente la performance des individus qui en sont dotés (comme la possession d'un cœur augmente notablement le domaine d'action des organismes qui en sont dotés). De ce fait, ces individus tendent à être davantage sélectionnés pour la reproduction de la nouvelle génération et par conséquent ces primitives tendent à se multiplier et à se répandre dans la population (le théorème des schémas montre que cet accroissement des bonnes primitives est exponentiel en cours d'évolution). Le croisement permettant la combinaison de ces primitives en super-primitives conduirait alors à l'émergence de superindividus.

Plus formellement, on recherche une hypothèse optimale *via* l'exploration de l'espace des génomes $\mathcal{G}$, ce qui revient à chercher un génome optimal. L'hypothèse fondamentale de Holland est qu'un génome optimal est une *superposition* de parties de génomes qu'il appelle des schémas, qui correspondent chacune à des propriétés particulières des hypothèses associées. Ainsi, un schéma est une primitive de construction dans la mesure ou le trait qu'il code dans $\mathcal{H}$ correspond à une propriété mesurable et que c'est l'*addition* de telles propriétés qui rend une hypothèse performante. Il est crucial de noter que cette hypothèse fondamentale n'est nullement triviale et correspond en fait à une contrainte très forte sur la relation entre $\mathcal{H}$ et $\mathcal{G}$. Cela signifie en effet que le codage des hypothèses dans $\mathcal{G}$ respecte une indépendance entre les propriétés pertinentes (pour la performance) des hypothèses, la superposition de deux schémas étant associée à l'addition de deux propriétés mesurables dans $\mathcal{H}$. Trouver un tel code n'est généralement pas une tâche aisée. Nous allons voir que le théorème des schémas, qui découle de cette hypothèse fondamentale, ajoute encore des contraintes sur ces schémas si l'on veut que les algorithmes opérant sur le génotype fonctionnent. La discussion qui suit s'applique aux algorithmes génétiques qui opérent sur des chaînes de bits de longueur fixe. L'extension à d'autres types de représentations a fait l'objet de travaux mais se heurte à des difficultés que nous soulignerons dans les sections correspondantes.

Soit L la longueur des chaîne de bits de $\mathcal{G}$.

Copyright © 2010 Eyrolles.

Définition 8.1 (Schéma)

Un schéma s *est une chaîne sur* $\{0, 1, *\}^L$ *où la signification du symbole* $*$ *est* 0 *ou* 1. *Il y a par conséquent* 3^L *shémas définis sur* $\mathcal{G}$

Un schéma correspond ainsi à un sous-espace projeté de $\mathcal{G}$.

— EXEMPLE —

Par exemple, le schéma s =0*10 défini sur un espace de chaînes de bits de longueur 4 est appariable à l'ensemble des chaînes de bits contenant exactement 0010 et 0110.

Inversement, une chaîne de bits g est appariable à plusieurs schémas : tous ceux qui correspondent aux sous-espaces de $\mathcal{G}$ contenant g. Par exemple, la chaîne de bits 0100 est contenue dans 2^4 schémas (c'est-à-dire dans les sous-espaces de $\mathcal{G}$ associés), dont par exemple **** (dimension 4), *1** (dimension 3), 0*0* (dimension 2) ou encore 01*0 (dimension 1). On peut donc considérer qu'un individu g de $\mathcal{G}$ est représentatif d'un certain nombre de schémas, dont les traits correspondants sont exprimés dans l'hypothèse h associée à g. Appelons $n(s,t)$ le nombre de schémas s représentés par une population de génomes à l'instant t. Le théorème des schémas décrit l'espérance de $n(s,t+1)$ en fonction de $n(s,t)$ et d'autres propriétés de la population et des paramètres de l'algorithme.

L'évolution de la population dans un algorithme génétique dépend de l'étape de sélection, puis des opérations de croisement et de mutation. Considérons d'abord l'effet de la sélection. Par abus de langage, on parlera de la performance $\phi(g)$ d'un génome g, quand en toute rigueur il s'agit de la performance mesurable de l'hypothèse associée h. Soit donc $\overline{\phi}(t)$ la performance moyenne des individus de la population $P_{\mathcal{G}}(t)$ à l'instant t :

$$\overline{\phi}(t) = \frac{1}{N} \sum_{g \in P_{\mathcal{G}}(t)} \phi(g)$$

D'après la règle de sélection proportionnelle à la performance, la probabilité pour un génome g_i d'être sélectionné est :

$$\mathsf{P}(g_i) = \frac{\phi(g_i)}{\sum_{g_j \in P_{\mathcal{G}}(t)} \phi(g_j)} = \frac{\phi(g_i)}{N\overline{\phi}(t)} \tag{8.1}$$

La probabilité de sélectionner un génome représentatif d'un schéma s est alors :

$$\mathsf{P}(g \in s) = \sum_{g \in s \cap P_{\mathcal{G}}(t)} \frac{\phi(g)}{N\overline{\phi}(t)} = \frac{\phi(s,t)}{N\overline{\phi}(t)} n(s,t) \tag{8.2}$$

où $\phi(s,t)$ est la performance moyenne des génomes de la population $P_{\mathcal{G}}(t)$ représentatifs du schéma s :

$$\phi(s,t) = \frac{\sum_{g \in s \cap P_{\mathcal{G}}(t)} \phi(g)}{n(s,t)}$$

L'équation 8.2 vaut pour un génome. L'espérance d'appartenance au schéma s pour l'ensemble des individus de la population $P_{\mathcal{G}}(t)$ résulte de N fois le même tirage indépendant, d'où la valeur de l'espérance :

$$E[n(s,t+1)] = \frac{\phi(s,t)}{\overline{\phi}(t)} n(s,t) \tag{8.3}$$

Copyright © 2010 Eyrolles.

Si l'on regarde seulement l'effet de la sélection d'une génération à la suivante, l'équation 8.3 montre que l'espérance du nombre de génomes représentatifs d'un schéma s à l'instant $t+1$ est proportionnelle à la perfomance moyenne de ce schéma à l'instant t et inversement proportionnelle à la performance moyenne de la population de génomes à l'instant t. Les schémas dont la valeur moyenne est supérieure à la performance moyenne générale devraient donc voir le nombre de leurs représentants augmenter au cours des générations. Inversement, les schémas associés à des performances médiocres devraient tendre à disparaître. Globalement, l'effet des algorithmes génétiques devrait être de révéler et de favoriser les primitives de construction intéressantes pour la tâche courante.

Cependant, cet effet sélectif n'est pas suffisant pour construire des superindividus. Il faut tenir compte des opérateurs de croisement et de mutation. Or ceux-ci, à côté de leurs éventuels effets bénéfiques, ont aussi des effets destructeurs. De fait, sous l'effet d'une mutation, un génome peut ne plus appartenir à un schéma donné. La probabilité que cela arrive dépend du nombre de bits définissant un schéma. Un croisement (ici on ne considère que les croisements à un point) peut de la même manière produire des descendants n'appartenant plus à un schéma donné. Il suffit pour cela que le point de croisement se trouve situé « à l'intérieur » du schéma considéré.

Pour énoncer complètement le théorème des schémas, il nous faut donc deux définitions exprimant des caractéristiques des schémas importantes lorsque les opérateurs génétiques sont employés.

Définition 8.2 (Ordre d'un schéma)

On appelle ordre $o(s)$ *d'un schéma s le complément à L de la dimension du sous-espace correspondant. C'est aussi le nombre de caractères dont la valeur est spécifiée dans s.*

— Exemple —

C'est ainsi que le schéma $**0**$ est d'ordre 1, alors qu'il représente un sous-espace de dimension 4 de $\{0,1,*\}^5$ et le schéma $10*11$ d'ordre 4 .

Définition 8.3 (Longueur caractéristique d'un schéma)

On appelle longueur caractéristique $d(s)$ *d'un schéma s, dans sa représentation comme une chaîne de L caractères de l'alphabet $\{0,1,*\}$, le nombre de bits entre le symbole non égal à * le plus à gauche dans la chaîne et celui le plus à droite, augmenté de 1.*

— Exemple —

C'est ainsi que $d(1***1)=4$, $d(*0*1*)=2$, $d(10*1*)=3$ et $d(**0**)=1$.

En notant P_{crois} la probabilité d'utilisation de l'opérateur de croisement et P_{mut} la probabilité de mutation d'un bit, on obtient alors l'équation suivante, appelée aussi *théorème des schémas.*

Théorème 8.1 (Le théorème des schémas (Holland, 1975))

L'espérance du nombre de génomes représentants un schéma s suit l'équation :

$$E[n(s,t+1)] = \frac{\phi(s,t)}{\bar{\phi}(t)} n(s,t) \left(1 - \mathsf{P}_{crois} \frac{d(s)}{L-1}\right) (1-\mathsf{P}_{mut})^{o(s)} \tag{8.4}$$

Copyright © 2010 Eyrolles.

Le premier terme de droite décrit l'effet de la sélection, conformément à l'équation 8.3. Le deuxième décrit l'effet destructeur du croisement. Il exprime la probabilité de survivance à l'effet d'un opérateur de croisement. Plus la longueur caractéristique d'un schéma est grande, plus la probabilité qu'un opérateur de croisement détruise l'appartenance à ce schéma d'un descendant est grande. Le troisième terme correspond à l'effet destructeur des mutations, dont la probabilité dépend de l'ordre du schéma considéré. Ainsi les effets destructeurs des opérateurs de croisement et de mutation s'accroissent avec l'ordre et la longueur caractéristique des schémas. Holland en conclut que l'effet des algorithmes génétiques est de favoriser l'apparition des représentants des schémas les plus performants, et ceci d'autant plus que ceux-ci sont courts (ordre et longueur caractéristique faibles).

Plusieurs remarques peuvent être formulées :

1. D'après ce théorème, l'idéal serait de disposer d'un codage des génomes tel que les primitives utiles soient décrites par des chaînes de bits courtes. Malheureusement, cela revient à connaître à l'avance le codage adéquat, c'est-à-dire les propriétés du problème que l'on cherche à résoudre. Les détracteurs de ce théorème se sont ingéniés à concevoir des problèmes dits « trompeurs » (*deceptive problems*) qui ne satisfont pas les codages usuels et mettent donc les algorithmes génétiques en défaut. On pourra noter ici que certains auteurs, inspirés aussi par la biologie, ont proposé des *opérateurs d'inversion* permettant le réarrangement dans l'espace des schémas afin de pouvoir réduire leur longueur caractéristique et de les soustraire aux effets destructeurs du croisement.
2. Le théorème des schémas est insuffisant car, bien qu'il prenne en compte l'effet destructeur des opérateurs génétiques, il ne traduit pas leurs éventuels effets bénéfiques, en particulier ceux concernant le croisement qui permet la combinaison de plusieurs schémas en superschémas. Cette étude-là en est encore à ces prolégomènes.
3. Une autre source d'insuffisance du théorème des schémas réside dans son incapacité à prendre en compte ce que l'on appelle l'*épistasie*, qui signifie que certaines parties du génome peuvent avoir une influence sur l'expression d'autres parties du génome. Plus généralement, toutes les non-linéarités dans le fonctionnement ou l'expression du génome échappent à l'analyse par schémas.
4. Une idée profonde de Holland est que les algorithmes génétiques, en calculant des populations successives de génomes, effectuent de fait une *recherche implicite dans l'espace des schémas*, pour découvrir les plus intéressants. Si c'est là effectivement la source de puissance des algorithmes génétiques, alors il faut favoriser cette recherche implicite qui se réalise grâce à l'appartenance des génomes à plusieurs schémas, lesquels se trouvent ainsi mis en compétition. Il faut donc maximiser le nombre de schémas qui sont représentés par une population de génomes de taille donnée. Cette remarque conduit à préférer les alphabets binaires, d'où le choix de la représentation prônée par Holland. En effet, par exemple, la chaîne 0101 est représentative de 2^4 schémas dans un alphabet binaire, alors que le nombre total de schémas est de 3^L. Dans un alphabet n-aire, la chaîne 0101 est toujours représentative de 2^4 schémas, tandis que leur nombre total est de n^4. On voit que l'exploration implicite permise par une population de taille donnée est ainsi plus efficace dans le cas d'un alphabet binaire.

L'étude décrite ci-dessus concerne des représentations séquentielles de taille fixe. Que se passe-t-il dans le cas de l'utilisation de représentations dont la taille peut varier et dont les primitives peuvent être réarrangées dans le génome ? On a encore beaucoup de mal à l'analyser, mais il existe plusieurs propositions très intéressantes d'algorithmes utilisant des représentations alternatives et qui ont été largement testées sur de nombreux problèmes.

Copyright © 2010 Eyrolles.

4. Les stratégies d'évolution

La technique dite des « stratégies d'évolution » a été initiée par Fogel, Owens et Walsh dans les années 1960 (voir [FOW66]). Elle consiste à utiliser une représentation par automates à états finis[3] pour l'espace $\mathcal{G}$ des génotypes et l'opérateur de mutation pour modifier les expressions de cet espace. Dans leur version originale, les stratégies d'évolution furent utilisées pour des tâches de prédiction de séquences.

La méthode démarre en produisant une population initiale d'automates à états finis. Ceux-ci sont parcourus en fonction de l'état courant et de l'entrée correspondant au caractère courant de la séquence. L'évaluation de l'automate tient compte de l'accord ou non de la prédiction faite alors par l'automate par rapport au caractère suivant de la séquence. De cette manière, chaque automate est évalué en fonction de la séquence entière à prédire. La sélection utilise une roulette proportionnelle à la performance. Les descendants de la génération courante sont calculés à partir des parents sélectionnés en leur appliquant des mutations qui peuvent ajouter des états, en retirer, changer les transitions entre états, changer l'état initial et changer les caractères de sortie. Très souvent, la génération suivante est obtenue par un procédé de remplacement élitiste consistant à conserver la meilleure moitié de la génération courante et en complétant par les descendants obtenus à partir de cette meilleure moitié. Souvent aussi, le taux de mutation est réduit au fur et à mesure des générations.

Les développements récents de cette technique ont élargi les recettes employées, mais en conservant la mutation comme seul opérateur génétique (voir [Fog98, Fog99]).

Les figures 8.7 et 8.8 illustrent l'évolution d'une population de cinquante automates à états finis ayant au maximum vingt états sur une tâche de prédition de la séquence :

`1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1`

La figure 8.7 montre la population initiale, tandis que la figure 8.8 montre l'évolution du meilleur automate toutes les dix générations depuis la génération 10 jusqu'à la génération 200 (voir [Jac01], chapitre 6 pour les détails).

5. La programmation génétique

Est-il possible d'envisager de produire automatiquement des programmes par un mécanisme d'évolution simulée ? Ici, l'espace $\mathcal{H}$ des hypothèses deviendrait un espace de programmes, la performance de chacun d'eux étant mesurable par test dans un environnement représentatif des tâches à réaliser. L'idée est séduisante, elle semble cependant complètement irréalisable. Tout programmeur a déjà eu l'occasion de pester contre une virgule ou un point-virgule mal placé responsable du dysfonctionnement d'un programme, alors comment une exploration au hasard, ou presque, de l'espace des expressions possibles pourrait parvenir à produire un programme valide, encore plus d'un programme solution, et qui plus est efficace ? !

Encore une fois la solution réside dans une représentation adéquate des programmes se prêtant bien à l'utilisation d'opérateurs d'exploration efficaces. En intelligence artificielle, dire ceci revient évidemment à énoncer une tautologie. Personne n'a de recette pour trouver une telle représentation ni les opérateurs associés. Cependant, toute une école de chercheurs, à la suite de pionniers tels que Cramer [Cra85] et Koza [Koz92, Koz94, KA99], a proposé une nouvelle

[3] Voir le chapitre 7.

Copyright © 2010 Eyrolles.

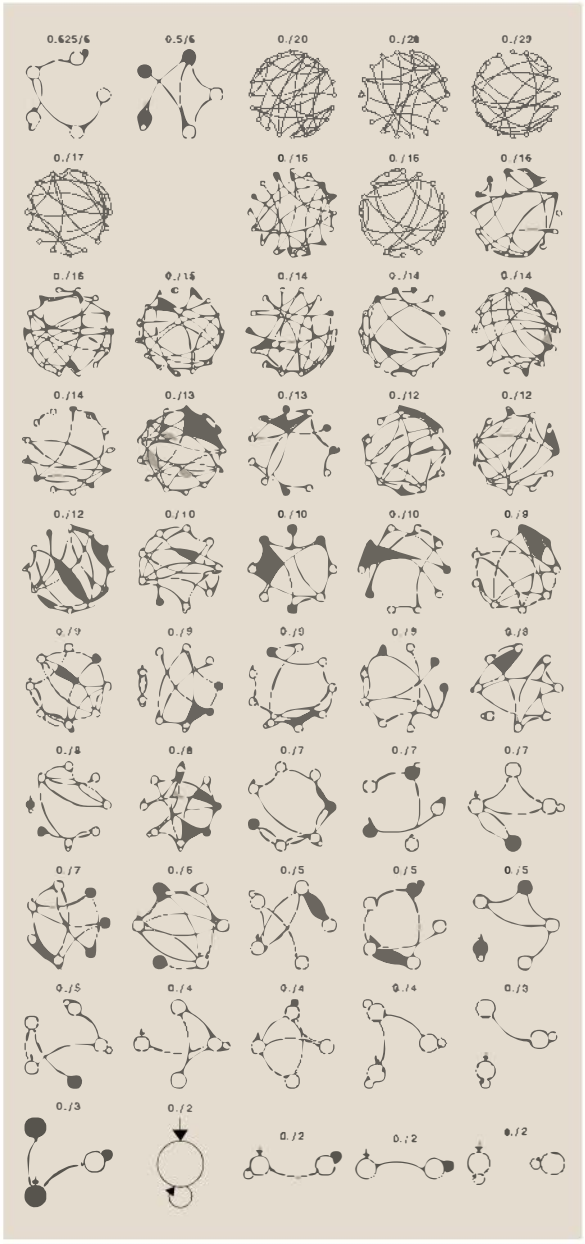


FIG. 8.7: *La génération 0 d'automates à états finis.*

Copyright © 2010 Eyrolles.

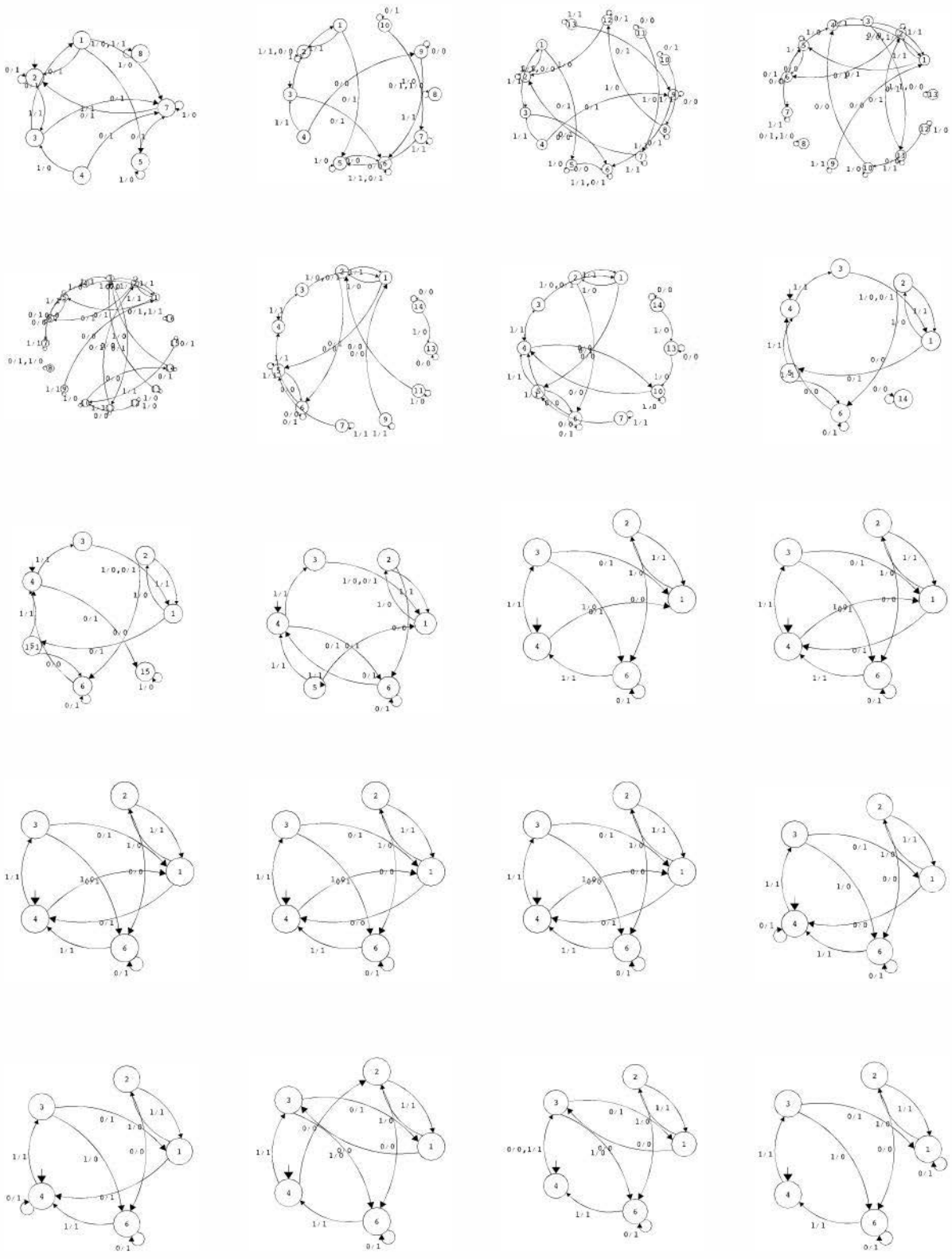


FIG. 8.8: *Le meilleur automate toutes les dix générations de la génération 10 à la génération 200. Dès la génération 70, le meilleur automate prédit parfaitement la séquence. Le reste de l'évolution conduit à une simplification du meilleur automate.*

Copyright © 2010 Eyrolles.

approche qui a été testée de manière prometteuse sur une grande variété de problèmes. L'idée fondamentale est de s'appuyer sur une représentation des programmes par arbres et d'utiliser des langages de programmation dont la syntaxe est simple et telle que des manipulations syntaxiques également simples produisent encore des programmes licites. Cette idée, déjà largement exploitée par Douglas Lenat dans son programme AM (Automated Mathematician [Len78]), conduit à s'intéresser à des langages de programmation tels que Lisp ou des dérivés comme Scheme ou Mathematica.

5.1 La représentation des programmes

La différence essentielle entre les algorithmes génétiques et la programmation génétique est que dans ces derniers les structures de programmation ne sont pas codées par des génomes linéaires, mais par des termes ou des expressions symboliques correspondant à un langage de programmation. Les unités objets de mutation ou de recombinaison ne sont plus des caractères ou des sous-séquences de chaînes, mais des modules fonctionnels (comme en Lisp) représentés par des arbres. Une telle représentation conduit à une redéfinition des opérateurs génétiques. Ainsi, les primitives de représentation sont maintenant des termes (feuilles de l'arbre) ou des sous-termes (sous-arbres) qui peuvent être soit remplacés (mutation) soit échangés entre des arbres représentant des programmes (recombinaison). En comparaison avec les algorithmes utilisant des génomes linéaires, cela ouvre des possibilités inédites. Par exemple, la taille des génomes n'est plus fixée ni même en principe limitée en taille ou en profondeur. Les adaptations possibles des structures de programmes sont ainsi potentiellement illimitées, correspondant à de grandes variations tant dans la sémantique que dans les aspects algorithmiques des programmes. Certains pensent même que cela ouvre des perspectives radicalement nouvelles par rapport à la biologie dont le matériel génétique est codé sur des chromosomes linéaires.

5.1.1 L'ensemble des terminaux et des fonctions

En programmation génétique, les fonctions et les terminaux sont les primitives servant à décrire les programmes. Qualitativement, on peut dire que les *terminaux* fournissent des valeurs au programme tandis que les *fonctions* traitent ces valeurs. Les terminaux correspondent aux feuilles des arbres, tandis que les fonctions correspondent aux nœuds internes.

Définition 8.4 (Ensemble des terminaux)

*L'*ensemble des terminaux *est constitué par les entrées des programmes, les constantes définies* a priori *et les fonctions d'arité nulle ayant des effets de bord calculés en cours d'exécution.*

Les terminaux retournent une valeur numérique sans recevoir d'arguments en entrée. En général ces terminaux sont constitués des valeurs numériques rencontrées dans l'échantillon d'apprentissage en plus de constantes jugées utiles par le concepteur. D'autres valeurs numériques pourront être calculées par l'usage de fonctions prenant des terminaux en argument.

Définition 8.5 (Ensemble des fonctions)

*L'*ensemble des fonctions *est constitué des déclarations, opérateurs et fonctions disponibles pour le système.*

— Exemple —

Voici quelques exemples de fonctions qui peuvent être utilisées en programmation génétique.

- *Fonctions booléennes.* Exemples : `AND`, `OR`, `NOT`, `XOR`

Copyright © 2010 Eyrolles.

- *Fonctions arithmétiques.* Exemples : `PLUS, MINUS, MULTIPLY, DIVIDE`
- *Fonctions transcendantales.* Exemples : fonctions trigonométriques et logarithmiques.
- *Formes conditionnelles.* Exemples : `IF ... THEN ... ELSE`
- *Structures itératives.* Exemples : boucles `REPEAT, WHILE, DO,...`
- *Sous-routines.* Le domaine des fonctions utilisables est en fait bien plus considérable que les fonctions prédéfinies d'un langage et peuvent inclure n'importe quelle primitive définie par le programmeur, par exemple en réponse à un problème spécifique. Nous en verrons un exemple dans la section 5.5 à propos d'un robot simulé.

5.1.2 Le choix de l'ensemble des terminaux et des fonctions

Le choix de l'ensemble des terminaux et des fonctions doit obéir à quelques critères. Il doit être assez riche pour permettre l'expression d'une solution au problème. Mais il ne doit pas l'être trop sous peine de rendre trop coûteuse l'exploration de l'espace de recherche. En général, il est préférable de commencer avec un ensemble restreint et de l'enrichir si l'expérience en fait sentir le besoin. Une autre propriété est nécessaire : la propriété de *clôture*. Chaque fonction de l'ensemble des fonctions doit en effet pouvoir traiter toutes les valeurs qu'il peut recevoir en entrée. Ceci est très important car les entrées d'une fonction donnée peuvent varier énormément au cours de l'évolution simulée de programmes. Il ne faut pas que cela provoque un arrêt du programme (comme cela peut arriver si une division par zéro est demandée). La propriété de clôture est plus facile à vérifier avec des langages dont la syntaxe est uniforme comme c'est le cas de Lisp ; elle exige cependant une certaine attention.

5.1.3 Phénotypes et génotypes en programmation génétique

Les algorithmes utilisant le génotype pour la recherche d'une bonne solution ou hypothèse découplent les problèmes d'expressivité du langage de représentation (ceux de coût mémoire ou d'exécution qui sont propres à l'espace $\mathcal{H}$ des phénotypes) des problèmes de recherche dans l'espace des programmes (qui sont du ressort de l'espace $\mathcal{G}$ des génotypes). Dans le cas de la programmation génétique cependant, la distinction entre les deux espaces n'existe généralement pas, et la recherche par évolution simulée s'opère directement dans l'espace $\mathcal{H}$ des programmes exécutables et testables. Les avocats de la programmation génétique estiment par conséquent que la représentation et les opérateurs dont ils disposent sont efficaces directement dans $\mathcal{H}$.

5.1.4 Initialisation d'une population de programmes

Au début d'un cycle d'évolution simulée, il faut produire une population initiale de programmes. Cela est généralement réalisé par un tirage aléatoire d'arbres représentant des programmes. Il est souvent spécifié une profondeur limite ou un nombre maximal de nœuds que les arbres produits aléatoirement ne peuvent dépasser. Le processus de création des arbres est récursif, avec à chaque pas la création d'un nœud de l'arbre par sélection aléatoire d'une fonction ou d'un terminal. Dans ce dernier cas, une feuille de l'arbre est atteinte. Plusieurs méthodes de génération d'arbres ont été étudiées. Nous renvoyons le lecteur par exemple à [BNKF98] pp.118-122.

5.2 Les opérateurs génétiques sur les programmes

Comme pour les algorithmes génétiques, les trois opérateurs les plus employés en programmation génétique sont : la mutation, le croisement et la reproduction.

Copyright © 2010 Eyrolles.

1. **La mutation**. C'est un opérateur défini de $\mathcal{G} \rightarrow \mathcal{G}$ opérant donc sur un seul génome et produisant un autre génome par altération aléatoire du premier. Plus précisément, la mutation sur une représentation par arbre opère en sélectionnant au hasard un sous-arbre et en le remplaçant par un autre sous-arbre engendré aléatoirement, en respectant les limites de profondeur ou de taille définies pour l'initialisation de la population (voir figure 8.9).

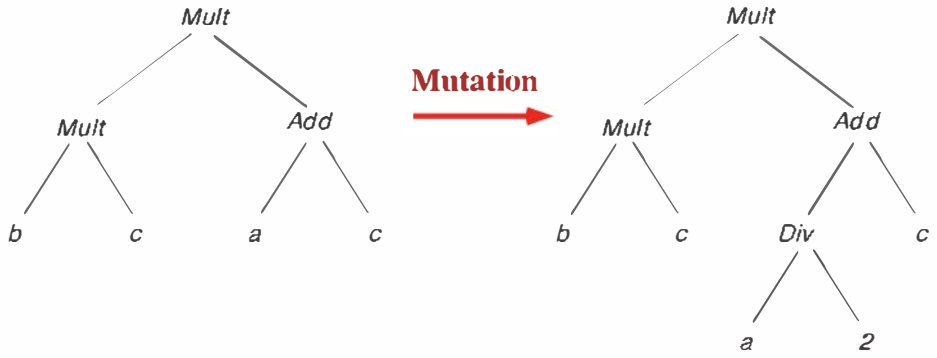


FIG. 8.9: *Exemple d'une opération de mutation.*

2. **Le croisement** est un opérateur défini de $\mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G} \times \mathcal{G}$ produisant deux génomes à partir de la donnée de deux génomes en entrée. Il sélectionne au hasard un sous-arbre dans chacun des génomes fournis en entrée et les échange pour produire deux nouveaux génomes (voir figure 8.10). Il est évident que cette opération doit respecter la propriété de clôture définie plus haut (5.1.2).

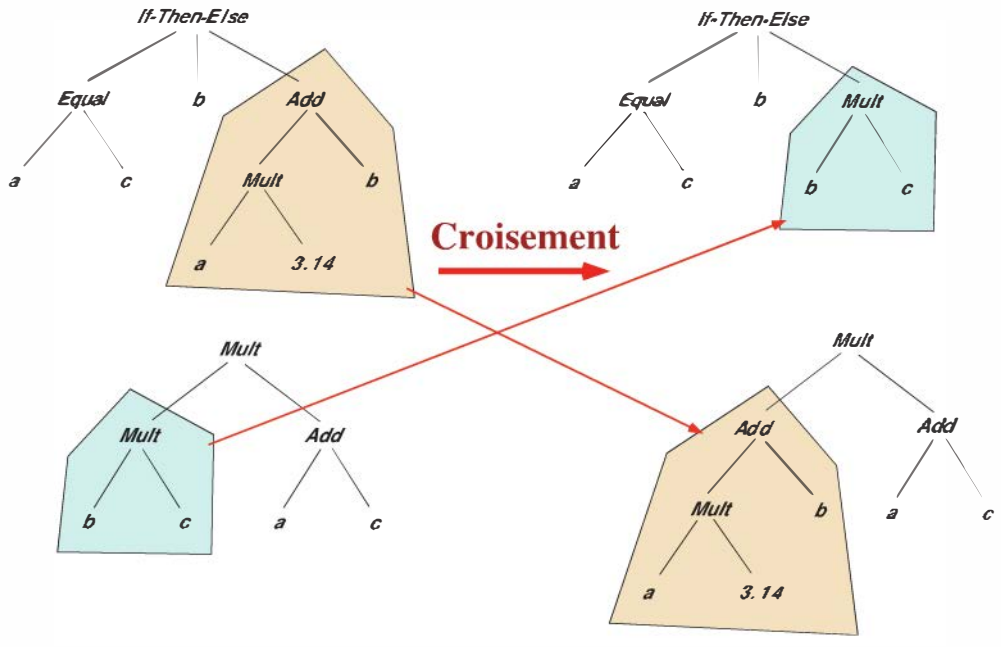


FIG. 8.10: *Exemple d'une opération de croisement.*

3. **La reproduction** consiste simplement en une recopie d'un individu d'une génération à la suivante.

Copyright © 2010 Eyrolles.

5.3 Évaluation et sélection

L'évaluation des génomes et leur sélection suivent les mêmes principes que ceux des algorithmes génétiques. La performance d'un génome est définie comme la performance mesurée de la solution correspondante dans $\mathcal{H}$. La sélection des individus utilisés pour produire la génération suivante se fait également selon les mêmes méthodes.

5.4 Le fonctionnement

Le fonctionnement de la programmation génétique suit les mêmes étapes que les algorithmes génétiques.

5.5 Illustrations

Nous illustrons brièvement le fonctionnement de la programmation génétique sur deux problèmes. Le premier est un problème d'induction classique. Il s'agit, étant donné un échantillon d'exemples et de contre-exemples décrits dans un espace numérique $\mathcal{X} = \mathbb{R}^d$, de trouver l'équation d'une séparatrice entre les deux classes d'objets. Le second problème concerne l'induction du programme de contrôle d'un robot simulé (censé modéliser une fourmi) dans un environnement simplifié dans lequel se trouve de la « nourriture ».

5.5.1 Induction d'une séparatrice entre deux classes

L'expérience rapportée dans cette section concerne l'induction d'une surface séparatrice dans un espace d'exemples appartenant à deux classes et décrits par des descripteurs numériques. Le principe de l'expérience est le suivant : une fonction de séparation cible est choisie aléatoirement par combinaison d'un certain nombre fixé de descripteurs en utilisant un répertoire de fonctions fourni *a priori*. Par exemple, en fixant le nombre de descripteurs à deux et le jeu de fonctions à : $+$, $-$, $\times$, $\div$, il est possible de produire la séparatrice : $f(x) = \frac{x_1 \times x_2}{x_1 + x_2}$. Un ensemble de points de $\mathcal{X}$ tirés aléatoirement sont alors étiquetés par comparaison avec la fonction séparatrice. Par exemple, tous les points pour lesquels $f(x) \geq 0$ sont étiquetés '+'. On obtient ainsi un ensemble d'apprentissage employé par un algorithme d'apprentissage s'appuyant sur la programmation génétique. Cet algorithme utilise un jeu de terminaux incluant des constantes et des variables représentant les descripteurs (qui ne font pas tous forcément partie de la fonction cible : ce sont des descripteurs non pertinents) et un ensemble de fonctions incluant par exemple : $+$, $-$, $\times$; $\div$, cos, sin, ... La fonction de performance mesure le risque empirique, par exemple par un critère entropique (voir les arbres de décision dans le chapitre 13), une fonction de coût quadratique ou simplement le nombre d'exemples mal classés. De plus, pour favoriser l'obtention d'expressions simples, un terme de pénalité a été introduit dans la fonction de performance, prenant en compte le nombre de nœuds dans les arbres produits. Une fois que l'algorithme a trouvé une séparatrice, on évalue son risque réel en comptant le nombre d'exemples mal classés sur un échantillon de test.

Dans les expériences réalisées [BTC96], les paramètres avaient les valeurs suivantes :

- Taille de l'*ensemble d'apprentissage* : 200 exemples générés aléatoirement
- Taille de l'*ensemble de test* : 10000 exemples générés aléatoirement
- Taille de la *population* : 500 (i.e. 500 fonctions de décision en compétition dans la population)
- Nombre de *générations* : 50 (pour chaque exécution)
- Nombre d'*exécutions* : 1000
- *Profondeur* maximale des arbres : 8
- Ensemble de *fonctions* : $+, -, *, \div$ $(cos, sin, exp, log, \sqrt{})$

Copyright © 2010 Eyrolles.

- Taux des *opérateurs* :
  - reproduction : 10 %
  - croisement : 45 %
  - mutation : 45 %

Une exécution de l'algorithme peut ainsi produire une séquence de populations de fonctions séparatrices dont la meilleure évolue comme suit à différents instants :

$$\frac{tog(z)}{\frac{z}{y}} + x - sin(z) \quad \rightsquigarrow \quad \frac{\frac{z}{y}}{exp(x)} - \sqrt{sin(z)} \quad \rightsquigarrow \quad x * \frac{z}{y} + \sqrt{x} \quad \rightsquigarrow \quad y * \frac{x}{z} \quad \rightsquigarrow \quad \frac{x.y}{y+z}$$

5.5.2 Induction d'un programme de contrôle de robot simulé

Nous voulons simplement ici montrer comment un programme de contrôle simple en robotique peut être produit par programmation génétique. Le travail décrit succinctement dans cette section est tiré du livre de Christian Jacob *Illustrating evolutionary computation with Mathematica* [Jac01].

On s'intéresse à un robot placé dans un univers bidimensionnel consistant en une grille de 18 × 18 cases tel que celui de la figure 8.11. Quarante-quatre cases de nourriture ou de phéromones sont distribuées sur la grille. Ces cases sont organisées suivant un chemin avec d'abord des séquences de nourriture ininterrompues, puis de plus en plus de cases intermittentes de phéromones qui aident à orienter le robot fourmi. Le but du robot est d'essayer de visiter toutes les cases contenant de la nourriture en quatre-cents actions maximum. Les actions disponibles sont au nombre de quatre :

- `advance` : avancer d'une case en avant ;
- `turnLeft` : tourner de 90° à gauche ;
- `turnRight` : tourner de 90° à droite ;
- `nop` : pas d'action.

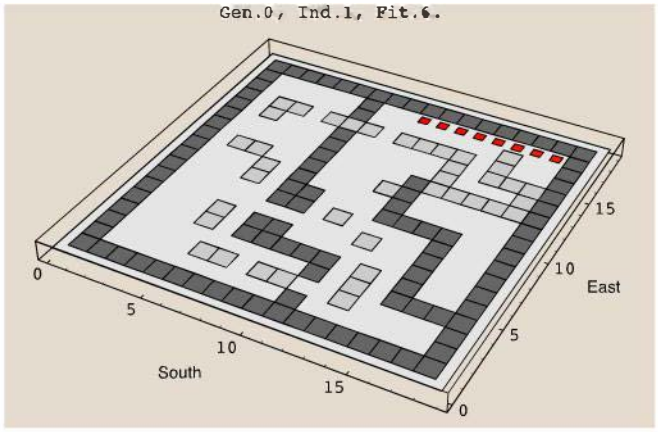


FIG. 8.11: *Le monde du robot fourmi. Les cases en noir correspondent à des murs infranchissables, celles en gris foncé à de la nourriture, celles en gris clair à des traces de phéromone (d'après [Jac01], p.401).*

Un robot peut également déterminer son action en fonction des conditions environnantes. Pour cela un robot est équipé d'un senseur qui fournit des informations sur la case qui se trouve directement devant : « mur », « nourriture », « phéromone » ou « poussière ». Cela autorise deux commandes conditionnelles :

Copyright © 2010 Eyrolles.

- `ifSensor[signal][A]` : l'action `A` est exécutée si le robot est devant une case correspondant à `signal` ;
- `ifSensor[signal][A,B]` : l'action `A` est exécutée si le robot est devant une case correspondant à `signal`, sinon c'est l'action `B` qui est exécutée.

Ces actions ou commandes sont intégrées dans une expression `seq` correspondant à une séquence. Chaque action ou commande, en comptant aussi les requêtes au senseur, est comptée comme une action. Le robot s'arrête s'il a trouvé toutes les cases contenant de la nourriture ou s'il a dépassé le nombre maximal d'actions autorisées, ici 400.

Un programme est représenté par une séquence de commandes ou d'actions qui peuvent éventuellement correspondre à des itérations si les commandes `stop` et `again` sont incluses.

EXEMPLE **Exemple de programmes**

Par exemple, le programme dont l'expression est :

`ifSensor[food][seq[advance,again],seq[nop]]`

correspond à une boucle permettant de suivre des cases consécutives de nourriture. La requête `ifSensor` sera exécutée aussi longtemps qu'il y a aura de la nourriture juste devant le robot déclenchant alors une action `advance`. Sinon, la boucle est quittée et le robot continuera d'exécuter les commandes suivantes s'il y en a.

La performance d'un programme de contrôle de robot h est mesurée par le nombre n de cases de nourriture visitées pondéré par un terme prenant en compte la complexité du programme, afin de favoriser les programmes simples :

$$\phi(h) = \begin{cases} n+1 & \text{si } d_h \leq d_{max}, \\ \frac{d_{max}}{d_h} \cdot n + 1 & \text{si } d_h > d_{max}. \end{cases}$$

Les opérateurs génétiques utilisés dans les expériences décrites par Ch. Jacob incluent la mutation et le croisement, ainsi que des délétions, des duplications, des permutations, etc. Ces opérateurs sont définis par des moules (*templates*) qui spécifient quels motifs dans un programme peuvent être remplacés et par quoi. Par ailleurs, il existe un programme de simplification des programmes obtenus qui est appliqué à chaque génération. Ce type de programme de simplification est essentiel dans la programmation génétique.

Les figures 8.13 et 8.15 donnent une idée du genre de programmes qui peuvent être obtenus dans un processus d'évolution jouant sur une population d'individus de cinquante programmes. Le meilleur individu de la cinquième génération va droit devant avant d'être arrêté par le mur. À la génération 9 (voir la figure 8.12) le meilleur programme ne diffère du meilleur de la génération 5 que par une instruction, la dernière, mais celle-ci correspond à tourner à gauche si un mur est rencontré. La trajectoire obtenue à partir du point initial est alors une boucle. Des améliorations sensibles apparaissent de temps en temps lors des générations suivantes, jusqu'à l'obtention à la génération 59 d'un programme permettant, sur le problème posé, de passer sur toutes les cases de nourriture (voir figures 8.14 et 8.15).

La figure 8.16 permet de visualiser l'évolution des performances des cinquante programmes sur les trente-neuf premières générations. On y observe une amélioration graduelle des performances moyennes avec des variations assez brutales de la performance du meilleur individu. La partie droite de cette figure montre l'évolution de la performance du meilleur individu, de la performance moyenne et de la performance du pire individu sur quatre-vingt-six générations.

Sur ces expériences, Ch. Jacob rapporte que sur vingt évolutions simulées, chacune limitée à cent générations, et impliquant une population de cinquante individus engendrés aléatoirement,

Copyright © 2010 Eyrolles.

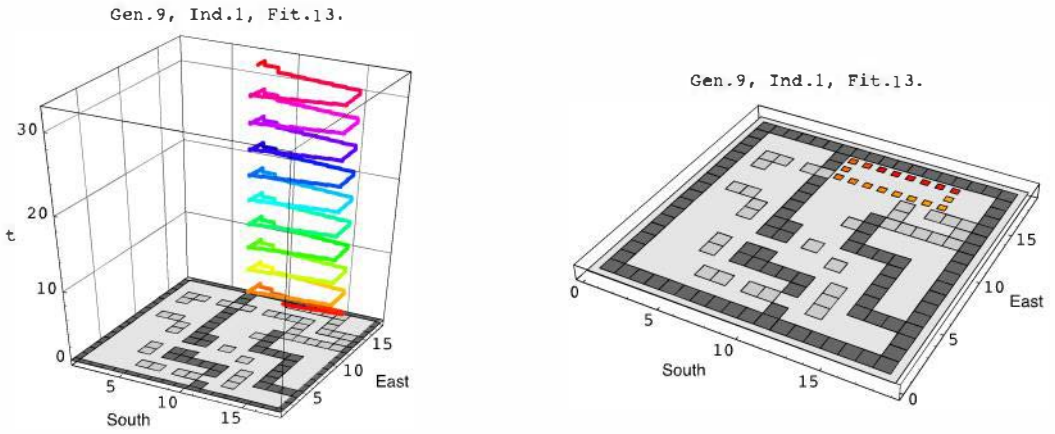


FIG. 8.12: *Le meilleur individu de la génération 9 correspond à un robot qui avance tout droit sauf lorsqu'il se trouve face à un mur, auquel cas il tourne à gauche (d'après [Jac01], p.401).*

```
AntTracker
 [seq[advance, ifSensor[dust][seq[stop], advance],
      ifSensor[dust][seq[turnLeft, again], nop],
       advance,
       ifSensor[wall][
   seq[turnLeft, again], advance]]]
```

FIG. 8.13: *Le programme du meilleur individu de la génération 9 (d'après [Jac01], p.401).*

dix programmes ont finalement été obtenus qui visitent les quarante-quatre cases de nourriture en moins de quatre cents actions. Il serait évidemment intéressant de savoir comment se comportent ces programmes sur la même tâche mais dans un environnement différent.

5.6 Une analyse de la programmation génétique

Le théorème des schémas est difficile à appliquer dans le cas de la programmation génétique notamment parce que les primitives de construction peuvent migrer d'un endroit à l'autre en cours d'évolution et aussi parce que la représentation des individus est de longueur variable. Nous reportons le lecteur intéressé à [BNKF98] pp.145 et suivantes pour une discussion sur les différentes tentatives d'analyse formelle de la programmation génétique. Il y a là encore un territoire à explorer.

Finalement, il est important de noter que dans le cas de la programmation génétique, les espaces des hypothèses $\mathcal{H}$ et des génotypes $\mathcal{G}$ sont de fait confondus. C'est en effet le même espace qui sert à l'exploration (rôle de $\mathcal{G}$) et à l'évaluation (rôle de $\mathcal{H}$). Un programme est ainsi à la fois l'objet subissant les opérateurs génétiques et l'objet s'exprimant dans l'univers pour y produire une certaine performance.

Copyright © 2010 Eyrolles.

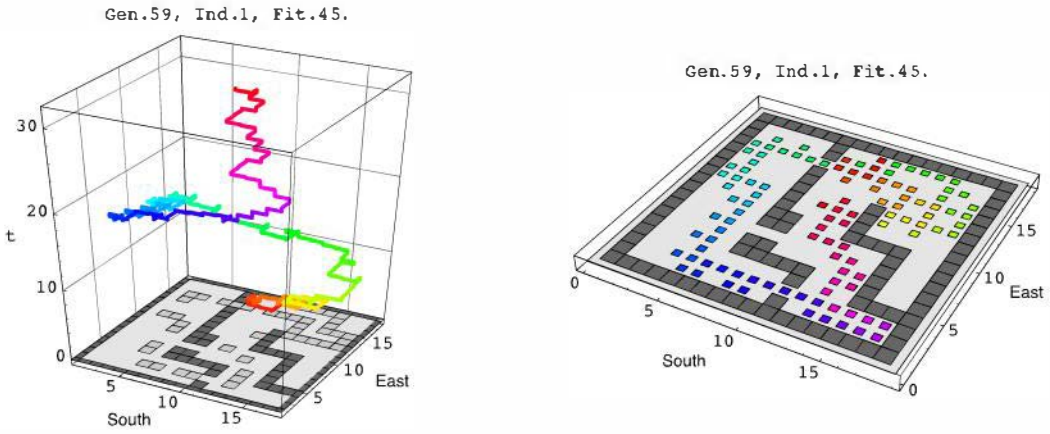


FIG. 8.14: *Le comportement du meilleur programme de la génération 59. Toutes les cases de nourriture sont visitées sans détours inutiles (d'après [Jac01], p.401).*

```
AntTracker
 [seq[{},
  ifSensor[wall][seq[turnLeft, turnLeft, again],
      seq[{},
     ifSensor[wall][seq[turnLeft, again], turnRight],
     ifSensor[wall][seq[turnLeft, again], advance]]],
      ifSensor[dust][seq[turnLeft, again]],
      ifSensor[wall][seq[ifSensor[dust][advance]],
       seq[ifSensor[dust][seq[turnLeft, again]],
         ifSensor[wall][seq[turnLeft, again], advance]]]]
```

FIG. 8.15: *Le programme du meilleur individu de la génération 59 (d'après [Jac01], p.401).*

6. La coévolution

Les approches mentionnées jusqu'ici cherchent à optimiser une fonction de performance fixée une fois pour toutes par l'objectif et par l'environnement connu par l'intermédiaire d'un échantillon d'apprentissage. La recherche d'une bonne hypothèse s'effectue alors dans l'espace des hypothèses $\mathcal{H}$ grâce à des manipulations dans l'espace génotypique $\mathcal{G}$. On espère que la représentation choisie pour $\mathcal{G}$ et les opérateurs de transformation sélectionnés rendront la recherche efficace. Cependant, il est peut-être possible de réussir encore mieux en *adaptant dynamiquement la fonction de performance* que l'algorithme cherche à optimiser. L'idée serait de faciliter l'exploration de l'espace d'hypothèses en cherchant d'abord des optima faciles à trouver, puis en raffinant progressivement le critère objectif. D'une certaine manière, il s'agit là de l'idée mise en œuvre dans le recuit simulé par exemple. Un processus de recherche de moins en moins stochastique permettait de focaliser le système sur les zones les plus prometteuses de l'espace de recherche. Dans le cas des algorithmes opérant sur le génotype par évolution de populations, il est possible de réaliser cette idée d'une autre manière inspirée par l'observation de l'évolution dans la nature.

Copyright © 2010 Eyrolles.

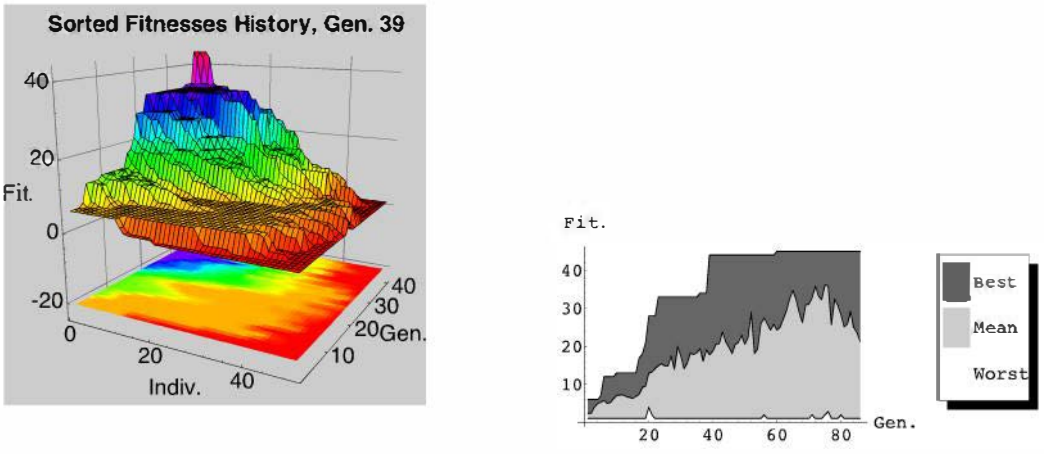


FIG. 8.16: *À gauche, l'évolution des performances des cinquante programmes sur les trente-neuf premières générations. À droite, l'évolution des performances du meilleur individu, de la moyenne et du pire individu, sur quatre-vingt-cinq générations (d'après [Jac01], p.418).*

Le principe est de faire dépendre la fonction de performance utilisée pour la sélection des meilleurs individus d'une population $P_{\mathcal{G}_1}(t)$ d'une autre population $P_{\mathcal{G}_2}(t)$, elle-même sous la dépendance d'une fonction de performance commandée par $P_{\mathcal{G}_1}(t)$. La métaphore biologique est celle de la coévolution proie-prédateur, dans laquelle au fur et à mesure que les prédateurs améliorent leur performance contre les proies, celles-ci modifient leur comportement pour diminuer leur vulnérabilité. Il s'ensuit une sorte de course aux armements qui à la fois rend plus performants les individus des deux populations, mais aussi les rend davantage spécialisés face à un environnement.

D'un point de vue pratique, plusieurs scénarios permettent de réaliser une fonction de performance dépendant d'une autre population. Par exemple, dans certains cas, il est possible de tester chaque individu contre tous les individus de la population adverse. On peut aussi tester chaque individu contre le meilleur adversaire. Ou encore, on peut tester des individus pris au hasard dans une population contre des adversaires également tirés au hasard dans l'autre population. De la sorte, chaque population devrait améliorer sa performance face à la fonction de performance évoluant dynamiquement.

Cette approche a été étudiée dans le cadre de la modélisation de phénomènes naturels. Mais elle a aussi été utilisée pour des problèmes d'optimisation. Ainsi, [Hil92, AP93, Sim94] ont documenté des améliorations parfois considérables obtenus par coévolution dans plusieurs problèmes d'optimisation. Par exemple, [Ron96] a étudié le problème du robot fourmi décrit plus haut dans lequel les morceaux de nourriture eux-mêmes évoluaient afin déchapper à une population de robots fourmis. Il a montré alors une évolution plus rapide du comportement des robots fourmis. Cette approche a également été utilisée par Jannink [Jan94] pour optimiser des générateurs de nombres pseudo-aléatoires. L'idée était qu'une population de testeurs de caractère aléatoire de séquences produites par une population de générateurs coévoluait avec celle-ci.

L'idée de coévolution est encore préliminaire et ouverte à recherche. Une voie à explorer est celle de coopération plutôt que celle de confrontation. La théorie des jeux est un guide utile en ce domaine.

Copyright © 2010 Eyrolles.

Citons pour finir les travaux sur l'évolution des opérateurs génétiques eux-mêmes pendant le processus d'optimisation.

6.1 Un exemple d'écologie : les systèmes de classeurs

Un intéressant exemple d'écologie d'hypothèses est fourni par le systèmes de classeurs (*classifier systems*) imaginé par John Holland en 1975, bien avant que l'on parle de coévolution.

Le principe est celui d'un agent placé dans un environnement dynamique, contrôlé par un système expert dont les règles servent à interpréter le monde et à prendre des décisions qui sont alors mises en œuvre à l'aide d'effecteurs. Les règles appelées *classeurs* ont le format suivant : `if` *condition* `then` *action*.

Comme dans un système expert classique, chaque règle peut être déclenchée par la situation courante qui résulte à la fois de la perception et des inférences conduites par le système. Cette situation est représentée dans une mémoire à court terme ou mémoire de travail. À chaque instant l'ensemble des règles courantes (la population) examine la mémoire de travail pour voir si ses conditions de déclenchement sont réalisées. Contrairement à un système expert classique où seule une règle est déclenchée après sélection, ici toutes les règles déclenchables sont déclenchées et contribuent ainsi à l'inférence courante. Celle-ci modifie alors la situation courante et le cycle recommence (voir figure 8.17).

D'un certain côté, on peut concevoir ce système de règles comme une écologie avec des règles se spécialisant dans certains types de tâches et contribuant ainsi à la performance globale. Le problème est de faire évoluer un tel système écologique. Pour ce faire, Holland a proposé un système de rétribution de mérite fonctionnant un peu comme une économie où des entreprises en compétition pour des contrats rémunèrent leurs sous-traitants, devenant plus riches si le contrat est remporté, ou s'appauvrissant dans le cas contraire. Cette redistribution des mérites est assez proche des méthodes d'apprentissage par renforcement avec trace d'éligibilité (voir chapitre 16). Le mérite attribué de la sorte sert alors de mesure de performance pour chaque règle et le fonctionnement classique d'un algorithme génétique est alors utilisé pour déterminer les parents de la génération suivante.

Nous n'entrons pas davantage dans les détails, reportant le lecteur intéressé à [BGH89]. Malgré la séduction exercée par cette idée, elle n'a pas permis jusqu'à présent d'obtenir de résultats probants, notamment parce que la redistribution des mérites est très lente et ne permet pas une évolution rapide de la population. Peut-être que les prochaines générations de machines et des conceptions neuves sur la coévolution donneront un jour une nouvelle jeunesse à cette idée séduisante car elle correspond à l'évolution d'une société d'individus spécialisés fonctionnant sur le mode de compétition/coopération.

Notes historiques et approfondissements

La métaphore de l'évolution comme base possible pour des algorithmes d'apprentissage en intelligence artificielle remonte aux années 1950 et 1960. Dans les années 1960, Rechenberg introduisit l'approche des *stratégies d'évolution* qu'il utilisa pour optimiser des fonctions à paramètres à valeurs réelles pour la conception de systèmes mécaniques. Cette idée a été développée par Schwefel en Allemagne dans les années 1970. Elle connaît un grand regain d'intérêt ces dernières années car elle s'applique assez naturellement à de nombreux problèmes d'optimisation. Par ailleurs, elle évite certains problèmes de codage que l'on rencontre avec les algorithmes basés

Copyright © 2010 Eyrolles.

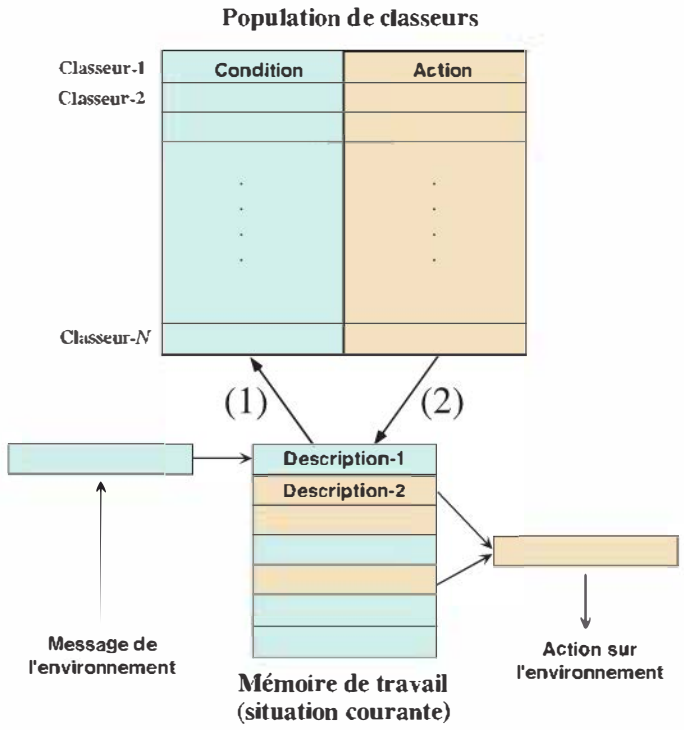


FIG. 8.17: *Schéma d'un système de classeurs. En (1), les messages de la mémoire de travail sont comparés aux règles du système. En (2), les règles déclenchées inscrivent des messages dans la mémoire de travail, correspondant soit à des interprétations sur le monde, soit à des décisions qui seront mises en œuvre par des effecteurs.*

sur une représentation par chaînes de bits. Nous n'en avons pas parlé par manque de place, mais cette approche a fait l'objet récemment de nombreux articles dans des conférences et des revues portant sur les algorithmes d'évolution simulée.

Les premiers modèles de processus d'adaptation portant sur des populations d'individus par *algorithmes génétiques* sont dus à John Holland et ont été popularisés par son ouvrage [Hol75]. Outre une modélisation de l'adaptation par utilisation d'opérateurs génétiques et de sélection des meilleurs individus d'une population à la suivante, Holland proposa une analyse théorique de ces algorithmes par le théorème des schémas. Son intérêt principal alors et maintenant portait sur l'évolution de population de règles (*classifier systems*). Les algorithmes génétiques connurent un vif développement dans les années 1980 et lancèrent véritablement le mouvement pour les algorithmes évolutionnaires.

L'approche initiée par Fogel, Owens et Walsh dans les années 1960 sur les *stratégies d'évolution* dans le cadre de populations d'automates à états finis reste minoritaire dans les recherches et les applications actuelles. Il n'est pas impossible que le développement des applications en prédiction temporelle ne relance cette voie.

Une approche liée aux stratégies d'évolution et qui étend les algorithmes génétiques de Holland à des représentations plus larges est celle de la *programmation génétique* dont Koza s'est fait un avocat puissant [Koz92, Koz94, KA99]. Des conférences et des revues sont maintenant entièrement consacrées à cette famille d'algorithmes.

Globalement, la technique des algorithmes d'évolution artificielle est devenue très populaire, autant que les réseaux connexionnistes. Elle le doit en particulier au fait qu'elle peut s'appliquer

Copyright © 2010 Eyrolles.

dans de très nombreux contextes pour lesquels les techniques d'optimisation traditionnelles sont mises en échec par manque de « régularité » de l'espace des fonctions cible. Les recherches sont actives, tant pour investiguer de nouvelles variations algorithmiques que pour essayer d'en faire une analyse théorique. Cette dernière est difficile. Elle est liée à la fois aux théories de l'optimisation et à des théories sur les processus de diffusion par exemple.

Résumé

Les algorithmes évolutionnaires exploitent un découplage de l'espace d'hypothèses en un *espace phénotypique* (dans lequel les hypothèses ont une description portant sur le monde et sont donc évaluables) et un *espace génotypique* (dans lequel les hypothèses, représentées par des génomes, subissent un processus d'évolution simulée). L'espoir est que les *opérateurs génétiques* utilisés pour faire évoluer les hypothèses, en particulier le croisement et la mutation, fassent évoluer la population d'hypothèses vers les régions de $\mathcal{H}$ où se trouvent les hypothèses les meilleures.
L'un des intérêts des algorithmes évolutionnaires est leur large domaine d'applications puisqu'ils permettent d'explorer des espaces d'hypothèses dans lesquels la fonction de performance attachée aux hypothèses n'est pas dérivable. C'est pourquoi ils sont fréquemment utilisés.
Plusieurs familles d'algorithmes ont été développées, dont principalement : les *algorithmes génétiques* utilisant plutôt une représentation binaire des génomes, les *stratégies d'évolution* reposant sur une représentation par automates, et la *programmation génétique* travaillant dans un espace de programmes représentable par des arbres. Il existe également une extension appelée *coévolution* pour laquelle la fonction d'évaluation des hypothèses elle-même résulte d'un processus évolutif dans un jeu de compétition.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Troisième partie

Apprentissage par optimisation

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 9

L'apprentissage de modèles linéaires

Ce chapitre traite de l'apprentissage de concepts s'exprimant par des formules linéaires dans des espaces numériques. Ainsi, dans le cas de la régression, on cherche la droite rendant au mieux compte des données ; dans le cas de la discrimination, on cherche un hyperplan permettant de séparer au mieux les exemples des contre-exemples.

Ces méthodes ont une longue histoire, d'abord en statistique, depuis les années 1930, puis en informatique puisqu'on les a proposées comme l'archétype de l'apprentissage artificiel dès le début des années 1960. Elles ont toujours une grande importance pour plusieurs raisons. La première est la simplicité du calcul de la décision. Ensuite, elles sont robustes au bruit et elles peuvent être plus performantes que des méthodes non linéaires, en particulier lorsque le jeu de données d'apprentissage est réduit. Finalement, on dispose d'une variété d'algorithmes d'apprentissage, dont la plupart permettent une interprétation sous la forme de l'optimisation d'un critère compréhensible sur l'ensemble d'apprentissage.

Ces méthodes ont été généralisées à la fin des années 1980, en particulier par les réseaux connexionnistes. De plus, l'approche consistant à utiliser un espace de redescription par des transformations non linéaires, grâce par exemple aux méthodes à noyaux, leur a donnéé une nouvelle et vigoureuse impulsion depuis le milieu des années 1990.

La compréhension des méthodes linéaires est un préalable essentiel à l'étude des méthodes non linéaires et c'est pourquoi nous leurs consacrons un chapitre à part entière.

Copyright © 2010 Eyrolles.

Sommaire

1 Introduction 300
2 Régression linéaire 301
2.1 Le problème de la régression 301
2.2 Le cas de la régression linéaire 302
2.3 Quelques propriétés intéressantes 304
2.3.1 Interprétabilité et réduction de dimension 304
2.3.2 Théorème de Gauss-Markov 305
2.3.3 Complexité calculatoire 305
2.3.4 Et avec une autre fonction d'erreur ? 306
3 Séparatrices linéaires 306
3.1 Généralités 306
3.1.1 Hyperplans séparateurs et discriminants dans un problème à deux classes 307
3.1.2 Un peu de géométrie dans $\mathbb{R}^d$ 308
3.2 L'apprentissage d'un hyperplan pour discriminer deux classes 309
3.2.1 Une solution globale 309
3.2.2 Une méthode itérative : l'algorithme de Ho et Kashyap 311
3.3 Un autre calcul : l'algorithme du perceptron 312
3.3.1 L'algorithme stochastique 313
3.3.2 Convergence 313
3.3.3 La version non stochastique et son interprétation 313
3.4 L'hyperplan discriminant de Fisher 315
3.5 Et pour plus de deux classes? 316
4 Modèles linéaires par morceaux et combinaisons de modèles locaux 317
4.1 La régression linéaire par morceaux 318
4.2 La régression spline 318
4.3 La combinaison linéaire de fonctions de base 319
5 La recherche des facteurs pertinents 320
5.1 Méthodes de détermination ou de pondération des facteurs pertinents 320
5.2 Optimisation du critère inductif par chemin de régularisation 323

Copyright © 2010 Eyrolles.

1. Introduction

La partie 2 de cet ouvrage était dédiée aux apprentissages dans lesquels l'espace des hypothèses est structuré par une relation d'ordre de généralité induite par la relation d'inclusion dans l'espace des entrées $\mathcal{X}$. Une telle relation d'inclusion signifie que les hypothèses manipulées sont du type booléen, c'est-à-dire qu'il s'agit d'apprentissage de concept, ce qui en limite la portée. En revanche l'existence d'une relation d'ordre permet d'utiliser des algorithmes d'exploration de $\mathcal{H}$ puissants. Un des points communs des apprentissages étudiés était que les hypothèses s'exprimaient dans des langages expressifs : logiques et grammaires en particulier.

Dans ce chapitre, nous examinons des langages d'hypothèses plus restreints, dans lesquels, que les hypothèses soient de type discriminatif (fonctions de décision) ou de type génératif (distributions de probabilité), elles s'expriment sous formes de combinaisons linéaires d'éléments de

base. L'absence d'une relation d'ordre de généralité dans $\mathcal{H}$ conduit à l'utilisation d'algorithmes d'optimisation classiques, par exemple, par descente de gradient, pour trouver une (ou plusieurs) hypothèse(s) optimisant le critère inductif choisi.

2. Régression linéaire

Supposons que nous voulions un système capable de prédire la durée de vol d'un volatile, étant donnés son âge, son poids, la température de l'air, l'heure de la journée et la latitude. Ou bien, supposons que, pour résoudre un problème de navigation automatique d'un robot mobile, nous cherchions à déterminer l'angle de braquage à donner aux roues pour éviter des obstacles, en fonction de mesures telles que la vitesse de l'engin, son poids, sa position fournie par GPS, etc. On appelle problèmes de *régression* de tels problèmes, dans lesquels la sortie (ici, consommation électrique ou angle de braquage) est numérique, généralement un vecteur de réels, supposé dépendre de la valeur d'un certain nombre de facteurs en entrée.

2.1 Le problème de la régression

Le vecteur d'entrée $\mathbf{x} = (x_1, x_2, \ldots, x_d)^\top$ est souvent appelé *variable indépendante*, tandis que le vecteur de sortie $\mathbf{y}$ est appelé *variable dépendante*[1]. On formalise le problème en supposant que la sortie résulte de la somme d'une fonction déterministe f de l'entrée et d'un bruit aléatoire ε :

$$\mathbf{y} = f(\mathbf{x}) + \varepsilon \tag{9.1}$$

où $f(\mathbf{x})$ est la fonction inconnue que nous souhaitons approcher par un estimateur $h(\mathbf{x}|\mathbf{w})$, où h est défini à l'aide d'un vecteur $\mathbf{w}$ de paramètres.

Si l'on suppose que le bruit ε est un phénomène gaussien de moyenne nulle et de variance constante σ^2, c'est-à-dire $\varepsilon = \mathcal{N}(0, \sigma^2)$, alors, en plaçant notre estimateur $h(\cdot)$ à la place de la fonction inconnue, on devrait avoir la densité conditionnelle réelle $\mathbf{p}(\mathbf{y}|\mathbf{x})$ vérifiant :

$$\mathbf{p}(\mathbf{y}|\mathbf{x}) = \mathcal{N}(h(\mathbf{x}|\mathbf{w}), \sigma^2) \tag{9.2}$$

On peut estimer le vecteur de paramètres $\mathbf{w}$ grâce au principe de maximisation de la vraisemblance. On suppose que les couples $(\mathbf{x}_t, \mathbf{y}_t)$ de l'échantillon d'apprentissage sont tirés par tirages indépendants d'une distribution de probabilité jointe inconnue $\mathbf{p}(\mathbf{x}, \mathbf{y})$, qui peut s'écrire :

$$\mathbf{p}(\mathbf{x}, \mathbf{y}) = \mathbf{p}(\mathbf{y}|\mathbf{x})\, \mathbf{p}(\mathbf{x})$$

où $\mathbf{p}(\mathbf{y}|\mathbf{x})$ est la probabilité de la sortie étant donnée l'entrée, et $\mathbf{p}(\mathbf{x})$ est la densité de probabilité sur les entrées.

Étant donné un échantillon d'apprentissage $\mathcal{S} = \langle(\mathbf{x}_t, \mathbf{y}_t)\rangle_{1 \le t \le m}$ supposé tiré de manière indépendante et identiquement distribué (i.i.d.), sa *log vraisemblance* se définit par :

$$\mathcal{L}(\mathbf{w}|\mathcal{S}) = \log \prod_{t=1}^{t=m} \mathbf{p}(\mathbf{x}_t, \mathbf{y}_t) = \log \prod_{t=1}^{t=m} \mathbf{p}(\mathbf{y}_t|\mathbf{x}_t) + \log \prod_{t=1}^{t=m} \mathbf{p}(\mathbf{x}_t) \tag{9.3}$$

[1] Le vecteur y pourrait être considéré comme une supervision et noté u ou u. Nous gardons la notation y qui est conforme à l'usage.

Copyright © 2010 Eyrolles.

Nous cherchons à la maximiser. Le second terme ne dépendant pas de l'estimateur peut être ignoré pour cette maximisation, d'où :

$$\begin{aligned}\mathcal{L}(\mathbf{w}|\mathcal{S}) &= \log \prod_{t=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{[\mathbf{y}_t - h(\mathbf{x}_t|\mathbf{w}))]^2}{2\sigma^2}\right] \\ &= \log\left\{\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^m \exp\left[-\frac{1}{2\sigma^2}\sum_{t=1}^{m}[\mathbf{y}_t - h(\mathbf{x}_t|\mathbf{w}))]^2\right]\right\} \\ &= -m\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}\sum_{t=1}^{m}[\mathbf{y}_t - h(\mathbf{x}_t|\mathbf{w}))]^2\end{aligned} \tag{9.4}$$

Le premier terme étant indépendant du vecteur de paramètres $\mathbf{w}$ peut être éliminé, de même que le facteur $1/\sigma^2$. Maximiser l'expression résultante revient alors à minimiser la somme des carrés des erreurs (SCE) :

$$\mathrm{SCE}(\mathbf{w}|\mathcal{S}) = \frac{1}{2}\sum_{t=1}^{m}[\mathbf{y}_t - h(\mathbf{x}_t|\mathbf{w}))]^2 \tag{9.5}$$

ce qui est la fonction d'erreur la plus utilisée. Le vecteur de paramètres $\mathbf{w}$ minimisant cette expression est alors naturellement appelé l'*estimation des moindres carrés* (*least square estimate* en anglais)[2].

2.2 Le cas de la régression linéaire

Dans le cas de la régression linéaire, on suppose que la fonction de régression $\mathrm{E}(\mathbf{y}|\mathbf{x})$ est linéaire par rapport au paramètre $\boldsymbol{w}$ ou qu'elle peut raisonnablement être approchée par un modèle linéaire de $\boldsymbol{w}$:

$$h(\mathbf{x}_t|w_0, w_1, \ldots, w_d) = w_0 + w_1 x_t^1 + w_2 x_t^2 + \ldots + w_d x_t^d \tag{9.6}$$

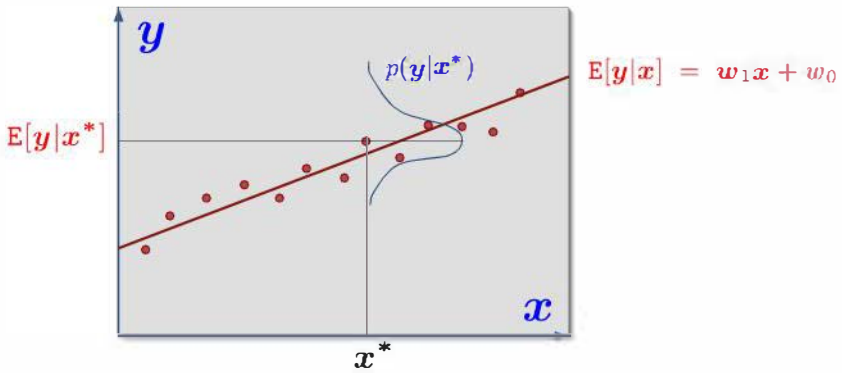


FIG. 9.1: *La régression linéaire en supposant un bruit gaussien de moyenne nulle.*

On suppose dans la suite que la sortie y est numérique. Le cas d'un vecteur de sortie $\mathbf{y}$ est similaire mais plus compliqué à écrire (il suffit de considérer un problème de régression par composante du vecteur de sortie $\mathbf{y}$).

[2] Ce critère est encore valide, même si les $\mathbf{x}_t$ ne sont pas tirés aléatoirement, du moment que les $\mathbf{y}_t$ sont conditionnellement indépendants étant données les entrées.

Copyright © 2010 Eyrolles.

En définissant les vecteurs et matrice suivants :

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \dots & x_1^d \\ 1 & x_2^1 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_m^1 & x_m^2 & \dots & x_m^d \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \mathbf{S}_y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

le critère de la somme des carrés des erreurs s'écrit alors :

$$\text{SCE}(\mathbf{w}|\mathcal{S}) = \frac{1}{2}(\mathbf{S}_y - \mathbf{X}\mathbf{w})^\top (\mathbf{S}_y - \mathbf{X}\mathbf{w}) \tag{9.7}$$

Il suffit de prendre la dérivée de la somme des carrés des erreurs (équation 9.5) par rapport à $\mathbf{w}$ pour obtenir les équations :

$$\begin{aligned} \frac{\partial \text{SCE}}{\partial \mathbf{w}} &= -\mathbf{X}^\top(\mathbf{S}_y - \mathbf{X}\mathbf{w}) \\ \frac{\partial^2 \text{SCE}}{\partial \mathbf{w} \partial \mathbf{w}^\top} &= -\mathbf{X}^\top \mathbf{X} \end{aligned}$$

En supposant que la matrice $\mathbf{X}$ est non singulière, et donc que $\mathbf{X}^\top\mathbf{X}$ est positive définie, et en posant que la dérivée première est nulle, on obtient :

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{S}_y \tag{9.8}$$

soit encore :

$$\begin{aligned} \sum_t y_t &= m\,w_0 + w_1 \sum_t x_t^1 + w_2 \sum_t x_t^2 + \dots + + w_d \sum_t x_t^d \\ \sum_t y_t x_t^1 &= w_0 \sum_t x_t^1 + w_1 \sum_t (x_t^1)^2 + w_2 \sum_t x_t^1 x_t^2 + \dots + w_d \sum_t x_t^1 x_t^d \\ \sum_t y_t x_t^2 &= w_0 \sum_t x_t^2 + w_1 \sum_t x_t^1 x_t^2 + w_2 \sum_t (x_t^2)^2 + \dots + w_d \sum_t x_t^1 x_t^d \\ &\vdots \\ \sum_t y_t x_t^d &= w_0 \sum_t x_t^d + w_1 \sum_t x_t^1 x_t^d + w_2 \sum_t x_t^2 x_t^d + \dots + w_d \sum_t (x_t^d)^2 \end{aligned}$$

à partir de quoi on peut calculer l'unique solution par :

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{S}_y \tag{9.9}$$

Il s'agit là de l'une des équations les plus célèbres de l'algèbre linéaire : le *filtre*[3] *linéaire de Wiener*. Qualitativement, cette équation stipule qu'il faut prendre en compte les corrélations entre les entrées et les sorties (partie $\mathbf{X}^\top \mathbf{S}_y$ de l'équation) mais en prenant garde de ne pas surestimer les corrélations entre les entrées (partie $(\mathbf{X}^\top \mathbf{X})^{-1}$ de 9.9).

La valeur $\hat{y}_n$ prédite pour une entrée $\mathbf{x}_n$ est donc : $\hat{y}_n = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{S}_y \mathbf{x}_n$. Il est possible d'estimer le bruit par l'estimateur non biaisé :

$$\sigma^2 \approx \frac{1}{m - d - 1} \sum_m (y_n - \mathbf{w}^\top \mathbf{x})^2$$

[3] Il s'agit en effet de séparer l'entrée du bruit.

Copyright © 2010 Eyrolles.

de même que la variance des paramètres :

$$\text{Var}[\mathbf{w}] = (\mathbf{X}\mathbf{X}^\top)^{-1}\sigma^2$$

La prédiction $p(\hat{y}|\mathbf{x})$ est donc gaussienne de moyenne : $\mathbf{w}\mathbf{x}$ et de variance : $\mathbf{w}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{w} + \sigma^2$. L'incertitude provient à la fois du bruit sur les entrées σ et de l'incertitude sur les paramètres estimés $\mathbf{w}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{w}$.

Il se peut que les colonnes de la matrice $\mathbf{X}$ ne soient pas linéairement indépendantes, et donc que $\mathbf{X}$ ne soit pas de rang maximal. Cela peut arriver, par exemple, si plusieurs entrées sont corrélées (e.g. $\mathbf{x}_{t_1} = \lambda\mathbf{x}_{t_2}$). Dans ce cas, $\mathbf{X}^\top\mathbf{X}$ est singulière, et l'estimation de $\hat{\mathbf{w}}$ n'est pas déterminée de manière unique. Cela signifie seulement qu'il y a plusieurs manières d'exprimer la solution. Il existe généralement des moyens naturels d'éliminer les colonnes redondantes de $\mathbf{X}$, et la plupart des packages de régression comportent des méthodes pour cela. Le problème du rang de la matrice peut également se produire lorsque la dimension des entrées d excède la taille m de l'échantillon d'apprentissage, comme par exemple en analyse du signal, en traitement d'images ou en analyse du transcriptome. Dans ce cas, on cherche généralement à réduire la dimension de l'espace d'entrée, par exemple par des techniques de réduction de dimension (voir chapitre 3). On peut aussi avoir recours à des techniques de régularisation afin de contrôler l'adaptation du modèle appris aux données (voir chapitre 3).

2.3 Quelques propriétés intéressantes

2.3.1 Interprétabilité et réduction de dimension

L'un des avantages des modèles linéaires est qu'il est possible d'en tirer aisément des informations sur l'importance des différents attributs x^i. Ainsi, le signe de w_i indique si l'attribut x^i a un effet positif ou non sur la sortie. De même, si les x^i sont normalisés, les valeurs absolues relatives des w_i indiquent l'importance relative des attributs et permettent de les trier par ordre d'importance, voire d'éliminer ceux qui sont associés à des paramètres dont la valeur est proche de zéro.

Plus précisément, si l'on suppose que l'erreur ε est gaussienne de moyenne nulle et de variance σ^2, c'est-à-dire $\varepsilon \sim \mathcal{N}(0,\sigma^2)$, alors :

$$\hat{\mathbf{w}} \sim \mathcal{N}(\mathbf{w}, (\mathbf{X}^\top\mathbf{X})^{-1}\sigma^2)$$

qui est une distribution normale multivariée. De plus :

$$(m - d - 1)\hat{\sigma}^2 \sim \sigma^2\chi^2_{m-p-1}$$

est une distribution de χ^2 à $m-d-1$ degrés de liberté. Les variables $\hat{\mathbf{w}}$ et $\hat{\sigma}^2$ sont statistiquement indépendantes. Ces propriétés permettent de construire des tests statistiques et de calculer des intervalles de confiance pour les paramètres w_i.

Pour tester si un coefficient $w_i = 0$, on utilise le score Z :

$$z_j = \frac{\hat{w}_j}{\hat{\sigma}\sqrt{v_j}} \tag{9.10}$$

où v_j est le j-ème élément diagonal de $(\mathbf{X}^\top\mathbf{X})^{-1}$.

Sous l'hypothèse nulle selon laquelle $w_j = 0$, z_j est distribué selon une t-distribution à $m-d-1$ degrés de liberté t_{m-d-1}. La constatation d'une valeur de z_j élevée conduit alors au rejet de cette hypothèse nulle (par exemple, un score Z de valeur absolue > 2 est significativement non nul au niveau $p = 0.05$). Nous renvoyons ici le lecteur à tout bon ouvrage sur les statistiques, par exemple [Dod03].

Copyright © 2010 Eyrolles.

2.3.2 Théorème de Gauss-Markov

Le théorème de Gauss-Markov démontre que le modèle linéaire résultant de l'estimation du vecteur de paramètres $\boldsymbol{w}$ par les moindres carrés a la variance la plus faible parmi toutes les estimations de modèles linéaires non biaisés.

Soit par exemple la prédiction $y_T = [1, \mathbf{x}_T]\,\mathbf{w}$. Son estimation par les moindres carrés est :

$$\hat{y}_T = [1, \mathbf{x}_T]\,\bar{\mathbf{w}} = [1, \mathbf{x}_T]\,(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{S}_y$$

En supposant qu'un modèle linéaire soit correct, l'estimation est non biaisée :

$$\begin{aligned} \mathtt{E}([1, \mathbf{x}_T]\,\bar{\mathbf{w}}) &= \mathtt{E}([1, \mathbf{x}_T]\,(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{S}_y) \\ &= [1, \mathbf{x}_T]\,(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{X}\mathbf{w} \\ &= [1, \mathbf{x}_T]\,\mathbf{w} \end{aligned}$$

D'après le théorème de Gauss-Markov, pour tout autre estimateur linéaire $\tilde{y} = \mathbf{M}\,\mathbf{S}_y$ non biaisé de $[1, \mathbf{x}_T]\,\mathbf{w}$ (c'est-à-dire tel que : $\mathtt{E}(\mathbf{M}\,\mathbf{S}_y) = [1, \mathbf{x}_T]\,\mathbf{w}$), on a :

$$\mathrm{Var}([1, \mathbf{x}_T]\,\mathbf{w}) \leq \mathrm{Var}(\mathbf{M}\,\mathbf{S}_y)$$

Si l'on considère la *moyenne des carrés des erreurs* (MCE) (*Mean Squared Error* (MSE) en anglais) d'un estimateur $\bar{\mathbf{w}}$ de $\boldsymbol{w}$:

$$\mathtt{MCE}(\bar{\mathbf{w}}) = \mathtt{E}(\bar{\mathbf{w}} - \mathbf{w})^2 = \mathrm{Var}(\bar{\mathbf{w}}) + \left[\mathtt{E}(\bar{\mathbf{w}}) - \mathbf{w}\right]^2 \tag{9.11}$$

Le premier terme est la variance tandis que le second terme est le biais au carré. Le théorème de Gauss-Markov implique que l'estimation des moindres carrés conduit à la plus petite moyenne des carrés des erreurs parmi tous les estimateurs linéaires non biaisés. Cependant, il peut exister un estimateur biaisé de moyenne des carrés des erreurs moindre. Il suffit qu'il soit possible de compenser une augmentation de la variance par une diminution plus importante du biais. C'est pourquoi les estimateurs biaisés sont souvent utilisés. Par exemple, toute méthode qui met à zéro certains des coefficients de $\boldsymbol{w}$ pour réduire la dimension de l'espace peut conduire à un estimateur biaisé, mais de meilleure performance. En général, le choix d'un modèle (d'un estimateur) revient à déterminer le bon compromis entre le biais et la variance, en accord avec l'un des principaux messages du chapitre 2.

Finalement, il est intéressant de noter que la moyenne des carrés des erreurs est liée à la précision de la prédiction. Ainsi, par exemple, soit la prédiction $\mathbf{y}_t = f(\mathbf{x}_t + \varepsilon)$. L'espérance de l'erreur de prédiction de l'estimation $\tilde{f}(\mathbf{x}_t) = \mathbf{x}_t^\top \bar{\mathbf{w}}$ est :

$$\begin{aligned} \mathtt{E}(\mathbf{y}_t - \tilde{f}(\mathbf{x}_t))^2 &= \sigma^2 + \mathtt{E}(\mathbf{x}_t^\top \bar{\mathbf{w}} - f(\mathbf{x}_t))^2 \\ &= \sigma^2 + \mathtt{MCE}(\tilde{f}(\mathbf{x}_t)) \end{aligned} \tag{9.12}$$

L'espérance d'erreur de prédiction ne diffère de la moyenne des carrés des erreurs que par la constante σ^2, qui représente la variance de la nouvelle observation $\mathbf{y}_t$.

2.3.3 Complexité calculatoire

L'estimation par les moindres carrés est généralement réalisée soit par la décomposition de Cholesky de la matrice $\mathbf{X}^\top \mathbf{X}$, soit par décomposition QR[4] de $\mathbf{X}$. Avec m exemples de dimension d, la décomposition de Cholesky demande $d^3 + m\,d^2/2$ opérations, tandis que la décomposition QR demande $m\,d^2$ opérations. Selon les valeurs relatives de m et d, l'une ou l'autre de ces méthodes est la plus efficace.

[4] La décomposition QR est issue de la procédure d'orthogonalisation de Gram-Schmidt et cherche une bonne base orthogonale pour les colonnes de $\mathbf{X}$ (voir [HTF01]).

Copyright © 2010 Eyrolles.

2.3.4 Et avec une autre fonction d'erreur ?

Supposons que nous cherchions un modèle constant $y = a$ rendant au mieux compte de l'échantillon de données $\mathcal{S} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{1 \leq t \leq m}$. Que devrions-nous choisir pour la valeur de a ?

Cela dépend de la fonction d'erreur considérée. Pour la *fonction d'erreur quadratique*, la moyenne est la meilleur estimation. En effet :

$$\begin{aligned} \mathtt{E} &= \sum_{t=1}^{m} (y_t - a)^2 \\ \frac{\mathrm{dE}}{\mathrm{d}a} &= 2 \sum_{t=1}^{m} (y_t - a) \\ \hat{a} &= \frac{1}{m} \sum_{t=1}^{m} y_t \end{aligned}$$

Pour la fonction d'erreur *valeur absolue de la différence*, la médiane est la meilleure estimation :

$$\begin{aligned} \mathtt{E} &= \sum_{t=1}^{m} |y_t - a)| \\ \frac{\mathrm{dE}}{\mathrm{d}a} &= 2 \sum_{t=1}^{m} \mathrm{signe}(a - y_t) \\ &= (\#y_t \texttt{ plus petit que } a) - (\#y_t \texttt{ plus grand que } a) \\ \hat{a} &= \mathrm{mediane}[y_1, \ldots, y_m] \end{aligned}$$

Qu'en est-il alors lorsque l'on cherche non plus un modèle constant mais un modèle linéaire ? Il faut résoudre une problème de programmation linéaire :

$$\begin{cases} & \min \sum_{t=1}^{m} u_t \\ \texttt{soumis à} & -u_t \leq u_t - \mathbf{w}^\top \mathbf{x}_t \leq u_t \end{cases}$$

Chaque fonction d'erreur demande par conséquent une analyse propre afin de déterminer la solution optimale au problème d'apprentissage associé.

3. Séparatrices linéaires

3.1 Généralités

Nous nous plaçons désormais dans le cadre de l'apprentissage supervisé d'une règle de classification, c'est-à-dire que la variable dépendante ou à prédire prend ses valeurs dans un ensemble fini, que l'on associe généralement à un ensemble de classes. C'est le cas, par exemple, de la reconnaissance de lettres ou de classes de documents.

Lorsque l'on se place dans un espace de représentation euclidien, on peut librement faire des hypothèses sur la géométrie des classes ou sur celles de leurs surfaces séparatrices. La plus simple d'entre elles est de supposer que deux classes peuvent être séparées par une certaine surface, définie par une équation ; les paramètres qui régissent cette équation sont alors les variables à apprendre. Cette hypothèse n'est pas au fond tellement plus forte que de supposer que les classes

Copyright © 2010 Eyrolles.

sont de nature gaussienne, comme on le fera par exemple au chapitre 15 ; elle est souvent bien sûr fausse, mais peut mener à une probabilité d'erreur raisonnable dans un bon nombre de problèmes. Cette probabilité d'erreur doit évidemment être évaluée objectivement, sur un ensemble de test ou par les méthodes présentées au chapitre 3.

Le nombre de paramètres à calculer est minimal si l'on suppose cette surface linéaire ; aussi est-ce l'hypothèse qui prévaut souvent, en particulier lorsque l'échantillon de données est de taille réduite par rapport à la dimension de l'espace d'entrée, d'autant qu'elle permet de mener des calculs faciles et de visualiser précisément le résultat obtenu.

3.1.1 Hyperplans séparateurs et discriminants dans un problème à deux classes

Dans $\mathbb{R}^d$, une surface linéaire est un hyperplan A, défini par l'équation :

$$a_0 + \mathbf{a}^T\mathbf{x} = 0 \tag{9.13}$$

avec a vecteur de dimension d et a_0 scalaire. Si deux classes ω_1 et ω_2 sont *séparables* par A, tous les points de la première classe sont par exemple tels que :

$$\mathbf{x} \in \omega_1 \quad \Rightarrow \quad a_0 + \mathbf{a}^T\mathbf{x} \geq 0 \tag{9.14}$$

et ceux de la seconde vérifient alors :

$$\mathbf{x} \in \omega_2 \quad \Rightarrow \quad a_0 + \mathbf{a}^T\mathbf{x} \leq 0 \tag{9.15}$$

Dans un espace de dimension $d = 1$, une séparation linéaire se réduit à la comparaison à un seuil. Prenons ce cas particulier pour donner deux exemples où un problème de discrimination à deux classes ne peut pas en pratique être complètement résolu par une séparatrice linéaire.

—— Exemple ——

Cherchons à classer les hommes et les femmes sur la seule mesure de leur taille. Bien que les hommes soient en général plus grands que les femmes, aucun échantillon d'apprentissage représentatif ne permettra de fixer une valeur pertinente pour le seuil, étant donné qu'il existera évidemment toujours certains hommes plus petits que certaines femmes. Dans ce cas, c'est l'espace de représentation qui est trop pauvre : d'autres critères doivent être mis en œuvre pour décider sans erreur du sexe d'un individu.

Supposons pour un autre exemple qu'un avimateur cherche à discriminer les eiders (*Somatiera mollissima*) mâles des femelles par le niveau de gris moyen de leur plumage. Là encore, un seuil sur cette mesure donnera une classification confuse ; mais, cette fois, l'espace de mesure est suffisant ; c'est en revanche la définition des classes qui est trop pauvre. En effet, les femelles sont toutes marron clair ; mais les mâles sont soit blanc brillant quand on les observe de face, soit noirs quand ils sont vus de dos. Il faudrait donc subdiviser cette dernière espèce, ce qui rendrait possible une bonne séparation linéaire en trois classes (eider mâle vu de face, eider mâle vu de dos, eider femelle) par deux seuils sur les couleurs. Il suffirait ensuite de regrouper les deux classes extrêmes pour avoir résolu le problème dans l'espace de représentation initial. Cet exemple sera détaillé au début du chapitre suivant, pour présenter les réseaux connexionnistes.

Définition 9.1 (Hyperplan séparateur, séparatrice linéaire, hyperplan discriminant.)

On appelle hyperplan séparateur *ou* séparatrice linéaire *un hyperplan qui sépare parfaitement deux classes, c'est-à-dire qui vérifie les équations 9.14 et 9.15 ; en particulier, il sépare parfaitement leurs points d'apprentissage. Un* hyperplan discriminant *est un classificateur linéaire pour deux classes qui ne sont pas linéairement séparables.*

Copyright © 2010 Eyrolles.

Il n'est en général pas possible de trouver un hyperplan séparateur. On se contentera donc de chercher un hyperplan discriminant par approximation, au sens d'un certain critère.

Le problème de l'apprentissage de classificateurs linéaires se ramène donc à la recherche des paramètres $\mathbf{a}$ et a_0 d'un hyperplan discriminant, séparant le mieux possible les points d'apprentissage de ω_1 et de ω_2 dans l'espace $\mathbb{R}^d$ et conduisant à la meilleure faculté de généralisation possible.

3.1.2 Un peu de géométrie dans $\mathbb{R}^d$

Un hyperplan dans $\mathbb{R}^d$, où les coordonnées sont notées $x_1, \ldots, x_d$, est défini par $d+1$ paramètres $a_0, a_1, \ldots, a_d$. Il répond à l'équation :

$$a_0 + a_1 x_1 + \cdots + a_d x_d = 0 \tag{9.16}$$

En notant vectoriellement $\mathbf{a}^\top = (a_1, \ldots, a_d)$ et $\mathbf{x}^\top = (x_1, \ldots, x_d)$, l'équation 9.16 s'écrit de manière équivalente : $a_0 + \mathbf{a}^\top \mathbf{x} = 0$.

Un hyperplan d'équation $a_0 + \mathbf{a}^\top \mathbf{x} = 0$ a pour *vecteur normal* $\mathbf{a}$. Sa *distance algébrique* à l'origine vaut $a_0 / \| \mathbf{a} \|$ avec $\| \mathbf{a} \|^2 = \sum_{i=1}^{d} a_i^2$.

La figure 9.2 montre un hyperplan à deux dimensions qui est la droite d'équation $g(\mathbf{x}) = a_0 + a_1 x_1 + a_2 x_2 = 0$, ainsi que son vecteur normal $\mathbf{a}$.

Cette figure donne aussi la distance de deux points particuliers à cet hyperplan : l'origine, à la distance $a_0 / \| \mathbf{a} \|$, et un point $\mathbf{y}$ de l'autre côté de la droite, à la distance $g(\mathbf{y}) / \| \mathbf{a} \|$. Les signes de ces deux distances algébriques sont opposés, puisque les deux points ne sont pas du même côté de l'hyperplan. Le signe de la distance de l'origine à l'hyperplan est le même que celui de a_0.

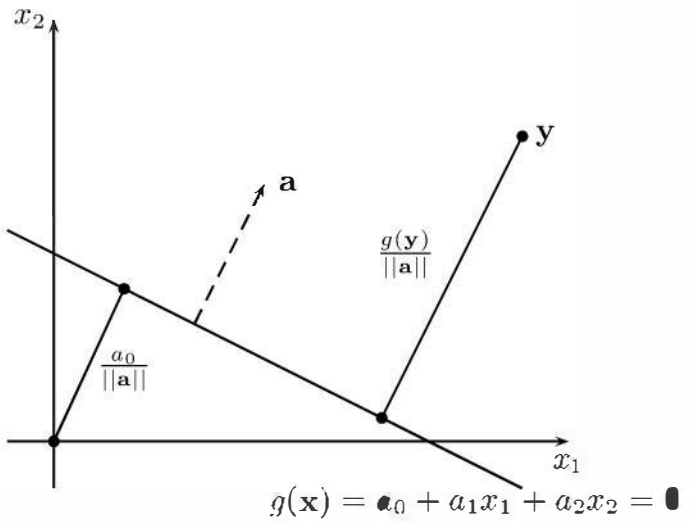


FIG. 9.2: *La géométrie d'un hyperplan.*

Pour caractériser l'hyperplan d'équation $a_0 + \mathbf{a}^\top \mathbf{x} = 0$, on peut rassembler tous ses paramètres dans un vecteur $\mathbf{A}$ de dimension $d+1$ construit comme :

$$\mathbf{A}^\top = (a_0, \mathbf{a}^\top) = (a_0, a_1, \ldots, a_d)$$

Copyright © 2010 Eyrolles.

Par extension, on note $\mathbf{A}$ l'hyperplan lui-même. Notons qu'en réalité il suffit de d paramètres pour définir un hyperplan : on peut sans perdre de généralité imposer par exemple : $a_0 = 1$.

Posons $g(\mathbf{x}) = a_0 + \mathbf{a}^\top \mathbf{x}$ et notons $\Delta(\mathbf{y}, \mathbf{A})$ la distance entre l'hyperplan $\mathbf{A}$ d'équation $g(\mathbf{x}) = 0$ et un point $\mathbf{y}$ de $\mathbb{R}^d$. Elle est définie comme la distance entre $\mathbf{y}$ et le point de $\mathbf{A}$ le plus plus proche de y. Ce point n'est autre que la projection orthogonale de $\mathbf{y}$ sur $\mathbf{A}$. La valeur $\Delta(y, \mathbf{A})$ est, comme on l'a vu, une distance algébrique donnée par la formule :

$$\Delta(y, \mathbf{A}) = \frac{g(y)}{|| \mathbf{a} ||}$$

3.2 L'apprentissage d'un hyperplan pour discriminer deux classes

3.2.1 Une solution globale

3.2.1.1 Le cas séparable Supposons pour le moment que les deux classes ω_1 et ω_2 soient séparables par un hyperplan. Il existe donc un vecteur $\mathbf{a}$ caractérisant cet hyperplan séparateur tel que :

- $\mathbf{a}^\top \mathbf{x}$ positif pour $\mathbf{x} \in \omega_1$;
- $\mathbf{a}^\top \mathbf{x}$ négatif pour $\mathbf{x} \in \omega_2$.

Notons qu'il est facile de se ramener à ce cas quand c'est le contraire qui se produit, en inversant simplement le signe de toutes les coordonnées de $\mathbf{a}$. La formalisation de la séparation linéaire pour un problème à deux classes se fait alors en plongeant le problème dans l'espace de dimension $d + 1$. Le vecteur $\mathbf{a}$ et le scalaire a_0 sont transformés en un vecteur $\mathbf{A}$ en ajoutant a_0 à $\mathbf{a}$ comme une coordonnée de rang 0, ce qui se note : $\mathbf{A}^\top = (a_0, \mathbf{a}^\top)$. On impose une valeur non nulle à a_0, par exemple $a_0 = 1$.

Soit $\mathcal{S}$ l'ensemble des données d'apprentissage : il est composé de m exemples $(\mathbf{x}, \omega)$ où $\mathbf{x}$ est un vecteur de $\mathbb{R}^d$ et $\omega \in \{\omega_1, \omega_2\}$. On transforme ces exemples en vecteurs $\mathbf{X}$ de $\mathbb{R}^{d+1}$ par :

$$\begin{cases} \mathbf{x} \in \omega_1 & \Rightarrow \quad \mathbf{X}^\top = (1, \mathbf{x}^\top) \\ \mathbf{x} \in \omega_2 & \Rightarrow \quad \mathbf{X}^\top = (-1, -\mathbf{x}^\top) \end{cases} \tag{9.17}$$

Par conséquent, quelle que soit la classe de $\mathbf{x}$, on a désormais : $\mathbf{A}^\top \mathbf{X} \geq 0$ (rappelons que l'hyperplan caractérisé par le vecteur $\mathbf{A}$ est pour le moment supposé séparateur : tous les points sont bien classés).

Si on range les m vecteurs $\mathbf{X}$ comme les colonnes d'une matrice M, il est alors facile de voir le problème de séparation linéaire comme la recherche d'un vecteur $\mathbf{A}$ dans $\mathbb{R}^{d+1}$ tel que :

$$\mathbf{A}^\top M = \mathbf{B}^\top \tag{9.18}$$

où $\mathbf{B}$ est un *vecteur positif* de $\mathbb{R}^m$ (dont toutes les coordonnées sont positives).

Ceci revient à dire que l'on recherche un hyperplan dans l'espace de représentation augmenté d'une dimension, avec les contraintes qu'il passe par l'origine et que $\mathbf{B}$ reste positif. Cette manipulation mathématique n'a d'autre but que de rendre plus aisée la résolution du problème.

Remarquons que cette équation ne peut pas se résoudre dans le cas général, puisque M n'est pas inversible (c'est une matrice à $d + 1$ lignes et m colonnes) et que $\mathbf{B}$ est inconnu.

3.2.1.2 Le cas non séparable Traitons maintenant le cas général où les classes et les données d'apprentissage ne sont pas séparables. On se ramène donc à la recherche d'une séparatrice linéaire en cherchant un hyperplan $\mathbf{A}$ le meilleur possible pour distinguer les deux classes. Le principe *ERM* (voir les chapitres 2 et 21) nous indique de choisir celui qui minimise le nombre

Copyright © 2010 Eyrolles.

d'exemples mal classés. Mais c'est une valeur discontinue en fonction des paramètres d'un hyperplan. Pour des raisons de facilité de calcul, on va s'intéresser à des critères presque identiques, mais dérivables. Par exemple, on peut minimiser le critère :

$$J(\mathbf{A},\mathbf{B}) = \frac{1}{2} \parallel \mathbf{A}^T M - \mathbf{B}^\top \parallel^2 \tag{9.19}$$

Pourquoi ce critère ? Si les deux classes sont séparables, il existera un couple $(\mathbf{A},\mathbf{B})$ avec $\mathbf{B}$ positif pour lequel ce critère sera exactement nul. En imposant dans le cas général la contrainte $\mathbf{B}$ positif ou nul, on peut donc comprendre intuitivement qu'une bonne solution approchée pour $\mathbf{A}$ rendra $J(\mathbf{A},\mathbf{B})$ assez petit.

— EXEMPLE —

Il est possible de donner une justification plus profonde à cet argument. En effet, $J(\mathbf{A},\mathbf{B})$ peut s'écrire :

$$J(\mathbf{A},\mathbf{B}) = \frac{1}{2}\sum_{j=1}^{m}[\mathbf{A}^T\mathbf{X}_j - b_j]^2$$

$\mathbf{X}_j$ étant la projection dans l'espace de dimension $d+1$ du $j^{ème}$ point d'apprentissage $\mathbf{x}_j$. On sait (voir le chapitre 3, paragraphe 3.1.2) que la distance $\Delta(\mathbf{x}_j,\mathbf{A})$ du point $\mathbf{x}_j$ à l'hyperplan caractérisé par le vecteur $\mathbf{A}$ a pour valeur :

$$\Delta(\mathbf{x}_j,\mathbf{A}) = \frac{\mathbf{a}^\top \mathbf{x}_j + a_0}{\parallel \mathbf{a} \parallel} = \frac{\mathbf{A}^T\mathbf{X}_j}{\parallel \mathbf{a} \parallel}$$

Son carré peut donc s'écrire :

$$[\Delta(\mathbf{x}_j,\mathbf{A})]^2 = \frac{(\mathbf{A}^T\mathbf{X}_j)^2}{\parallel \mathbf{a} \parallel^2}$$

On en déduit :

$$J(\mathbf{A},\mathbf{B}) = \frac{1}{2 \parallel \mathbf{a} \parallel^2}\sum_{j=1}^{m}[\Delta(\mathbf{x}_j,\mathbf{A}) - b_j \parallel \mathbf{a} \parallel]^2$$

Pour les points de A bien classés par l'hyperplan $\mathbf{A}$, le terme $[\Delta(\mathbf{x}_j,\mathbf{A}) - b_j \parallel \mathbf{a} \parallel]^2$ peut être annulé en choisissant $b_j = \frac{\Delta(\mathbf{x}_j,\mathbf{A})}{\|\mathbf{a}\|}$, qui est positif. Pour les points mal classés, à défaut de pouvoir prendre b_j négatif, le mieux que l'on puisse faire est d'imposer b_j nul.

Minimiser $J(\mathbf{A},\mathbf{B})$ sous la contrainte $\mathbf{B}$ positif ou nul revient donc d'une certaine manière à se placer dans le cadre du critère des moindres carrés : on cherche l'hyperplan $\mathbf{A}$ qui minimise la somme des distances entre $\mathbf{A}$ et les points de l'ensemble d'apprentissage que $\mathbf{A}$ classe mal. Si $\mathbf{A}$ et $\mathbf{B}$ minimisent effectivement $J(\mathbf{A},\mathbf{B})$, on a :

$$[J(\mathbf{A},\mathbf{B})] = \sum_{\mathbf{x}\in S \text{ mal classé par } \mathbf{A}} [\Delta(\mathbf{x},\mathbf{A})]^2$$

Par conséquent, sous l'hypothèse que les distances entre $\mathbf{A}$ et les points mal classés par $\mathbf{A}$ sont réparties selon une distribution gaussienne, et que ceux-ci sont les seuls à compter dans le positionnement de $\mathbf{A}$, la minimisation de $J(\mathbf{A},\mathbf{B})$ est la recherche du meilleur hyperplan *a posteriori* (au sens bayésien), c'est-à-dire le plus probable connaissant les données (voir le chapitre 15, paragraphe 1.6).

Copyright © 2010 Eyrolles.

Résolution. Les équations classiques de l'algèbre linéaire fournissent la valeur du gradient de $J(\mathbf{A}, \mathbf{B})$ par rapport à $\mathbf{A}$:

$$\nabla_{\mathbf{A}} J(\mathbf{A}, \mathbf{B}) = (\mathbf{A}^\top M - \mathbf{B}^\top) M^\top \tag{9.20}$$

Comme $J(\mathbf{A}, \mathbf{B})$ est toujours positif ou nul, son minimum est atteint pour :

$$\nabla_{\mathbf{A}} J(\mathbf{A}, \mathbf{B}) = 0 \tag{9.21}$$

ce qui fournit quand la matrice $M^\top M$ est inversible (ce qui est vrai en général) :

$$\mathbf{A}^\top = \mathbf{B}^\top M^+ \tag{9.22}$$

avec $M^+ = M^\top (M M^\top)^{-1}$ *pseudo-inverse* de M.

Par conséquent, si le vecteur $\mathbf{B}$ était connu, la solution à notre problème consisterait simplement à calculer $\mathbf{A}$ par la formule :

$$\mathbf{A}^\top = \mathbf{B}^\top M^+ \tag{9.23}$$

À défaut de connaître exactement $\mathbf{B}$, on sait qu'il est positif quand les données sont séparables. On peut, conformément à ce qui a été dit plus haut, chercher une solution qui approche ce cas idéal en fournissant des valeurs strictement positives pour les coordonnées de $\mathbf{B}$ correspondant à des points bien classés, et des valeurs nulles pour les points mal classés. L'algorithme cherché doit donc réaliser une minimisation de $J(\mathbf{A}, \mathbf{B})$ sous la contrainte $\mathbf{B}$ positif ou nul.

On dispose alors de deux types de méthodes pour finir de résoudre le problème : celles reposant sur un calcul numérique global direct et celles qui procèdent de manière itérative sur les données d'apprentissage (on en verra une au paragraphe suivant). La méthode globale la plus simple est la *programmation linéaire* : si l'on ne retient de $\mathbf{B}$ que sa positivité, l'équation ci-dessus se ramène à m inéquations linéaires dans l'espace de dimension $d+1$. Le problème est donc de grande taille si le nombre d'exemples est important. Les méthodes globales sont de ce fait peu employées ici.

3.2.2 Une méthode itérative : l'algorithme de Ho et Kashyap

Cette méthode consiste, à partir d'un vecteur $\mathbf{B}_0$ arbitraire, à en déduire une suite convergente de valeurs $\mathbf{A}_t$ et $\mathbf{B}_t$ de vecteurs paramètres du problème. Son principe est le suivant : si l'on suppose connaître un certain $\mathbf{B}_t$, on sait que l'on peut calculer $\mathbf{A}_t$ par l'équation 9.22, soit ici :

$$\mathbf{A}_t^\top = \mathbf{B}_t^\top M^+ \tag{9.24}$$

On cherche à minimiser le critère $J(\mathbf{A}_t, \mathbf{B}_t)$. Comment modifier $\mathbf{B}_t$ pour diminuer cette valeur ? Simplement en calculant le gradient de $J(\mathbf{A}_t, \mathbf{B}_t)$ par rapport à $\mathbf{B}_t$ et en en déduisant une valeur $\mathbf{B}_{t+1}$ telle que :

$$J(\mathbf{A}_t, \mathbf{B}_{t+1}) \leq J(\mathbf{A}_t, \mathbf{B}_t) \tag{9.25}$$

Ce gradient vaut :

$$\nabla_{\mathbf{B}_t} J(\mathbf{A}_t, \mathbf{B}_t) = \nabla_{\mathbf{B}_t} \mid\mid \mathbf{A}_t^\top M - \mathbf{B}_t^\top \mid\mid = -2(\mathbf{A}_t^\top M - \mathbf{B}_t^\top) \tag{9.26}$$

Une façon de réduire $J(\mathbf{A}_t, \mathbf{B}_t)$ est d'utiliser la méthode de la *descente de gradient* (voir l'annexe 8), qui consiste ici à retrancher à $\mathbf{B}_t$ un vecteur colinéaire à $\nabla_{\mathbf{B}_t} J(\mathbf{A}_t, \mathbf{B}_t)$, donc à faire :

$$\mathbf{B}_{t+1}^\top = \mathbf{B}_t^\top + \alpha(\mathbf{A}_t^\top M - \mathbf{B}_t^\top) \tag{9.27}$$

où α est un coefficient positif qui règle la vitesse de convergence de l'algorithme.

Copyright © 2010 Eyrolles.

Il y a cependant une précaution à prendre : celle d'éviter de rendre négatifs des termes de $\mathbf{B}_{t+1}$; pour cela, on remplace simplement dans $(\mathbf{A}_t^\top M - \mathbf{B}_t^\top)$ tous les termes négatifs par 0, ce qui donne un vecteur noté $\lfloor \mathbf{A}_t^\top M - \mathbf{B}_t^\top \rfloor$.

On démontre que cette procédure converge vers une valeur nulle de J quand les deux classes sont séparables, vers une valeur positive sinon ; cette dernière valeur peut cependant ne pas être le minimum global de $J(\mathbf{A}, \mathbf{B})$ sur toutes les valeurs possibles de $\mathbf{A}$ et $\mathbf{B}$. L'algorithme 9 décrit cette procédure.

Algorithme 9 : Algorithme de Ho et Kashyap

début
  Prendre $\mathbf{B}_0$ et α positif quelconques
  $t \leftarrow 0$
  tant que *critère d'arrêt non satisfait* **faire**
    $\mathbf{A}_t^\top \leftarrow \mathbf{B}_t^\top M^+$
    $\mathbf{B}_{t+1}^\top \leftarrow \mathbf{B}_t^\top + \alpha \lfloor \mathbf{A}_t^\top M - \mathbf{B}_t \rfloor^\top$
    $t \leftarrow t + 1$
  fin tant que
fin

Le critère d'arrêt peut être, par exemple : $J(\mathbf{A}_t, \mathbf{B}_t) \simeq J(\mathbf{A}_{t+1}, \mathbf{B}_{t+1})$, ou bien : $t \geq t_{max}$.

La valeur α peut être considérée, du point de vue de l'apprentissage, comme l'importance de la punition ou de la récompense que l'on applique aux paramètres pour les modifier ; c'est d'ailleurs une remarque générale pour toutes les méthodes de gradient. Grande, elle permet une bonne exploration de l'espace décrit par J, mais peut faire varier erratiquement les vecteurs $\mathbf{A}_t$ et $\mathbf{B}_t$ et interdire en pratique leur convergence ; petite, elle donne une vitesse de convergence lente et peut mener trop facilement à un optimum local.

Il existe diverses possibilités pour régler automatiquement α qui ont été largement étudiées dans le cadre des méthodes générales d'optimisation par les techniques de descente de gradient. L'une d'entre elles a par exemple pour principe de réduire cette valeur au fur et à mesure de la croissance de t : on peut la voir comme un survol exploratoire à grande vitesse du « paysage » J, progressivement remplacé par une descente de plus en plus lente vers le minimum entrevu comme le plus prometteur[5].

3.3 Un autre calcul : l'algorithme du perceptron

L'algorithme du perceptron travaille directement sur le vecteur $\mathbf{a}$ qui caractérise la surface discriminante cherchée. On n'a donc plus besoin ici de se placer dans l'espace de représentation de dimension $d + 1$ ni d'utiliser le vecteur $\mathbf{A}$. Cet algorithme utilise un *protocole d'apprentissage itératif* : il prend les données d'apprentissage les unes après les autres, chacune étant choisie soit par un passage systématique dans l'ensemble d'apprentissage (version « non stochastique »), soit par un tirage au hasard dans celui-ci (version « stochastique »). Son nombre d'étapes effectives peut être important : un seul (exactement ou en moyenne) passage des données n'est en effet pas suffisant pour le faire converger.

[5] Ou comme la tactique normale d'un joueur de golf, qui joue des clubs de moins en moins « longs » pour être plus précis au fur et à mesure qu'il se rapproche du trou.

Copyright © 2010 Eyrolles.

3.3.1 L'algorithme stochastique

Dans sa version stochastique, l'algorithme du perceptron se décrit par l'algorithme 10 : à l'étape t de son calcul, le vecteur $\mathbf{a}$ est noté $\mathbf{a}_t$; le nombre maximal d'étapes est fixé à t_{max}.

Algorithme 10 : Le perceptron, version stochastique

```
début
    Prendre a_(0) et α positif quelconques
    t ← 0
    tant que t ≤ t_max faire
        tirer au hasard une donnée d'apprentissage x
        si x est bien classé alors
            a_{t+1} ← a_t
        sinon
            si x ∈ ω1 alors
                a_{t+1} ← a_t + αx
            sinon
                a_{t+1} ← a_t − αx
            fin si
        fin si
        t ← t + 1
    fin tant que
fin
```

On constate qu'il n'y a pas de modification si la donnée lue est classée correctement. En revanche, en cas de mauvais classement, la valeur courante $\mathbf{a}_t$ de $\mathbf{a}$ est corrigée.

3.3.2 Convergence

Le calcul suivant montre empiriquement en quoi cette tactique est fondée : supposons que la donnée d'apprentissage $\mathbf{x}$ appartienne à la classe ω_1 ; on sait que dans ce cas, puisqu'il est mal classé, $\mathbf{A}_t^\top \mathbf{x}$ est négatif au lieu d'être positif.

$$\mathbf{a}_{t+1}^\top \mathbf{x} = (\mathbf{a}_t + \alpha \mathbf{x})^\top \mathbf{x} = \mathbf{a}_t^\top \mathbf{x} + \alpha \mathbf{x}^\top \mathbf{x} = \mathbf{a}_t^\top \mathbf{x} + \alpha \parallel \mathbf{x} \parallel \geq \mathbf{a}_t^\top \mathbf{x} \tag{9.28}$$

Par conséquent, puisque la valeur $\mathbf{a}^\top \mathbf{x}$ a augmenté, la donnée $\mathbf{x}$ devrait avoir à son prochain passage plus de chances de vérifier l'inégalité $\mathbf{a}^\top \mathbf{x} \geq 0$ et d'être donc bien classée dans la classe ω_1. Le calcul est analogue dans le cas où une donnée devant appartenir à ω_2 est mal classée : la valeur $\mathbf{a}_{t+1}^\top \mathbf{x}$ devient inférieure à $\mathbf{a}_t^\top \mathbf{x}$, ce qui la rapproche de la valeur négative qu'elle aurait dû prendre.

Cette justification est en fait rigoureusement démontrée par un théorème de convergence qui, comme dans le cas précédent, assure que dans le cas de classes séparables, un hyperplan convenable est trouvé en un nombre fini d'étapes[6].

3.3.3 La version non stochastique et son interprétation

Reprenons cet algorithme dans sa version non stochastique, dans laquelle les données d'apprentissage sont proposées dans l'ordre. L'itération centrale de l'algorithme (appelée une *époque*

[6] Le théorème de Novikoff prouve que l'algorithme du perceptron converge en un nombre fini T d'étapes : $T \leq 2R^2/M^2$ où $M = \min_{1 \leq i \leq m} \{u, d(x_i, h)\}$ pour un séparateur h et d la distance utilisée.

Copyright © 2010 Eyrolles.

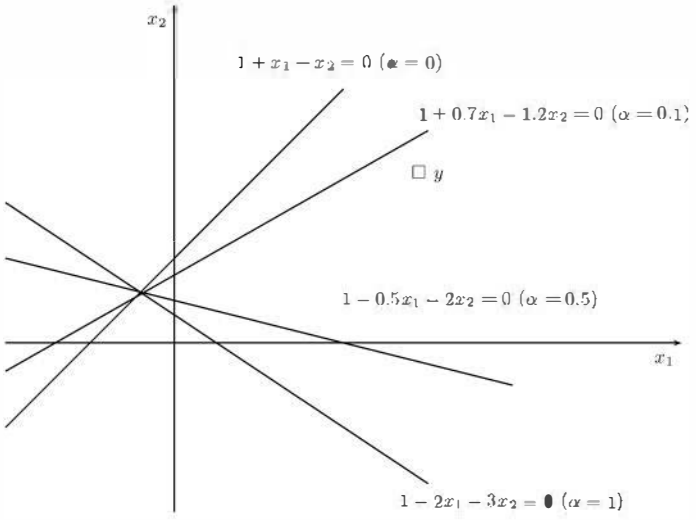


FIG. 9.3: *Fonctionnement de l'algorithme du perceptron.*

d'apprentissage) porte sur le calcul de la modification (notée ci-dessous *modif*) que subit $\mathbf{a}$ par le cumul des contributions de tous les exemples. La base des exemples intervient plusieurs fois. L'algorithme 11 décrit le processus.

On peut voir cette version de l'algorithme comme la minimisation du critère

$$J(\mathbf{a}) = - \sum_{\mathbf{x} \in \mathcal{S} \text{ mal classé par } \mathbf{a}} \mathbf{a}^\top \mathbf{x}$$

En effet, en supposant a_0 fixé à la valeur 1, on a :

$$\nabla_{\mathbf{a}} J(\mathbf{a}) = - \sum_{\mathbf{x} \in \mathcal{S} \text{ mal classé par } \mathbf{a}} \mathbf{x} = \frac{modif}{\alpha}$$

Par conséquent, la modification apportée au vecteur $\mathbf{A}$ par une époque d'apprentissage est proportionnelle au gradient du critère $J(a)$. Cet algorithme minimise donc $J(\mathbf{a})$, qui est un critère un peu différent de celui de l'algorithme de Ho et Kashyap. L'expérience prouve en général que ce dernier est plus efficace du point de vue de la généralisation.

EXEMPLE **Fonctionnement**

Prenons l'espace de représentation égal à $\mathbb{R}^2$ (figure 9.3) et supposons qu'à l'itération t, l'équation de la surface séparatrice courante soit : $x_1 - x_2 + 1 = 0$.

Autrement dit, on a fixé $a_0 = 1$ et le vecteur courant $\mathbf{a}_t$ vaut : $\mathbf{a}_t = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

Maintenant, supposons que l'exemple suivant soit : $(\mathbf{x}, c) = \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \omega_2$

La valeur de l'équation de la séparatrice $x_1 - x_2 + 1 = 0$ sur les coordonnées de l'exemple est donc : $3 - 2 + 1 = 2$

Or, puisque l'exemple $(\mathbf{x}, c)$ appartient à la classe ω_2, cette valeur devrait être négative. Il faut donc modifier $\mathbf{a}_t$.

Le calcul à faire, puisque $(\mathbf{x}, c) \in \omega_2$, est le suivant : $\mathbf{a}_{t+1} = \mathbf{a}_t - \alpha.\mathbf{x}$

Copyright © 2010 Eyrolles.

Algorithme 11 : Le perceptron, version non stochastique

```
début
    Prendre A_(0) quelconque et α positif quelconque
    t ← 0
    tant que t ≤ t_max faire
        modif ← 0
        pour chaque donnée d'apprentissage x faire
            si x est mal classé alors
                si x ∈ ω1 alors
                    modif ← modif + αx
                sinon
                    modif ← modif − αx
                fin si
            fin si
        fin pour
        A_{t+1} ← A_t + modif
        t ← t + 1
    fin tant que
fin
```

Pour $\alpha = 0.1$, on obtient :

$$\mathbf{a}_{t+1} = \mathbf{a}_t - 0.1 \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} - \begin{pmatrix} 0.3 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 0.7 \\ -1.2 \end{pmatrix}$$

La surface séparatrice courante a donc pour nouvelle équation :

$$0.7x_1 - 1.2x_2 + 1 = 0$$

On constate graphiquement qu'elle s'est rapprochée du nouvel exemple, sans toutefois bien le classer. Pour un pas plus grand, par exemple : $\alpha = 1$, on obtiendrait la séparatrice :

$$-2x_1 - 3x_2 + 1 = 0$$

qui classe correctement le nouveau point.

On verra au chapitre 10 une extension du perceptron au cas de plus de deux classes ; on verra aussi comment on peut le généraliser en réseau connexionniste.

3.4 L'hyperplan discriminant de Fisher

La méthode de Fisher diffère des précédentes en ce qu'elle ramène d'abord le problème à une dimension avant de construire un hyperplan. Pour cela, elle va chercher la droite F de vecteur directeur f passant par l'origine telle que la projection des points d'apprentissage sur F sépare au mieux les deux classes ω_1 et ω_2. Ensuite, elle cherche sur F le point par lequel passe l'hyperplan discriminant, orthogonal à F.

Notons μ_1 et μ_2 la moyenne des points d'apprentissage de ω_1 de ω_2, m_1 et m_2 le nombre de points d'apprentissage dans ω_1 et ω_2 et définissons :

$$\overline{\mu_1} = \frac{1}{m_1} \sum_{\mathbf{x} \in \omega_1} \mathbf{f}^\top \mathbf{x} \tag{9.29}$$

Copyright © 2010 Eyrolles.

comme le scalaire donnant la moyenne des valeurs des projections des points de ω_1 sur F, et notons de même $\overline{\mu_2}$ pour la seconde classe. Définissons aussi :

$$\overline{s_1} = \sum_{\mathbf{x} \in \omega_1} (\mathbf{f}^\top \mathbf{x} - \overline{\mu_1})^2 \tag{9.30}$$

et $\overline{s_2}$ la *dispersion* de ces projections. Le critère de Fisher consiste à chercher la droite F telle que la valeur

$$J(F) = \frac{(\overline{\mu_1} - \overline{\mu_2})^2}{\overline{s_1}^2 + \overline{s_2}^2} \tag{9.31}$$

soit maximale. Cette valeur est en effet une mesure de la séparation des projections des deux classes sur la droite F. On peut d'ailleurs l'exprimer autrement.

Définissons les valeurs :

$$S_1 = \sum_{\mathbf{x} \in \omega_1} (\mathbf{x} - \mu_1)(\mathbf{x} - \mu_1)^\top \tag{9.32}$$

et de même S_2 ; la valeur

$$S_I = S_1 + S_2 \tag{9.33}$$

est appelée la *variance intraclasse totale* des données d'apprentissage. Définissons aussi la *variance interclasse* par :

$$S_J = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^\top \tag{9.34}$$

Il est alors possible de prouver que :

$$J(F) = \frac{\mathbf{f}^\top S_J f}{f^\top S_I \mathbf{f}} \tag{9.35}$$

Finalement, on démontre que le critère $J(F)$ est maximal pour :

$$\widehat{\mathbf{f}} = S_I^{-1}(\mu_1 - \mu_2) \tag{9.36}$$

L'hyperplan séparateur de Fisher étant orthogonal à la droite F, il a donc pour équation :

$$\widehat{\mathbf{f}}\mathbf{x} - f_0 = 0$$

La dernière inconnue est la valeur scalaire f_0. Puisque le problème a été ramené à une dimension, il suffit pour la déterminer de supposer par exemple que les projections de chaque classe sur F sont gaussiennes et de procéder comme au chapitre 15.

3.5 Et pour plus de deux classes ?

Dans le cas où l'on a un ensemble d'apprentissage représentatif de plus de deux classes, il existe plusieurs façons de généraliser la discrimination linéaire. La première est d'apprendre pour chaque classe un hyperplan qui la discrimine de toutes les autres, ce qui revient à considérer la réunion de leurs exemples comme négatifs et ceux de la classe en apprentissage comme positifs. Mais cette façon de procéder conduit, comme le montre la figure 9.4, à de larges ambiguïtés de classification.

Une autre solution est de chercher une surface discriminante entre chaque couple de classes (figure 9.4, à droite) ; elle présente l'avantage de dessiner moins de zones ambiguës, mais elle a l'inconvénient de demander le calcul de $C(C-1)/2$ jeux de paramètres au lieu de C.

Copyright © 2010 Eyrolles.

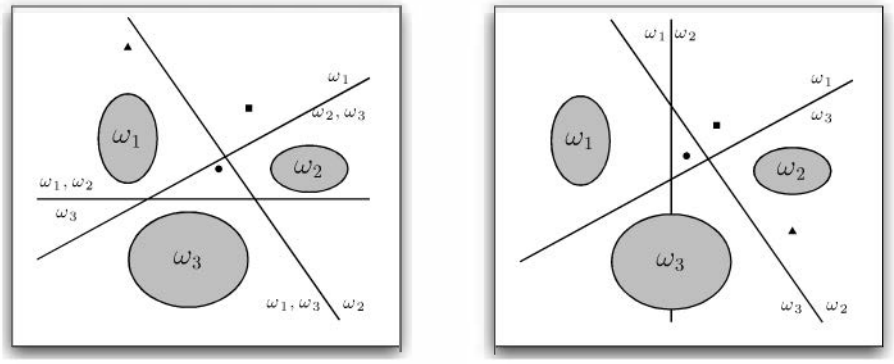


FIG. 9.4: Séparation linéaire à plus de deux classes. *À gauche, on sépare chaque classe de toutes les autres : il y a C hyperplans. Le point en triangle est attribué à la classe ω_1, le point en carré est ambigu entre ω_1 et ω_2, le point central est ambigu entre les trois classes. Sur les sept zones, quatre sont ambiguës. À droite, on sépare chaque classe de chaque autre : il y a $\frac{C(C-1)}{2}$ hyperplans. Le point en triangle et le point en carré sont attribués à la classe ω_2, le point central est ambigu entre les trois classes. Sur les sept zones, une seule est ambiguë.*

Dans le cas de plus de trois classes, la géométrie de la séparation par hyperplans devient nettement plus complexe : l'espace $\mathbb{R}^d$ est en effet partagé dans le cas général en $\frac{n^2+n+2}{2}$ zones convexes par n hyperplans.

Les problèmes d'ambiguïté peuvent être résolus par un système ultérieur de décision sur l'ensemble des classes, prenant en compte des paramètres supplémentaires, par exemple la distance aux hyperplans appris (qui se déduit directement des formules présentées ci-dessus) et/ou les probabilités *a priori* des classes. On peut aussi regrouper provisoirement les classes par des méthodes de classification hiérarchique, et réaliser un processus de décision arborescent sur cette base. Ceci n'est pas sans rapport avec les méthodes des *arbres de décision* présentées au chapitre 13. Notons aussi que l'algorithme du perceptron, donné ici dans le cas de deux classes, possède une généralisation naturelle à plus de deux classes qui sera présentée au chapitre 10.

Ce problème de l'adaptation de classificateurs construits pour deux classes à un nombre quelconque de classes sera traité d'une autre manière et plus en détail au chapitre 17, section 3.2.

4. Modèles linéaires par morceaux et combinaisons de modèles locaux

Une généralisation des modèles linéaires consiste à supposer que la variable dépendante (de sortie) y est linéaire *localement* dans l'espace d'entrée $\mathcal{X}$, avec une dépendance linéaire variable suivant la région de $\mathcal{X}$. On parle alors de *modèle linéaire par morceaux* (*Piecewise Linear Model*).

Géométriquement, le modèle consiste en un ensemble d'hyperplans de dimension d, chacun couvrant une région de l'espace d'entrée $\mathcal{X}$ disjointe des autres. Les paramètres de ce type de modèles sont les paramètres décrivant chaque hyperplans et les paramètres spécifiant les régions (ou frontières) de ces hyperplans.

Copyright © 2010 Eyrolles.

4.1 La régression linéaire par morceaux

Pour des raisons de simplicité de présentation, nous supposons désormais que la sortie $y \in \mathbb{R}$, de même que l'entrée est supposée uni-dimensionnelle $x \in \mathbb{R}$. Une fonction linéaire par morceaux $f(x)$ est obtenue en divisant le domaine $\mathcal{X}$ en intervalles contiguës et en définissant une fonction linéaire dans chacun de ces intervalles.

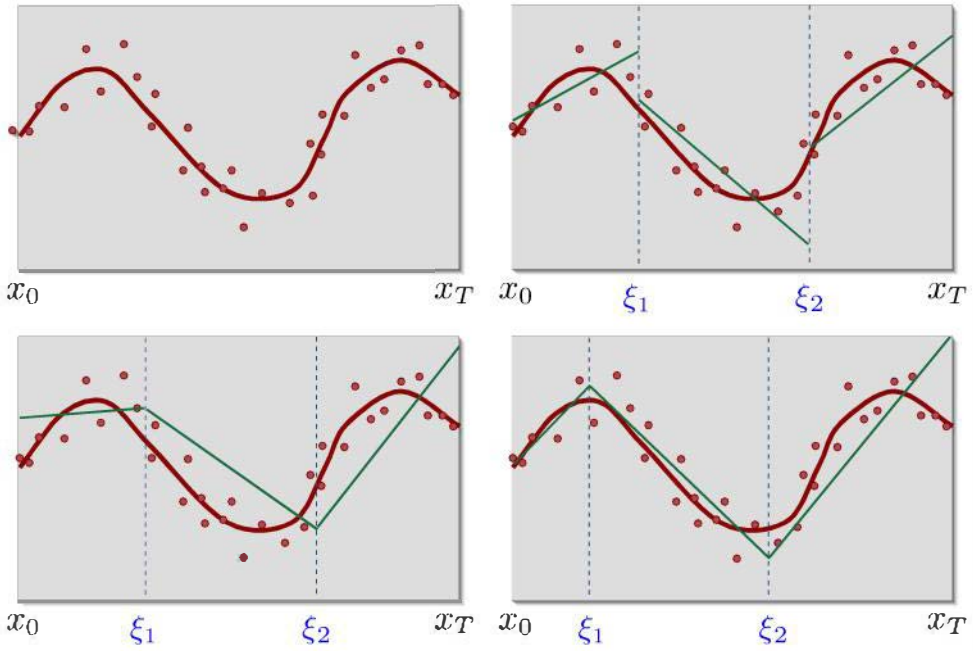


FIG. 9.5: Régression linéaire par morceaux. *La courbe représente la fonction cible et les points, des données engendrés à partir de la courbe avec un bruit gaussien. En haut à droite, les nœuds ξ_1 et ξ_2 sont fixés a priori, et les droites de régression sont non restreintes. En bas à gauche, les nœuds ξ_1 et ξ_2 sont fixés a priori, et les droites de régression sont restreintes à être continues en chaque nœud. En bas à droite, les nœuds ξ_1 et ξ_2 sont calculés, et les droites de régression sont contraintes à être continues en chaque nœud.*

Les paramètres à déterminer deviennent plus nombreux que dans la régression linéaire simple. Il faut maintenant déterminer à la fois les paramètres de chaque droite de régression dans chacun des intervalles, mais aussi spécifier les frontières entre intervalles, souvent appélés *nœuds*.

Pour de nombreuses applications, il peut être gênant d'avoir des discontinuités entre les modèles locaux (par exemple, on ne souhaite pas que l'impôt saute brutalement d'une tranche de revenus à la suivante). On impose alors des conditions de continuité entre modèles à chaque nœud (voir figure 9.5).

4.2 La régression spline

Une généralisation supplémentaire du modèle précédent consiste à remplacer les modèles locaux linéaires par des modèles plus riches, comme des polynômes de degré fixés. Il faut alors déterminer le nombre d'intervalles, les paramètres de chacun de ces modèles dans chaque intervalle, ainsi que les nœuds spécifiant les intervalles. Dans le cas où des conditions de continuité sont imposées entre les modèles locaux, on parle de *spline* pour décrire la fonction de régression obtenue. On

Copyright © 2010 Eyrolles.

peut imposer également des conditions de continuité sur les dérivées en chaque nœud. Une *spline d'ordre* M à K nœuds est une régression par K morceaux utilisant des polynômes de degré $\leq M$ dans chaque intervalle.

Les fonctions de base utilisées sont dans ce cas :

$$\begin{cases} h_j(x) & = x^{j-1} & , \ j = 1, \ldots, M \\ h_{M+l}(x) & = (x - \xi_l)_+^{M-1} & , \ l = 1, \ldots, K \end{cases}$$

où $(x - \xi_l)_+$ désigne la fonction prise « à droite » de ξ_l. D'autres bases, plus pratiques pour les calculs, incluent les $B - splines$[7].

4.3 La combinaison linéaire de fonctions de base

Au lieu de se restreindre à des fonctions polynômes, il est possible d'utiliser n'importe quelle base ou *dictionnaire* de *fonctions de base* $\Phi = \{\phi_i(\mathrm{x})\}_{i=1,N}$ définies de $\mathcal{X}$ dans $\mathcal{Y}$[8]. On peut alors considérer des fonctions linéaires définies à l'aide de ce dictionnaire :

$$f(\mathrm{x}) = w_0 + \sum_{j=1}^{N} w_j \, \phi_j(\mathrm{x}) \tag{9.37}$$

Par exemple, on peut imaginer de projeter le vecteur d'entrée x défini dans l'espace vectoriel $\mathcal{X}$ décrit par les vecteurs de base $\{e_i\}_{1 \leq i \leq d}$, où d est la dimension de l'espace d'entrée, dans un nouvel espace où les vecteurs de base sont des combinaisons de type $e_i e_j \ldots$ On traite alors de régularités polynomiales.

L'intérêt de ces modèles est que, bien qu'ils permettent d'exprimer des relations non linéaires entre variables d'entrée et variables de sortie, ils sont cependant linéaires en leurs paramètres, qui sont par ailleurs en nombre limité ; de ce fait, les techniques d'estimation de ces paramètres (les w_i) restent faciles et du ressort des techniques exposées pour les modèles linéaires simples. Ces modèles sont très généraux.

1. Lorsque l'on cherche à estimer la fonction $f(\mathrm{x})$ à partir d'un échantillon de données $\mathcal{S}_m = \langle (\mathrm{x}_1, f(\mathrm{x}_1) + \varepsilon), (\mathrm{x}_2, f(\mathrm{x}_2) + \varepsilon), \ldots, (\mathrm{x}_m, f(\mathrm{x}_m) + \varepsilon) \rangle$, il s'agit d'un **problème de régression**.
2. Si l'on cherche à estimer une fonction de décision de la forme 9.37, il s'agit d'un **problème de classification** linéaire dans l'espace défini par les fonctions de base.
3. Finalement, on peut aussi chercher à **estimer une densité de probabilité** en utilisant un modèle de mélange (*mixture model*, en anglais) de la forme :

$$\mathsf{p}(\mathrm{x}) = \sum_{i=1}^{N} \mathsf{p}_i(\mathrm{x}|\theta_i) \, \pi_i \tag{9.38}$$

Dans ce cas, la densité de probabilité générale sur $\mathcal{X}$ est décomposée en une somme pondérée de fonctions de densité. Chacune de ces fonctions de base $\mathsf{p}_i(\mathrm{x}|\theta_i)$ consiste typiquement en une fonction paramétrique simple (de paramètre θ_i), comme une densité normale. π_i représente la probabilité qu'un point tiré au hasard ait été engendré à partir de la densité i, que l'on peut interpréter comme une classe (on a donc $\sum_i \pi_i = 1$).

[7] Voir par exemple [HTF01].

[8] En général, les fonctions ϕ_i sont choisies continues sur $\mathcal{X}$.

Copyright © 2010 Eyrolles.

Il est évident que le nombre N de fonctions de base ou de composants est lié à la complexité des modèles considérés, et doit donc être contrôlé avec soin.

Notons que les méthodes à base de dictionnaires de fonctions de base sont apparentées mais différentes des *méthodes à base de fonctions noyaux* (voir chapitre 14) dans lesquelles les fonctions de bases elles-mêmes sont déterminées en cours d'apprentissage, et non fixées *a priori*. Cependant, nous verrons que les méthodes à base de fonctions noyau utilisent des fonctions de base d'un type prédéterminé. Par contraste, les réseaux connexionnistes (voir chapitre 10) peuvent être considérés comme des combinaisons de fonctions de base adaptatives, ce qui rend leur contrôle compliqué. Dans les deux cas, une question fondamentale concerne le choix des fonctions de base.

5. La recherche des facteurs pertinents

Ce chapitre traite des régularités ayant la forme de combinaisons linéaires de facteurs, ceux-ci pouvant être des variables ou des modèles locaux. Nous avons fait allusion, dans la section 2.3.1, au fait que ces méthodes permettent d'associer automatiquement un poids aux facteurs et donc d'identifier les facteurs importants. Cette observation était cependant quelque peu hâtive. Il peut arriver en effet que certains facteurs soient corrélés, ce qui entraîne que leur coefficient ne soit pas représentatif de leur importance intrinsèque (e.g. un facteur peut prendre un très fort coefficient car il est contrebalancé par un facteur corrélé ayant un fort coefficient opposé). Par ailleurs, dans certaines applications, par exemple en bioinformatique, le nombre de facteurs (e.g. des gènes) peut largement dépasser le nombre d'observations (e.g. des biopuces). Dans ce cas, le problème est mal conditionné et peut donc conduire à des résultats très instables. Finalement, et le plus important, les méthodes traditionnelles fondées sur l'estimation de la corrélation intrinsèque de chaque facteur avec la sortie sont biaisées parce que l'on choisit ces facteurs avec les données. On a le même problème que pour la recherche d'une meilleure hypothèse à partir d'un échantillon de données. C'est pourquoi, au milieu des années quatre-vingt-dix, le grand statisticien Brad Efron (1938 -) déclarait que le plus grand problème à résoudre était celui de l'identification des facteurs pertinents (témoignage cité dans l'article de synthèse [HCMF08]).

5.1 Méthodes de détermination ou de pondération des facteurs pertinents

Les **méthodes classiques** sont essentiellement la *sélection pas à pas* (« *stepwise selection* »), l'exploration de tous les sous-ensembles de facteurs (« *all-subsets* »), et la « *ridge regression* ». Elles sont toutes sujettes à instabilité face à des petites variations de l'échantillon d'apprentissage, elles sont biaisées et sont parfois coûteuses à mettre en œuvre.

- *All-subsets selection.* Supposons que le nombre total de facteurs soit p. La méthode d'exploration de tous les sous-ensembles cherche pour chaque $k \in \{1, 2, \ldots, p\}$ le sous-ensemble de k facteurs donnant la performance maximale, par exemple la plus petite somme des erreurs au carré résiduelle dans le cas de la régression. Il est évident que cette erreur résiduelle décroît avec k croissant. Il faut donc trouver un critère pour choisir k. Celui-ci implique en général un compromis entre biais et variance, favorisant par exemple les modèles les plus parcimonieux. Un problème critique de cette approche est son coût calculatoire qui la rende impraticable au-delà de 30 à 40 facteurs. Il faut donc avoir recours à des méthodes itératives.
- *Stepwise selection.* Les méthodes pas à pas sont des méthodes gloutonnes et myopes recherchant à chaque pas le facteur restant dont l'ajout (*forward-stepwise selection*) ou le retrait (*backward-stepwise selection*) améliore le plus l'adéquation aux données. Ici encore se pose

Copyright © 2010 Eyrolles.

le problème de la sélection du nombre de facteurs k.

- *Ridge regression.* Plutôt que d'organiser une recherche dans le graphe des facteurs pertinents (et de leur poids associé), une autre approche consiste à pénaliser l'usage de facteurs dans l'hypothèse calculée. Parce que ces méthodes cherchent à réduire l'importance des facteurs dans l'hypothèse, elles sont appelées *shrinkage methods* en anglais. Un des avantages de cette approche est d'être moins sujette à une variance élevée car les facteurs ne sont plus ajoutés ou retirés de manière brutale, mais leurs poids sont ajustés. Dans le cas de la *ridge regression*, la pénalité est de type norme L_2, c'est-à-dire une somme des carrés des coefficients w_j :

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\text{Argmin}}\left\{\sum_{i=1}^{m}\left(u_i - w_0 - \sum_{j=1}^{p} w_j x_i^j\right)^2 + \lambda \sum_{j=1}^{p} w_j^2\right\} \tag{9.39}$$

  $\lambda \geq 0$ étant un paramètre de contrôle de la pénalité. Il faut noter que w_0 n'est pas pris en compte dans la pénalisation, pour ne pas faire dépendre le résultat de la position des points par rapport à l'origine. De même, il faut normaliser les entrées pour ne pas faire dépendre le résultat de l'échelle choisie.

- *La méthode Lasso* est également une méthode employant une pénalisation, mais celle-ci est de type norme L_1 (somme des valeurs absolues des coefficients) :

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\text{Argmin}}\left\{\sum_{i=1}^{m}\left(u_i - w_0 - \sum_{j=1}^{p} w_j x_i^j\right)^2 + \lambda \sum_{j=1}^{p} |w_j|\right\} \tag{9.40}$$

La méthode *Lasso* est donc très proche de la *ridge regression*. Pourquoi la considérer et en faire même un des sujets de recherche très actifs actuellement? Une raison majeure est que la **pénalisation en norme L_1 conduit naturellement à des modèles parcimonieux**, c'est-à-dire comportant peu de facteurs. Une manière de comprendre pourquoi consiste à ré-exprimer les problèmes 9.39 et 9.40, ce qui donne pour le *ridge regression* :

$$\begin{cases} \mathbf{w}^\star = \text{Argmin}_{\mathbf{w}}\left\{\sum_{i=1}^{m}\left(u_i - w_0 - \sum_{j=1}^{p} w_j x_i^j\right)^2\right\} \\ \text{soumis à} \quad \sum_{j=1}^{p} w_j^2 \leq t \end{cases}$$

et pour le *Lasso* :

$$\begin{cases} \mathbf{w}^\star = \text{Argmin}_{\mathbf{w}}\left\{\sum_{i=1}^{m}\left(u_i - w_0 - \sum_{j=1}^{p} w_j x_i^j\right)^2\right\} \\ \text{soumis à} \quad \sum_{j=1}^{p} |w_j| \leq t \end{cases}$$

Le paramètre t est directement lié au paramètre λ.

Dans le cas de données décrites à l'aide de deux facteurs x_1 et x_2, la figure 9.6 permet de comparer les solutions produites. Les ellipsoïdes décrivent les courbes d'équi-erreurs, tandis que les zones grisées correspondent aux contraintes sur les coefficients notés ici β_1 et β_2. La solution correspond à la première intersection trouvée entre la courbe d'équi-erreur et la zone grisée traduisant les contraintes. Dans le cas d'une norme L_1, le volume grisé présente des angles (et encore plus en grande dimension) qui favorisent des solutions pour lesquelles seuls quelques coefficients w_j (ou β_j) sont non nuls.

La régularisation utilisant la norme L_1 a pris son essor ces dernières années, grâce en particulier aux propriétés de parcimonie qu'elle entraîne sur les modèles appris. Cela a conduit au développement du domaine appelé *compressed sensing* en traitement du signal [Don06, CW08].

Copyright © 2010 Eyrolles.

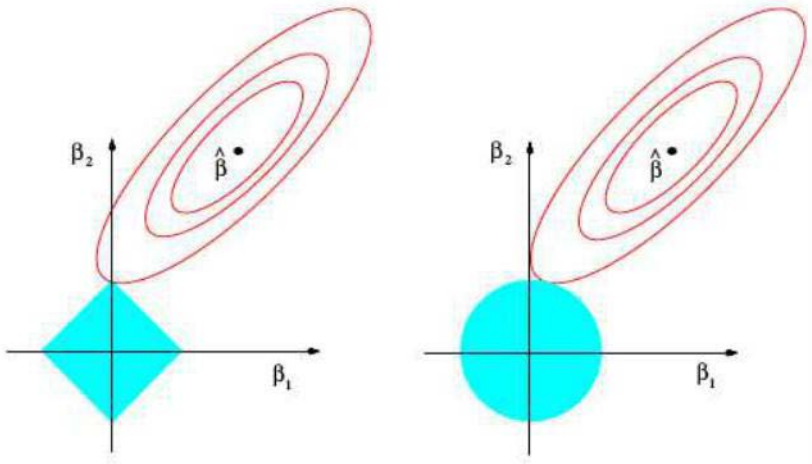


FIG. 9.6: *L'intersection de la courbe d'equi-erreur avec la zone de contraintes sur les coefficients* β_j *(*$|\beta_1| + |\beta_2| \leq t$ *pour* Lasso *et* $\beta_1^2 + \beta_2^2 \leq t$ *pour la* ridge regression*) conduit naturellement à favoriser des coefficients nuls pour le* Lasso *à gauche, par rapport à la* ridge regression*, à droite.*

Une **nouvelle méthode**, appelée LAR pour *Least Angle Regression* a été introduite récemment par Brad Efron [EHJT04]. Il s'agit d'une méthode itérative, comme la *forward-stepwise selection*, mais au lieu d'ajouter à chaque pas un facteur, elle accroît son poids d'autant qu'il le mérite. Dans un premier temps, l'algorithme identifie le facteur le plus corrélé avec le résidu (la différence entre la prédiction du modèle actuel et les vraies valeurs sur l'échantillon d'apprentissage). Il augmente alors le poids de ce facteur jusqu'à ce qu'un autre facteur devienne davantage corrélé avec le résidu, etc. jusqu'à ce que tous les facteurs aient été introduits (voir l'algorithme 12). Par construction, les coefficients changent linéairement par morceaux, ce qui signifie que le nombre de pas est fini. En fait, l'algorithme LAR est très efficace, requérant à peu près le même nombre de calculs que la régression linéaire par moindres carrés utilisant les p facteurs.

Algorithme 12 : Algorithme LAR (*Least Angle Regression*)

début

  répéter

    1. Normaliser les facteurs pour qu'ils aient une moyenne nulle et une norme unité. Commencer avec le résidu $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $w_1, w_2, \ldots, w_p = 0$.
    2. Identifier le facteur $\mathbf{x}_j$ le plus corrélé avec $\mathbf{r}$.
    3. Changer le poids w_j de 0 à sa valeur de corrélation de moindre carré $\langle \mathbf{x}_j, \mathbf{r} \rangle$, jusqu'à ce qu'un autre facteur $\mathbf{x}_k \neq \mathbf{x}_j$ ait une corrélation égale avec le résidu courant.
    4. Changer les coefficients w_j et w_k dans la direction définie par leur corrélation de moindre carré avec le résidu courant, jusqu'à ce qu'un autre facteur $\mathbf{x}_l$ ait une corrélation égale avec le résidu courant.

  jusqu'à *ce que les p facteurs aient été pris en compte.*

fin

Copyright © 2010 Eyrolles.

La description donnée ici, brève par nécessité, ne fait pas totalement justice de ces méthodes. Nous renvoyons le lecteur intéressé à [HTF09], chapitre 3 par exemple.

Beaucoup plus généralement, on voit que le choix de l'ensemble de facteurs pertinents, ou celui du réglage de paramètres d'apprentissage peut ne dépendre que d'un paramètre unique. On peut alors envisager d'explorer toutes les solutions associées aux variations de ce paramètre. C'est ce que l'on appelle des « méthodes de chemin » (*path algorithms*). La figure 9.7 montre l'ensemble des poids des facteurs obtenus pour les valeurs possibles d'un paramètre de contrôle.

Depuis la publication de l'algorithme LAR [EHJT04], de nombreux travaux ont été consacrés au développement d'algorithmes pour calculer le chemin de toute une variété de problèmes.

Lorsque le paramètre en jeu contrôle le compromis biais-variance, ou encore la régularisation du critère inductif, on parle de *chemin de régularisation.*

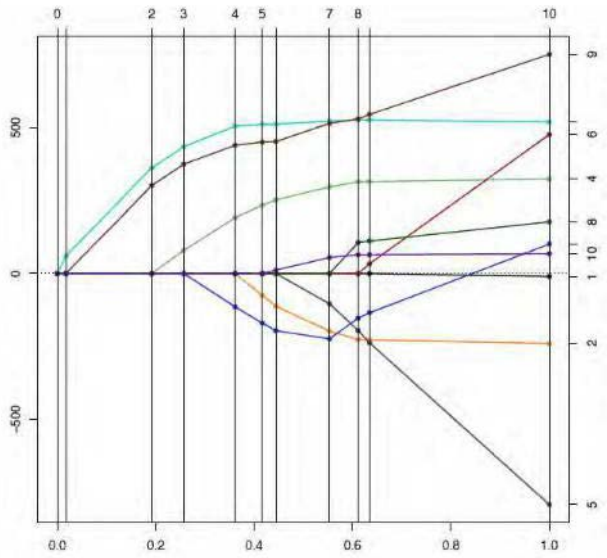


FIG. 9.7: *Ensemble des poids associés aux différents facteurs pour un jeu de données artificiel lorsque l'on fait varier la somme des valeurs absolues des poids des facteurs de 0 à la somme maximale possible dans une procédure LAR. En haut figure le nombre de facteurs pris en compte (de 0 à 10). À gauche, un coefficient normalisé pour les facteurs. À droite, les numéros des facteurs. On voit que au fur et à mesure que l'on augmente le paramètre de contrôle, de plus en plus de facteurs prennent un poids non nul (tiré d'un tutoriel de Trevor Hastie en sept. 2006)*

5.2 Optimisation du critère inductif par chemin de régularisation

Comme nous l'avons vu au chapitre 2, le problème d'induction supervisée sous l'hypothèse de données i.i.d. conduit à chercher une hypothèse optimisant un risque empirique régularisé :

$$h^*_\lambda = \underset{h \in \mathcal{H}}{\text{Argmin}} \left[R_{\text{Emp}}(h) + \lambda \, \text{Reg}(h) \right] \tag{9.41}$$

où $\lambda \in \mathbb{R}^+$ est un paramètre de contrôle arbitrant entre la fidélité aux données d'apprentissage et un biais de régularité sur les hypothèses.

Un problème essentiel est de régler λ de manière appropriée, et si possible d'en trouver la valeur optimale λ^* permettant d'identifier l'hypothèse de meilleur risque réel.

Copyright © 2010 Eyrolles.

Une première approche consiste à explorer méthodiquement l'espace des valeurs possibles de λ, par exemple en faisant varier sa valeur de manière discrète sur un intervalle donné. Une autre approche, proposée récemment, est de calculer le « chemin de régularisation », c'est-à-dire de calculer toutes les solutions h^*_λ pour $\lambda \in [0, \infty[$ ou pour un intervalle jugé intéressant. *A priori*, il apparaît déraisonnable d'explorer ainsi toutes les solutions de ce qui apparaît comme une infinité de problèmes. Mais si les poids des facteurs en jeu évoluent linéairement par morceaux en fonction du paramètre de contrôle, alors, le nombre de points à calculer devient fini, et il devient possible d'explorer exhaustivement l'ensemble des solutions selon toutes les valeurs possibles du paramètre de contrôle. De là l'engouement récent pour ce type de démarche (voir ainsi la section 6.3 du chapitre 14 sur les méthodes à noyaux).

Notes historiques et approfondissements

La théorie de la discrimination linéaire remonte aux années 1930, lorsque Fisher proposa une procédure de classification. Dans le domaine de l'intelligence artificielle, l'attention fut attirée vers cette question par les travaux de Rosenblatt qui commença à étudier la règle d'apprentissage du perceptron à partir de 1956. Minsky et Papert, dans leur célèbre livre [MP69], analysèrent les limitations des machines linéaires dans le cadre de la classification. Le livre de Duda, Hart et Stork [DHS01] offre un panorama très complet des recherches sur ces machines.

Résumé

- La régression linéaire a pour objet de trouver l'hyperplan le plus adéquat pour représenter un ensemble de points.
- La discrimination linéaire cherche le meilleur hyperplan pour séparer deux classes dans un ensemble d'apprentissage. Les critères utilisés cherchent à approcher le principe ERM et les méthodes utilisent souvent une optimisation par descente de gradient.
- L'extension naturelle est la recherche d'hyperplans linéaires par morceaux.
- La recherche des facteurs pertinents revient à trouver des régularités sous la forme de combinaisons linéaires d'attributs ou de modèles locaux.

Copyright © 2010 Eyrolles.

Chapitre 10

L'apprentissage de réseaux connexionnistes

Dans ce chapitre, nous présentons une technique d'apprentissage fondée au départ sur une analogie avec la physiologie de la transmission de l'information et de l'apprentissage dans les systèmes cérébraux : les modèles connexionnistes (on dit aussi les réseaux de neurones artificiels). Le but du développement de cette approche était à l'origine de modéliser le fonctionnement du cerveau; cela reste un des axes de recherche du domaine, mais ici nous traiterons seulement de l'application de certains modèles informatiques élémentaires à l'apprentissage automatique de règles de classification.

Le cerveau est un organe caractérisé par l'interconnexion d'un nombre élevé d'unités de traitement simples, les cellules nerveuses ou neurones. Le comportement de ce réseau naturel de neurones est déterminé par son architecture, c'est-à-dire le nombre des cellules et la manière dont elles sont connectées, ainsi que par les poids affectés à chacune des connexions. Chaque connexion entre deux neurones est caractérisée par son poids qui mesure le degré d'influence du premier neurone vers le second. La capacité d'apprentissage de ces systèmes est reliée à la mise en mémoire de nouvelles connaissances par la modification des poids des connexions à partir d'exemples. Pour donner un ordre de grandeur de sa complexité, le cerveau humain comporte environ cent milliards de neurones, chacun relié en moyenne à dix mille autres.

Nous abandonnons maintenant toute forme de référence biologique pour nous intéresser aux réseaux de neurones artificiels, ou réseaux connexionnistes, et en particulier à leur application à l'apprentissage automatique.

Le calcul par réseaux connexionnistes est fondé sur la propagation d'informations entre des unités élémentaires de calcul. Les possibilités de chacune sont faibles, mais leur interconnexion permet d'effectuer un calcul global complexe. Du point de vue de l'apprentissage, les poids des connexions entre ces unités peuvent être réglés sur des ensembles d'exemples : le réseau ainsi entraîné pourra réaliser des tâches de classification ou de régression.

Copyright © 2010 Eyrolles.

Sommaire

1 Les différents éléments d'un réseau connexionniste 328
2 L'architecture multicouche 330
2.1 La transmission de l'information dans un réseau multicouche 330
2.2 Le protocole d'apprentissage 334
2.3 Le codage des exemples d'apprentissage 335
3 L'algorithme d'apprentissage 335
3.1 Retour sur le perceptron 335
3.1.1 Le perceptron pour deux classes 335
3.1.2 Le perceptron pour plus de deux classes 337
3.1.3 Plus de deux classes et une sigmoïde en sortie 339
3.2 L'apprentissage par rétropropagation du gradient de l'erreur 340
3.3 L'organisation des calculs 341
3.4 Une variante 343
3.5 Quand arrêter l'apprentissage ? 344
3.6 Le problème des minima locaux 344
4 Quelques résultats théoriques sur les réseaux connexionnistes . . . 344
4.1 Pouvoir d'expression 344
4.2 Complexité 345
5 Comment choisir l'architecture d'un réseau ? 345
6 Les réseaux à architecture profonde 346
7 Réseaux et régime dynamique : le *Reservoir Computing* 348
7.1 Les réseaux de neurones récurrents 348
7.2 Le traitement de données temporelles 349
7.3 Le *Reservoir Computing* 350
7.4 Bilan très provisoire et perspectives 350

Copyright © 2010 Eyrolles.

LES EIDERS (*Somatiera mollissima*) sont des canards marins pour lesquels les guides ornithologiques[1] sont pour une fois assez péremptoires : « Le mâle est le seul canard qui paraisse blanc quand on le voit de face et noir quand il est vu de dos... Le plumage de la femelle est brun. » Supposons qu'une bande d'eiders vogue à quelque distance des côtes. Comment un avimateur peut-il distinguer un eider mâle d'un eider femelle uniquement sur la couleur[2] ?

Le problème est en apparence assez simple. Il faut définir une échelle qui va du blanc au noir et y définir trois zones. On aura une décision du type de celle de la figure 10.1.

Notons x le niveau de gris d'un eider dont on cherche à déterminer le sexe. Les connaissances sont les suivantes :

- *Si* $x \leq 1$ *ou* $x \geq 2$ *alors* mâle.
- *Si* $1 \leq x \leq 2$ *alors* femelle.

[1] H. Heintzel, R. Fitter and J. Parslow *Oiseaux d'Europe*. Delachaux et Niestlé, 1972.

[2] On suppose bien sûr qu'il n'y a que des eiders dans la zone observée et qu'aucun mâle ne se présente de profil.

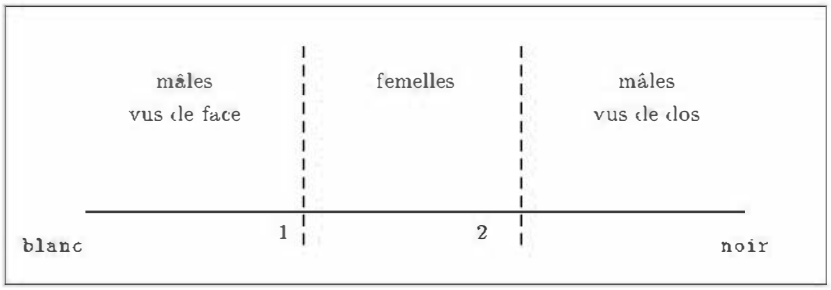


FIG. 10.1: *Répartition des eiders sur l'axe x :* `blanc/noir`.

Le concept « sexe de l'animal observé » ne peut donc pas se traduire directement par une seule comparaison.

Les techniques des surfaces séparatrices linéaires que nous avons vues au chapitre précédent sont-elles valables ici ? Non, car ce concept est impossible à décrire de cette façon. En effet, une décision linéaire dans un espace à une seule dimension (ici, la couleur x est le seul attribut du problème) revient à comparer l'attribut à un seuil. Mais ici, deux seuils interviennent et non pas un seul. Il va falloir trouver une technique de décision plus élaborée.

Nous allons décomposer le problème en deux étages de décision linéaire. Le premier produira deux valeurs binaires notées y_1 et y_2. La première indique si oui ou non x est inférieur au seuil de valeur 2, la seconde si x est supérieur au seuil de valeur 1. Le deuxième étage combinera ces deux valeurs binaires pour décider si le concept est satisfait ou non. C'est ce que montrent le tableau ci-dessous et la figure 10.2.

	$x \leq 1$	$1 \leq x \leq 2$	$x \geq 2$
y_1	$VRAI$	$VRAI$	$FAUX$
y_2	$FAUX$	$VRAI$	$VRAI$

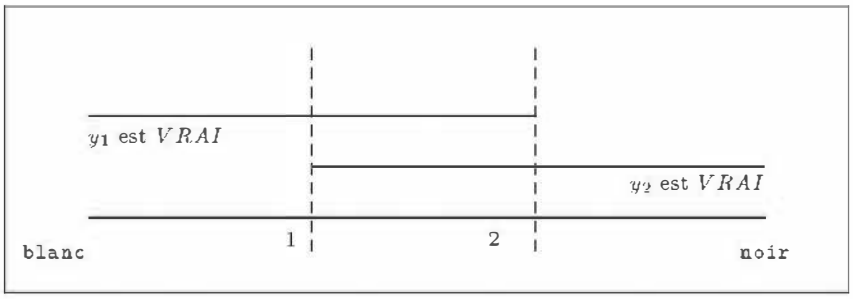


FIG. 10.2: *Deux décisions élémentaires à combiner.*

À partir de y_1 et y_2, la décision est donc la suivante :

y_1	y_2	Concept
$VRAI$	$VRAI$	Femelle
$VRAI$	$FAUX$	Mâle
$FAUX$	$VRAI$	Mâle
$FAUX$	$FAUX$	(impossible)

Copyright © 2010 Eyrolles.

Ce problème a été présenté jusqu'ici dans le cadre de la logique, mais il est facile de le transformer en une décision numérique. Il suffit pour cela d'attribuer les valeurs réelles 0 et 1 à y_1 et y_2 selon qu'ils sont $VRAI$ ou $FAUX$. Il est en effet peu intuitif, mais immédiat de vérifier que la valeur numérique z

$$z = y_1 + y_2 - 1.5$$

est positive quand l'eider est femelle (elle vaut $z = 1 + 1 - 1.5 = 0.5$) et négative quand il est mâle (elle vaut $z = 1 + 0 - 1.5 = -0.5$ ou $z = 0 + 1 - 1.5 = -0.5$). Donc, le concept est aussi exprimé par le signe de la valeur z.

Le concept appris est par conséquent ramené à une décision par surface séparatrice linéaire, non pas dans l'espace de représentation de départ, réduit à x, mais sur deux valeurs binaires y_1 et y_2, extraites elles-mêmes de décisions linéaires sur x.

En quoi cette technique de décision est-elle conforme aux premières notions que nous avons données sur les réseaux connexionnistes? Il suffit pour le voir de se placer dans une représentation graphique (figure 10.3). L'information se propage de la gauche vers la droite. Les deux cercles centraux ont pour valeur de sortie y_1 et y_2, celui de droite a pour valeur de sortie z ; il exprime donc le concept cherché. Cette représentation sera bien sûr plus longuement expliquée dans ce chapitre.

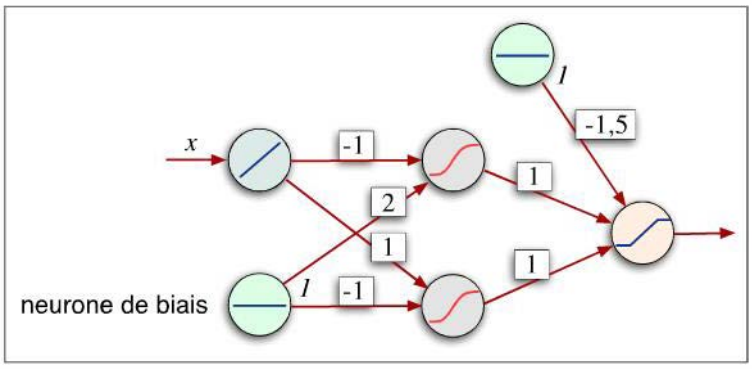


FIG. 10.3: *Un réseau connexionniste qui distingue les eiders mâles des eiders femelles.*

1. Les différents éléments d'un réseau connexionniste

Détaillons d'abord les différentes notions nécessaires à la compréhension des réseaux connexionnistes, en particulier des réseaux *multicouches* que nous étudions dans ce chapitre.

L'espace de représentation

Les données d'entrée sont des vecteurs de $\mathbb{R}^d$, notés comme d'habitude (en transposition) $\mathbf{x}^\top = (x_1, \ldots, x_d)$. Les réseaux connexionnistes que nous présentons dans ce chapitre sont donc des règles de classification de données numériques.

Le neurone formel

L'unité de traitement élémentaire dans un réseau connexionniste n'est capable que de réaliser certaines opérations simples. Ces unités sont souvent appelées *neurones formels* pour leur simi-

Copyright © 2010 Eyrolles.

litude grossière avec les neurones du cerveau. Les modèles de réseaux connexionnistes qui nous intéressent particulièrement, les *réseaux multicouches*, classent les unités selon qu'elles sont des neurones d'entrée, cachés, ou de sortie.

- Un *neurone d'entrée* ou, simplement, une *entrée*, est une unité chargée de transmettre une composante du vecteur x des données (en particulier, les données d'apprentissage pendant la phase d'apprentissage).
- Un *neurone de sortie* est une unité qui fournit une hypothèse d'apprentissage, par exemple dans un problème de classification, une décision sur la classe à laquelle est attribué x.
- Enfin, un *neurone caché* est un neurone qui n'est ni un neurone d'entrée, ni un neurone de sortie. Sa fonction est de faire des traitements intermédiaires.

Il existe d'autres modèles, par exemple la *machine de Boltzmann* pour laquelle tous les neurones formels, y compris d'entrée et de sortie, sont connectés les uns aux autres.

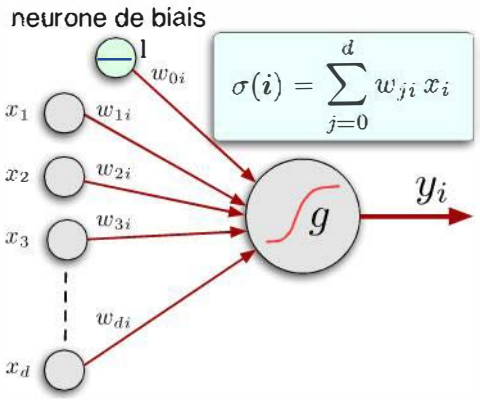
L'état d'un neurone formel

Il est commode de décrire un réseau connexionniste à un moment de son fonctionnement par un ensemble de valeurs σ_i, une pour chaque neurone formel i. Lorsque le neurone i est un neurone d'entrée, on a : $\sigma_i = x_i$ où x_i est la composante de rang i du vecteur x. Dans tous les autres cas, σ_i est l'*état* du neurone i, calculé par la règle de propagation décrite ci-dessous au paragraphe 2.1.

Comment fonctionne un neurone formel

Un neurone formel est caractérisé par une *une fonction de sortie* g (parfois aussi appelée *fonction d'activation*) qui permet de calculer pour chaque neurone i une valeur de sortie y_i en fonction de son état d'activation σ_i :

$$y_i = g(\sigma_i) \qquad (10.1)$$



TAB. 10.1: *Modèle d'un neurone formel.*

Remarque

La notation y est utilisé dans ce chapitre par tradition. Elle désigne le calcul effectué par le classificateur et non pas, comme dans le chapitre 2 la véritable sortie que l'on cherche à approcher. Tout ce qu'on connaît de cette véritable sortie est la supervision u, ou sortie désirée, fournie par l'expert et supposée égale à cette sortie véritable sur les exemples d'apprentissage.

On peut envisager plusieurs sortes de fonctions de sortie, mais le plus souvent on utilise soit la fonction signe (comme dans l'exemple d'introduction), soit une fonction sigmoïde[3] d'équation

$$y_i = g(\sigma_i) = \frac{1}{1 + e^{-\lambda \sigma_i}}$$

La figure 10.4 montre le graphe de cette fonction pour les valeurs $\lambda = 0.1, 1, 2$ et 10.

[3] C'est-à-dire en forme de s.

Copyright © 2010 Eyrolles.

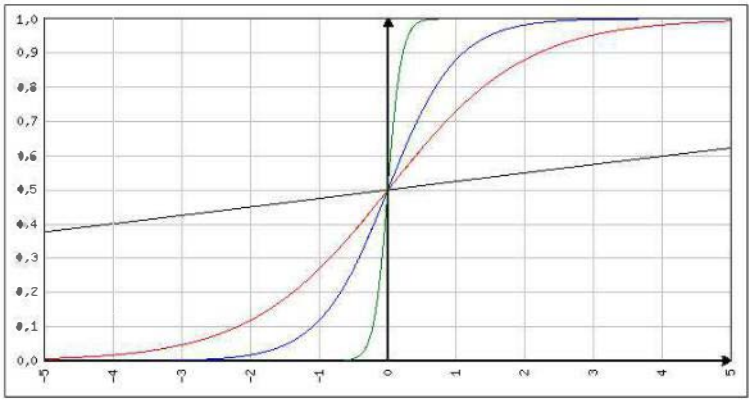


FIG. 10.4: *Graphe de la fonction* sigmoïde *d'équation* $y_i = g(\sigma_i) = \frac{1}{1+e^{-\lambda\sigma_i}}$. *Cette fonction est paramétrée par sa pente à l'origine* λ. *Pour* λ *très grand, on approche de la fonction seuil. Pour* λ *très petit, cette fonction est pratiquement linéaire dans une vaste région autour de l'origine. Les sigmoïdes de cette figure correspondent à* $\lambda = 0.1, 1, 2$ *et* 10 *pour* $\sigma_i \in [-5,5]$.

2. L'architecture multicouche

Un réseau est caractérisé par son architecture, c'est-à-dire la structure selon laquelle les neurones formels qui le composent sont reliés les uns aux autres. Certains réseaux, comme les machines de Boltzmann, ont une connectivité complète (chaque neurone formel est relié à toutes les autres unités) ; d'autres, ceux dont on va parler dans ce chapitre, ont une architecture en couches successives. La caractéristique de ces réseaux est que les unités d'une couche sont reliées à toutes celles de la couche suivante mais à aucune autre. À chaque lien entre deux unités i et j, on associe un poids correspondant à la force de la connexion entre ces deux unités, noté $w(i,j)$, ou w_{ij}.

Dans ces modèles, la couche d'entrée sert à la lecture des données et la couche de sortie à traduire la décision. En général, il s'agit d'une décision de classification.

2.1 La transmission de l'information dans un réseau multicouche

Le fonctionnement d'un réseau connexionniste, pour des poids de connexions donnés, se résume à définir une *règle de propagation,* qui décrit comment calculer l'état d'activation d'une unité j en fonction des unités i pour lesquelles il existe un poids $w(i,j)$. Appelons $source(j)$ l'ensemble de ces unités i. Le shéma est donné à la figure 10.5.

La règle la plus souvent utilisée consiste à calculer la somme des valeurs de sortie y_i des unités $i \in source(j)$, pondérées par les poids des connexions correspondantes.

$$\sigma_j = \sum_{i \in source(j)} w(i,j)\, y_i$$

Cette règle n'est pas utilisée pour calculer l'état d'activation des neurones d'entrée, puisque leur rôle est simplement de transmettre les entrées. Dans leur cas, on a donc simplement $\sigma_j = x_j$.

Copyright © 2010 Eyrolles.

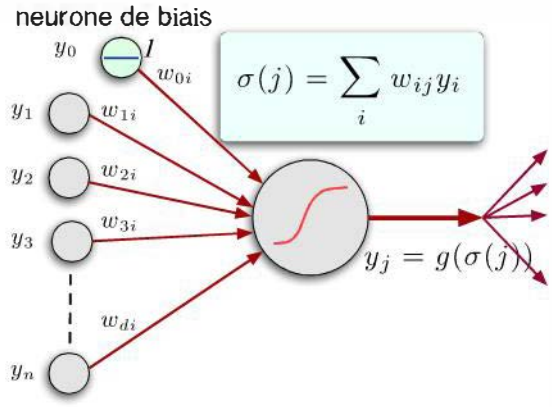


FIG. 10.5: *Le neurone formel : notation des connexions.*

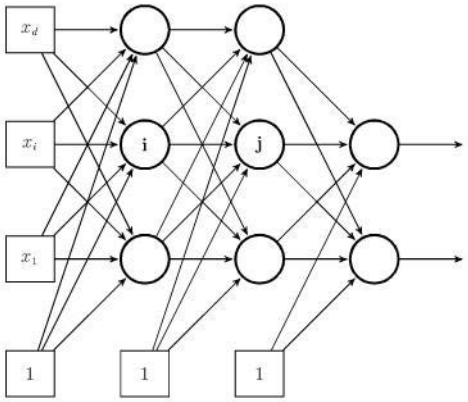


FIG. 10.6: *Le réseau multicouche général. Chaque entrée est reliée à tous les éléments de la première couche cachée et seulement à eux. Chaque élément d'une couche cachée est relié à tous les éléments de la couche cachée suivante et seulement à eux. Chaque élément de la dernière couche cachée est relié à tous les éléments de la couche de sortie et seulement à eux.*

De plus, on doit ajouter au vecteur d'entrée une composante supplémentaire, appelée *offset*[4], noté y_0 dans la figure 10.5 et dont la valeur est le plus souvent fixée arbitrairement à 1. Bien que ce ne soit pas nécessaire, une entrée du même type est souvent ajoutée à chaque couche cachée, pour des raisons d'homogénéité dans les algorithmes de reconnaissance et d'apprentissage. Pour chaque couche, on ajoute donc un neurone formel dans lequel n'arrive aucune connexion, dont l'activité est toujours égale à 1 et dont les transitions vers les neurones formels j de la couche supérieure sont notées $w(0, j)$.

Finalement, chaque neurone j prend l'état :

$$\sigma_j = w(0,j) + \sum_{i \in source(j)} w(i,j)\, y_i \tag{10.2}$$

et produit la sortie :

$$y_j = g(\sigma_j) \tag{10.3}$$

[4] Le mot français serait malheureusement « biais », qui a un autre sens en apprentissage.

Copyright © 2010 Eyrolles.

En résumé, dans le cas d'un *réseau à couches*, la couche d'entrée est activée par l'arrivée d'une donnée, en recevant une composante du vecteur x sur chacune de ses unités. La première couche cachée effectue le calcul ci-dessus (équations 10.2 et 10.3) pour chacune de ses unités, puis c'est au tour de la seconde couche cachée, etc. Finalement, l'unité de la couche de sortie ayant la valeur la plus forte indique la classe calculée pour l'entrée. Un réseau connexionniste multicouche général est représenté sur la figure 10.6.

EXEMPLE **Propagation des valeurs dans un réseau multicouche**

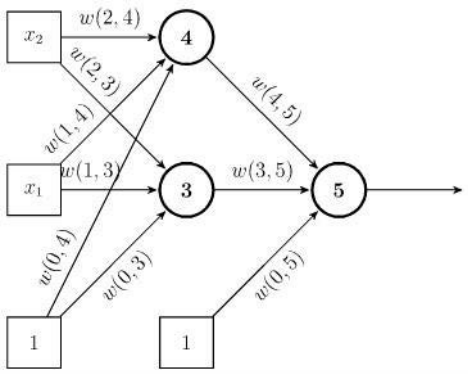


FIG. 10.7: *Un exemple de réseau multicouche : la notation des neurones formels et des poids des connexions. Les entrés sont des carrés, les neurones sont des cercles (leur numéro est en gras).*

Considérons le réseau à deux entrées x_1 et x_2 de la figure 10.7. Il possède une couche cachée composée des neurones formels numérotés 3 et 4. Sa couche de sortie est composée d'un seul neurone formel, numéroté 5. Il y a une valeur d'offset fixée à 1 pour le vecteur d'entrée et une autre pour la couche cachée.

Fixons maintenant (figure 10.7) la valeur des poids comme suit :

$$\begin{array}{lll} w(0,3) = \mathbf{0}.2 & w(1,3) = 0.1 & w(2,3) = 0.3 \\ w(0,4) = -\mathbf{0}.3 & w(1,4) = -0.2 & w(2,4) = 0.4 \\ w(0,5) = \mathbf{0}.4 & w(3,5) = 0.5 & w(4,5) = -\mathbf{0}.4 \end{array}$$

et prenons pour vecteur d'entrée $\mathrm{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

La propagation des calculs s'effectue alors comme indiqué dans la table ci-dessous :

Neurone formel j	σ_j	y_j
3	$\mathbf{0}.2 + 0.1 \times 1 + 0.3 \times 1 = 0.6$	$\frac{1}{1+e^{-0.6}} \simeq \mathbf{0.65}$
4	$-\mathbf{0}.3 + -\mathbf{0}.2 \times 1 + 0.4 \times 1 = -0.1$	$\frac{1}{1+e^{0.1}} \simeq \mathbf{0.48}$
5	$0.4 + 0.5 \times \mathbf{0.65} - \mathbf{0}.4 \times \mathbf{0.48} = \mathbf{0}.53$	$\frac{1}{1+e^{-0.53}} \simeq \mathbf{0}.63$

L'intérêt des couches cachées avec neurones à fonction d'activation non linéaire est qu'elle permettent une redescription des entrées dans un autre espace. Les neurones de la dernière couche, même s'ils sont linéaires, peuvent alors simplement effectuer la séparation ou la régression nécessaire. C'est ce que nous allons illustrer avec le cas de la fonction XOR.

Copyright © 2010 Eyrolles.

EXEMPLE **Le problème de la fonction « XOR »**

Cet exemple a pour but de montrer comment l'introduction de la fonction non linéaire de sortie des neurones formels et de l'architecture en couches permet d'obtenir des surfaces séparatrices non linéaires.

Plaçons nous dans $\mathbb{R}^2$, avec quatre points d'apprentissage situés au quatre coins du carré unité. Chaque paire de points opposés en diagonale forme une classe. Les exemples ont donc la forme suivante :

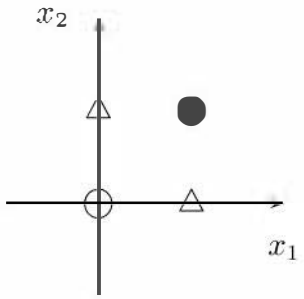


FIG. 10.8: *Le problème XOR : les deux points △ sont des exemples de la même classe, les deux points ○ des exemples d'une autre classe. Les deux classes ne sont pas linéairement séparables dans le plan* $\{x_1, x_2\}$.

Le réseau connexionniste de la figure 10.9 permet de résoudre le problème. En choisissant $y_i = g(\sigma_i)$ comme la fonction seuil, la propagation des calculs se fait comme indiqué dans le tableau associé.

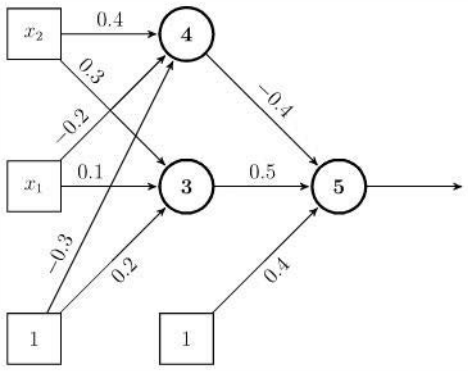


FIG. 10.9: *Un réseau réalisant la fonction « XOR ».*

x_1	x_2	σ_3	y_3	σ_4	y_4	σ_5	y_5
●	0	-0.5	0	-1.5	●	-0.5	0
●	1	0.5	1	-0.5	●	●.5	1
1	0	0.5	1	-0.5	●	●.5	1
1	1	1.5	1	●.5	1	-0.5	0

On voit dans ce tableau que les points d'entrée ont été re-décrits dans le plan $\{y_3, y_4\}$. Les deux points ● ont conservés leurs coordonnées (0,0) et (1,1). Mais les deux points △ sont maintenant confondus en (1,●). Il devient ainsi possible de les séparer des points par une séparatrice linéaire.

Copyright © 2010 Eyrolles.

Un interprétation est possible en logique booléenne : au lieu de considérer les valeurs 0 et 1 comme des coordonnées numériques, prenons-les comme des valeurs logiques. Dans ce cas, on peut interpréter les sorties intermédiaires et la sortie finale comme des fonctions booléennes sur les entrées. Le réseau réalise la fonction XOR (le *OU exclusif*), qui vaut 0 pour les deux points △ et 1 pour les deux points ●. Ceci est rendu possible par les non-linéarités du système de calcul (les sorties des unités sont bornées dans $]0,1[$) (voir la figure 10.10).

$$\begin{aligned} y_3 &= x_1 \vee x_2 \\ y_4 &= x_1 \wedge x_2 \\ y_5 &= y_3 \wedge \neg y_4 = x_1 \ XOR \ x_2 \end{aligned}$$

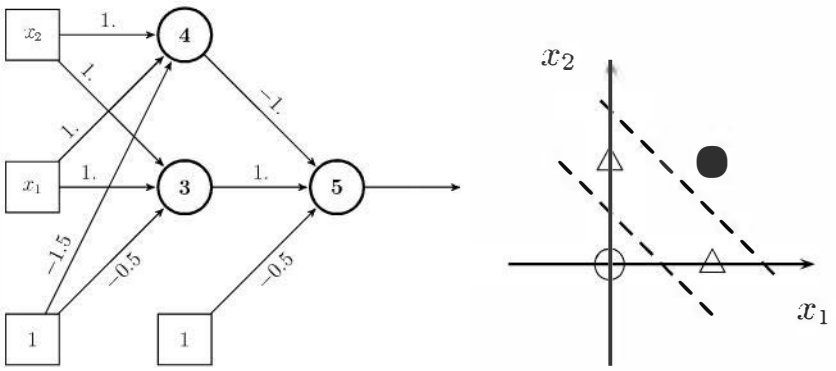


FIG. 10.10: *Une des façons de résoudre le problème XOR avec un réseau connexionniste à une couche cachée. La zone affectée à la classe △ est le « couloir » compris entre les deux droites en pointillé, celle affectée à la classe ◯ est à l'extérieur. La première droite répond à l'équation $x_1 + x_2 - 0.5 = 0$ et réalise un OU logique. La seconde répond à l'équation $x_1 + x_2 - 0.5 = 0$ et réalise un ET logique. Elles sont réalisées par la première couche du réseau. La seconde couche combine les deux décisions linéaires en une décision non linéaire.*

2.2 Le protocole d'apprentissage

Dans le cas des réseaux connexionnistes, les données d'apprentissage sont en général présentées séquentiellement ; l'apprentissage est donc incrémental. Chaque étape emploie une donnée pour modifier les poids des connexions. La suite des données utilisées peut être construite par un tirage aléatoire avec remise dans l'ensemble des exemples ou par plusieurs passages successifs de la totalité de cet ensemble. Au total, le nombre de données utilisées pour l'apprentissage est en général bien supérieur au nombre d'exemples : chacun est utilisé en moyenne ou exactement un grand nombre de fois (couramment une centaine de fois).

Copyright © 2010 Eyrolles.

2.3 Le codage des exemples d'apprentissage

Les modèles connexionnistes dont nous parlons utilisent pour leur apprentissage des données couplées à une classe d'appartenance, laquelle est représentée par un autre vecteur de *sorties désirées* noté $\mathbf{u}$ (conformément aux notations du chapitre 2 ; voir également la page de notations, page xxvii) ; ce vecteur est de dimension égale aux nombre C de classes.

Un exemple d'apprentissage $\mathbf{z} = (\mathbf{x}, \mathbf{u})$ est donc composé d'un vecteur $\mathbf{u}$ de sortie désirée associé à un vecteur d'entrée $\mathbf{x}$. Par conséquent, chaque neurone formel de sortie correspond à une classe et une seule[5].

L'apprentissage d'une règle de classification se fera en général en attribuant une classe à chaque coordonnée du vecteur des sorties : la classe ω_1 sera codée $\mathbf{u}^T = (1, 0, 0, ...0)$, la classe ω_C : $\mathbf{u}^T = (0, 0, ...1)$.

Un réseau connexionniste multicouche peut en réalité apprendre des associations plus complexes : la sortie étant un vecteur de $\mathbb{R}^C$, il peut approximer toute fonction de $\mathbb{R}^d$ dans $\mathbb{R}^C$.

3. L'algorithme d'apprentissage

La caractéristique la plus intéressante d'un réseau de neurones artificiels est sa capacité d'apprendre, c'est-à-dire de modifier les poids de ses connexions en fonction des données d'apprentissage, de telle sorte qu'après un certain temps d'entraînement il ait acquis une faculté de généralisation.

Pour procéder graduellement, nous allons d'abord redécrire l'algorithme du perceptron, en le considérant comme un réseau connexionniste. Nous montrerons ensuite que cet algorithme d'apprentissage peut être considéré comme un problème d'optimisation qui se résout par une méthode de gradient. Ce qui permettra de généraliser d'abord à un perceptron travaillant avec plus de deux classes, puis au réseau connexionniste multicouche.

3.1 Retour sur le perceptron

3.1.1 Le perceptron pour deux classes

Fonction seuil en sortie

Le *perceptron* a déjà été étudié au chapitre 9 dans le cadre des séparateurs linéaires. On va le voir ici comme un réseau connexionniste à couches, comportant une couche de neurones d'entrées, un neurone de sortie unique et pas de couche cachée. Les connexions sont donc établies directement entre la couche d'entrée et le neurone de sortie, ce qui se traduit dans le cas de deux classes par une décision par seuil sur une combinaison linéaire des valeurs d'entrée.

La figure 10.11 montre comment le perceptron peut être représenté comme un réseau connexionniste sans couche cachée, avec un seul neurone formel de sortie.

f est ici la fonction seuil, calculée à partir de σ de la manière suivante :

$$\sigma \geq 0 \Leftrightarrow y = g(\sigma) = 1$$

$$\sigma \leq 0 \Leftrightarrow y = g(\sigma) = 0$$

[5] Il existe d'autres codages dans lesquels une classe est associée à un sous-ensemble de neurones de sortie. Ceci permet en particulier d'utiliser la technique des codes correcteurs d'erreur (voir le chapitre 17).

Copyright © 2010 Eyrolles.

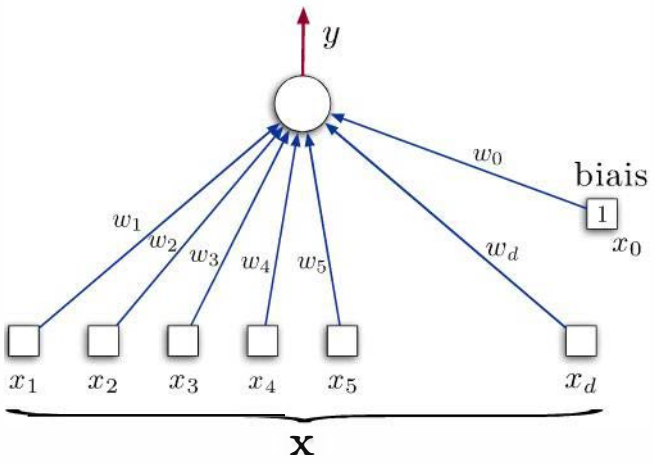


FIG. 10.11: *Le perceptron de Rosenblatt est le réseau connexionniste le plus simple. Il effectue le calcul :* $y = g(\sigma) = w_0 + \sum_{i=1}^{d} w_i\, x_i$ *où* x_i *est la* i*-ème composante de* $\mathbf{x}$.

L'apprentissage dans le perceptron se fait par la règle de modification des poids qui a été décrite au chapitre 9. Dans le cas de deux classes, il n'y a qu'une seule sortie, et la décision d'appartenance à une des deux classes est prise en comparant la valeur de sortie à un seuil. L'algorithme d'apprentissage se contente de modifier le vecteur des poids en lui ajoutant ou lui enlevant un vecteur proportionnel à l'entrée $\mathbf{x}$, dans le cas où celle-ci conduit à une valeur du mauvais côté du seuil; dans le cas contraire, il n'effectue aucune action.

Au chapitre 9, nous avons formalisé cet apprentissage, pour deux classes ω_1 et ω_2 par l'algorithme 10.

Récrivons-le un peu différemment pour nous placer dans les notations de ce chapitre. Le vecteur $\mathbf{a}$ du chapitre 9 correspond directement à l'ensemble des poids des connexions. Nous notons ici $w(i)$ le poids de la connexion menant de l'entrée x_i au neurone de sortie.

$\mathbf{u}$, σ et $\mathbf{y}$ sont pour le moment des vecteurs ayant une seule composante. Le perceptron effectue donc le calcul fondé sur l'équation 10.2 :

$$\mathbf{y} = g(\sigma) = g\left(w(0) + \sum_{i=1}^{d} w(i)\, x_i\right)$$

En phase d'apprentissage, la modification apportée à $w(i)$ au cours de l'apprentissage pour le mener de sa valeur à l'instant t à celle à l'instant $t+1$ par une entrée $\mathbf{x}$ se note Δ_i ; elle peut maintenant s'écrire de la manière suivante :

$$\Delta_i = \alpha\, x_i\, (\mathbf{u} - \mathbf{y}) \tag{10.4}$$

En effet, quand $\mathbf{x}$ est bien classé, le terme $(\mathbf{u} - \mathbf{y})$, qui est un scalaire (les vecteurs $\mathbf{u}$ et $\mathbf{y}$ sont de dimension 1), vaut 0. Quand $\mathbf{x}$ est mal classé, ce terme vaut $+1$ ou -1 selon que $\mathbf{x}$ est un exemple ou un contre-exemple. L'algorithme d'apprentissage devient donc :

Fonction identité en sortie

Pour simplifier, enlevons provisoirement la fonction seuil à la sortie. On a désormais :

$$\mathbf{y} \equiv \sigma$$

Copyright © 2010 Eyrolles.

Algorithme 13 : Reformulation de l'apprentissage du perceptron

début
  Prendre $a_{(0)}$ quelconque et α positif quelconque
  $t = 0$
  tant que $t \leq t_{max}$ **faire**
    tirer au hasard une donnée d'apprentissage x parmi les m
    pour $i = 1, d$ **faire**
      $\Delta_i \leftarrow \alpha\, x_i\, (\mathbf{u} - \mathbf{y})$
      $w_i \leftarrow w_i + \Delta_i$
    fin pour
    $t \leftarrow t + 1$
  fin tant que
fin

Rien n'empêche d'appliquer la même modification de poids que précédemment, c'est-à-dire :

$$\Delta_i = \alpha\, x_i\, (\mathbf{u} - \sigma)$$

On se trouve alors dans un apprentissage du type « punition-récompense » : les poids sont modifiés proportionnellement à l'entrée et à la conformité de la sortie avec la sortie désirée. Pour être plus précis, définissons par

$$E = \frac{1}{2}(\mathbf{u} - \mathbf{y})^2 \tag{10.5}$$

la fonction d'erreur entre la sortie $\mathbf{y} = \sum_{i=0}^{d} w_i\, x_i$ et la sortie désirée $\mathbf{u}$. Puisque E ne dépend de w_i que par l'intermédiaire de $\mathbf{y}$, on peut appliquer la règle de chaînage des dérivations partielles de la manière suivante :

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial w_i} = -(\mathbf{u} - \sigma) x_i$$

Δ_i peut donc être interprétée comme une modification de w_i par la technique du gradient pour minimiser E.

Voyons maintenant comment on peut, en gardant le même principe d'apprentissage, généraliser le perceptron pour lui donner des capacités beaucoup plus fortes, selon deux directions : en augmentant la taille de sa couche de sortie, puis en introduisant des couches cachées.

3.1.2 Le perceptron pour plus de deux classes

La transformation précédente permet maintenant d'appliquer directement le même calcul à l'apprentissage d'une règle de classification pour un nombre quelconque C de classes. Il suffit, comme on l'a vu, de construire pour chaque donnée d'apprentissage un vecteur de sortie désirée $\mathbf{u}$ de dimension C valant 1 sur la coordonnée correspondant à la classe et 0 partout ailleurs (figure 10.12).

En notant $w(i, j)$ le poids de la connexion menant de l'entrée d'indice i à la sortie d'indice j, on peut calculer pour une entrée x donnée chaque sortie par :

$$y_j = \sigma_j = w(0, j) + \sum_{i=1}^{i=d} w(i, j) x_i$$

Idéalement, un perceptron ayant parfaitement appris les données d'apprentissage devrait fournir pour chacune d'elles une sortie $\mathbf{y} = \sigma$ égale à la sortie désirée $\mathbf{u}$.

Copyright © 2010 Eyrolles.

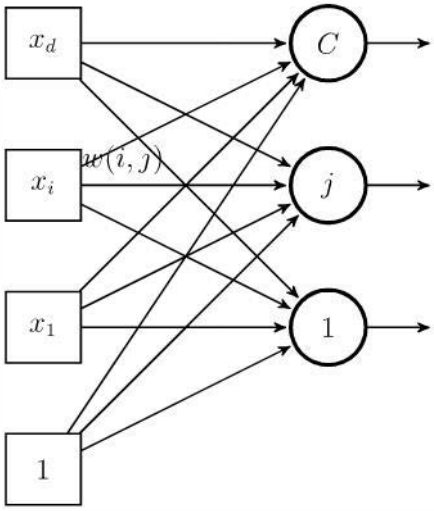


FIG. 10.12: *Le perceptron pour C classes.*

Notons ici que les surfaces séparatrices produites par le perceptron pour C classes sont encore de nature linéaire : chaque classe est délimitée par une frontière convexe composée par l'intersection d'hyperplans. En dimension 2, pour $C = 5$, on a des séparatrices du type de celles de la figure 10.13.

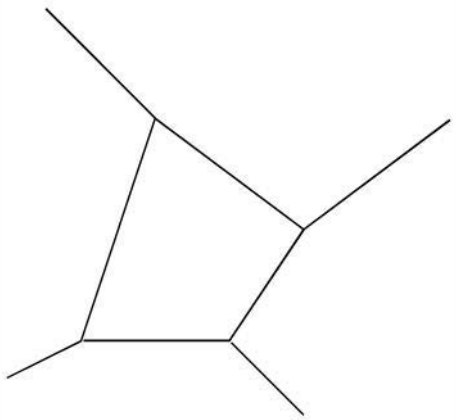


FIG. 10.13: *Un exemple de surfaces séparatrices du perceptron pour 5 classes en dimension 2.*

On mesure alors l'*erreur de classification pour l'entrée* $\mathbf{x}$ comme la distance euclidienne[6] $D(\mathbf{u},\mathbf{y})$ entre la sortie désirée $\mathbf{u}$ et la sortie calculée $\mathbf{y}$:

$$\mathcal{D}(\mathbf{u},\mathbf{y}) = \mathcal{D}(\mathbf{u},\sigma) = \frac{1}{2}\sum_{j=1}^{j=C}(u_j - \sigma_j)^2 \tag{10.6}$$

On peut appliquer pour l'apprentissage une généralisation de la technique du paragraphe précé-

[6] On note dans ce chapitre la distance euclidienne D et non pas Δ, pour éviter la confusion avec la notation traditionnelle de la « règle delta » de l'apprentissage des réseaux connexionnistes.

Copyright © 2010 Eyrolles.

dent : la *règle delta*. Elle consiste à modifier le poids $w(i,j)$ d'une quantité :

$$\Delta_{ij} = \alpha\, x_i\, (u_j - y_j) \tag{10.7}$$

où α est une valeur positive comprise entre 0 et 1.

Ceci revient encore à appliquer la technique de l'*optimisation par gradient* (voir l'annexe 8), qui consiste ici à constater que la contribution du poids $w(i,j)$ à l'erreur $D(\mathbf{u},\mathbf{y})$ peut s'écrire :

$$\frac{\partial}{\partial w(i,j)} D(\mathbf{u},\sigma)$$

Puisque l'on a :

$$\sigma_j = w(0,j) + \sum_{i=1}^{i=d} w(i,j)\, x_i$$

et :

$$D(\mathbf{u},\mathbf{y}) = D(\mathbf{u},\sigma) = \frac{1}{2} \sum_{j=1}^{j=C} (u_j - \sigma_j)^2$$

le calcul s'effectue donc ainsi :

$$\begin{aligned} \frac{\partial}{\partial w(i,j)} D(\mathbf{u},\sigma) &= \frac{\partial}{\partial \sigma_j} D(\mathbf{u},\sigma) \cdot \frac{\partial \sigma_j}{\partial w(i,j)} \\ &= \frac{1}{2} \cdot \frac{\partial}{\partial \sigma_j} (u_j - \sigma_j)^2 \cdot \frac{\partial \sigma_j}{\partial w(i,j)} \\ &= (\sigma_j - u_j)\, x_i \\ &= (y_j - u_j)\, x_i \end{aligned}$$

Selon la technique du gradient, $w(i,j)$ doit être modifié d'une valeur Δ_{ij} proportionnellement et en sens inverse à la contribution du poids $w(i,j)$ à l'erreur $D(\mathbf{u},\mathbf{y})$. D'où la formule 10.7 donnée ci-dessus pour la règle delta.

La règle delta est donc à la fois une généralisation de la règle d'apprentissage du perceptron pour le cas à deux classes et une technique de minimisation par gradient de l'erreur quadratique moyenne.

3.1.3 Plus de deux classes et une sigmoïde en sortie

Dans ce paragraphe, nous continuons à progresser en supposant que la sortie σ_j est transformée en $y_j = g(\sigma_j)$, où f est la fonction sigmoïde de paramètre $\lambda = 1$ (voir la figure 10.4) :

$$y_j = g(\sigma_j) = \frac{1}{1 + e^{-\sigma_j}}$$

On a alors :

$$\frac{\partial}{\partial w(i,j)} \left(\frac{1}{2}(u_j - y_j)^2\right) = \frac{1}{2}\left(\frac{\partial}{\partial y_j}(u_j - y_j)^2\right) \cdot \frac{\partial y_j}{\partial \sigma_j} \cdot \frac{\partial \sigma_j}{\partial w(i,j)} = (y_j - u_j)\, y_j\, (1 - y_j)\, x_i$$

D'où :

$$\Delta_{ij} = \alpha (u_j - y_j)\, y_j\, (1 - y_j)\, x_i \tag{10.8}$$

En effet, comme indiqué dans l'annexe 9, la fonction f répond à l'équation différentielle :

$$g' = g(1 - g)$$

Copyright © 2010 Eyrolles.

3.2 L'apprentissage par rétropropagation du gradient de l'erreur

C'est seulement en 1986 que la généralisation de la règle delta aux réseaux à couches cachées a été formulée. Cette généralisation, *la règle de la rétropropagation du gradient de l'erreur*, consiste à propager l'erreur obtenue à une unité de sortie d'un *réseau à couches* comportant une ou plusieurs couches cachées à travers le réseau par descente du gradient dans le sens inverse de la propagation des activations. La figure 10.14 montre une illustration du principe.

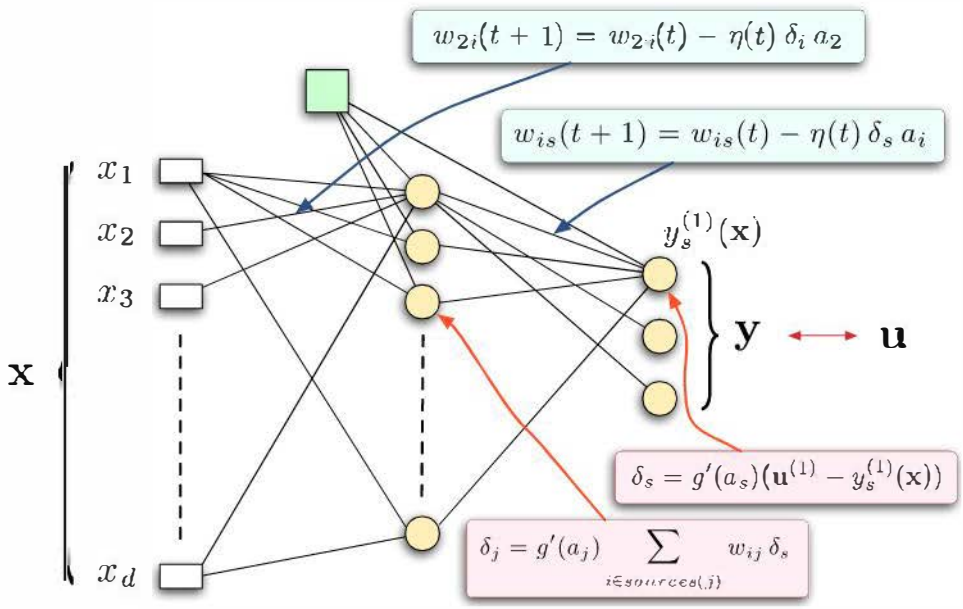


FIG. 10.14: *Schéma du modèle de la rétropropagation de l'erreur. La modification à apporter aux poids entre la couche source(j) et le neurone formel j ne peut être calculée que si on connaît déjà la modification qu'il faut apporter aux poids w(j, k) entre j et les éléments de dest(j).*

Rappellons qu'un réseau à couches est composé d'un ensemble de neurones formels groupés en sous-ensembles distincts (les couches) de telle sorte qu'il n'y ait aucune connexion entre deux neurones d'une même couche[7].

À la fin de l'apprentissage, lorsque le réseau a appris à modéliser son environnement, le comportement souhaité du réseau est le suivant : on présente un vecteur d'entrée au réseau, celui-ci propage vers la sortie les valeurs d'activation correspondantes (en utilisant une règle de propagation), afin de générer, par l'intermédiaire des neurones de sortie, un vecteur de sortie. Celui-ci devrait correspondre à la sortie désirée, telle qu'apprise lors de la phase d'apprentissage.

La généralisation de la règle delta aux réseaux multicouches utilise une méthode de *descente du gradient*, permettant de calculer la modification des poids des connexions entre les couches cachées (pour plus de détails, voir l'annexe 9 ou la référence [RHW86]). Afin de pouvoir calculer le gradient de l'erreur par rapport aux poids du réseau, la fonction de sortie d'un neurone doit être différentiable et non linéaire (sinon, on pourrait réduire le réseau à un perceptron). La fonction la plus souvent utilisée est, comme on l'a déjà dit, la *sigmoïde* :

[7] Une amélioration de la règle de rétropropagation permet l'introduction de cycles entre les couches, pour obtenir des *réseaux récurrents*.

Copyright © 2010 Eyrolles.

$$y_j = g(\sigma_j) = \frac{1}{1 + e^{-\sigma_j}} \tag{10.9}$$

La règle delta généralisée dicte alors le changement de poids entre le neurone i et le neurone j de la façon suivante :

$$\Delta w(i,j) = \alpha\ \delta_j\ y_i \tag{10.10}$$

c'est-à-dire de façon proportionnelle à une mesure d'erreur δ_j caractéristique du neurone j et à la valeur d'entrée notée ici[8] y_i. Pour les connexions aboutissant aux neurones de sortie, cette mesure d'erreur est évidemment calculée ainsi :

$$\delta_j = (u_j - y_j)\ y_j\ (1 - y_j) \tag{10.11}$$

Le calcul de l'erreur aux unités cachées se fait ensuite récursivement par la descente du gradient. Soit $dest(j)$ l'ensemble des neurones auxquels j se connecte :

$$\delta_j = y_j\ (1 - y_j) \sum_{k \in dest(j)} \delta_k\ w(j,k) \tag{10.12}$$

Le calcul est détaillé dans l'annexe 9.

Lorsque l'on applique la règle delta généralisée sur le réseau de façon itérative pour un ensemble de vecteurs d'entrées (correspondant à l'environnement), le réseau tentera de minimiser l'erreur obtenue à la sortie, et donc de modéliser le mieux possible la fonction désirée entre les entrées et les sorties.

3.3 L'organisation des calculs

Les calculs s'organisent de la façon donnée dans l'algorithme 14. Le point à remarquer est que l'actualisation des poids ne se fait qu'une fois la rétropropagation terminée : il ne faut en effet pas changer trop tôt la valeur d'un poids puisque celle-ci intervient dans le calcul concernant la couche suivante.

EXEMPLE **revisité...**

Reprenons l'exemple du paragraphe 10.7 en supposant que la sortie désirée au vecteur d'entrée $\mathbf{x}^\top = (1,1)$ vaille $\mathbf{u} = 0$. Après modification des poids sur cet exemple, son nouveau passage dans le réseau doit conduire à une sortie inférieure à la valeur précédente, qui était de 0.63.

Pour le neurone formel de sortie, on a :

$$\Delta w(i,j) = \alpha\ \delta_j\ y_i$$

avec :

$$\delta_j = (u_j - y_j)\ y_j\ (1 - y_j)$$

On prend d'abord : $i = 3$ et $j = 5$, ce qui mène à :

$$\delta_5 = (0. - 0.63) \times 0.63 \times (1. - 0.63) = -0.147$$

d'où :

$$\Delta w(3,5) = -0.147 \times 0.65 \simeq -0.1$$

[8] C'est en effet la sortie du neurone i.

Copyright © 2010 Eyrolles.

Algorithme 14 : Apprentissage du perceptron multicouche

```
début
    tant que l'apprentissage n'a pas convergé faire
        tirer au hasard un point d'apprentissage
        pour chaque couche, en partant de celle du haut faire
            pour chaque neurone formel de cette couche faire
                calculer δ_j
                pour chaque connexion w(i,j) menant au neurone formel j faire
                    calculer Δw(i,j) = α δ_j y_i
                fin pour
            fin pour
        fin pour
        pour chaque connexion w(i,j) faire
            w(i,j) ← w(i,j) + Δw(i,j)
        fin pour
    fin tant que
fin
```

en fixant la valeur α à 1. De même, pour $i = 4$, on obtient :

$$\Delta w(4,5) = 0.48 \times -0.147 \simeq -0.07$$

$$\Delta w(0,5) = -0.147 \times 1. = -0.147$$

Pour le neurone formel caché noté 4, on a d'abord, puisque $dest(4) = \{5\}$:

$$\delta_4 = y_4 \times (1 - y_4) \times \delta_5 \times w(4,5) = 0.48 \times (1 - 0.48) \times -0.147 \times -0.4 = \simeq 0.015$$

D'où :

$$\Delta w(1,4) = 0.015 \times 1. = 0.015$$
$$\Delta w(2,4) = 0.015 \times 1. = 0.015$$
$$\Delta w(0,4) = 0.015 \times 1. = 0.015$$

De même, puisque $dest(3) = \{5\}$:

$$\delta_3 = y_3 \times (1 - y_3) \times \delta_5 \times w(3,5) = 0.65 \times (1 - 0.65) \times -0.147 \times 0.5 = \simeq -0.017$$

D'où :

$$\Delta w(1,3) = 0.016 \times 1. = -0.017$$
$$\Delta w(2,3) = 0.016 \times 1. = -0.017$$
$$\Delta w(0,3) = 0.016 \times 1. = -0.017$$

Copyright © 2010 Eyrolles.

Après modification, les poids deviennent donc :

$$\begin{aligned}
w(0,5) + \Delta w(0,5) &= 0.4 - 0.147 \simeq 0.25 \\
w(3,5) + \Delta w(3,5) &= 0.5 - 0.1 = 0.4 \\
w(4,5) + \Delta w(4,5) &= -0.4 - 0.07 = -0.47
\end{aligned}$$

$$\begin{aligned}
w(0,3) + \Delta w(0,3) &= 0.2 - 0.017 = 0.183 \\
w(1,3) + \Delta w(1,3) &= 0.1 - 0.017 = 0.083 \\
w(2,3) + \Delta w(2,3) &= 0.3 - 0.017 = 0.283
\end{aligned}$$

$$\begin{aligned}
w(0,4) + \Delta w(0,4) &= -0.3 + 0.015 = -0.285 \\
w(1,4) + \Delta w(1,4) &= -0.2 + 0.015 = -0.185 \\
w(2,4) + \Delta w(2,4) &= 0.4 + 0.015 = 0.415
\end{aligned}$$

Dans le réseau modifié, le calcul sur le vecteur d'entrée devient par conséquent :

Neurone formel j	σ_j	y_j
3	$0.183 + 0.083 \times 1 + 0.283 \times 1 = 0.55$	$\frac{1}{1+e^{0.55}} \simeq \mathbf{0.63}$
4	$-0.285 + -0.185 \times 1 + 0.415 \times 1 = -0.055$	$\frac{1}{1+e^{-0.055}} \simeq \mathbf{0.51}$
5	$0.25 + 0.4 \times \mathbf{0.63} - 0.47 \times \mathbf{0.51} = 0.26$	$\frac{1}{1+e^{0.205}} \simeq 0.56$

Si on compare la valeur de sortie à celle du tableau de la section 2.1 on constate qu'elle est passée de 0.63 avant apprentissage à 0.56 après : elle s'est rapprochée de la valeur désirée 0.

3.4 Une variante

Il est possible de transformer un réseau connexionniste multicouche en un système de décision bayésien (voir les chapitres 2 et 15) en changeant la distance entre la sortie calculée et la sortie désirée, sans modifier la règle de rétropropagation du gradient de l'erreur.

On désire ici une valeur du neurone de sortie y_j qui soit égale à la probabilité que le vecteur d'entrée appartienne à la classe j. Soit $X = (X_1, \ldots, X_C)$ une variable aléatoire multidimensionnelle qui représente, sous une hypothèse multinomiale, la distribution des sorties désirées. La probabilité de X s'exprime en fonction des probabilités *a priori* y_j d'appartenir à la classe j :

$$P(X_1 = u_1, \ldots, X_C = u_C) = \prod_{j=1}^{C} y_j^{u_j} (1 - y_j)^{\sum_{i=1}^{C} u_i - u_j}$$

Au maximum de vraisemblance, chercher les paramètres qui maximisent cette quantité est équivalent à minimiser la fonction d'erreur *entropie croisée* :

$$E = \sum_{j=1}^{C} -u_j \cdot Log(y_j) - (1 - u_j) \cdot Log(1 - y_j) \tag{10.13}$$

Il faut calculer :

$$\frac{\partial E}{\partial w(i,j)} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \sigma_j} \cdot \frac{\partial \sigma_j}{\partial w(i,j)}$$

Copyright © 2010 Eyrolles.

On a :

$$\begin{aligned}\frac{\partial E}{\partial y_j} &= -\frac{u_j}{y_j} + \frac{1-u_j}{1-y_j} = \frac{y_j - u_j}{y_j(1-y_j)} \\ \frac{\partial y_j}{\partial \sigma_j} &= y_j(1-y_j) \\ \frac{\partial \sigma_j}{\partial w(i,j)} &= y_i\end{aligned}$$

Finalement :

$$\frac{\partial E}{\partial w(i,j)} = (y_j - u_j) \cdot y_i \tag{10.14}$$

Il ne reste plus qu'à appliquer la règle de rétropropagation du gradient de l'erreur avec cette valeur, au lieu de la règle delta de l'équation 10.8, qui minimise la distance euclidienne entre la sortie calculée et la sortie désirée.

3.5 Quand arrêter l'apprentissage ?

Il est difficile de trouver un critère général pour arrêter cet algorithme. Le problème est que le risque empirique tend à diminuer lentement et à ne jamais se stabiliser complètement, ce qui mène à un surapprentissage. La meilleure manière d'éviter ce phénomène est d'utiliser un ensemble de validation (voir chapitre 3, paragraphe 4.3.1).

3.6 Le problème des minima locaux

Comme tous les algorithmes d'optimisation basés sur la descente du gradient, l'algorithme de la rétropropagation est susceptible de s'arrêter aux minima locaux. Par conséquent, la solution trouvée sera fortement reliée au choix des poids initiaux du réseau. Si les poids sont choisis près d'un minimum local sous-optimal, l'algorithme ne pourra pas trouver la solution désirée. Afin de contourner ce problème, on peut utiliser plusieurs techniques :

- Relancer l'apprentissage plusieurs fois en utilisant des poids initiaux différents, ce qui entraîne un temps de calcul plus élevé.
- Introduire du bruit dans la recherche pour pouvoir sortir des minima locaux.
- Utiliser les techniques avancées de descente de gradient : second ordre, gradient conjugué, etc. (voir [Bis95, Hay99]).

4. Quelques résultats théoriques sur les réseaux connexionnistes

Plusieurs résultats théoriques sur l'apprentissage des réseaux connexionnistes ont été obtenus, particulièrement sur leur pouvoir d'expression, leur complexité, ainsi que leur capacité de généralisation. Nous donnons dans cette section quelques-uns de ces résultats.

4.1 Pouvoir d'expression

Le *pouvoir d'expression* d'un réseau de neurones connexionniste, comme de toute règle de classification, est une mesure du nombre de fonctions différentes que celui-ci peut approximer. Il est en effet intéressant de connaître *a priori* les familles de fonctions auxquelles vont appartenir les

Copyright © 2010 Eyrolles.

surfaces de décision. Plusieurs résultats montrent par exemple qu'un réseau de neurones artificiels multicouche peut approximer avec une précision arbitraire n'importe quelle transformation continue d'un espace à dimension finie vers un autre espace à dimension finie, s'il possède suffisamment de neurones formels cachés (voir [Hay99]). En ce sens, on dit qu'il est un *approximateur universel*. Certains résultats montrent même qu'à l'exception de cas extrêmes, une seule couche cachée est suffisante.

Il faut cependant noter que ces résultats ne fournissent aucun indice sur la méthode à utiliser pour trouver directement les poids correspondant à l'approximation d'une fonction donnée. On ne sait les calculer que par apprentissage. Ce résultat était connu avant la découverte de l'algorithme de rétropropagation du gradient de l'erreur, qui a alors permis de l'utiliser en pratique.

4.2 Complexité

Les réseaux connexionnistes ayant un si grand pouvoir d'expression, il devient intéressant de connaître les aspects de complexité reliés à ce modèle (voir [Orp92] pour une revue). Ainsi, il a été montré le résultat suivant :

> Étant donné un réseau de neurones artificiels arbitraire R et une tâche arbitraire T devant être résolue par R, le problème consistant à décider si dans l'espace de tous les paramètres de R (ses poids, sa structure) il existe une solution qui résout adéquatement T, est *NP-difficile*.

Malgré cela, il est possible [Bau89] de trouver une solution (un ensemble de poids) pour T en temps polynomial si on peut utiliser des algorithmes d'apprentissage *constructifs*[9]. Il existe un certain nombre de ces algorithmes mais aucune preuve de convergence en temps polynomial n'existe actuellement pour eux.

D'un point de vue pratique, certaines expériences empiriques (dont [Hin89]) montrent qu'on peut faire apprendre une tâche complexe à un réseau de neurones artificiels en utilisant l'algorithme de la rétropropagation de l'erreur en temps $\mathcal{O}\left(W^3\right)$ où W représente le nombre de poids du réseau. En effet, bien qu'il faille un temps exponentiel (sur le nombre de poids) pour obtenir la solution optimale, on peut souvent en pratique se contenter d'une solution sous-optimale satisfaisante obtenue en temps polynomial.

5. Comment choisir l'architecture d'un réseau ?

Un des problèmes majeurs des réseaux connexionnistes est la difficulté de décider de leur architecture. Devant une tâche à résoudre par un réseau connexionniste, l'ingénieur doit prendre des décisions d'architecture non évidentes et pourtant très importantes : par exemple, il faut décider du nombre de neurones cachés, du nombre de couches cachées, et de leur interconnexion. Ceci se fait souvent de façon *ad hoc* ou en utilisant quelques règles heuristiques simples. Souvent on procède en essayant diverses architectures pour un problème donné et en calculant l'erreur de généralisation pour chacune sur un ensemble de validation. En effet, hormis une recherche exhaustive, aucune méthode n'est connue pour déterminer l'architecture optimale pour un problème donné. Or tous les résultats théoriques sur les réseaux connexionnistes (leur puissance de calcul ou leur faculté de généralisation) ne tiennent que si l'on utilise l'architecture idéale (ou tout au moins suffisante et nécessaire).

[9] Ayant la possibilité d'ajouter des neurones et des connexions durant l'apprentissage.

Copyright © 2010 Eyrolles.

Une solution à ce problème consiste à utiliser des algorithmes *constructifs* qui commencent avec une architecture minimale et ajoutent des neurones et des connexions au fur et à mesure de l'apprentissage. D'autres solutions utilisent plutôt une technique inverse : à partir d'une architecture complète, ils éliminent certains neurones et/ou connexions qui semblent non essentiels.

Il a également été proposé d'utiliser des méthodes d'optimisation pour chercher l'architecture idéale. Ainsi, plusieurs travaux font appel à des algorithmes génétiques pour optimiser l'architecture des réseaux de neurones (voir le chapitre 8).

Un autre problème tient au choix des paramètres des divers algorithmes d'apprentissage. En effet, chaque règle d'apprentissage utilise généralement un certain nombre de paramètres pour guider l'apprentissage. Ainsi, la règle de la rétropropagation de l'erreur est basée notamment sur le taux d'apprentissage noté α dans ce chapitre. Ce taux varie d'une tâche à l'autre, et, encore une fois, on utilise souvent des règles heuristiques simples pour déterminer sa valeur idéale. Dans la même veine que pour le choix des architectures, on utilise maintenant des méthodes comme les algorithmes génétiques pour choisir ces paramètres.

6. Les réseaux à architecture profonde

Lorsque l'on cherche à apprendre des dépendances complexes, il faut naturellement que le système apprenant puisse les représenter, donc ait accès à un espace d'hypothèses suffisamment expressif.

Dans le cas des réseaux connexionnistes, cela signifie grossièrement posséder suffisamment de neurones cachés. Il existe deux approches : soit utiliser plusieurs couches en espérant que les neurones des couches internes vont calculer les sous-dépendances utiles à la tâche, soit mettre beaucoup de neurones sur la couche cachée. Le premier cas, sans précaution, conduit généralement à l'obtention de piètres minima locaux, le second demande énormément de neurones cachés, et donc un nombre encore plus grand de connexions, donc de paramètres à régler. L'apprentissage de dépendances complexes nécessite alors le plus souvent une grande quantité d'exemples.

Pourtant ces dépendances complexes peuvent souvent se décomposer en dépendances plus simples, mettant en jeu des corrélations locales et des relations plus globales. Il existe souvent des invariances, par exemple en traitement d'images, ou en analyse de texte. De même qu'une fonction booléenne complexe peut être réalisée par des circuits mettant en jeu une combinaison bien pensée de portes logiques simples, de même on aimerait pouvoir approcher une dépendance complexe par des réseaux connexionnistes à architecture profonde dans lesquels les couches cachées seraient organisées de telle manière à décomposer le problème, en partant de descripteurs de bas niveau et en les combinant progressivement en des relations plus abstraites et de plus haut niveau.

Si le concept est séduisant, le défi est de trouver comment apprendre automatiquement, en limitant au maximum l'intervention des experts, ces réseaux connexionnistes complexes.

Les recherches sur cette question importante en sont encore à leurs débuts, et toutes les idées des jeunes pionniers sont les bienvenues. Deux approches ont été proposées.

La première reprend l'idée des *réseaux à convolution* mis au point par Yann Le Cun et ses collègues dans les années quatre-vingt-dix pour la reconnaissance des chiffres manuscrits. Une idée essentielle pour la conception de ces réseaux est d'identifier les invariances de la tâche de reconnaissance pour concevoir des détecteurs adaptés. Ainsi, s'il est important de découvrir un certain type de corrélation locale n'importe où dans une image, on peut utiliser des neurones en couche caché qui ne perçoivent qu'une sous-partie de l'image, et avoir autant de tels neurones,

Copyright © 2010 Eyrolles.

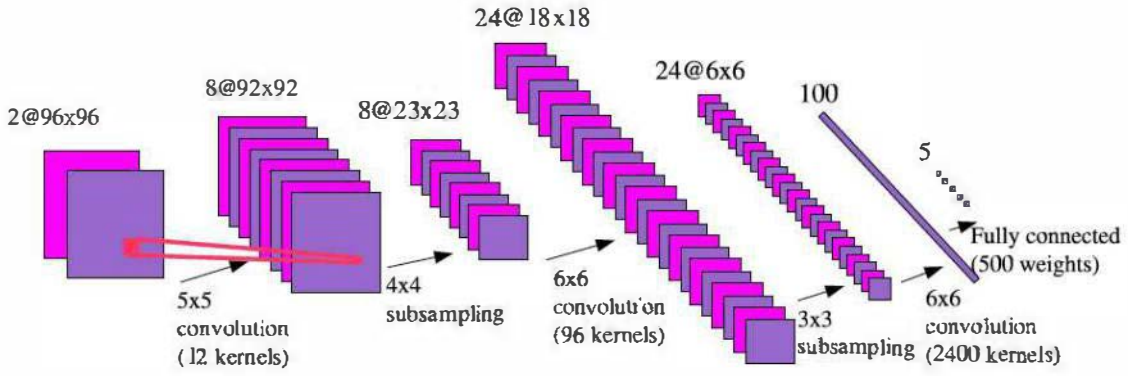


FIG. 10.15: *Architecture d'un réseau connexionniste utilisé pour traiter des images de la base NORB. L'entrée consiste en une paire d'images, dont le système extraie 8 descripteurs de taille* 92×92 *calculés sur des imagettes* 5×5. *Les sorties de ces descripteurs sont reprises par 8 descripteurs* 23×23, *24 descripteurs* 18×18, *24 descripteurs* 6×6 *et une couche de 100 neurones complètement connectés aux 5 neurones de sortie qui donnent la distance avec les vecteurs cibles. (Repris de [BL07].)*

partageant les mêmes poids et percevant chacun une sous-région de l'image. En empilant de tels « détecteurs » dans les couches cachées, et en les combinant de manière non locale, on peut ainsi concevoir un système très performant avec relativement peu de paramètres à régler (typiquement quelques milliers), et qui peut être entraîné en utilisant l'algorithme de rétro-propagation de gradient standard (voir figure 10.15).

La deuxième approche est plus générale et porte le nom de *réseaux à croyance profonds* (*deep-belief networks*). Introduite par Hinton et ses collègues, [HOT06], elle consiste à modifier l'algorithme d'apprentissage en introduisant une première étape d'apprentissage non supervisé successivement dans chaque couche, avant d'utiliser pour finir la retro-propagation de gradient sur les paramètres initialisés grâce à la première étape. L'idée principale de l'étape non supervisée est d'apparier les entrées et les sorties des couches cachées successives de telle manière que les entrées puissent être reconstruites à partir des sorties. Ce critère de reconstruction garantit ainsi que l'essentiel de l'information est préservé après le passage par les neurones des couches cachées.

L'adaptation de l'algorithme d'apprentissage est simple et les résultats obtenus sur plusieurs bases d'apprentissage surpassent ceux obtenus avec des réseaux connexionnistes classiques ou avec des méthodes à noyaux (voir chapitre 14).

Globalement, la réflexion sur les réseaux connexionnistes à architecture profonde renouvelle les questions sur l'apprentissage bien au-delà des réseaux connexionnistes. Sans préjuger des nouvelles méthodes d'apprentissage qui verront peut-être le jour, ces réflexions valent la peine d'être connues. Nous renvoyons le lecteur par exemple à [BL07] qui est d'une lecture très stimulante.

Copyright © 2010 Eyrolles.

7. Réseaux et régime dynamique : le *Reservoir Computing*

7.1 Les réseaux de neurones récurrents

Les réseaux connexionnistes présentés jusque-là sont des réseaux multicouches à passe avant (*feed-forward neural networks*), ce qui signifie que le signal se propage dans un seul sens, depuis la couche d'entrée jusqu'à la couche de sortie. De ce fait, ces réseaux sont équivalents à des fonctions, sans mémoire du passé. Par contraste, les neurones du cerveau sont connectés dans des boucles de rétroaction et il semble bien que de tels circuits récurrents soient nécessaires pour la prise en compte et l'intégration d'informations de nature temporelle.

Les réseaux connexionnistes récurrents possèdent des connexions introduisant des boucles dans les interactions entre neurones formels. En conséquence, ces réseaux peuvent entretenir des formes d'activation temporelles : il s'agit de *systèmes dynamiques* à part entière. Lorsqu'ils sont soumis à un signal d'entrée, éventuellement temporel, ils peuvent préserver dans leur état interne une mémoire dynamique de leur histoire passée. Malheureusement, il n'est pas facile de comprendre comment les connexions et leur poids doivent être modifiés pour réaliser un apprentissage dans ces réseaux. Leur apprentissage est en effet ardu car les approches par minimisation de gradient sont difficiles à mettre en œuvre avec des systèmes dynamiques, particulièrement quand les dépendances temporelles à capturer sont à long terme. De plus, ils sont notoirement difficiles à contrôler car leur dynamique présente souvent des points de bifurcation et d'instabilité ce qui rend la convergence de cette dynamique très aléatoire. Finalement, leur fonctionnement implique de nombreux cycles, ce qui peut être coûteux en calcul.

C'est pourquoi les premiers travaux sur le connexionnisme, à l'exception de quelques systèmes très particuliers, avaient soigneusement évité de s'aventurer dans les problèmes de contrôle de systèmes dynamiques. Cependant, depuis quelques années un mouvement de recherche a émergé pour explorer les possibilités ouvertes par ces réseaux.

Les réseaux récurrents peuvent se catégoriser en deux classe principales :

1. Les **réseaux opérant par minimisation (stochastique) d'une énergie**. Dans ces réseaux, inspirés de la physique statistique, on associe à l'état du réseau une *fonction d'énergie*. C'est par exemple possible lorsque tous les neurones du réseau sont connectés entre eux avec des connexions symétriques, ce qui est le cas des réseaux de Hopfield. D'autres exemples incluent les machines de Boltzman, et les réseaux profonds de croyances (*Deep Belief Networks*) devenus à la mode récemment.
   Ces réseaux sont essentiellement entraînés en *mode non supervisé*. Ils servent pour des applications de mémoire associative[10], de compression de données et dans la modélisation de distributions de données. En général, le fonctionnement du réseau implique plusieurs étapes de calcul d'activation des neurones avant de converger vers un minimum local de la fonction d'énergie correspondant à un état que l'on peut alors lire pour connaître la sortie.
2. Les **réseaux à connexions orientées et fonctionnement déterministe**. Ces réseaux, inspirés par la théorie des systèmes dynamiques, calculent en continu un signal temporel de sortie en réponse à un signal temporel d'entrée, à l'instar de filtres non linéaires, mais en prenant en compte une mémoire du passé. L'apprentissage s'opère le plus souvent en mode supervisé. Ce sont en général ces réseaux que l'on qualifie de *Reservoir Computing*.
   Ces réseaux sont très séduisants *a priori* car on peut montrer que sous des conditions faibles, ils sont des approximateurs universels des systèmes dynamiques. De plus, ils sont

[10] Dans une mémoire associative, on associe à une forme d'entrée (souvent dégradée) la forme typique (ou parfaite) associée.

Copyright © 2010 Eyrolles.

beaucoup plus proches d'une modélisation réaliste des réseaux de neurones biologiques que ne le sont les réseaux à couches.

L'approche maintenant dénommée *Reservoir Computing* a été proposée initialement à peu près simultanément par Wolfgang Mass, sous le nom de *Liquid State Machines* et par Herbert Jaeger, sous le nom de *Echo State Networks*. Elle est caractérisée par une innovation radicale. L'idée est de considérer le circuit récurrent comme une partie à part et générique du réseau, sans apprentissage, le *réservoir*, et de confier l'apprentissage à *un réseau linéaire à une couche* prenant ses entrées en certains neurones du réservoir (voir figure 10.16).

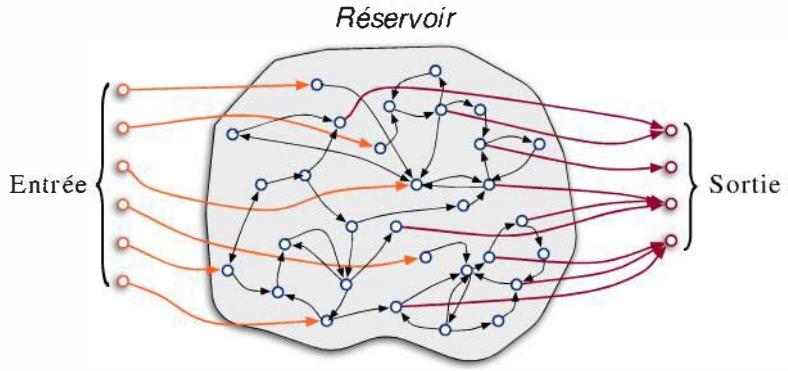


FIG. 10.16: *Un réseau connexionniste à « réservoir ». L'apprentissage ne se fait que sur la couche de sortie.*

Ainsi, le signal entrant excite le système dynamique et provoque une trajectoire dans l'espace des états possibles de ce système. La partie apprenante du réseau mesure cet état continuement ou de temps en temps et apprend à lui associer des sorties désirées. Ainsi, la source des difficultés rencontrées dans les réseaux récurrents classiques est évacuée, et l'apprentissage peut se faire avec des méthodes éprouvées.

On peut considérer ces réseaux à l'aune des SVM et autres techniques utilisant une redescription des entrées dans un nouvel espace permettant une détection plus aisée des régularités recherchées. Ici, l'espace de redescription est calculé grâce au « réservoir », et si celui-ci est à la fois assez complexe et en même temps doté de bonnes propriétés dynamiques, il peut faire apparaître des motifs caractéristiques exploitables par le système d'apprentissage en aval, de la même manière que la redescription grâce à l'emploi de fonctions noyau permet aux SVM de trouver des régularités linéaires dans l'espace de redescription.

7.2 Le traitement de données temporelles

Dans les applications classiques, non temporelles, la redescription des données se fait généralement par *expansion*, c'est-à-dire par projection dans un espace dans lequel les dimensions correspondent à des combinaisons (ou encore des moments d'ordre supérieur dans le langage des statistiques) de descripteurs originaux. De nombreuses méthodes temporelles reprennent ce principe, la différence étant que les combinaisons prennent maintenant en compte des mesures passées du signal. Les fonctions d'expansion ont ainsi une mémoire $\Phi_t(\mathbf{x}_t) = \Phi(\ldots, \mathbf{x}_{t-1}, \mathbf{x}_t)$, c'est-à-dire que la fonction de redescription prend en compte l'entrée courante $\mathbf{x}_t$ ainsi que son histoire, potentiellement infinie.

Copyright © 2010 Eyrolles.

Comme cette fonction a un nombre de paramètres *a priori* non borné, on utilise souvent en lieu et place une *formalisation récursive* : $\Phi(\mathbf{x}_t) = f\big(\Phi(\mathbf{x}_{t-1}), \mathbf{x}_t\big)$.

De la même manière que l'on peut voir les redescriptions classiques de points d'entrées $\{\mathbf{x}_i\}_{(1 \leq i \leq m)}$ comme des projections sur des sous-variétés de $\Phi(\mathcal{X})$, on peut voir ces redescriptions du signal dans le cas temporel comme des projections sur des sous-variétés correspondant à des attracteurs dynamiques.

7.3 Le *Reservoir Computing*

Une question fondamentale des méthodes à base de réservoir est donc de concevoir un réservoir tel que les expansions utiles du signal temporel y apparaissent naturellement. Il s'agit à l'heure actuelle d'un défi qui reste encore mal maîtrisé.

Dans l'approche dite *Echo State Networks*, le réservoir est constitué d'un ensemble de neurones utilisant généralement une fonction d'activation classique, sigmoïde ou $tanh(\cdot)$, et la couche de sortie mesure l'activité de certains de ces neurones pour calculer, à l'aide d'une fonction linéaire, la sortie y_t du réseau.

L'approche dite *Liquid State Machines* essaie de se rapprocher d'un modèle plus plausible du fonctionnement du cerveau par l'emploi de neurones à décharges (*spiking neurons*) ayant des propriétés temporelles propres d'intégration des signaux et d'émission de sortie. Si leur capacité expressive et de calcul sont certainement supérieures aux approches moins sophistiquées, l'apprentissage y est en revanche nettement plus compliqué.

Même si la conception des réservoirs est encore mal maîtrisée, certains principes semblent émerger. Afin d'engendrer des dynamiques suffisamment riches tout en conservant la possibilité d'apprendre en sortie, il semble qu'il faille :

1. un réservoir *suffisamment grand* ;
2. avec des connexions *clairsemées* ;
3. et tirées *aléatoirement*.

Concrètement, on utilise des réservoirs de plusieurs dizaines de milliers de neurones, avec un taux de connexion moins de 20 %, et des poids de connexions tirés uniformément dans un intervalle de valeurs centré en zéro.

Par ailleurs, on recherche également à obtenir la propriété dite *echo state property*, c'est-à-dire d'extinction progressive de l'effet d'un signal passé (au lieu d'une amplification possible). Pour cela, il faut contrôler le spectre de la matrice $\mathbf{W}$ correspondant aux connexions du réservoir (voir [LJ09], section 5.1 pour des spécifications précises). Une grande variété de topologies ont été testées, et la détermination de topologies les plus favorables reste du domaine de la recherche. Un certain nombre d'indicateurs et de mesures pour qualifier ces réservoirs ont également été proposés.

De la même manière, tout un ensemble de techniques ont été essayées pour la conception et l'apprentissage des réseaux *en aval* du réservoir.

7.4 Bilan très provisoire et perspectives

Ces nouvelles architectures de réseaux sont très attrayantes car elles permettent de dépasser les limites des approches classiques par réseaux à couche. Alors que cette approche en est encore à ses premiers pas, les résultats obtenus sur certaines tâches sont extrêmement prometteurs : par exemple dans la prédiction de séries temporelles chaotiques (trois ordres de magnitude en terme d'exactitude de prédiction), dans l'égalisation non linéaire de canaux sans fil (deux ordres

Copyright © 2010 Eyrolles.

de magnitude), la reconnaissance de voyelles japonaises (0 % d'erreur en test sur le benchmark classique au lieu de 1.8 % pour le meilleur résultat jusque-là), etc.

Cependant, nous sommes encore loin de comprendre et de maîtriser ces réseaux. Il est clair qu'utiliser un réservoir engendré de manière aléatoire n'est pas satisfaisant et qu'il vaut mieux comprendre comment adapter un type de réservoir à une famille de tâches. De même, l'apprentissage en aval du réseau doit également être mieux adapté aux situations d'apprentissage rencontrées.

Le reservoir computing a donc permis d'ouvrir de nouvelles perspectives dans l'approche connexionniste de l'apprentissage, et il y a là un domaine de recherche excitant à défricher.

Nous conseillons au lecteur débutant dans ce champ de consulter [LJ09] pour une très bonne synthèse et [PMB09] pour une synthèse sur les réseaux à décharge (*spiking neurons*).

Notes historiques et sources bibliographiques

L'ouvrage de Dreyfus et al. ([DMS+02], seconde édition [DMS+08]) est recommandé : encyclopédique, écrit en français et récent. Les livres de R. Golden [Gol96], C. Bishop [Bis95] et B. Ripley [Rip96] fournissent des panoramas sur la théorie et de la pratique de ces outils. Le chapitre 6 du livre de R. Duda, P. Hart et R. Stork [DHS01] est une remarquable introduction théorique et pratique. Les deux livres de S. Haykin [Hay99] et [Hay08] sont originaux dans leur point de vue et couvrent une grande partie de l'apprentissage par réseaux connexionnistes et au-delà. Ils sont absolument à lire. On pourra aussi consulter les ouvrages suivants, intéressants à divers titres : [Sch92], [WK91], [HKP91].

Le travail fondateur le plus souvent cité est celui de W. Mc Culloch et W. Pitt [MP43] bien que l'apprentissage n'y soit pas réellement abordé. Encore une fois, on ne peut qu'admirer la clairvoyance de A. Turing, en 1948, étudiant une organisation d'unités logiques élémentaires en un ensemble au comportement complexe [Tur92]. Les travaux sur les réseaux de neurones formels, souvent inspirés par des motivations de modélisation biologique, n'ont pas cessé depuis cette époque jusqu'à nos jours. On citera comme étapes majeures, la description du perceptron par Rosenblatt [Ros62] en 1962, la parution du livre de Minsky et Papert, *Perceptrons*, en 1969 [MP69] remarquable mais qui eut l'effet d'une douche froide sur le connexionnisme, puis en 1982, la découverte de réseaux échappant aux limites dénoncées dans [MP69], les réseaux dus au physicien Hopfield [Hop82]. Leur effet psychologique fut considérable, libérant les esprits pour chercher de nouveaux types de réseaux et de règles d'apprentissage.

C'est ainsi que se produisit en 1986 un tournant majeur, avec la découverte indépendante et quasi simultanée d'un résultat important, en l'occurence, les formules de la rétropropagation de gradient de l'erreur. Le livre de P. Werbos [Wer84] et l'article de B. Widrow [Wid94] relatent l'historique de cette découverte et des réseaux connexionnistes en général.

Une véritable explosion de publications et d'applications très diverses a suivi et continue de nos jours, couvrant aussi bien les améliorations des techniques d'apprentissage que leur généralisation à des modèles plus complexes, ou leur application à des données séquentielles ou bidimensionnelles : signaux et images. Les liens des réseaux connexionnistes avec la théorie bayésienne, les théories de l'apprentissage et l'analyse (au sens mathématique du terme) ont aussi été éclaircis. Les travaux de modélisation biologiques et situés dans le domaine des sciences cognitives sont également très nombreux. L'apparition des SVM (Séparateurs à Vastes Marges) en 1995 (voir chapitre 14) avait semblé s'accompagner du déclin des méthodes connexionnistes, mais la capa-

Copyright © 2010 Eyrolles.

cité de ces derniers à réaliser de l'apprentissage hiérarchique et en-ligne les a remis en selle, de même que l'étude de réseaux à neurones plus complexes, comme les *spiking neurons* qui laisse espérer de nouvelles possibilités.

Résumé

- Les réseaux connexionnistes sont des mécanismes de calcul permettant en particulier d'affecter une classe à un vecteur de données numériques.
- Par un processus d'apprentissage par optimisation, un réseau connexionniste peut adapter ses paramètres à un ensemble de données supervisées, dans le but d'en généraliser les caractéristiques.
- Les capacités de généralisation de ces systèmes sont bonnes. L'algorithme d'apprentissage classique, la *rétropropagation du gradient de l'erreur*, est éprouvé. Il a donné lieu à de nombreuses améliorations, en particulier en ce qui concerne sa rapidité et son arrêt avant la surgénéralisation.
- Les réseaux connexionnistes ont été étendus à des architectures permettant de réaliser l'apprentissage de règles de classification de séquences. Ils permettent aussi l'apprentissage de fonctions de régression.
- Loin d'être un domaine de recherche clos, le connexionnisme connaît un renouveau d'intérêt et est le théâtre de nouvelles idées profondes (*deep-belief networks*) et dynamiques (*reservoir computing*) qui, au moins pour certaines tâches, améliorent considérablement les performances mesurées.

Copyright © 2010 Eyrolles.

Chapitre 11

L'apprentissage de réseaux bayésiens

On présente dans ce chapitre un cas particulier de modèles permettant d'exprimer des relations probabilistes entre des ensembles de faits. Ces relations diffèrent des relations logiques en ce qu'elles n'autorisent pas un raisonnement implicatif, mais conditionnel. *Deux faits peuvent en effet être en relation causale sans que l'un implique l'autre. Dans la base de données d'une compagnie d'assurances, il est par exemple possible que, pour une majorité des entrées dans une certaine ville, les items « contravention de stationnement » soient vrais quand un autre item comme « le conducteur aime les légumes » soient corrélés. Il serait trop rapide et peu fructueux d'en conclure que le second fait implique statistiquement le premier. Une analyse en probabilité conditionnelle des faits pourrait en effet révéler que la majorité des contraventions sont collées le samedi, jour du marché. Il existe bien une « cause » commune (ou en tout cas une condition commune de forte probabilité), mais la logique n'a pas réellement à intervenir dans cette affaire. L'ensemble des faits et des probabilités conditionnelles d'un système de raisonnement de ce type peut s'organiser en graphe, sous certaines conditions d'indépendance probabiliste. On peut alors raisonner, c'est-à-dire calculer la probabilité conditionnelle de n'importe quel ensemble de faits connaissant n'importe quel autre ensemble. C'est une technique puissante pour exploiter utilement les bases de données. Cependant, un système de ce type ne peut être complètement utile que si un programme est capable d'extraire automatiquement les faits significatifs et le réseau de leurs relations conditionnelles. Une base de données comporte couramment des dizaines de milliers d'entrées ventilées sur des centaines de faits VRAI ou FAUX. Aucun expert ne peut extraire seul une structure de dépendance probabiliste d'une telle quantité de données. C'est ici qu'intervient l'apprentissage artificiel...*

Copyright © 2010 Eyrolles.

Sommaire

1 Les modèles graphiques 355
2 Les réseaux d'inférence bayésiens 357
2.1 Définitions et notations 358
2.2 Condition de Markov sur les réseaux bayésiens 359
2.3 Définition formelle d'un réseau bayésien 360
2.4 La d-séparation 361
2.5 L'équivalence de Markov 364
2.6 Couverture de Markov 364
3 Les inférences dans les réseaux bayésiens 365
3.1 Schémas d'inférence 366
4 L'apprentissage des réseaux bayésiens 369
4.1 Apprentissage des paramètres (avec *structure connue*) 369
4.1.1 ... et *données complètes* 370
4.1.2 ... et *données incomplètes* 371
4.2 Apprentissage de la *structure* avec *données complètes* 373
4.2.1 Apprentissage par recherche directe des dépendances conditionnelles 374
4.2.2 Les fonctions de score 374
4.2.3 L'apprentissage avec fonction de score 376
4.3 Apprentissage de la *structure* avec *données incomplètes* 376
4.3.1 Le problème des données incomplètes 376
4.3.2 Techniques d'apprentissage 377
5 L'inférence de relations causales 378
6 Applications 379
7 Quelques logiciels 380

Ce matin, il y a un cygne à bec jaune sur ma pelouse. Deux possibilités : soit il s'est échappé du zoo voisin, soit c'est un oiseau migrateur. Pour le savoir, je me dirige vers le poissonnier du coin et j'observe que le prix du saumon norvégien a augmenté. J'en déduis que le cygne est sauvage.

Ce genre de raisonnement possède deux caractéristiques un peu surprenantes. La première est que la décision prise n'est pas forcément la bonne : le cygne peut très bien s'être échappé du zoo et je peux aussi me tromper à propos du cours du saumon. Surtout, il n'y a pas d'implication (de relation de cause à effet) entre le fait d'avoir un cygne sur ma pelouse et celui d'avoir à payer le saumon plus cher. Cependant, constater le second change ma confiance dans l'origine du premier. Pourquoi ? S'il fait très froid dans le Nord de l'Europe, deux phénomènes en apparence indépendants ont une probabilité forte d'arriver : la migration jusqu'à nos latitudes d'une espèce qui est en général plus septentrionale et l'augmentation du cours du poisson récolté sur place, plus difficile à pêcher et à traiter.

On peut isoler quatre faits dans cet univers, associés à une probabilité d'être $VRAI$: il y a un cygne sur ma pelouse, le zoo est mal surveillé, le saumon a augmenté, l'hiver est froid dans le Nord. Les données de départ sont les suivantes : d'abord, il y a un cygne sur ma pelouse avec une très forte probabilité (je suis expert ornithologue et peu sujet à des hallucinations). Ensuite,

Copyright © 2010 Eyrolles.

le prix du saumon norvégien a augmenté, avec une bonne probabilité. D'autre part, je connais un certain nombre de relations comme :

- Les cygnes à bec jaune migrent parfois jusqu'ici quand il fait froid dans le Nord.
- Les animaux peuvent s'échappent du zoo si celui-ci a eu un problème technique.
- Le prix de saumon augmente quand les conditions de pêche sont mauvaises dans le Nord de l'Europe.

Je dois pour chaque relation donner la valeur de deux probabilités conditionnelles : celle que ce cygne sauvage soit sur ma pelouse sachant qu'il fait particulièrement froid dans le Nord et celle que ce cygne sauvage soit sur ma pelouse sachant qu'il ne fait pas particulièrement froid dans le Nord (la somme des deux ne vaut pas forcément 1). Et de même pour les deux autres relations.

Un cygne est sur ma pelouse soit parce c'est un évadé du zoo, soit parce que c'est un migrateur. Les deux causes sont, disons, *a priori* plausibles au même degré. Mais observer l'augmentation du prix du saumon m'indique qu'il est probable qu'il y ait des mauvaises conditions de pêche en Norvège (dans mon ensemble de faits, c'est le seul qui puisse intervenir). Donc, l'hypothèse que le cygne est migrateur devient plus vraisemblable que celle de la défaillance technique au zoo.

Dit autrement : s'il fait très froid dans le Nord de l'Europe, deux phénomènes en apparence indépendants ont une probabilité forte d'arriver ensemble : la migration jusqu'à nos latitudes d'une espèce qui est en général plus septentrionale et l'augmentation du cours du poisson récolté sur place.

Supposons maintenant que j'observe un certain nombre de fois les quatre faits cités ci-dessus. Puis-je *apprendre* à raisonner ainsi en me trompant le moins possible dans la conclusion ? C'est en effet le sujet de ce chapitre que de montrer comment raisonner dans un réseau de probabilités conditionnelles et surtout comment apprendre à le construire à partir d'observations conjointes des faits de base.

1. Les modèles graphiques

Aussi bien pour le raisonnement, au cœur de la décision, que pour la reconnaissance de forme et l'apprentissage, l'incertitude joue un rôle fondamental. Il est donc nécessaire de savoir exprimer l'incertitude et de pouvoir en faire l'objet d'un calcul. C'est ce que permet le calcul des probabilités et le théorème de Bayes.

Dans ce cadre, un concept de base est celui de *variable aléatoire*, que nous noterons ici par une majuscule, par exemple X. Une variable aléatoire peut prendre une valeur parmi un ensemble fini ou infini de valeurs. La probabilité d'un événement est la fraction du nombre de fois où cet événement se produit sur le nombre total d'essais, dans la limite d'un nombre d'essais infini. On peut alors se poser des questions telles que : « quelle est la probabilité que la variable X prenne la valeur x_i et que la variable Y prenne la valeur y_j ? », que l'on notera $\mathsf{P}(X = x_i, Y = y_j)$, ou bien, « quelle est la probabilité que la variable X prenne la valeur x_k quand on sait que la variable Y a la valeur y_l ? », notée $\mathsf{P}(X = x_k | Y = y_l)$.

Supposons que la variable Y puisse prendre une valeur parmi L, nous aurons :

$$\mathsf{P}(X = x_i) = \sum_{j=1}^{L} \mathsf{P}(X = x_i, Y = y_j)$$

ce que l'on appelle la *règle de somme sur les probabilités marginales.*

Copyright © 2010 Eyrolles.

De même, la probabilité conditionnelle $\mathbf{P}(X = x_k|Y = y_l)$ s'obtient par la *règle de produit des probabilités* :

$$\mathbf{P}(X = x_i, Y = y_j) = \mathbf{P}(Y = y_j|X = x_i) \cdot \mathbf{P}(X = x_i) = \mathbf{P}(X = x_i|Y = y_j) \cdot \mathbf{P}(Y = y_j)$$

De manière simplifiée, on écrira :

$$\mathbf{P}(X) = \sum_Y \mathbf{P}(X, Y) \qquad \text{Somme sur les probabilités marginales} \qquad (11.1)$$

$$\mathbf{P}(X, Y) = \mathbf{P}(Y|X)\,\mathbf{P}(X) \qquad \text{Règle de produit} \qquad (11.2)$$

Ces deux règles simples forment la base de tous les calculs utilisées dans ce chapitre. En particulier, le *théorème de Bayes de révision des probabilités* s'en déduit :

$$\mathbf{P}(Y|X) = \frac{\mathbf{P}(X|Y)\,\mathbf{P}(Y)}{\mathbf{P}(X)} \qquad (11.3)$$

Ce théorème dit que la probabilité *a posteriori* de l'événement Y (prenant un certaine valeur) lorsque l'événement X (prenant une certaine valeur) a été observé est fonction des probabilités *a priori* de X et de Y et de la probabilité conditionnelle de X sachant Y. Soit encore :

$$\text{Probabilité } \textit{a posteriori} \propto \text{Vraisemblance} \times \text{Probabilité } \textit{a priori}$$

On peut noter que si deux variables X et Y sont indépendantes ($\mathbf{P}(X, Y) = \mathbf{P}(X)\,\mathbf{P}(Y)$), on a : $\mathbf{P}(Y|X) = \mathbf{P}(Y)$ et $\mathbf{P}(X|Y) = \mathbf{P}(X)$.

Ce qui a été dit précédemment pour des probabilités d'événements discrets se généralise aux densités de probabilité que nous notons par $\mathbf{p}$.

Ces règles de base permettent, en principe, de calculer n'importe quelle probabilité à partir du moment où les probabilités *a priori* et les probabilités conditionnelles nécessaires sont connues. Cependant, cela implique de connaître la distribution jointe sur l'ensemble des variables (e.g. $\mathbf{P}(X_1, X_2, \ldots X_N)$ si il y a N variables), ce qui nécessiterait la connaissance, le stockage et des calculs prenant en compte un nombre exponentiel de nombres.

Heureusement, dans de nombreux cas, il existe des indépendances conditionnelles entre les variables, ce qui permet potentiellement de réduire considérablement les tables de probabilités conditionnelles à connaître pour effectuer les calculs. Les *modèles graphiques* (*graphical models*) permettent de tirer parti de ces indépendances et d'exprimer les relations entre variables dans une représentation à base de graphes. Le principe général est d'exprimer la distribution jointe comme un produit de facteurs, où chaque facteur dépend d'un nombre réduit de variables aléatoires.

Ces représentations présentent trois avantages :

1. Elles fournissent un moyen simple de *visualiser* des connaissances incertaines probabilistes.
2. Il est ainsi possible de *déterminer les propriétés du modèle*, en particulier les indépendances conditionnelles. Dans certains cas, lorsque des précautions particulières ont été prises, il est possible d'interpréter les relations du graphe comme des liens de causalité.
3. Les *calculs associés aux inférences et à l'apprentissage* peuvent s'exprimer à l'aide d'opérations locales sur le graphe et de propagations dans le graphe, ce qui permet de simplifier et de contrôler les algorithmes et l'ensemble des opérations. Il est possible de raisonner à partir de connaissances incomplètes.

Un graphe est composée de nœuds et d'arêtes ou arcs les reliant. Dans un modèle graphique, les nœuds représentent les variables aléatoires et les arcs expriment des relations probabilistes entre variables.

Les modèles graphiques incluent :

Copyright © 2010 Eyrolles.

- les *réseaux bayésiens*, fondés par Judea Pearl à partir de 1982 ;
- les *champs markoviens aléatoires* (*random Markov fields*) ;
- les *graphes chaînés* ;
- les *graphes de facteurs*.

Ce chapitre est essentiellement consacré aux réseaux bayésiens. On décrira, d'une part, les méthodes d'inférence, et, d'autre part, des approches permettant d'apprendre ces réseaux à partir de données.

2. Les réseaux d'inférence bayésiens

Les *réseaux d'inférence bayésiens* sont des modèles qui permettent de représenter des situations de raisonnement probabiliste à partir de connaissances incertaines. Ils sont une représentation efficace pour les calculs d'une distribution de probabilités. Plus précisément, les réseaux bayésiens conjuguent deux aspects (voir la figure 11.1) :

- Une **partie qualitative** exprimant des indépendances conditionnelles entre variables et, parfois, des liens de causalité. Cela se fait grâce à un graphe orienté acyclique[1] dont les nœuds correspondent à des variables aléatoires (dont nous supposerons qu'elles ne peuvent prendre qu'un ensemble fini de valeurs, et même souvent les seules valeurs `Vrai` ou `Faux`).
- Une **partie quantitative** constituée des tables de probabilités conditionnelles de chaque variable étant donnés ses parents dans le graphe.

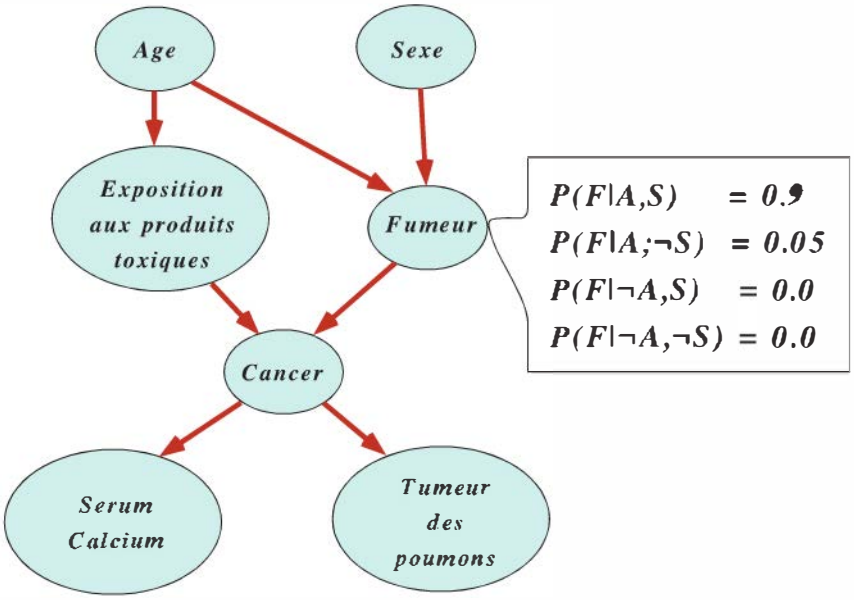


FIG. 11.1: *Les réseaux bayésiens sont une représentation compacte et efficace pour les calculs d'une distribution de probabilités grâce à l'exploitation des indépendances conditionnelles entre variables. À chaque nœud est associée une table de probabilités conditionnelles.*

[1] On dit souvent un *DAG*, de l'anglais *directed acyclic graph*.

Copyright © 2010 Eyrolles.

Prises ensemble ces deux parties définissent une distribution de probabilité unique sous forme factorisée. Ainsi, si $\mathcal{V}$ est l'ensemble des N variables X_i impliquées dans la description d'un domaine, la loi jointe $\mathsf{P}(\mathcal{V})$ se décompose en un produit de lois conditionnelles locales :

$$\mathsf{P}(\mathcal{V}) = \prod_{i=1}^{N} \mathsf{P}(X_i | parents(X_i))$$

Par exemple, dans le cas de la figure 11.1 :

$$\mathsf{P}(A,S,E,F,C,Se,T) = \mathsf{P}(A) \cdot \mathsf{P}(S) \cdot \mathsf{P}(E|A) \cdot \mathsf{P}(F|A,S) \cdot \mathsf{P}(C|E,F) \cdot \mathsf{P}(Se|C) \cdot \mathsf{P}(T|C)$$

Cette factorisation prend en compte les indépendances conditionnelles exprimées dans le graphe. Ainsi, par exemple, *Serum Calcium* et *Tumeur des poumons* sont deux variables dépendantes, mais elles deviennent indépendantes si l'on connaît la valeur de la variable *Cancer*.

Si l'on ne tenait pas compte de ces indépendances, il faudrait écrire la distribution jointe de probabilités comme :

$$\mathsf{P}(A,S,E,F,C,Se,T) = \mathsf{P}(A) \cdot \mathsf{P}(S|A) \cdot \mathsf{P}(E|A,S) \cdot \mathsf{P}(F|A,S,E) \cdot \mathsf{P}(C|A,S,E,F) \cdot \mathsf{P}(Se|A,S,E,F,C) \cdot \mathsf{P}(T|A,S,E,F,C,Se)$$

Dans le cas présent où les variables sont binaires, à valeur *Vrai* ou *Faux*, la distribution de probabilités totale nécessite donc ici la connaissance de 16 paramètres au lieu de $2^{15} = 32768$ si on ne tenait pas compte des indépendances conditionnelles codées dans le graphe. Plus grand est le nombre de variables et plus l'économie est potentiellement considérable. Nous verrons également que les inférences et l'apprentissage en seront facilités d'autant.

Chaque fait est représenté graphiquement par un nœud et les relations directes entre nœuds sont des arcs orientés entre les nœuds. L'exemple de l'introduction comporte donc quatre nœuds et trois arcs. Complètement défini, il représenterait de manière condensée toute l'information sur les dépendances entre les faits.

Nous allons d'abord définir un réseau d'inférence bayésien en y introduisant toutes les valeurs de probabilités nécessaires; nous verrons ensuite comment un tel modèle permet de raisonner. Enfin, nous donnerons quelques techniques pour apprendre ces modèles à partir d'exemples.

2.1 Définitions et notations

Un *réseau d'inférence bayésien*, ou *réseau bayésien*, est un système de raisonnement probabiliste construit sur un graphe orienté sans cycle. Nous emploierons donc dans ce chapitre le vocabulaire classique de la théorie des graphes : par exemple, un nœud F sera dit *descendant* d'un nœud A s'il existe un *chemin* (une suite d'arcs, ici orientés) entre F et A. Les termes de *parent*, *descendant direct* ou *fils*, *descendant*, *non-descendant* et *ancêtre* seront utilisés (des exemples sont donnés à la figure 11.2). Chaque nœud d'un réseau bayésien porte une étiquette qui est un des attributs du problème. Ces attributs sont binaires, pouvant prendre (avec une certaine probabilité) la valeur *Vrai* ou *Faux*, ce qui signifie qu'une variable aléatoire est associée à chaque attribut. Comme à chaque nœud est associé un attribut, donc une variable aléatoire différente, nous pouvons confondre par la suite un nœud, un attribut et la variable aléatoire associée.

Nous notons la probabilité que la variable X soit $VRAI$ par $\mathsf{P}(X = VRAI)$, ou en raccourci $\mathsf{P}(X)$. On a : $\mathsf{P}(X = FAUX) = 1 - \mathsf{P}(X = VRAI)$, ce que nous notons : $\mathsf{P}(\neg X) = 1 - \mathsf{P}(X)$.

Copyright © 2010 Eyrolles.

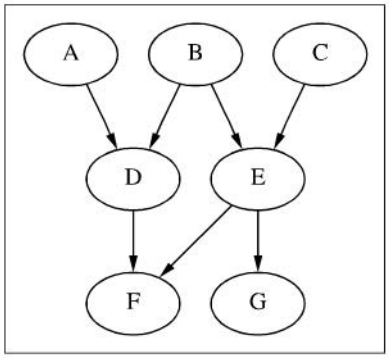


FIG. 11.2: *Un graphe orienté sans cycle. Les* parents *de F sont D et E. Les* ancêtres *de F sont D, E, A, B et C. Les* fils *de B sont D et E. Les* descendants *de B sont D, E, F et G. Les* non-descendants *de A sont B, C, E et G.*

2.2 Condition de Markov sur les réseaux bayésiens

Les indépendances conditionnelles codées par le graphe sont calculables grâce à un critère formel de théorie des graphes que l'on appelle la *d-séparation*. Ce critère graphique permet le calcul des indépendances conditionnelles en temps polynomial en fonction du nombre de variables.

Ainsi par exemple, un critère plus faible est celui de *condition de Markov* sur un réseau bayésien.

Définition 11.1 (Condition de Markov)

Dans un réseau bayésien, tout nœud est conditionnellement indépendant de ses non-descendants, connaissant ses parents.

En termes plus formels, notons $\mathcal{A}(X)$ n'importe quel ensemble de nœuds qui ne sont *pas* des descendants de X et $\mathcal{P}(X)$ l'ensemble des parents de X. Alors la propriété précédente s'écrit :

$$\mathbf{P}(X|\mathcal{A}(X), \mathcal{P}(X)) = \mathbf{P}(X|\mathcal{P}(X)) \tag{11.4}$$

Autrement dit, l'ensemble des valeurs $\mathbf{P}(X|\mathcal{P}(X))$, avec X parcourant l'ensemble des nœuds du graphe, suffit à déterminer complètement l'ensemble de toutes les probabilités conditionnelles d'un réseau bayésien.

Compte tenu de la structure particulière du graphe, on peut démontrer [Nea04] (pp.39-40) que la condition 11.4 peut se réécrire sous la forme suivante :

Théorème 11.1

Soit $\mathcal{V} = \{X_1, \ldots, X_N\}$ l'ensemble des nœuds du graphe. On a :

$$\mathbf{P}(X_1, \ldots, X_N) = \prod_{i=1}^{N} \mathbf{P}(X_i|\mathcal{P}(X_i)) \tag{11.5}$$

Ce théorème permet de réduire le problème de la détermination d'un immense nombre de valeurs de probabilités à celui d'un nombre généralement beaucoup plus restreint.

— EXEMPLE —

Copyright © 2010 Eyrolles.

Nous adaptons de [Nil98] l'exemple dont le graphe est donné à la figure 11.3.

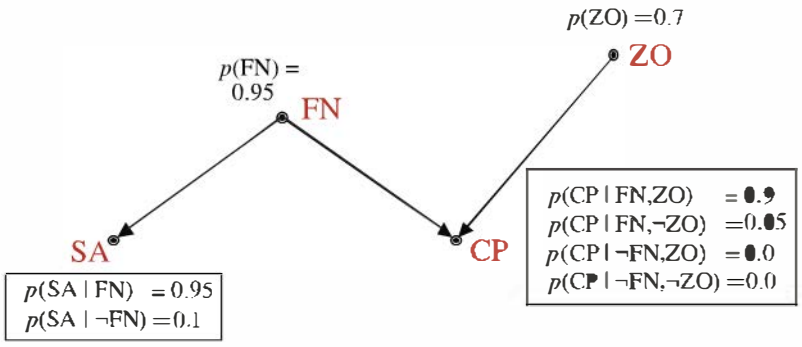


FIG. 11.3: *Le graphe du réseau bayésien qui exprime « Il y a un cygne à bec jaune sur ma pelouse ».*

Ce réseau reprend le problème de l'introduction, avec les notations suivantes :
- FN : il fait Froid en Norvège ;
- ZO : la cage des cygnes est ouverte au ZOO ;
- CP : il y a un Cygne sauvage sur ma Pelouse ;
- SA : le prix de Saumon a Augmenté ;

FN et ZO ont une influence directe sur CP puisque ce sont donc ses nœuds parents. De même, FN a une influence directe sur SA.

La spécification complète d'un réseau bayésien doit affecter à chaque variable toutes les probabilités conditionnelles significatives (celles de l'équation 11.5). Compte tenu des dépendances de notre exemple, il faut donc d'abord spécifier les valeurs $\mathbf{P}(FN)$ et $\mathbf{P}(ZO)$ qui sont non conditionnelles, puisque les nœuds correspondants sont tous les deux sans ancêtres. Notons que les variables FN et ZO sont indépendantes pour la même raison. Il nous faut, pour le nœud SA, donner $\mathbf{P}(SA \mid FN)$ et $\mathbf{P}(SA \mid \neg FN)$ et, pour le nœud CP, donner quatre valeurs : $\mathbf{P}(CP \mid FN, ZO)$,$\mathbf{P}(CP \mid FN, \neg ZO)$, $\mathbf{P}(CP \mid \neg FN, ZO)$ et $\mathbf{P}(CP \mid \neg FN, \neg ZO)$.

Notre réseau sera donc complet si nous lui ajoutons par exemple le tableau :

$\mathbf{P}(FN)$	0.95
$\mathbf{P}(ZO)$	0.7
$\mathbf{P}(SA \mid FN)$	0.95
$\mathbf{P}(SA \mid \neg FN)$	0.1
$\mathbf{P}(CP \mid FN, ZO)$	0.9
$\mathbf{P}(CP \mid FN, \neg ZO)$	0.05
$\mathbf{P}(CP \mid \neg FN, ZO)$	0
$\mathbf{P}(CP \mid \neg FN, \neg ZO)$	0

Ces valeurs sont données arbitrairement pour spécifier notre exemple. Par exemple, $\mathbf{P}(SA|FN) = 0.95$ signifie que nous posons qu'il y a 95 % de chances que le prix du saumon augmente s'il fait froid en Norvège et $\mathbf{P}(SA \mid \neg FN) = 0.1$ qu'il y a 10 % de chances qu'il augmente sans qu'il fasse particulièrement froid dans ce pays. Ces valeurs peuvent être obtenues par apprentissage, comme nous le verrons dans la section 4.

2.3 Définition formelle d'un réseau bayésien

Compte tenu de ce qui a été dit, nous pouvons maintenant donner une définition complète d'un réseau bayésien.

Copyright © 2010 Eyrolles.

Définition 11.2 (Réseau bayésien)

Un réseau bayésien *est un couple* (G, Θ)*, avec :*

- G *est un graphe orienté sans cycle.*
- Θ *est une distribution de probabilité définies sur les variables du graphe.*
- *À chaque nœud de* G *est associée une variable aléatoire et une seule.*
- *Soit* $\{X_1, \ldots, X_N\}$ *l'ensemble de ces variables aléatoires. La propriété suivante découle de la structure de graphe sans cycle :*

$$\Theta = \mathbf{P}(X_1, \ldots, X_N) = \prod_{i=1}^{N} \mathbf{P}(X_i | \mathcal{P}(X_i)) \tag{11.6}$$

avec $\mathcal{P}(X_i)$ *l'ensemble des variables associées aux parents du nœud associé à* X_i*.*

Un réseau bayésien est donc complètement spécifié quand son graphe a été décrit et quand, pour chaque nœud de ce graphe, les probabilités conditionnelles de ce nœud connaissant chacun de ces parents sont données.

Par définition, un réseau bayésien satisfait la condition de Markov 11.4.

2.4 La d-séparation

Il est possible de généraliser la condition de Markov au calcul de l'indépendance de sous-ensembles de nœuds dans le graphe, ce qui permettra d'accélérer notoirement les calculs d'inférence. L'idée est d'identifier les indépendances conditionnelles entre sous-ensembles de variables $\mathcal{V}_1$ et $\mathcal{V}_2$, étant donné un autre sous-ensemble $\mathcal{V}_3$ uniquement sur des critères graphiques dans G, de telle manière que ces indépendances correspondent bien à *toutes* les indépendances conditionnelles existantes dans Θ.

Soit un ensemble de nœuds $\{X_1, X_2, \ldots, X_k\}$ dans G, où $k \geq 2$ et tel que (X_{i-1}, X_i) ou (X_i, X_{i-1}) soit un arc de G. Alors l'ensemble des arcs (non orientés) connectant les k nœuds est appelée *chaîne* entre X_1 et X_k. Afin de préciser l'orientation des arcs entre variables, on utilise la notation $X_{i-1} \rightarrow X_i$ ou $X_{i-1} \leftarrow X_i$.

On a alors les propriétés élémentaires suivantes :

- Dans une chaîne $X \rightarrow Z \rightarrow Y$ (séquence simple), X et Y sont indépendantes conditionnellement à Z, car la connaissance de la valeur de Z rend inutile la connaissance de X (ou de Y) pour connaître la valeur de Y (ou de X).
- Dans une chaîne $X \rightarrow Z \leftarrow Y$ (appelée une V-structure), X et Y sont indépendantes, sauf si Z est connue.
- Dans une chaîne $X \leftarrow Z \rightarrow Y$ (Λ-structure), X et Y sont dépendantes sauf si Z est connue, car le fait de connaître la valeur de Z rend inutile la connaissance de X pour connaître celle de Y, et vice versa.

On peut alors énoncer une propriété sur les chaînes en fonction d'un sous-ensemble de nœuds.

Copyright © 2010 Eyrolles.

Définition 11.3 (Chaîne bloquée)

Soient un graphe orienté acyclique $G = \{\mathcal{N}, \mathcal{A}\}$ constitué des nœuds $\mathcal{N}$ et des arcs $\mathcal{A}$, un sous-ensemble de nœuds $\mathcal{Z} \in \mathcal{N}$, X et Y deux nœuds dans $\mathcal{N} \setminus \mathcal{Z}$, et ρ une chaîne entre X et Y. Alors la chaîne ρ est **bloquée** *par $\mathcal{Z}$ si l'une des conditions suivantes est vérifiée :*

1. *Il y a un nœud Z dans $\mathcal{Z}$ sur la chaîne ρ, et les arcs incidents à Z sur ρ se rencontrent selon une séquence simple $(\ldots \rightarrow Z \rightarrow \ldots)$.*
2. *Il y a un nœud Z dans $\mathcal{Z}$ sur la chaîne ρ, et les arcs incidents à Z sur ρ se rencontrent selon une Λ-structure $(\ldots \leftarrow Z \rightarrow \ldots)$.*
3. *Il y a un nœud Z dans $\mathcal{Z}$ sur la chaîne ρ, et les arcs incidents à Z sur ρ se rencontrent selon une V-structure $(\ldots \rightarrow Z \leftarrow \ldots)$.*

Il peut y avoir davantage qu'un seul nœud dans $\mathcal{Z}$ vérifiant ces propriétés. On dit qu'une chaîne est **active** *si elle n'est pas bloquée par $\mathcal{Z}$.*

EXEMPLE **(Blocage de chemin)**

Soit le graphe acyclique orienté de la figure 11.4. Nous avons les propriétés suivantes :

1. La chaîne $[Y, X, Z, S]$ est bloquée par $\{X\}$ car les arcs incidents à X sur la chaîne se rencontrent selon une Λ-structure en X. Cette chaîne est également bloquée par $\{Z\}$ car les arcs incidents à Z se rencontrent selon une séquence simple.
2. La chaîne $[W, Y, R, Z, S]$ est bloquée par $\emptyset$ car $R \notin \emptyset$, $T \notin \emptyset$ et les arcs incidents à R sur la chaîne se rencontrent selon une V-structure.
3. La chaîne $[W, Y, R, S]$ est bloquée par $\{R\}$ car les arcs incidents à R sur le chemin se rencontrent selon une séquence simple.
4. La chaîne $[W, Y, R, Z, S]$ est *non* bloquée par $\{R\}$ car les arcs incidents en R sur le chemin se rencontrent selon une V-structure. De plus la chaîne est *non* bloquée par $\{T\}$ car T est un descendant de R.

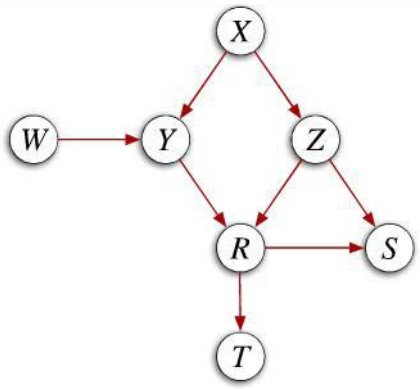


FIG. 11.4: *Un graphe acyclique orienté pour illustrer le blocage de chemin.*

(Tiré de [Nea04] pp. 75-76.)

Définition 11.4 (d-séparation de variables)

Soient un graphe orienté acyclique $G = \{\mathcal{N}, \mathcal{A}\}$ constitué des nœuds $\mathcal{N}$ et des arcs $\mathcal{A}$, un sous-ensemble de nœuds $\mathcal{Z} \in \mathcal{N}$, X et Y deux nœuds dans $\mathcal{N} \setminus \mathcal{Z}$. Les nœuds X et Y sont **d-séparés** *par $\mathcal{Z}$ dans G si toutes les chaînes entre X et Y sont bloquées par $\mathcal{Z}$.*

Copyright © 2010 Eyrolles.

— Exemple (d-séparation) —

Soit le graphe acyclique orienté de la figure 11.5. Nous avons les propriétés suivantes :

1. X et R sont d-séparés par $\{Y, Z\}$ car la chaîne $[X, Y, R]$ est bloquée en Y, et la chaîne $[X, Z, R]$ est bloquée en Z.
2. X et T sont d-séparées par $\{Y, Z\}$ car la chaîne $[X, Y, R, T]$ est bloquée en Y, la chaîne $[X, Z, R, T]$ est bloquée en Z, et la chaîne $[X, Z, S, R, T]$ est bloquée en Z et en S.
3. W et T sont d-séparées par $\{R\}$ car les chaînes $[W, Y, R, T]$ et $[W, Y, X, Z, R, T]$ sont bloquées toutes les deux en R.
4. Y et Z sont d-séparés par $\{X\}$ car la chaîne $[Y, X, Z]$ est bloquée en X, la chaîne $[Y, R, Z]$ est bloquée en R, et la chaîne $[Y, R, S, Z$ est bloquée en S.
5. W et X sont d-séparées par $\emptyset$ car la chaîne $[W, Y, X]$ est bloquée en Y, la chaîne $[W, Y, R, Z, X]$ est bloquée en R, et la chaîne $[W, Y, R, S, Z, X]$ est bloquée en S.
6. W et X *ne sont pas* d-séparées par $\{Y\}$ car la chaîne $[W, Y, X]$ n'est pas bloquée en Y puisque $Y \in \{Y\}$ et qu'elle ne peut pas être bloquée ailleurs.
7. W et T *ne sont pas* d-séparés par $\{Y\}$ car même si la chaîne $[W, Y, R, T]$ est bloquée en Y, la chaîne $[W, Y, X, Z, R, T]$ n'est pas bloquée en Y puisque $Y \in \{Y\}$ et cette chaîne n'est bloquée nulle part ailleurs car aucun autre nœud n'est dans $\{Y\}$ et qu'il n'y a pas d'autres V-structure sur elle.

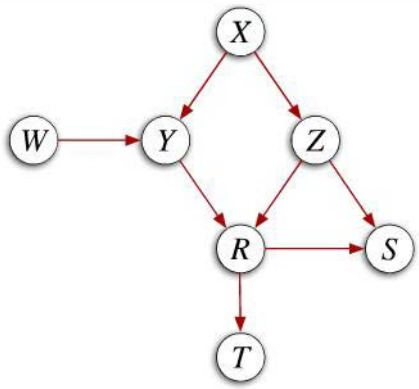


Fig. 11.5: *Un graphe acyclique orienté pour illustrer le blocage de chemin.*

(Tiré de [Nea04] pp. 76-77.)

Définition 11.5 (d-séparation de sous-ensembles de variables)

*Soient un graphe orienté acyclique $G = \{\mathcal{N}, \mathcal{A}\}$ constitué des nœuds $\mathcal{N}$ et des arcs $\mathcal{A}$, un sous-ensemble de nœuds $\mathcal{Z} \in \mathcal{N}$, $\mathcal{X}$ et $\mathcal{Y}$ deux sous-ensembles de nœuds dans $\mathcal{N} \setminus \mathcal{Z}$. $\mathcal{X}$ et $\mathcal{Y}$ sont **d-séparés** par $\mathcal{Z}$ dans G si toutes les chaînes entre nœuds de $\mathcal{X}$ et nœuds de $\mathcal{Y}$ sont bloquées par $\mathcal{Z}$.*

Il est prouvé que le concept de d-séparation défini sur les graphes correspond bien au concept d'indépendance conditionnelle entre les variables d'une distribution de probabilité jointe (voir la preuve dans [Nea04] pp.77-80).

Il existe des algorithmes efficaces pour déterminer les d-séparations.

Copyright © 2010 Eyrolles.

2.5 L'équivalence de Markov

De nombreux graphes acycliques orientés sont équivalents au sens où ils ont les mêmes d-séparations. On dit qu'ils sont équivalents au sens de Markov.

Définition 11.6 (Équivalence de Markov)

Soient deux graphes $G_1 = \{\mathcal{N}, \mathcal{A}_1\}$ et $G_2 = \{\mathcal{N}, \mathcal{A}_2\}$ ayant le même ensemble de nœuds $\mathcal{N}$. Les graphes G_1 et G_2 sont dits **équivalents au sens de Markov** *si pour tout triplet de sous-ensembles de nœuds disjoints $\mathcal{X}$, $\mathcal{Y}$ et $\mathcal{Z}$, $\mathcal{X}$ et $\mathcal{Y}$ sont d-séparés par $\mathcal{Z}$ dans G_1 si et seulement si $\mathcal{X}$ et $\mathcal{Y}$ sont d-séparés par $\mathcal{Z}$ dans G_2.*

EXEMPLE (Équivalence de Markov)

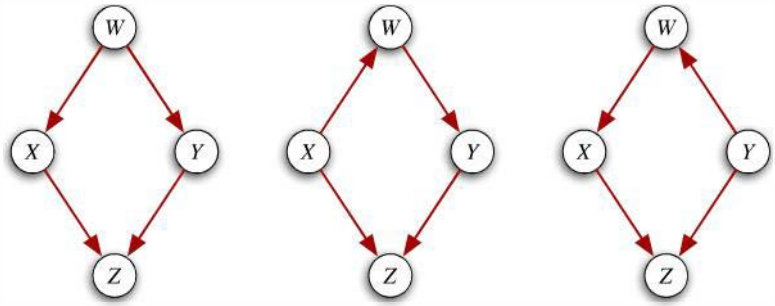


FIG. 11.6: *Trois graphes acycliques orientés équivalents au sens de Markov. Aucun autre graphe acyclique orienté n'est équivalent à eux.*

Théorème 11.2 (Équivalence de Markov)

Deux graphes acycliques orientés $\mathcal{G}_1$ et $\mathcal{G}_2$ sont équivalents au sens de Markov, si et seulement si ils partagent les mêmes liens (sans distinction d'orientation) et le même ensemble de V-structures.

Ce théorème, démontré pour la première fois par [PGV89], permet de développer des algorithmes polynomiaux pour déterminer si deux graphes sont équivalents au sens de Markov.

2.6 Couverture de Markov

Un réseau bayésien peut comporter de nombreux nœuds, et les probabilités conditionnelles d'un nœud peuvent être influencées par la valeur d'un nœud distant dans le graphe. Cependant, généralement, l'instanciation d'un ensemble de nœuds proches peut isoler un nœud de l'effet de tous les autres nœuds.

Définition 11.7 (Couverture de Markov)

Soient $\mathcal{N}$ un ensemble de variables aléatoires, Θ leur distribution de probabilité jointe, et X une variable de $\mathcal{N}$. Alors **une couverture de Markov** *M de X est tout sous-ensemble de variables de $\mathcal{N}$ tel que X soit conditionnellement indépendant de toutes les autres variables étant donné M.*

Copyright © 2010 Eyrolles.

Un théorème permet d'identifier une telle couverture facilement.

Théorème 11.3 (Couverture de Markov)

Soit un réseau $(G, \bullet)$ satisfaisant la condition de Markov. Alors pour chaque variable X, l'ensemble de tous les parents de X, enfants de X et parents des enfants de X constitue une couverture de Markov de X.

La notion de couverture de Markov n'est pas seulement utile dans le cadre des réseaux bayésiens, elle est également utilisée comme base de méthodes pour la sélection d'attributs.

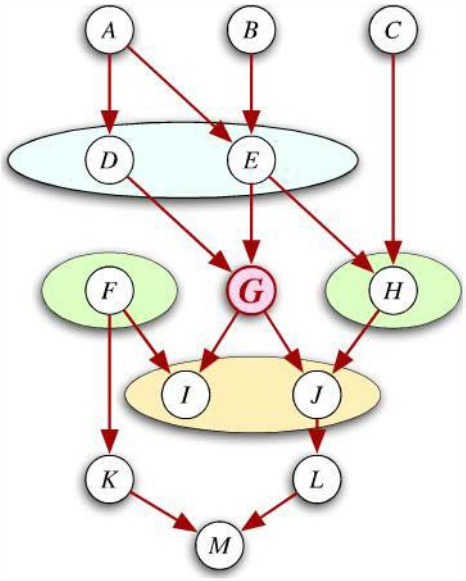


FIG. 11.7: *La couverture de Markov du nœud G est constituée de ses parents $\{D, E\}$, de ses fils $\{I, J\}$, et des parents de ses fils $\{F, H\}$.*

3. Les inférences dans les réseaux bayésiens

Un réseau bayésien est donc un graphe de corrélation entre variables auquel est associée une représentation probabiliste sous-jacente. La correspondance qui existe entre la structure graphique et la structure probabiliste correspondante permet de ramener l'ensemble des problèmes d'inférence à des problèmes de théorie des graphes, qui restent cependant assez complexes.

Ainsi, une fois qu'un réseau bayésien a été construit pour rendre compte d'un domaine, on cherche souvent à l'utiliser pour déterminer des probabilités correspondant à certains événements, certaines questions, certaines dépendances. En général, ces probabilités ne sont pas stockées dans le réseau et il faut donc les calculer. Comme un réseau bayésien code la distribution de probabilité jointe pour l'ensemble des variables du domaine, il permet en principe de calculer n'importe quelle probabilité d'intérêt. Les règles du calcul probabiliste sont utilisées pour cela. Dans la pratique on rencontre cependant deux difficultés : d'une part, les calculs ne sont vraiment possibles que pour des variables à valeur discrète, et d'autre part, il faut savoir tenir compte d'indépendances

Copyright © 2010 Eyrolles.

conditionnelles dans le réseau pour maîtriser la complexité des calculs. Plusieurs méthodes ont été mises au point pour cela.

3.1 Schémas d'inférence

Nous cherchons maintenant à calculer d'autres probabilités conditionnelles pour exprimer l'influence des variables les unes sur les autres. Par exemple :

- *Quelle est la probabilité pour qu'il y ait un cygne sur ma pelouse sachant que le zoo a laissé une cage ouverte ?*
- *Quelle est la probabilité que le zoo soit en état normal sachant qu'il ne fait pas particulièrement froid en Norvège ?*
- *Quelle est la probabilité que le zoo soit en état normal sachant que le saumon n'a pas augmenté et qu'il ne fait pas particulièrement froid en Norvège ?*

En termes formels, il s'agit de calculer les probabilités conditionnelles $\mathbf{P}(CP|ZO)$, $\mathbf{P}(\neg ZO|\neg CP)$ et $\mathbf{P}(\neg ZO \mid \neg FN, \neg CP)$. Le **premier cas** s'appelle une *inférence causale* ou *descendante* : en effet, le nœud ZO est un ancêtre du nœud CP. On peut donc considérer ZO comme une *cause*[2] de CP. Mais cela ne veut pas dire qu'il n'y ait pas de dépendance inverse. Le **deuxième calcul** s'appelle une *inférence ascendante* ou un *diagnostic* : il s'agit de comprendre une variable par ses conséquences. Le **troisième cas** est une combinaison des deux premiers, appelée une *explication*.

D'une manière générale, le calcul de ces probabilités conditionnelles se base sur la **formule de Bayes** :

$$\mathbf{P}(X, Y) = \mathbf{P}(X|Y)\mathbf{P}(Y) = \mathbf{P}(Y|X)\mathbf{P}(X)$$

et sur la règle du *chaînage* des probabilités conditionnelles :

$$\begin{aligned}\mathbf{P}(X_1, \ldots, X_N) &= \mathbf{P}(X_N|X_{N-1}, \ldots, X_1)\mathbf{P}(X_{N-1} \mid X_{N-2}, \ldots, X_1) \ldots \mathbf{P}(X_2|X_1)\mathbf{P}(X_1) \\ &= \prod_{i=1}^{N} \mathbf{P}(X_i|X_{i-1}, \ldots, X_1)\end{aligned}$$

Par exemple, pour quatre variables :

$$\mathbf{P}(X_1, X_2, X_3, X_4) = \mathbf{P}(X_4|X_3, X_2, X_1)\mathbf{P}(X_3|X_2, X_1)\mathbf{P}(X_2|X_1)\mathbf{P}(X_1)$$

—— EXEMPLE **Calcul d'inférence causale** ——

Pour calculer $\mathbf{P}(CP|ZO)$, il faut faire intervenir l'autre parent de CP :

$$\mathbf{P}(CP|ZO) = \mathbf{P}(CP, FN|ZO) + \mathbf{P}(CP, \neg FN|ZO)$$

puis « conditionner » ce parent par rapport à ZO :

$$\mathbf{P}(CP|ZO) = \mathbf{P}(CP|FN, ZO)\mathbf{P}(FN|ZO) + \mathbf{P}(CP|\neg FN, ZO)\mathbf{P}(\neg FN|ZO)$$

Nous savons, puisque FN n'a pas de parent, que $\mathbf{P}(FN|ZO) = \mathbf{P}(FN)$ et que de même $\mathbf{P}(\neg FN|ZO) = \mathbf{P}(\neg FN)$. D'où :

$$\begin{aligned}\mathbf{P}(CP|ZO) &= \mathbf{P}(CP|FN, ZO)\mathbf{P}(FN) + \mathbf{P}(CP|\neg FN, ZO)\mathbf{P}(\neg FN) \\ &= 0.9 \times 0.95 + 0. \times (1 - 0.95) = 0.855\end{aligned}$$

La généralisation du calcul menée sur cet exemple est facile à imaginer. On la trouvera développée formellement dans [Nil98] et [BN99].

[2] Le terme *cause* est à considérer avec prudence : il ne signifie pas ici implication logique.

Copyright © 2010 Eyrolles.

Exemple Calcul de diagnostic

Ici, il faut renverser l'ordre des variables en utilisant la règle de Bayes :

$$\mathbf{P}(\neg ZO|\neg M) = \frac{\mathbf{P}(\neg M|\neg ZO)\mathbf{P}(\neg ZO)}{\mathbf{P}(\neg M)}$$

Nous pouvons maintenant calculer $\mathbf{P}(\neg M, \neg ZO)$ par inférence causale :

$$\begin{aligned}\mathbf{P}(\neg CP, \neg ZO) &= \mathbf{P}(\neg CP, FN|\neg ZO) + P(\neg CP, \neg FN|\neg ZO)\\ &= \mathbf{P}(\neg CP|FN, \neg ZO)\mathbf{P}(FN, \neg ZO)\\ &\quad +\mathbf{P}(\neg CP|\neg FN, \neg ZO)\mathbf{P}(\neg FN, \neg ZO)\\ &= \mathbf{P}(\neg CP|FN, \neg ZO)\mathbf{P}(FN) + \mathbf{P}(\neg CP|\neg FN, \neg ZO)\mathbf{P}(\neg FN)\\ &= (1 - 0.05) \times 0.95 + (1.) \times 0.05 \;=\; 0.9525\end{aligned}$$

D'où :

$$\mathbf{P}(\neg ZO, \neg CP) = \frac{0.9525 \times P(\neg ZO)}{P(\neg CP)} = \frac{0.9525 \times 0.3}{P(\neg CP)} = \frac{0.28575}{P(\neg CP)} \qquad (11.7)$$

Nous ne connaissons pas $P(\neg CP)$, mais nous contournons la difficulté en le traitant comme un facteur de normalisation en calculant, sans donner encore une fois le détail :

$$\mathbf{P}(ZO|\neg CP) = \frac{\mathbf{P}(\neg CP \mid ZO)\mathbf{P}(ZO)}{\mathbf{P}(\neg CP)} = \frac{0.0595 \times 0.7}{\mathbf{P}(\neg CP)} = \frac{0.03665}{\mathbf{P}(\neg CP)} \qquad (11.8)$$

Comme $\mathbf{P}(\neg ZO|\neg M) + \mathbf{P}(ZO|\neg CP) = 1$, on déduit des équations 11.7 et 11.8 :

$$\mathbf{P}(\neg ZO|\neg M) = 0.88632$$

La généralisation de ce calcul à tout réseau bayésien est également facile.

Exemple Calcul d'explication

Ici, le calcul se fait en employant à la fois la règle de Bayes et la règle de chaînage des probabilités conditionnelles.

$$\begin{aligned}\mathbf{P}(\neg ZO|\neg FN, \neg CP) &= \frac{\mathbf{P}(\neg CP, \neg FN, |\neg ZO)\mathbf{P}(ZO)}{\mathbf{P}(\neg FN, \neg CP)} && \text{Règle de Bayes}\\ &= \frac{\mathbf{P}(\neg CP|\neg FN, \neg ZO)\mathbf{P}(\neg FN|\neg ZO)\mathbf{P}(\neg ZO)}{\mathbf{P}(\neg FN, \neg CP)} && \text{Définition des probabilités conditionnelles}\\ &= \frac{\mathbf{P}(\neg CP|\neg FN, \neg ZO)\mathbf{P}(\neg FN)\mathbf{P}(\neg ZO)}{\mathbf{P}(\neg FN, \neg CP)} && FN \text{ et } ZO \text{ indépendants}\end{aligned}$$

Tous les termes de cette expression sont définis par le réseau, sauf $\mathbf{P}(\neg FN, \neg CP)$, que l'on peut calculer par diagnostic. On trouve finalement $\mathbf{P}(\neg ZO|\neg FN, \neg CP) = 0.03$.

On peut remarquer que cette valeur est inférieure à $\mathbf{P}(\neg ZO, \neg CP)$, ce qui signifie que savoir en plus que le prix du saumon n'a pas augmenté réduit la probabilité que le zoo soit en état normal sachant qu'il ne fait pas spécialement froid en Norvège.

La généralisation de ce calcul à tout réseau bayésien est encore possible, mais l'organisation des calculs demande évidemment un algorithme plus complexe.

Copyright © 2010 Eyrolles.

Complexité des calculs

Comme on l'a entrevu ci-dessus, il est donc possible en organisant correctement les calculs de dériver (à partir de la structure et des probabilités données au départ) toutes les probabilités conditionnelles du type $P(\mathcal{V}|\mathcal{W})$, où $\mathcal{V}$ et $\mathcal{W}$ sont des ensembles de nœuds.

Malheureusement, il a été démontré que ce calcul est NP-complet, c'est-à-dire qu'il n'existe vraisemblablement pas, si $\mathcal{V}$ et $\mathcal{W}$ sont quelconques, d'algorithme dont le temps de calcul soit polynomial en fonction du nombre total de nœuds du réseau. Il faut donc chercher à réduire ce temps de calcul en tenant compte le mieux possible de la structure, ce qui peut se faire de deux manières. La première technique consiste à trouver une relation entre la géométrie du graphe du réseau et l'*indépendance conditionnelle* de sous-ensembles de ses nœuds en étudiant la *d-séparation* des nœuds du graphe. La seconde consiste à contraindre la structure pour que les calculs se développent facilement (par exemple dans les graphes particuliers que sont les *polyarbres*).

Encore n'avons-nous parlé là que de réseaux sans boucles. Dans le cas de réseaux à boucles, c'est-à-dire dans lesquels il peut exister plusieurs chemins entre deux nœuds, il faut avoir recours à d'autres techniques, principalement :

- Les **méthodes de conditionnement** dans lesquelles on cherche à étendre les propriétés d'indépendance conditionnelles dans le graphe en cherchant des sous-ensembles de variables séparant d'autres sous-ensembles de variables.
- Les **méthodes de regroupement** (*méthode des arbres de jonction*) . Elles consistent à se ramener à un réseau sans boucle en créant des nœuds plus complexes qui représentent plusieurs nœuds du graphe original. Nous ne parlerons pas davantage de ces méthodes ici (voir sur ce sujet par exemple [BN99]).
- Les **méthodes d'approximation utilisant des méthodes de Monte-Carlo** pour estimer les probabilités en chaque nœud connaissant la probabilité de certains nœuds.

La d-séparation exploitée au niveau local des nœuds d'un graphe dans la section 2 (voir équation 11.6) peut être généralisée au calcul de l'indépendance de sous-ensembles de nœuds dans le graphe.

Il existe plusieurs familles d'algorithmes exploitant les indépendances conditionnelles codées par les réseaux bayésiens.

Des **algorithmes exacts**, dont :

- *Message passing* de Pearl (1988) pour les réseaux ayant une structure d'arbre.
- *Arbres de jonction* (*Junction tree*) de Jensen (1990).
- Algorithme de Shafer-Shenoy (1990).

Ces algorithmes sont malheureusement sujet à une explosion combinatoire pour les graphes fortement connectés. Des **méthodes approchées** ont donc été proposées s'appuyant sur :

- des techniques d'échantillonnage ;
- des méthodes variationnelles.

La figure 11.8 illustre le type de calcul, organisé comme propagation de probabilités, possible avec ces méthodes.

L'objet de cet ouvrage n'étant pas de décrire les méthodes d'inférence, nous renvoyons les lecteurs intéressés à l'abondante littérature sur ces techniques, dont l'ouvrage en français [NWL+07].

Copyright © 2010 Eyrolles.

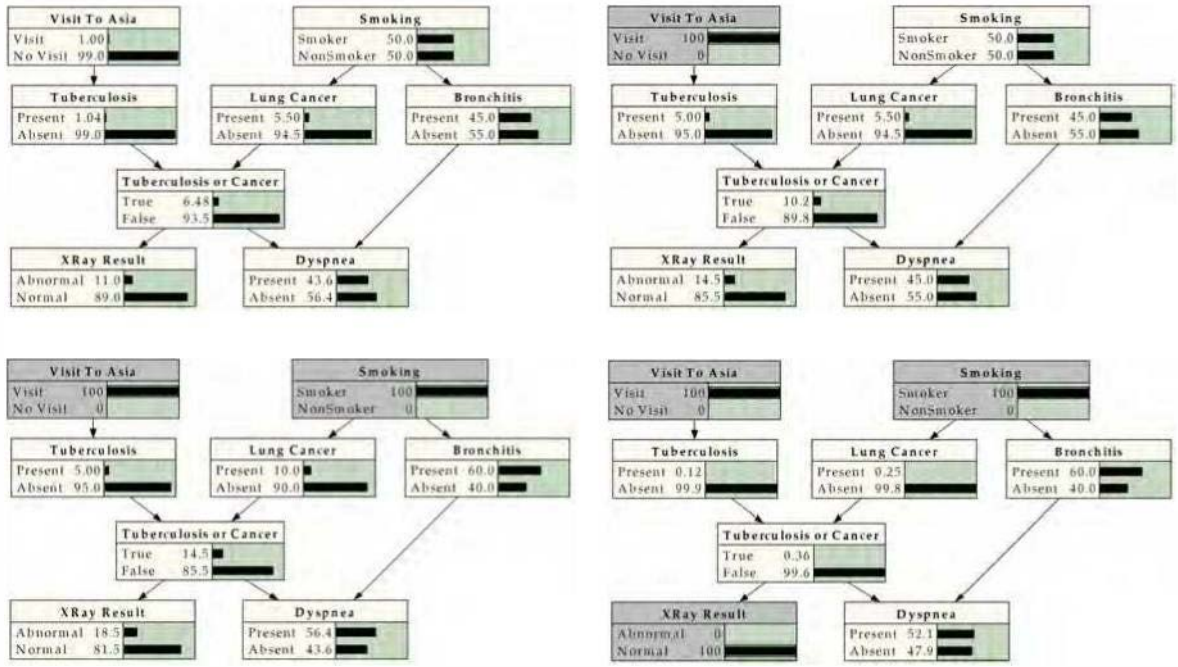


FIG. 11.8: *Exemples d'inférence dans un petit réseau bayésien. Les barres noires indiquent la probabilité de la valeur associée de la variable correspondant à un nœud. Dans chacune des figures, les nœuds grisés correspondent à des variables mesurées, appelées evidence en anglais, que nous appellerons « symptômes ». (Figures tirées d'un tutoriel de Philippe Leray à la conférence plate-forme AFIA-2009.)*

4. L'apprentissage des réseaux bayésiens

Jusqu'au début des années quatre-vingt-dix, les réseaux bayésiens étaient construits à la main, par un expert du domaine. Il s'agissait cependant d'un travail laborieux et délicat. On a donc depuis cherché des techniques d'apprentissage permettant d'apprendre ou d'affiner les paramètres d'un réseau bayésien dont la structure est donnée, ou bien même à apprendre sa structure. L'apprentissage consiste à trouver un réseau bayésien modélisant les données connues en s'appuyant éventuellement sur des connaissances préalables (voir la figure 11.9). Quatre grandes familles de problèmes sont rencontrées qui correspondent à des classes de méthodes d'apprentissage spécifiques. Nous examinons tour à tour ces différents types de problèmes.

4.1 Apprentissage des paramètres (avec *structure connue*)

Dans le cas où la structure du réseau est connue (grâce à un expert par exemple), le problème est d'*estimer les paramètres de ce réseau*, c'est-à-dire les tables de probabilités conditionnelles en chaque nœud. Il s'agit de trouver le réseau le « plus proche » de la loi de probabilité ayant engendré les données. Cette tâche est similaire à celle consistant à estimer un paramètre Θ permettant de modéliser les données $\mathcal{S} = (x_1, x_2, \ldots, x_m)$ par une distribution $P(\mathcal{S}|\Theta)$.

Copyright © 2010 Eyrolles.



FIG. 11.9: *Un réseau bayésien comprend à la fois une structure et des paramètres associés aux nœuds de cette structure (les probabilités conditionnelles). L'apprentissage consiste à estimer les paramètres et parfois aussi la structure à partir de données et éventuellement de connaissances préalables.*

	Structure connue (estimation de paramètres)	*Structure inconnue*
Données complètes	**Estimation statistique** paramétrique (*maximum de vraisemblance* ou *maximum a posteriori*)	**Optimisation discrète sur les structures** (algorithmes de recherche discrète)
Données incomplètes	**Optimisation paramétrique** (EM, descente de gradient...)	**Combinaison de méthodes** (EM structurelle, mélange de modèles...)

4.1.1 ... et *données complètes*

En général, le choix du paramètre dépend du choix de la famille de distributions de probabilités : multinomiale, Gaussienne, de Poisson, etc. Mais le principe est toujours celui de chercher la valeur du paramètre Θ maximisant la **fonction de vraisemblance**[3] :

$$\mathcal{L}(\Theta : \mathcal{S}) = \mathsf{P}(\mathcal{S}|\Theta) = \prod_{i=1}^{m} \mathsf{P}(\boldsymbol{z}_i|\bullet)$$

Dans le cas d'un réseau bayésien, la vraisemblance s'écrit en prenant en compte les indépendances conditionnelles des nœuds n_j en fonction de leurs parents Pa_j. On aura ainsi pour m données $\mathbf{x}_i$, chacune étant décrite sur d attributs (soit ici d nœuds du réseau) :

$$\mathcal{L}(\Theta : \mathcal{S}) = \prod_{i=1}^{m} \mathsf{P}(\mathbf{x}_i(1), \ldots, \mathbf{x}_i(d) : \Theta) = \prod_{i=1}^{m} \prod_{j=1}^{d} \mathsf{P}(\mathbf{x}_i(j)|Pa_j : \Theta_j)$$

$$= \prod_{j=1}^{d} \prod_{i=1}^{m} \mathsf{P}(\mathbf{x}_i(j)|Pa_j : \Theta_j) = \prod_{j=1}^{d} \mathcal{L}_j(\Theta_j : \mathcal{S})$$

[3] Des détails sur le principe du maximum de vraisemblance figurent dans les chapitres 2 et 15.

Copyright © 2010 Eyrolles.

où $x_i(j)$ représente la $j$ème coordonnées de la donnée x_i.

La vraisemblance se décompose selon la structure du réseau. De ce fait, on a affaire maintenant à plusieurs problèmes indépendants d'estimation de vraisemblance : le problème est factorisé. Si les paramètres de chaque famille ne sont pas reliés, ils peuvent être estimés indépendamment les uns des autres.

Nous avons examiné ci-dessus le problème comme celui de l'estimation de la valeur la plus probable du paramètre Θ : il s'agit là de l'approche du *maximum de vraisemblance*. Nous pourrions aussi envisager ce problème comme celui de l'estimation de densité θ et non plus de sa valeur la plus probable. C'est l'approche de l'*estimation bayésienne*. Nous n'en donnons pas le détail dans la cas présent, mais le principe général est décrit dans le chapitre 21.

Dans tous les cas, il faut avoir recours à des fréquences calculées pour estimer les probabilités conditionnelles. Nous illustrons cela ci-dessous.

— EXEMPLE **Estimation de probabilités conditionnelles par fréquence** —

On suppose ici posséder la structure du réseau et chercher à estimer les probabilités conditionnelles nécessaires. Pour réaliser cette estimation, on dispose d'un certain nombre d'observations sur les variables, c'est-à-dire d'exemples de son comportement.

Reprenons l'exemple de la figure 11.3, en supposant disposer de la table suivante des observations :

SA	CP	FN	ZO	Nombre d'exemples
$VRAI$	$VRAI$	$VRAI$	$VRAI$	54
$VRAI$	$VRAI$	$VRAI$	$FAUX$	1
$VRAI$	$FAUX$	$VRAI$	$VRAI$	7
$VRAI$	$FAUX$	$VRAI$	$FAUX$	27
$FAUX$	$VRAI$	$VRAI$	$VRAI$	3
$FAUX$	$FAUX$	$VRAI$	$FAUX$	2
$FAUX$	$VRAI$	$FAUX$	$VRAI$	4
$FAUX$	$FAUX$	$FAUX$	$FAUX$	2
				100

Prenons le nœud CP du réseau, dont les parents sont FN et ZO. Nous cherchons à estimer les huit valeurs $\mathbf{P}(CP|FN, ZO)$, $\mathbf{P}(CP|\neg FN, ZO)$, ..., $\mathbf{P}(\neg CP|\neg FN, \neg ZO)$. Par exemple, $\mathbf{P}(CP|FN, \neg ZO)$ sera estimée en comptant dans la table le nombre d'exemples pour lesquels CP est $VRAI$, FN est $VRAI$ et ZO est $FAUX$, divisé par le nombre d'exemples pour lesquels FN est $VRAI$ et ZO est $FAUX$. Cette estimation s'écrit donc : $\widehat{P}(CP|FN, \neg ZO) = \frac{1}{1+27+2} = 0.033$

On trouve ainsi, pour compléter :

$\widehat{P}(FN)$	$(54 + 1 + 7 + 27 + 3 + 2)/100$	=	0.94
$\widehat{P}(ZO)$	$(54 + 7 + 4 + 4)/100$	=	0.69
$\widehat{P}(SA \mid FN)$	$(54 + 1 + 7)/(54 + 1 + 7 + 27 + 3 + 2)$	=	0.66
$\widehat{P}(SA \mid \neg FN)$	$0/(4 + 2)$	=	0.0
$\widehat{P}(CP \mid FN, ZO)$	$54/(54 + 7 + 3)$	=	0.84
$\widehat{P}(CP \mid FN, \neg ZO)$	$1/(1 + 27 + 2)$	=	0.033
$\widehat{P}(CP \mid \neg FN, ZO)$	$0/4$	=	0.0
$\widehat{P}(CP \mid \neg FN, \neg ZO)$	$0/2$	=	0.0

4.1.2 ... et *données incomplètes*

La différence essentielle avec l'apprentissage de paramètres en présence de données complètes est que le problème de trouver les paramètres correspondant au maximum de vraisemblance

Copyright © 2010 Eyrolles.

est maintenant un problème d'optimisation non linéaire avec potentiellement de très nombreux optima locaux. Les méthodes en descente de gradient deviennent donc plus problématiques. On utilise généralement une méthode *EM* (Expectation-Maximization). (Voir figure 11.10.)

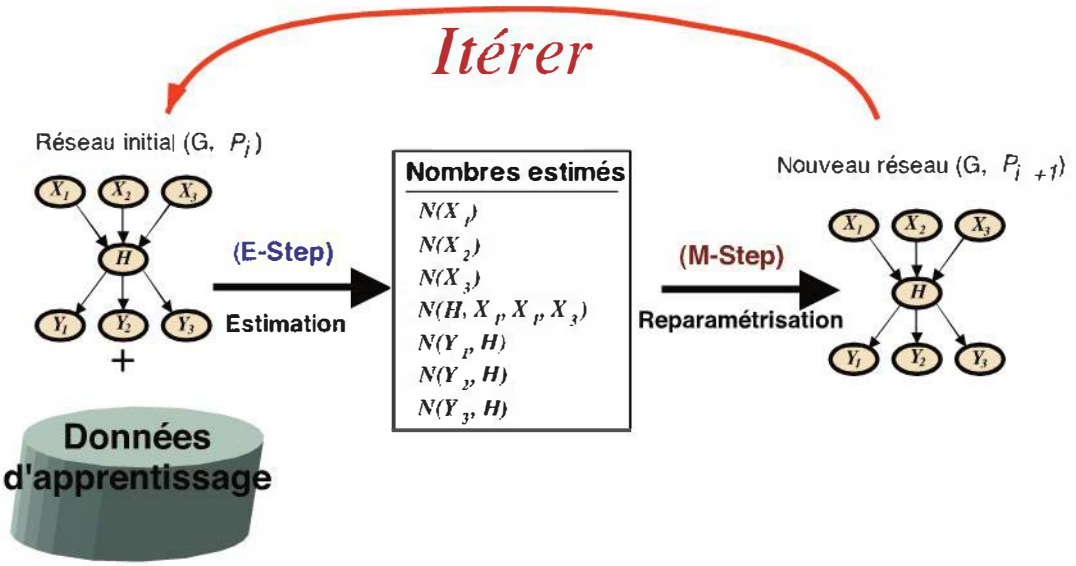


FIG. 11.10: *Principe général de la méthode EM. (D'après un tutoriel de Nir Friedman.)*

La méthode *EM* est décrite dans l'annexe 7. Nous l'illustrons ici sur l'exemple vu plus haut.

EXEMPLE **Méthode *EM* pour les réseaux bayésiens**

Reprenons l'exemple précédent en supposant que les informations que nous possédons sont incomplètes, c'est-à-dire que certaines instanciations des variables sont inconnues. Dans cet exemple, trois valeurs manquent : elles sont indiquées par « ? ».

SA	CP	FN	ZO	Nombre d'exemples
VRAI	*VRAI*	*VRAI*	*VRAI*	54
VRAI	*VRAI*	*VRAI*	*FAUX*	1
?	?	*VRAI*	*VRAI*	7
VRAI	*FAUX*	*VRAI*	*FAUX*	27
FAUX	*VRAI*	?	*VRAI*	3
FAUX	*FAUX*	*VRAI*	*FAUX*	2
FAUX	*VRAI*	*FAUX*	*VRAI*	4
FAUX	*FAUX*	*FAUX*	*FAUX*	2
				100

Est-il encore possible de réaliser une estimation rationnelle des probabilités conditionnelles caractéristiques ? La réponse (positive) est donnée par l'utilisation de l'algorithme *EM* (voir l'annexe 7). Mais il faut d'abord transformer un peu le problème.

Prenons les trois exemples pour lesquels il manque la valeur de B. Dans le tableau des données, en tenant compte de ce que $\mathbf{P}(\neg FN|\neg SA,CP,ZO) = 1 - \mathbf{P}(FN|\neg SA,CP,ZO)$, on peut les remplacer par les « exemples virtuels » suivants :

SA	CP	FN	ZO	Nombre d'exemples
FAUX	*VRAI*	*VRAI*	*VRAI*	$3 \times \mathbf{P}(\neg FN|\neg SA,CP,ZO)$
FAUX	*VRAI*	*VRAI*	*VRAI*	$3 \times (1 - \mathbf{P}(FN|\neg SA,CP,ZO))$
				100

Ceci appelle deux remarques : d'abord que le « nombre » de certains exemples n'est plus entier ; mais cela ne gêne pas les calculs. Ensuite, que la valeur $\mathbf{P}(\neg FN|\neg SA,CP,ZO)$ est

Copyright © 2010 Eyrolles.

inconnue et que pour la calculer par les techniques d'inférence, il faudrait connaître toutes les probabilités conditionnelles caractéristiques du réseau.

Cette seconde objection paraît insurmontable ; en réalité, l'algorithme *EM* est capable d'estimer itérativement la valeur cachée $\mathsf{P}(\neg FN \mid \neg SA, CP, ZO)$. La technique est la suivante. Supposons cette valeur connue; on peut compléter l'ensemble d'apprentissage comme ci-dessus, par des exemples virtuels. Cet ensemble étant maintenant complet, on peut appliquer les méthodes d'inférence du paragraphe précédent pour estimer les probabilités conditionnelles caractéristiques du réseau... parmi laquelle se trouve la valeur inconnue $\mathsf{P}(\neg FN|\neg SA, CP, ZO)$. Cette nouvelle valeur sert à fabriquer un nouvel ensemble d'apprentissage, et ainsi de suite jusqu'à stabilisation de la valeur inconnue.

Le même raisonnement peut s'appliquer aux deux valeurs manquantes des sept exemples pour lesquels on a $FN = VRAI$ et $ZO = VRAI$. Mais ici, il y a deux valeurs cachées[4] : $\mathsf{P}(SA, CP|FN, ZO)$ et $\mathsf{P}(SA, \neg CP \mid FN, ZO)$[5].

La raison de la convergence de l'algorithme *EM* est esquissée dans l'annexe 7, mais la preuve complète sur ce cas particulier des réseaux bayésiens est hors du champ de cet ouvrage. D'autres exemples sont également développés dans cette annexe. D'autres applications de cet algorithme à l'apprentissage se trouvent au chapitre 12, qui traite des *modèles de Markov cachés* et au chapitre 18, dont le sujet est la classification non supervisée.

4.2 Apprentissage de la *structure* avec *données complètes*

Il se peut que nous ne disposions pas de modèle des indépendances conditionnelles entre variables *a priori*. Il faut alors apprendre à la fois le réseau bayésien codant ces indépendances et les paramètres associés. Cela correspond à induire un Graphe Acyclique Orienté (DAG, ou *Directed Acyclic Graph*) satisfaisant les conditions de Markov sur la distribution de probabilité Θ engendrant les données.

L'apprentissage de la structure du réseau est intéressante à plusieurs titres :

- Cela peut conduire à une *meilleure généralisation* à partir des données. En effet, le réseau code des indépendances, ce qui signifie qu'il y a moins de paramètres à apprendre, et donc un espace d'hypothèses plus contraint.
- Cela permet d'obtenir des propriétés structurales inaccessibles avec d'autres représentations non structurées. On peut ainsi mettre à jour des indépendances, mais aussi des relations de cause à effet entre variables.

Malheureusement, il faut tout de suite dire que le nombre de structures possibles à partir de n noeuds est super-exponentiel [Rob77] :

$$NS(n) = \begin{cases} 1 & n = 0 \text{ ou } 1 \\ \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-1)} NS(n-i) & \text{si } n > 1 \end{cases} \tag{11.9}$$

qui est une formule récursive. Ainsi, $NS(5) = 29281$, et $NS(10) = 4.2 \times 10^{28}$. Il est clair qu'une recherche exhaustive est exclue. Il faut donc avoir recours à des méthodes de recherche heuristique.

Il existe deux grandes familles d'approches pour apprendre la structure d'un réseau bayésien à partir de données :

1. **Les approches basées sur les contraintes.** Le principe est de tester les indépendances conditionnelles et de chercher une structure de réseau cohérente avec les dépendances et indépendances observées.

[4] Pour un exemple ayant K valeurs inconnues, il y a 2^{K-1} valeurs cachées.

[5] Il y a en réalité quatre probabilités conditionnelles inconnues, mais les deux autres, $\mathsf{P}(\neg SA, CP|FN, ZO)$ et $\mathsf{P}(\neg SA, \neg CP|FN, ZO)$ se déduisent des deux premières.

Copyright © 2010 Eyrolles.

2. **Les approches utilisant une fonction de score**. Dans ces approches, un score est associé à chaque réseau candidat, mesurant l'adéquation des (in)dépendances codées dans le réseau avec les données. On cherche alors un réseau maximisant ce score.

Ces deux familles d'approches sont bien fondées (du point de vue des statistiques), c'est-à-dire qu'avec suffisamment de données, l'apprentissage converge vers une structure correcte dans les deux cas. Cependant les premières sont sensibles aux erreurs dans les tests d'indépendance, tandis que pour les secondes la recherche d'une structure optimale est un problème NP-difficile.

4.2.1 Apprentissage par recherche directe des dépendances conditionnelles

L'idée générale de ces approches est de déterminer dans un premier temps un graphe non dirigé exprimant les dépendances entre variables détectées par des tests statistiques (e.g. par χ^2). Cela permet d'obtenir un représentant de tous les graphes équivalents au sens de la relation d'équivalence de Markov (voir la section 2.5).

En conséquence d'une telle équivalence, B_1 et B_2 partagent les mêmes V-stuctures et « arcs inférés ». Tous les graphes équivalents peuvent être représentés par un graphe partiellement orienté (squelette, V-structures et arcs inférés) appelé CPDAG (*Completed Partially Directed Acyclic Graph*). Ce CPDAG est le représentant de la classe d'équivalence.

La construction du CPDAG est abordée différemment en fonction des algorithmes :

- par une approche montante : *ajout d'arêtes* (algorithmes de Pearl et Verma) ;
- par une approche descendante : *suppressions d'arêtes* (algorithmes SGS).

Dans un deuxième temps, ce CPDAG est transformé pour obtenir un graphe orienté, par propagation de contraintes sur les orientations des arcs, et incluant éventuellement des V-structures.

Les difficultés principales rencontrées concernent :

- la fiabilité du test d'indépendance conditionnelle quand les variables sont en grand nombre et les données en nombre limité ;
- l'explosion du nombre de tests à effectuer.

Ces difficultés conduisent à l'utilisation d'heuristiques pour y faire face et qui distinguent les différents algorithmes proposés.

Les approches utilisant une fonction de score étant les plus utilisées, nous nous concentrons sur celles-ci dans la suite.

4.2.2 Les fonctions de score

Le score naturel pour évaluer une structure est sa vraisemblance. Sans entrer dans les détails de sa dérivation, celle-ci peut s'écrire :

$$\begin{aligned} I(G : \mathcal{S}) &= \log \mathcal{L}(G : \mathcal{S}) \\ &= m \sum_{j=1}^{d} \left(I(\mathrm{x}(j) : Pa^G(j)) - H(\mathrm{z}(j)) \right) \end{aligned}$$

où $H(X)$, l'*entropie*, mesure combien X encode d'information et où $I(X;Y)$ est l'*information mutuelle* entre les variables X et Y, à savoir $I(X;Y) = H(X) - H(X|Y)$, et mesure l'information que chaque variable fournit sur l'autre variable ($I(X;Y) \geq 0$, $I(X;Y) = 0$ ssi X et Y sont indépendantes, et $I(X;Y) = H(X)$ ssi X est totalement prédictible connaissant Y).

Cette formule est séduisante car elle correspond bien à une mesure intuitive de la qualité du réseau : elle favorise les structures dans lesquelles les variables sont maximalement dépendantes de leurs parents, ce qui est ce que l'on veut.

Copyright © 2010 Eyrolles.

Malheureusement, ce score conduit à préférer les réseaux trop proches des données car il est toujours meilleur d'ajouter un arc (puisque $I(X;Y) \leq I(X;Y,Z)$). De ce fait, il y a risque de surapprentissage, c'est-à-dire que le réseau code des corrélations accidentelles dans les données, qui ne correspondent pas à des régularités vraies. C'est pourquoi il faut utiliser des techniques permettant de combattre ce phénomène, c'est-à-dire permettant de contrôler l'induction (voir les chapitres 2, 3 et 21).

- *Restriction de l'espace d'hypothèses.* Par exemple en limitant le nombre de parents possibles pour chaque nœud ou le nombre de paramètres dans les tables de probabilités conditionnelles.
- *Régularisation par utilisation du principe de description minimale (MDLP).* Ce principe est décrit plus précisément dans le chapitre 21, mais il consiste essentiellement à chercher un compromis entre l'adéquation aux données et la complexité du modèle choisi pour en rendre compte.
- *Estimation bayésienne.* (voir chapitre 21). Il s'agit de calculer une moyenne sur toutes les valeurs de paramètres possibles.
- On peut aussi vérifier la qualité du réseau appris en utilisant une *technique de validation* par un ensemble test (voir chapitre 3).
- D'autres techniques de contrôle de l'espace d'hypothèses et de sélection de modèles existent (voir le chapitre 21). Elles sont moins utilisées pour l'apprentissage de réseaux bayésiens.

Le score le plus fréquemment utilisé est celui du *principe de description minimale (MDLP)*. Selon le $MDLP$, il faut choisir le réseau $\mathcal{B}$ tel que la somme de la longueur de description du réseau et celle des données encodées à l'aide de $\mathcal{B}$ soit minimale. Ce principe conduit à chercher un réseau juste assez complexe pour pouvoir raisonnablement décrire les données en l'utilisant. La description d'un réseau $\mathcal{B}$ implique celle du graphe G et celle de l'ensemble des distributions de probabilités conditionnelles associées Θ. D'où la formule à minimiser sur $\mathcal{B}$:

$$L(\mathcal{S}) \;=\; L(\mathcal{B}) \,+\, L(\mathcal{S}|\mathcal{B}) \;=\; L(G) \,+\, L(\Theta) \,+\, L(\mathcal{S}|\mathcal{B})$$

Pour **décrire le graphe** acyclique orienté G, il suffit de coder pour chaque variable X_j une description de ses parents $Pa(j)$. Pour cela, il faut coder le nombre k de parents et l'index de l'ensemble $Pa(j)$ dans l'énumération de tous les ensembles de taille k parmi n nœuds $\binom{n}{k}$. Comme k peut être codé en utilisant $\log n$ bits et l'index par $\log \binom{n}{k}$ bits, la longueur de description de la structure est :

$$L(G) \;=\; \sum_j \left(\log n + \log \binom{n}{|Pa(j)|} \right)$$

où les log sont en base 2 pour obtenir une longueur en bits, et où $\binom{n}{k}$ représente en notation américaine le nombre de combinaisons de n éléments pris k à k.

Pour **décrire** Θ, il faut coder les paramètres de chaque table de probabilités conditionnelles associées à chaque nœud. Pour la table associée à une variable X_j, il faut coder $\|Pa(j)\|(\|X_j\|-1)$ paramètres. La longueur de description de ces paramètres dépend du nombre de bits utilisés pour coder chaque paramètre numérique. Le choix usuel est $1/2 \log m$ (voir [M.99], p.428). D'où :

$$L(X_j, Pa(j)) \;=\; \frac{1}{2} \|Pa(j)\| \, (\|X_j\| - 1) \, \log m$$

Pour décrire les données, on utilise la mesure de probabilité $\mathsf{P}_{\mathcal{B}}$ définie par le réseau $\mathcal{B}$ pour construire un code de Huffman pour les exemples dans $\mathcal{S}$. Dans ce code, la longueur de chaque mot de code dépend de la probabilité assignée à cet exemple. Cette longueur de description est

Copyright © 2010 Eyrolles.

ainsi approximée par :

$$L(\mathcal{S}|\mathcal{B}) = -\sum_{i=1}^{m} \log \mathbf{P}_{\mathcal{B}}(\mathbf{x})$$

que l'on peut réécrire après calculs ([M.99], p.429)) : $m \sum_j H(X_j|Pa(j))$

D'où l'expression finale de la longueur totale de description associée à un réseau $\mathcal{B}$ et des données $\mathcal{S}$:

$$L(\mathcal{S}) = \underbrace{\sum_j \left(\log n + \log \binom{n}{|Pa(j)|} \right)}_{L(G)} + \underbrace{\frac{1}{2} \|Pa(j)\| \left(\|X_j\| - 1\right) \log m}_{L(\theta)} + \underbrace{m \sum_j H(X_j|Pa(j))}_{L(\mathcal{S}|\mathcal{B})}$$

qui a l'avantage d'être décomposable et donc de se prêter à des méthodes de recherche locale, où l'on cherche à améliorer le score par des modifications de chaque nœud.

D'**autres fonctions de score** ont été proposées. À notre connaissance, elles reposent toutes sur l'utilisation de la fonction de vraisemblance $\log \mathcal{L}(S|\mathcal{B})$, ou encore $\log \mathcal{L}(\mathcal{S}|\mathcal{B}, \Theta)$ et sur une mesure de la complexité du réseau bayésien inféré $\dim(\mathcal{B})$.

Par exemple, en *utilisant le critère AIC* (voir chapitre 21) :

$$S_{AIC} = \log \mathcal{L}(S|\Theta^{MV}, G) - \mathrm{Dim}(G)$$

où P^{MV} est la valeur des paramètres estimés par maximum de vraisemblance.

En *utilisant le critère BIC* (voir chapitre 21), on a :

$$S_{BIC} = \log \mathcal{L}(S|\Theta^{MV}, G) - \frac{1}{2}\mathrm{Dim}(G) \log(N)$$

4.2.3 L'apprentissage avec fonction de score

Une fois un score défini sur les réseaux, l'apprentissage consiste à trouver le réseau minimisant ce score. Malheureusement, les résultats actuels suggèrent qu'il s'agit là d'un problème NP-difficile. En effet, le nombre de graphes est plus qu'exponentiel en fonction du nombre (donné) de variables. Il faut donc avoir recours à des techniques d'exploration heuristiques. Nous ne discutons ici que de la plus simple : la *descente de gradient*.

La démarche consiste à démarrer avec un réseau (souvent le réseau vide) et à appliquer itérativement sur le réseau candidat courant l'opérateur (par exemple *ajout* ou *retrait* ou *inversion* d'arc) conduisant au meilleur accroissement du score. Cette procédure est répétée jusqu'à ce qu'un maximum (local) soit atteint. Les résultats expérimentaux montrent que cette technique est souvent très efficace malgré son caractère myope et glouton. L'un des problèmes est de vérifier que l'on conserve à chaque pas un graphe acyclique orienté. Le choix de la direction d'un arc est également un problème non trivial à résoudre.

D'autres techniques incluent la recherche tabou, le recuit simulé, etc. (voir le chapitre 3).

4.3 Apprentissage de la *structure* avec *données incomplètes*

4.3.1 Le problème des données incomplètes

Il arrive fréquemment que les données réelles soient incomplètes, c'est-à-dire que *des variables utiles ne soient pas mesurées*. Cela peut se produire en raison de valeurs manquantes (par exemple tous les patients rentrant à l'hôpital ne subissent pas les mêmes examens), mais aussi parce que des variables sont inobservables ou insoupçonnées comme c'est le cas lorsqu'un phénomène est

Copyright © 2010 Eyrolles.

mal connue et que des variables causales sous-jacentes sont ignorées. On parle alors de *variables latentes* ou de variables cachées.

Le problème est que l'absence de ces variables non seulement obscurcit la compréhension des dépendances propres au domaine, mais peut également conduire à apprendre trop de paramètres comme le montre la figure 11.11.

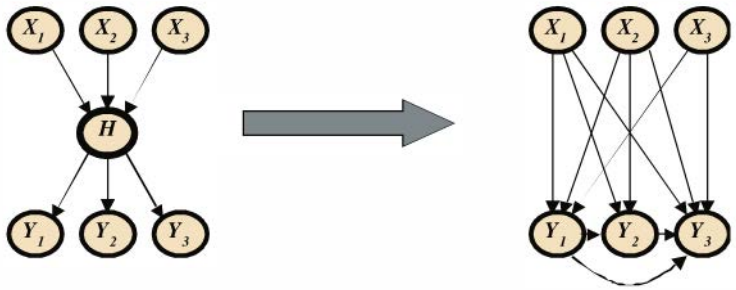


FIG. 11.11: *L'absence d'une variable peut conduire à avoir à estimer 59 paramètres (à droite) au lieu de 17 (à gauche). (D'après un tutoriel de Nir Friedman.)*

La recherche de variables latentes est l'un des plus gros problèmes en apprentissage de réseaux bayésiens.

4.3.2 Techniques d'apprentissage

L'apprentissage de structure en présence de données incomplètes conjugue donc tous les problèmes examinés jusqu'ici. Pour le moment, il s'agit encore d'un domaine de recherche avec des méthodes en cours de conception et d'examen.

L'approche la plus simple est de combiner les méthodes utilisées pour l'apprentissage de paramètres et l'apprentissage de structure. L'idée est pour chaque structure G d'estimer les paramètres optimaux P en utilisant soit une technique de descente de gradient, soit une technique EM, puis d'associer un score à G. On examine alors les graphes obtenus à partir de G avec des opérateurs de changement de structure, et on choisit celui ayant le meilleur score.

Le problème majeur de cette approche est son coût calculatoire très élevé. Elle n'est praticable que pour de très petits réseaux.

Une autre approche a été proposée utilisant une généralisation de la méthode EM : l'*EM*-structurel [Fri97]. L'idée est d'utiliser les paramètres trouvés pour les structures précédentes pour aider à évaluer de nouvelles structures portant sur le même ensemble de variables aléatoires.

Le principe général en est le suivant (voir la figure 11.12) :

1. Effectuer une recherche dans l'espace des structures et des paramètres.
2. Utiliser des itérations à la EM, en employant la meilleure solution précédemment trouvée comme d'une base pour soit :
   - trouver les meilleurs paramètres (en fonction du score) : EM « paramétrique » ;
   - trouver la meilleure structure (en fonction du score) : EM « structurel ».

Des développements plus récents essaient d'échapper au problème des optima locaux dans lesquels EM-structurel peut rester piégé (voir par exemple [EF05]). De même, des extensions de ces techniques existent pour apprendre des réseaux bayésiens dans lesquels les variables (dans les nœuds) peuvent prendre des valeurs continues (voir par exemple [ENF07]). Il faut alors émettre des hypothèses fortes sur le type de distribution qui régissent ces variables.

Copyright © 2010 Eyrolles.

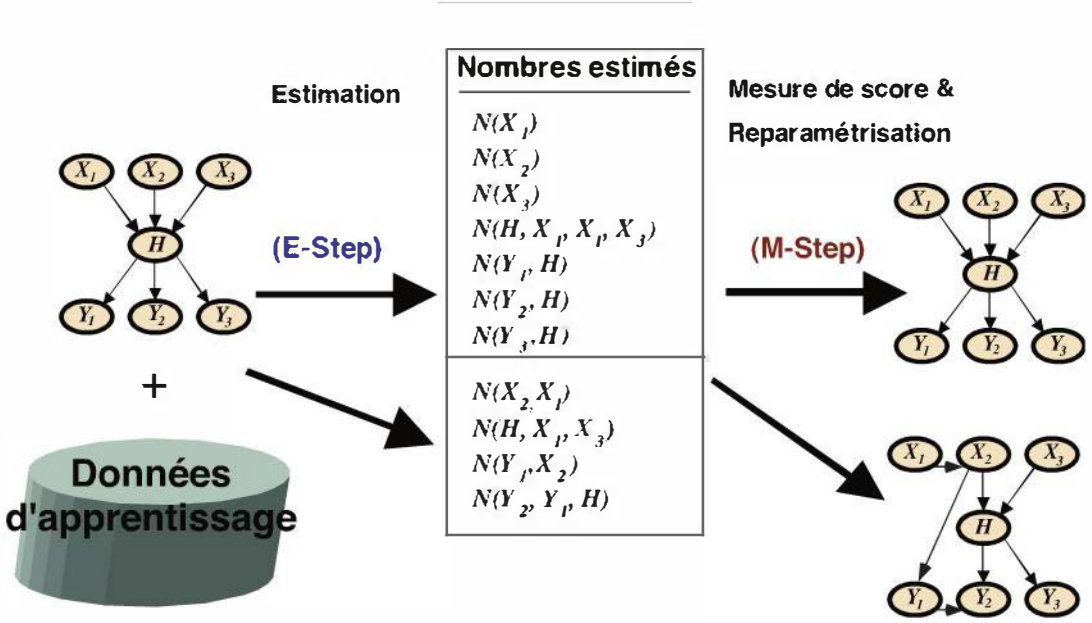


FIG. 11.12: *Principe général de la méthode EM-structurel. (D'après un tutoriel de Nir Friedman.)*

5. L'inférence de relations causales

Il faut être conscient qu'un réseau bayésien exprime des dépendances et indépendances conditionnelles, mais pas des relations de causalité. Par exemple, un lien $A \rightarrow B$ ne signifie pas qu'il existe un lien de causalité entre A et B, et encore moins que A est une cause de B.

Il est vrai qu'un graphe construit par un expert reflète souvent des liens de causalité car c'est le mode de pensée privilégié par les humains. En revanche, un graphe appris à partir de données n'a aucune raison de respecter ou d'exprimer des causalités. Il faut bien distinguer la notion de corrélation de celle de causalité.

En inférence classique, on observe par exemple que $B = b$ et on veut calculer $\mathbf{P}(A|B = b)$. Il n'entre pas de notion de causalité dans ce calcul. Si la causalité n'est pas utile pour l'inférence, pourquoi alors chercher un réseau bayésien causal (dans lequel les arcs expriment des relations de causalité directe) ?

La raison en est que, dans de nombreuses situations, on cherche à modéliser un phénomène complexe pour **pouvoir agir** dessus. Ainsi, par exemple, on voudra modéliser problème de santé publique afin de déterminer les facteurs sur lesquels agir pour réduire la mortalité ou des effets néfastes observés sur la population. On introduit alors la notion d'*action sur une variable* (voir [Pea09]) que l'on notera $do(B = b)$ par exemple, pour reprendre l'illustration précédente.

Si A est la cause de B, on aura :

$$\mathbf{P}(A|do(B = b)) = \mathbf{P}(A)$$
$$\mathbf{P}(B|do(A = a)) = \mathbf{P}(B|A = a)$$

Le principe général des méthodes existantes est d'apprendre d'abord un graphe acyclique orienté à partir des données, ce graphe étant un représentant de la classe d'équivalence au sens

Copyright © 2010 Eyrolles.

de Markov. Ensuite, à partir de données d'expérimentation si c'est possible, on cherche à orienter les arcs du graphe pour refléter les causalités.

Pour cette seconde étape, on cherche comment la loi jointe change lorsque l'on manipule certaines variables ($do(M = m)$). Intuitivement, on oublie les « causes officielles » de M (ses parents dans le graphe) et on observe les effets que déclenche $M = m$ pour ses « enfants », des variables X_i.

$$\mathsf{P}(X_i(M = m)) = \left(\prod_{X_i \in \mathcal{N} \setminus \{M\}} \mathsf{P}(X_i | \mathcal{P}a((X_i)) \right)_{M=m} \tag{11.10}$$

Les grandes familles d'approches sont :

- **Fondées sur les contraintes**. Elles s'appuient sur des tests d'indépendance conditionnelles. Les techniques correspondantes incluent les algorithmes IC (*Inductive Causation*) (Pearl), PC (Peter Spirtes et Clarck Glymour), TPDA (*Three-Phase Dependency Analysis*) (Cheng).
- **Fondées sur des scores**. L'exploration se fait dans la classe des graphes acycliques orientés, ou d'un espace de graphes, selon un certain critère à optimiser. On citera particulièrement l'algorithme GES (*Greedy Equivalence Search*) (Meek et Chickering).
- **Mixtes**. Elles utilisent à la fois des tests d'indépendance et des fonctions de score. C'est le cas de l'algorithme MMHC (*Max-Min Hill Climbing*) (Tsamardinos).

Ces approches supposent généralement que toutes les variables utiles sont présentes dans le graphe (propriété de *suffisance causale*). Si celle-ci n'est pas vérifiée, on se retrouve à devoir identifier ces variables en plus (voir la section 4.3 sur l'apprentissage de structure à partir de données incomplètes).

Deux ouvrages de référence ont introduit les problèmes de la détermination de la causalité, ainsi que des concepts utiles et des algorithmes pour aborder ces problèmes [Pea09, SGS01].

À côté du problème récurrent d'estimation de probabilités conditionnelles lorsque les échantillons d'apprentissage sont de petite taille, l'un des obstacles essentiels aux méthodes existantes est leur complexité calculatoire. C'est pourquoi il a été récemment proposé d'avoir recours aux couvertures de Markov (voir section 2.6) pour réduire le nombre de variables à considérer à l'intérieur des algorithmes. L'expérience montre que cela permet de traiter des réseaux plus gros et d'améliorer l'apprentissage.

6. Applications

Les réseaux bayésiens possèdent plusieurs propriétés qui les rendent intéressants pour des applications.

- Ils s'adaptent sans problème aux bases de données incomplètes.
- Ils sont conçus pour rendre compte de relations causales.
- Ils permettent d'intégrer des connaissances du domaine et des données plus facilement que beaucoup d'autres techniques.
- Par leur parenté avec les approches d'induction bayésienne, ils sont mieux armés pour éviter les problèmes de surapprentissage (voir le chapitre 3).

Si chacun de ces points peut être soumis à discussion, il n'en reste pas moins que les réseaux bayésiens sont des modèles très séduisants.

Copyright © 2010 Eyrolles.

Peu utilisent les fonctions d'apprentissage. Cependant il est certain que, compte tenu du nombre déjà grand d'applications, cet aspect ne peut qu'être appelé à se développer. Pour donner un ordre de grandeur, certains réseaux fonctionnent actuellement avec plusieurs milliers de nœuds[6].

- La compagnie AT&T a mis en place un réseau bayésien pour la détection des mauvais payeurs, à partir d'une base de données de plusieurs millions d'appels.
- La NASA utilise un système graphique pour suivre en temps réel les paramètres de contrôle des moteurs.
- La société Ricoh utilise ces réseaux pour le diagnostic de ses appareils de photocopie.
- L'investissement de Microsoft dans les réseaux bayésiens est important, tant en ce qui concerne le développement de logiciels que la recherche. Les applications visées sont : le diagnostic sur les réseaux, l'aide à l'utilisateur (le « trombone » d'Office en est en exemple), la validation de gros logiciels, etc.
- Et beaucoup d'applications dans le domaine de la santé, du militaire, etc.

7. Quelques logiciels

Sans être exhaustive, la liste suivante fournit une idée de logiciels disponibles pour apprendre des réseaux bayésiens.

- Le logiciel `Tetrad` `www.phil.cmu.edu/projects/tetrad/` peut être téléchargé gratuitement. Il peut apprendre un Graphe Acyclique Orienté à partir d'une loi jointe de probabilité.
- Le logiciel `Belief Network Power Constructor` (approche basée sur des contraintes) téléchargeable sur `www.cs.ualberta.ca/~jcheng/bnpc.htm`
- Le logiciel `Bayesware` peut apprendre la structure et les paramètres. Téléchargeable sur `www.bayesware.com/`.
- Le logiciel `Bayes Net Toolbox` téléchargeable sur `bnt.sourceforge.net/`.
- Le logiciel `Probabilistic Net Library` téléchargeable sur `www.eng.itlab.unn.ru/?dir=139`.
- Le logiciel `deal` écrit en `R` peut être téléchargé librement à partir du Comprehensive R Archive Network (CRAN) `http://cran.R-project.org/`. Cette librairie permet de définir des lois de probabilité *a priori*, estimer le paramètres du réseau, et apprendre une structure en utilisant une fonction de score pour des réseaux décrivant des variables mixtes : à la fois discrètes et continues.

Notes historiques et sources bibliographiques

Les réseaux bayésiens sont nés des travaux de J. Pearl [Pea88]. Les années 1990 ont vu le développement des théories et des réalisations, ces dernières en très grand nombre (voir ci-dessous). Les concepts les plus amont regroupent les HMM, les réseaux bayésiens et les réseaux connexionnistes sous le terme général de « modèles graphiques ». Il est possible que cette vision très générale permette d'envisager des algorithmes

[6] Un réseau a été mis au point à Stanford pour modéliser le fonctionnement de la cellule. Il comporte vingt-deux millions de nœuds!

Copyright © 2010 Eyrolles.

d'apprentissage plus puissants, mais cette unification théorique ne semble pas avoir encore porté de fruits, sur le plan algorithmique du moins. Les réseaux bayésiens sont actuellement l'objet de gros projets de recherche et de développement. Ces outils sont encore loin d'avoir montré toutes leurs possibilités.

Le texte de ce chapitre a été en partie inspiré par l'ouvrage remarquable (et pas seulement sur ce sujet) de N. Nilsson [Nil98]. Le livre de Patrick Naïm, Pierre-Henri Wuillemin, Philippe Leray, Olivier Pourret et Anna Becker [NWL+07] est une introduction très recommandable aux principes et aux applications de ces outils.

Résumé

- Les réseaux bayésiens sont des modèles permettant de décrire les relations de probabilités conditionnelles entre des faits. Cette représentation repose sur un graphe orienté sans cycle (DAG) dans lequel chaque nœud, c'est-à-dire chaque variable du monde modélisé, possède une table de probabilités conditionnelles, et où chaque arc représente une dépendance directe entre les variables reliées. Ces réseaux représentent alors la distribution de probabilités jointes de l'ensemble des variables de manière compacte, en s'appuyant sur les relations d'indépendance conditionnelle.
- Moyennant une propriété locale d'indépendance, il est possible d'effectuer le calcul de la probabilité de tout groupe de faits connaissant tout autre groupe.
- L'apprentissage automatique des valeurs des probabilités conditionnelles peut se faire à partir d'un ensemble d'apprentissage, même incomplet, si la structure du réseau est donnée.
- Une tâche essentielle concerne l'apprentissage ou l'adaptation de la structure de ces modèles à partir de données. Ce problème NP-complet est typiquement traité comme un problème d'optimisation combinatoire avec utilisation de méthodes d'exploration heuristiques telles que des méthodes de gradient plus ou moins sophistiquées.
- Les nombreuses méthodes d'apprentissage de structure de réseaux bayésiens peuvent être catégorisées en deux grandes approches : les méthodes fondées sur une fonction de score avec recherche dans un espace de possibilités, et les méthodes à base de contraintes.
- Les méthodes basées sur une fonction de score évaluent la qualité de chaque structure possible par rapport aux données d'apprentissage tout en contrôlant la complexité du modèle considéré, en utilisant typiquement des mesures telles que AIC, BIC, MDL. Ce type de procédure examine les modifications locales qui peuvent être apportées au réseau, les évalue généralement grâce à une fonction de score, et applique la modification associée au meilleur gain et cela jusqu'à ce qu'un optimum local soit atteint.
- Les méthodes à base de contraintes essaient d'estimer des propriétés d'indépendance conditionnelle entre les variables dans les données. Ces méthodes sont efficaces et peuvent identifier des modèles causaux et des causes communes latentes sous certaines conditions, en particulier de disposer d'échantillons de taille suffisante.
- Même dans le cas de petits réseaux, l'apprentissage de structure est coûteux en raison du nombre exponentiel de modifications possibles à considérer et du coût de chaque évaluation. Le problème est encore plus difficile dans le cas de valeurs manquantes car le problème d'optimisation devient non linéaire. Cependant le problème le plus formidable est celui de l'apprentissage de structure avec ajout possible de variables cachées ou latentes. De ce fait, les applications actuelles concernent encore surtout l'apprentissage des paramètres d'un réseau.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 12

L'apprentissage de modèles de Markov cachés

Quand les objets sur lesquels porte l'apprentissage sont des séquences d'événements, le concept extrait doit refléter à la fois la nature de ces événements et la manière dont ils s'enchaînent. On a déjà vu au chapitre 2 et au chapitre 3 des exemples d'objets de structure séquentielle. On a vu aussi au chapitre 7 des méthodes pour extraire des concepts sous forme de grammaires à partir d'exemples et de contre-exemples. Nous présentons dans ce chapitre un outil puissant pour induire un concept de nature statistique à partir seulement de séquences d'apprentissage appartenant à ce concept : les modèles de Markov cachés, ou HMM[1].

Par leur nature statistique, les HMM *se situent facilement dans le cadre de la décision bayésienne, qui a été présentée au chapitre 2. En particulier, le principe du maximum de vraisemblance a posteriori (MAP) prescrit d'attribuer une séquence inconnue à la classe qui a la plus grande probabilité de l'avoir engendrée. L'apprentissage consiste donc dans ce cadre à apprendre pour chaque classe de séquences le* HMM *le plus vraisemblable. En pratique, le problème revient à apprendre indépendamment un* HMM *par classe, sans tenir compte des contre-exemples.*

[1] En anglais : *Hidden Markov Models.*

Copyright © 2010 Eyrolles.

Sommaire

1 Les modèles de Markov observables 386
2 Les modèles de Markov cachés (HMM) 387
2.1 Définition 387
2.2 Pourquoi faut-il des variables cachées? 387
2.3 Notations 389
2.4 Deux types de HMM 391
2.5 Comment un HMM engendre-t'il une séquence? 391
3 Les HMM comme règles de classification de séquences 392
3.1 Les trois problèmes des HMM 392
3.2 Les HMM et la classification bayésienne 392
4 L'évaluation de la probabilité d'observation 393
5 Le calcul du chemin optimal : l'algorithme de Viterbi 395
6 L'apprentissage 398
7 Approfondissements 403
8 Applications 404

REVENONS sur l'étang où nagent des oies et des cygnes. L'ornithologue que nous avons connu débutant au commencement de ce livre est maintenant plus expérimenté. Ce matin, il arrive très tôt pour observer les oiseaux se poser. La veille, il était venu dans la matinée, quand tous les animaux étaient là et il avait observé une trentaine de cygnes et environ quatre-vingts oies. Il espère donc voir arriver une bonne centaine d'oiseaux.

De fait, les vols commencent et il y a bientôt sur le lac les premiers oiseaux. Mais pas dans la proportion attendue : vingt cygnes et quatre oies se sont d'abord posés. Dans les minutes qui suivent, une dizaine d'oiseaux se posent, moitié oies, moitié cygnes. Finalement, arrive le reste de la troupe des oies, soit environ soixante-dix éléments, et de temps en temps un cygne. Au total, ces derniers sont finalement une trentaine.

L'observateur comprend alors que les deux espèces n'ont pas les mêmes habitudes : les cygnes sont plus matinaux, ce qui explique que la proportion entre les deux espèces varie énormément dans le temps d'arrivée. En notant `O` l'arrivée d'une oie et `C` celle d'un cygne, et en mettant un intervalle entre les trois phases d'arrivée, la séquence observée peut se dénoter ainsi :

```
CCCCCCOCCCCOCCCOCCCCOCO      OCCOCOOOCC
OOOOCOOOOOOOOOOOOOOCOCOOOCOOOOOOOOOOOOOOOO
```

Comment décrire ce phénomène? Attention : il ne s'agit pas seulement d'apprendre cette séquence par cœur. Il faut tenir compte du fait que l'ordre exact d'arrivée des oiseaux ne se reproduira pas exactement à l'identique chaque matin : certains oiseaux ne viendront pas, certains voleront plus ou moins vite, etc. Si l'ornithologue veut expliquer ce qu'il observe et prédire ce qu'un autre pourra observer, il doit donc faire l'apprentissage d'un concept qui décrive de manière satisfaisante les propriétés de telles séquences.

Si cet avimateur a lu le chapitre qui suit, il produira peut-être un concept exprimé sous la forme du tableau et du graphique (figure 12.1) qui sont donnés ci-dessous.

Copyright © 2010 Eyrolles.

	Probabilité d'observer un cygne	Probabilité d'observer une oie
Période 1	0.8	0.2
Période 2	0.5	0.5
Période 3	0.1	0.9

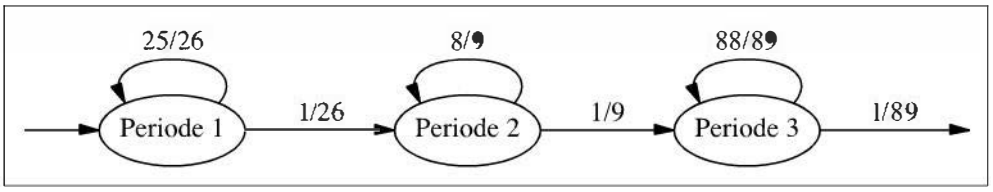


FIG. 12.1: *Comment les cygnes et les oies arrivent sur l'étang.*

Comment interpréter ce modèle ? Le cercle étiqueté « période 1 » correspond à la phase d'arrivée majoritaire des cygnes, celui étiqueté « période 2 » au moment où les populations sont en fréquence d'arrivée égale, et le dernier à l'arrivée massive des oies (avec quelques cygnes parmi elles). La succession temporelle se traduit par un parcours de gauche à droite en suivant les flèches, avec la règle que chaque passage dans un état correspond exactement à l'observation d'un oiseau, cygne ou oie. Quand on est dans un cercle (appelons-le désormais un état), on a deux solutions : soit y rester en faisant une boucle locale, soit passer au suivant. Le passage d'un état à lui-même ou au suivant est commandé par le chiffre situé sur la flèche, qui est une probabilité. Par exemple, dans l'état 1, la probabilité de passer à l'état 2 est de 1/26, celle de rester dans l'état 1 est de 25/26.

Et les oiseaux ? Leur observation est commandée par la table donnée au-dessus du graphe des états. Chaque passage dans l'état 1 correspond avec une probabilité de 0.8 à l'observation d'un cygne et donc de 0.2 à celle d'une oie. Dans l'état 2, l'observation est équiprobable. Quand on est dans l'état 3, il est 9 fois plus probable d'observer une oie qu'un cygne.

Effectuons maintenant un petit calcul. Combien d'oiseaux sont en moyenne observés pendant le séjour dans l'état 1 ? Environ 25, selon une formule simple du calcul des probabilités[2]. Par conséquent, compte tenu des proportions affectées par le tableau, on observera en moyenne $0.8 \times 25 = 20$ cygnes et $0.2 \times 25 = 5$ oies durant la première période représentée par cet état. Un calcul analogue montre que la durée moyenne de séjour dans l'état 2 est d'environ 8 : on y observera donc (en moyenne) 4 cygnes et 4 oies. Finalement, comme la probabilité de bouclage dans l'état 3 est la plus forte, on y reste en moyenne plus longtemps (le calcul donne 88) et on observe, toujours en moyenne, 80 oies et 8 cygnes.

Au total, la séquence moyenne engendrée par ce modèle comporte une trentaine de cygnes et presque trois fois plus d'oies, avec une proportion d'arrivées des cygnes beaucoup plus forte au début qu'à la fin.

Comme nous allons le voir, le concept décrit ci-dessus est un cas particulier de modèle de Markov caché (HMM). Dans un tel modèle, une séquence est donc considérée comme une suite temporelle gérée par ses états. À chaque instant, un nouvel événement de la séquence est analysé. La théorie des HMM décrit comment passer d'état en état à l'aide de probabilités de transitions et comment chaque élément de la séquence peut être émis par un état du HMM, à l'aide de probabilités d'observation par état. Il permet aussi de calculer la probabilité qu'une séquence donnée ait été émise par un HMM donné.

[2] Si x est la probabilité de boucler dans un état et $1 - x$ celle d'en sortir, la durée moyenne de séjour dans cet état vaut $\frac{x}{(1-x)}$.

Copyright © 2010 Eyrolles.

Les méthodes HMM sont robustes et fiables grâce à l'existence de bons algorithmes d'apprentissage ; de plus, la règle de décision est rapide à appliquer.

Notations utiles pour le chapitre

n	Le nombre d'états du modèle de Markov caché, ou HMM
$S = \{s_1, s_2, \ldots s_n\}$	Les états du HMM
M	La taille de l'alphabet des observations quand celles-ci sont de nature discrète
$V = \{v_1, v_2, \ldots, v_M\}$	L'alphabet des observations
A	La matrice des probabilités de transitions entre les états
a_{ij}, $i, j \in [1,n]$	Un élément de A
B	La matrice des probabilités d'observation des symboles de V
$b_j(k)$, $j \in [1,n], k \in [1, M]$	Un élément de B
π	Le vecteur des probabilités initiales du HMM
$\Lambda = (A, B, \pi)$	Un HMM
T	La longueur d'une séquence observée
$O = O_1 \ldots O_t \ldots O_T$ avec $O_t \in V$	Une séquence observée
$O(i:j) = O_i \ldots O_j$	Une sous-séquence de O
$q_1 \ldots q_t \ldots q_T$ avec $q_t \in S$	La suite des états qui a émis une séquence
$\mathbf{P}(O \mid \Lambda)$	La probabilité que le HMM Λ ait émis la séquence O
$\mathcal{O} = O^1 \ldots O^m$	Un ensemble d'apprentissage composé de m séquences
$\mathbf{P}(\Lambda \mid \mathcal{O})$	La probabilité que l'ensemble de séquences $\mathcal{O}$ ait été émis par le HMM Λ.

1. Les modèles de Markov observables

Avant de décrire les HMM proprement dits, nous présentons un modèle probabiliste plus simple pour l'observation de séquences : les *modèles de Markov observables*.

D'une manière générale, un *processus* ou *modèle stochastique observable* est un processus aléatoire qui peut changer d'état s_i, $i = 1, \ldots, n$ au hasard, aux instants $t = 1, 2, \ldots, T$. Le résultat observé est la suite des états dans lesquels il est passé. On peut aussi dire de ce processus qu'il *émet* des séquences d'états $S = s_1, s_2, \ldots, s_T$. Chaque séquence est émise avec une probabilité[3] $\mathbf{P}(S) = \mathbf{P}(s_1, s_2, \ldots, s_T)$. Pour calculer $\mathbf{P}(S)$, il faut se donner la probabilité initiale $\mathbf{P}(s_1)$ et les probabilités d'être dans un état s_t, connaissant l'évolution antérieure.

Un processus stochastique est *markovien*[4] (ou *de Markov*) si son évolution est entièrement déterminée par une probabilité initiale et des probabilités de transitions entre états. Autrement dit, en notant $(q_t = s_i)$ le fait que l'état observé à l'instant t est s_i

$$\forall t, \quad \mathbf{P}(q_t = s_i \mid q_{t-1} = s_j, q_{t-2} = s_k \ldots)\mathbf{P}(q_t = s_i \mid q_{t-1} = s_j)$$

[3] Dans ce chapitre, nous étudions principalement des distributions de probabilité sur des ensembles finis.

[4] Au sens strict : markovien *d'ordre 1*.

Copyright © 2010 Eyrolles.

d'où :

$$\mathbf{P}(q_1 \dots q_T) = \mathbf{P}(q_1) \times \mathbf{P}(q_2 \mid q_1) \times \dots \times \mathbf{P}(q_T \mid q_{T-1})$$

Nous supposons pour simplifier que les processus de Markov auxquels nous avons affaire sont *stationnaires* c'est-à-dire que leurs probabilités de transition ne varient pas dans le temps. Cela autorise à définir une *matrice de probabilité de transitions* $A = [a_{ij}]$ telle que :

$$a_{ij} = \mathbf{P}(q_t = s_j \mid q_{t-1} = s_i) \quad 1 \leq i \leq n, \quad 1 \leq j \leq n$$

avec :

$$\forall i,j \quad a_{ij} \geq 0, \quad \forall i \ \Sigma_{j=1}^{j=n} a_{ij} = 1$$

Nous appellerons maintenant pour simplifier *modèle de Markov observable* un processus stochastique observable, markovien et stationnaire.

Dans un tel modèle, il y a un lien direct à tout instant entre l'état où se trouve le processus et l'observation réalisée à cet instant, comme l'illustre la figure 12.2. C'est ce qui caractérise pour nous[5] le fait que ce processus soit observable. Nous allons maintenant voir comment nous débarasser de cette contrainte en présentant d'autres processus stochastiques : les modèles de Markov cachés. Ensuite, nous comparons leur puissance de modélisation sur un exemple.

2. Les modèles de Markov cachés (HMM)

2.1 Définition

Le modèle de Markov caché généralise le modèle de Markov observable car il produit une séquence en utilisant deux suites de variables aléatoires; l'une cachée et l'autre observable.

- La suite cachée correspond à la suite des états $q_1, q_2, \dots, q_T$, notée $Q(1 : T)$, où les q_i prennent leur valeur parmi l'ensemble des n états du modèle $\{s_1, s_2, \dots, s_n\}$.
- La suite observable correspond à la *séquence des d'observations* $O_1, O_2, \dots, O_T$, notée $O(1 : T)$, où les O_i sont des lettres d'un alphabet de M *symboles observables* $V = \{v_1, v_2, \dots, v_M\}$.

Par conséquent, pour un HMM, un état n'est pas associé exclusivement à une lettre donnée qu'il émettrait à coup sûr : chaque lettre a désormais une certaine probabilité d'être émise par chaque état. En outre, ce ne sont pas les états qui sont observés, mais les lettres qu'ils émettent. Une conséquence importante est que l'on peut maintenant travailler avec des alphabets infinis. Une « lettre » est alors émise avec une certaine densité de probabilité, correspondant à une distribution propre à chaque état.

En pratique, on cherche à construire des HMM représentant des concepts dans l'espace de représentation des séquences. Nous prendrons ici pour simplifier des séquences construites sur un alphabet $V = \{v_1, v_2, \dots, v_M\}$ de taille finie. Mais la remarque ci-dessus doit être gardée à l'esprit : la taille de cet alphabet peut être infinie, ce qui signifie en pratique que chaque état peut émettre une variable continue ou un vecteur de $\mathbb{R}^d$.

2.2 Pourquoi faut-il des variables cachées ?

Montrons sur l'exemple de l'introduction la différence entre le modèle de Markov observable et le modèle de Markov caché. Quand on observe l'arrivée des oiseaux sur un étang, on obtient

[5] Si la même observation peut être affectée à plusieurs états, on peut améliorer la capacité de représentation des modèles observables. Nous ne discutons pas cette possibilité ici.

Copyright © 2010 Eyrolles.

une suite sur l'alphabet $V = \{\texttt{O}, \texttt{C}\}$. Une séquence observée sera par exemple :

$$O = \texttt{O O C O C O}$$

Les probabilités *a priori* d'observer un cygne ou une oie peuvent être différentes. La construction de deux types de modèles de Markov pour modéliser les séquences sur V va nous conduire à préciser un certain nombre d'éléments relatifs à la nature des états, à leur nombre ainsi qu'aux probabilités de transition et d'observation. Un modèle de Markov observable pour ce problème est représenté dans la figure 12.2.

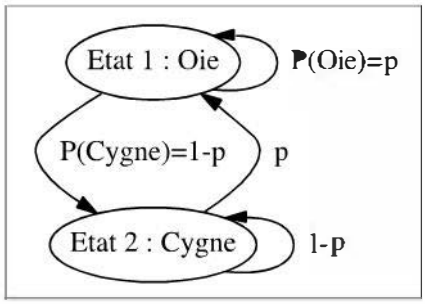


FIG. 12.2: *Le modèle de Markov observable qui modélise la suite des observations des oies et des cygnes.*

Il est composé de deux états; chacun correspond directement à une observation possible : Oie (O) ou Cygne (C). Dans ce modèle, la suite d'états associée à une séquence observée est facile à déterminer : l'observation de O correspond à l'état 1 et l'observation de C correspond à l'état 2. Si la probabilité d'observer O à l'état 1 est $p = \mathrm{P}(\textit{Oie})$, alors la probabilité d'observer C à l'état 2 est $1 - p$. La probabilité d'observer la séquence $O(1:6) = \texttt{O O C O C O}$ se calcule facilement ; elle vaut :

$$p\ p\ (1-p)\ p\ (1-p)\ p = p^4\ (1-p)^2$$

Elle est par conséquent indépendante de l'ordre d'arrivée des oiseaux et ne tient compte que de leur nombre dans la séquence. Ce modèle n'exprime que les probabilités d'apparition *a priori* des observations.

La figure 12.3, accompagnée du tableau 12.1 définit un modèle de Markov *caché* (HMM) à deux états.

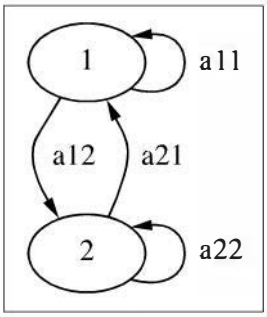


FIG. 12.3: *Le* HMM *à deux états.*

Copyright © 2010 Eyrolles.

Etat	1	2
P(O)	p_1	p_2
P(C)	$1-p_1$	$1-p_2$

TAB. 12.1: *Les probabilités d'émission du* HMM *à deux états.*

Sans entrer encore dans les détails, on voit qu'un HMM est d'abord caractérisé par une probabilité a_{ij} de passer d'un état à un autre, ensuite qu'à chaque état est associée une probabilité de générer O ou C. À chaque instant, il y a, non pas un, mais deux tirages aléatoires : le premier pour tirer une lettre de l'alphabet des observations, le second pour changer d'état. L'observation d'une séquence de O et de C n'est donc plus directement liée à une suite unique d'états. Par exemple, comme on le voit sur la figure 12.3, la séquence O C C peut être engendrée avec une certaine probabilité (on verra plus loin comment on la calcule) par la suite d'états 1 2 2 ou la suite 2 2 2. Dans le modèle présenté, n'importe quelle suite d'états peut en réalité engendrer n'importe quelle suite d'observations avec une certaine probabilité.

Cette différence peut apparaître inutilement subtile. En réalité, elle est très importante. Précisons l'exemple pour mesurer la différence de puissance de modélisation entre un modèle de Markov observable et un HMM. Rappelons que la probabilité pour le modèle de Markov observable d'engendrer une séquence de longueur $2n$ comportant autant de O que de C est exactement $p^n(1-p)^n$, indépendamment de la répartition des O et des C dans cette séquence.

Dans le cas du HMM, en prenant a_{11}, p_1, a_{22} et p_2 proches de 1, il est intéressant de constater que la phrase O O C C aura une forte probabilité d'être émise, alors que la phrase C C O O aura une probabilité faible. Pourtant, ces deux phrases comportent le même nombre de O et de C. D'une manière générale, une phrase ayant plus de O dans sa première moitié aura une probabilité plus forte que sa symétrique d'être émise par ce HMM. Cet exemple peut convaincre que si le HMM est plus complexe que le modèle observable, il a en retour la possibilité de représenter des concepts plus élaborés. En l'occurence, avec deux états, il est capable de représenter qu'il y a une différence entre les instants d'arrivée des oies et ceux des cygnes[6]. On verra le développement de cet exemple au paragraphe 6.

Remarquons aussi ceci : bien que l'alphabet des séquences observables soit composé de deux lettres, le HMM n'a plus de raison d'avoir exactement deux états. La figure 12.4, associée au tableau 12.2, présente un HMM à trois états. Les remarques sur le HMM à deux états sont encore valables : n'importe quelle suite d'états de ce HMM peut engendrer n'importe quelle suite d'observations de O et C avec une certaine probabilité. Ajoutons la remarque suivante : puisqu'on n'associe pas dans un HMM un état à une observation, il est possible de définir des observations appartenant à un alphabet infini.

2.3 Notations

Un HMM est noté $\lambda = (A, B, \pi)$ et se définit par :

- Ses états, en nombre n, qui composent l'ensemble $S = \{s_1, s_2, \ldots s_n\}$. L'état où se trouve le HMM à l'instant t est noté q_t ($q_t \in S$).

[6] Pour être complètement exact, un modèle observable pourrait aussi représenter une telle dépendance. Avec deux états, on peut en réalité représenter quatre probabilités différentes pour chaque séquence de deux observations (O O, O C), etc. et donc traduire une dépendance d'un événement avec l'événement précédent. En associant cette remarque à celle formulée précédemment en note de bas de page, on voit que le pouvoir d'expression des modèles observables peut être augmenté si on les sophistique... mais seulement sur des alphabets finis.

Copyright © 2010 Eyrolles.

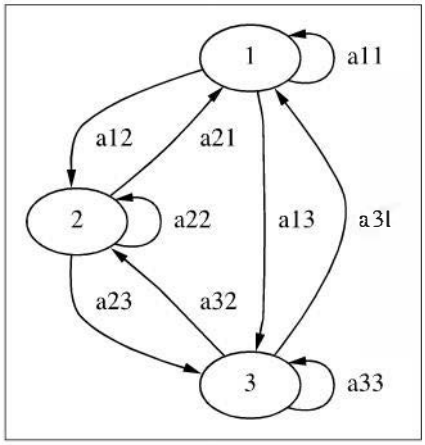


FIG. 12.4: *Le* HMM *à trois états.*

Etat	1	2	3
P(0)	p_1	p_2	p_3
P(C)	$1-p_1$	$1-p_2$	$1-p_3$

TAB. 12.2: *Les probabilités d'émission du* HMM *à trois états.*

- M symboles observables dans chaque état. L'ensemble des observations possibles (l'alphabet) est noté $V = \{v_1, v_2, \ldots, v_M\}$. $O_t \in V$ est le symbole observé à l'instant t.
- Une matrice A de *probabilités de transition* entre les états : a_{ij} représente la probabilité que le modèle évolue de l'état i vers l'état j :

$$a_{ij} = A(i,j) = \mathbf{P}(q_{t+1} = s_j \mid q_t = s_i) \ \forall i,j \in [1 \ldots n] \ \forall t \in [1 \ldots T]$$

avec :

$$a_{ij} \geq 0 \quad \forall i,j \qquad \text{et :} \qquad \sum_{j=1}^{n} a_{ij} = 1$$

- Une matrice B de *probabilités d'observation* des symboles dans chacun des états du modèle : $b_j(k)$ représente la probabilité que l'on observe le symbole v_k alors que le modèle se trouve dans l'état j, soit :

$$b_j(k) = \mathbf{P}(O_t = v_k \mid q_t = s_j) \quad 1 \leq j \leq n, \ 1 \leq k \leq M$$

avec :

$$b_j(k) \geq 0 \quad \forall j,k \qquad \text{et :} \qquad \sum_{k=1}^{M} b_j(k) = 1$$

- Un vecteur π de *probabilités initiales* : $\pi = \{\pi_i\}_{i=1,2,\ldots,n}$. Pour tout état i, π_i est la probabilité que l'état de départ du HMM soit l'état i :

$$\pi_i = \mathbf{P}(q_1 = s_i) \quad 1 \leq i \leq n$$

avec :

$$\pi_i \geq 0 \quad \forall i \qquad \text{et :} \qquad \sum_{i=1}^{n} \pi_i = 1$$

Copyright © 2010 Eyrolles.

- Un ou plusieurs *états finals*. Ici, nous supposons pour simplifier que le processus peut s'arrêter dans n'importe quel état, autrement dit que tout état est final.

2.4 Deux types de HMM

En pratique, on utilise deux types de modèles de Markov cachés, le modèle *ergodique* et le modèle *gauche-droite*.

Le modèle ergodique est sans contrainte : toutes les transitions d'un état vers un autre sont possibles. Les exemples présentés précédemment sont de ce type.

Le modèle gauche-droite est un modèle contenant des contraintes résultant de la mise à zéro de certaines valeurs a_{ij}. Dans le modèle le plus utilisé, celui de la figure 12.5, l'état i n'est relié par une transition de probabilité non nulle qu'à trois états : lui-même, l'état $i+1$ et l'état $i+2$. D'où le nom de *modèle gauche-droite*[7].

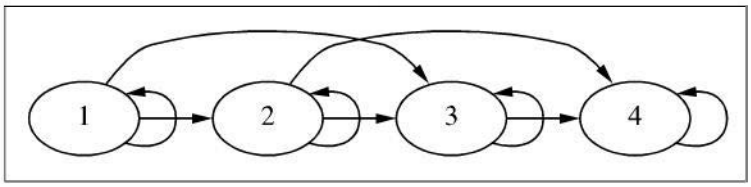


FIG. 12.5: *Le* HMM *gauche-droite à quatre états.*

2.5 Comment un HMM engendre-t'il une séquence ?

Un HMM peut être vu comme un processus permettant d'engendrer une séquence ; inversement, on peut considérer une séquence comme une suite d'observations sur un HMM en fonctionnement. En se plaçant du premier point de vue, la génération d'une séquence peut se décrire par l'algorithme 15 : c'est une procédure itérative gérée par des tirages aléatoires.

Algorithme 15 : Génération d'une séquence par un HMM

début
  $t \leftarrow 1$
  Choisir l'état initial $q_1 = s_i$ avec la probabilité π_i
  tant que $t \leq T$ **faire**
    Choisir l'observation $o_t = v_k$ avec la probabilité $b_i(k)$
    Passer à l'état suivant $q_{t+1} = s_j$ avec la probabilité a_{ij}
    $t \leftarrow t + 1$
  fin tant que
fin

Répétons ici qu'une séquence donnée peut en général être engendrée de plusieurs façons distinctes par un HMM.

[7] Ou modèle de Bakis. Dans l'exemple d'introduction, le HMM présenté est encore plus simple.

Copyright © 2010 Eyrolles.

3. Les HMM comme règles de classification de séquences

3.1 Les trois problèmes des HMM

Les définitions précédentes ne sont utilisables que si l'on sait calculer la probabilité qu'une séquence soit engendrée par un HMM et surtout si l'on sait apprendre un HMM à partir d'exemples. On doit donc chercher des algorithmes pour résoudre les problèmes suivants.

- *L'évaluation de la probabilité de l'observation d'une séquence.* Étant donné la séquence d'observations O et un HMM $\Lambda = (A, B, \pi)$, comment évaluer la probabilité d'observation $\mathbf{P}(O \mid \Lambda)$? La réponse à cette question est importante : dans un problème de classification, on attribuera à une séquence la classe que modélise le HMM le plus probable étant donnée la séquence.
- *La recherche du chemin le plus probable.* Étant donnés la suite d'observations O et un HMM Λ, comment trouver une suite d'états $Q = q_1, q_2, \ldots q_T$ qui maximise la probabilité d'observation de la séquence ?
- *L'apprentissage.* Comment ajuster les paramètres (A, B, π) d'un HMM Λ pour maximiser

$$\mathbf{P}(\mathcal{O} \mid \Lambda) = \prod_{O \in \mathcal{O}} \mathbf{P}(O \mid \Lambda)$$

à partir d'un ensemble $\mathcal{O}$ de séquences d'apprentissage?

Notons que la résolution du second problème n'est pas indispensable à l'utilisation des HMM en décision bayésienne. On reviendra sur son utilité au paragraphe 7.

3.2 Les HMM et la classification bayésienne

Le principe est d'apprendre un HMM par classe à partir des exemples de cette classe. L'apprentissage d'un HMM s'effectue à partir d'un modèle initial ; le HMM se modifie, mais en gardant jusqu'à sa convergence certaines caractéristiques du modèle initial (une certaine *architecture*) :

- le nombre d'états reste inchangé ;
- une transition de probabilité nulle entre deux états du modèle initial garde toujours une valeur nulle.

Le mieux est de prendre pour chaque classe un modèle initial ayant la même architecture : par exemple un modèle ergodique ou un modèle de Bakis. Pour chaque classe, le modèle initial peut simplement être pris avec le même nombre d'états[8].

Après C apprentissages indépendants, on dispose donc de C HMM, que l'on peut noter $\Lambda_1, \cdots, \Lambda_C$

Étant donnée une séquence quelconque O, on a pour la classe de rang k :

$$\mathbf{P}(\Lambda_k \mid O) = \frac{\mathbf{P}(O \mid \Lambda_k).\mathbf{P}(\Lambda_k)}{\mathbf{P}(O)}$$

Le modèle qui doit être choisi par la règle bayésienne est celui qui maximise $\mathbf{P}(\Lambda_k \mid O)$ (règle *MAP* : maximum *a posteriori*), ou si l'on suppose les classes équiprobables, celui qui maximise $\mathbf{P}(O \mid \Lambda_k)$ (maximum de vraisemblance), comme indiqué au chapitre 2.

On doit donc être capable de calculer cette dernière valeur pour tout i. Cela nécessite un algorithme capable d'évaluer la probabilité qu'une phrase soit émise par un HMM. C'est le sujet que nous allons développer au paragraphe suivant.

[8] Cette simplification n'est nullement nécessaire à l'application du principe *MAP*, mais elle est utilisée en l'absence de connaissances qui pourraient la mettre en cause.

Copyright © 2010 Eyrolles.

4. L'évaluation de la probabilité d'observation

L'évaluation directe

Remarquons d'abord que la probabilité de la suite d'observations O, étant donné le modèle Λ, est égale à la somme sur toutes les suites d'états possibles Q des probabilités conjointes de O et de Q :

$$\mathbf{P}(O \mid \Lambda) = \sum_{Q} \mathbf{P}(O, Q \mid \Lambda) = \sum_{Q} \mathbf{P}(O \mid Q, \Lambda)\mathbf{P}(Q \mid \Lambda)$$

Or, on a les relations :

$$\mathbf{P}(Q \mid \Lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T}$$

$$\mathbf{P}(O \mid Q, \Lambda) = b_{q_1}(O_1) b_{q_2}(O_2) \dots b_{q_T}(O_T)$$

On déduit donc des formules précédentes, en réarrangeant les termes :

$$\mathbf{P}(O \mid \Lambda) = \sum_{Q=q_1,q_2,\dots,q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

Cette formule directe nécessite d'énumérer toutes les suites d'états de longueur T, soit une complexité en $\Theta(n^T)$. Il existe heureusement une méthode plus rapide.

L'évaluation par les fonctions forward-backward.

Dans cette approche [Bau72], on remarque que l'observation peut se faire en deux temps : d'abord, l'émission du début de l'observation $O(1:t)$ en aboutissant à l'état q_i au temps t, puis, l'émission de la fin de l'observation $O(t+1:T)$ sachant que l'on part de q_i au temps t. Ceci posé, la probabilité de l'observation est donc égale à :

$$\mathbf{P}(O \mid \Lambda) = \sum_{i=1}^{n} \alpha_t(i)\beta_t(i)$$

où $\alpha_t(i)$ est la probabilité d'émettre le début $O(1:t)$ et d'aboutir à q_i à l'instant t, et $\beta_t(i)$ est la probabilité d'émettre la fin $O(t+1:T)$ sachant que l'on part de q_i à l'instant t. Le calcul de α s'effectue avec t croissant tandis que le calcul de β est réalisé avec t décroissant, d'où l'appellation *forward-backward*.

Le calcul de α

On a :

$$\alpha_t(i) = \mathbf{P}(O_1 O_2 \dots O_t,\ q_t = s_i \mid \Lambda)$$

$\alpha_t(i)$ se calcule par l'algorithme 16, qui exprime que pour émettre le début de l'observation $O(1:t+1)$ et aboutir dans l'état s_j au temps $t+1$, on doit nécessairement être dans l'un des états s_i à l'instant t. Cette remarque permet d'exprimer $\alpha_{t+1}(j)$ en fonction des $\alpha_t(i)$ et d'utiliser un algorithme de programmation dynamique pour le calcul de tous les $\alpha_t(i)$ pour tout i, puis des $\alpha_{t+1}(i)$ pour tout i, etc.

Ce calcul a une complexité en $\Theta(n^2 T)$.

Le calcul de β

De manière analogue, $\beta_t(i)$ se calcule par l'algorithme 17.

Copyright © 2010 Eyrolles.

Algorithme 16 : Calcul de la fonction *forward* α

début
  pour $i = 1, n$ **faire** $\alpha_1(i) \leftarrow \pi_i b_i(O_1)$
  $t \leftarrow 1$
  tant que $t < T$ **faire**
    $j \leftarrow 1$
    tant que $j \leq n$ **faire**
      $\alpha_{t+1}(j) \leftarrow [\sum_{i=1}^{n} \alpha_t(i) a_{ij}] b_j(O_{t+1})$
      $j \leftarrow j + 1$
    fin tant que
    $t \leftarrow t + 1$
  fin tant que
  $\mathsf{P}(O \mid \Lambda) \leftarrow \sum_{i=1}^{n} \alpha_T(i)$
fin

Algorithme 17 : Calcul de la fonction *backward* β

début
  pour $i = 1, n$ **faire** $\beta_T(i) \leftarrow 1$
  $t \leftarrow T - 1$
  tant que $t \geq 1$ **faire**
    $i \leftarrow 1$
    tant que $i \leq n$ **faire**
      $\beta_t(i) \leftarrow \sum_{j=1}^{n} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$
      $i \leftarrow i + 1$
    fin tant que
    $t \leftarrow t - 1$
  fin tant que
  $\mathsf{P}(O \mid \Lambda) \leftarrow \sum_{i=1}^{n} \beta_1(i)$
fin

Le calcul de β est lui aussi en $\mathcal{O}(n^2 T)$.

Le calcul de la probabilité d'observation

Finalement, la probabilité d'observation d'une séquence est obtenue en prenant les valeurs de α et de β à un instant t quelconque : $\mathsf{P}(O \mid \Lambda) = \sum_{i=1}^{n} \alpha_t(i)\beta_t(i)$. Cependant, on utilise le plus souvent les valeurs obtenues pour deux cas particuliers ($t = 0$) ou ($t = T$), ce qui donne :

$$\mathsf{P}(O \mid \Lambda) = \sum_{i=1}^{n} \alpha_T(i) = \sum_{i=1}^{n} \pi_i \beta_0(i)$$

EXEMPLE

Soit le modèle $\Lambda = (A, B, \pi)$ (figure 12.6) comprenant trois états 1, 2, 3 chacun permettant d'observer un symbole de l'alphabet $V = \{a, b\}$.

$$A = \begin{pmatrix} 0.3 & 0.5 & 0.2 \\ 0 & 0.3 & 0.7 \\ 0 & 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 \\ 0.5 & 0.5 \\ 0 & 1 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.6 \\ 0.4 \\ 0 \end{pmatrix}$$

Copyright © 2010 Eyrolles.

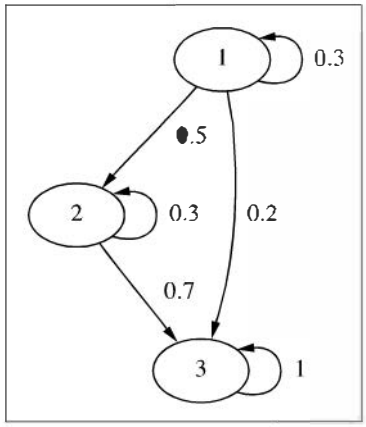


FIG. 12.6: *Un exemple de* HMM.

Etat	1	2	3
P(a)	1	0.5	0
P(b)	0	0.5	1

TAB. 12.3: *La matrice B de ce* HMM.

La figure 12.7 illustre le calcul de α pour la suite d'observations : *a a b b*.

$$
\begin{aligned}
\alpha_1(1) &= \pi_1 b_1(a) = 0.6 \times 1 = 0.6 \\
\alpha_1(2) &= \pi_2 b_2(a) = 0.4 \times 0.5 = 0.2 \\
\alpha_1(3) &= \pi_3 b_3(a) = 0 \times 0 = 0 \\
\alpha_2(1) &= (\alpha_1(1)a_{11} + \alpha_1(2)a_{21} + \alpha_1(3)a_{31})b_1(a) \\
&= (0.6 \times 0.3 + 0.2 \times 0 + 0 \times 0) \times 1 \\
&= (0.18) \times 1 = 0.18 \\
\alpha_2(2) &= (\alpha_1(1)a_{12} + \alpha_1(2)a_{22} + \alpha_1(3)a_{32})b_2(a) \\
&= (0.6 \times 0.5 + 0.2 \times 0.3 + 0 \times 0) \times 0.5 \\
&= (0.36) \times 0.5 = 0.18 \\
\ldots & \quad \ldots \\
\mathbf{P}(a\ a\ b\ b \mid \Lambda) &= \sum_{q_i} \alpha_4(i) = 0.2228
\end{aligned}
$$

5. Le calcul du chemin optimal : l'algorithme de Viterbi

Il s'agit maintenant de déterminer le meilleur chemin correspondant à l'observation, c'est-à-dire de trouver dans le modèle Λ la *meilleure suite d'états* Q, qui maximise la quantité :

$$\mathbf{P}(Q, O \mid \Lambda)$$

Pour trouver $Q = (q_1, q_2, \ldots, q_T)$ pour une séquence d'observations $O = (O_1, O_2, \ldots, O_T)$, on définit la variable intermédiaire $\delta_t(i)$ comme la probabilité du meilleur chemin amenant à l'état s_i à l'instant t, en étant guidé par les t premières observations :

$$\delta_t(i) = \underset{q_1,\ldots,q_{t-1}}{Max}\ \mathbf{P}(q_1, q_2, \ldots, q_t = s_i, O_1, O_2, \ldots, O_t \mid \Lambda)$$

Copyright © 2010 Eyrolles.

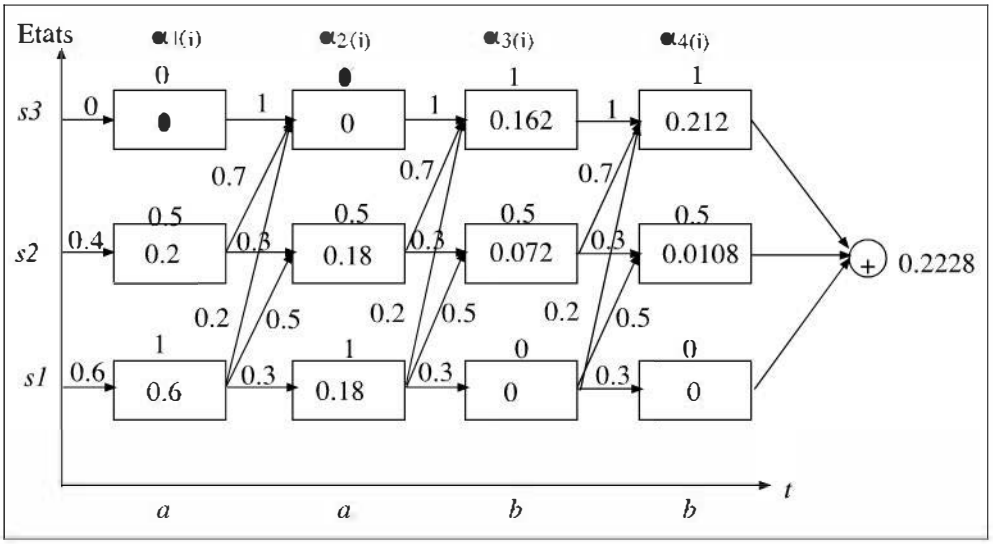


FIG. 12.7: *Calcul de α pour la suite d'observations aabb.*

Par récurrence, on calcule

$$\delta_{t+1}(j) = [\underset{i}{\text{Max}}\ \delta_t(i)\, a_{ij}]\, b_j(O_{t+1})$$

en gardant trace, lors du calcul, de la suite d'états qui donne le meilleur chemin amenant à l'état s_i à t dans un tableau ψ.

On utilise une variante de la programmation dynamique, l'*algorithme de Viterbi* (algorithme 18) pour formaliser cette récurrence. Il fournit en sortie la valeur $\mathbf{P}^*$ de la probabilité de l'émission de la séquence par la meilleure suite d'états $(q_1^*, \cdots, q_T^*)$.

La fonction *Argmax* permet de mémoriser l'indice i, entre 1 et n, avec lequel on atteint le maximum des quantités $(\delta_{t-1}(i)a_{ij})$. Le coût des opérations est également en $\bullet(n^2T)$.

EXEMPLE

D'après [BB92].

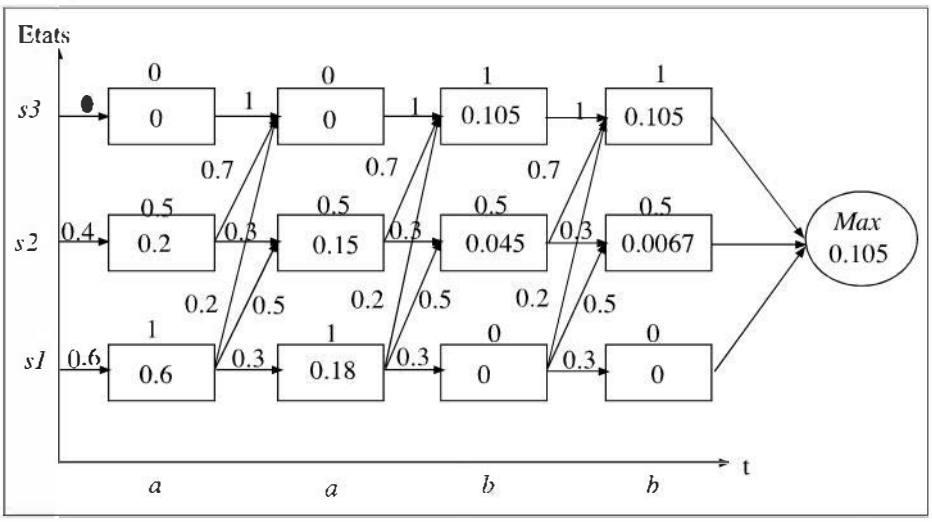


FIG. 12.8: *Calcul de δ pour la suite d'observations aabb.*

Copyright © 2010 Eyrolles.

Algorithme 18 : Algorithme de Viterbi

début
  pour $i = 1, n$ **faire**
    $\delta_1(i) \leftarrow \pi_i b_i(O_1)$
    $\psi_1(i) \leftarrow 0$
  fin pour
  $t \leftarrow 2$
  tant que $t \leq T - 1$ **faire**
    $j \leftarrow 1$
    tant que $j \leq n$ **faire**
      $\delta_t(j) \leftarrow \text{Max}_{1 \leq i \leq n} [\delta_{t-1}(i)\, a_{ij}]\, b_j(O_t)$
      $\psi_t(j) \leftarrow \text{ArgMax}_{1 \leq i \leq n} [\delta_{t-1}(i)\, a_{ij}]$
      $j \leftarrow j + 1$
    fin tant que
    $t \leftarrow t + 1$
  fin tant que
  $\mathbf{P}^* \leftarrow \text{Max}_{1 \leq i \leq n} [\delta_T(i)]$
  $q_T^* \leftarrow \text{ArgMax}_{1 \leq i \leq n} [\delta_T(i)]$
  $t \leftarrow T$
  tant que $t \geq 1$ **faire**
    $q_t^* \leftarrow \psi_{t+1}(q_{t+1}^*)$
    $t \leftarrow t - 1$
  fin tant que
fin

À partir de la figure 12.8 qui illustre le calcul de δ, on peut calculer les quantités δ, ψ et q^* comme suit :

$$\begin{array}{lll}
\delta_1(1) & = \pi_1 b_1(a) = 0.6 \times 1 = 0.6 & \psi_1(1) = 0, \\
\delta_1(2) & = \pi_2 b_2(a) = 0.4 \times 0.5 = 0.2 & \psi_1(2) = 0, \\
\delta_1(3) & = \pi_3 b_3(a) = 0 \times 0 = 0 & \psi_1(3) = 0, \\
\delta_2(1) & = max_{1 \leq i \leq n}(\delta_1(i) a_{i1}) b_1(a) & \\
 & = max \left\{ \begin{array}{l} \delta_1(1) a_{11} \\ \delta_1(2) a_{21} \\ \delta_1(3) a_{31} \end{array} \right\} \times b_1(a) & \\
 & = max \left\{ \begin{array}{l} \underline{0.6 \times 0.3} \\ 0 \\ 0 \end{array} \right\} \times 1 = 0.18 & \psi_2(1) = 1,
\end{array}$$

...

Finalement :

$$\begin{array}{llll}
\underline{\psi_1(1) = 0} & \psi_2(1) = 1 & \psi_3(1) = 1 & \psi_4(1) = 1 \\
\psi_1(2) = 0 & \underline{\psi_2(2) = 1} & \psi_3(2) = 1 & \psi_4(2) = 2 \\
\psi_1(3) = 0 & \psi_2(3) = 2 & \underline{\psi_3(3) = 2} & \underline{\psi_4(3) = 3},
\end{array}$$

Copyright © 2010 Eyrolles.

$$q_4^* = max \begin{pmatrix} \delta_4(1) \\ \delta_4(2) \\ \delta_4(3) \end{pmatrix} = 3,$$

$$\begin{aligned} q_3^* &= \psi_4(3) = 3, \\ q_2^* &= \psi_3(3) = 2, \\ q_1^* &= \psi_2(2) = 1. \end{aligned}$$

On déduit donc de ce calcul que la *meilleure suite d'états*, celle qui engendre la phrase *a a b b* avec la plus forte probabilité, est : *1 2 3 3*.

6. L'apprentissage

Principe

Supposons disposer d'un ensemble de séquences $\mathcal{O} = \{\mathcal{O}^1, \cdots, \mathcal{O}^m\}$, dont l'élément courant est noté O^k. Le but de l'apprentissage est de déterminer les paramètres d'un HMM d'architecture fixée $\Lambda = (A, B, \pi)$, qui maximisent la probabilité $\mathbf{P}(\mathcal{O} \mid \Lambda)$. Comme on suppose les séquences d'apprentissages tirées indépendamment, on cherche donc à maximiser :

$$\mathbf{P}(\mathcal{O} \mid \Lambda) = \prod_{k=1}^{m} \mathbf{P}(O^k \mid \Lambda)$$

L'idée est d'utiliser une procédure de réestimation qui affine le modèle petit à petit selon les étapes suivantes :

- choisir un ensemble initial Λ_0 de paramètres ;
- calculer Λ_1 à partir de Λ_0, puis Λ_2 à partir de Λ_1, etc.
- répéter ce processus jusqu'à un critère de fin.

Pour chaque étape p d'apprentissage, on dispose de Λ_p et on cherche un Λ_{p+1} qui doit vérifier :

$$\mathbf{P}(\mathcal{O} \mid \Lambda_{p+1}) \geq \mathbf{P}(\mathcal{O} \mid \Lambda_p)$$

soit :

$$\prod_{k=1}^{m} \mathbf{P}(O^k \mid \Lambda_{p+1}) \geq \prod_{k=1}^{m} \mathbf{P}(O^k \mid \Lambda_p)$$

Λ_{p+1} doit donc améliorer la probabilité de l'émission des observations de l'ensemble d'apprentissage. La technique pour calculer Λ_{p+1} à partir de Λ_p consiste à utiliser l'algorithme EM. Pour cela, on effectue un comptage de l'utilisation des transitions A et des distributions B et π du modèle Λ_p quand il produit l'ensemble $\mathcal{O}$. Si cet ensemble est assez important, ces fréquences fournissent de bonnes approximations *a posteriori* des distributions de probabilités A, B et π et sont utilisables alors comme paramètres du modèle Λ_{p+1} pour l'itération suivante.

La méthode d'apprentissage EM consiste donc dans ce cas à regarder comment se comporte le modèle défini par Λ_p sur $\mathcal{O}$, à réestimer ses paramètres à partir des mesures prises sur $\mathcal{O}$, puis à recommencer cette réestimation jusqu'à obtenir une convergence. L'annexe 7 donne quelques détails sur cette méthode.

Copyright © 2010 Eyrolles.

Dans les calculs qui suivent, on verra apparaître en indice supérieur la lettre k quand il faudra faire référence à la séquence d'apprentissage concernée. L'indice p, qui compte les passes d'apprentissage, sera omis : on partira d'un modèle noté simplement Λ et on calculera celui qui s'en déduit.

Les formules de réestimation

On définit $\xi_t^k(i,j)$ comme la probabilité, étant donnés une phrase O^k et un HMM Λ, que ce soit l'état s_i qui ait émis la lettre de rang t de O^k et l'état s_j qui ait émis celle de rang $t+1$. Donc :

$$\xi_t^k(i,j) = \mathbf{P}(q_t = s_i,\ q_{t+1} = s_j \mid O^k, \Lambda)$$

Ce qui se récrit :

$$\xi_t^k(i,j) = \frac{\mathbf{P}(q_t = s_i,\ q_{t+1} = s_j,\ O^k \mid \Lambda)}{\mathbf{P}(O^k \mid \Lambda)}$$

Par définition des fonctions *forward-backward*, on en déduit :

$$\xi_t^k(i,j) = \frac{\alpha_t^k(i) a_{ij} b_j(O_{t+1}^k) \beta_{t+1}^k(j)}{\mathbf{P}(O^k \mid \Lambda)}$$

On définit aussi la quantité $\gamma_t^k(i)$ comme la probabilité que la lettre de rang t de la phrase O^k soit émise par l'état s_j.

$$\gamma_t^k(i) = \mathbf{P}(q_t = s_i \mid O^k, \Lambda)$$

Soit :

$$\gamma_t^k(i) = \sum_{j=1}^{n} \mathbf{P}(q_t = s_i,\ q_{t+1} = s_j \mid O^k, \Lambda) = \frac{\sum_{j=1}^{n} \mathbf{P}(q_t = s_i,\ q_{t+1} = s_j, O^k \mid \Lambda)}{\mathbf{P}(O^k \mid \Lambda)}$$

On a la relation :

$$\gamma_t^k(i) = \sum_{j=1}^{n} \xi_t(i,j) \frac{\alpha_t^k(i) \beta_t^k(i)}{\mathbf{P}(O^k \mid \Lambda)}$$

Le nouveau modèle HMM se calcule à partir de l'ancien en réestimant π, A et B par comptage sur la base d'apprentissage. On mesure les fréquences :

$$\overline{a}_{ij} = \frac{\text{nombre de fois où la transition de } s_i \text{ à } s_j \text{ a été utilisée}}{\text{nombre de transitions effectuées à partir de } s_i}$$

$$\overline{b}_j(l) = \frac{\text{nombre de fois où le HMM s'est trouvé dans l'état } s_j \text{ en observant } v_l}{\text{nombre de fois où le HMM s'est trouvé dans l'état } s_j}$$

$$\overline{\pi}_i = \frac{\text{nombre de fois où le HMM s'est trouvé dans l'état } s_i \ \ldots}{\text{nombre de fois où le HMM } \ldots}$$

$$\frac{\ldots \text{ en émettant le premier symbole d'une phrase}}{\ldots \text{ a émis le premier symbole d'une phrase}}$$

Compte tenu de ces définitions :

$$\overline{\pi}_i = \frac{1}{m} \sum_{k=1}^{m} \gamma_1^k(i)$$

Copyright © 2010 Eyrolles.

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{m} \sum_{t=1}^{|O^k|-1} \xi_t^k(i,j)}{\sum_{k=1}^{N} \sum_{t=1}^{|O^k|-1} \gamma_t^k(i)}$$

$$\overline{b}_j(l) = \frac{\sum_{k=1}^{m} \sum_{t=1 \text{ avec } O_t^k = v_l}^{|O^k|-1} \gamma_t^k(j)}{\sum_{k=1}^{m} \sum_{t=1}^{|O^k|-1} \gamma_t^k(j)}$$

Ces formules ont été établies par Baum [Bau72], comme une application de la procédure EM à l'apprentissage des paramètres HMM. La suite des modèles construits par l'algorithme de Baum-Welsh [RJ93] vérifie la relation cherchée :

$$\mathbf{P}(\mathcal{O} \mid \Lambda_{p+1}) \geq \mathbf{P}(\mathcal{O} \mid \Lambda_p)$$

Algorithme 19 : Algorithme de Baum-Welch

début
  Fixer des valeurs initiales (A, B, π)
  On définit le HMM de départ comme $\Lambda_0 = (A, B, \pi)$.
  $p \leftarrow 0$
  tant que *la convergence n'est pas réalisée* **faire**
    On possède le HMM Λ_p
    On calcule pour ce modèle, sur l'ensemble d'apprentissage, les valeurs :
      $\xi(i,j)$, $\gamma_t(i)$ $\quad 1 \leq i,j \leq n \qquad 1 \leq t \leq T-1$
    On en déduit $\overline{\pi}$, $\overline{A}$, $\overline{B}$ en utilisant les formules de réestimation.
    Le HMM courant est désormais défini par $\Lambda_{p+1} = (\overline{\pi}, \overline{A}, \overline{B})$
    $p \leftarrow p+1$
  fin tant que
fin

Remarques

- Le choix du modèle initial influe sur les résultats ; par exemple, si certaines valeurs de A et B sont égales à 0 au départ, elles le resteront jusqu'à la fin de l'apprentissage. Ceci permet en particulier de garder la structure dans les modèles gauches-droits.
- L'algorithme converge vers des valeurs de paramètres qui assurent un *maximum local* de $\mathbf{P}(\bullet \mid \Lambda)$. Il est donc important, si l'on veut être aussi près que possible du minimum global, de bien choisir la structure et l'initialisation.
- Le nombre d'itérations est fixé empiriquement. L'expérience prouve que, si le point précédent a été correctement traité, la stabilisation des paramètres ne correspond pas à un surapprentissage : il n'y a donc en général pas besoin de contrôler la convergence par un ensemble de validation. Mais cette possibilité est évidemment toujours à disposition.

Copyright © 2010 Eyrolles.

EXEMPLE

En partant du HMM λ_0 défini par les paramètres suivants :

$$A = \begin{pmatrix} 0.45 & 0.35 & 0.20 \\ 0.10 & 0.50 & 0.40 \\ 0.15 & 0.25 & 0.60 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 0.5 & 0.5 \\ 0.0 & 1.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.5 \\ 0.3 \\ 0.2 \end{pmatrix}$$

on peut calculer que, s'il émet sur l'alphabet à deux lettres $V = \{a, b\}$, on a :

$$\mathsf{P}(a\ b\ b\ a\ a \mid \lambda_0) = 0.0278$$

Si on prend comme ensemble d'apprentissage cette seule phrase, l'application de l'algorithme de Baum-Welsh doit augmenter sa probabilité de reconnaissance.
Après une réestimation[9], on trouve le HMM λ_1 :

$$A = \begin{pmatrix} 0.346 & 0.365 & 0.289 \\ 0.159 & 0.514 & 0.327 \\ 0.377 & 0.259 & 0.364 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 0.631 & 0.369 \\ 0.0 & 1.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.656 \\ 0.344 \\ 0.0 \end{pmatrix}$$

$$\mathsf{P}(a\ b\ b\ a\ a \mid \lambda_1) = 0.0529$$

Après quinze itérations :

$$A = \begin{pmatrix} 0.0 & 0.0 & 1.0 \\ 0.212 & 0.788 & 0.0 \\ 0.0 & 0.515 & 0.485 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 0.969 & 0.031 \\ 0.0 & 1.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \end{pmatrix}$$

$$\mathsf{P}(a\ b\ b\ a\ a \mid \lambda_{15}) = 0.2474$$

Après cent cinquante itérations, la convergence est réalisée. La figure 12.9 et le tableau 12.4 décrivent le résultat, que l'on peut donner aussi sous la forme suivante :

$$A = \begin{pmatrix} 0.0 & 0.0 & 1.0 \\ 0.18 & 0.82 & 0.0 \\ 0.0 & 0.5 & 0.5 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \end{pmatrix}$$

$$\mathsf{P}(a\ b\ b\ a\ a \mid \lambda_{150}) = 0.2500$$

Etat	1	2	3
P(a)	1	1	0
P(b)	0	0	1

TAB. 12.4: *La matrice B de ce* HMM.

Il peut paraître curieux que ce HMM ne génère pas son unique phrase d'apprentissage avec la probabilité 1 et toutes les autres séquences avec la probabilité 0. Ceci vient du fait que le nombre des états est trop petit pour réaliser un apprentissage par cœur. Mais si l'on part d'un HMM initial à cinq états, il converge en revanche vers un HMM λ défini par :

[9] Les calculs sont très pénibles à faire à la main, même sur un exemple simple comme celui-ci.

Copyright © 2010 Eyrolles.

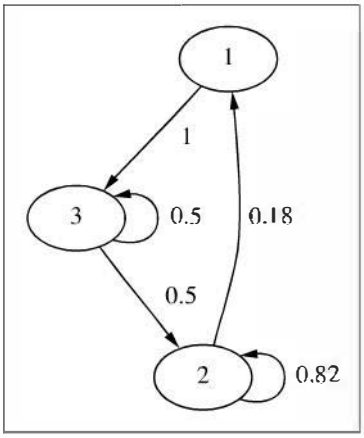


FIG. 12.9: *Le* HMM *entraîné sur une seule phrase, après convergence. Le seul état initial possible est l'état 1.*

$$A = \begin{pmatrix} 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 1.0 & 0.0 \\ 1.0 & 0.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$$

et l'on a :

$$\mathbf{P}(a\ b\ b\ a\ a \mid \lambda) = 1.0$$

Ce HMM est en réalité une variante d'un modèle observable : à chaque état est associée l'émission certaine d'une lettre. La différence avec les modèles présentés en début de ce chapitre est que la même lettre peut être associée à plusieurs états. Ici, par exemple, la lettre a aux états 1, 4 et 5.

EXEMPLE

Reprenons maintenant l'exemple du paragraphe 2.2. Nous partons du HMM suivant, dont les paramètres ont été tirés aléatoirement :

$$A = \begin{pmatrix} 0.40 & 0.60 \\ 0.52 & 0.48 \end{pmatrix} \quad B = \begin{pmatrix} 0.49 & 0.51 \\ 0.40 & .60 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.31 \\ 0.69 \end{pmatrix}$$

Nous lui faisons subir deux apprentissages sur deux ensembles de séquences : le premier est composé d'éléments ayant à peu près autant de a que de b, ces derniers étant situés en majorité dans la seconde partie ; l'autre ensemble d'apprentissage est composé de phrases de type symétrique.

$$\mathcal{O}_1 = \{aaabb, abaabbb, aaababb, aabab, ab\}$$

$$\mathcal{O}_2 = \{bbbaa, babbaa, bbbabaa, bbabba, bbaa\}$$

Après convergence, on obtient deux HMM différents donnés sur la figure 12.10.

A_1 est défini par :

$$A = \begin{pmatrix} 1.0 & 0.0 \\ 0.69 & 0.31 \end{pmatrix} \quad B = \begin{pmatrix} 0.36 & 0.64 \\ 1.0 & 0.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix}$$

Copyright © 2010 Eyrolles.

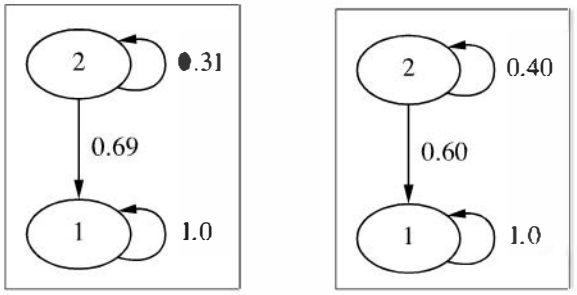


FIG. 12.10: *Les* HMM Λ_1 *et* Λ_2.

Son unique état de départ est l'état 2 qui émet a avec la probabilité 1. La probabilité d'émettre b par l'état 1 est de 0.64.

Λ_2 est défini par :

$$A = \begin{pmatrix} 1.0 & 0.0 \\ 0.60 & 0.40 \end{pmatrix} \quad B = \begin{pmatrix} 0.65 & 0.34 \\ 0.0 & 1.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix}$$

Son unique état de départ est l'état 2 qui émet b avec la probabilité 1. La probabilité d'émettre a par l'état 1 est de 0.65.

Les deux HMM appris sont dont assez semblables, quand on permute a et b. On remarque qu'ils ne font pas jouer un rôle symétrique à leurs états.

Sur deux phrases n'ayant pas participé à l'apprentissage, on obtient le résultat attendu :

$$\mathbf{P}(a\ a\ b\ a\ b\ b\ b \mid \Lambda_1) = 0.0437$$

$$\mathbf{P}(a\ a\ b\ a\ b\ b\ b \mid \Lambda_2) = 0.000$$

$$\mathbf{P}(b\ b\ a\ b\ a\ a\ a \mid \Lambda_1) = 0.000$$

$$\mathbf{P}(b\ b\ a\ b\ a\ a\ a \mid \Lambda_2) = 0.0434$$

7. Approfondissements

Comme on l'a dit plus haut, il est possible de définir des HMM produisant des séquences de valeurs continues et même des séquences de vecteurs de valeurs continues. Dans ce cas, elles ne sont évidemment plus construites sur un alphabet fini. Il faut alors remplacer la matrice B par un ensemble de distributions éventuellement multidimensionnelles de probabilités ; les calculs précédents restent valables, mais les formules, en particulier celles de la réestimation pour l'apprentissage doivent porter sur des paramètres caractéristiques des distributions de probabilité en question. Généralement, on suppose celles-ci gaussiennes, ou multigaussiennes.

C'est souvent le cas en reconnaissance de la parole : prenons l'exemple de la reconnaissance du vocabulaire des dix chiffres. L'échantillon d'apprentissage permet de créer dix ensembles d'exemples, chacun supervisé par un chiffre différent. Pour chaque chiffre, on va apprendre un HMM. En phase de reconnaissance, un son inconnu sera classé comme le chiffre associé au HMM qui peut l'émettre avec la plus forte probabilité. Qu'est-ce qu'une séquence représentant la prononciation d'un son? Au départ, un signal échantillonné, soit huit ou seize mille valeurs réelles

Copyright © 2010 Eyrolles.

par seconde. Ce signal est transformé par des techniques de type transformation de Fourier pour en extraire ses caractéristiques fréquencielles. Au final, on dispose en général d'un codage de chaque centième de seconde de parole émise par un vecteur d'une dizaine de valeurs réelles. La prononciation d'un chiffre est donc une séquence dont la longueur est de l'ordre de quelques dizaines et dont chaque élément est un vecteur de $\mathbb{R}^{10}$.

Il est dès lors impossible de définir la matrice B puisque dans chaque terme $b_j(k)$, k devrait parcourir tous les vecteurs différents représentant un centième de seconde de parole et donc prendre un nombre infini de valeurs. Le plus simple est d'effectuer une estimation paramétrique de la distribution des composantes du vecteur émis par chaque état, comme au chapitre 15. On supposera par exemple que ce sont des tirages aléatoires d'une gaussienne de dimension 10 : dans ce cas, pour chaque état, il faudra estimer la moyenne et la covariance de la densité de probabilité d'émission d'un vecteur de dimension 10. L'algorithme de Baum-Welsh s'adapte facilement à cette situation.

Des approfondissements et des alternatives ont été proposés : il est commun de supposer que les vecteurs émis sont des *sommes pondérées* de plusieurs distributions gaussiennes multidimensionnelles. L'algorithme *EM* permet encore de calculer les moyennes et les covariances de chacune, ainsi que les coefficients de pondération (voir le chapitre 18). Il a été aussi proposé d'estimer les densités d'émission par la méthode non paramétrique des k plus proches voisins qui est présenté au chapitre 15 ou par des réseaux connexionnistes (chapitre 10).

8. Applications

Les HMM sont actuellement les outils d'apprentissage les plus efficaces pour la classification des séquences : ils ne réclament que peu de connaissances *a priori*; à condition de disposer de suffisamment de données d'apprentissage, ils sont très efficaces. Un grand nombre de raffinements leur ont été apportés, en particulier pour résoudre des problèmes aussi complexes que celui de la reconnaissance de la parole ou de l'écriture manuscrite. Ces outils sont également très employés dans les séquences biologiques, en particulier pour la prédiction des structures secondaires et tri-dimensionnelles des protéines. On les utilise aussi en fouille de données, pour la recherche approchée de séquences dans des textes ou dans des masses de données de bio-séquences.

À l'heure actuelle, par exemple, presque tous les systèmes de reconnaissance de la parole sont construits à base de HMM, parfois hybridés de réseaux connexionnistes. Dans la plupart des cas, les informations phonétiques, lexicales et syntaxiques sont « compilées » dans un HMM de plusieurs centaines d'états, dont chacun vise à posséder une signification linguistique ; l'apprentissage se fait sur des très grandes quantités de données. La reconnaissance est effectuée en récupérant la séquence d'états par l'algorithme de Viterbi. En effet, les états ont dans ce cas une signification linguistique. Ce n'est pas tant la probabilité finale qui est intéressante que le chemin par lequel passe la meilleure façon d'engendrer le signal.

Notes historiques et sources bibliographiques

L'algorithme de Baum-Welsh [Bau72] est une application aux modèles de Markov de la technique générale *EM* d'estimation statistique de paramètres cachés. L'annexe 7 qui traite de cette

Copyright © 2010 Eyrolles.

technique est en partie reprise du document [Ros97]. L'utilisation en apprentissage vient de la communauté de la reconnaissance de la parole. Les premières références que l'on peut relier à cette technique datent des années 1970 [Jel76, Bak75]. L'utilisation en reconnaissance bayésienne pour des séquences, comme présenté dans ce chapitre, est surtout fondée sur les travaux de Rabiner et de son équipe, qui ont démarré vers 1983 [LRS83]. L'article [Rab89] est un exposé incontournable sur les HMM et leur application à la parole. Les ouvrages [Jel97] et surtout [RJ93, JM00] et donnent un panorama de l'utilisation des HMM en reconnaissance de la parole. Pour leurs applications au traitement des séquences biologiques et des images, des références intéressantes sont [DEKM98, BB01] et [BS97, MS01]. L'exemple 12.6 et les figures associées sont empruntés, avec l'aimable accord des auteurs, à [BB92]. Ce dernier ouvrage (entre autres mérites) présente une bonne introduction à ces techniques d'apprentissage et à leur application pour la reconnaissance de l'écriture manuscrite ; ce chapitre lui a fait quelques autres emprunts.

Résumé

- Les HMM sont des modèles probabilistes d'émission de séquences, discrètes ou continues (et dans ce cas, éventuellement vectorielles).
- En version de base, ils sont utilisés en classification bayésienne.
- L'algorithme *forward-backward* permet de connaître la probabilité qu'un HMM ait émis une séquence.
- L'algorithme de *Viterbi* permet de connaître la suite des états du HMM qui a la plus forte probabilité d'avoir émis une séquence.
- L'algorithme de *Baum-Welsh* permet d'ajuster les paramètres d'un HMM au maximum de vraisemblance à partir d'un ensemble de séquences d'apprentissage.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 13

Apprentissage par inférence d'arbres

*L'une des grandes familles d'approches pour la résolution de problèmes comme pour l'apprentissage est la technique consistant à « diviser pour régner » (*divide and conquer*). Elle se résume à identifier des sous-problèmes, à leur trouver une solution, puis à combiner ces solutions pour résoudre le problème général.*

C'est sur ce principe que sont fondés les algorithmes d'apprentissage par arbres de décision. Ils apprennent à identifier les sous-espaces de l'espace d'entrée pour lesquels la solution est identique. Lorsqu'un nouveau cas est soumis au système, celui-ci identifie le sous-espace correspondant et retourne la réponse associée.

Ces méthodes sont très versatiles, non paramétriques et généralement efficaces en termes calculatoires. Elles constituent un outil de base en apprentissage supervisé.

Copyright © 2010 Eyrolles.

Sommaire

1 Les arbres de décision 409
1.1 Principe 409
1.2 La construction récursive d'un arbre de décision 411
1.2.1 Introduction 411
1.2.2 Le cas des attributs binaires 412
1.2.3 Le cas des attributs non binaires 416
1.3 Comment élaguer un arbre trop précis ? 417
1.3.1 Pourquoi élaguer ? 417
1.3.2 Une première solution : le préélagage 417
1.3.3 Le post-élagage par un ensemble indépendant de validation . . 418
1.3.4 L'élagage par validation croisée 421
1.4 Les arbres de décision et la logique des propositions 422
2 Les arbres de régression 423
2.1 Le principe 423
2.2 La construction 424
2.3 La fin de la construction et l'élagage 425

Les manuels d'ornithologie et les « flores » (les livres pour l'identification des plantes à fleurs) ne sont en général pas organisés de la même manière. Pour les premiers, on trouve les oiseaux dans un ordre invariable, correspondant à l'énumération savante des ordres et des espèces. Un dessin de chaque oiseau est donné, accompagné d'une description imagée et de détails permettant de différencier l'espèce en question avec celles qui peuvent prêter à confusion. Par exemple, si on observe un grand oiseau blanc sur un plan d'eau, un rapide parcours des figures amène sans ambiguïté à la page des cygnes, où seulement trois espèces sont décrites[1]. Il reste à se décider avec le texte et les détails des dessins, en lisant par exemple : « Au repos, le cygne tuberculé a l'habitude de tenir le cou recourbé » ou : « Le cygne chanteur a un bec jaune ; il est migrateur hivernal et niche dans la toundra. »

En ce qui concerne les flores, une organisation différente est souvent adoptée. La raison principale est qu'il y a beaucoup plus d'espèces de plantes à fleurs que d'oiseaux. À l'observation d'une fleur, il est impossible de parcourir au hasard des milliers d'illustrations en espérant trouver la bonne page avec une probabilité suffisante. C'est pourquoi le système de recherche est fondé sur un questionnaire structuré. Supposons avoir devant nous une certaine fleur et dans la main une édition de la flore Bonnier [2]. Une séquence de reconnaissance (un peu raccourcie pour la lisibilité) sera par exemple la suivante, en supposant que la fleur « réponde » positivement à chaque test :

- *Plante ayant des fleurs, avec des étamines ou un pistil, ou les deux à la fois ?*
- *Fleurs non réunies en capitule entouré d'une collerette de bractées ?*

[1] On parle ici de manuels concernant l'avifaune européenne.

[2] *Nouvelle flore pour la détermination facile des plantes, sans mots techniques, représentant toutes les espèces vasculaires des environs de Paris dans un rayon de 100 kilomètres, des départements de l'Eure, de l'Eure et Loire, etc.*, Ouvrage couronné par l'académie des sciences et par l'académie d'agriculture de France. Par G. Bonnier, membre de l'institut et professeur à la Sorbonne, et Georges de Layens, lauréat de l'académie des sciences. Quatorzième édition, augmentée (...) ; Librairie générale de l'enseignement, 1926.

Copyright © 2010 Eyrolles.

- *Fleurs à deux enveloppes de couleur et de consistance différentes ?*
- *Corolle non papilionacée ?*
- *Pétales libres entre eux ?*
- *Fleur ayant plus de douze étamines ?*
- *Étamines réunies entre elles ?*
- *Plante herbacée ?*
- *Fleurs à l'aisselle des feuilles ?*
- *Calicule à trois bractées libres et stigmate obtus ?*
- *Une seule fleur à l'aisselle des feuilles ?*
- *Bractées du calicule étroites ; carpelle velue ?*

Décision : la plante est la *Malva rotundifolia L.* (Mauve à feuille rondes)

À chaque question, qui porte sur un attribut de la fleur, la réponse est donc positive ou négative. Si aucune erreur n'est commise dans ces réponses, l'identification est réalisée.

Évidemment, les premiers chapitres de la flore Bonnier sont consacrés à des notions sur l'anatomie des plantes. Un index, comportant des mots comme « carpelle », « bractée », est également fourni. La difficulté est pour le lecteur de prendre une décision sans erreur pour chaque question posée. Le problème est d'organiser l'ensemble des questions de manière aussi efficace que possible, c'est-à-dire d'éviter les questions inutiles et de veiller à ce que chaque plante, en moyenne, puisse être identifiée par le plus petit nombre possible de questions.

1. Les arbres de décision

1.1 Principe

La technique des *arbres de décision* est fondée sur l'idée simple de réaliser la classification d'un objet par une suite de tests sur les attributs qui le décrivent. Ces tests sont organisés de telle façon que la réponse à l'un d'eux indique le prochain test auquel on doit soumettre cet objet.

Ce type de classification est, comme on l'a vu, couramment employé en sciences naturelles Dans ce cas, l'espace de représentation est défini par l'observation des caractéristiques anatomiques utiles de la plante (*étamines*, *corolle*, *calicule*, *bractées*, *etc.*) ainsi que de leur *existence conjointe*, *position relative*, *nombre*, *topologie*, etc. Il faut connaître la signification et la mesure d'une bonne centaine de tels termes (c'est la taille de l'espace de représentation, le nombre d' attributs) pour classer toute plante répertoriée, dont le nombre d'espèces possibles est ici de 1 500 à 2 000 (c'est le nombre de classes).

Le principe de cette règle de décision est d'organiser l'ensemble des tests possibles comme un arbre[3]. Une feuille de cet arbre désigne une des C classes (mais à chaque classe peuvent correspondre plusieurs feuilles) et à chaque nœud est associé un test (un *sélecteur*) portant sur un ou plusieurs attributs, éléments de l'espace de représentation ; la réponse à ce test désignera le fils du nœud vers lequel on doit aller. La classification s'effectue donc en partant de la racine pour poursuivre récursivement le processus jusqu'à ce qu'on rencontre une feuille. Une telle structure est appelée *arbre de décision*. La question qui nous intéresse particulièrement est de réaliser

[3] La botanique n'a plus rien à voir ici.

Copyright © 2010 Eyrolles.

l'apprentissage de telles structures de décision à partir d'exemples. Prenons une illustration dans un tout autre univers pour approcher ce problème.

—— EXEMPLE **Arbre de décision** ——

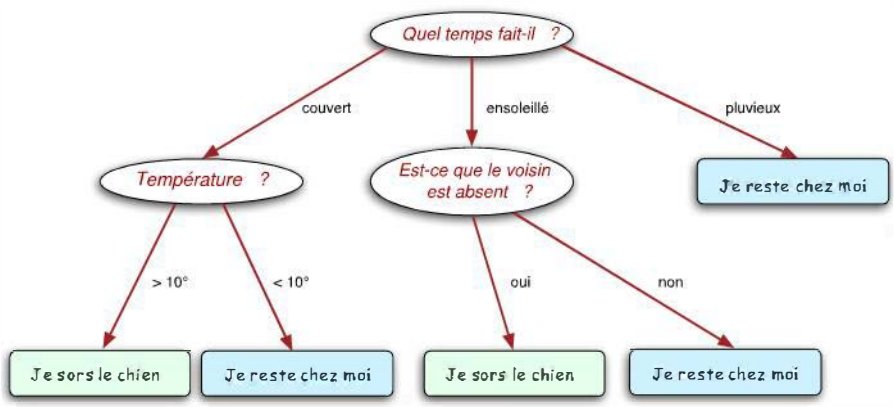


FIG. 13.1: *Un arbre de décision pour la promenade du chien.*

Supposons que j'aie à prendre la décision suivante : *vais-je sortir le chien ou non ?* Pour cela, j'observe les attributs suivants :

- *Quel temps fait-il ?* C'est un attribut nominal pouvant prendre les valeurs *pluvieux*, *ensoleillé* ou *couvert*.
- *Quelle est la température extérieure ?* Cet attribut est numérique.
- *Est-ce que le voisin est parti en week-end avec son chat ?* Cet attribut est binaire.

Mon expérience m'a prouvé que la présence du chat du voisin rend la promenade assez pénible ; mais je sais que cet animal déteste l'humidité. D'autre part, le retour d'un chien mouillé n'est pas très plaisant pour mon tapis. Pour finir, ajoutons que je suis plutôt frileux. Moyennant quoi, je peux par exemple organiser ma décision selon la hiérarchie de la figure 13.1.

Cet *arbre de décision* se lit ainsi : j'observe d'abord le ciel. Si je remarque que le temps est couvert je dois ensuite regarder le thermomètre pour me décider. Si le temps est ensoleillé, je dois alors m'intéresser à la présence de mon voisin. S'il pleut, ma décision est toute prise.

Quelques avantages

Si l'on connaît un arbre de décision associé à un problème de classification, on voit immédiatement les avantages de ce type de règle de classification :

- Le nombre moyen de tests à effectuer sur une forme peut être extrêmement réduit (si les d attributs sont tous binaires, ce nombre est limité par d).
- La structure de décision est globale : on n'a pas de problème pour traiter C classes.
- Le test de tous les attributs de chaque objet à chaque nœud n'est pas nécessaire ; dans la plupart des cas pratiques, on se limite même à un seul test.

Évidemment, ces avantages ne valent que s'il est possible de construire un arbre de décision à partir d'un ensemble d'apprentissage en remplissant au mieux deux conditions : celle de la proximité du *risque empirique* et du *risque réel* et celle de la *simplicité* de l'arbre obtenu, comme indiqué au chapitre 3.

Copyright © 2010 Eyrolles.

1.2 La construction récursive d'un arbre de décision

1.2.1 Introduction

Dans l'exemple botanique proposé dans l'introduction, l'expertise joue un rôle très important : l'arbre de décision est construit à partir de connaissances sur la correspondance complexe entre les caractéristiques observables d'une plante et la définition de son espèce (fondée elle-même sur la possibilité de reproduction, la distribution géographique, etc.). La structure de cet arbre de décision est donc le résultat de l'expérience des botanistes. Mais pour réaliser la construction automatique d'un arbre de décision, il faut s'appuyer seulement sur un ensemble d'apprentissage et non pas sur une expertise. Comment sous cette hypothèse *apprendre* un arbre de décision performant en généralisation ?

Notons d'abord qu'il est hors de question d'explorer exhaustivement l'ensemble des arbres possibles pour déterminer le plus performant au sens d'un critère inductif comme l'*ERM* (chapitre 2) ou le principe de compression maximale (chapitre 21). En effet, le nombre d'arbres possibles est gigantesque, croissant exponentiellement avec le nombre d d'attributs et le nombre moyen a de valeurs possibles par attributs. Le nombre d'arbres à n nœuds de degré au plus 2 est ainsi fourni par le nombre de Catalan $C_n = \frac{1}{n+1}\begin{pmatrix} 2n \\ n \end{pmatrix}$. Il faut donc un moyen « intelligent » d'explorer l'espace des hypothèses.

L'apprentissage des arbres de décision procède par une exploration du général au particulier en commençant par un arbre à un nœud racine correspondant à une partition simple de l'espace $\mathcal{X}$ des exemples, puis en raffinant progressivement cette partition par ajout successif de nœuds dans l'arbre, ce qui revient à subdiviser itérativement les partitions de l'espace des exemples.

L'approche, appelée *induction descendante d'arbres de décision* (*top-down induction of decision tree*), procède de manière descendante, en partant de l'échantillon des données d'apprentissage toutes classes confondues. Tant que l'échantillon courant de données n'est pas « pur » (tous les exemples de la même classe) ou qu'il reste au moins un attribut à tester, un attribut est sélectionné, selon un critère décrit plus bas, pour servir de test permettant de subdiviser l'échantillon d'apprentissage courant en sous-échantillons distincts. À l'arrêt, on obtient donc un arbre de tests (nœuds) dont les feuilles correspondent à des échantillons d'exemples aussi « purs » que possible, c'est-à-dire idéalement appartenant à la même classe. Ce n'est pas en général possible, mais on garde l'idée de ramifier l'arbre autant qu'il le faudra pour arriver à une configuration où chaque feuille représente des données appartenant toutes à la même classe. Cette technique, basée sur le principe *ERM*, produit un arbre dont chaque feuille ne couvre plus qu'un faible nombre de données pures. Parce qu'il est trop dépendant des données d'apprentissage, on sait qu'il donnera vraisemblablement une mauvaise généralisation. C'est pourquoi on essaie de contrebalancer ce « surapprentissage » par un mécanisme limitant la complexité de l'arbre (donc du modèle) appris. On retrouve là le problème de la sélection de modèles (voir chapitres 2 et 3).

Si l'on a assez de données d'apprentissage, la façon la plus efficace est de procéder en deux passes : d'abord utiliser une partie $\mathcal{A}$ de l'ensemble d'apprentissage pour construire un arbre T_{max} dont toutes les feuilles sont aussi pures que possible ; ensuite élaguer (simplifier) cet arbre avec une autre partie $\mathcal{V}$ des données (un ensemble de *validation* comme défini au chapitre 3). Le reste des données, sous forme d'ensemble de test $\mathcal{T}$, sert enfin à évaluer le risque réel de l'arbre construit. Si les données sont peu nombreuses, une technique un peu plus complexe de validation croisée est nécessaire.

Au cours de la construction de T_{max}, le test mis en place à chaque nœud est basé sur le seul examen de la meilleure façon de séparer en classes le sous-ensemble considéré des points d'apprentissage qu'il régit. Le paragraphe suivant présente comment fabriquer de tels critères.

Copyright © 2010 Eyrolles.

On montrera ensuite comment élaguer T_{max}.

Pour simplifier l'exposé, nous commençons par le cas d'attributs binaires, mais tout ce qui suit est immédiatement généralisable au cas d'attributs multivalués.

1.2.2 Le cas des attributs binaires

Position du problème

On dispose d'un ensemble d'apprentissage $\mathcal{S}$ de m exemples dont l'un est noté (x,ω)[4]. Cet exemple est décrit par d attributs $\{x_i, i = 1,d\}$ et par une classe $\omega \in \mathcal{C} = \{\omega_1,...,\omega_C\}$. On cherche d'abord, en applicant le principe ERM, à construire un arbre de classification dont l'erreur apparente est nulle. On suppose pour l'instant que les attributs sont à valeur binaire, avant de considérer plus loin le cas où ils sont nominaux ou continus[5].

L'algorithme de construction, décrit informellement ci-dessus, s'écrit récursivement :

Algorithme 20 : Construction récursive d'un arbre de décision

Procédure *Construire-arbre (nœud X)*
début
  si *Tous les points de X appartiennent à la même classe* **alors**
    Créer une feuille portant le nom de cette classe
  sinon
    Choisir le meilleur attribut pour créer un nœud
    Le test associé à ce nœud sépare X en deux parties notées X_g et X_d
    `Construire-arbre`(X_g)
    `Construire-arbre`(X_d)
  fin si
fin

Par conséquent, quand l'arbre est partiellement construit, à chaque nœud correspond un sous-ensemble des exemples d'apprentissage : ceux qui satisfont tous les tests binaires menant à ce nœud. Si ce sous-ensemble n'est pas constitué de points appartenant tous à la même classe, la construction doit se poursuivre. Il faut alors choisir le meilleur attribut à tester.

L'appel de cette procédure récursive se fait sur l'ensemble d'apprentissage $\mathcal{S}$. Il est à noter que dans certains cas, le test d'arrêt ne peut pas être satisfait : il peut exister plusieurs exemples ayant les mêmes attributs et des classes différentes[6]. Dans ce cas, la classe est attribuée par un « vote » des données concernées ou par un tirage au sort pondéré par l'importance relative des classes à cette feuille.

Une interprétation probabiliste

Plaçons-nous au cours de cette construction à un nœud auquel sont attachés n points de l'échantillon $\mathcal{S}$, répartis en C classes ω_j comportant chacune n_j points ($\sum_{j=1,C} n_j = n$).

Considérons un attribut binaire a, dont l'indice n'a pas besoin d'être précisé. Il partage chaque sous-ensemble n_j en deux parties, comportant respectivement l_j et r_j points pour test sur $a = VRAI$ et test sur $a = FAUX$.

[4] On n'a pas besoin ici d'indicer les exemples dans l'ensemble d'apprentissage.

[5] Les attributs à domaine arborescent ou séquentiels ne sont pas traités simplement par les arbres de décision.

[6] Soit parce qu'il n'y a pas assez d'attributs pour les décrire et les discriminer, soit parce qu'il y a des erreurs de description ou d'étiquetage des exemples.

Copyright © 2010 Eyrolles.

Notons :

$$l = \sum_{j=1}^{C} l_j \quad \text{et} \quad r = \sum_{j=1}^{C} r_j \quad \text{avec :} \quad r + l = n \tag{13.1}$$

On peut considérer que les n points d'apprentissage sont des tirages aléatoires selon deux distributions discrètes possibles : celle des C valeurs que prend la valeur ω de la classe et celle des deux valeurs de $\boldsymbol{a}$. On en déduit que :

- l_j/n et r_j/n sont des estimations des probabilités $P(\boldsymbol{a} = VRAI, \omega = \omega_j)$ et $P(\boldsymbol{a} = FAUX, \omega = \omega_j)$.
- l/n et r/n sont des estimations de $P(\boldsymbol{a} = VRAI)$ et de $P(\boldsymbol{a} = FAUX)$.
- n_j/n est une estimation de $P(\omega = \omega_j)$.

$\boldsymbol{a} =$	$VRAI$	$FAUX$	
$\omega = \omega_1$	l_1	r_1	n_1
	...	...	...
$\omega = \omega_j$	l_j	r_j	n_j
	...	...	...
$\omega = \omega_C$	l_C	r_C	n_C
	l	r	n

Une mesure pour choisir l'attribut

La théorie de l'information nous fournit une mesure naturelle de l'homogénéité entre deux distributions de probabilités à valeurs discrètes : l'*information mutuelle*, ou *entropie croisée* [CT91]. En notant ω la première variable et $\boldsymbol{a}$ la seconde, $\mathcal{D}_\omega$ et $\mathcal{D}_{\boldsymbol{a}}$ les ensembles finis des valeurs qu'elles peuvent prendre, l'entropie croisée de ω et de $\boldsymbol{a}$ est donnée par la formule[7] :

$$I(\omega, \boldsymbol{a}) = \sum_{u,v \in \mathcal{D}_\omega \times \mathcal{D}_{\boldsymbol{a}}} P(\omega = u, \boldsymbol{a} = v) log \frac{P(\omega = u, \boldsymbol{a} = v)}{P(\omega = u)P(\boldsymbol{a} = v)} \tag{13.2}$$

$I(\omega, \boldsymbol{a})$ présente un minimum à 0 quand on a $P(\omega = u, \boldsymbol{a} = v) = P(\omega = u)P(\boldsymbol{a} = v)$ sur tout le domaine $\mathcal{D}_\omega \times \mathcal{D}_{\boldsymbol{a}}$, c'est-à-dire quand les deux distributions sont indépendantes[8] ; elle est en revanche maximale quand les distributions sont parfaitement corrélées, c'est-à-dire quand pour tout élément u_i du domaine $\mathcal{D}_\omega$ il existe un et un seul élément v_j du domaine $\mathcal{D}_{\boldsymbol{a}}$ avec : $P(\omega = u_i) = P(\boldsymbol{a} = v_j)$.

La variable aléatoire ω possède une *entropie* $H(\omega)$ qui se définit par :

$$H(\omega) = - \sum_{u \in \mathcal{D}_\omega} P(\omega = u) log(P(\omega = u))$$

De même, on peut définir l'entropie de ω *conditionnée par* $\boldsymbol{a}$ comme :

$$H(\omega|\boldsymbol{a}) = - \sum_{u,v \in \mathcal{D}_\omega \times \mathcal{D}_{\boldsymbol{a}}} P(\omega = u, \boldsymbol{a} = v) log(P(\omega = u|\boldsymbol{a} = v))$$

Un résultat classique de théorie de l'information [CT91] nous affirme alors que :

$$I(\omega, \boldsymbol{a}) = H(\omega) - H(\omega|\boldsymbol{a})$$

[7] Dans tout ce chapitre, la base des logarithmes est prise à deux : $log(a)$ doit se lire comme $log_2(a)$.

[8] $0 \log 0$ est pris égal à 0.

Copyright © 2010 Eyrolles.

Dans le cas que nous traitons ici, la variable $\boldsymbol{a}$ est un attribut binaire, donc $\mathcal{D}_{\boldsymbol{a}} = \{VRAI, FAUX\}$ et ω représente la distribution des données sur les C classes.

Compte tenu de ce qui a été dit plus haut, les valeurs $H(\omega)$, $H(\omega|\boldsymbol{a})$ et $I(\omega,\boldsymbol{a})$ peuvent donc s'estimer par :

$$\widehat{I}(\omega,\boldsymbol{a}) = -\sum_{j=1}^{C} \frac{l_j}{n} \log \frac{l_j/n}{(l/n)\times(n_j/n)} + \frac{r_j}{n} \log \frac{r_j/n}{(r/n)\times(n_j/n)}$$

$$\widehat{I}(\omega) = -\sum_{j=1}^{C} \frac{n_j}{n} \log \frac{n_j}{n}$$

$$\widehat{H}(\omega \mid \boldsymbol{a}) = -\sum_{j=1}^{C} \frac{l_j}{n} \log \frac{l_j}{l} + \frac{r_j}{n} \log \frac{r_j}{r} \tag{13.3}$$

Et on peut vérifier que : $\widehat{I}(\omega,\boldsymbol{a}) = \widehat{I}(\omega) - \widehat{H}(\omega \mid \boldsymbol{a})$. Pour faciliter les calculs, on note :

$$J(\boldsymbol{a} = VRAI) = \sum_{j=1}^{C} \frac{l_j}{l} \log \frac{l_j}{l} \quad \text{et} \quad J(\boldsymbol{a} = FAUX) = \sum_{j=1}^{C} \frac{r_j}{r} \log \frac{r_j}{r}$$

et donc :

$$\widehat{H}(\omega \mid \boldsymbol{a}) = \frac{l}{n} J(\boldsymbol{x} = VRAI) + \frac{r}{n} J(\boldsymbol{a} = FAUX) \tag{13.4}$$

Pour construire un nœud dans l'arbre, une idée naturelle et interprétable en terme de théorie de l'information est donc de chercher parmi les d attributs celui qui possède la plus grande corrélation avec la répartition en classes, autrement dit celui qui a la meilleure entropie croisée avec la distribution des points d'apprentissage sur les classes.

Par conséquent, chercher parmi tous les attributs celui qui possède l'information mutuelle la plus grande avec la distribution en classes des n points d'apprentissage revient à trouver celui qui minimise la quantité $\widehat{H}(\omega \mid \boldsymbol{a})$, ou, si l'on préfère, à rechercher l'attribut d'indice $i^* \in \{1,d\}$ tel que :

$$i^* = \underset{i=1,d}{\text{ArgMin}} \ \widehat{H}(\omega \mid \boldsymbol{a}_i) \tag{13.5}$$

D'autres mesures pour choisir l'attribut

L'entropie croisée n'est pas le seul critère à pouvoir être utilisé : on l'a en effet interprétée comme une mesure de distance entre deux distributions de probabilités. Pourquoi ne pas employer d'autres distances, en quittant le strict cadre de la théorie de l'information? Par exemple, la métrique de Gini [Gin38] est très employée en pratique. Son estimation se calcule comme suit, dans les mêmes notations que précédemment :

$$Gini(\omega \mid \boldsymbol{a}) = \frac{l}{n} \sum_{j=1}^{C} \frac{l_j}{n_j}\left(1 - \frac{l_j}{n_j}\right) + \frac{r}{n} \sum_{j=1}^{C} \frac{r_j}{n_j}\left(1 - \frac{r_j}{n_j}\right) \tag{13.6}$$

On remarque que ce critère est nul quand, pour chaque classe ω_j, l'une des deux valeurs l_j ou r_j est nulle. Il est en revanche maximal quand $l_j = n_j$ pour toutes les classes.

On peut aussi utiliser le critère du χ^2, avec des propriétés semblables, qui s'estime par la formule :

$$\chi^2(c \mid \boldsymbol{a}) = \sum_{j=1}^{C} \left(\frac{l_j - (ln_j/n)}{\sqrt{ln_j/n}}\right)^2 + \left(\frac{r_j - (rn_j/n)}{\sqrt{rn_j/n}}\right)^2 \tag{13.7}$$

Copyright © 2010 Eyrolles.

—— EXEMPLE **Apprentissage d'arbre de décision** ——

Dans l'exemple qui suit, le problème d'apprentissage consiste à trouver une règle de décision binaire à partir de huit exemples sur quatre paramètres binaires. Le problème qui se pose à un enfant qui revient de l'école est le suivant : *peut-il aller jouer chez son voisin ou non?* L'expérience, qu'il a acquise par punition récompense sur les huit jours d'école précédents, est résumée dans le tableau n des huit exemples d'apprentissage suivants :

	Mes Devoirs sont-ils Finis ?	Maman est-elle de Bonne Humeur ?	Est-ce qu'il Fait Beau?	Mon Goûter est-il Pris?	DÉCISION
1	oui	non	oui	non	JOUER
2	non	oui	non	oui	JOUER
3	oui	oui	oui	non	JOUER
4	oui	non	oui	oui	JOUER
5	non	oui	oui	oui	PAS JOUER
6	non	oui	non	non	PAS JOUER
7	oui	non	non	oui	PAS JOUER
8	oui	oui	non	non	PAS JOUER

Pour construire la racine de l'arbre de décision, il faut d'abord trouver l'attribut dont la distribution possède l'entropie mutuelle la plus faible avec celle de la décision.

Notons, pour simplifier, $H(\omega|DF)$ pour $H(\omega|Mes\ Devoirs\ sont\text{-}ils\ Finis\ ?)$ et de la même manière : $H(\omega|DF)$, $H(\omega|BH)$, $H(\omega|FB)$, $H(\omega|GP)$.

On a :

$$H(\omega|DF) = \frac{5}{8}J(DF = oui) + \frac{3}{8}J(DF = non)$$

avec

$$J(DF = oui) = -\frac{3}{5}log(\frac{3}{5}) - \frac{2}{5}log(\frac{2}{5})$$

et

$$J(DF = non) = -\frac{1}{3}log(\frac{1}{3}) - \frac{2}{3}log(\frac{2}{3})$$

Soit :

$$H(\omega|DF) \approx 0.93$$

On trouve par un calcul analogue :

$$H(\omega|FB) \approx 0.80$$

$$H(\omega|BH) \approx 0.93$$

$$H(\omega|GP) = 1.$$

On choisit donc pour racine de l'arbre le test *Est-ce qu'il Fait Beau ?* qui minimise l'entropie croisée avec la distribution en classes.

Sous la branche gauche portant la valeur *oui* se trouve le tableau suivant des exemples corrects pour ce test.

	Mes Devoirs sont-ils Finis ?	Maman est-elle de Bonne Humeur ?	Mon Goûter est-il Pris ?	DÉCISION
1	oui	non	non	JOUER
3	oui	oui	non	JOUER
4	oui	non	oui	JOUER
5	non	oui	oui	PAS JOUER

Copyright © 2010 Eyrolles.

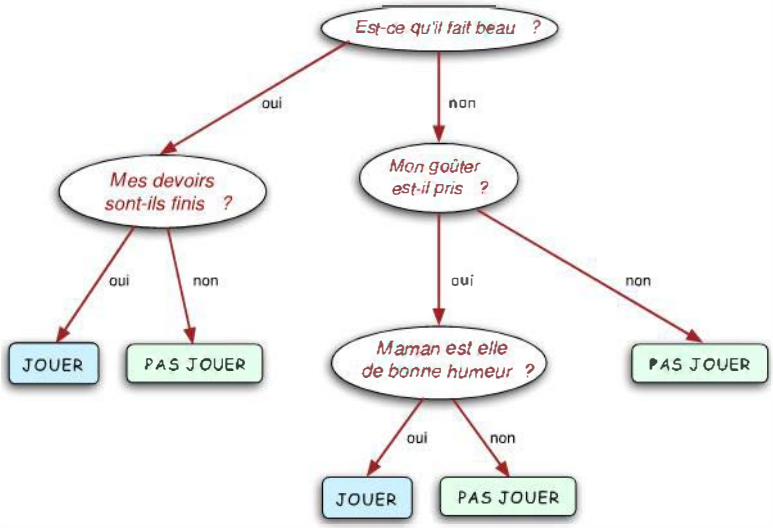


FIG. 13.2: *L'arbre de décision construit sur les huit exemples précédents.*

Sous la branche droite, portant la valeur *non* se trouve le tableau suivant.

	mes Devoirs sont-ils Finis ?	Maman est-elle de Bonne Humeur?	Mon Goûter est-il Pris ?	DÉCISION
2	non	oui	oui	JOUER
6	non	oui	non	PAS JOUER
7	oui	non	oui	PAS JOUER
8	oui	non	non	PAS JOUER

La poursuite du procédé conduit finalement à l'arbre de décision de la figure 13.2.

1.2.3 Le cas des attributs non binaires

On a traité ci-dessus le cas des attributs binaires. Mais il est possible de calculer de la même manière une entropie croisée avec la distribution en classes pour les autres types d'attributs.

Le cas binaire

Pour mémoire, le test consiste ici à descendre dans un sous-arbre si le test sur l'attribut choisi vaut $VRAI$, dans l'autre s'il vaut $FAUX$.

Le cas nominal

Le cas où les attributs sont à valeurs discrètes se généralise facilement quand le test que l'on construit se réduit à opposer une valeur à toutes les autres : on est alors ramené au cas binaire. Par exemple, s'il existe un attribut `couleur` prenant ses valeurs dans l'ensemble $\{bleu, rouge, vert, jaune\}$, il est simple de l'éclater en quatre attributs binaires, tel `couleur-bleu`, qui est $VRAI$ ou $FAUX$ sur chaque donnée d'apprentissage. On se replace alors dans le cas exposé ci-dessus, avec la transformation d'un attribut nominal à k valeurs possibles en k attributs binaires que l'on traite indépendemment.

Cette technique a l'inconvénient d'oublier la signification globale de l'attribut ; en effet, si `couleur-rouge` est $VRAI$ pour un attribut, `couleur-bleu` est automatiquement $FAUX$; mais cette propriété n'apparaît plus explicitement dans les données. Une autre solution est alors de

Copyright © 2010 Eyrolles.

calculer directement l'information mutuelle entre les deux variables à valeurs discrètes que sont d'une part cet attribut et d'autre part l'ensemble des classes. Si celle-ci se révèle la meilleure pour tous les attributs, on crée alors un nœud non binaire dans l'arbre de décision (dans l'exemple précédent, le test de l'attribut « couleur » donne quatre réponses possibles). Le seul inconvénient est qu'il faut gérer une structure de données plus complexe.

Le cas continu

Traiter un attribut continu peut paraître plus difficile, mais en pratique ce n'est pas fondamentalement différent : puisque le nombre de données d'apprentissage est fini, le nombre des valeurs que prend cet attribut sur les exemples est aussi fini. Mieux, ses valeurs sont ordonnées, contrairement au cas nominal. Le sélecteur consistera donc à comparer les valeurs à un seuil pour construire un nœud binaire.

Pour un attribut a continu, on procède alors ainsi : on trie les points d'apprentissage selon la valeur de cet attribut, puis on cherche le seuil $s(a)$ qui minimise l'un des critères précédents[9].

Il est à noter que l'arbre de décision est le seul modèle permettant de gérer de manière homogène les attributs de nature variée, en particulier les mélanges continus et binaires.

L'utilisation simultanée de plusieurs attributs continus

Dans le cas où un attribut est continu, chaque test sur cet attribut n'est autre que la comparaison à un seuil ; si tous les attributs sont continus, on obtient donc finalement dans $\mathbb{R}^d$ des surfaces de séparation entre les régions attribuées aux classes qui sont composées d'hyperplans *orthogonaux aux axes*. Il est tentant de relâcher cette contrainte, pour éviter de construire des arbres complexes sur des situations simples. On peut alors chercher des séparatrices linéaires non parallèles aux axes en utilisant des tests sur des combinaisons linéaires d'attributs à chaque nœud, et non sur un seul attribut. On réalise alors un arbre de décision *oblique*.

Cependant, la méthode proposée pour un attribut continu ne peut pas se généraliser pour la combinaison d'attributs continus : l'espace de recherche est cette fois infini, puisque l'on cherche à chaque nœud des valeurs (en nombre $d+1$) non contraintes comme précédemment par une relation d'ordre. On emploie alors des techniques d'optimisation, comparables à celles du chapitre 9. Les méthodes présentées dans [MKS94] et [BU92] sont des exemples efficaces.

1.3 Comment élaguer un arbre trop précis ?

1.3.1 Pourquoi élaguer ?

On a vu que la poursuite de l'algorithme de construction jusqu'à son terme naturel fabrique un arbre T_{max} dont les feuilles sont pures, c'est-à-dire correspondent à des exemples de la même classe ; il y a là clairement un risque de mésestimation de la probabilité d'erreur par le taux d'erreur apparent, qui vaut ici exactement 0. On se trouve donc dans le cas exposé au chapitre 2 : le nombre de nœuds de l'arbre de décision est un critère de complexité simple et efficace pour lequel les courbes présentées à la figure 3.16 dans le chapitre 3 sont caractéristiques. Chercher la valeur « optimale » k_0 du nombre de nœuds revient donc à trouver une technique pour contrôler la taille de l'arbre. Il s'agit donc d'une méthode de régularisation ou de sélection de modèle (voir chapitre 2).

1.3.2 Une première solution : le préélagage

Une solution simple consiste à cesser de diviser un nœud quand la pureté des points qu'il domine est non pas parfaite, mais suffisante. Une fois sélectionné le meilleur attribut, on regarde

[9] Ceci nécessite, pour les n données, l'examen de $n-1$ seuils : par exemple les valeurs médianes entre deux points d'apprentissage dans leur liste triée.

Copyright © 2010 Eyrolles.

si la valeur du critère de la division est inférieure à un certain seuil ; en pratique, ceci revient à admettre que, s'il existe une classe suffisamment majoritaire sous un nœud, on peut considérer ce dernier comme une feuille et lui attribuer la classe en question. Selon le critère de division utilisé, diverses heuristiques ont été proposées pour régler le seuil précédent. Sa valeur peut d'ailleurs être variable selon le nœud où l'on se trouve, dépendant de l'estimation de la probabilité *a priori* des classes, de l'estimation empirique de la difficulté à les séparer, etc.

Ces méthodes présentent certains inconvénients, dont le principal est qu'elles sont myopes (puisqu'elles ne prennent en compte qu'un critère local à la feuille examinée), et peuvent de ce fait manquer un développement de l'arbre qui serait excellent. C'est pourquoi on leur préfère souvent des méthodes d'élagage *a posteriori*, une fois que l'arbre a été entièrement développé.

1.3.3 Le post-élagage par un ensemble indépendant de validation

Une autre technique, plus valide théoriquement et plus efficace en pratique, consiste à d'abord construire l'arbre de décision complètement, puis seulement après à chercher à le simplifier en l'élaguant progressivement en remontant des feuilles vers la racine. Pour juger quand il est bon d'arrêter d'élaguer l'arbre, on utilise un critère de qualité qui exprime souvent un compromis entre l'erreur commise par l'arbre et une mesure de sa complexité.

L'erreur commise est mesurée grâce à un ensemble de validation (voir chapitre 3). On supposera donc dans ce paragraphe que l'ensemble d'apprentissage est assez important pour être coupé en deux parties : l'une (ensemble d'apprentissage proprement dit) pour construire l'arbre de décision T_{max}, l'autre (ensemble de validation) pour choisir le meilleur parmi les élagages proposés.

L'algorithme optimal consisterait à calculer le taux d'erreur de l'ensemble de validation sur tous les arbres qu'il est possible d'obtenir par élagage de T_{max}. Mais leur nombre croît très rapidement avec la taille de T_{max}. On utilise donc des solutions sous-optimales, dont la plus classique (un algorithme *glouton*) consiste à construire sans retour en arrière une séquence d'arbres par élagages successifs, en remontant des feuilles vers la racine.

Cette séquence se note $S = (T_{max}, T_1, ..., T_k, ...T_n)$. T_n est l'arbre constitué d'une seule feuille comprenant les m points d'apprentissage. C'est donc l'arbre élagué au maximum. Pour passer de T_k à T_{k+1}, il faut transformer un nœud dans T_k en feuille. Pour savoir si cet élagage serait bénéfique, l'idée générale est de comparer le « coût » de l'arbre élagué et celui de l'arbre non élagué, et d'arrêter l'élagage quand le coût du premier dépasse le coût du second. Pour évaluer ce coût, plusieurs critères ont été proposés qui prennent tous en compte à la fois l'erreur commise par l'arbre et une mesure de sa complexité (voir en particulier les articles de synthèse [BA97, EMS97, Min89]).

Nous examinons ici le critère consistant à choisir le nœud ν qui minimise sur l'ensemble des nœuds de T_k la valeur suivante :

$$\varpi(T_k, \nu) = \frac{MC(\nu, k) - MCT(\nu, k)}{n(k).(nt(\nu, k) - 1)} \tag{13.8}$$

où :

- $MC(\nu, k)$ est le nombre d'exemples de l'ensemble d'apprentissage mal classés par le nœud ν de T_k *dans l'arbre élagué* à ν.
- $MCT(\nu, k)$ est le nombre d'exemples de l'ensemble d'apprentissage mal classés sous le nœud ν *dans l'arbre non élagué*.
- $n(k)$ est le nombre de feuilles de T_k.
- $nt(\nu, k)$ est le nombre de feuilles du sous-arbre de T_k situé sous le nœud ν.

Copyright © 2010 Eyrolles.

Ce critère permet donc d'élaguer un nœud de T_k de façon à ce que T_{k+1}, l'arbre obtenu, possède le meilleur compromis entre taille et taux d'erreur apparent.

Finalement, la suite $S = (T_{max}, T_1, ..., T_k, ...T_n)$ possède un élément T_{k_0} pour lequel le nombre d'erreurs commises est minimal *sur l'ensemble de validation* : c'est cet arbre-là qui sera finalement retenu par la procédure d'élagage.

Algorithme 21 : Élagage d'un arbre de décision

Procédure *élaguer*(T_{max})
début
  $k \leftarrow 0$
  $T_k \leftarrow T_{max}$
  tant que T_k *a plus d'un nœud* **faire**
    pour chaque *nœud* ν *de* T_k **faire**
      calculer le critère $\varpi(T_k, \nu)$ sur l'ensemble d'apprentissage
    fin
    choisir le nœud ν_m pour lequel le critère est maximum
    T_{k+1} se déduit de T_k en y remplaçant ν_m par une feuille
    $k \leftarrow k + 1$
  fin tant que
  Dans l'ensemble des arbres $\{T_{max}, T_1, ..., T_k, ...T_n\}$,
    choisir celui qui a la plus petite erreur de classification sur l'ensemble de validation.
fin

EXEMPLE **Élagage**

Les figures 13.3 et 13.4 représentent un petit ensemble d'exemples à deux classes et deux attributs numériques, ainsi que l'arbre de décision T_{max} appris par l'algorithme donné ci-dessus.

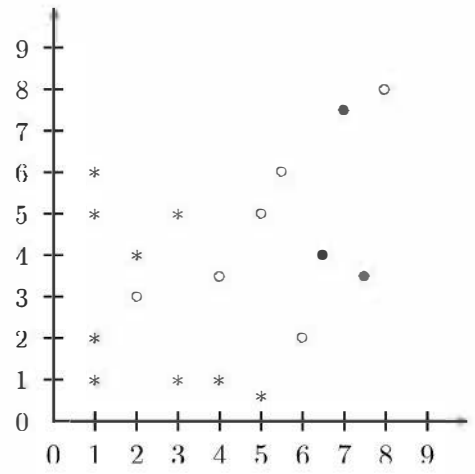


FIG. 13.3: *L'arbre de décision géométrique.*

En appellant n_1 le nœud racine de T_{max}, n_2 et n_3 ses fils gauche et droit et n_4 son dernier

Copyright © 2010 Eyrolles.

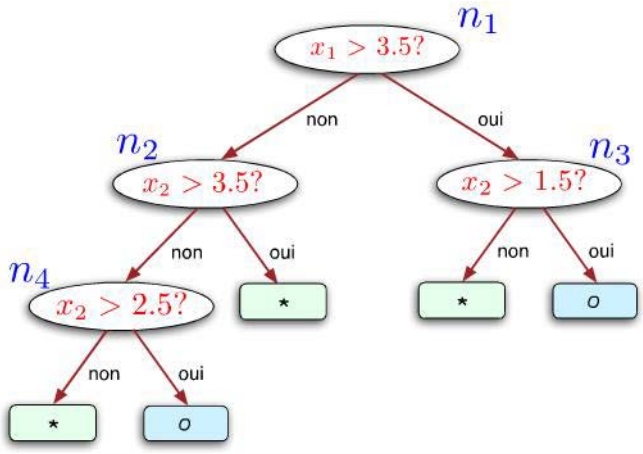


FIG. 13.4: *L'arbre de décision logique T_{max}.*

nœud intérieur (le fils gauche de n_2) :

$$\varpi(T_{max}, n_1) = \frac{MC(n_1,k) - MCT(n_1,k)}{n(k).(nt(n_1,k)-1)} = \frac{9-0}{5.(5-1)} = 9/20 \quad (13.9)$$

$$\varpi(T_{max}, n_2) = \frac{1-0}{5.(3-1)} = 1/10 \quad (13.10)$$

$$\varpi(T_{max}, n_3) = \frac{2-0}{5.(2-1)} = 2/5 \quad (13.11)$$

$$\varpi(T_{max}, n_4) = \frac{1-0}{5.(2-1)} = 1/5 \quad (13.12)$$

Par conséquent, l'arbre T_1 sera le résultat de l'élagage de T_{max} au nœud n_2 (figure 13.5). En travaillant désormais sur T_1, on trouve les valeurs :

$$\varpi(T_1, n_1) = \frac{9-1}{3.(3-1)} = 4/3 \quad (13.13)$$

$$\varpi(T_1, n_3) = \frac{2-0}{3.(2-1)} = 2/3 \quad (13.14)$$

L'arbre T_2 choisi résultera de l'élagage de n_3 dans T_1 ; il aura donc pour seul nœud la racine de T_{max}, avec une feuille pour chaque classe.

Supposons maintenant disposer d'un ensemble de validation (donné sur la figure 13.6). C'est en le testant sur les arbres T_{max}, T_1 et T_2 que l'on vient de calculer[10] que l'on choisira celui qui possède la meilleure estimation de taux d'erreur de classification. La procédure d'élagage sera alors terminée. Le résultat est donné sur la figure 13.7. On y voit que l'arbre T_{max}, avec son erreur apparente nulle, donne une estimation de l'erreur réelle de valeur 2/16 = 12.5 %. L'optimum de cette estimation est pour T_1 ; c'est donc cet arbre qui sera retenu au final.

Cet exemple est un peu trompeur, car compte tenu du faible nombre de points de l'ensemble de validation, les estimations ont des intervalles de confiance larges. Il vise cependant à montrer le comportement général de l'élagage par ensemble de validation et de manière plus générale, à présenter la régularisation sur un petit cas calculable « à la main ».

[10] Plus l'arbre noté plus haut T_N, composé d'une seule feuille et qui ne représente que la probabilité *a priori* des classes dans l'ensemble d'apprentissage.

Copyright © 2010 Eyrolles.

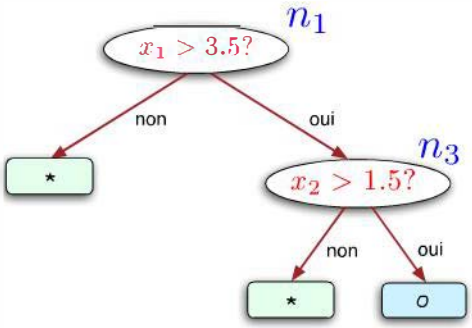


FIG. 13.5: *Un arbre de décision logique après un premier élagage : T_1.*

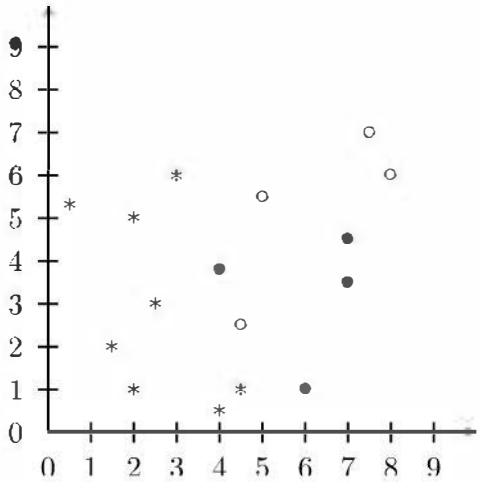


FIG. 13.6: *L'ensemble de validation sur l'arbre T_{max}.*

1.3.4 L'élagage par validation croisée

Dans le cas où les données d'apprentissage sont peu nombreuses, on a vu au chapitre 3 qu'il existe des méthodes comme le *leave-one-out* pour réaliser malgré tout un apprentissage statistiquement justifié. Une technique de ce genre a été développée spécifiquement pour les arbres de décision, fondée sur une sophistication de la méthode expliquée ci-dessus.

Elle commence par la construction de T_{max} et de celle de la suite

$$S = (T_{max}, T_1, ..., T_k, ...T_N)$$

sur l'ensemble d'apprentissage complet. Mais on ne dispose plus maintenant d'ensemble indépendant de validation pour passer optimalement de T_k à T_{k+1} ; l'astuce employée est la suivante : on divise aléatoirement l'ensemble d'apprentissage $\mathcal{S}$ en v parties $\mathcal{S}^1, ..., \mathcal{S}^v$ (en général, une dizaine) et on construit comme ci-dessus v séquences

$$S^i = (T^i_{max}, T^i_1, ..., T^i_k, ...T^i_N)$$

Pour chaque i, l'ensemble d'apprentissage est constitué de $\mathcal{S}^1 \cup \ldots \mathcal{S}^{i-1} \cup \mathcal{S}^{i+1}, \ldots, \cup \mathcal{S}^v$ et l'ensemble de validation est $\mathcal{S}^i$. Chaque arbre T^i_k possède donc une estimation $R(T^i_k)$ du taux d'erreur de classification : c'est celle qu'il fournit sur $\mathcal{S}^i$.

Copyright © 2010 Eyrolles.

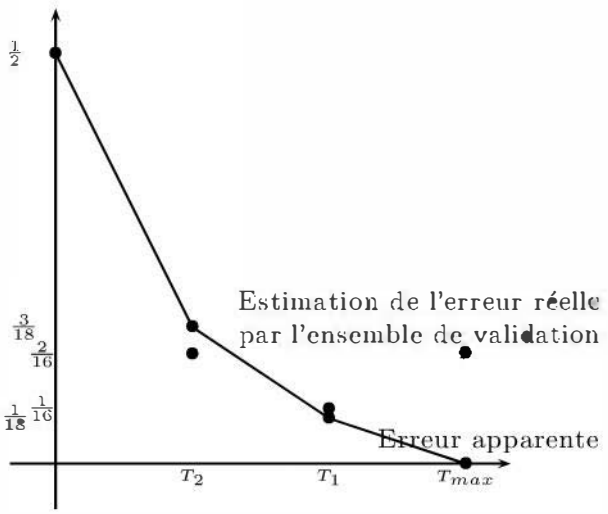


FIG. 13.7: *L'élagage d'un arbre de décision par ensemble de validation : un exemple de régularisation des hypothèses d'apprentissage.*

Pour regrouper ces informations, la suite du procédé consiste à rechercher dans chaque séquence S^i un élément considéré comme optimal (l'analogue de T_{k_0} dans le paragraphe précédent), que l'on doit extraire de la suite S. On réutilise le critère d'élagage précédent de la façon suivante : notons $\alpha_k = \varpi(T_{k-1}, d^*)$ et $\alpha_k^i = \varpi(T_{k-1}^i, \nu^*)$ les valeurs optimales des critères associées aux éléments des suites S et S^i.

Pour chaque arbre T_k de S, on définit $\alpha'_k = \sqrt{\alpha_k \alpha_{k+1}}$; on dira alors que l'arbre *le plus proche* de T_k dans la séquence S^i est celui pour lequel α_k^i est la plus grande possible, tout en restant inférieure à α'_k.

On dispose donc dans chaque suite S_i d'un élément Θ_k^i choisi comme le plus proche de T_k ; les Θ_k^i forment un ensemble de v éléments qui permet d'obtenir une estimation du taux d'erreur de T_k, par le calcul :

$$R_\Theta(T_k) = \frac{1}{v} \sum_{i=1}^{v} R(\Theta_k^i)$$

Le meilleur élément de la suite $S = \{T_{max}, T_1, ..., T_k, ...T_N\}$, qui sera celui retenu en fin de compte comme l'élagage optimal de T_{max}, est celui qui minimise $R_\Theta(T_k)$.

1.4 Les arbres de décision et la logique des propositions

Dans l'optique du chapitre 3, on peut voir les arbres de décision comme la construction imbriquée de sélecteurs et de règles de généralisation en logique des propositions. Chaque branche de l'arbre correspond à une conjonction de tests associés à une classe. L'ensemble des branches peut donc être considéré comme une disjonction exhaustive et exclusive de conjonctions (tous les exemples possibles sont couverts chacun par une règle et une seule). Nous l'illustrons sur un exemple.

Supposons les données d'apprentissage décrites par trois variables : la première à valeur continue : `fièvre`, la seconde nominale : `qualité`, pouvant prendre une des trois valeurs `homme`, `femme`, `enfant` et la troisième binaire : `réaction positive ou négative au test T`. La clas-

Copyright © 2010 Eyrolles.

sification à effectuer porte sur le diagnostic d'un certain syndrome S. Supposons que l'algorithme de construction ait calculé l'arbre suivant à partir des données d'apprentissage :

```
Si qualité = enfant
Alors      S = FAUX
Sinon :    Si réaction négative à T
           Alors      S = FAUX
           Sinon      Si qualité = femme
                      Alors      S = VRAI
                      Sinon      Si fièvre ≥ 39°
                                 Alors S = VRAI
                                 Sinon S = FAUX
```

Le concept appris, représenté maintenant en logique des propositions, peut se décrire ainsi : l'algorithme d'apprentissage a créé trois sélecteurs pour définir sa nouvelle représentation des connaissances.

- Un seuil à 39°, qui permet de binariser l'utilisation de la variable continue `fièvre` (remarquons qu'en général, plusieurs seuils peuvent apparaître dans les variables continues). Notons a_1 le fait qu'un patient ait une fièvre supérieure à 39°, $\bar{a}_1$ le contraire.
- La transformation de la variable nominale `qualité` en trois variables binaires `enfant`, `homme`, `femme`, que nous notons a_2, a_3 et a_4. Il est à remarquer que désormais, le fait que deux d'entre elles ne puissent pas être vraies à la fois apparaîtra implicitement.
- La variable binaire `réaction au test T`, qui reste binaire : notons-la a_5.

Finalement, l'apprentissage réalisé par la construction de l'arbre de décision ci-dessus peut se traduire dans la nouvelle représentation des connaissances par la découverte du concept :

$$(\bar{a}_2 \wedge a_5 \wedge a_3) \vee (\bar{a}_2 \wedge a_5 \wedge \bar{a}_3\ a_1)$$

D'une manière générale, un arbre de décision est une représentation de l'apprentissage d'un concept sous la forme d'une *disjonction de conjonctions*.

2. Les arbres de régression

2.1 Le principe

Nous avons vu, par exemple sur la figure 13.3, que les arbres de classification découpent l'espace des entrées $\mathbb{R}^d = \{x_1, \ldots x_i, \ldots, x_d\}$ en régions dont les côtés sont des hyperplans perpendiculaires aux axes. À l'intérieur de chacune de ces régions, la valeur prédite est constante : c'est une des classes ω_j, pour $i = 1, C$. Les arbres de régression utilisent le même mécanisme, à ceci près que les valeurs à prédire sont continues : après apprentissage sur un ensemble $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, on saura associer à chaque objet $x \in \mathbb{R}^d$ une valeur réelle de sortie y. Les arbres de régression prévoient une valeur numérique c_k constante pour chaque région construite après l'apprentissage et l'élagage. En supposant que le modèle induit contienne M régions $R_1, \ldots, R_M$ et en notant $\mathcal{I}(R_k)$ la fonction caractéristique de la région R_k, qui vaut 1 pour les points appartenant à R_k

Copyright © 2010 Eyrolles.

et 0 ailleurs, la fonction de sortie y associée à ce modèle est donc :

$$y = \sum_{k=1}^{M} c_k \, \mathcal{I}(R_k)$$

Nous considérons seulement le mécanisme le plus utilisé, celui de la régression quadratique. Il construit l'arbre en tentant de minimiser le carré de la différence entre les valeurs observées et valeurs prévues. Pour les m exemples d'apprentissage, il faut donc idéalement minimiser, en notant Δ la distance euclidienne :

$$\sum_{i=1}^{m} \Delta^2(\boldsymbol{u}_i - \boldsymbol{y}_i)$$

En considérant maintenant une optimisation région par région, cette expression est minimale pour la région R_n contenant n entrées lorsqu'elle est égale à la moyenne g des valeurs des points d'apprentissage qu'elle contient (voir une démonstration par exemple au chapitre 15).

2.2 La construction

L'algorithme utilisé pour construire un arbre de régression est glouton, comme celui de la construction d'un arbre de décision. L'attribut choisi à l'étape courante est celui qui peut se discrétiser par un découpage en deux minimisant le critère de la somme des écarts à la moyenne dans chaque partie du découpage. Ainsi, pour chaque attribut x_k et pour chaque valeur de découpage d_{jk} de cet attribut, on définit deux régions séparées par cette valeur et on obtient deux valeurs du critère que nous notons C^G_{jk} et C^D_{jk}.

On choisit au total d'utiliser la variable x_k et le découpage d_{jk} qui minimisent la somme des deux valeurs ci-dessus.

— EXEMPLE —

Voyons une étape de la construction d'un arbre de régression à deux dimensions. Les données d'apprentissage sont au nombre de seize et le découpage courant est celui donné à la figure 13.8.

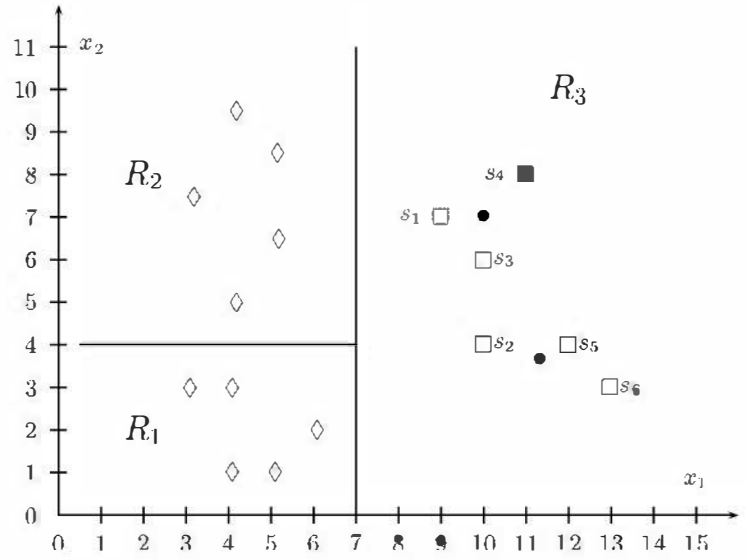


FIG. 13.8: *Une étape de la construction d'un arbre de régression.*

Copyright © 2010 Eyrolles.

Continuons la construction par la séparation en deux de la région R_3. Les valeurs de seuil possibles pour x_1 sont celles qui passent au milieu de deux points de R_3, c'est-à-dire les quatre valeurs 9.5, 10.5, 11.5 et 12.5. La troisième correspond à la droite verticale qui a été tracée en pointillés.

Quelle est la valeur du critère pour cette hypothèse ? Elles sépare les points de $\mathcal{R}_3$ en deux groupes : $\{s_1, s_2, s_3, s_4\}$ et $\{s_5, s_6\}$. Le centre de gravité (la moyenne) g_1 du premier groupe est aux coordonnées $(10, 5.75)$ et son homologue g_2 pour le second groupe est aux coordonnées $(12.5, 3.5)$. Le critère vaut donc :

$$\mathcal{C}_{31}^G + \mathcal{C}_{31}^D = \sum_{i\in\{1,2,3,4\}} \Delta^2(s_i - g_1) + \sum_{i\in\{5,6\}} \Delta^2(s_i - g_2) \simeq 11.85$$

Il faut aussi s'intéresser à x_2. Quatre valeurs de seuil sont possibles pour cet attribut : 3.5, 5, 6.5, 7.5. La droite correspondant à la seconde valeur 5 a été tracée en pointillés. Elle sépare les points de $\mathcal{R}_3$ en deux groupes : $\{s_4, s_1, s_3\}$ et $\{s_2, s_5, s_6\}$. Le centre de gravité h_1 du premier est aux coordonnées $(10, 7)$ et h_2 est aux coordonnées $(11.3, 3.7)$ (ces deux points sont indiqués sur la figure par le symbole •). Le critère vaut ici :

$$\mathcal{C}_{22}^G + \mathcal{C}_{22}^D = \sum_{i\in\{4,1,3\}} \Delta^2(s_i - h_1) + \sum_{i\in\{2,5,6\}} \Delta^2(s_i - h_2) \simeq 9$$

Le calcul complet montrerait que parmi toutes les séparations possibles sur x_1 et x_2, c'est cette dernière qui est la meilleure du point de vue du critère quadratique employé. La région R_3 sera donc divisée en deux par la droite d'équation $x_2 = 5$. Il est intéressant de noter que ce n'est pas la médiatrice entre h_1 et h_2.

2.3 La fin de la construction et l'élagage

Cette construction se poursuit jusqu'à ce que chaque point soit sur une feuille, ou lorsque les moyennes des deux régions les meilleures à séparer sont trop proches. Cette dernière façon de procéder est cependant dangereuse dans la mesure où un attribut très séparateur peut succéder à un qui ne l'est pas.

C'est pourquoi on a développé pour les arbres de régression des méthodes d'élagage puissantes. Après avoir laissé croître l'arbre jusqu'à ce que chaque feuille ne contienne qu'un petit nombre de points, voire un seul, on élague en réunissant les feuilles selon un critère de complexité dont Breiman [BFOS84] a montré qu'il est optimal pour un arbre donné. En d'autres termes, la procédure d'élagage ne transforme pas un arbre sous-optimal en un arbre optimal, bien entendu, elle se contente d'être capable, pour un arbre donné, de trouver l'élagage optimal.

Notes historiques et sources bibliographiques

Un certain nombre de travaux préliminaires sur la reconnaissance des formes par des méthodes hiérarchiques ont été compilés dans des articles bibliographiques, par exemple [SL91], avant la parution en 1984 des travaux décisifs dans « CART », le livre vraiment fondateur des techniques des arbres de décision et de régression [BFOS84]. Cet ouvrage ne développe pas seulement l'algorithme d'apprentissage de base, mais explique la validité statistique de l'élagage et donne des exemples sur des attributs binaires et numériques. La relève est ensuite principalement prise par R. Quinlan, qui développe les algorithmes ID3 et C4.5 [Qui93, QR89] et applique la méthode à

Copyright © 2010 Eyrolles.

des données numériques et symboliques variées. Un aspect original, le développement incrémental des arbres de décision, a été proposé par P. Utgoff [Utg89].

En lisant les ouvrages récents sur la fouille de données, par exemple [HK01], on constate l'importance pratique de ces méthodes, encore une fois pratiquement les seules à savoir traiter de manière homogène les exemples décrits par (presque) tous les types d'attributs.

Le livre récent de Zighed et Rakotomalala [ZR00] dresse un panorama complet sur les arbres de décision et analyse leur extension à des *graphes*, tout en présentant une grande variété d'applications, en particulier à la classification non supervisée et à la régression. Il donne aussi une remarquable bibliographie.

On trouvera des exposés pédagogiques sur les arbres de décision dans de nombreux livres. Les chapitres sur le sujet dans [WK91] et [Mit97] sont particulièrement didactiques. On pourra consulter une bonne bibliographie dans [Mit97] et [DHS01].

Le matériel présenté ici pour les arbres de décision a été en particulier inspiré par le texte de O. Gascuel dans [Nic93]. On reprend dans ce chapitre le critère de sélection d'un attribut par l'entropie proposé par Quinlan dans la méthode ID3 et la technique d'élagage de CART.

Résumé

Les *arbres de décision* permettent de classer des objets ayant des attributs de nature discrète. Ils sont contruits récursivement par spécialisation progressive sur l'espace des exemples. La classe d'un objet est ainsi prédite par la classe de la région calculée par l'arbre de décision dans laquelle il se trouve.

- À chaque nœud correspond un test à n valeurs (souvent binaire) sur un attribut.
- On classe un objet en partant de la racine et en suivant le chemin dans les nœuds donné par la réponse aux tests, jusqu'à une feuille (à laquelle est affectée une classe).
- L'apprentissage se produit récursivement, en choisissant pour racine de l'arbre l'attribut le plus corrélé avec la distribution en classes, ce qui sépare les exemples en n parties sur lesquelles on recommence ce choix.
- On arrête l'apprentissage lorsque les feuilles de l'arbre sont suffisamment « pures » ou qu'aucun test n'est plus disponible.
- Une phase d'élagage est ensuite nécessaire pour réduire l'arbre et diminuer sa probabilité d'erreur de classification.
- De par leur facilité d'utilisation et la relative transparence des hypothèses produites, les arbres de décision sont très largement employés.

Les *arbres de régression* sont tout à fait analogues, pour un problème d'apprentissage différent.

Copyright © 2010 Eyrolles.

Quatrième partie

Apprentissage par approximation et interpolation

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 14

Méthodes à noyaux

Les méthodes linéaires d'analyse de données et d'apprentissage ont été parmi les premières à être développées. Elles ont également été intensivement étudiées, en particulier parce qu'elles se prêtent bien à l'analyse mathématique. Cependant, de nombreuses applications requièrent des modèles non linéaires pour rendre compte des dépendances et des régularités sous-jacentes dans les données.

Les méthodes à noyaux permettent de trouver des fonctions de décision non linéaires, tout en s'appuyant fondamentalement sur des approches linéaires. Une fonction noyau correspond à un produit scalaire dans un espace de redescription des données, souvent de grande dimension. Dans cet espace, qu'il n'est pas nécessaire de manipuler explicitement, les méthodes linéaires peuvent être mises en œuvre pour y trouver des régularités linéaires, correspondant à des régularités non linéaires dans l'espace d'origine.

Grâce à l'utilisation de fonctions noyau, il devient ainsi possible d'avoir le meilleur de deux mondes : utiliser des techniques simples et rigoureusement garanties, et traiter des problèmes non linéaires. C'est pourquoi ces méthodes sont devenues très populaires récemment.

Copyright © 2010 Eyrolles.

Sommaire

1 Trois voies vers les méthodes à noyau 431
1.1 Dans l'esprit des plus proches voisins et des fenêtres de Parzen 431
1.2 Passage par un espace de redescription et astuce des noyaux 431
1.3 Approximation de fonctions dans un espace de Hilbert 437
2 Philosophie des méthodes à noyaux 441
2.1 Les méthodes linéaires remises au goût du jour 441
2.2 L'importance de la matrice de Gram 441
3 Les Séparatrices à Vastes Marges (SVM) 443
3.1 La résolution du problème d'optimisation 443
3.1.1 Formulation primale du problème 443
3.1.2 Formulation duale du problème 444
3.2 Le cas d'un échantillon non linéairement séparable dans $\mathcal{X}$ 446
3.3 Échantillon difficilement séparable et marge douce (ou poreuse) 447
3.4 Utilisation de fonctions noyau : illustrations 449
3.5 Séparateurs à Vastes Marges et généralisation 452
4 Autres types d'induction avec fonctions noyau 453
4.1 La régression 453
4.2 Induction avec une seule classe ou non supervisée 454
4.3 Induction de sorties structurées 456
5 Ingénierie des fonctions noyau 458
5.1 Les fonctions noyau : signification et capacité d'approximation 458
5.1.1 Signification des fonctions noyau 458
5.1.2 Les fonctions noyau et la capacité du RKHS associé 459
5.2 L'espace des fonctions noyau : règles simples de construction 459
5.3 Construction de fonctions noyau sur des espaces non vectoriels 465
5.3.1 Noyaux sur des ensembles 465
5.3.2 Noyaux sur des textes 466
5.3.3 Fonctions noyau sur des séquences et des chaînes de caractères 466
5.3.4 Fonctions noyau à convolution 468
5.3.5 Calcul des fonctions noyau à convolution 469
5.3.6 Fonctions noyau sur les arbres 470
5.3.7 Fonctions noyau sur et entre graphes 471
5.3.8 Les noyaux génératifs 476
5.4 Aspects calculatoires 477
6 Les méthodes à noyaux en pratique 477
6.1 Aperçu des approches calculatoires 477
6.2 Aperçu de logiciels spécifiques 479
6.3 La détermination des hyper-paramètres 480
6.4 Limites des méthodes à noyaux 482
7 Bilan et perspectives 483

Copyright © 2010 Eyrolles.

Les méthodes d'apprentissage dites **à noyaux** (*kernel-based methods*) sont actuellement très en vogue. Cela n'est pas le fruit du hasard puisqu'elles sont à la convergence de plusieurs points de vue différents.

1. Trois voies vers les méthodes à noyau

Nous décrivons dans la suite trois points de vue qui conduisent aux méthodes à noyaux. Nous les exposons en ordre d'abstraction croissante. C'est aussi, à peu près, l'ordre dans lequel les méthodes à noyaux en apprentissage ont été considérées dans les travaux de recherche.

1.1 Dans l'esprit des plus proches voisins et des fenêtres de Parzen

Historiquement, il est fait mention des approches à noyaux dès les années soixante. L'idée essentielle est d'utiliser les exemples d'apprentissage $\langle(\mathbf{x}_i, u_i)\rangle_{i=1,m}$ pour réaliser une interpolation sur l'espace des entrées $\mathcal{X}$ grâce à une pondération des sorties associées aux entrées $(\mathbf{x}_i)_{i=1,m}$. Ainsi, pour une entrée $\mathbf{x}$ inconnue, la réponse $\hat{y}(\mathbf{x})$ est estimée par :

$$\hat{y}(\mathbf{x}) = h_m(\mathbf{x}) = \frac{1}{\sum_{i=1}^{m} K(\mathbf{x}, \mathbf{x}_i)} \sum_{i=1}^{m} u_i \, K(\mathbf{x}, \mathbf{x}_i) \tag{14.1}$$

Le poids $K(\mathbf{x}, \mathbf{x}_i)$ qui est associé à chaque sortie u_i dépend de la position relative de $\mathbf{x}_i$ dans l'espace des entrées $\mathcal{X}$ et du point $\mathbf{x}$ considéré. La fonction $K(\mathbf{x}, \mathbf{x}')$ définissant ces poids est appelée, dans ces approches, *fonction noyau* ou *noyaux de Parzen* ou *fenêtres de Parzen*. Le plus souvent, la forme des fonctions noyau est du type :

$$K(\mathbf{x}, \mathbf{x}') = g\left(\frac{d(\mathbf{x}, \mathbf{x}')}{\sigma}\right)$$

où $d(\mathbf{x}, \mathbf{x}')$ est une « distance » définie sur $\mathcal{X}$, σ est un facteur d'échelle (ou d'adoucissement), et $g(\cdot)$ est une fonction (usuellement monotone) décroissante. Un choix fréquent pour cette fonction est $g(z) = \exp(-z^2/2)$. L'utilisation de ce type de fonction noyau conduit à une estimation $\hat{y}$ comme moyenne pondérée des $(u_i)_{i=1,m}$ avec un poids plus fort pour les exemples pour lesquels la distance $d(\mathbf{x}, \mathbf{x}_i)$ est petite, la notion de « petite » étant déterminée par la valeur de σ.

Ces méthodes, par interpolation, qui sont très liées aux méthodes de plus proches voisins, seront détaillées dans le chapitre 15. On peut cependant d'ores et déjà en évoquer les avantages. Elles sont un approximateur universel : lorsque le nombre d'exemples d'apprentissage croît et devient arbitrairement grand, $m \to \infty$, l'estimation (14.1) approche la fonction de prédiction optimale, $h_m(\mathbf{x}) \to h^*(\mathbf{x})$, si le paramètre d'échelle σ est une fonction de m décroissant vers 0, $\sigma(m) \to 0$, à une vitesse inférieure à $1/m$. Ce résultat est valable pour presque toute fonction de distance (sous de faibles hypothèses telles que la convexité). De plus, ces méthodes d'interpolation ne demandent pas d'apprentissage à proprement parler puisqu'*il n'y a pas d'hypothèse apprise*, mais utilisation directe de l'ensemble d'apprentissage.

1.2 Passage par un espace de redescription et astuce des noyaux

Un autre point de vue sur les méthodes à noyaux introduit la notion d'espace de redescription.

À titre d'exemple, nous considérons ici le **cas de la régression linéaire** déjà étudié dans le chapitre 9. On cherche à trouver une fonction linéaire h définie dans $\mathcal{X} = \mathbb{R}^d$ qui interpole au mieux des données d'apprentissage $\mathcal{S} = \langle(\mathbf{x}_1, u_1), \ldots, (\mathbf{x}_m, u_m)\rangle$ de points $\mathbf{x}_i$ pris dans $\mathcal{X}$ et de valeur associée $u_i \in \mathbb{R}$:

$$h(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i \, x^i \tag{14.2}$$

Copyright © 2010 Eyrolles.

où les x^i sont les coordonnées d'un point x. Nous écrirons aussi : $h(\mathrm{x}) = \langle \mathrm{w}, \mathrm{x} \rangle$ en étendant le vecteur d'entrée par l'ajout d'une composante zéro de valeur 1, ou bien $\mathbf{w} \cdot \mathbf{x} + w_0$ quand nous voudrons mettre en exergue le coefficient w_0.

Utilisons le principe inductif de minimisation du risque empirique, en prenant l'écart quadratique comme fonction de perte (voir figure 14.1) :

$$R_{\mathrm{Emp}}(h) = R_{\mathrm{Emp}}(\mathrm{w}) = \frac{1}{m}\sum_{i=1}^{m} l\big(h(\mathbf{x}_i), u_i\big) = \frac{1}{m}\sum_{i=1}^{m} (h(\mathbf{x}_i) - u_i)^2 = \frac{1}{m}\sum_{i=1}^{m} \xi_i^2 \qquad (14.3)$$

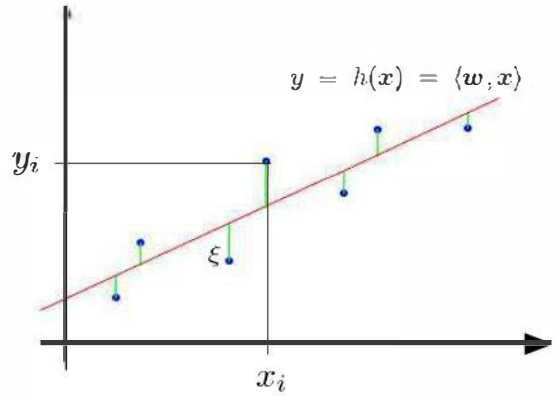


FIG. 14.1: *Régression linéaire.*

On cherche le vecteur de paramètres w minimisant ce risque empirique. Écrivons ξ le vecteur des écarts ξ_i : $\xi = \mathbf{y} - \mathbf{X}\mathbf{w}$.

On a alors : $R_{\mathrm{Emp}}(\mathrm{w}) = \|\xi\|_2^2 = (y - \mathbf{X}\mathbf{w})^\top (y - \mathbf{X}\mathbf{w})$.

On peut chercher la valeur optimale de w en calculant la dérivée du risque par rapport au vecteur de paramètres w et en la posant égale au vecteur nul :

$$\frac{\partial R_{\mathrm{Emp}}(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}^\top y + 2\mathbf{X}^\top \mathbf{X}\mathbf{w}$$

On obtient alors les équations dites « normales » : $\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top y$.

Si l'inverse de $\mathbf{X}\mathbf{X}^\top$ existe, la solution peut s'exprimer sous la forme :

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-2}\mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \alpha$$

ce qui signifie que w est une combinaison linéaire des points d'apprentissage : $\mathrm{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i$.

L'hypothèse recherchée a alors la forme, appelée ***représentation duale*** :

$$h(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle \qquad (14.4)$$

qui a la particularité de *ne dépendre que de produits scalaires* entre vecteurs de $\mathcal{X}$.

Ce qui est remarquable est que l'on peut retrouver (voir par exemple [STC04], pp.31-32) la même forme pour l'hypothèse h si l'on cherche à minimiser le *risque empirique régularisé* dans

Copyright © 2010 Eyrolles.

lequel on favorise les fonctions de norme faible :

$$R(h) = \frac{1}{m}\sum_{i=1}^{m}(u_i - h(\mathbf{x}_i))^2 + \lambda \min_{\mathbf{w}} ||\mathbf{w}||^2$$

On trouve en effet : $\mathbf{w} = \lambda^{-1}\mathbf{X}^\top(\mathbf{u} - \mathbf{X}\mathbf{w}) = \mathbf{X}^\top\alpha$, avec $\alpha = \lambda^{-1}(\mathbf{u} - \mathbf{X}\mathbf{w})$.
Soit encore : $(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_m)\alpha = \mathbf{y}$, d'où :

$$\boxed{\alpha = (\mathbf{K} + \lambda\mathbf{I}_m)^{-1}\mathbf{u}} \tag{14.5}$$

où $\mathbf{K}$ est la matrice de Gram (voir plus loin).

Ce **résultat** est **important** car, généralisé (il le sera dans le *théorème de représentation*, section 1.3), il montre que *les problèmes régularisés* de la forme :

$$R(h) = \frac{1}{m}\sum_{i=1}^{m}(u_i - h(\mathbf{x}_i))^2 + \lambda \mathrm{Reg}(h)$$

ont naturellement des solutions de la forme : $h(\mathbf{x}) = \sum_{i=1}^{m}\alpha_i\langle\mathbf{x}_i, \mathbf{x}\rangle$, c'est-à-dire d'une combinaison linéaire de produits scalaires.

Supposons que l'on cherche à résoudre cette fois-ci un **problème de régression non linéaire** dans $\mathcal{X}$ (voir figure 14.2), on peut être tenté d'utiliser pour cela une fonction de redescription ϕ de $\mathcal{X}$ dans un nouvel espace F (*feature space*), de telle manière que le problème devienne un problème de régression linéaire dans ce nouvel espace.

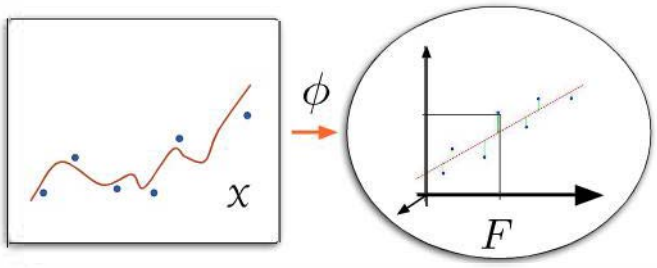


FIG. 14.2: *Régression non linéaire rendue possible par la transformation en un problème de régression linéaire dans l'espace de redescription F.*

Dans ce cas, la solution du problème de régression linéaire, dans l'espace de redescription F, prendra la *forme primale* $h(\mathbf{x}) = \langle\mathbf{w}, \phi(\mathbf{x})\rangle = 0$, que l'on peut ré-exprimer en mettant en évidence la base de fonctions non linéaires $\{\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots\}$, éventuellement infinie, de l'espace F :

$$h(\mathbf{x}) = \sum_{j=1}^{\infty} w_j\,\phi_j(\mathbf{x}) + w_0 = 0 \tag{14.6}$$

En résolvant comme précédemment, cette fois-ci dans l'espace de redescription, on obtient à nouveau une *expression duale* :

$$\boxed{h(\mathbf{x}) = \sum_{i=1}^{m}\alpha_i\langle\Phi(\mathbf{x}_i), \Phi(\mathbf{x})\rangle} \tag{14.7}$$

Copyright © 2010 Eyrolles.

Et, en rapprochant les équations (14.6) et (14.7), on tire :

$$h(\mathbf{x}) = \sum_{j=1}^{\infty} w_j \, \phi_j(\mathbf{x}) + w_0 = \mathbf{w}^\top \cdot \Phi(\mathbf{x})$$
$$= \sum_{i=1}^{m} \alpha_i \, \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) = 0$$

d'où :

$$\boxed{\mathbf{w}^\top = \sum_{i=1}^{m} \alpha_i \, \Phi(\mathbf{x}_i)} \tag{14.8}$$

Un problème est évidemment que la nouvelle formulation implique, d'une part, de trouver une fonction de redescription Φ adéquate, et, d'autre part, de calculer des produits scalaires dans un espace F dont la dimension peut être grande (voire infinie). Cependant, il peut parfois être possible de ne pas avoir à effectuer explicitement ces produits scalaires dans F grâce à l'utilisation de *fonctions noyau*. C'est ce que l'on appelle l'« astuce des noyaux » (*kernel trick*).

Définition 14.1 (Fonction noyau, *Kernel function*)

Une fonction noyau *est une fonction* $\kappa : \mathbf{x}, \mathbf{x}' \in \mathcal{X}^2 \rightarrow \mathbb{R}$ *satisfaisant :*

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$$

où Φ *est une fonction de* $\mathcal{X}$ *vers un espace de redescription* F *doté d'un produit scalaire :*

$$\Phi : \mathbf{x} \mapsto \Phi(\mathbf{x}) \in F$$

L'utilisation de fonctions noyau permet ainsi de calculer implicitement un produit scalaire dans un espace de dimension éventuellement infini par un calcul n'impliquant qu'un nombre fini de termes, ce nombre étant le nombre m des exemples d'apprentissage (voire éventuellement moins, comme nous le verrons). Par ailleurs, il apparaît donc que la spécification de la fonction noyau est *suffisante*. Il n'est pas nécessaire de calculer le vecteur poids $\mathbf{w}$ pour spécifier la fonction de décision $h(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \, \kappa(\mathbf{x}_i, \mathbf{x})$.

En un sens profond, **une fonction noyau correspond à une mesure de similarité** entre entrées $\mathbf{x}$ et $\mathbf{x}'$, mesure qu'il est naturel d'utiliser en induction puisqu'un *a priori* évident est de supposer que deux entrées similaires doivent être associées à des sorties similaires. De fait, les fonctions noyaux peuvent être considérées comme une *généralisation des fonctions de covariance*. En effet, une fonction noyau peut s'exprimer comme une somme de produits sur un ensemble de fonctions :

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \phi_i(\mathbf{x}) \, \phi_i(\mathbf{x}')$$

ce qui suggère de considérer les fonctions noyau comme des fonctions de covariance dépendante d'une mesure de probabilité μ sur une classe de fonctions $\mathcal{F}$:

$$\kappa_\mu(\mathbf{x}, \mathbf{x}') = \int_{\mathcal{F}} f(\mathbf{x}) f(\mathbf{x}') \mu(f) \mathrm{d}f$$

Copyright © 2010 Eyrolles.

On appellera ces fonctions noyau des *noyaux de covariance*. On peut montrer que toute fonction noyau peut être obtenue comme noyau de covariance avec une mesure μ particulière sur $\mathcal{F}$.

Étant donné un ensemble de points $\{\mathbf{x}_i | i = 1, \ldots, m\}$, nous pouvons calculer la *matrice de Gram* (ou *matrice noyau*) $\mathbf{K}$ dont les éléments sont : $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

Une matrice réelle $\mathbf{K}$ de dimension $m \times m$ vérifiant la forme quadratique $Q(\mathbf{v}) = \mathbf{v}^\top \mathbf{K}\mathbf{v} \geq 0$ pour tout vecteur $\mathbf{v} \in \mathbb{R}^m$ est dite *positive semi-définie*. Si $Q(\mathbf{v}) = 0$ seulement quand $\mathbf{v} = 0$, alors la matrice $\mathbf{K}$ est dite positive définie. De manière équivalente, une matrice symétrique est positive semi-définie si et seulement si toutes ses valeurs propres sont positives ou nulles.

Une matrice de Gram correspondant à une fonction de covariance doit être positive semi-définie.

Une *fonction noyau* est dite *positive semi-définie* si

$$\int \kappa(\mathbf{x}, \mathbf{x}')\, f(\mathbf{x}) f(\mathbf{x}')\, d\mu(\mathbf{x}) d\mu(\mathbf{x}') \geq 0 \tag{14.9}$$

où μ et μ' dénotent des mesures sur $\mathcal{X}$ et pour toute fonction $f \in L_2(\mathcal{X}, \mu)$[1]

De manière équivalente, une fonction noyau définissant une matrice de Gram positive semi-définie pour tout ensemble de points d'entrée $\mathcal{S}_m$ et pour tout $m \in \mathbf{N}$ est positive semi-définie.

Jusqu'à présent, le seul critère que nous ayons énoncé pour savoir si une fonction symétrique $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est une fonction noyau était celui d'exhiber un espace de redescription F et de vérifier que $\kappa(\cdot, \cdot)$ correspondait à un produit scalaire dans F. Plusieurs théorèmes permettent de caractériser les fonctions noyau sans passer explicitement par l'espace de redescription. Ainsi :

Théorème 14.1 (Les fonctions symétriques positives semi-définies sont des noyaux)

Une fonction $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est une fonction noyau si et seulement si elle est symétrique et positive semi-définie.

Une démonstration peut être trouvée dans [SC08], p.118.

Un autre théorème, le **théorème de Mercer**, fournit des conditions pour qu'une fonction symétrique $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ soit une fonction noyau. Il permet en outre d'identifier une décomposition spectrale des fonctions noyau.

L'appellation *fonction noyau* provient de leur rôle dans des équations très importantes en physique, les *équations intégrales* du type :

$$f(x) = \lambda \int_a^b K(x, y)\, f(y)\, dy + h(x)$$

dans lesquelles l'inconnue est la fonction $f(x)$, et $K(x,y)$ est appelée le noyau de l'équation intégrale. Les fonctions propres de cette équation dépendent des propriétés de la fonction noyau. Si celle-ci est symétrique, $K(x,y) = K(y,x)$ (ce qui correspond en un sens profond à une conservation de l'énergie du système décrit), alors on peut prouver que les valeurs propres cherchées λ_k sont réelles et que les fonctions propres ψ_k associées sont orthogonales. Cela conduit évidemment

[1] Une fonction f définie de $\mathbb{R}$ sur $\mathbb{R}$ est dite de carré sommable par rapport à la mesure μ ($f \in L_2(\mathcal{X}, \mu)$)définie sur $\mathbb{R}$ si sa *norme* associée $||f||$ définie par :

$$||f||^2 = \int_a^b [f(x)]^2\, \mathrm{d}\mu(x)$$

est bornée sur $\mathbb{R}$ tout entier.

Copyright © 2010 Eyrolles.

à *pouvoir représenter chaque fonction par une série de fonctions propres* (cette propriété permet d'envisager de réaliser une analyse en composantes principales généralisée). On peut aussi réécrire l'équation :

$$f(x) = \lambda \int_a^b K(x,y)\, f(y)\, dy$$

comme

$$f = \lambda A f$$

où A est un opérateur linéaire, représentable par une matrice.

Le **théorème de Mercer** permet d'exprimer les fonctions noyau en termes de valeurs propres et de fonctions propres.

Une fonction $\phi(\cdot)$ vérifiant l'équation intégrale

$$\int \kappa(\mathbf{x}, \mathbf{x}')\, \phi(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x}) = \lambda\, \phi(\mathbf{x}')$$

est appelée une *fonction propre* du noyau κ associée à la valeur propre $\lambda \in \mathbb{R}$ en fonction de la mesure μ. Il existe en général une infinité de fonctions propres, que nous noterons $\phi_1(\mathbf{x}), \phi_2(\mathbf{x}) \ldots$ en supposant l'ordre $\lambda_1 \geq \lambda_2 \geq \ldots$ Les fonctions propres sont orthogonales en fonction de μ et peuvent être normalisées de telle manière que $\int \phi_i(\mathbf{x})\, \phi_i(\mathbf{x})\, d\mu(\mathbf{x}) = \delta_{ij}$ où δ_{ij} est le symbole de Kronecker égal à 1 seulement quand $i = j$.

Théorème 14.2 (Théorème de Mercer)

Si $\kappa(\cdot,\cdot)$ est une fonction noyau continue symétrique d'un opérateur intégral

$$g(\mathbf{y}) = A f(\mathbf{y}) = \int_a^b \kappa(x,y)\, f(y)\, dy + h(x)$$

vérifiant :

$$\int_{\mathcal{X}\times\mathcal{X}} \kappa(\mathbf{x}, \mathbf{x}')\, f(\mathbf{x})\, f(\mathbf{x}')\, d\mathbf{x} d\mathbf{x}' \geq 0$$

pour toute fonction $f \in L_2(\mathcal{X})$ (de carré sommable) ($\mathcal{X}$ étant un sous-espace compact de $\mathbb{R}^d$), alors la fonction $\kappa(\cdot,\cdot)$ peut être développée en une série uniformément convergente en fonction des valeurs propres positives λ_i et des fonctions propres ψ_i :

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{N} \lambda_j\, \psi_j(\mathbf{x})\, \psi_j(\mathbf{x}') \tag{14.10}$$

où N est le nombre de valeurs propres positives (nombre éventuellement infini).

On peut alors décrire la fonction Φ de redescription des entrées comme :

$$\Phi(\mathbf{x}) = (\sqrt{\lambda_1}\, \psi_1(\mathbf{x}), \sqrt{\lambda_2}\, \psi_2(\mathbf{x}), \ldots) \tag{14.11}$$

Cette décomposition est l'analogue en dimension infinie de la diagonalisation d'une matrice Hermitienne. Le taux de décroissance des valeurs propres λ_i fournit une information sur la régularité du noyau. Plus cette décroissance est rapide, plus régulière est la fonction noyau.

Copyright © 2010 Eyrolles.

Si les conditions théoriques exposées ci-dessus sont intéressantes, elles ne sont pas faciles à vérifier en pratique. La plupart des praticiens utilisent l'une des fonctions noyau connues ou bien font usage de propriétés particulières de combinaisons de fonctions noyau pour construire de nouvelles fonctions noyau adaptées à leur problème (voir section 5 de ce chapitre).

Il n'y a pas de bijection entre les fonctions noyau et les espaces de redescription pour lesquelles ces fonctions sont analogues à un produit scalaire. *Une fonction noyau peut ainsi être associée à plusieurs espaces de redescription.*

EXEMPLE Non bijection entre fonction noyau et espace de redescription

Soit un espace d'entrée $\mathcal{X} = \mathbb{R}^2$, et la projection $\Phi_1 : (x^1, x^2) \mapsto \left((x^1)^2, (x^2)^2, x^1x^2, x^2x^1\right)$, où x^i dénote la i-ème composante du vecteur $\mathbf{x}$.

Alors le produit scalaire dans F s'écrit :

$$\langle \phi_1(\mathbf{x}, \phi_1(\mathbf{x}')\rangle = (x^1)^2(x'^1)^2 + (x^2)^2(x'^2)^2 + 2x^1x^2x'^1x'^2 = \langle \mathbf{x}, \mathbf{x}'\rangle^2$$

La fonction noyau $\kappa(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}'\rangle^2$ correspond donc à un produit scalaire dans l'espace F à quatre dimensions défini par la projection Φ_1.

Mais il est facile de voir qu'elle correspond aussi à un produit scalaire dans l'espace de redescription F à trois dimensions défini par la projection $\Phi_2 : (x^1, x^2) \mapsto \left((x^1)^2, (x^2)^2, \sqrt{2}x^1x^2\right)$. En effet :

$$\langle \phi_2(\mathbf{x}, \phi_2(\mathbf{x}')\rangle = (x^1)^2(x'^1)^2 + (x^2)^2(x'^2)^2 + 2x^1x^2x'^1x'^2 = \langle \mathbf{x}, \mathbf{x}'\rangle^2$$

1.3 Approximation de fonctions dans un espace de Hilbert

Le point de vue dominant en apprentissage considère celui-ci essentiellement comme l'approximation d'une fonction cible inconnue à partir d'informations obtenues par les observations. Il est donc naturel de consulter les idées et résultats de l'analyse fonctionnelle. Celle-ci a en particulier pour but d'étudier comment caractériser ou calculer des fonctions à partir de bases (éventuellement infinies) de fonctions. L'exemple le plus célèbre est celui de l'analyse de Fourier (1772-1837) qui permet d'associer à toute fonction stationnaire continue une série (une somme) de fonctions périodiques (sinus et cosinus). On décompose ainsi une fonction selon ses fréquences, de même qu'un prisme décompose la lumière en couleurs. Le théorème de Mercer, vu à la section précédente, en est une illustration.

Plus généralement, une fonction est définie par l'ensemble des valeurs qu'elle prend, soit une infinité de nombres. Une fonction peut donc être vue comme un point dans un espace de dimension infinie. Mais ce point de vue n'est généralement pas très révélateur sur la nature de la fonction. On cherche alors à caractériser une fonction à l'aide de fonctions analysantes (fonctions trigonométriques, ondelettes, ...) prises dans une base. Chaque fonction analysante détermine un axe de projection de la fonction à analyser, et décomposer une fonction dans une base choisie revient à la représenter par une infinité de coordonnées en la projetant sur une infinité d'axes. Ce langage n'est utilisable que si les notions de longueur et d'angle gardent une signification dans des espaces de dimension élevée (infinie), ce qui n'est pas évident. L'étude des conditions sous lesquelles cela est possible a conduit à la théorie des espaces de Hilbert (1862-1943). Un **espace de Hilbert** est essentiellement une extension algébrique de la notion d'espace euclidien ordinaire dans laquelle l'espace est défini par une liste infinie de fonctions orthogonales (analogues aux axes d'un espace euclidien) et où chaque point de cet espace peut être décrit par une combinaison linéaire de fonctions de bases.

Copyright © 2010 Eyrolles.

L'exemple le plus classique d'un espace de Hilbert est celui des *séries de Fourier* permettant de représenter n'importe quelle fonction $f(\phi)$ dans le domaine $0 \leq \phi \leq 2\pi$ comme une série :

$$f(\phi) = \sum_{n=0}^{\infty} a_n e^{in\phi}$$

où les fonctions $e^{in\phi}$ servent de fonctions de bases orthogonales (on a $\int_0^{2\pi} e^{in\phi} e^{-im\phi} d\phi = 2\pi\delta_{nm}$ avec $\delta_{nm} = 1$ si $n = m$ et 0 sinon) et où les a_n sont les coordonnées par rapport à cette base.

Définition 14.1 (Espace de Hilbert)

Un espace de Hilbert *est un espace vectoriel sur* $\mathbb{R}$ *ou* $\mathbb{C}$*, muni d'un produit scalaire dont l'espace normé associé est complet. Dans le cas des espaces fonctionnels de fonctions numériques continues sur un intervalle* $[a, b]$ *de* $\mathbb{R}$*, le produit scalaire utilisé est généralement :*

$$\langle f, g \rangle = \int_a^b f(x)\, g(x)\, dx$$

La norme associée $||f||$ *est alors définie par :*

$$||f||^2 = \int_a^b [f(x)]^2\, dx$$

Lorsque la norme est bornée sur $\mathbb{R}$ tout entier, f^2 est intégrable et on parle de *fonctions de carrés intégrables*. Lorsque l'intégrale considérée est celle de Lebesgue (1875-1941) (ce qui permet de traiter des fonctions non continues), cet espace se note $L_2[a, b]$.

Dans notre cas, nous considérerons les produits scalaires et normes par les opérations définies sur $\mathcal{X}$: $\langle f, g \rangle = \int_{\mathcal{X}} f(\mathbf{x})\, g(\mathbf{x})\, d\mathbf{x}$ et $||f||^2 = \int_{\mathcal{X}} [f(\mathbf{x})]^2\, d\mathbf{x}$. Nous pourrons aussi introduire une mesure μ sur $\mathcal{X}$, ce qui conduit au produit scalaire et à la norme suivants : $\langle f, g \rangle = \int_{\mathcal{X}} f(\mathbf{x})\, g(\mathbf{x})\, d\mu\mathbf{x}$ et $||f||^2 = \int_{\mathcal{X}} [f(\mathbf{x})]^2\, d\mu\mathbf{x}$.

En plus d'être muni d'un produit scalaire, un espace de Hilbert est séparable et complet[2].

Tout espace de redescription muni d'un produit scalaire et complet et séparable peut être doté d'une base. Dans le cas d'un espace de Hilbert fonctionnel, il s'agit d'une base de fonctions.

Pour récapituler, une fonction noyau est associée à un ou plusieurs espaces de redescription muni(s) d'un produit scalaire. Une condition nécessaire et suffisante pour qu'une fonction $\kappa(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ soit une fonction noyau est qu'elle soit positive semi-définie ou qu'elle vérifie la condition de Mercer. Le théorème de Mercer (14.2) fournit en plus une description explicite de l'espace de redescription par une base de fonctions analysantes orthogonales. Ce n'est cependant pas la seule base de fonctions analysantes possibles. Une autre base (qui peut parfois coïncider avec la base de Mercer) est donnée par les *noyaux reproduisants*.

[2] Un espace de fonctions $\mathcal{F}$ est *complet* si toute séquence de Cauchy $\{f_n\}_{n \geq 1}$ d'éléments de $\mathcal{F}$ converge vers un élément $f \in \mathcal{F}$, une séquence de Cauchy satisfaisant la propriété que :

$$\sup_{m > n} ||f_n - f_m|| \rightarrow 0, \text{quand } n \rightarrow \infty$$

Un espace $\mathcal{F}$ est *séparable* s'il existe un ensemble énumérable d'éléments de $\mathcal{F}$ $f_1, \ldots, f_i, \ldots$ de $\mathcal{F}$ tel que pour tout élément $f \in \mathcal{F}$ et tout $\varepsilon > 0$ il existe une fonction f_i telle que : $||f_i - f|| < \varepsilon$.

Copyright © 2010 Eyrolles.

Espace de Hilbert à noyau reproduisant (RKHS ou *Reproducing Kernel Hilbert Space).*

Étant donnée une fonction noyau, il est possible de lui associer un espace de Hilbert de fonctions (suffisamment) régulières. En un certain sens, cet espace est l'espace de fonctions minimal associé à la fonction noyau considérée et peut donc servir d'espace de redescription canonique.

Définition 14.2 (Espace de Hilbert à noyau reproduisant)

Soit $\mathcal{F}$ un espace de Hilbert de fonctions réelles définies sur un ensemble indexé $\mathcal{X}$:

$$\mathcal{F} = \left\{ \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}_i, \cdot) : m \in \mathbb{N}, \mathbf{x}_i \in \mathcal{X}, \alpha_i \in \mathbb{R}, i = 1, \ldots, m \right\}$$

$\mathcal{F}$ est appelé espace de Hilbert à noyau reproduisant *doté d'un produit scalaire noté $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ (et de la norme $||f||_{\mathcal{F}} = \sqrt{\langle f, f \rangle_{\mathcal{F}}}$) s'il existe une fonction $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ ayant les propriétés suivantes :*

1. *pour tout élément $\mathbf{x} \in \mathcal{X}, \kappa(\mathbf{x}, \cdot)$ appartient à $\mathcal{F}$, et*
2. *la fonction κ est une* fonction noyau reproduisante, *c'est-à-dire telle que pour toute fonction $f \in \mathcal{F}$, on a : $\langle f, \kappa(\mathbf{x}, \cdot) \rangle_{\mathcal{F}} = \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) = f(\mathbf{x})$.*

Le fait que la fonction noyau soit *reproduisante* signifie que toute fonction $f \in \mathcal{F}$ est égale à un produit scalaire qui est aussi une combinaison linéaire finie de fonctions de base (voir par exemple [Aub87], pp.137-141, pour les détails et une démonstration).

Le *produit scalaire* sur $\mathcal{F}$ est alors défini comme suit. Soient les fonctions $f, g \in \mathcal{F}$ définies par :

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) \quad \text{et} \quad g(\mathbf{x}) = \sum_{i=1}^{n} \beta_j \kappa(\mathbf{z}_j, \mathbf{x})$$

alors :

$$\langle f, g \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \beta_j \kappa(\mathbf{x}_i, \mathbf{x}_j) = \sum_{i=1}^{m} \alpha_i g(\mathbf{x}_i) = \sum_{j=1}^{n} \beta_j f(\mathbf{x}_j) \tag{14.12}$$

Il est intéressant de noter qu'alors que la base de Mercer dépend de la mesure μ définie sur $\mathcal{X}$, ce n'est pas le cas de la base des noyaux reproduisants qui ne dépend que de la fonction noyau.

Par ailleurs, l'espace de Hilbert des fonctions L_2 (de produit scalaire $\langle f, g \rangle_{L_2} = \int f(\mathbf{x}) g(\mathbf{x}) \mathrm{d}\mathbf{x}$) contient de nombreuses fonctions irrégulières. Dans L_2, la fonction de Dirac δ est la fonction de représentation, i.e. $f(\mathbf{x}) = \int f(\mathbf{x}') \delta(\mathbf{x} - \mathbf{x}') \mathrm{d}\mathbf{x}'$. Les fonctions noyaux jouent un rôle analogue à la fonction δ dans les espaces de Hilbert à noyau reproduisant, qui sont de ce fait plus réguliers, leur degré de régularité dépendant de la régularité de la fonction noyau associée. En fait, on peut montrer de manière plus générale que les propriétés de mesurabilité, de continuité et de différentiabilité des fonctions de $\mathcal{F}$ dépendent des propriétés associées de la fonction noyau correspondante. Un résultat fondamental étant effectivement le théorème suivant dû à Moore et Aronszajn en 1950.

Théorème 14.3 (Bijection entre noyau et RKHS associé)

Soit $\mathcal{X}$ un ensemble énumérable, alors pour toute fonction positive semi-définie $\kappa(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, il existe un unique espace de Hilbert avec ce noyau reproduisant, et vice versa.

Une caractéristique cruciale de ces fonctions analysantes $\kappa(\mathbf{x}_i, \cdot)$ est donc qu'elles *ne sont pas issues d'un dictionnaire donné* a priori, comme c'est le cas par exemple des fonctions trigonomé-

Copyright © 2010 Eyrolles.

triques ou des ondelettes, mais qu'elles *dépendent directement des points* x de l'espace $\mathcal{X}$. Cela est capital dans l'estimation de fonctions à partir de points particuliers, car, dans ce cas, les fonctions analysantes utilisées pour estimer la fonction vont dépendre des points connus $\{\mathbf{x}_i\}_{1\leq i\leq m}$.

La fonction estimée prendra donc la forme :

$$h(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i) \tag{14.13}$$

Il devient alors naturel de se demander quelle est la qualité de l'approximation fournie par h vis-à-vis d'une fonction cible f en fonction du nombre m de données et de la « représentativité » de l'échantillon $\{(\mathbf{x}_i)\}_{i=1,m}$ (on notera ici l'absence de référence explicite aux réponses mesurées u_i. Celles-ci interviendront naturellement dans l'expression des coefficients α_i).

Il est évidemment impossible de répondre à cette question, mal posée, sans hypothèses additionnelles. Cette connaissance *a priori* prend souvent la forme d'une préférence pour des fonctions hypothèses régulières ou issues d'un espace d'hypothèses de capacité limitée, comme nous l'avons vu au chapitre 2. Le théorème de représentation établit un lien fondamental entre les critères inductifs régularisés, réglant le compromis biais-variance, et les espaces hilbertiens à noyau reproduisant.

Critère inductif régularisé et espace d'hypothèses

On utilise généralement une version régularisée du risque empirique, de la forme :

$$\hat{h}(\mathbf{x}) = \underset{h\in\mathcal{H}}{\text{ArgMin}} \left\{ R_{\text{Emp}}(h, \mathcal{S}) + \lambda \,\text{Capacité}(\mathcal{H}) \right\} \tag{14.14}$$

ou de la forme :

$$\hat{h}(\mathbf{x}) = \underset{h\in\mathcal{H}}{\text{ArgMin}} \left\{ R_{\text{Emp}}(h, \mathcal{S}) + \lambda \,\text{Reg}(h) \right\} \tag{14.15}$$

où *Capacité* est un estimateur de la capacité de l'espace d'hypothèses à offrir des hypothèses de faible risque empirique quel que soit l'échantillon d'apprentissage, et où *Reg* est une fonction mesurant la « régularité » de l'hypothèse (par exemple $\text{Reg}(h) = ||h||_{\mathcal{H}_\kappa}$, avec $||h||^2_{\mathcal{H}_\kappa} := \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j \kappa(\mathbf{x}_i,\mathbf{x}_j)$).

Un théorème essentiel affirme que **toute fonction $\hat{h}$ minimisant un risque empirique régularisé admet une représentation de la forme** :

$$\hat{h}(\cdot) = \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}_i, \cdot) \tag{14.16}$$

Plus précisément (voir, par exemple, |Her02| pp.48-49 et 257-258 pour la preuve, ou |STV04|, pp.47-48) :

Théorème 14.4 (Théorème de représentation, *representer theorem*)

Soit un noyau reproduisant $\kappa : \mathcal{X}\times\mathcal{X}\to\mathbb{R}$, un échantillon d'apprentissage $S_m \in (\mathcal{X}\times\mathcal{Y})^m$, et un risque empirique quelconque R_{Emp} (muni d'une fonction de perte ℓ). Soit $\text{Reg} : \mathbb{R}\to[0,\infty[$ *une fonction croissante strictement monotone. Soit $\mathcal{H}_\kappa$ l'espace hilbertien induit par le noyau reproduisant κ. Alors, toute fonction $\hat{h}\in\mathcal{H}_\kappa$ minimisant le risque régularisé :*

$$\hat{h}(\mathbf{x}) = \underset{h\in\mathcal{H}}{\text{ArgMin}} \left\{ R_{\text{Emp}}(h, \mathcal{S}) + \lambda \,\text{Reg}(h) \right\}$$

admet une représentation de la forme : $\hat{h}(\cdot) = \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}_i, \cdot) \quad \alpha_i \in \mathbb{R}, \forall i.$

Copyright © 2010 Eyrolles.

Cela signifie que la ou les hypothèses minimisant un risque empirique régularisé se trouve(nt) dans le sous-espace vectoriel des fonctions engendrées par les combinaisons linéaires de fonctions noyau (analysantes) dépendantes des données d'apprentissage $\kappa(\mathbf{x}_i, \cdot)$. Il s'ensuit en particulier que la solution $\hat{h}$ est prise dans un espace de dimension m au maximum, même si l'espace $\mathcal{H}_\kappa$ est lui-même de dimension éventuellement infinie. De plus, bien entendu, il y aura un lien fort entre la forme des fonctions noyau κ et le type de régularisation imposée dans le critère inductif.

Pour résumer, trois motivations différentes, à savoir :

- l'adaptation de la méthode des plus proches voisins ;
- le passage par un espace de redescription et la dépendance des résultats sur des produits scalaires, et donc *in fine* sur des fonctions noyau ;
- l'approche de l'analyse fonctionnelle et le théorème de représentation

conduisent à considérer l'espace des hypothèses $\mathcal{H}$ de la forme : $h(\cdot) = \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}_i, \cdot)$.

2. Philosophie des méthodes à noyaux

2.1 Les méthodes linéaires remises au goût du jour

La section 1.2 a fourni une première illustration de l'utilisation de fonctions noyau comme moyen de transformer la recherche de régularités non linéaires en recherche de régularités linéaires grâce au passage par un espace de redescription virtuel F (voir la figure 14.1). **Quatre grandes idées** sous-tendent l'approche générale :

1. Les données décrites dans l'espace d'entrée $\mathcal{X}$ sont projetées dans un espace vectoriel de redescription F
2. Des régularités linéaires sont cherchées dans cet espace F
3. Les algorithmes de recherche n'ont pas besoin de connaître les coordonnées des projections des données dans F, mais seulement leurs produits scalaires
4. Ces produits scalaires peuvent être calculés efficacement grâce à l'utilisation de fonctions noyau.

Cette approche permet ainsi d'employer tout l'arsenal des méthodes linéaires développées jusque dans les années soixante pour découvrir des relations non linéaires dans les données. Au fil des années récentes ont été ainsi revisitées l'analyse en composantes principales (ACP), la méthode des filtres de Kalman, des méthodes linéaires de clustering, la discrimination linéaire de Fisher, etc.

La figure 14.3 résume les étapes génériques de la méthode. Il est crucial de réaliser que, dans cette approche, l'essentiel des informations sur les données, en dehors de leurs étiquettes, s'exprime dans la *matrice noyau* K, ou encore *matrice de Gram*, qui encode les produits scalaires entre les projections des données d'apprentissage $\mathbf{G} = \langle \Phi(\mathbf{X}), \Phi(\mathbf{X}^\top) \rangle$.

2.2 L'importance de la matrice de Gram

La matrice de Gram (ou matrice noyau) contient toute l'information utilisée par les méthodes à noyaux sur les données. Elle est symétrique : $\mathbf{G}_{ij} = \mathbf{G}_{ji}$. L'utilisation exclusive des informations contenues dans cette matrice à propos des données d'apprentissage a pour conséquence qu'une partie des informations sur ces données est perdue. Par exemple, cette matrice est invariante par

Copyright © 2010 Eyrolles.

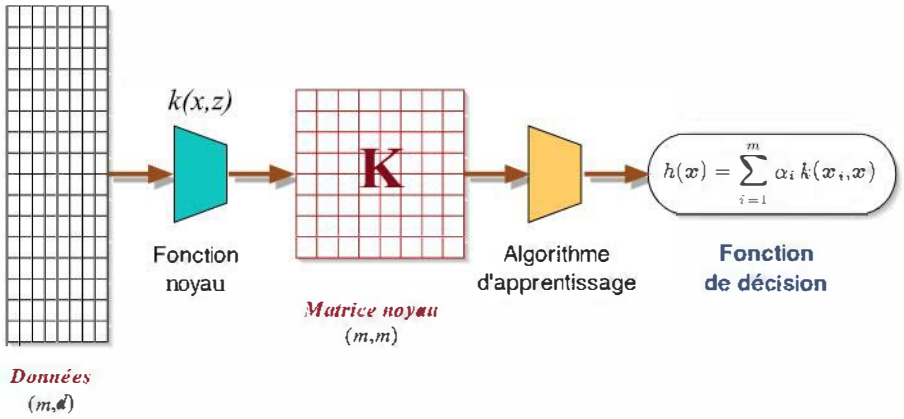


FIG. 14.3: *Chaîne de traitements générique des méthodes à noyaux.*

rotation des points dans l'espace d'entrée $\mathcal{X}$. La position angulaire des données par rapport à un référentiel est donc ignorée, de même que les alignements éventuels des données sur les axes.

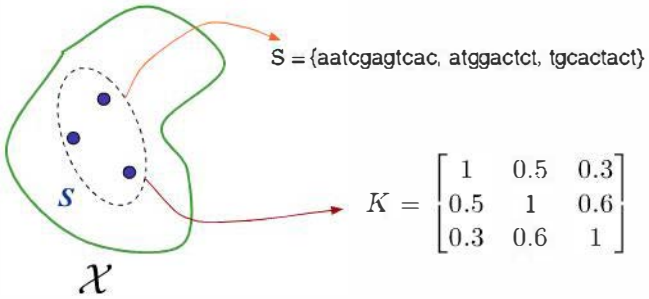


FIG. 14.4: *Les méthodes à noyau remplacent les informations sur les objets comparés par la matrice de leurs distances : la matrice de Gram. C'est un changement de perspective radical.*

Étant donné le caractère fondamental de la matrice de Gram dans la traduction des données, il n'est pas surprenant que l'on puisse estimer certaines propriétés de l'apprentissage à partir de l'examen des caractéristiques de cette matrice. De même, il est possible de modifier l'apprentissage en effectuant certaines opérations sur la matrice de Gram préalablement à l'apprentissage.

Ainsi, s'il existe une structure dans les données, elle doit se refléter dans la matrice de Gram, si, du moins, la fonction noyau choisie est appropriée pour détecter les similarités sous-jacentes. Cette fonction noyau agit de fait comme un filtre sensible à certaines « fréquences » et pas à d'autres. Par exemple, si la fonction noyau est mal choisie, ou si les données ne présentent pas de structures, les éléments de la matrice de Gram auront des valeurs indifférenciées. Toute entrée est alors proche de n'importe quel autre point, et tous les éléments sont classés dans une seule classe. C'est du sous-apprentissage sévère. Dans les cas où les éléments hors diagonale sont de valeur proche de 0, seul un apprentissage par cœur devient possible.

Inversement, un moyen d'influencer l'apprentissage est de modifier la matrice de Gram. Ainsi, par exemple, en vertu de l'équation (14.5), l'ajout d'une constante sur les termes diagonaux de la matrice de gram **G** revient à accentuer la régularisation.

Copyright © 2010 Eyrolles.

3. Les Séparatrices à Vastes Marges (SVM)

Nous nous intéressons ici à la *classification binaire* c'est-à-dire à l'approximation de fonctions de décision permettant de distinguer entre deux classes, et ceci par des hyperplans $h : \mathcal{X} \to \mathcal{Y}$, d'équation $h(\mathbf{x}) = \langle \mathbf{w} , \mathbf{x} \rangle + w_0 = \sum_{i=0}^{d} w_i x_i = 0$, en supposant que $\mathcal{X}$ soit de dimension d (par exemple, $\mathcal{X} = \mathbb{R}^d$). On utilisera alors la fonction de décision $g(\mathbf{x}) = \text{signe}\big(\langle \mathbf{w} , \mathbf{x} \rangle + w_0\big)$ pour prédire la classe ($+$ ou $-$) de l'entrée $\mathbf{x}$.

Selon le principe de minimisation du risque empirique classique, un tel « meilleur » hyperplan sera celui qui minimise le nombre d'exemples mal classés par cette fonction de décision dans l'échantillon d'apprentissage $\mathcal{S} = \langle (\mathbf{x}_1, u_1), \ldots, (\mathbf{x}_m, u_m) \rangle$. Ce critère conduit par exemple à l'algorithme du perceptron (voir section 3.3).

Vapnik et ses collègues [BGV92] ont proposé de s'intéresser plutôt à un critère de *confiance* ou encore de *robustesse* de la fonction de décision obtenue. Ce critère découle de l'étude de la théorie statistique de l'apprentissage conduite en particulier par Vapnik et Chervonenkis depuis les années soixante [VC71]. Un moyen de traduire la robustesse de la fonction de décision, ou encore du classifieur, est de considérer la *marge* séparant les exemples de la classe '+' des exemples de la classes '−'.

La distance d'un point $\mathbf{x}'$ à l'hyperplan d'équation $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$ est égale à $\langle \mathbf{w}, \mathbf{x}' \rangle$ puisque $\mathbf{w}$ est un vecteur orthogonal à $h(\mathbf{x})$. Afin de pouvoir comparer plusieurs hyperplans séparateurs de vecteurs directeurs $\mathbf{w}$ différents, on normalise la distance en la divisant par la norme du vecteur $\mathbf{w}$: $\langle \mathbf{w} , \mathbf{x} \rangle / \| \mathbf{w} \|$

Lorsqu'il existe une séparatrice linéaire entre les points d'apprentissage, il en existe en général une infinité. On peut alors chercher parmi ces séparatrices celle qui sépare « au mieux » les deux nuages de points exemples et contre-exemples. Cet hyperplan optimal est défini par :

$$\underset{\mathbf{w}, w_0}{\text{Argmax}} \ \min\{\| \mathbf{x} - \mathbf{x}_i \| \ : \ \mathbf{x} \in \mathbb{R}^d \ , \ (\mathbf{w}^\top \mathbf{x} + w_0) = 0 \ , \ i = 1, \ldots, m\}$$

c'est-à-dire l'hyperplan dont la distance minimale aux exemples d'apprentissage (voir figure 14.5) est la plus grande possible. Dans ce cas, la marge normalisée, appelée aussi *marge géométrique* vaut : $2/\| \mathbf{w} \|$.

Chercher un hyperplan séparateur de marge maximale présente plusieurs avantages. D'abord, même si les arguments théoriques sont délicats (voir chapitre 21), il est montré que *la capacité de l'espace d'hypothèses constitué par les hyperplans séparateurs diminue lorsque la marge augmente*. Ainsi, on peut espérer que l'hyperplan de marge maximale séparant les exemples des deux classes est l'hypothèse satisfaisant au mieux un risque empirique régularisé, favorisant donc la performance en généralisation. Ensuite, il existe un unique hyperplan séparateur de marge maximale et, le *problème* étant *convexe*, il peut être résolu par des méthodes de programmation quadratique, donc de manière efficace.

3.1 La résolution du problème d'optimisation

3.1.1 Formulation primale du problème

La recherche de l'hyperplan optimal revient donc à résoudre le problème d'optimisation suivant qui porte sur les paramètres $\mathbf{w}$ et w_0 :

$$\begin{cases} \text{Minimiser} & \frac{1}{2} \| \mathbf{w} \|^2 \\ \text{sous les contraintes} & u_i \, (\mathbf{w}^\top \mathbf{x}_i + w_0) \geq 1 \quad i = 1, \ldots, m \end{cases} \tag{14.17}$$

Copyright © 2010 Eyrolles.

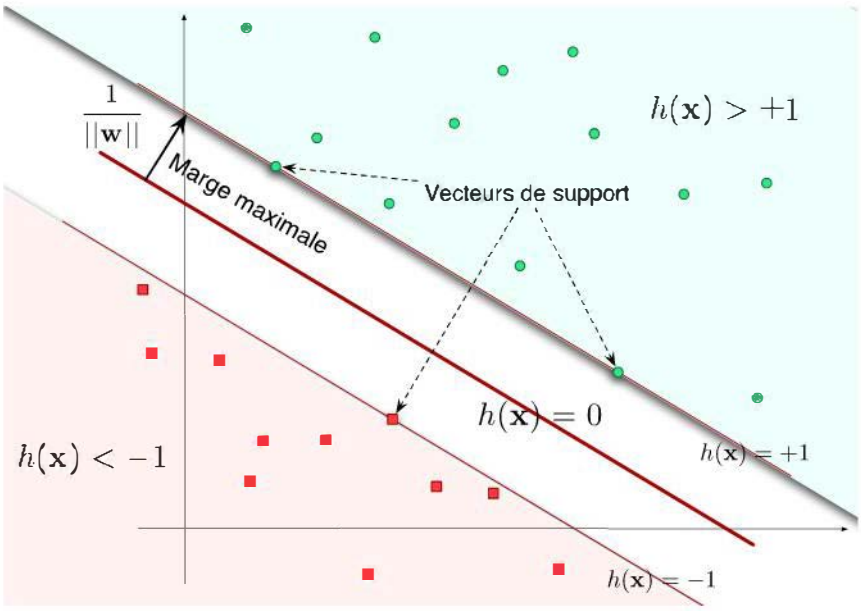


FIG. 14.5: *L'hyperplan optimal est perpendiculaire au segment de droite le plus court joignant un exemple d'apprentissage à l'hyperplan. Ce segment a pour longueur* $\frac{1}{\|\mathbf{w}\|}$ *lorsqu'on normalise convenablement les paramètres* $\mathbf{w}$ *et* w_0.

Cette écriture du problème, appelée *formulation primale*, implique le réglage de $d+1$ paramètres, d étant la dimension de l'espace des entrées $\mathcal{X}$. Cela est possible avec des méthodes de programmation quadratique[3] pour des valeurs de d assez petites, mais devient inenvisageable pour des valeurs de d dépassant quelques centaines. En effet, les calculs sont de complexité en $\mathcal{O}(d^3)$. Heureusement, il existe une transformation de ce problème dans une formulation duale que l'on peut résoudre en pratique.

3.1.2 Formulation duale du problème

D'après la théorie de l'optimisation, un problème d'optimisation possède une forme duale dans le cas où la fonction objectif et les contraintes sont strictement convexes. Dans ce cas, la résolution de l'expression duale du problème est équivalente à la solution du problème original. Ces critères de convexité sont réalisés dans le problème (14.17). Pour résoudre ces types de problèmes, on utilise une fonction que l'on appelle *lagrangien* qui incorpore des informations sur la fonction objectif et sur les contraintes et dont le caractère stationnaire peut être utilisé pour détecter des solutions. Plus précisément, le lagrangien est défini comme étant la somme de la fonction objectif et d'une combinaison linéaire des contraintes dont les coefficients $\alpha_i \geq 0$ sont appelés *multiplicateurs de Lagrange* ou encore *variables duales*.

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\| \mathbf{w} \|^2 - \sum_{i=1}^{m} \alpha_i \left(u_i.((\mathbf{w}\mathbf{x}_i^{\top}) + w_0) - 1\right) \qquad (14.18)$$

Un théorème de Kuhn-Tucker, couronnant des travaux commencés par Fermat (1601-1665), puis poursuivis par Lagrange (1736-1813), démontre que le problème primal et sa formulation

[3] Les problèmes d'optimisation pour lesquels la fonction et les contraintes sont linéaires ressortent des techniques de programmation linéaire, tandis que les problèmes d'optimisation dans lesquels la fonction est quadratique et les contraintes linéaires ressort des techniques de la programmation quadratique : voir le chapitre 3.

Copyright © 2010 Eyrolles.

duale ont la même solution. Celle-ci correspond à un *point-selle* du lagrangien (il faut le minimiser par rapport aux variables primaires $\mathbf{w}$ et w_0 et le maximiser par rapport aux variables duales α_i).

Au point-selle, la dérivée du lagrangien par rapport aux variables primaires doit s'annuler :

$$\frac{\partial L}{\partial \mathbf{w}}(\mathbf{w}, w_0, \alpha) = \mathbf{0}, \quad \frac{\partial L}{\partial w_0}(\mathbf{w}, w_0, \alpha) = 0 \tag{14.19}$$

d'où :

$$\sum_{i=1}^{m} \alpha_i u_i = 0 \tag{14.20}$$

et :

$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i u_i \mathbf{x}_i \tag{14.21}$$

Nous vérifions ici le théorème de représentation selon lequel le vecteur paramètres $\mathbf{w}^*$ de la fonction optimisant le critère régularisé est combinaison linéaire impliquant les exemples d'apprentissage $\mathbf{x}_1, \ldots, \mathbf{x}_m$.

Plus précisément, il est montré (*conditions de Karush-Kuhn-Tucker*) que seuls les points qui sont sur les hyperplans frontière $(\mathbf{w}^* \mathbf{x}_i^\top) + w_0^* = \pm 1$ jouent une rôle. Ces points pour lesquels les multiplicateurs de Lagrange sont non nuls sont appelés *vecteurs de support* par Vapnik. Nous utiliserons aussi le terme plus imagé d'*exemples critiques* puisque ce sont eux qui déterminent l'hyperplan optimal, tandis que les autres exemples ne jouent pas de rôle dans cette analyse[4].

L'hypothèse optimale présente ainsi deux **propriétés remarquables de *parcimonie***. D'une part, elle ne s'exprime qu'à l'aide d'un sous-ensemble (éventuellement très restreint) des exemples d'apprentissage, les exemples critiques. D'autre part, parce qu'elle correspond à la solution la plus « robuste », elle tolère d'être spécifiée avec moins de précision, donc avec moins de bits. On retrouve là un lien avec les principes inductifs par compression d'information (par exemple le « principe de minimisation de la taille de description » ou MDLP) (voir chapitre 21).

En substituant (14.20) et (14.21) dans (14.19), on élimine les variables primaires et l'on obtient la *forme duale* du problème d'optimisation, dans laquelle on cherche les multiplicateurs de Lagrange tels que :

$$\begin{cases} \text{Max}_\alpha \left[\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j u_i u_j \langle \mathbf{x}_i , \mathbf{x}_j \rangle \right] \\ \text{avec} \quad \begin{cases} \alpha_i \geq 0, \quad i = 1, \ldots, m \\ \sum_{i=1}^m \alpha_i u_i = 0 \end{cases} \end{cases} \tag{14.22}$$

L'**hyperplan solution** correspondant peut alors être écrit :

$$h^*(\mathbf{x}) = (\mathbf{w}^* \mathbf{x}) + w_0^* = \sum_{i=1}^{m} \alpha_i^* u_i . \langle \mathbf{x}_i , \mathbf{x} \rangle + w_0^* \tag{14.23}$$

où les α_i^* sont solution de (14.22) et w_0 est obtenue en utilisant n'importe quel exemple critique $(\mathbf{x}_c, u_c)$ dans l'équation :

$$w_0 = u_c - \mathbf{w}^\top \mathbf{x}_c = u_c - \sum_{i=1}^{m_c} u_i \alpha_i \mathbf{x}_i^\top \mathbf{x}_c \tag{14.24}$$

[4] On pourrait au contraire estimer que la distance des autres exemples à l'hyperplan séparateur doit participer aussi à l'évaluation de la qualité de l'hyperplan. C'est le cas dans l'analyse PaC-bayésienne des SVM.

Copyright © 2010 Eyrolles.

où m_c est le nombre d'exemples critiques, c'est-à-dire d'exemples situés sur la marge. Pour des raisons de stabilité numérique, on fait en général ce calcul pour tous les exemples critiques et on calcule w_0 en prenant la moyenne des résultats.

Deux choses sont remarquables. D'abord, *l'hyperplan solution ne requiert que le calcul des produits scalaires* $\langle \mathbf{x}_i \,.\, \mathbf{x} \rangle$ entre des vecteurs de l'espace d'entrée $\mathcal{X}$. C'est ce qui va permettre d'utiliser l'astuce des noyaux. Ensuite, *la solution ne dépend plus de la dimension d de l'espace d'entrée*, mais dépend de la taille m de l'échantillon de données et même, plus précisément, du nombre m_c d'exemples critiques qui est généralement bien inférieur à m. Les méthodes d'optimisation quadratique standards suffisent donc pour de nombreux problèmes pratiques.

3.2 Le cas d'un échantillon non linéairement séparable dans $\mathcal{X}$

L'analyse précédente suppose que les classes d'exemples sont linéairement séparables dans l'espace d'entrée $\mathcal{X}$. Ce cas est rare dans la pratique et c'est ce qui a motivé en particulier le développement des perceptrons multicouches (chapitre 10). En un sens, les couches cachées de ces réseaux connexionnistes calculent de nouvelles corrélations et, partant, de nouveaux descripteurs des formes d'entrée, ce qui permet à la couche de sortie de trouver une séparatrice simple, éventuellement linéaire, entre les entrées ainsi redécrites. La question est naturellement d'identifier les bons descripteurs. Dans le cas des réseaux connexionnistes, une manière de procéder est d'introduire des couches cachées complètement connectées entre elles et avec les couches d'entrée et de sortie et de faire confiance à l'algorithme de calcul des poids des connexions pour découvrir les corrélations pertinentes. Une autre manière de faire est de réfléchir à la nature du problème et d'introduire les bons blocs de neurones, comme par exemple dans le réseau à convolution `LeNet` développé pour reconnaître les chiffres manuscrits [MBB+92].

L'approche par fonctions noyau offre une autre possibilité. Nous l'avons vu en section 1.2, les fonctions noyau sont associées à un produit scalaire dans un espace de redescription, or un produit scalaire est un type de corrélation particulier entre les points de l'espace de redescription F, c'est-à-dire entre les projections des formes d'entrée $\Phi(\mathbf{x})$. L'utilisation de fonctions noyau va donc permettre d'exploiter automatiquement un certain type de corrélation entre les formes d'entrée, et, si possible, d'identifier les exemples critiques par rapport auxquels une frontière de décision pourra être trouvée. Le choix de la (ou des) fonction noyau va naturellement implicitement déterminer le type de corrélations considérées (voir section 5). Ainsi, par exemple, si l'on estime que ce sont des corrélations mettant en jeu des sous-régions de 10 pixels dans une image qui permettent de distinguer des classes d'objets, on choisira une fonction noyau permettant de tester toutes les corrélations de 10 pixels au moins.

Soit donc Φ une transformation non linéaire de l'espace d'entrée $\mathcal{X}$ en un *espace de redescription* $\Phi(\mathcal{X})$:

$$\mathbf{x} = (x_1, \ldots, x_d)^\top \;\mapsto\; \Phi(\mathbf{x})^\top = (\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x}), \ldots)^\top \tag{14.25}$$

Le problème d'optimisation se transcrit dans ce cas par :

$$\begin{cases} \mathrm{Max}_\alpha \left\{ \sum_{i=1}^m \alpha_i \;-\; \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j u_i u_j \langle \Phi(\mathbf{x}_i) \,,\, \Phi(\mathbf{x}_j) \rangle \right\} \\ \alpha_i \geq 0, \quad i = 1, \ldots, m \\ \sum_{i=1}^m \alpha_i u_i \;=\; 0 \end{cases} \tag{14.26}$$

L'équation de l'hyperplan séparateur dans le nouvel espace devient :

$$h(\mathbf{x}) \;=\; \sum_{i=1}^{m} \alpha_i^\star \, u_i \, \langle \Phi(\mathbf{x}) \,.\, \Phi(\mathbf{x}_i) \rangle \;+\; w_0^\star \tag{14.27}$$

Copyright © 2010 Eyrolles.

où les coefficients $\alpha_i^\star$ et $w_0^\star$ sont obtenus comme précédemment par résolution de (14.26).

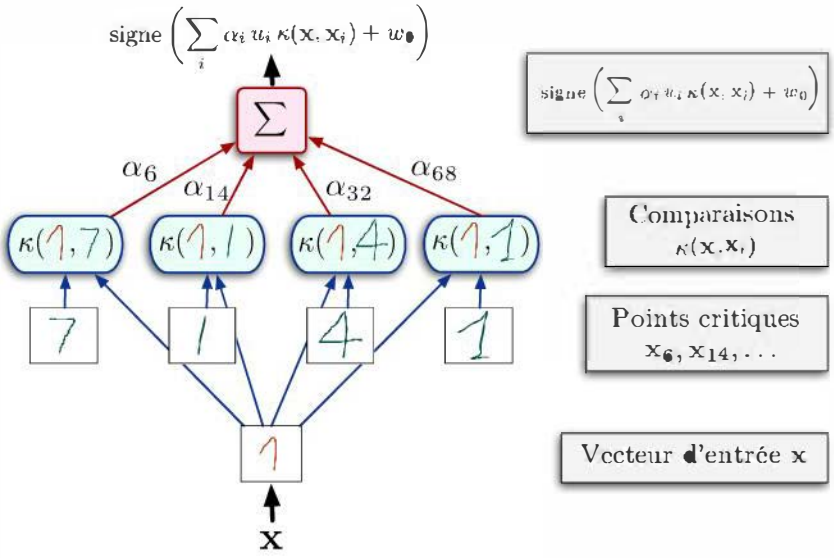


FIG. 14.6: *Cette figure résume le fonctionnement des séparateurs à vastes marges et montre le rôle des fonctions noyau. Lors de l'apprentissage, ici de chiffres manuscrits, un certain nombre d'exemples critiques sont retenus pour définir la fonction de décision (ici deux exemples positifs de '1' et deux exemples négatifs, un '4' et un '7'). Lorsqu'une nouvelle entrée est présentée au système, elle est comparée aux exemples critiques à l'aide des fonctions noyau qui réalisent un produit scalaire dans l'espace de redescription $\Phi(\mathcal{X})$. La sortie est calculée en faisant une somme pondérée (une combinaison linéaire) de ces comparaisons.*

Comme nous l'avons vu, les fonctions noyau correspondent à un (ou des) produits scalaires dans un espace de redescription F. On peut donc remplacer le problème précédent dans lequel il fallait déterminer la bonne transformation non linéaire Φ, par celui du choix d'une bonne fonction noyau $\kappa(\cdot,\cdot)$. Le problème d'optimisation associé devient alors :

$$\begin{cases} \text{Max}_\alpha \left\{ \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j u_i u_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right\} \\ \alpha_i \geq 0, \quad i = 1, \ldots, m \\ \sum_{i=1}^m \alpha_i u_i = 0 \end{cases} \tag{14.28}$$

dont la solution est l'hyperplan séparateur donnant la fonction de décision :

$$h(\mathbf{x}) = \text{signe}\left\{ \sum_{i=1}^m \alpha_i^\star u_i \kappa(\mathbf{x}, \mathbf{x}_i) + w_0^\star \right\} \tag{14.29}$$

où les coefficients $\alpha_i^\star$ et $w_0^\star$ sont obtenus comme précédemment par résolution de (14.28). Le processus total est illustré par la figure 14.6.

3.3 Échantillon difficilement séparable et marge douce (ou poreuse)

Nous avons supposé jusqu'ici qu'il existait un hyperplan (éventuellement dans l'espace de redescription) permettant de séparer les exemples des deux classes. Or, d'une part, il n'est pas

Copyright © 2010 Eyrolles.

nécessairement souhaitable de rechercher absolument un tel hyperplan, cela peut en effet conduire à une suradaptation aux données, d'autre part, il se peut que du bruit ou des erreurs dans les données ne permette tout simplement pas de trouver un tel hyperplan. Pour ces raisons, une version moins contrainte du problème de la recherche d'une séparatrice à vastes marges est le plus souvent considérée.

L'idée est de pénaliser les séparatrices admettant des exemples qui ne sont pas du bon côté des marges, sans cependant interdire une telle possibilité. On définit pour cela une fonction de coût particulière introduisant une pénalité pour tous les exemples mal classés et qui sont à une distance ξ de la marge qu'ils devraient respecter. On considère généralement des fonctions de coût qui soient compatibles avec la fonction de perte traditionnelle $\{0,1\}$-perte qui compte un coût de 1 pour chaque exemple mal classé. En particulier, on cherche des fonctions de coût qui conduisent à un critère inductif compatible avec le critère classique de minimisation du nombre d'exemples mal classés et donc à des solutions optimales égales ou proches. On parle de *fonctions de coût de substitution* (*surrogate loss functions*) Toute une gamme de fonctions sont possibles. Les deux plus étudiées sont la « fonction coude » (*hinge loss*) et la *fonction quadratique assymétrique* (voir figure 14.7). On parle alors de « marges douces » (*soft margin*)[5].

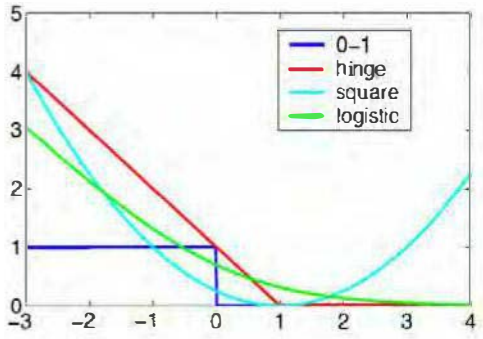


FIG. 14.7: *Approximations de la fonction de perte* $\{0,1\}$ *(ligne faisant une marche) par les fonction de perte « coude » (ligne), « quadratique » (courbe symétrique en 1), et « logistique » (courbe en descente douce). L'axe des abscisses correspond à la quantité* $y\,h(\mathbf{x})$ *qui est négative si l'exemple* $\mathbf{x}$ *est mal classé par* h.

On a alors :

$$\ell_{0-1}(h(\mathbf{x}),y) = \mathbf{I}_{-y\,h(\mathbf{x})>0} \leq \max\{1 - y\,h(\mathbf{x}), 0\} = \ell_{\text{lin}}(h(\mathbf{x}),y) \tag{14.30}$$

$$\ell_{0-1}(h(\mathbf{x}),y) = \mathbf{I}_{-y\,h(\mathbf{x})>0} \leq \max\{1 - y\,h(\mathbf{x}), 0\}^2 = \ell_{\text{quad}}(h(\mathbf{x}),y) \tag{14.31}$$

Approximation linéaire (fonction coude)

On cherche maintenant à minimiser un risque régularisé avec un taux de compromis λ :

$$R_{\text{reg}}(h) = \frac{1}{m}\sum_{i=1}^{m} \ell_{\text{lin}}(h(\mathbf{x}),u_i) + \lambda \parallel h \parallel^2 \tag{14.32}$$

[5] En raison du manque d'espace, nous ne discutons pas plus en détail les *fonctions de coût de substitution* ici. Il faut cependant être conscient que pour être valables, ces fonctions doivent vérifier un certain nombre de conditions. Leur choix doit se faire en fonction de la tâche : classification, régression, estimation de densité. De même, il est parfois utile d'avoir une fonction de classification *auto-calibrée*, c'est-à-dire pouvant être interprétée en terme de probabilité d'appartenance à la classe calculée. Ici encore, le choix de la fonction de coût de substitution est important (voir par exemple [SC08] pour une étude détaillée et théorique).

Copyright © 2010 Eyrolles.

soit encore :

$$\begin{cases} \text{Maximiser} & \sum_{i=1}^{m} \xi_i + \lambda m \parallel \mathbf{w} \parallel^2 \\ \text{avec} & \begin{cases} u_i \langle \mathbf{x}_i, \mathbf{w} \rangle \geq 1 - \xi_i & i = 1, \ldots, m \\ \xi \geq 0 \end{cases} \end{cases} \tag{14.33}$$

Le passage par la formulation duale conduit à :

$$\begin{cases} \text{Max}_{\alpha} \left[\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j u_i u_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right] \\ \forall i, \quad 0 \leq \alpha_i \leq C = \frac{1}{2\lambda m} \\ \sum_{i=1}^{m} \alpha_i u_i = 0 \end{cases} \tag{14.34}$$

Le coefficient C règle le compromis entre la marge possible entre les exemples et le nombre d'erreurs admissibles. Il doit être choisi par l'utilisateur. Une valeur de C grande correspond à une valeur de λ petite, c'est-à-dire à une grande pénalisation associée à chaque exemple mal classé. Il est à noter que puisque $C = \frac{1}{2\lambda m}$, l'intervalle des valeurs possibles pour les α_i décroît lorsque le nombre d'exemples m augmente.

Pour le cas d'une ***fonction de coût quadratique***, moins utilisée, nous reportons le lecteur par exemple à [Her02], pp.55-56.

3.4 Utilisation de fonctions noyau : illustrations

EXEMPLE **avec $\mathcal{X} = \mathbb{R}$**

Afin d'illustrer le fonctionnement des méthodes à noyaux, nous allons considérer un des problèmes de discrimination les plus simples imaginables. On suppose que l'espace des entrées est réduit à $\mathbb{R}$, c'est-à-dire à une dimension. On suppose que l'on a cinq points d'apprentissage associés à deux classes '+1' et '−1' :

$$S = \{(\mathbf{x}_1 = 1, u_1 = 1), (\mathbf{x}_2 = 1, u_2 = 1), (\mathbf{x}_3 = 4, u_3 = -1), (\mathbf{x}_4 = 5, u_4 = -1), (\mathbf{x}_5 = 6, u_5 = 1)\}$$

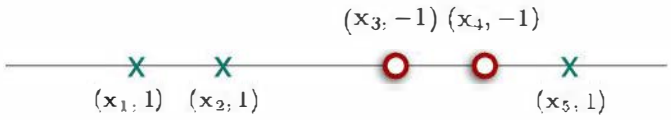


FIG. 14.8: *Un problème de discrimination à une dimension.*

Il n'est pas possible de trouver une séparatrice linéaire (un seul seuil sur l'axe $\mathbb{R}$) permettant de distinguer les exemples '1' des exemples '−1'.

Supposons que l'on choisisse d'utiliser la fonction noyau : $\kappa(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^2$. Cela signifie que l'on est disposé à considérer les corrélations au deuxième ordre entre les positions des exemples.

En choisissant $C = 100$, le problème d'optimisation associé est :

$$\begin{cases} \max\limits_{\alpha} \sum\limits_{i=1}^{5} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{5} \alpha_i \alpha_j u_i u_j (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2 \\ \sum\limits_{i,j=1}^{5} \alpha_i u_i = 0 \\ 0 \leq \alpha_i \leq 100 \quad (\forall i) \end{cases}$$

Copyright © 2010 Eyrolles.

Un programme de résolution de problème quadratique de ce type retourne la solution :

$$\alpha_1 = 0 \; ; \; \alpha_2 = 2.5 \; ; \; \alpha_3 = 0 \; ; \; \alpha_4 = 7.333 \; ; \; \alpha_5 = 4.833$$

La frontière de décision correspondante, de la forme $D(\mathrm{x}) = \sum_{i=1}^{m} \alpha_i^* u_i \kappa(\mathrm{x}, \mathrm{x}_i) + w_0^*$, est ainsi :

$$\begin{aligned} D(\mathrm{x}) &= 2.5\,(1)\,(2x+1)^2 + 7.333\,(1)\,(5x+1)^2 + 4.833\,(1)\,(6x+1)^2 + w_0 \\ &= 0.6667\,x^2 - 5.333\,x + w_0 \end{aligned}$$

où w_0 est obtenue en résolvant $h(2) = 1$, ou $h(5) = -1$ ou $h(6) = 1$, puisque x_2, x_4 et x_5 sont sur la droite $u_i(\mathrm{w}^\top \Phi(\mathrm{x}) + w_0 = 1$ pour $i \in \{2,4,5\}$. Ce qui donne $w_0 = 9$, d'où :

$$h(\mathrm{x}) = \mathrm{signe}\{0.6667\,x^2 - 5.333\,x + 9\}$$

C'est une frontière quadratique qui se traduit par deux seuils dans l'espace d'origine $\mathcal{X} = \mathbb{R}$. Les exemples x_2,x_4,x_5 sont les exemples critiques.

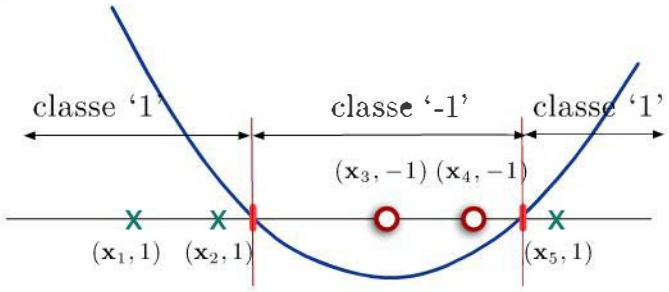
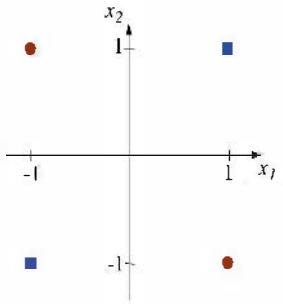


FIG. 14.9: *Un problème de discrimination à une dimension. La séparatrice est issue d'une fonction à deux dimensions qui se traduit par deux seuils sur* $\mathcal{X} = \mathbb{R}$.

EXEMPLE **avec $\mathcal{X} = \mathbb{R}^2$: la fonction XOR**

Soient des données décrites en deux dimensions $\mathcal{X} = \mathbb{R}^2$, et soient les exemples d'apprentissage décrits dans la table suivante :

Indice i	x_i	u_i
1	$(1, 1)$	1
2	$(1, -1)$	-1
3	$(-1, -1)$	1
4	$(-1, 1)$	-1



Supposons que l'on choisisse la fonction noyau polynomiale de degré 2 :

$$\begin{aligned} \kappa(\mathrm{x},\mathrm{x}') &= (1 + (\mathrm{x} \cdot \mathrm{x}'))^2 \\ &= (1 + (x_1 x_1' + x_2 x_2'))^2 \\ &= 1 + 2(x_1 x_1' + x_2 x_2') + (x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2\,x_1 x_1' + 2\,x_2 x_2' + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2\,x_1 x_1' x_2 x_2' \end{aligned}$$

Copyright © 2010 Eyrolles.

qui correspond à la projection Φ :

$$(x_1, x_2)^\top \overset{\Phi}{\mapsto} (1, \sqrt{2}\,x_1,\ \sqrt{2}\,x_2,\ x_1^2,\ x_2^2,\ \sqrt{2}\,x_1 x_2)^\top$$

On doit alors résoudre le problème :

$$\begin{cases} \max\limits_{\alpha} \sum\limits_{i=1}^{4} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{4} \alpha_i \alpha_j u_i u_j \, (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2 \\ \sum\limits_{i,j=1}^{4} \alpha_i u_i \ = \ 0 \\ 0 \leq \alpha_i \leq C \quad (\forall i) \end{cases}$$

où l'on doit maximiser :

$$\begin{aligned} &\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 \\ &\quad -\frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + \\ &\qquad 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2) \end{aligned}$$

Soit, en dérivant par rapport à chaque multiplicateur de Lagrange α_i :

$$\begin{cases} 9\alpha_1 - \alpha 2 - \alpha_3 + \alpha_4 \ = \ 1 \\ \alpha_1 - 9\alpha 2 - \alpha_3 + \alpha_4 \ = \ 1 \\ \alpha_1 - \alpha 2 - 9\alpha_3 + \alpha_4 \ = \ 1 \\ \alpha_1 - \alpha 2 - \alpha_3 + 9\alpha_4 \ = \ 1 \end{cases}$$

On trouve alors la valeur optimale des multiplicateurs de Lagrange :

$$\alpha_1^\star \ = \ \alpha_2^\star \ = \ \alpha_3^\star \ = \ \alpha_4^\star \ = \ \frac{1}{8}$$

Les quatre exemples sont donc tous des exemples critiques, et la fonction de décision est alors :

$$h(\mathrm{x}) \ = \ \mathrm{signe}\left\{ \sum_{i=1}^{m_s} \alpha_i^\star \, u_i \, \kappa(\mathbf{x}, \mathbf{x}_i) + w_0 \right\} \ = \ \mathrm{signe}\left\{ \frac{1}{8} \sum_{i=1}^{4} u_i \, [(\mathrm{x}^\top \cdot \mathbf{x}_i) + 1]^2 \right\}$$

soit :

$$h(\mathrm{x}) \ = \ \mathrm{signe}\left\{ \frac{1}{8} \sum_{i=1}^{4} u_i \, [1 + (\mathbf{x} \cdot \mathbf{x}_i)]^2 \right\}$$

Après calculs, on arrive à l'*équation de la séparatrice dans l'espace d'entrée* $\mathcal{X}$:

$$h(\mathbf{x}) \ = \ \mathrm{signe}\{-x_1 x_2\}$$

qui est une fonction non linéaire (voir figure 14.10, à gauche).
Quelle est l'équation de la séparatrice dans l'espace de redescription F ? On a :

$$D(\mathrm{x}) = \sum_{i=1}^{4} \alpha_i^\star u_i \kappa(\mathbf{x}_i, \mathbf{x}) + w_0 = \sum_{i=1}^{4} \alpha_i^\star u_i \langle \phi(\mathbf{x}_i), \phi(\mathrm{x}) \rangle + w_0 = \mathbf{w}^{\star\top} \mathbf{\Phi}(\mathrm{x}) + w_0$$

avec : $\mathbf{w}^\star = \sum_{i=1}^{4} \alpha_i^\star u_i \Phi(\mathbf{x}_i)$, et $w_0 = 0$ dans ce problème symétrique par rapport à l'origine.
On a alors : $\mathbf{w}^\star = \frac{1}{8}[-\Phi(\mathbf{x}_1) + \Phi(\mathbf{x}_2) + \Phi(\mathbf{x}_3) - \Phi(\mathbf{x}_4)]$.

Copyright © 2010 Eyrolles.

En calculant les projections $\Phi(\mathbf{x}_i)$ des quatre points, on trouve :

$$\mathbf{w}^* = \frac{1}{8}\left\{ -\begin{pmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \\ \sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \\ -\sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ -\sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \\ \sqrt{2} \end{pmatrix} - \begin{pmatrix} 1 \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \\ -\sqrt{2} \end{pmatrix} \right\} = \begin{pmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

D'où l'équation de la frontière de décision dans l'espace de redescription F :

$$\begin{aligned} D_F(\mathbf{x}) &= \mathbf{w}^{*\top}\, \Phi(\mathbf{x}) \\ &= (0, 0, -1/\sqrt{2}, 0, 0, 0)\ (1, \sqrt{2}x_1,\ \sqrt{2}x_2,\ x_1^2,\ x_2^2,\ \sqrt{2}x_1x_2)^\top \\ &= -\sqrt{2}x_1x_2 = 0 \end{aligned}$$

qui est donc une équation linéaire dans l'espace F (sixième coordonnée constante) (voir figure 14.10 à droite).

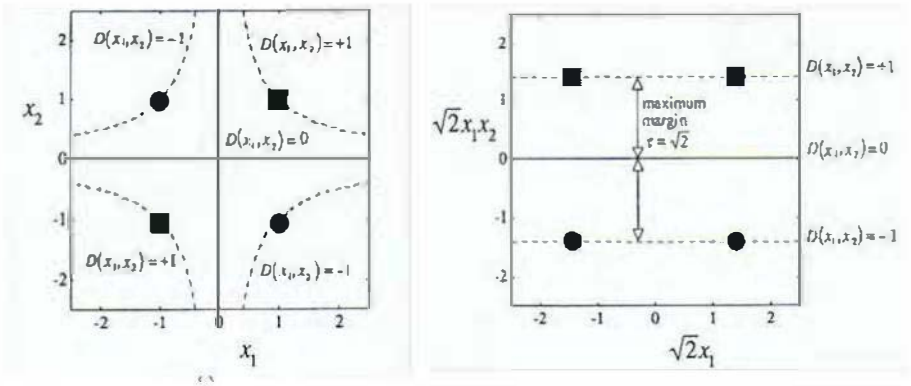


FIG. 14.10: *Les frontières de décision pour le problème du XOR. À gauche dans l'espace d'entrée $\mathcal{X}$, à droite dans l'espace de redescription F en projetant sur deux coordonnées parmi les 6.*

3.5 Séparateurs à Vastes Marges et généralisation

Selon la théorie statistique de l'apprentissage, la dimension de Vapnik-Chervonenkis de la classe des séparateurs linéaires de marge γ est égale à R^2/γ^2, où R est le rayon de la plus petite sphère englobant tous les points d'apprentissage.

Il a été montré plus récemment [SSTSW99, WSS01] qu'il est possible de borner l'erreur en généralisation en utilisant le spectre d'un opérateur intégral associé et le spectre[6] de la matrice de Gram $\kappa(\mathbf{x}_i, \mathbf{x}_j)$. En effet, ces nouvelles analyses tirent avantage du fait que, pour la plupart des applications, la distribution des données n'est pas isotrope dans l'espace, et, de ce fait, les bornes utilisant la mesure de sphère sont moins précises.

[6] Le spectre d'une matrice est l'ensemble de ses valeurs propres.

Copyright © 2010 Eyrolles.

4. Autres types d'induction avec fonctions noyau

4.1 La régression

Nous avons vu dans la section 1.2 qu'il est possible d'utiliser l'approche par fonction noyau pour résoudre des tâches de régression non linéaire, c'est-à-dire de prédiction d'une valeur réelle à partir d'une entrée prise dans un espace, éventuellement multidimensionnel, typiquement $\mathbb{R}^d$.

Le problème inductif revient à chercher une fonction $h(\mathbf{x}) = y \in \mathbb{R}$ telle que, pour tous les points d'apprentissage $\{(\mathbf{x}_i, u_i)\}_{1 \leq i \leq m}$, $h(\mathbf{x}_i)$ soit le plus « proche » possible de u_i, tout en se plaçant dans un espace d'hypothèses contrôlé pour éviter le risque de sur-apprentissage.

Une analyse de la convergence du risque empirique sur le risque réel, mesuré en terme d'écart quadratique, ainsi que la recherche d'une solution parcimonieuse en terme d'exemples support, a conduit Vapnik à proposer d'utiliser la fonction de perte suivante :

$$|y - h(\mathbf{x})|_\varepsilon = \max\{0, |y - h(\mathbf{x})| - \varepsilon\} \tag{14.35}$$

Alors, afin d'estimer la régression linéaire :

$$h(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + w_0 \tag{14.36}$$

avec une précision de ε, on minimise :

$$\frac{1}{2}\| \mathbf{w} \|^2 + C \sum_{i=1}^{m} |u_i - f(\mathbf{x}_i)|_\varepsilon \tag{14.37}$$

On peut réexprimer ce problème sous forme d'un problème d'optimisation sous contraintes (voir [Vap95]) :

$$\begin{cases} \text{Minimiser} & \tau(\mathbf{w}, \xi, \xi^\star) = \dfrac{1}{2}\| \mathbf{w} \|^2 \\ \text{sous les contraintes} & \begin{cases} ((\mathbf{w} \cdot \mathbf{x}_i) + w_0) - u_i \leq \varepsilon + \xi_i \\ u_i - ((\mathbf{w} \cdot \mathbf{x}_i) + w_0) \leq \varepsilon + \xi_i \\ \xi_i, \xi_i^\star \geq 0 \end{cases} \end{cases} \tag{14.38}$$

pour tous les $i = 1, \ldots, m$. On peut noter à nouveau que toute erreur plus petite que ε ne requiert pas une valeur non nulle de ξ ou de ξ_i et ne doit donc pas être prise en compte par la fonction objectif (14.38). La figure 14.11 illustre le rôle des contraintes.

Comme dans le cas de la classification, on peut généraliser à la régression non linéaire en passant par un espace de redescription des entrées grâce à l'utilisation de fonctions noyau. L'introduction de multiplicateurs de Lagrange conduit au problème d'optimisation suivant, dans lequel les constantes $C > 0$ et $\xi \geq 0$ sont choisies *a priori* :

$$\begin{cases} \text{Maximiser} & W(\alpha, \alpha) = -\varepsilon \sum_{i=1}^{m} (\alpha_i^\star + \alpha_i) + \sum_{i=1}^{m} (\alpha_i^\star - \alpha_i) u_i \\ & \qquad - \frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^\star - \alpha_i)(\alpha_i^\star - \alpha_j) k(\mathbf{x}_i, \mathbf{x}_i) \\ \text{sous les contraintes} & \begin{cases} 0 \leq \alpha_i, \alpha_i^\star \leq C \quad i = 1, \ldots, m \\ \sum_{i=1}^{m} (\alpha_i - \alpha_i^\star) = 0 \end{cases} \end{cases} \tag{14.39}$$

Copyright © 2010 Eyrolles.

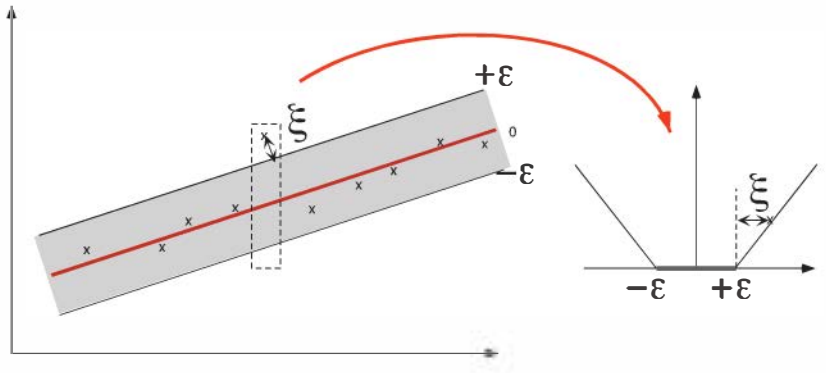


FIG. 14.11: *Dans la régression par SVM, au lieu d'imposer une marge entre les points des deux classes, on impose une sorte de « chaussette » autour des points grâce au paramètre ε. Le compromis entre la complexité du modèle (la fonction de régression) et la fidélité aux points d'apprentissage est règlée par la variable ressort ξ.*

L'estimation de la régression prend alors la forme :

$$h(\mathbf{x}) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i)\, k(\mathbf{x}_i, \mathbf{x}) + w_0 \tag{14.40}$$

où b est calculé en utilisant le fait que la contrainte $((\mathbf{w} \cdot \mathbf{x}_i) + w_0) - u_i \leq \varepsilon + \xi_i$ devient une égalité avec $\xi_i = 0$ si $0 < \alpha_i < C$, et la contrainte $u_i - ((\mathbf{w} \cdot \mathbf{x}_i) + w_0) \leq \varepsilon + \xi_i$ devient une égalité avec $\xi_i^* = 0$ si $0 < \alpha_i^* < C$.

4.2 Induction avec une seule classe ou non supervisée

À partir du moment où l'on considère l'astuce des noyaux (utilisation des fonctions noyau pour changer d'espace de description dans lequel des méthodes linéaires s'appuyant sur des produits scalaires peuvent être employées), il est tentant de chercher à l'utiliser dans tous les domaines de l'apprentissage. C'est ce qui a été également fait en apprentissage non supervisé. Contrairement aux problèmes d'apprentissage supervisé dans lesquels la tâche fondamentale est de prédire l'étiquette d'une nouvelle forme d'entrée, et pour lesquels le critère de performance est assez aisé à définir, ces problèmes sont moins faciles à préciser. Il s'agit essentiellement de trouver une caractérisation des données permettant de déterminer des régularités intéressantes sur leur structure ou sur la distribution de probabilité sous-jacente (voir chapitre 18).

Les méthodes d'apprentissage non supervisé incluent :

- La recherche d'un « repère » ou référentiel permettant de décrire au mieux les données en comprimant leur description. Les techniques d'*Analyse en Composantes Principales* par noyau (*Kernel PCA*) répondent en partie à ce problème (voir [SS02] pour une description générale).
- La *recherche de sous-variétés* dans les données préservant leur topologie (voir [LV07]).
- La recherche de nuages de points (*clustering*) ou de catégories dans les données.
- L'estimation de densité sous-jacente aux données.

Dans cette section, nous étudions un problème plus simple, binaire : celui de trouver une région de l'espace d'entrée $\mathcal{X}$ contenant une grande partie des données (proportion à préciser). Comme dans tout problème d'induction, le principe est d'identifier cette région sur la base d'un

Copyright © 2010 Eyrolles.

échantillon d'apprentissage jugé représentatif. Une tâche souvent associée à ce problème est celui de l'identification de formes exceptionnelles ou anormales (*outliers*).

Suivant en cela le principe philosophique de Vapnik selon lequel il ne faut pas résoudre un problème plus difficile ou plus général que celui considéré, il n'est pas nécessaire d'induire la densité de probabilité génératrice des données. Il suffit de trouver une frontière de décision entre la région considérée, couvrant une proportion minimale des données, et le reste de l'espace $\mathcal{X}$. C'est l'objet des SVM mono-classe (*one-classe SVM*).

À l'instar des autres problèmes inductifs, il est nécessaire de résoudre un compromis entre disposer d'une classe de fonctions de décision suffisamment riches pour bien approcher la région recherchée et éviter cependant de risquer d'être victime de sur-adaptation. La méthode des SVM permet de résoudre ce problème de régularisation.

En fait, l'origine de la méthode proposée réside dans une question théorique : peut-on déterminer avec précision le centre de masse de données générées à partir d'une distribution génératrice inconnue sur la base d'un échantillon de données tirées aléatoirement et indépendamment ? Soit $\mathbb{E}_{\mathsf{P}_\mathcal{X}}[\Phi(\mathbf{x})] = \int_\mathcal{X} \Phi(\mathbf{x}) d\mathsf{P}_\mathcal{X}(\mathbf{x})$ le « vrai » centre de masse. Une analyse statistique permet de borner la valeur de $\|\frac{1}{m}\sum_{i=1}^m \Phi(\mathbf{x}_i) - \mathbb{E}_{\mathsf{P}_\mathcal{X}}[\Phi(\mathbf{x})]\|$ et de montrer que cette borne ne dépend pas de la dimension de l'espace, mais est de l'ordre de $\sqrt{1/m}$ comme attendu, ainsi que du rayon de l'hypersphère englobant les projections de données dans $\Phi(\mathcal{X})$.

Cette analyse suggère un algorithme de détection de nouveauté ou d'anomalie. Considérons en effet la boule ou hypersphère minimale englobant les m points d'apprentissage dans l'espace $\Phi(\mathcal{X})$. La probabilité qu'un nouveau point ne soit pas englobé dans une boule minimale *centrée en* $\frac{1}{m}\sum_{i=1}^m \Phi(\mathbf{x}_i)$ est $\leq \frac{1}{m+1}$ (par un raisonnement simple de symétrie entre les données et l'emploi d'une inégalité triangulaire ; voir par exemple [STC04], pp.116-117). On pourra ainsi considérer que tout point tombant en dehors de cette boule, représentant les exemples « normaux », est anormal.

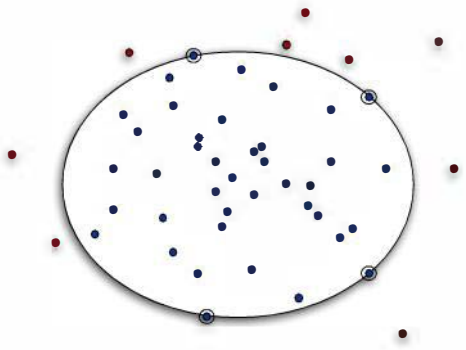


FIG. 14.12: *Exemple d'une « hypersphère » définie sur un ensemble de points d'apprentissage. Les points entourés sont les exemples support. Les points bleus, extérieurs à l'hypersphère, sont jugés « nouveaux ».*

Une analyse plus fine[7] prenant en compte le compromis entre le rayon de l'hypersphère et la sensibilité à des points normaux mais peu représentatifs de la distribution de probabilité considérée conduit au problème suivant. Étant donné un échantillon d'apprentissage $S = \langle \mathbf{x}_1, \ldots, \mathbf{x}_m \rangle$,

[7] Voir par exemple [STC04], pp.196-205.

Copyright © 2010 Eyrolles.

trouver le vecteur $\alpha^\star$ optimisant l'expression :

$$\begin{cases} \text{Maximiser} & \sum_{i=1}^m \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^m \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ \text{sous contraintes de :} & \begin{cases} \sum_{i=1}^m \alpha_i = 1 \\ 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m. \end{cases} \end{cases} \tag{14.41}$$

En choisissant alors i tel que $0 < \alpha_i^\star < C$, on calcule :

$$\begin{aligned} r^\star &= \sqrt{\kappa(\mathbf{x}_i, \mathbf{x}_i) - 2\sum_{j=1}^m \alpha_j^\star \kappa(\mathbf{x}_j, \mathbf{x}_i) + \sum_{i,j=1}^m \alpha_i^\star \alpha_j^\star \kappa(\mathbf{x}_i, \mathbf{x}_j)} \\ D &= \sum_{i,j=1}^m \alpha_i^\star \alpha_j^\star \kappa(\mathbf{x}_i, \mathbf{x}_j) - (r^\star)^2 - \gamma \end{aligned}$$

Et la fonction de décision :

$$\boxed{h(\mathbf{x}) = H\left[\kappa(\mathbf{x},\mathbf{x}) - 2\sum_{i=1}^m \alpha_i^\star \kappa(\mathbf{x}_i, \mathbf{x}) + D\right]} \tag{14.42}$$

où H est la fonction de Heaviside :

$$\forall x \in \mathbb{R}, \qquad H(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad x \geq 0. \end{cases} \tag{14.43}$$

La fonction h retourne la valeur 1 pour une entrée tombant en dehors de l'hypersphère, et devant donc être interprétée comme une forme nouvelle.

Encore une fois, la résolution du problème n'implique généralement qu'une partie réduite des exemples d'apprentissage qui se trouvent être sur la frontière de l'hypersphère calculée.

Dans le cas où les données sont normalisées, on peut montrer que le problème peut également être résolu en cherchant un hyperplan dans $\Phi(\mathcal{X})$ séparant au maximum les données de l'origine.

4.3 Induction de sorties structurées

Il est des tâches correspondant à l'induction de fonctions liant des entrées à des sorties ayant une structure. Par exemple, un système d'analyse grammaticale prend en entrée du texte et produit en sortie un arbre syntaxique correspondant à la structure syntaxique de la phrase (voir figure 14.13). D'autres tâches consistent à associer la structure tertiaire d'une protéine à la séquence de ses acides aminés. On pourrait aussi imaginer vouloir associer une structure spatiale d'éléments saillants (e.g. des personnages) à partir d'une image video.

Une première approche du problème pourrait être de décomposer la structure visée en sous-composants et d'apprendre des fonctions de reconnaissance pour chacun de ces composants puis de les assembler d'une manière ou d'une autre. Mais cela nécessiterait la modélisation d'un grand nombre d'interdépendances entre ces sous-composants. Surtout, cela ne permettrait pas de représenter et de tirer parti de relations de similarité dans l'espace de sortie, et donc de généraliser. Une autre approche pourrait être d'apprendre directement à prédire chaque sortie possible (e.g. chaque arbre syntaxique). Mais le nombre de sorties possibles est en général exponentiel avec le nombre de sous-composants en jeu, et il est impossible d'imaginer d'avoir suffisamment de données pour apprendre à prédire chacune de ces sorties.

Copyright © 2010 Eyrolles.

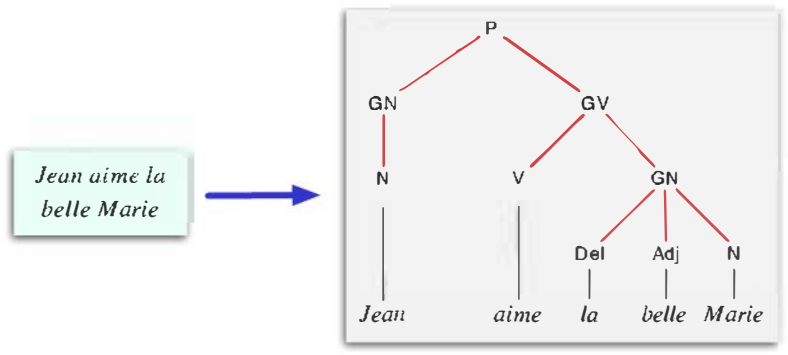


FIG. 14.13: *Exemple d'une transformation de texte en arbre syntaxique.*

L'idée centrale des méthodes d'apprentissage de fonctions à sorties structurées est de passer par un espace de redescription qui prend en compte à la fois l'espace d'entrée $\mathcal{X}$, mais aussi l'espace de sortie $\mathcal{Y}$. Ainsi, au lieu de passer (virtuellement) par un espace de redescription $\Phi(\mathcal{X})$ comme dans l'approche standard des méthodes à noyaux, on passe désormais (virtuellement) par un espace de redescription $\Psi(\mathcal{X}, \mathcal{Y})$. Cet espace de redescription qui mêle des descripteurs appartenant à l'espace d'entrée et à l'espace de sortie permet de prendre en compte des propriétés combinées de ces deux espaces. Au lieu de la forme classique : $h(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i) + w_0$, les fonctions hypothèse auront la forme[8] :

$$h(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \kappa\big((\mathbf{x}, \mathbf{y}), (\mathbf{x}_i, \mathbf{y}_i)\big) + w_0 = \sum_{i=1}^{m} \alpha_i \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}_i, \mathbf{y}_i) \rangle + w_0 \qquad (14.44)$$

Mais de quel problème d'optimisation une fonction de ce type peut-elle être solution ? L'approche actuelle prend pour point de départ le problème multi-classe. Jusqu'à présent, nous n'avons traité avec les SVM que le problème de discrimination à deux classes. Supposons que nos exemples appartiennent à plus de deux classes. Comment formaliser ce problème ?

On va supposer qu'il existe un vecteur $\mathbf{w}_\mathbf{y}$ pour chaque classe $\mathbf{y}$. Chaque exemple $\mathbf{x}$ a donc un score par rapport à chaque classe $\mathbf{y}$ grâce à une fonction dite de compatibilité $g(\mathbf{x}, \mathbf{y}) = \mathbf{w}_\mathbf{y}\, \Phi(\mathbf{x})$. La règle de classification étant alors :

$$h(\mathbf{x}) = \underset{\mathbf{y} \in \mathcal{Y}}{\text{ArgMax}}\, g(\mathbf{x}, \mathbf{y})$$

Cette règle fournit la bonne réponse (au moins pour les exemples d'apprentissage) sous réserve que les vecteurs de poids calculés $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{x}_{|\mathcal{Y}|})$ aient été déterminés pour respecter $g(\mathbf{x}, \bar{\mathbf{y}}) < g(\mathbf{x}, \mathbf{y})$ pour toutes les sorties incorrectes $\bar{\mathbf{y}} \neq \mathbf{y}$.

Pour un échantillon d'apprentissage $\mathcal{S}_m = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y})\}$, cette contrainte conduit à la forme primale du problème suivant, où l'on requiert l'existence d'une marge minimale (=1) entre les différentes classes :

$$\begin{cases} & \text{Min}_\mathbf{w}\ \frac{1}{2}\,||\mathbf{w}||^2 \\ \text{tel que :} & g(\mathbf{x}_i, \mathbf{y}_i) - g(\mathbf{x}_i, \bar{\mathbf{y}}) \geq 1 \quad (\forall i \in \{1, \ldots, m\} \text{ et } \bar{\mathbf{y}} \neq \mathbf{y}_i) \end{cases}$$

Pour un problème à $k = |\mathcal{Y}|$ classes, ce problème d'optimisation a $m(k-1)$ inégalités au total, linéaires en $\mathbf{w}$ puisque l'on peut écrire : $g(\mathbf{x}_i, \mathbf{y}_i) - g(\mathbf{x}_i, \bar{\mathbf{y}}) = (\mathbf{w}_{\mathbf{y}_i} - \mathbf{w}_{\bar{\mathbf{y}}}) \cdot \Phi(\mathbf{x}_i)$. C'est donc un problème pouvant être résolu par programmation quadratique convexe.

[8] Pour rappeler que la sortie est un objet structuré, nous utilisons par extension la notation $\mathbf{y}$ usuellement utilisée pour les vecteurs.

Copyright © 2010 Eyrolles.

Cependant, comme nous l'avons dit, cette formulation ne permet pas une généralisation à l'intérieur de l'espace de sortie $\mathcal{Y}$. D'où une nouvelle formulation pour la fonction de compatibilité : $g(\mathbf{x}, \mathbf{y}) = \mathbf{w} \cdot \Psi(\mathbf{x}, \mathbf{y})$. Le problème devient alors :

$$\begin{cases} & \text{Min}_{\mathbf{w}} \ \frac{1}{2} ||\mathbf{w}||^2 \\ \text{tel que :} & \mathbf{w} \cdot \Psi(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w} \cdot \Psi(\mathbf{x}_i, \bar{\mathbf{y}}) \geq 1 \quad (\forall i \in \{1, \ldots, m\} \text{ et } \bar{\mathbf{y}} \neq \mathbf{y}_i) \end{cases} \tag{14.15}$$

En d'autres termes, on cherche un vecteur $\mathbf{w}$ d'une fonction de compatibilité g linéaire par rapport aux descripteurs d'un espace de redescription $\Psi(\mathcal{X}, \mathcal{Y})$, tel que pour chaque exemple d'apprentissage, cette fonction donne un score à la classe correcte supérieur d'au moins 1 au score donné pour toute autre classe, tout en limitant la complexité (c'est-à-dire en minimisant la norme $||\mathbf{w}||$).

Pour résoudre complètement ce problème, il faut résoudre certaines difficultés :

- Le nombre de contraintes linéaires est encore de $m(|\mathcal{Y}| - 1)$, et il faut donc des techniques particulières pour y faire face (e.g. algorithme de sections de plans (« *cutting-plane algorithm* »)).
- Il faut pouvoir tenir compte d'erreurs dans les données et admettre des marges douces, donc introduire des variables ressort.
- Il faut surtout définir un bon espace de redescription $\Psi(\mathcal{X}, \mathcal{Y})$, soit directement, soit indirectement via la définition d'une fonction noyau appropriée $\kappa\big((\mathbf{x}, \mathbf{y}), (\mathbf{x}_i, \mathbf{y}_i)\big)$. C'est là le plus gros problème. Par exemple, dans le problème de l'analyse syntaxique de textes, il a été suggéré de compter combien chaque règle de production de la grammaire sous-jacente avait été employés pour un couple $(\mathbf{x}, \mathbf{y})$. Il faut surtout, si l'on peut, tenir compte des décompositions raisonnables dans l'espace $\mathcal{X} \times \mathcal{Y}$ pour simplifier le problème.

L'apprentissage de fonctions produisant des sorties structurées est un problème très important en raison en particulier du grand nombre d'applications faisant intervenir de telles fonctions. Si le nombre de travaux se rapportant à ce problème s'accroît rapidement et a déjà apporté des solutions d'une ingéniosité impressionnante, il reste encore beaucoup à faire. Nous renvoyons le lecteur intéressé à un article de synthèse constituant une bonne porte d'entrée à cette famille d'approches [JHYY09]. (Voir aussi la section 4 du chapitre 20).

5. Ingénierie des fonctions noyau

5.1 Les fonctions noyau : signification et capacité d'approximation

5.1.1 Signification des fonctions noyau

La section 1 a montré que l'utilisation de fonctions noyau était liée à des raisons fondamentales diverses. En résumant, on peut dire que le choix d'une fonction noyau correspond implicitement au choix :

- d'une *mesure de similarité* entre éléments de l'espace d'entrée $\mathcal{X}$;
- d'une *projection* des éléments de $\mathcal{X}$ dans un espace vectoriel F doté d'un produit scalaire ;
- d'un *espace fonctionnel* doté d'une base de fonctions analysantes (e.g. base de Mercer ou base d'un espace de Hilbert à noyau reproduisant (RKHS)) dans laquelle est cherchée une fonction hypothèse ;

Copyright © 2010 Eyrolles.

- d'un *critère inductif régularisé* puisque le théorème de représentation montre que chaque noyau correspond à un terme de régularisation ;
- d'une *fonction de covariance* définissant comment les éléments de $\mathcal{X}$ sont corrélés ;
- d'une *mesure de probabilité sur un ensemble de fonctions* comme le montre l'interprétation des fonctions noyau comme fonction de covariance.

Le choix d'une fonction noyau est donc essentiel pour l'apprentissage et doit refléter au mieux toute connaissance *a priori* sur le domaine étudié.

5.1.2 Les fonctions noyau et la capacité du RKHS associé

Nous avons vu qu'une fonction noyau induisait l'espace de fonctions dans lequel était cherché une fonction hypothèse satisfaisant au mieux un critère inductif régularisé. Le chapitre 2 a par ailleurs amplement insisté sur l'importance du contrôle de la capacité de l'espace des hypothèses pour garantir, en probabilité, un lien entre le risque empirique et le risque réel. Il semble donc essentiel de contrôler le choix de la fonction noyau afin d'éviter le risque de sous-apprentissage (espace de fonctions trop pauvre) ou de surapprentissage (espace de fonctions trop riche). En fait, de nombreuses fonctions noyau sont associées à des espaces fonctionnels universels, c'est-à-dire permettant d'approcher n'importe quelle fonction sur les points d'apprentissage (en d'autres termes des espaces fonctionnels de dimension de Vapnik-Cervonenkis infinie).

Définition 14.3 (Noyau universel)

Une fonction noyau continue sur un espace compact métrique $\mathcal{X}$ est dit universel *si l'espace de Hilbert à noyau reproduisant (RKHS) $\mathcal{F}$ associé est dense dans l'espace $\mathcal{C}(\mathcal{X})$ des fonctions continues de $\mathcal{X}$ dans $\mathbb{R}$, c'est-à-dire si pour toute fonction $g \in \mathcal{C}(\mathcal{X})$ et pour tout $\varepsilon > 0$, il existe une fonction $f \in \mathcal{F}$ telle que :*

$$||f - g||_\infty \leq \varepsilon$$

où $||\cdot||_\infty$ indique la norme supremum $||x||_\infty := \sup_{s\in[0,1]} x(s)$.

Une fonction noyau universelle permet en particulier de séparer tout sous-ensemble de points de $\mathcal{X}$ en deux sous-ensembles quelconques disjoints, ce qui correspond à une dimension de Vapnik-Cervonenkis infinie.

Il peut être montré que les fonctions noyau strictement positives peuvent séparer ainsi tout sous-ensemble fini de points. De nombreuses fonctions noyau standard possèdent cette propriété :

- les *noyaux exponentiels* : $\kappa(\mathbf{x}, \mathbf{x}') = \exp(\langle \mathbf{x}, \mathbf{x}' \rangle)$
- les *noyaux gaussiens (RBF)* : $\kappa(\mathbf{x}, \mathbf{x}') = \exp -\frac{||\mathbf{x}-\mathbf{x}'||}{\gamma^2}$
- les *noyaux binomiaux* : $\kappa(\mathbf{x}, \mathbf{x}') = (1 - \langle \mathbf{x}, \mathbf{x}' \rangle)^{-\alpha}$

Si l'espace fonctionnel correspondant à l'espace des hypothèses est de capacité d'approximation universelle, l'induction n'est possible que si le critère inductif est proprement régularisé. C'est ce que réalise en particulier la technique des Séparateurs à Vastes Marges (SVM).

5.2 L'espace des fonctions noyau : règles simples de construction

Les fonctions noyau ont pour propriété d'être positives définies, c'est-à-dire d'être associées à des matrices de Gram : $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$ positives semi-définies. La construction de fonctions noyau à partir de fonctions noyau connues utilise des opérations qui préservent cette propriété.

Copyright © 2010 Eyrolles.

Théorème 14.5 (Règles de construction)

Soient κ_1 et κ_2 deux fonctions noyau sur $\mathcal{X} \times \mathcal{X}$, $\mathcal{X} \in \mathbb{R}^d$, $c \in \mathbb{R}^+$, $f(\cdot)$ une fonction réelle sur $\mathcal{X}$, poly$(\cdot)$ un polynome avec des coefficients positifs ou nuls, $\Phi(\cdot)$ une fonction de $\mathcal{X}$ sur $\mathbb{R}^D$, κ_3 une fonction noyau définie sur $\Phi(\mathcal{X}) \times \Phi(\mathcal{X})$, $\mathbf{A}$ une matrice positive semi-définie, $\mathbf{x}_a$ et $\mathbf{x}_b$ des variables avec $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$ et κ_a et κ_b des fonctions noyau dans leur espace respectif. Alors les fonctions suivantes sont des fonctions noyau :

$$
\begin{array}{lll}
\text{(i)} & \kappa(\mathbf{x}, \mathbf{x}') = & c\,\kappa_1(\mathbf{x}, \mathbf{x}') \\
\text{(ii)} & \kappa(\mathbf{x}, \mathbf{x}') = & f(\mathbf{x})\,\kappa_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \\
\text{(iii)} & \kappa(\mathbf{x}, \mathbf{x}') = & \text{poly}\big(\kappa_1(\mathbf{x}, \mathbf{x}')\big) \\
\text{(iv)} & \kappa(\mathbf{x}, \mathbf{x}') = & \exp\big(\kappa_1(\mathbf{x}, \mathbf{x}')\big) \\
\text{(v)} & \kappa(\mathbf{x}, \mathbf{x}') = & \kappa_1(\mathbf{x}, \mathbf{x}') + \kappa_2(\mathbf{x}, \mathbf{x}') \\
\text{(vi)} & \kappa(\mathbf{x}, \mathbf{x}') = & \kappa_1(\mathbf{x}, \mathbf{x}')\,\kappa_2(\mathbf{x}, \mathbf{x}') \\
\text{(vii)} & \kappa(\mathbf{x}, \mathbf{x}') = & \kappa_3\big(\Phi(\mathbf{x}), \Phi(\mathbf{x}')\big) \\
\text{(viii)} & \kappa(\mathbf{x}, \mathbf{x}') = & \mathbf{x}^\top \mathbf{A}\, \mathbf{x}' \\
\text{(ix)} & \kappa(\mathbf{x}, \mathbf{x}') = & \kappa_a(\mathbf{x}_a, \mathbf{x}'_a) + \kappa_b(\mathbf{x}_b, \mathbf{x}'_b) \\
\text{(x)} & \kappa(\mathbf{x}, \mathbf{x}') = & \kappa_a(\mathbf{x}_a, \mathbf{x}'_a)\,\kappa_b(\mathbf{x}_b, \mathbf{x}'_b)
\end{array}
\tag{14.46}
$$

La règle de construction (v) correspond à une simple « addition » de deux espaces. En effet :

$$
\begin{aligned}
\kappa(\mathbf{x}, \mathbf{x}') &= \kappa_1(\mathbf{x}, \mathbf{x}') + \kappa_2(\mathbf{x}, \mathbf{x}') \\
&= \langle \Phi_1(\mathbf{x}), \Phi_1(\mathbf{x}') \rangle + \langle \Phi_2(\mathbf{x}), \Phi_2(\mathbf{x}') \rangle \\
&= \langle [\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}')], [\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}')] \rangle
\end{aligned}
\tag{14.47}
$$

On peut facilement généraliser cette règle de construction à la somme tensorielle de noyaux.

Théorème 14.6 (Noyau *somme directe*)

Soient κ_1 et κ_2 deux fonctions noyau définies respectivement sur $\mathcal{X}_1 \times \mathcal{X}_1$ et sur $\mathcal{X}_2 \times \mathcal{X}_2$, alors leur somme directe

$$
(\kappa_1 \oplus \kappa_2)(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}'_1, \mathbf{x}'_2) = \kappa_1(\mathbf{x}_1, \mathbf{x}'_1) + \kappa_2(\mathbf{x}_2, \mathbf{x}'_2) \tag{14.48}
$$

est une fonction noyau sur $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$. Ici, $\mathbf{x}_1, \mathbf{x}'_1 \in \mathcal{X}_1$ et $\mathbf{x}_2, \mathbf{x}'_2 \in \mathcal{X}_2$.

La règle de construction (vi) est particulièrement importante. Elle permet de combiner des caractéristiques ou attributs des deux espaces de redescription associés à κ_1 et à κ_2 entre eux. On obtient ainsi le *produit tensoriel de noyaux*. Par exemple, ici, les attributs de l'espace de redescription sont les produits de toutes les paires d'attributs du 1$^{\text{er}}$ et du 2$^{\text{nd}}$ espaces. En effet :

Copyright © 2010 Eyrolles.

$$\begin{aligned}\kappa(\mathbf{x},\mathbf{x}') &= \kappa_1(\mathbf{x},\mathbf{x}')\,\kappa_2(\mathbf{x},\mathbf{x}') \\ &= \sum_{i=1}^{D_1} \Phi_1(\mathbf{x})_i \Phi_1(\mathbf{x}')_i \sum_{j=1}^{D_2} \Phi_2(\mathbf{x})_j \Phi_2(\mathbf{x}')_j \\ &= \sum_{i=1}^{D_1}\sum_{j=1}^{D_2} \Phi_1(\mathbf{x})_i\, \Phi_1(\mathbf{x}')_i\, \Phi_2(\mathbf{x})_j\, \Phi_2(\mathbf{x}')_j \\ &= \sum_{i=1}^{D_1}\sum_{j=1}^{D_2} \Phi(\mathbf{x})_{ij}\, \Phi(\mathbf{x}')_{ij}\end{aligned} \tag{14.49}$$

en posant $\Phi(\mathbf{x})_{ij} = \Phi_1(\mathbf{x})_i\, \Phi_2(\mathbf{x})_j$.

Exemple Fonction noyau à exposant simple

Soit la fonction noyau : $\kappa(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}')^s$, on obtient par induction l'espace associé qui est constitué de tous les monômes de degré s :

$$\Phi_i(\mathbf{x}) = \Phi_1(\mathbf{x})_1^{i_1}\, \Phi_1(\mathbf{x})_2^{i_2} \ldots \Phi_1(\mathbf{x})_N^{i_N} \tag{14.50}$$

avec i satisfaisant $\sum_{j=1}^{N} i_j = s$.

Par exemple, pour $s = 5$, et $d = 3$ la dimension de $\mathcal{X}$, on pourrait avoir $\Phi_i(\mathbf{x}) = x_1^2 x_2^1 x_3^2$ où les x_i sont les coordonnées de $\mathbf{x}$ dans $\mathcal{X}$.

Chacune des composantes $\Phi_i(\mathbf{x})$ est un monôme de degré s. Il peut en effet être intéressant de considérer les combinaisons de s attributs de base des objets dans $\mathcal{X}$.

Par exemple, en reconnaissance de scènes visuelles, lorsque les images sont décrites à l'aide de pixels, ne prendre que les produits scalaires directs entre les images correspond à multiplier les intensités des pixels entre elles. Cela signifie aussi que l'*on pourrait redistribuer les pixels arbitrairement dans l'image sans changer le résultat.* Mais la reconnaissance d'objets peut impliquer la présence de plusieurs caractéristiques structurelles ou relationnelles à l'intérieur de l'image. Ainsi, un '8' implique la présence d'un arc de cercle convexe dans le sommet de l'image et d'un arc de cercle concave en bas de l'image. L'absence d'une seule de ces caractéristiques suffit à éliminer la possibilité d'un '8'. Il s'agit donc d'un critère très non linéaire au niveau des pixels. Pour parvenir à le prendre en compte, il faut considérer des régions entières de l'image, c'est-à-dire des combinaisons de pixels.

Il faut noter que l'on arrive très vite à des espaces de redescription de très grande dimension. Ainsi, la taille de l'espace de redescription pour $|\mathcal{X}| = d$ et pour un degré s d'un noyau à exposant simple est de :

$$|\Phi(\mathcal{X})| = \binom{s+d-1}{s} \tag{14.51}$$

où l'on utilise la notation anglo-saxonne pour dénoter le coefficient binomial C_{s+d-1}^{s}.

Ce qui donne par exemple pour des images en 16 × 16 pixels, et des combinaisons de 5 pixels ($s = 5$) : $|\Phi(\mathcal{X})| \approx 10^{10}$. Il est heureux que la fonction noyau à exposant simple nous permette de ne pas manipuler directement ce genre d'espace.

Plus généralement, les fonctions noyau *polynomial*, noyau de *tous les sous-ensembles* et noyau *ANOVA* utilisent les règles de construction (14.46) et correspondent à un produit scalaire dans un espace de redescription construit à partir de certaines combinaisons des attributs de l'espace $\mathcal{X}$ (voir figure 14.14).

Copyright © 2010 Eyrolles.

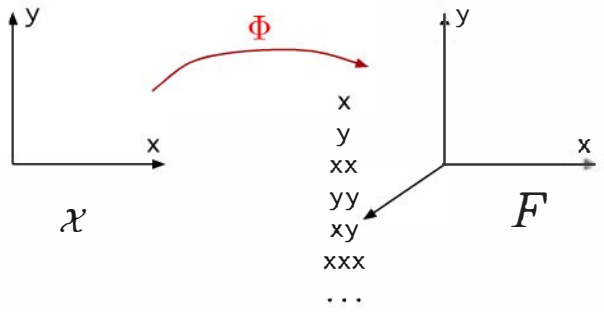


FIG. 14.14: *Construction d'un espace de redescription à partir de combinaisons des attributs de l'espace d'origine $\mathcal{X}$.*

—— EXEMPLE **Caractéristique associée à un noyau polynomial** ——

$$\Phi_p(\mathbf{x}) = x_1^3\, x_2^4\, x_4\, x_5^3 \ldots x_d^5 \tag{14.52}$$

Les noyaux polynomiaux

L'application de la règle (iii) permet de construire des noyaux polynomiaux définis comme $\kappa(\mathbf{x}, \mathbf{x}') = \text{poly}(\kappa_1(\mathbf{x}, \mathbf{x}'))$ où poly est un polynôme à coefficients positifs.

Définition 14.2 (Noyau polynomial)

Un cas particulier est appelé noyau polynomial *:*

$$\kappa_s(\mathbf{x},\mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + R)^p = \sum_{s=0}^{p} \binom{p}{s} R^{p-s} \langle \mathbf{x}, \mathbf{x}' \rangle^s \tag{14.53}$$

Contrairement au noyau à exposant simple qui ne considère que les monômes d'un degré donné p, le noyau polynomial *permet de prendre en compte tous les monômes de degré* $\leq p$.

La dimension de l'espace de redescription $\Phi(\mathcal{X})$ pour le noyau polynomial $\kappa(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + R)^p$ est :

$$\binom{d+p-1}{p-1} + \binom{d+p-1}{p} = \binom{d+p}{p}$$

Il faut noter qu'augmenter le paramètre p peut conduire à du surapprentissage. Le paramètre de pondération R permet de contrôler le poids de chaque monôme. Ainsi, le poids des monômes d'ordre s est $\binom{p}{s} R^{p-s}$. De ce fait, augmenter R diminue les poids des monômes d'ordre élevés.

Les noyaux « tous sous-ensembles »

Un autre type de combinaison des attributs d'origine consiste à considérer les attributs Φ_i de l'espace de redescription $\Phi(\mathcal{X})$ constitués de tous les sous-ensembles d'attributs d'origine, y compris le sous-ensemble vide.

Comme pour les noyaux polynomiaux, on considère ainsi toutes les combinaisons d'attributs jusqu'à un ordre donné p. Mais, contrairement aux noyaux polynomiaux, l'exposant de chaque attribut retenu vaut 1.

Copyright © 2010 Eyrolles.

Soit A un sous-ensemble des d attributs d'origine : $A \subseteq \{1, 2, \ldots, d\}$. On a :

$$\Phi_A(\mathbf{x}) = x_1^{i_1} x_2^{i_2} \ldots x_d^{i_d}$$

avec $\mathbf{i} = (i_1, i_2, \ldots, i_d) \in \{0, 1\}^d$.

On a alors :

$$\kappa_{\subseteq}(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle = \sum_{A \subseteq \{1,\ldots,d\}} \prod_{i \in A} x_i x'_i = \prod_{i=1}^{d} (1 + x_i x'_i) \tag{14.54}$$

où la dernière équation découle d'une application de la loi distributive. On définit ainsi le noyau « tous sous-ensembles ».

Définition 14.3 (Noyau « tous sous-ensemble »)

Le noyau « tous sous-ensemble » est défini par :

$$\kappa_{\subseteq}(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{d} (1 + x_i x'_i) \tag{14.55}$$

On remarquera qu'il s'agit d'un noyau défini comme une somme de produits dans l'espace des caractéristiques (les dimensions de l'espace de redescription), mais qui se calcule, dans $\mathcal{X}$, comme le produit de sommes.

—— Exemple **Caractéristique associée à un noyau « tous sous-ensemble »** ——

$$\Phi_p(\mathbf{x}) = x_1\, x_2\, x_4\, x_5 \ldots x_d \tag{14.56}$$

Les noyaux ANOVA

Une autre méthode pour prendre en compte des combinaisons d'attributs de base fait appel aux fonctions noyau ANOVA. Similaires aux fonctions noyau « tous sous-ensembles », elles permettent de ne prendre en compte que des sous-ensembles de variables de cardinal donné p.

Définition 14.4 (Noyau ANOVA)

Le noyau ANOVA *est défini par :*

$$\kappa_p(\mathbf{x}, \mathbf{x}') = \langle \Phi_p(\mathbf{x}), \Phi_p(\mathbf{x}') \rangle = \sum_{|A|=p} \Phi_A(\mathbf{x})\, \Phi_A(\mathbf{x}') = \sum_{1 \le i_1 < i_2 < \ldots < i_p \le d} \prod_{j=1}^{p} x_{i_j} x'_{i_j} \tag{14.57}$$

—— Exemple **Caractéristique associée à un noyau ANOVA** ——

$$\Phi_p(\mathbf{x}) = \underbrace{x_1\, x_2\, x_4\, x_5 \ldots x_d}_{p} \tag{14.58}$$

Copyright © 2010 Eyrolles.

Il est possible de généraliser les noyaux ANOVA. On peut en effet remarquer qu'il est tentant de remplacer les produits scalaires de base $x_i x'_i$ par des noyaux de base $\kappa_i(\mathrm{x}, \mathrm{x}') = a_i\, x_i\, x'_i$. On peut même aller plus loin et remplacer ces fonctions noyau par des fonctions noyau générales permettant de comparer les objets x et x', et définies éventuellement sur des sous-ensembles non nécessairement disjoints des d attributs de base. On a alors une version plus générale des fonctions noyau ANOVA :

$$\kappa_p(\mathrm{x},\mathrm{x}') = \sum_{1 \leq i_1 < i_2 < \ldots < i_p \leq d} \prod_{j=1}^{p} \kappa_{ij}(\mathrm{x}, \mathrm{x}') \tag{14.59}$$

Ce point de vue mène alors assez naturellement à l'idée de comparer des objets structurés ou composites entre eux. En effet, la comparaison peut se faire par une sorte de calcul récursif dans lequel la comparaison au plus haut niveau résulte de comparaisons menées à des niveaux inférieurs, elles-mêmes pouvant éventuellement être définies récursivement, jusqu'à un niveau de description élémentaire.

Avant de passer à la description de fonctions noyau définies sur des structures, il est intéressant de noter que les noyaux polynomiaux, « tous sous-ensembles » ou ANOVA peuvent *se prêter facilement à des procédés de calcul récursifs*. Nous reportons le lecteur à [STC04] par exemple, pour des détails sur ces approches. Mais, ces procédés de calcul, d'abord poursuivis pour des raisons de complexité calculatoire, ont également inspiré les chercheurs à définir des fonctions noyaux applicables sur des objets structurés. C'est ainsi par deux chemins, *a priori* indépendants, que l'on aboutit à la même idée.

Les noyaux gaussiens

La fonction noyau gaussienne est très employée.

Définition 14.5 (Noyau gaussien)

Le noyau gaussien *est défini par :*

$$\kappa_p(\mathbf{x}, \mathbf{x}') = e^{(-||\mathbf{x}-\mathbf{x}'||^2/2\sigma^2)} \tag{14.60}$$

Il s'agit bien d'une fonction noyau car :

$$||\mathbf{x} - \mathbf{x}'||^2 = \mathbf{x}^\top \mathbf{x} + (\mathbf{x}')^\top \mathbf{x}' - 2\mathbf{x}^\top \mathbf{x}'$$

ce qui donne :

$$\kappa_p(\mathrm{x},\mathrm{x}') = e^{(-\mathrm{x}^\top \mathrm{x}/2\sigma^2)}\, e^{(\mathrm{x}^\top \mathrm{x}'/\sigma^2)}\, e^{(-(\mathrm{x}')^\top \mathrm{x}'/2\sigma^2)}$$

et en vertu de (ii) et (iv) et du fait que $\kappa(\mathrm{x}, \mathrm{x}') = \mathrm{x}^\top \mathrm{x}'$ est une fonction noyau, il s'agit bien d'une fonction noyau.

L'espace de redescription associé est de dimension infinie.

Il est important de noter que le noyau gaussien n'est pas restreint à l'utilisation de la distance euclidienne. On peut remplacer $\mathrm{x}^\top \mathrm{x}'$ par un noyau non linéaire $\kappa(\mathrm{x}, \mathrm{x}')$, ce qui donne :

$$\kappa_p(\mathrm{x},\mathrm{x}') = \exp\left\{-\frac{1}{2\sigma^2}\left(\kappa(\mathbf{x},\mathbf{x}) + \kappa(\mathrm{x}',\mathrm{x}') - 2\kappa(\mathrm{x},\mathrm{x}')\right)\right\} \tag{14.61}$$

Le paramètre σ permet de contrôler la flexibilité de la fonction noyau comme le degré d d'une fonction noyau polynomiale. De petites valeurs de σ correspondent à de grandes valeurs de d,

Copyright © 2010 Eyrolles.

permettant de s'adapter aisément à tout échantillon d'apprentissage et donc à être sujet au sur-apprentissage. Dans ce cas, la matrice de Gram devient proche de la matrice identité. Au contraire, de grandes valeurs de σ réduisent la fonction noyau à une fonction constante, donc à une grande difficulté à s'adapter à autre chose qu'à des classifications triviales.

On peut aussi se représenter chaque point $\mathbf{x}$ comme associé à une fonction dans un espace de Hilbert :

$$\mathbf{x} \longmapsto \Phi(\mathbf{x}) = \kappa(\mathbf{x}, \cdot) = \exp\left(-\frac{||\mathbf{x} - \cdot||^2}{2\sigma^2}\right) \tag{14.62}$$

Chaque point d'apprentissage représente ainsi une fonction de base potentiellement dans une direction orthogonale aux autres fonctions. Mais plus proches sont deux points $\mathbf{x}$ et $\mathbf{x}'$, et plus les directions de leurs fonctions associées sont alignées.

5.3 Construction de fonctions noyau sur des espaces non vectoriels

Connaissant à la fois la signification des fonctions noyau et leurs propriétés, il est possible de chercher à construire des fonctions noyau plus complexes, adaptées à des domaines d'application particuliers. Il est nécessaire que la fonction noyau $\kappa(\mathbf{x}, \mathbf{x}')$ soit symétrique et définie semi-positive, et qu'elle exprime la forme appropriée de similarité entre les objets manipulés. Dans la suite, nous examinons un certain nombre de fonctions noyau qui ont été développées pour traduire des distances dans des espaces particuliers.

L'idée générale est de remplacer les n coordonnées réelles de l'espace de redescription $\mathcal{R}$ par n fonctions noyau de base servant à comparer des aspects des objets considérés.

5.3.1 Noyaux sur des ensembles

L'un des exemples les plus simples de fonction noyau s'appliquant à des données symboliques est celui du *noyau ensemble*. Soit un espace d'entrée constitué de tous les sous-ensembles possibles d'un ensemble A, par exemple les lettres d'un alphabet, ou les pixels d'une image. Si A_1 et A_2 sont deux sous-ensembles de A (e.g. deux ensembles de lettres, ou deux sous-images particulières), une fonction noyau simple permettant de mesurer leur similarité est :

$$\boxed{\kappa(A_1, A_2) = 2^{|A_1 \cap A_2|}}$$

où $A_1 \cap A_2$ est l'intersection de A_1 et de A_2 et $|A|$ est le cardinal de l'ensemble A.

On mesure donc la taille de l'ensemble des parties définissables sur l'intersection entre les ensembles A_1 et A_2 (la taille de l'ensemble des parties d'un ensemble de taille n étant 2^n).

Cette mesure est très utilisé dans la classification de textes, après que des prétraitements aient ramenés ces textes à des « sacs de mots » (*bags of words*, aussi appelée « modèle de l'espace vecteur », *vector space model* (VSM)).

Il est également possible de pondérer les sous-ensembles de A par une distribution de probabilité μ sur ses éléments. On peut alors définir le **noyau intersection** par :

$$\boxed{\kappa_\cap(A_1, A_2) = \mu(A_1 \cap A_2)}$$

c'est-à-dire la probabilité de l'intersection des deux sous-ensembles.

On peut montrer que ces fonctions sont bien des fonctions noyau car elles correspondent à des produits scalaires dans un espace de redescription.

Copyright © 2010 Eyrolles.

5.3.2 Noyaux sur des textes

L'approche la plus rudimentaire pour représenter des textes est d'utiliser la représentation en sacs de mots. Si cette approche a le mérite de la simplicité et, en général, de l'efficacité, en revanche elle ignore des aspects importants des textes, en particulier leur sémantique. Ainsi, des mots synonymes ne sont pas pris en compte dans la similarité entre textes, tandis, qu'au contraire, des mots à plusieurs significations vont compter dans la similarité alors même qu'ils sont peut-être employés dans des sens différents. De même, la position des mots n'est pas prise en compte.

Pour considérer la sémantique des mots, ou du moins leur proximité sémantique, une manière simple consiste à utiliser une matrice de similarité ou de sémantique **S** qui sera employée dans la définition du noyau :

$$\kappa(d_1, d_2) = \phi(d_1)\,\mathbf{S}\mathbf{S}^\top\,\phi(d_2)^\top$$

où d_1 et d_2 sont des textes ou des documents, et $\phi(d)$ est la projection de d dans un espace de redescription.

La matrice **S** peut résulter de la composition de plusieurs étapes de traitement. Par exemple, une opération de pondération des termes, qui peut utiliser l'approche *tf-idf*[9], et une opération mesurant la proximité entre les mots :

$$\mathbf{S} = \mathbf{RP}$$

où **R** est une matrice diagonale spécifiant le poids des termes, et **R** est la matrice de proximité, dans laquelle $\mathbf{P}_{ij} > 0$ quand les termes t_i et t_j sont sémantiquement liés.

Une manière de calculer automatiquement cette matrice de proximité est de calculer les co-occurences des termes dans les documents (des termes en co-occurrences fréquentes étant jugés sémantiquement liés). Une méthode particulière pour cela est l'*analyse sémantique latente* (LSA) (voir chapitre 18). Le lecteur intéressé pourra trouver des détails sur ces techniques en particulier dans [STC04].

5.3.3 Fonctions noyau sur des séquences et des chaînes de caractères

Les fonctions noyau sur les séquences et les chaînes[10] s'appuient généralement sur une projection explicite dans un espace de redescription choisi pour que les distances ou similarités d'intérêt se calculent efficacement. Il s'agit donc de déterminer quelle notion de similarité devrait être calculable dans l'espace de redescription. Par exemple, souhaitons-nous regrouper les chaînes ou séquences par longueur, par composition similaire en sous-séquences, ou toute autre propriété ?

La plupart des fonctions noyau proposées jusqu'ici reposent sur le comptage de sous-chaînes ou de sous-séquences communes aux objets comparés. En génomique, par exemple, cela a un sens car on suppose que l'évolution a opéré par mutations, délétions et insertions successives et, par conséquent, le comptage de sous-séquences ou de sous-chaînes en commun permet d'estimer une distance évolutive entre génomes, et éventuellement, une similarité de fonction.

[9] La représentation *tf-idf* prend en compte la fréquence du terme dans le document étudié (tf) et la fréquence du terme dans l'ensemble des documents (idf). Plus précisément, soit le document d_j et le terme t_i, alors la fréquence du terme dans le document est : $\mathrm{df}_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}$, où $n_{i,j}$ est le nombre d'occurrences du terme t_i dans d_j. Le dénominateur compte le nombre d'occurrences de tous les termes dans le document. La fréquence inverse du document (idf) est défini par : $\mathrm{idf}_i = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$, où $|D|$ est le nombre total de documents, et $|\{d_j : t_i \in d_j\}|$ est le nombre de documents où le terme t_j apparait. Alors $\text{tf-idf}_{i,j} = \mathrm{tf}_{i,j} \cdot \mathrm{idf}_i$.

[10] Conformément à l'usage en informatique, nous appellerons *chaîne* une suite consécutive d'éléments de base, tandis que le terme *séquence* sera utilisé pour une suite ordonnée mais non nécessairement consécutive d'éléments.

Copyright © 2010 Eyrolles.

Soit s une séquence définie sur un alphabet fini de symboles $|\Sigma|$. La séquence vide est notée ϵ. On note Σ^n l'ensemble de toutes les chaînes de taille n, et $\Sigma^\star$ l'ensemble de toutes les chaînes :

$$\Sigma^\star = \bigcup_{n=0}^{\infty} \Sigma^n$$

Les fonctions noyau considérées sont définies par une projection explicite de l'espace $\mathcal{X}$ de toutes les séquences finies définies sur l'alphabet Σ sur un espace vectoriel $\Phi(\mathcal{X})$. Le plus souvent, les coordonnées de $\Phi(\mathcal{X})$ correspondent à un sous-ensemble I de séquences de l'espace d'entrée. Par exemple, cela pourrait être l'espace de toutes les séquences de longueur p, c'est-à-dire un espace de redescription de dimension $|\Sigma|^p$. Les projections Φ sont définies comme des applications :

$$\Phi : s \rightarrow (\phi_u(s))_{u \in I} \in \Phi(\mathcal{X})$$

où I désigne l'ensemble des coordonnées de $\Phi(\mathcal{X})$.

Pour chaque espace de redescription, il peut y avoir plusieurs projections possibles. Par exemple, pour un même espace de redescription, une projection peut consister à calculer chaque coordonnée $\phi_u(s)$ comme étant le nombre de fois où la *chaîne* u est présente dans s, tandis qu'une autre projection pourrait consister à calculer chaque coordonnée comme étant le nombre de fois où la *séquence* u est présente dans s. Dans ce dernier cas, on peut envisager de pondérer la somme des séquences présentes par le nombre de trous dans les séquences considérées.

Comme exemples de fonctions noyau envisagées dans la littérature, nous noterons les suivantes, souvent utilisées, mais n'épuisant pas l'ensemble des fonctions proposées (voir [STC04] pour une description plus complète de ces fonctions noyau).

Fonction noyau spectre

Le spectre d'ordre p d'une séquence s est l'histogramme des fréquences de toutes ses sous-chaînes de longueur p. La comparaison des p_spectres de deux séquences permet de mesurer un certain type de similarité entre ces séquences. Le produit interne entre p_spectres est une fonction noyau, appelée *fonction noyau spectre* (*spectrum kernel function*).

Définition 14.6 (Fonction noyau spectre)

À une fonction noyau à p_spectre est associé un espace de redescription indicé par $I = \Sigma^p$, et la projection définie par :

$$\phi_u^p(s) = |\{(v_1, v_2) : s = v_1 u v_2\}|, \quad u \in \Sigma^p.$$

La fonction noyau spectre *correspondante est définie par :*

$$\kappa_p(s,t) = \langle \Phi^p(s), \Phi^p(t) \rangle = \sum_{u \in \Sigma^p} \phi_u^p(s) \phi_u^p(t) \tag{14.63}$$

— Exemple **Fonction noyau 3_spectre** —

Soient deux chaînes construites sur l'alphabet {A,C,G,T} :

$s =$ GAGTTCTAAT

$t =$ GGATCACTAA

Les sous-chaînes de longueur 3 présentes dans ces deux chaînes sont :

GAG, AGT, GTT, TTC, TCT, CTA, TAA, AAT

Copyright © 2010 Eyrolles.

GGA, GAT, ATC, TAC, CAC, ACT, CTA, TAA

Les sous-chaînes en commun sont CTA et TAA, ce qui donne un produit interne $\kappa(s,t) = 2$.

Des méthodes de calcul récursives permettent de calculer efficacement cette fonction noyau.

Fonction noyau de séquences avec pondération en fonction des trous

Cette fonction noyau prend en compte les trous dans les sous-séquences prises comme coordonnées de l'espace de redescription. Ainsi, comme dans la fonction noyau p_spectre, la ressemblance entre deux séquences est d'autant plus importante que le nombre de sous-séquences communes est grand, mais le poids de ses sous-séquences prend en compte leur dispersion dans les séquences comparées.

EXEMPLE **Pondération des sous-séquences (1)**

Par exemple, la séquence ACG est présente dans les séquences ACGT, ACTTGA et AGGCATGA, mais la première occurrence est plus significative car elle apparaît comme une sous-séquence consécutive de la séquence considérée, alors que la dernière est la moins significative car impliquant une grande dispersion de la sous-séquence ACG dans AGGCATGA.

Afin de pondérer les occurrences des sous-séquences, on introduit un facteur d'amoindrissement $\lambda \in [0,1]$. Ainsi, si la sous-séquence u de longueur $|u|$ est trouvée dans la séquence s avec une longueur $l(u)$, alors on pondère la sous-séquence u avec le poids $\lambda^{l(u)-|u|}$.

EXEMPLE **Pondération des sous-séquences (2)**

En reprenant l'exemple précédent, et avec $\lambda = 0.9$, on aurait le poids de la sous-séquence u = ACG valant 1 dans ACGT, $0.9^{5-3} = 0.81$ dans ACTTGA et $0.9^{7-3} = 0.9^4 \approx 0.66$.

5.3.4 Fonctions noyau à convolution

Étant donné un ensemble d'objets composites, c'est-à-dire constitués de sous-parties, est-il possible de définir une relation de similarité, c'est-à-dire ici une fonction noyau, entre ces objets ?

Par exemple, supposons que l'objet[11] $\mathbf{x} \in \mathcal{X}$ soit composé de sous-parties $x_p \in \mathcal{X}_p$, où $p = 1, \ldots, P$ (les ensembles $\mathcal{X}_p$ n'ayant pas à être égaux ou de tailles égales). Ces sous-parties elles-mêmes pouvant à leur tour être des objets composites.

EXEMPLE **Chaînes**

Soit la chaîne x = ATG et $P = 2$. La chaîne peut être considérée comme composée de x_1 = AT et x_2 = G, ou bien de x_1 = A et de x_2 = TG.

On peut considérer l'ensemble des décompositions licites comme une relation $R(x_1, x_2, \ldots, x_P, \mathbf{x})$ signifiant que « $x_1, x_2, \ldots, x_P$ constituent l'objet composite $\mathbf{x}$ ».

Haussler [Hau99] a été l'un des premiers à examiner comment définir une fonction noyau entre objets composites à partir de mesures de similarité entre leurs sous-parties, c'est-à-dire de fonctions noyau plus locales κ_p définies sur $\mathcal{X}_p \times \mathcal{X}_p$. L'idée est de mesurer la similarité entre les sous-objets grâce à des fonctions noyau locales, puis en remontant récursivement, niveau par niveau, d'effectuer la somme des différentes contributions à la ressemblance globale. On parle de

[11] On continue ici à noter $\mathbf{x}$ les objets de $\mathcal{X}$ même si ce ne sont plus des vecteurs.

Copyright © 2010 Eyrolles.

noyau de convolution car le calcul réalise une sorte de moyenne sur les différents choix possibles de décompositions en sous-objets.

Définition 14.7 (Fonction noyau de convolution)

Le noyau de convolution *de* $\kappa_1, \kappa_2, \ldots, \kappa_P$ *selon la relation de décomposition* R *est défini par :*

$$(\kappa_1 \star \ldots \star \kappa_P)(\mathbf{x}, \mathbf{x}') = \sum_{R} \prod_{p=1}^{P} \kappa_p(x_p, x'_p) \tag{14.64}$$

où la somme est calculée selon toutes les décompositions permises par R *pour décomposer l'objet* $\mathbf{x}$ *en* $x_1, \ldots, x_P$ *et l'objet* $\mathbf{x}'$ *en* $x'_1, \ldots, x'_P$.

Par convention, on posera qu'une somme « vide » est égale à 0, c'est-à-dire que si $\mathbf{x}$ ou $\mathbf{x}'$ ne peut pas être décomposé, alors $(\kappa_1 \star \ldots \star \kappa_P)(\mathbf{x}, \mathbf{x}') = 0$.

S'il n'y a qu'un nombre fini de décompositions possibles on dit que R est finie et on peut montrer que la fonction de convolution ainsi définie est une fonction noyau.

Les exemples les plus simples de fonctions noyau à convolution sont les noyaux *tous sous-ensembles* (*all-subsets kernels*), les noyaux polynomiaux et les noyaux ANOVA.

5.3.5 Calcul des fonctions noyau à convolution

Notons que la définition récursive d'une fonction noyau a une traduction algorithmique permettant d'organiser de manière efficace les calculs. La programmation dynamique est en général utilisée. Une description des algorithmes correspondants peut être trouvée dans la littérature, à commencer par [STC04].

Nous présentons l'algorithme de calcul du noyau « toutes sous-séquences » pour donner un exemple de type d'algorithme. Le noyau « toutes sous-séquences » associe à chaque chaîne un vecteur dont chaque composante est le nombre d'occurrences $\Phi_u^U(x)$ d'une sous-séquence continue ou non dans la chaîne x.

Par exemple, toutes les sous-séquences des chaînes `bac`, `baa` et `cab` sont données dans la table suivante (e.g. la chaîne `baa` contient les sous-chaînes `ba` et `b_a` d'où le 2 dans la colonne `ba`) :

	λ	a	b	c	aa	ab	ac	ba	bc	ca	cb	bac	baa	cab
bac	1	1	1	1	0	0	1	1	1	0	0	1	0	0
baa	1	2	1	0	1	0	0	2	0	0	0	0	1	0
cab	1	1	1	1	0	1	0	0	0	1	1	0	0	1

Cela permet de compter le nombre d'alignements entre chaînes grâce à la fonction noyau : $\kappa(x,y) = \sum_{u \in \Sigma^*} \Phi_u^U(x)\, \Phi_u^U(y)$. Par exemple :

$$\kappa(\texttt{baa}, \texttt{bac}) = 1 + 3 + 2 = 6$$

L'expression $\kappa(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m)$ compte le nombre d'alignements entre les chaînes $\mathbf{x}$ et $\mathbf{y}$ que l'on a décomposées en caractères. On peut noter que ce calcul peut être décomposé en les alignements dans lesquels x_n n'est pas utilisé, soit $\kappa(x_1, x_2, \ldots, x_{n-1}, y_1, y_2, \ldots, y_m)$ et les alignements dans lesquels x_n est aligné avec l'une des $y_1 \ldots y_m$. Ce dernier compte est stocké dans une variable auxiliaire notée `Aux`.

Copyright © 2010 Eyrolles.

Algorithme 22 : Calcul du noyau toutes sous-séquences

Entrées : $x_1, \ldots, x_n, y_1, \ldots, y_m$
Résultat : $\mathrm{K}[n][m] = \kappa(x_1, \ldots, x_n, y_1, \ldots, y_m)$
début
  pour $j \in \{1, .., m\}$ /* L'alignement vide */
  faire
    $\mathrm{K}[0][j] \leftarrow 1$
  fin pour
  pour $i \in \{1, .., n\}$ **faire**
    dernier $\leftarrow 0$
    $\mathrm{Aux}[0] \leftarrow 0$
    pour $j \in \{1, .., m\}$ **faire**
      $\mathrm{Aux}[j] \leftarrow \mathrm{Aux}[dernier]$
    fin pour
    if $x_i = y_i$ **then**
      $\mathrm{Aux}[j] \leftarrow \mathrm{Aux}[dernier] + \mathrm{K}[i-1][j-1]$
    end
    dernier $\leftarrow j$
    pour $j \in \{1, .., m\}$ **faire**
      $\mathrm{K}[i][j] \leftarrow \mathrm{K}[i-1][j] + \mathrm{Aux}[j]$
    fin pour
  fin pour
  retourner $K[n][m]$
fin

5.3.6 Fonctions noyau sur les arbres

Les arbres et les graphes s'inscrivent naturellement parmi les objets composites et structurés. La comparaison de tels objets a été abordée depuis longtemps, en particulier dans ce qui s'appelait la « reconnaissance des formes structurelle ». Cependant, l'avènement des méthodes à noyaux a conduit à un renouvellement des recherches dans la mesure où ces méthodes sont génériques et peuvent s'appliquer à une grande variété de domaines dès lors que l'on est capable de concevoir une fonction noyau, c'est-à-dire, *in fine*, une mesure de similarité appropriée.

Les techniques de calcul récursif découvertes pour certains types de données ont amorcé une pléthore de travaux pour définir, selon une même approche, des noyaux sur des objets structurés susceptibles d'être décomposés en sous-objets, et ainsi de suite.

La structure d'arbre, par exemple, est omniprésente en informatique, et est spécialement utile en biologie évolutive, en traitement du langage naturel, dans le traitement des programmes informatiques ou des documents XML, ou encore pour certains traitements d'image. Il est donc important d'être capable de détecter des régularités dans ces types de données et cela requiert de pouvoir calculer des ressemblances entre arbres.

Étant donné un arbre T, l'arbre S est un *sous-arbre* de T si et seulement si il est sous-graphe connecté de T et les étiquettes des sommets et des arcs de S s'apparient avec les étiquettes des sommets et arcs correspondants dans T. Nous noterons $\mathcal{S}(T)$ l'ensemble des sous-arbres de T.

La plupart des fonctions noyau partagent la même structure : les fonctions noyau évaluées sur les arbres T_1 et T_2, $\kappa(T_1, T_2)$, sont exprimées comme une somme de fonctions noyau locales $\kappa_{\mathcal{S}}(S_1, S_2)$ définies sur toutes les paires de sous-arbres possibles S_1 et S_2 des arbres T_1 et T_2.

À titre d'illustration, nous considérons la fonction noyau *tous sous-arbres*.

Copyright © 2010 Eyrolles.

Définition 14.8 (Fonction noyau *tous sous-arbres*)

La fonction noyau tous sous-arbres *est définie comme :*

$$\kappa(T_1, T_2) = \sum_{S_1 \in \mathcal{S}(T_1), S_2 \in \mathcal{S}(T_2)} \kappa_{\mathcal{S}}(S_1, S_2)$$

où $\kappa_{\mathcal{S}}(S_1, S_2)$ *est elle-même une fonction noyau entre arbres.*

Il en découle que la fonction κ est un noyau, si les fonctions $\kappa_{\mathcal{S}}$ le sont. Plusieurs fonctions locales $\kappa_{\mathcal{S}}$ sont possibles.

1. Si cette fonction locale retourne 1 lorsque les sous-arbres comparées sont égaux et 0 sinon (fonction δ de Kronecker) :
$$\kappa_{\mathcal{S}}(S_1, S_2) = \delta(S_1 = S_2)$$
alors $\kappa(T_1, T_2)$ compte simplement le *nombre de sous-arbres communs* entre T_1 et T_2.
2. Il est aussi possible de compter le *nombre de sous-arbres communs d'une taille N donnée* (mesurée en nombre de nœuds).
$$\kappa_{\mathcal{S}}(S_1, S_2) = \delta(S_1, S_2)\,\delta(n(S_1) = N)\,\delta(n(S_2) = N)$$
où $n(S_i)$ est le nombre de nœuds de S_i, et N une profondeur prédéfinie.
3. On peut également prendre en compte des *sous-arbres communs qui ne sont pas à la même profondeur* dans T_1 et dans T_2 dans une limite fixée à D.
$$\kappa_{\mathcal{S}}(S_1, S_2) = \delta(S_1, S_2)\max(D + 1 - |d(S_1) - d(S_2)|, 0)$$
où $d(S_i)$ est la profondeur de la racine de S_i dans l'arbre T_i, et D est la différence maximale tolérée entre les profondeurs des sous-arbres.

La figure 14.15 montre une base possible de sous-arbres définissant l'espace de redescription $\mathcal{R}$.

Ces fonctions noyau peuvent être calculées en utilisant la programmation dynamique avec une complexité proportionnelle au carré du nombre de sommets [STV04].

5.3.7 Fonctions noyau sur et entre graphes

Les applications impliquant des structures de graphes sont de plus en plus nombreuses. D'une part, la description d'objets sous forme de graphe est très général, d'autres part, les objets eux-mêmes peuvent être liés par des réseaux d'inter-relations qui forment des graphes. Les réseaux sociaux, les molécules complexes rencontrées en biologie, les codes HTML ou XML, les ontologies, appellent naturellement une représentation par graphes.

On distinguera deux types de structures :

1. Les *structures internes*, lorsque la description des objets s'opère en termes de graphe. C'est le cas de molécules biologiques.
2. Les *structures externes*, lorsque les objets sont liés entre eux par les relations d'un graphe, comme dans les réseaux sociaux.

1- Fonctions noyaux entre objets à structure interne sous forme de graphe

La comparaison d'objets décrits par des graphes (structure interne) fait naturellement appel à des noyaux de convolution dans lesquels on combine la comparaison de sous-structures pour

Copyright © 2010 Eyrolles.

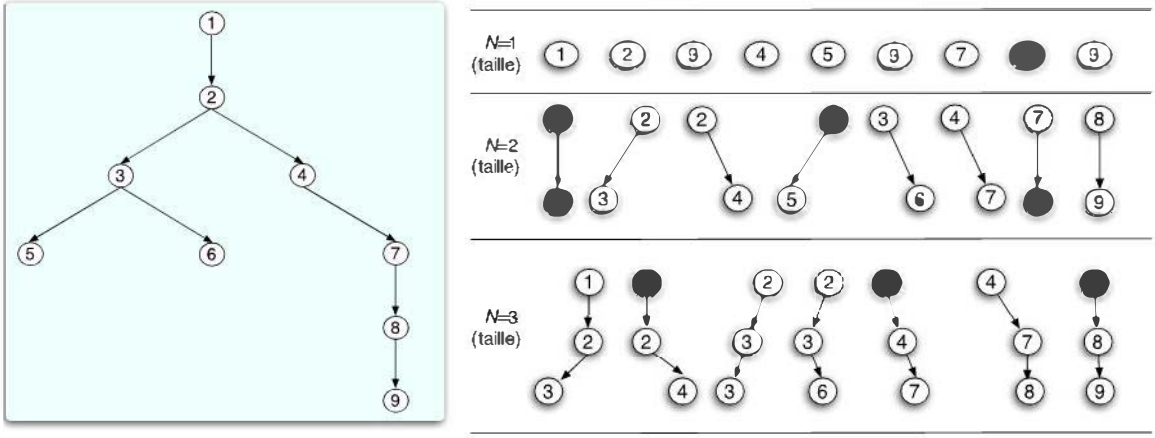


FIG. 14.15: *À gauche : un arbre. À droite, une décomposition en sous-arbres de tailles croissantes.*

obtenir un score de ressemblance global. Une application typique concerne la prédiction de la toxicité ou de la bio-activité de nouvelles molécules chimiques par comparaison avec la structure de molécules connues et répertoriées dans des chimiothèques.

Fonctions noyaux entre chemins

Les fonctions noyau de comparaison entre arbres s'appuyaient sur une décomposition en sous-arbres et sur des fonctions noyau locales de comparaison de sous-arbres. Il est tentant de recourir à la même approche dans le cas des graphes, en décomposant ceux-ci en *chemins* et en comparant ces derniers entre eux.

La majorité des fonctions noyau proposées pour la comparaison de graphes font appel à des sous-graphes sous la forme de chemins ou de séquences d'étiquettes obtenues par traversées des graphes considérés (voir par exemple la figure 14.16).

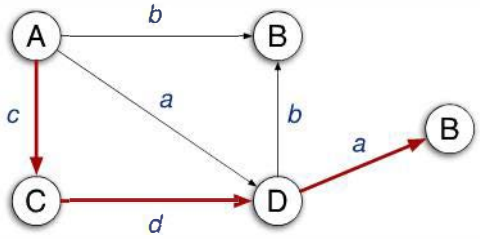


FIG. 14.16: *Exemple de graphe étiqueté. Les sommets sont étiquetés en majuscules et les arcs en minuscules. Le chemin en gras produit la séquence* (A,c,C,d,D,a,B).

L'idée générale est de définir un espace de redescription $\mathcal{R}$ en engendrant des chemins par marches aléatoires dans l'espace des graphes selon une distribution de probabilité à définir. À partir de ces chemins $\mathbf{c} = (x_1, x_2, \ldots, x_l)$, on obtient des séquences $\mathbf{s} = (v_{x_1}, e_{x_1,x_2}, v_{x_2}, \ldots, v_{x_l})$ où les v_{x_i} sont des sommets du graphe, et les e_{x_i,x_j} sont les arcs entre sommets.

On suppose l'existence de fonctions noyau pour la comparaison de nœuds $(\kappa_n(n, n'))$ et pour

Copyright © 2010 Eyrolles.

la comparaison d'arcs ($\kappa_a(a, a')$).

Un exemple de fonction noyau sur les nœuds pourrait être : $\kappa_n(n, n') = \delta(n, n')$, où δ est la fonction Dirac. Si les étiquettes des nœuds sont à valeur réelle, on peut alors prendre la fonction noyau gaussienne : $\kappa_n(n, n') = \exp(-\|f(n) - f(n')\|^2/2\sigma^2)$.

Une *fonction noyau entre chemins* peut alors prendre la forme [KTI03] :

$$\kappa_c(\mathbf{c}, \mathbf{c}') = \begin{cases} 0 & \text{si } l \neq l' \\ \kappa_n(n_{c_1}, n'_{c'_1}) \prod_{i=2}^{l} \kappa_a(a_{c_{i-1},c_i}, a'_{c'_{i-1},c'_i}) \times \kappa(n_{h_i}, n'_{h'_i}) & \text{si } l = l' \end{cases} \quad (14.65)$$

où c et c' sont deux chemins de longueur l et l' respectivement.

Fonctions noyaux entre graphes

Les fonctions noyau entre graphes (estimant une ressemblance entre graphes) diffèrent essentiellement par la manière d'engendrer des chemins et par les pondérations éventuellement utilisées.

En effet, lorsque les graphes considérés sont acycliques, les calculs impliqués dans la définition des noyaux à convolution peuvent s'effectuer par programmation dynamique. En revanche, il faut prendre des précautions pour les graphes cycliques, sous peine de considérer des séquences infinies. Dans ce cas, on utilise en particulier des poids inférieurs à 1 pour parvenir à des équations de point fixe.

Par exemple, une fonction noyau a été proposée (voir [STV04]) de la forme :

$$\kappa_G(G_1, G_2) = \sum_{\mathbf{c}_1} \sum_{\mathbf{c}_2} \kappa_l(\mathbf{c}_1, \mathbf{c}_2)\, \mathbf{P}(\mathbf{c}_1|G_1)\, \mathbf{P}(\mathbf{c}_2|G_2) \quad (14.66)$$

où $\kappa_l(\mathbf{c}_1, \mathbf{c}_2)$ est une fonction noyau locale définie sur des chemins de même longueur et inférieure à l. Les probabilités conditionnelles $\mathbf{P}(\mathbf{c}_i|G_i)$ sont définies à partir des probabilités de départ, de transition et de terminaison sur les sommets de G_i. Cette fonction noyau consiste à mesurer la similarité des distributions de probabilité de chemins aléatoires sur les deux graphes G_1 et G_2.

Nous avons utilisé la notation κ_G pour souligner que l'espace de redescription $\mathcal{R}$ associé est défini globalement, à partir de la notion de graphe et indépendamment des arguments de la fonction noyau. De même, la description en « sacs de mots » est indépendante des textes comparés. On parle de **redescription globale**. Nous allons voir dans la suite que la définition des espaces de redescription peut dépendre des données comparées.

2- Fonctions noyaux entre objets liés par un graphe (structure externe)

Une **tâche d'apprentissage** classique sur graphe consiste à *induire la valeur (réelle) associée à certains nœuds à partir de la connaissance de la valeur en certains autres nœuds*. Il s'agit ainsi de la traduction du problème classique d'induction d'une fonction réelle à partir d'exemples, sauf que la notion de voisinage est maintenant induite par la notion d'adjacence sur un graphe et que l'on a une connaissance *a priori* plus forte des points sur lesquels il faut faire la prédiction (tâche d'apprentissage transductif).

La similarité entre nœuds d'un graphe peut être évaluée de plusieurs manières.

1. La similarité entre les nœuds n_1 et n_2 peut correspondre à la *« facilité » avec laquelle on passe de l'un à l'autre dans le graphe*. Une mesure consiste à considérer un chemin optimal entre les deux nœuds. La sensibilité de cette mesure à l'insertion ou à la délétion

Copyright © 2010 Eyrolles.

de nœuds dans le graphe, nœuds correspondant généralement aux données d'apprentissage, fait préférer une mesure plus globale, correspondant à une sorte de « résistance » (par analogie à une résistance électrique) dans le graphe entre les nœuds. Une fonction ***noyau de diffusion*** a été proposée dans cette optique [KL02]. L'idée est de mesurer la facilité de passage entre deux nœuds du graphe. Pour cela, une possibilité est de faire appel au *Laplacien de graphe* (voir plus bas pour une définition plus précise) défini par :

$$L_{ij} = \begin{cases} d_i & \text{si } i = j \text{ et le degré du nœud } n_i \text{ est } d_i \\ -1 & \text{si les nœuds } n_i \text{ et } n_j \text{ sont directement connectés} \\ 0 & \text{sinon} \end{cases}$$

On définit alors la matrice de Gram $\mathbf{G}_{i,j} = \kappa(n_i, n_j)$ associée au noyau par :

$$\boxed{\mathbf{G} = \exp(-\beta L) = \mathbf{I} + (-\beta \mathbf{G}) + \frac{(-\beta \mathbf{G})^2}{2!} + \ldots} \qquad (14.67)$$

Parce que la fonction exponentielle est associée au développement limité : $1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots$, la matrice de Gram associée au noyau de diffusion prend successivement les corrélations de plus en plus lointaines entre nœuds dans le graphe en les affectant de poids de plus en plus réduits.

2. Une autre mesure de similarité prend en compte la *similarité entre les « relations » des deux nœuds*. Deux nœuds sont jugés d'autant plus semblables qu'ils sont liés (étroitement) à des nœuds communs dans le graphe.

Ces deux notions de similarité font appel à la notion de marches aléatoires dans le graphe, comme pour la comparaison de graphes. Cependant, cette fois-ci, l'espace de redescription associé à la fonction noyau $\kappa_n(n_1, n_2)$ dépend des deux nœuds comparés puisqu'elle s'appuie sur la comparaison des nœuds voisins de n_1 et de n_2. Il s'agit donc de **redescriptions locales**.

Laplacien de graphe

De même que la transformation de Fourier est essentielle dans l'analyse fonctionnelle, la notion de ***Laplacien de graphe*** est au cœur de l'étude des graphes et des variations sur les graphes. La matrice d'incidence et la matrice « Laplacien de graphe » peuvent être considérées comme les analogues discrets des opérateurs gradient et Laplacien dans les espaces euclidiens.

Soit G un graphe arbitraire doté d'un ensemble N de m sommets[12] et d'un ensemble d'arcs entre ces sommets A. Pour chaque arc $a = \{n_1, n_2\}$, on choisit arbitrairement l'un des sommets comme « positif » et l'autre comme « négatif ». La matrice ∇ définie par :

$$\nabla_{ij} = \begin{cases} +1 & \text{si le sommet } n_i \text{ est l'extrémité positive de l'arc } a_j \\ -1 & \text{si le sommet } n_i \text{ est l'extrémité négative de l'arc } a_j \\ 0 & \text{sinon} \end{cases}$$

est appelée *matrice d'incidence* du graphe G.

Soit maintenant une fonction f définie sur les sommets du graphe G, c'est-à-dire $f : N \to \mathbb{R}$, alors $\nabla f : A \to \mathbb{R}$ est définie par :

$$(\nabla f)(a) = f(n_1) - f(n_2) \qquad \text{où } a = \{n_1, n_2\}$$

[12] On ne prend pas m au hasard ici. Souvent en effet, les sommets du graphe correspondront aux exemples d'apprentissage, voir par exemple le chapitre 19 sur l'apprentissage transductif et l'apprentissage semi-supervisé.

Copyright © 2010 Eyrolles.

où n_1 est l'extrémité positive de l'arc a. Il s'agit de l'analogue le plus proche possible de l'opérateur différentiel sur un graphe.

La *matrice d'adjacence* d'un graphe G est telle que chaque composant $\mathbf{A}_{ij}$ est le poids de l'arc $a = \{n_i, n_j\}$. Ces poids doivent être non négatifs et symétriques.

Définition 14.9 (Laplacien de graphe)

Soit D la matrice diagonale des degrés de chaque sommets, c'est-à-dire que D_{nn} est le nombre d'arcs incidents à n, et soit A la matrice d'adjacence de G. Alors la matrice

$$-\Delta = \boldsymbol{D} - \boldsymbol{A}$$

est appelée le Laplacien *du graphe G.*

Ce Laplacien est l'analogue du Laplacien défini sur $\mathbb{R}^d$:

$$f \mapsto \nabla f = \sum_{i=1}^{d} \frac{\partial^2 f}{\partial x_i^2}$$

Il s'agit d'un opérateur symétrique, semi-défini positif et singulier. Le vecteur propre $(1, \ldots, 1)$ est associé à la valeur propre $\lambda_1 = 0$, dont la multiplicité égale le nombre de composantes connexes du graphe G.

Soient les valeurs propres $0 = \lambda_1 \leq \ldots \leq \lambda_m$ du Laplacien Δ et $\{\phi_i\}_{i=1,\ldots,m}$ l'ensemble des vecteurs propres orthogonaux associés. Alors, de même que les fonctions de base de Fourier sont des fonctions propres du Laplacien continu sur $\mathbb{R}^d$, les vecteurs propres de Δ peuvent être considérés comme les équivalents discrets d'une base de Fourier sur le graphe G, dont les fréquences augmentent avec la valeur des valeurs propres associées. En d'autres termes, toute fonction définie sur les sommets d'un graphe G peut être vue comme somme de fonctions de base définies sur ce graphe, et ces fonctions de base « oscillent » d'autant plus fortement sur le graphe que la valeur propre associée est élevée (pour $\lambda_1 = 0$, les fonctions propres associées sont constantes sur chaque composante connexe du graphe) (voir [Sta96] pour plus de détails, et les figures 19.8 et 19.9 du chapitre 19 pour une illustration).

Une fonction f peut donc être décomposée comme somme de fonctions de base sur G :

$$f = \sum_{i=1}^{m} \hat{f}_i \, \phi_i$$

où les ϕ_i sont les fonctions de base et les $\hat{f}_i$ sont les coefficients associés (voir aussi la section 5.2 du chapitre 19).

La matrice du Laplacien étant semi-définie positive, elle peut être employée comme matrice de Gram correspondant à une fonction noyau. On peut en effet démontrer que, pour toute fonction $f : n \in N \rightarrow \mathbb{R}$, on a :

$$f^\top \nabla f = \frac{1}{2} \sum_{i,j} \mathbf{A}_{ij} \left(f(n_i) - f(n_j) \right)^2 \geq 0 \tag{14.68}$$

où l'inégalité est due au fait que $\mathbf{A}$ n'a que des composantes non négatives.

L'équation 14.68 mesure aussi la régularité de la fonction f sur le graphe. En gros, f est régulière si les variations $f(n_i) - f(n_j)$ sont faibles quand $\mathbf{A}_{ij}$ est grand.

Copyright © 2010 Eyrolles.

On peut voir comme un cas particulier le cas de chaque fonction de base, pour lesquelles on a :

$$\phi_i^\top \mathbf{A} \phi_i = \lambda_i \tag{14.69}$$

Ainsi, les fonctions de base associées à des petites valeurs propres sont plus régulières. En conséquence, on peut aussi ré-exprimer la régularité de la fonction f comme :

$$f^\top \nabla f = \sum_{i=1}^{m} \hat{f}_i^2 \lambda_i \tag{14.70}$$

Pour réaliser l'induction de f à partir de la connaissance de valeurs sur un sous-ensemble de nœuds (on parle aussi d'apprentissage par transduction dans ce cas), on émet le plus souvent l'hypothèse que la fonction f est aussi régulière que possible sur le graphe.

En supposant qu'il y ait n sommets dans le graphe G, on considérera des fonctions noyau associées à des matrices de Gram de la forme :

$$\boxed{K = \sum_{i=1}^{m} \mu_i \phi_i \phi_i^\top} \tag{14.71}$$

où les ϕ_i sont les vecteurs propres du Laplacien du graphe, et les $\mu_i \geq 0$ sont les valeurs propres de la matrice de Gram associée à κ.

La matrice de Gram K définit un espace de Hilbert à noyau auto-reproduisant de norme associée :

$$\|f\|_K^2 = \langle f, f \rangle_K = \sum_{i=1}^{m} \frac{\hat{f}_i^2}{\mu_i}$$

pour une fonction prenant la forme : $f = \sum_{i=1}^{m} \hat{f}_i \phi_i$ (on prendra $\frac{\hat{f}_i^2}{\mu_i}$ si $\mu_i = 0$).

5.3.8 Les noyaux génératifs

Il arrive que l'on dispose d'information sur le processus générant les données. Il peut alors être intéressant d'en tirer parti pour définir des fonctions noyau mieux adaptées à la tâche considérée. Plusieurs approches ont été proposées. Nous décrivons brièvement ici les fonctions noyau dérivées de la matrice d'information de Fisher[13].

L'idée sous-jacente dans cette approche est de mesurer l'effort d'adaptation nécessaire du modèle existant pour accommoder la donnée considérée. En particulier, il est nécessaire de disposer d'un modèle de génération des données qui soit paramétré continûment de telle manière que les dérivées par rapport à ces paramètres puissent être calculées.

Soit un modèle paramétrique $\mathbf{p}_{\theta^0}(x) = \mathbf{p}(\mathbf{x}|\boldsymbol{\theta}^0)$ correspondant à la probabilité que l'entrée $\mathbf{x}$ soit produite par le processus génératif dont le vecteur de paramètre θ est mis à $\boldsymbol{\theta}^0$.

On peut mesurer ce qui est nécessaire pour que le modèle accommode une nouvelle donnée en calculant les dérivées partielles de $\mathbf{p}_{\boldsymbol{\theta}^0}(\mathbf{x})$ par rapport à chacun des paramètres et en mesurant dans quelle direction aller pour augmenter la vraisemblance de cette donnée. On peut alors comparer deux données $\mathbf{x}_i$ et $\mathbf{x}_j$ en comparant les adaptations nécessaires pour augmenter la vraisemblance de chacune. Si les adaptations sont similaires, les données seront considérées

[13] Sir Ronald Aylmer Fisher (1890-1962) est l'un des pionniers qui a posé les fondations des statistiques actuelles. Il a en particulier établi les principes de l'analyse d'expériences et élaboré les outils d'analyse de la variance. Il est également à l'origine du principe de vraisemblance maximale.

Copyright © 2010 Eyrolles.

comme proches. Plus précisément, on comparera les gradients associés à l'accommodation de chacune des données.

Formellement, soit $\log \mathcal{L}_{\theta^0}$ la log-vraisemblance du modèle associé à $\theta^0(\mathbf{x})$. Le *score de Fisher* est le gradient de la log-vraisemblance.

$$\mathbf{g}(\theta, \mathbf{x}) = \left(\frac{\partial \log \mathcal{L}_\theta(\mathbf{x})}{\partial \theta_i} \right)_{i=1}^{N}$$

en supposant qu'il y ait N paramètres pour définir le modèle.

La *matrice d'information de Fisher* pour le modèle est :

$$\mathbf{I} = \mathbb{E}\left[\mathbf{g}(\theta^0, \mathbf{x})\, \mathbf{g}(\theta^0, \mathbf{x})^\top\right]$$

où l'espérance est calculée par rapport aux points $\mathbf{x}$ engendrés par la distribution générative des données.

Le score de Fisher correspond à une application sur $\mathbb{R}^N$, ce qui suggère une fonction noyau.

Définition 14.10 (Noyau de Fisher)

Le noyau de Fisher *par rapport au modèle génératif de vecteur de paramètre θ^0 est :*

$$\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\theta^0, \mathbf{x})^\top \mathbf{I}^{-1} \mathbf{g}(\theta^0, \mathbf{x}')$$

Le noyau de Fisher a été utilisé pour plusieurs types de données. Nous renvoyons le lecteur à [STC04] pour plus d'informations.

5.4 Aspects calculatoires

La formulation primale des méthodes à noyaux implique une complexité de l'ordre de $\mathcal{O}(d^3)$, où d est la dimension de l'espace d'entrées. Le passage par un espace de redescription de (très) grande dimension interdisait donc d'envisager d'y avoir recours. Heureusement, la formulation duale et l'astuce des noyaux permettent de passer à une complexité calculatoire de l'ordre de $\mathcal{O}(tm^3)$ où t est la complexité associée au calcul de la fonction noyau et où m est le nombre d'exemples. Il n'en reste pas moins que la complexité reste élevée lorsque le nombre d'exemples est important. C'est pourquoi de nombreux travaux ont porté sur la réduction de cette complexité. L'article de revue [KIKS09] en dresse un panorama.

Il est également important de rendre le calcul des quantités $\kappa(\mathbf{x}, \mathbf{x}')$ efficace puisqu'il doit être répété de très nombreuses fois. Une complexité de l'ordre de $\mathcal{O}(|\mathbf{x}| + |\mathbf{x}'|)$ ou de l'ordre de $\mathcal{O}(|\mathbf{x}| \cdot |\mathbf{x}'|)$ est généralement considérée comme correcte.

6. Les méthodes à noyaux en pratique

6.1 Aperçu des approches calculatoires

Par contraste avec les méthodes d'optimisation adaptées par exemple aux réseaux connexionnistes multicouches qui explorent un espace à multiples optima locaux, les méthodes requises pour les méthodes à noyaux explorent un problème *a priori* plus simple puisqu'il s'agit d'une

Copyright © 2010 Eyrolles.

optimisation d'un problème quadratique qui ne présente qu'un optimum global, qui plus est dans un paysage quadratique.

Plus précisément, le problème à résoudre a la forme générale de la minimisation d'un risque régularisé :

$$h_S^\star = \underset{h\in\mathcal{H}}{\mathrm{ArgMin}}\left[R_{\mathrm{Emp}}(h) + \lambda\|h\|^2\right] \tag{14.72}$$

pour lequel il existe une solution de la forme :

$$h_S^\star(\mathbf{x}) = \langle \mathbf{w}(\alpha), \Phi(\mathbf{x})\rangle = \Big\langle \sum_{i=1}^{m} \alpha_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x})\Big\rangle = \Big\langle \sum_{i=1}^{m} \alpha_i\, \kappa(\mathbf{x}_i, \mathbf{x})\Big\rangle \tag{14.73}$$

Notons que l'on a :

$$\|\mathbf{w}(\alpha)\|^2 = \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j\, \kappa(\mathbf{x}_i, \mathbf{x}_j) = \alpha^\top \mathbf{K}\,\alpha \tag{14.74}$$

où $\mathbf{K}$ est la matrice noyau symétrique (ou encore matrice de Gram) de coefficients $\mathrm{K}_{i,j} := \kappa(\mathbf{x}_i, \mathbf{x}_j)$ et $\alpha^\top$ est la transposée du vecteur α.

En introduisant les équations 14.73 et 14.74 dans l'équation 14.72, on obtient :

$$\begin{aligned} h_S^\star &= \underset{h\in\mathcal{H}}{\mathrm{ArgMin}}\left[\frac{1}{m}\sum_{i=1}^{m} \ell(h(\mathbf{x}_i), u_i) + \lambda\|h\|^2\right] \\ &= \underset{\mathbf{w}}{\mathrm{ArgMin}}\left[\frac{1}{m}\sum_{i=1}^{m} \ell(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle, u_i) + \lambda\|h\|^2\right] \\ &= \underset{\alpha}{\mathrm{ArgMin}}\left[\frac{1}{m}\sum_{i=1}^{m} \ell\Big(\sum_{i=1}^{m} \alpha_i\, \kappa(\mathbf{x}_i, \mathbf{x}), u_i\Big) + \lambda\alpha^\top \mathbf{K}\,\alpha\right] \end{aligned} \tag{14.75}$$

Une somme finie de fonctions convexes, toutes définies sur le même ensemble convexe, est aussi une fonction convexe. De plus, la fonction noyau κ étant, par définition, définie positive, la forme quadratique $\alpha^\top \mathbf{K}\,\alpha$ est convexe par rapport à α. L'hypothèse $h_S^\star$ est donc la solution d'un problème convexe de dimension finie.

De nombreuses **méthodes** existent pour résoudre ces problèmes. Ce qui rend spécifique le problème lié aux SVM est le grand nombre d'exemples qui peut être traité et la dimension éventuellement élevée de $\mathcal{X}$. En revanche, on peut essayer de tirer profit du fait que, souvent, le nombre d'exemples critiques, ceux qui déterminent la solution, est réduit. Parmi les approches existantes, on citera, sans prétention à l'exhaustivité :

- Les *méthodes de gradient* et de *gradient conjugué* sont des méthodes de référence pour ces problèmes. Elles requièrent cependant, pour être efficaces, de stocker en mémoire la matrice noyau $\mathbf{K}$ afin de calculer des inversions de matrices. Cette matrice n'est pas, en général, parcimonieuse, et il faut donc être prêt à stocker de l'ordre de $m(m+1)/2$ coefficients, chacun requérant de l'ordre de 8 octets (en double précision). Cela limite l'usage de ces méthodes à des échantillons de taille inférieure au million d'éléments pour le moment.
- Les *méthodes de point intérieur* sont appropriées pour des tailles d'échantillon petites à modérées et elles sont à la fois fiables et précises. L'idée générale de ces méthodes est de résoudre simultanément la forme primale et la forme duale du problème, en imposant les conditions de Karush-Kuhn-Tucker de façon itérative. Cependant, comme pour les méthodes de gradient, ces méthodes précalculent la matrice $\mathbf{K}$, ce qui est maladroit lorsque de nombreux exemples ont une influence faible ou nulle sur la solution.

Copyright © 2010 Eyrolles.

- Lorsque l'échantillon de données est de grande taille (plusieurs centaines de milliers ou plusieurs millions), les *méthodes de décomposition* sont souvent une solution envisageable. Elles consistent à décomposer le problème en sous-problèmes qui sont résolus de manière itérative. Ici, c'est le vecteur α qui est considéré et optimisé par sous-parties. C'est en particulier le cas de la méthode SMO (*Sequential Minimal Optimization*) proposée par Platt [Pla99] qui pousse l'approche à son extrême en ne considérant à chaque pas que deux composantes du vecteur α.
- Une autre approche adaptée aux très grands échantillons d'apprentissage est une *optimisation en-ligne* dans laquelle les exemples sont considérés séquentiellement, sans réexamen ultérieur en général. Cela permet à la fois de réduire la complexité calculatoire, mais aussi d'obtenir une méthode qui peut bénéficier naturellement de la connaissance préalable d'une solution approchée. Cela ouvre aussi la voie à un apprentissage adaptatif, lorsque les données ne sont plus stationnaires. La méthode LASVM [LCB06] en est un exemple.

6.2 Aperçu de logiciels spécifiques

Il existe de nombreuses boîtes à outils et packages numériques dédiés à l'optimisation convexe (quadratique) sous contraintes et qui peuvent donc être utilisés dans les méthodes à noyau et en particulier pour les SVM. Cependant, des logiciels ont été spécifiquement conçus pour les méthodes à noyau. Ils sont généralement libres d'utilisation et de droit. Une liste assez complète peut être trouvée sur des sites tels que : `www.kernel-machines.org` et `www.support-vector-machines.org`. Nous en mentionnons certains des plus connus ici, sans prétendre qu'ils sont meilleurs que les autres. Chaque package a en général son domaine d'application privilégié, et il est intéressant de déterminer à quel domaine se rapporte son propre problème. À ce sujet, le tableau 10.1 figurant en p.280 de [SS02] constitue un arbre de décision utile pour savoir à quel type de problème on a affaire et quelle famille de méthode est *a priori* la plus adaptée.

`LIBSVM`

Ce logiciel permet de traiter des problèmes de classification, de régression et d'estimation de densité. Il est codé en `C++` et en `Java`. Il est régulièrement mis à jour et des interfaces existent avec d'autres langages tels que `R`, `MATLAB`, `Python`, `Perl` ou logiciels tels que `Weka`. Par ailleurs, ce logiciel est accompagné d'outils appelés `LIBSVM Tools` ainsi que d'une interface graphique très appréciable pour faire des démonstrations pédagogiques en classification ou en régression.

`SVM`^{light}

`SVM`^{light}, écrit en `C` a été l'une des premières implantations permettant de faire de la classification ou de la régression sur des bases de données de taille importante. Ce logiciel fait maintenant partie d'un logiciel plus général, `SVM`^{struct}, dédié à la prédiction de données multivariées ou structurées, tels que des arbres, des séquences ou des ensembles. Cela permet en particulier d'aborder des problèmes comme l'analyse de texte ou l'alignement de séquences biologiques.

`R`

`R` est un logiciel statistique qui peut être utilisé pour le calcul de SVMs pour la classification ou la régression en ayant recours, par exemple, à la fonction `svm` développée par D. Meyer à partir du package `e1071`. L'emploi conjoint des ressources graphiques de `R` permet d'aborder des échantillons de taille moyenne et de réaliser des interfaces faciles avec d'autres traitements statistiques. De plus, le package `svmpath` développé par T. Hastie permet de calculer le chemin de régularisation complet pour des petits échantillons et avec recours à la fonction de perte coude (*hinge loss*) (voir section 6.3).

Copyright © 2010 Eyrolles.

`SimpleSVM`

Écrite en `Matlab`, cette boîte à outils utilise la méthode d'optimisation itérative et de maintien des contraintes actives pour pouvoir traiter des échantillons d'apprentissage de (très) grande taille. `SimpleSVM` permet de réaliser de la classification à une classe (*One Class SVM*), de la classification et de la régression. Réalisant un apprentissage en ligne, `SimpleSVM` permet à la fois de faciliter l'apprentissage si une solution approchée est connue et fournie en entrée, et de réaliser des apprentissages lorsque l'environnement est non stationnaire.

6.3 La détermination des hyper-paramètres

Les paramètres à régler pour utiliser les SVM sont :

1. la *fonction noyau* employée (avec éventuellement ses propres paramètres) ;
2. le *paramètre* C contrôlant le compromis entre fidélité aux données et tolérance aux écarts.

Leur détermination revient à sélectionner le modèle (la classe d'hypothèses) le plus approprié (voir chapitre 3).

Le choix de la fonction noyau est prépondérant. Les figures 14.17 et 14.18 réalisées grâce à un logiciel de démonstration en Java[14] pour des données décrites sur un plan montrent les frontières de décision obtenues avec des fonctions noyau différentes.

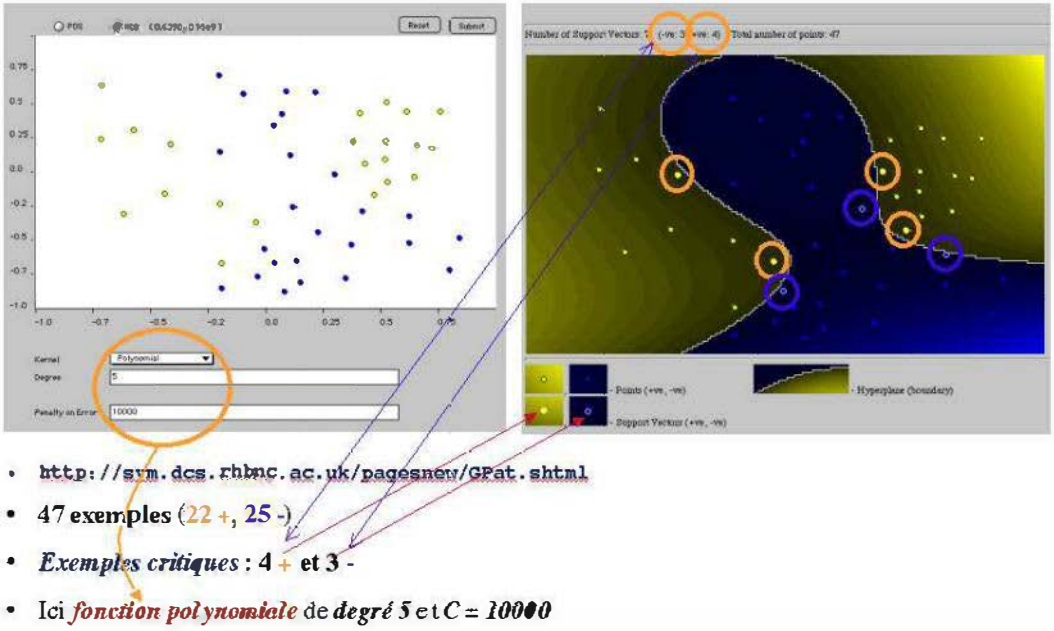


FIG. 14.17: *Dans la fenêtre de gauche, quarante-sept exemples d'apprentissage (vingt-deux de la classe '+' et vingt-cinq de la classe '-') ont été disposés par l'utilisateur. La fenêtre de droite montre la frontière de décision obtenue avec un SVM de noyau polynomial de degré 5 avec une constante $C = 10000$ (c'est-à-dire avec une faible tolérance aux exemples mal classés). La frontière s'appuie sur sept exemples critiques dont quatre dans la classe '+' et trois dans la classe '-'.*

La démarche habituelle pour choisir la fonction noyau et le paramètre C consiste à utiliser une procédure de validation croisée pour estimer la performance correspondant à chaque choix.

[14] Disponible à http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml

Copyright © 2010 Eyrolles.

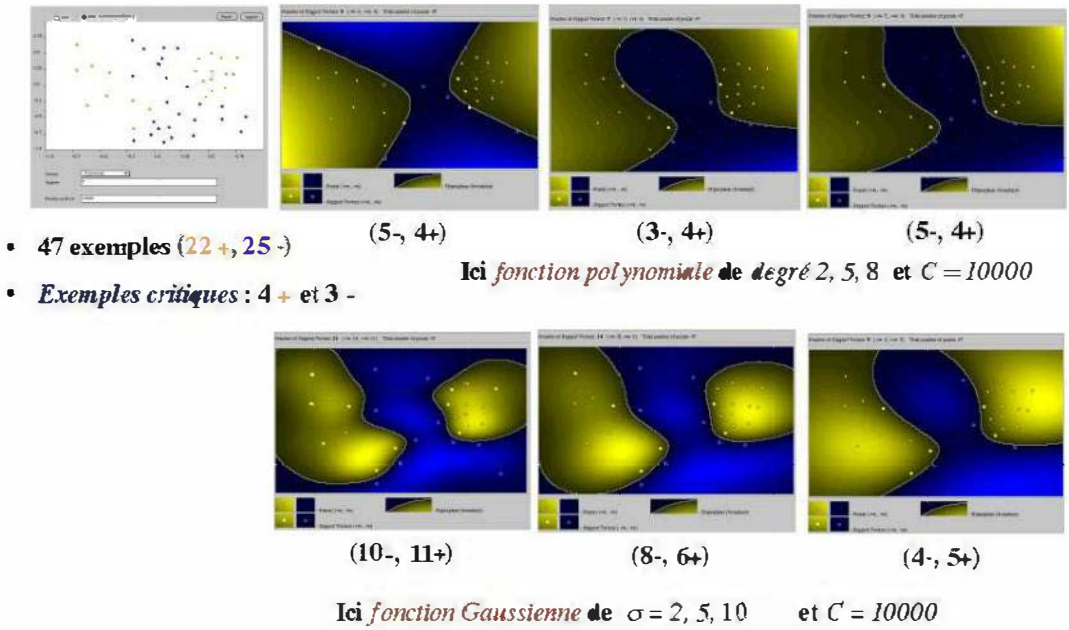


FIG. 14.18: *Pour les mêmes exemples d'apprentissage que ceux de la figure 14.17, on voit l'effet de différents choix de fonctions noyau (polynomiale de degré 2, 5 et 8 en haut et gaussienne d'écart-type 2, 5 et 10 en bas). Le nombre d'exemples critiques de chaque classe est indiqué entre parenthèses en-dessous des imagettes.*

Cependant, il s'agit là d'une approche par essais et erreurs seulement guidée par l'art de l'expert. Des recherches récentes ont porté sur des méthodes plus systématiques, et si possible automatiques, d'exploration.

Ainsi, on voudrait pouvoir sélectionner automatiquement la meilleure fonction noyau[15], ou bien même apprendre une combinaison de fonctions noyaux pour faire face à des données hétérogènes par exemple [SRSS06].

De même, des techniques ont été testées pour le réglage automatique du paramètre C. Celui-ci peut en effet être considéré comme un paramètre de régularisation : plus la valeur de C est grande, plus on favorise les frontières de décision respectant les exemples, c'est-à-dire les plaçant du bon côté des marges. Inversement, des valeurs de C petites correspondent à des solutions plus régulières. Il est donc tentant de chercher à importer les méthodes d'optimisation du paramètre de régularisation vues pour les méthodes linéaires (voir section 5.2 dans le chapitre 9) dans le cadre des méthodes à noyaux. Là aussi on fera varier continûment la valeur de C sur $[0, \infty[$, et on cherchera la valeur pour laquelle l'erreur estimée par validation croisée est minimale. Au lieu de faire varier les coefficients attachés à chaque variable, comme dans la régression linéaire par exemple, ce sont les multiplicateurs de Lagrange α_i attachés aux exemples d'apprentissage qui seront ici contrôlés par les variations de C. La formulation duale du problème d'optimisation à résoudre (voir l'équation 14.22) montre que l'on est proche d'une pénalisation L_1, ce qui est conforme à la parcimonie observée des solutions en terme d'exemples support. Par ailleurs, l'exploration du chemin de régularisation peut être conduite par morceaux (car les multiplicateurs de Lagrange $\alpha_i(C)$ sont des fonctions linéaires par morceaux en fonction de C), donc avec un

[15] Voir le Workshop *Kernel Learning : Automatic Selection of Optimal Kernels* à l'adresse : http://www.cs.nyu.edu/learning_kernels/ comme point d'entrée vers d'autres références sur ce sujet.

Copyright © 2010 Eyrolles.

coût raisonnable [HRTZ04]. Lorsque C s'accroît (ce qui correspond à un λ décroissant), la marge décroît car le système tente de mettre tous les exemples du bon côté de la marge. Les valeurs α_i associées aux points initialement mal placés par rapport à la marge passent de la valeur $\alpha_i = 1$ à la valeur $\alpha_i = 0$ quand ils passent en dehors de la marge. Leur valeur passe de 1 à 0 tandis qu'ils sont exactement sur la marge.

La figure 14.19 (tirée de la thèse de G. Loosli [Loo06]) illustre l'effet de la variation de C dans le système ν-SVM.

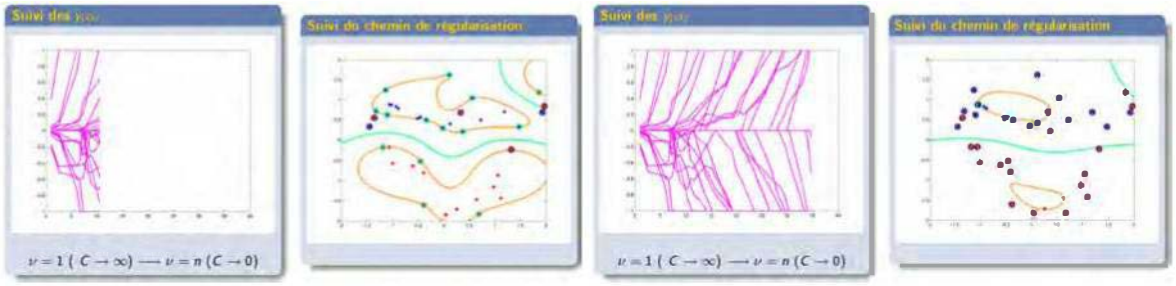


FIG. 14.19: *Pour deux valeurs de C. À gauche, la valeur de C en abscisse variant de ∞ à 0, et les valeurs des coefficients des exemples d'apprentissage en ordonnées $(y_i\alpha_i)$. À droite, la frontière de décision correspondante. Les exemples mal positionnés sont en encerclés.*

6.4 Limites des méthodes à noyaux

Les méthodes à noyaux présentent certains inconvénients et difficultés de mise en œuvre qui peuvent limiter leur mise en pratique. Le premier problème est lié au manque d'interprétabilité des hypothèses produites. Il est en général impossible de comprendre d'où vient la puissance (ou le manque de puissance) de la fonction de décision obtenue. Par exemple, il n'est pas possible de distinguer les variables pertinentes de celles qui ne le sont pas. De même, sauf par la définition des fonctions noyaux utilisées, il est impossible de tirer parti de connaissances *a priori* pour influencer et aider l'apprentissage. En ceci, encore davantage que les réseaux de neurones, les méthodes à noyaux sont des méthodes de type « boîte noire ».

Dans cet ordre idée, il est essentiel de réaliser que la force des méthodes à noyau, c'est-à-dire leur généralité, s'appuyant sur la propriété que de nombreux algorithmes peuvent se mettre sous une forme dans laquelle les données n'interviennent que par des produits scalaires $\langle \mathbf{x}, \mathbf{x}' \rangle$, est aussi une faiblesse, dans la mesure où seules les informations de distances entre points d'apprentissage sont prises en compte. Si d'autres informations, telles que l'orientation dans l'espace $\mathcal{X}$, sont importantes, elles seront, par nécessité, ignorées.

Par ailleurs, les méthodes à noyaux n'échappent pas au « fléau de la dimensionalité » dénoncé par Belhnan en 1961 [Bel61] selon lequel, dès que le nombre de dimensions de l'espace des entrées est important (supérieur à une dizaine), la performance en apprentissage, pour un échantillon d'apprentissage de taille m limitée, dépend de manière cruciale de la fonction de distance choisie pour comparer les entrées. Cela se traduit par la nécessité de définir la ou les fonctions noyau utilisées de manière appropriée au problème, ce qui est souvent loin d'être trivial. Malheureusement, les résultats sont généralement très sensibles à la fonction noyau retenue.

Grâce à la propriété de parcimonie qui peut être associée à l'utilisation de SVM, à savoir qu'une faible proportion des exemples sont critiques lorsque les classes ont peu de recouvrement (faible erreur de Bayes), la fonction de prédiction est facile à calculer. En revanche, le problème

Copyright © 2010 Eyrolles.

d'optimisation quadratique de l'apprentissage d'un hyperplan optimal implique une complexité calculatoire en m^2 multipliée par le nombre d'exemples critiques. De nombreux travaux visent à étendre le domaine d'application des SVM à de grandes tailles de bases de données. Il reste cependant difficile, pour le moment, de traiter des bases de taille $m \gtrsim 10^5$.

7. Bilan et perspectives

Le développement des méthodes à noyaux et particulièrement des séparateurs à vastes marges marque le point de convergence de plusieurs concepts essentiels : le *passage au non linéaire*, grâce à l'astuce des noyaux, d'une grande famille d'algorithmes linéaires ne s'appuyant que sur des produits scalaires dans l'espace des entrées $\mathcal{X}$, la *régularisation automatique* grâce au théorème de représentation, la *marge comme mesure de capacité* de l'espace des hypothèses, la notion de *sélection de modèle a posteriori*.

Finalement, les méthodes à noyaux encapsulent les données par rapport à l'algorithme et permettent ainsi l'utilisation d'un algorithme générique sur différents types de données et pour plusieurs tâches.

On comprend dès lors l'excitation des théoriciens et des praticiens pour cette nouvelle boîte à outils. De plus, ces méthodes peuvent être étendues à des espaces non vectoriels par la définition de distances et de similarité via des fonctions noyau adéquates. Finalement, il existe un lien fort entre ces méthodes et la recherche d'expressions parcimonieuses des régularités du monde, grâce à la régularisation en utilisant la norme ℓ_1, ce qui est au cœur de la théorie de l'information et qui est très séduisant comme démarche générale pour l'apprentissage.

Il reste cependant beaucoup à faire pour mieux maîtriser ces méthodes et les étendre à d'autres applications.

Par exemple, on sait encore très peu de choses sur la convergence du risque réel vers le risque empirique dans le cas de données et de sorties structurées (voir chapitre 20). Globalement, les solutions trouvées sont surtout de nature heuristique sans que les outils formels n'aient, encore, suivi.

Un domaine de recherche lié à celui-ci concerne l'*apprentissage des fonctions noyau* à partir des données.

De nombreux travaux portent sur le développement d'*algorithmes à noyaux en ligne* afin de pouvoir traiter de grandes bases de données (puisque le coût calculatoire des méthodes à noyaux est à peu près une fonction quadratique du nombre d'exemples) et les flux de données. L'adaptation à des concepts cibles dérivants reste encore à explorer (voir chapitre 20).

Finalement, il faut aussi réaliser **les limites** potentielles de ces approches. Ainsi la plupart des travaux privilégient l'utilisation de fonctions noyau gaussiennes, ou en tous les cas à support borné. Si les praticiens se sentent plus à l'aise avec ce genre de fonction noyau en raison de leur plus grande interprétabilité comme mesure de similarité, cela se traduit aussi, implicitement, par un biais vers la recherche de régularité « lisses », c'est-à-dire dont les dérivées sont sévèrement bornées. Or, il existe de nombreux domaines dans lesquels ce biais n'est pas approprié. En général, les régularités complexes impliquent des interdépendances non locales dans les données qui ne peuvent pas être capturées simplement si on utilise des noyaux « locaux » (voir la section 6 du chapitre 10, et l'excellent article de Bengio et Le Cun [BL07]). Plus généralement, il est douteux que les méthodes à noyaux puissent représenter les concepts relationnels complexes sous-jacents à certains problèmes. Cependant, il s'agit là de questions ouvertes.

Copyright © 2010 Eyrolles.

Notes historiques et sources bibliographiques

Le recherche de régularités dans les données en apprentissage artificiel a été dominée par deux courants distincts, l'un se concentrant sur l'apprentissage dans des espaces d'hypothèses à bases d'expressions symboliques et structurés par des opérations logiques, l'autre de nature plus géométrique et statistique. Dans ce dernier courant, trois grandes étapes peuvent être discernées. La *première*, née dans les années trente, en particulier avec Fisher, puis ensuite dans les années soixante avec le perceptron et autres techniques connexionnistes, s'est intéressée à des méthodes efficaces de recherche de régularités linéaires dans l'espace des entrées. Une *deuxième étape*, démarrée dans les années quatre-vingts a permis de dépasser les limitations des premières techniques en autorisant la recherche de régularités non linéaires. Ce fut la révolution des réseaux de neurones et des arbres de décision. Cependant ces méthodes avaient un caractère très heuristiques et étaient l'objet d'analyses théoriques incomplètes car très difficiles. La *troisième étape* a eu lieu vers le milieu des années quatre-vingt-dix avec l'émergence des approches à noyaux permettant la recherche de régularités non linéaires avec l'efficacité et des fondements théoriques réservés jusque là aux méthodes linéaires. Cela a provoqué une vague de travaux sur des développements de nouveaux algorithmes et de nouvelles analyses dans plusieurs communautés : apprentissage artificiel, statistique et même un renouveau d'intérêt en mathématique pour les espaces fonctionnels définis à partir de fonctions noyaux.

Plus précisément, tout en restant très schématique, l'histoire des méthodes à noyaux en apprentissage est marquée par les avancées suivantes.

- L'idée d'utiliser des fonctions noyau comme équivalent au calcul de produits scalaires dans un espace de redescription date de 1964 avec l'article de Aizermann, Bravermann et Rozoener [ABR64] sur la méthode des fonctions potentielles, citée dans la première édition du livre si influent de Duda et Hart [DH73].
- Cette idée a été reprise par Boser, Guyon et Vapnik en 1992, [BGV92], et combinée avec l'idée de recherche d'hyperplans à vastes marges, pour définir la méthode des Séparateurs à Vastes Marges (SVM).
- Entre temps, Vapnik avait introduit le principe de minimisation du risque structurel (SRM) pour résoudre le problème du contrôle de l'induction (et de la sélection de modèle) [Vap82]. Il en a fait une justification des SVM, remise en cause plus tard comme insuffisante.
- Une analyse plus prometteuse a été proposée en 1998 par Shawe-Taylor *et al.* [STBWA98] introduisant l'idée de fonction de félicité (*luckiness framework*) caractérisant une probabilité d'adéquation entre l'algorithme d'apprentissage et les données présentes.
- Pour faire face à la tendance des SVM à marge dure de faire du surapprentissage, la technique des marges douces et des variables ressort a été publiée en 1995 [Vap95]. Des fonctions de perte appropriées, c'est-à-dire plus facilement calculables mais conduisant au même optimum, telle que la fonction coude (*hinge loss*), ont été ensuite proposées. D'autres analyses, fondées sur une idée de robustesse de l'apprentissage à des variations de l'échantillon d'apprentissage ont été également proposées un peu plus tard [BE02].
- La pleine réalisation du potentiel des fonctions noyau pour étendre leur portée au delà des SVM est due en particulier à Schölkhopf [Sch97]. Haussler, de son côté, a fait œuvre de pionnier en montrant en 1999, dans un rapport technique [Hau99], comment construire des fonctions noyaux calculables pour des structures de données complexes non vectorielles.
- Le lien entre fonctions noyau et espace fonctionnel de Hilbert, et en particulier l'équivalence avec un opérateur de régularisation, a été explicité en apprentissage par Smola et Schölkhopf [SS98b] puis par d'autres. La preuve originale du théorème de représentation peut être trou-

Copyright © 2010 Eyrolles.

vée dans [SHSW01]. Une version plus simple de ce théorème avait été prouvé préalablement par Kimeldorf et Wahba [KW70].

- Les années récentes ont vu une analyse extensive du coût calculatoire des méthodes à noyaux (voir par exemple [Joa99]). Il s'en est suivi la mise au point de tout un ensemble de logiciels, cités en partie dans la section 6.2.

La très large popularité des méthodes à noyaux est associée à la publication de nombreux ouvrages entièrement dédiés. Nous citerons en particulier les ouvrages suivants :

- L'ouvrage de Shawe-Taylor et Cristianini [STC04] qui prend le parti de mettre l'accent sur les fonctions noyaux et toutes leurs déclinaisons à des structures de données variées. Leur livre précédent était tourné vers la description des SVM [CST00].
- L'ouvrage de Herbrich [Her02] assez théorique mais très complet sur les problèmes de classification.
- L'ouvrage de Schölkhopf et Smola [SS02] plus heuristique que les précédents et présentant en particulier des méthodes à noyaux pour des tâches diverses : classification, régression, analyse en composantes, etc. Un ouvrage très complet.
- L'ouvrage de Steinwart et Christmann [SC08] qui est le plus théorique. Plus difficile à lire que les autres, il offre aussi des points de vue différents et souvent novateurs.

Résumé

- Les méthodes à noyaux permettent de convertir la plupart des méthodes d'apprentissage linéaires en méthodes adaptées à la recherche de régularités non linéaires en conservant en grande partie les avantages des méthodes linéaires : facilité de mise en œuvre et bons fondements théoriques.
- Les fonctions noyau permettent une redescription implicite de l'espace des entrées dans un espace dont les dimensions sont des corrélations entre dimensions de description des entrées. L'espace de redescription peut également être vu comme un espace fonctionnel dans lequel on cherche une approximation de fonction.
- Les fonctions noyau calculent implicitement le produit scalaire des projections de deux points de l'espace d'entrée.
- Si on peut attacher une sémantique à la projection ϕ vers l'espace de redescription, il est possible d'interpréter le résultat de l'apprentissage. Sinon, l'apprentissage opère comme une « boîte noire ».
- Les fonctions noyaux ont la propriété que toute matrice de Gram associée de dimension finie est positive semi-définie. Le théorème de Mercer est une formulation équivalente de cette propriété pour les espaces vectoriels.
- Il est possible de construire des fonctions noyaux complexes qui peuvent être adaptées à des données non vectorielles en utilisant des opérations simples de combinaison de fonctions noyau plus simples.
- Les SVM se caractérisent par l'utilisation de fonctions noyau, l'absence de minima locaux et la capacité de contrôle de l'induction fournie par l'optimisation des marges. Les SVM permettent la recherche de régularités dans des espaces de grandes dimensions en contrôlant le risque de sur-apprentissage.
- Les méthodes à Vastes Marges forment une classe de méthodes rigoureusement fondée, parmi les plus utilisées et qui offrent un cadre unificateur pour de nombreuses techniques d'apprentissage.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 15

L'apprentissage bayésien et son approximation

Nous savons depuis le chapitre 2 que l'apprentissage bayésien consiste à partir d'hypothèses a priori *pour les réviser en fonction des données d'apprentissage. Cette opération est optimale au sens probabiliste : les hypothèses* a posteriori *obtenues de la sorte sont en effet les plus vraisemblables.*

D'un point de vue opérationnel, l'application de l'apprentissage bayésien nécessite donc, en principe, d'une part une connaissance a priori *sur la vraisemblance des hypothèses en concurrence, mais d'autre part celle la probabilité des données d'apprentissage conditionnellement à ces hypothèses. Si ces valeurs sont connues, l'hypothèse la plus vraisemblable compte tenu des données pourra être sélectionnée. En pratique, ces valeurs ne sont pas connues exactement : elles doivent être estimées. Le but de ce chapitre est donc de présenter quelques méthodes permettant de réaliser ces estimations.*

Il existe une littérature très abondante sur le sujet, provenant des statistiques et de la reconnaissance des formes ; cette dernière discipline insiste sur le côté algorithmique de l'estimation. Nous nous plaçons également de ce côté. Nous présentons d'abord les bases de l'apprentissage bayésien, en particulier dans le cas de la classification, et nous décrivons rapidement les liens entre cette approche et d'autres concepts utiles pour l'apprentissage automatique comme la régression, la comparaison de distributions de probabilités et le principe MDL. Nous décrivons ensuite les méthodes paramétriques d'estimation, qui connaissent un regain de faveur en raison de leurs liens avec les fonctions noyau, elle-mêmes liées aux séparateurs à vastes marges (chapitre 9) et aux réseaux connexionnistes (chapitre 10). Les méthodes non paramétriques, ou k plus proches voisins, sont ensuite présentées, avec une insistance sur l'aspect algorithmique. Pour finir, les méthodes semi-paramétriques sont abordées et le lien est établi avec d'autres méthodes d'apprentissage, comme les arbres de décision (chapitre 13.)

Copyright © 2010 Eyrolles.

Sommaire

1 L'apprentissage bayésien 490
1.1 Présentation 490
1.2 Un petit retour en arrière 492
1.3 L'apprentissage bayésien d'une règle de classification 493
1.4 La classification bayésienne est optimale en moyenne... 493
1.5 ...mais on ne peut que l'approcher 494
1.6 La règle bayésienne et la régression aux moindres carrés 495
1.7 La règle bayésienne et la minimisation de l'entropie croisée 496
1.8 La règle bayésienne et la longueur minimale de description 497
1.9 L'apprentissage bayésien non supervisé 499
2 Les méthodes paramétriques 499
2.1 L'estimation par maximum de vraisemblance 500
2.2 L'estimation des paramètres d'une distribution gaussienne 501
2.3 Le résultat de l'apprentissage 502
2.4 Des hypothèses simplificatrices 504
2.5 Les cas non gaussiens et multigaussiens 505
2.6 L'analyse discriminante 505
2.6.1 L'analyse discriminante linéaire 506
2.6.2 L'analyse discriminante non linéaire 506
2.7 La prédiction bayésienne de la distribution des paramètres 506
3 Les méthodes non paramétriques 509
3.1 Généralités : le problème de l'estimation locale d'une densité 509
3.2 Les fonctions noyau et les fenêtres de Parzen 511
3.2.1 Les fenêtres de Parzen : le cas élémentaire 512
3.2.2 Généralisation à des fonctions noyau gaussiennes 513
3.2.3 Astuces de calcul 514
3.3 Les k-plus proches voisins (k-ppv) 515
3.3.1 Le principe 515
3.3.2 La validité bayésienne 516
3.3.3 Quelques propriétés de convergence 517
3.3.4 Considérations pratiques 518
3.3.5 Les surfaces séparatrices de la règle de décision k-ppv 518
3.3.6 Algorithmique avancée pour les k-ppv 519
4 Les méthodes semi-paramétriques 523
4.1 La discrimination logistique 523
4.2 Les mélanges de distributions 525
4.3 Le cas des réseaux connexionnistes et des arbres de décision 526
4.4 Le cas des réseaux connexionnistes 527
4.5 Les méthodes par segmentation 528

Copyright © 2010 Eyrolles.

VOICI L'HISTOIRE d'un étudiant en mathématiques qui décide pour se changer les idées de s'initier à l'ornithologie, l'art et la science d'identifier les oiseaux. Il se rend vers

un étang sur lequel il a déjà aperçu nager des gros oiseaux. Il y en a quelques-uns ce jour-là. Mais comment savoir à quelle espèce ils appartiennent ? Non loin de là se trouve un autre observateur équipé d'une longue-vue. « Pardon, dit l'étudiant, savez vous quel genre d'oiseaux on voit ici? » « Certainement, répond l'autre. Il n'y a jamais que des cygnes et des oies. Et depuis des années que je viens ici, j'ai eu l'occasion de compter au total environ trois fois plus de cygnes que d'oies. » « Très bien, merci », dit l'étudiant, avant de se livrer à une observation plus attentive.

Il remarque vite que sans une paire de jumelles, tout ce qu'il peut voir sont des oiseaux plutôt blancs et d'autres plutôt sombres. « Excusez-moi de vous interrompre à nouveau, dit-il à son voisin, pouvez-vous m'indiquer le nom des oiseaux blancs et celui des oiseaux noirs ? » L'expert, un peu agacé, lui répond que c'est plus compliqué que cela : « Il y a des cygnes sombres : les jeunes (ils ont moins d'un an). Les adultes sont blancs. La durée de vie d'un cygne est d'environ vingt ans, soit dit en passant. D'autre part, les oies sauvages sont sombres, mais des oies blanches de la ferme d'à côté viennent parfois jusqu'ici. Je dirais qu'une oie sur dix est blanche. Est-ce que cela vous suffit ? » « Parfaitement, merci ». Avisant au loin un oiseau sombre, l'étudiant fait un petit calcul puis montre le bestiau à l'expert. « Je vous prends le pari que nage ici une oie ». L'autre, à la longue-vue, regarde, voit, s'étonne. « Il s'agit bien d'une oie. Vous n'êtes point débutant ! ». « Sachez qu'un bon calcul remplace une lorgnette. »

Que sait l'étudiant ? Que neuf oies sur dix sont sombres, qu'un cygne sur vingt est sombre et qu'un oiseau a trois chances sur quatre d'être un cygne. Que peut-il déduire rationnellement quand il observe un oiseau sombre ?

Avant toute observation, la probabilité *a priori* pour un oiseau d'être un cygne vaut 3/4 et elle vaut 1/4 d'être une oie. La probabilité qu'un oiseau soit sombre sachant qu'il est un cygne vaut 1/20, celle qu'il soit clair sachant qu'il est un cygne vaut 19/20. La probabilité qu'un oiseau soit sombre sachant qu'il est une oie vaut 9/10, celle qu'il soit clair sachant qu'il est une oie vaut 1/10.

De manière plus formelle :

- $\mathbf{P}(oiseau = cygne) = 3/4$. Notons en raccourci : $\mathbf{P}(cygne) = 3/4$
- $\mathbf{P}(oie) = 1/4$
- $\mathbf{P}(couleur = sombre \mid oiseau = cygne) = \mathbf{P}(sombre \mid cygne) = 1/20$
- $\mathbf{P}(clair \mid cygne) = 19/20$
- $\mathbf{P}(sombre \mid oie) = 9/10$
- $\mathbf{P}(clair \mid oie) = 1/10$

Comme l'oiseau observé est sombre, il faut calculer la probabilité conditionnelle $\mathbf{P}(oie \mid sombre)$ qu'il soit une oie sachant que sa couleur est sombre et la probabilité conditionnelle $\mathbf{P}(cygne \mid sombre)$ qu'il soit un cygne sachant que sa couleur est sombre. Il faut ensuite comparer ces deux probabilités conditionnelles. La plus forte des deux donnera l'hypothèse (oie ou cygne) la plus probable, c'est-à-dire la meilleure pour résumer les connaissances.

La *règle de Bayes*, connue de tous les étudiants en mathématiques, permet d'affirmer que :

$$\mathbf{P}(oie \mid sombre) = \frac{\mathbf{P}(sombre \mid oie)\mathbf{P}(oie)}{\mathbf{P}(sombre)}$$

$$\mathbf{P}(cygne \mid sombre) = \frac{\mathbf{P}(sombre \mid cygne)\mathbf{P}(cygne)}{\mathbf{P}(sombre)}$$

Donc :

$$\frac{\mathbf{P}(cygne \mid sombre)}{\mathbf{P}(oie \mid sombre)} = \frac{\mathbf{P}(sombre \mid cygne)\mathbf{P}(cygne)}{\mathbf{P}(sombre \mid oie)\mathbf{P}(oie)}$$

Copyright © 2010 Eyrolles.

et avec les valeurs numériques de notre exemple :

$$\frac{\mathbf{P}(cygne \mid sombre)}{\mathbf{P}(oie \mid sombre)} = \frac{(1/20).(3/4)}{(9/10).(1/4)} = \frac{1}{6}$$

Il y a six chances sur sept que l'oiseau soit une oie : un pari à prendre à six contre un.

Dans cette histoire, il a été question de décision, mais pas d'apprentissage. Où est-il caché ? En réalité, il est complètement résumé par les chiffres que donne l'expert. Par conséquent, ce chapitre développera non seulement les aspects de décision grâce à la règle de Bayes, mais surtout la manière dont l'expert a réussi à apprendre ces chiffres à partir d'exemples, de façon à ce que cette règle de décision soit la plus efficace possible.

1. L'apprentissage bayésien

1.1 Présentation

L'introduction générale à l'apprentissage bayésien a été donnée au chapitre 2. Rappelons que son but est de choisir l'hypothèse $h \in \mathcal{H}$ qui minimise *en moyenne* l'erreur commise vis-à-vis de la fonction cible f, après observation de l'ensemble d'apprentissage $\mathcal{S}$.

On possède un échantillon d'exemples supervisés :

$$\mathcal{S} = \{(\mathbf{x}_1, u_1), \ldots, (\mathbf{x}_i, u_i)), \ldots, (\mathbf{x}_m, u_m)\}$$

On cherche à apprendre la meilleure hypothèse $h \in \mathcal{H}$ pour les expliquer, c'est-à-dire à trouver l'application de $\mathcal{X}$ dans $\mathcal{U}$ qui généralise en moyenne le mieux les couples de $\mathcal{S}$. On suppose ici que l'ensemble $\mathcal{X}$ des valeurs $\mathbf{x}_i$ est numérique. En revanche, les u_i sont des valeurs de supervision qui peuvent être de natures variées. Par exemple :

- Si $\mathcal{X} = \mathbb{R}^d$ et $u_i \in \{\omega_1, \ldots, \omega_C\}$, on veut apprendre une règle de classification.
- Si $\mathcal{X} = \mathbb{R}^d$ et $u_i \in [0, 1]$, on veut apprendre une fonction aléatoire.
- Si $\mathcal{X} = \mathbb{R}^d$ et $u_i \in \mathbb{R}$, on veut apprendre une fonction réelle de d variables réelles.

La règle bayésienne d'apprentissage, ou décision bayésienne, consiste à élire comme meilleure hypothèse dans $\mathcal{H}$ celle qui a la plus grande probabilité connaissant les données d'apprentissage. Il s'agit donc d'une méthode générale dont nous allons allons d'abord décrire le principe avant de l'instancier sur quelques exemples. L'apprentissage d'une règle de classification sera particulièrement détaillé, compte tenu de son importance pratique.

Pour simplifier dans un premier temps, nous allons d'abord nous placer dans le cas où $\mathcal{H}$ est un ensemble fini, par exemple un ensemble de deux classes comme $\{oie, cygne\}$.

Le cas où $\mathcal{H}$ est fini

Commençons par ajouter provisoirement une nouvelle contrainte : supposons que l'on puisse définir une probabilité $\mathbf{P}(\mathcal{S})$ de voir apparaître l'échantillon d'apprentissage $\mathcal{S}$. Cela est facile à écrire si l'ensemble des échantillons possibles est lui aussi fini, mais c'est en général une hypothèse irréaliste. Mais peu importe, cette valeur disparaîtra rapidement des calculs.

Il est alors possible, puisque $\mathcal{H}$ est aussi fini, de définir la probabilité $\mathbf{P}(h)$ et la probabilité $\mathbf{P}(h \mid \mathcal{S})$ de h sachant $\mathcal{S}$, ceci pour tout élément de $\mathcal{H}$.

Copyright © 2010 Eyrolles.

La politique bayésienne consiste par définition à chercher l'hypothèse h^* la plus probable connaissant les exemples. h^* est définie par :

$$\boxed{h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}}\ \mathbf{P}(h \mid \mathcal{S})} \tag{15.1}$$

La formule de Bayes nous dit que :

$$\mathbf{P}(h, \mathcal{S}) = \mathbf{P}(h \mid \mathcal{S})\,\mathbf{P}(\mathcal{S}) = \mathbf{P}(\mathcal{S} \mid h)\,\mathbf{P}(h)$$

Définition 15.1 (Règle du maximum A posteriori : MAP)

L'hypothèse h^* choisie par l'apprentissage bayésien est telle que :

$$\boxed{h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}}\ \frac{\mathbf{P}(\mathcal{S} \mid h)\,\mathbf{P}(h)}{\mathbf{P}(\mathcal{S})} = \underset{h \in \mathcal{H}}{\text{ArgMax}}\ \mathbf{P}(\mathcal{S} \mid h)\,\mathbf{P}(h)} \tag{15.2}$$

La valeur $\mathbf{P}(h)$ *est appelée* probabilité a priori *de l'hypothèse* h.

Une fois les données d'apprentissage observées et la règle de Bayes appliquée, la probabilité de l'hypothèse h *devient* $\mathbf{P}(h \mid \mathcal{S})$ *: c'est la* probabilité a posteriori *de* h.

La règle bayésienne d'apprentissage désigne donc dans $\mathcal{H}$ *la règle de plus forte probabilité* a posteriori. *Cette règle s'appelle aussi la* règle MAP *(maximum* a posteriori*).*

Quand on suppose que les exemples sont des tirages indépendants (i.i.d.), ce qui est une hypothèse naturelle (voir le chapitre 2), on peut écrire :

$$\mathbf{P}(\mathcal{S} \mid h) = \mathbf{P}(h) \prod_{i=1}^{m} \mathbf{P}(\mathbf{z}_i \mid h)$$

Dans ce cas, la règle bayésienne d'apprentissage *MAP* consiste à choisir h^* comme :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}}\ [\mathbf{P}(h) \prod_{i=1}^{m} \mathbf{P}(\mathbf{z}_i \mid h)]$$

ce qui peut aussi s'écrire :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}}\ [\log(\mathbf{P}(h)) + \sum_{i=1}^{m} \log(\mathbf{P}(\mathbf{z}_i \mid h))]$$

Il est courant de se placer dans un cas simplifié, si c'est justifié :

Définition 15.2 (Le classificateur bayésien naïf ou *naive bayesian classifier*)

Si l'on suppose que les attributs de description $\{a_1, ..., a_d\}$ *de l'espace d'entrée* $\mathcal{X}$ *sont indépendants les uns des autres, alors on peut décomposer* $\mathbf{p}(\mathbf{x}|h)$ *en* $\mathbf{P}(a_1 = v_{1\mathbf{x}}|h) \ldots \mathbf{P}(a_d = v_{d\mathbf{x}}|f)$ *soit* $\prod_{i=1}^{d} \mathbf{P}(a_i = v_{i\mathbf{x}}|h)$. *Le classifieur utilisant la règle du aximum* a posteriori *basé sur ces hypothèses est appelé* classificur bayésien naïf.

Il faut noter que les attributs de description sont rarement indépendants les uns des autres (par exemple le *poids* et la *taille*). Pourtant le classifieur bayésien naïf donne souvent des résultats proches de ceux obtenus par les meilleures méthodes connues. Domingos et Pazzani, par exemple, dans [DP97] étudient pourquoi.

Copyright © 2010 Eyrolles.

Définition 15.3 (Règle du maximum de vraisemblance : MLE)

Si on suppose que toutes les hypothèses sont équiprobables dans $\mathcal{H}$, la règle MAP se simplifie et prend le nom de règle du maximum de vraisemblance *: Maximum Likelihood Estimator ou MLE) :*

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \prod_{i=1}^{m} \mathsf{P}(\mathbf{z}_i \mid h) = \underset{h \in \mathcal{H}}{\text{ArgMax}} \sum_{i=1}^{m} \log(\mathsf{P}(\mathbf{z}_i \mid h)) \tag{15.3}$$

Relâchons maintenant l'hypothèse peu crédible selon laquelle il est possible de définir une probabilité $\mathsf{P}(\mathcal{S})$, mais gardons un cadre probabiliste en supposant qu'il existe une *densité* de probabilité $\mathsf{p}(\mathcal{S})$ associée à chaque échantillon $\mathcal{S}$. On peut dans ce cas définir aussi $\mathsf{p}(\mathcal{S} \mid h)$, la densité conditionnelle de $\mathcal{S}$ connaissant l'hypothèse h.

Dans ce cas, la règle MAP s'écrit :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \ \mathsf{p}(\mathcal{S} \mid h)\, \mathsf{P}(h)$$

et la règle du maximum de vraisemblance :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \sum_{i=1}^{m} \log(\mathsf{p}(\mathbf{z}_i \mid h))$$

Le cas où $\mathcal{H}$ est infini

Dans ce cas, il n'est plus possible de définir une probabilité $\mathsf{P}(h)$. Néanmoins, on peut supposer qu'il existe une distribution de probabilités sur $\mathcal{H}$ et donc qu'une densité de probabilité $\mathsf{p}(h)$ peut être définie pour tout h. De même, il existe des densités conditionnelles $\mathsf{p}(h \mid \mathcal{S})$ et $\mathsf{p}(\mathcal{S} \mid h)$.

On cherche alors l'hypothèse h^* telle que :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \ \mathsf{p}(\mathcal{S} \mid h)\, \mathsf{p}(h)$$

soit, avec la règle MAP :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \ [\log(\mathsf{p}(h)) + \sum_{i=1}^{m} \log(\mathsf{p}(\mathbf{z}_i \mid h))]$$

1.2 Un petit retour en arrière

La règle bayésienne optimale a été présentée au chapitre 2 sous une forme légèrement différente, pour des raisons de cohérence des notations dans ce chapitre :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMin}} \ l(h|f)\, \mathsf{p}_{\mathcal{F}}(f)\, \mathsf{p}_{\mathcal{X}}(\mathbf{x}|f)$$

avec :

- $\mathsf{p}_{\mathcal{F}}(f)$ est la densité de probabilité *a priori* de la fonction cible, c'est-à-dire que l'état réel du monde soit f.
- $\mathsf{p}_{\mathcal{X}}(\mathbf{x}|f)$ est la probabilité de l'observation $\mathbf{x}$ connaissant la fonction cible f.
- $l(h|f)$ est le coût de choisir h au lieu de la vraie fonction f.

Ici, nous utilisons la fonction de perte la plus simple (voir annexe 1) :

$$l(u_i, h(\mathbf{x}_i)) \begin{cases} 0 & \text{si } u_i = h(\mathbf{x}_i) \\ 1 & \text{si } u_i \neq h(\mathbf{x}_i) \end{cases} \tag{15.4}$$

Copyright © 2010 Eyrolles.

1.3 L'apprentissage bayésien d'une règle de classification

La première application de l'apprentissage bayésien est la *reconnaissance statistique des formes* ou *l'apprentissage d'une règle de classification* : l'espace $\mathcal{X}$ est égal à $\mathbb{R}^d$ et on veut apprendre une application de $\mathbb{R}^d$ dans $\{\omega_1, \ldots, \omega_C\}$. On suppose qu'à chaque classe ω_i il est possible d'associer une probabilité *a priori* $\mathsf{P}(\omega_i)$ et que tout vecteur correspondant à un point de la classe ω_i peut être considéré comme le résultat d'un tirage aléatoire indépendant de densité $\mathsf{p}(\mathbf{x} \mid \omega_i)$.

Si on veut utiliser les données d'apprentissage pour attribuer avec la meilleure vraisemblance une classe à un point $\mathbf{x}$ quelconque, il faut calculer la probabilité *a posteriori* $\mathsf{P}(\omega_i \mid \mathbf{x})$ d'observer ce point conditionellement à chaque classe ; ce calcul se fait, à partir des valeurs précédentes, par la règle de Bayes[1] :

$$\mathsf{P}(\omega_i \mid \mathbf{x}) = \frac{\mathsf{p}(\mathbf{x} \mid \omega_i)\, \mathsf{P}(\omega_i)}{\mathsf{p}(\mathbf{x})} \tag{15.5}$$

La décision par la règle du maximum *a posteriori* (MAP) s'écrit alors en cherchant la classe ω^* de plus grande probabilité *a posteriori* :

$$\omega^* = \underset{i=1,C}{\text{ArgMax}}\ \mathsf{p}(\mathbf{x} \mid \omega_i)\, \mathsf{P}(\omega_i) \tag{15.6}$$

Le problème de l'apprentissage d'une règle de classification serait donc résolu si l'on connaissait $\mathsf{P}(\omega_i)$ et $\mathsf{p}(\mathbf{x} \mid \omega_i)$. C'est là l'essence des méthodes génératives . Comme on le verra au paragraphe 1.5, ce problème se ramène essentiellement à estimer chaque $\mathsf{p}(\mathbf{x} \mid \omega_i)$ à partir des échantillons d'apprentissage supervisés par la classe ω_i.

La séparation des classes peut aussi se formuler de manière géométrique, selon l'approche dite discriminante.

Définition 15.4

On appelle surface séparatrice Υ_{ij} *entre les deux classes* ω_i *et* ω_j *le lieu des points où les probabilités* a posteriori *d'appartenir à* ω_i *et à* ω_j *sont égales.*

$$\Upsilon_{ij} \text{ est définie par : } \mathsf{P}(\omega_i \mid \mathbf{x}) = \mathsf{P}(\omega_j \mid \mathbf{x})$$

1.4 La classification bayésienne est optimale en moyenne...

On a donc défini la *règle de classification bayésienne* h^* en attribuant au point $\mathbf{x}$ la classe ω^* qui a la plus forte probabilité conditionnellement à $\mathbf{x}$ parmi toutes les classes :

$$h^* \text{ attribue la classe } \omega^* = \underset{i\in\{1,\ldots,C\}}{\text{ArgMax}}\ [\mathsf{P}(\omega_i \mid \mathbf{x})] = \underset{i\in\{1,\ldots,C\}}{\text{ArgMax}}\ [\mathsf{p}(\mathbf{x} \mid \omega_i)\, \mathsf{P}(\omega_i)] \tag{15.7}$$

Cette règle est *optimale en moyenne* : parmi toutes les règles de classification possibles, elle est celle qui minimise la probabilité d'erreur, connaissant la probabilité *a priori* des classes ω_i.

Ce résultat nous est déjà connu : on a traité aux chapitres 2 et 3 de cette notion de probabilité d'erreur et on a vu que la règle de classification bayésienne minimise l'espérance (la moyenne statistique) de mauvaise classification de l'objet $\mathbf{x}$ quand il parcourt $\mathcal{X}$.

[1] Cette formule mélange probabilités et densités de probabilités ; il est facile de vérifier sa validité en partant de sa forme originale $\mathsf{P}(\omega_i \mid x) = \frac{\mathsf{P}(x\in V \mid \omega_i)\mathsf{P}(\omega_i)}{\mathsf{P}(x\in V)}$. V est un volume fini de $\mathbb{R}^d$ et on a par définition : $\mathsf{P}(x \in V) = \int_V \mathsf{p}(x)dx$.

Copyright © 2010 Eyrolles.

La valeur $R(h^*)$ est appelée *erreur bayésienne de classification* ; les algorithmes d'apprentissage de règles de classification visent donc souvent à l'approcher. Une démonstration directe de cette optimalité est donnée en annexe 6.

1.5 ...mais on ne peut que l'approcher

Le problème de l'apprentissage d'une règle de classification serait résolu, sous les hypothèses précédentes, si l'on possédait une connaissance exacte pour chaque classe de la probabilité *a priori* $\mathbf{P}(\omega_i)$ et de la densité conditionnelle $\mathbf{p}(\mathbf{x} \mid \omega_i)$. Mais on ne dispose en pratique que des données d'apprentissage $\mathcal{S} = \{(\mathbf{x}_1, u_1), \ldots, (\mathbf{x}_m, u_m)\}$.

Considérons l'une après l'autre les deux valeurs qui interviennent dans la formule

$$h^* \text{ attribue la classe } \quad \omega^* = \underset{i \in \{1,\ldots,C\}}{\text{ArgMax}} \; [\mathbf{p}(\mathbf{x} \mid \omega_i)\, \mathbf{P}(\omega_i)]$$

L'estimation de la probabilité *a priori* des classes

Pour estimer les valeurs $\mathbf{P}(\omega_i)$, les probabilités *a priori* des classes, on peut procéder de plusieurs manières.

1. Soit on dispose de connaissances *a priori* sur les données, extérieures à l'ensemble d'apprentissage $\mathcal{S}$. Dans ce cas, on doit les utiliser. Par exemple, si on cherche à reconnaître les lettres manuscrites, on doit se fier aux statistiques de leur apparence dans les textes[2], même si cette proportion n'est pas respectée dans $\mathcal{S}$.
2. Sinon, en l'absence d'information particulière, on peut les supposer égales entre elles et donc prendre l'estimateur : $\widehat{\mathbf{P}(\omega_i)} = \frac{1}{C}$.
3. On peut aussi (ce qui est fréquemment fait implicitement) supposer l'échantillon d'apprentissage représentatif et les estimer par les fréquences d'apparition de chaque classe dans cet ensemble[3] : $\widehat{\mathbf{P}(\omega_i)} = \frac{m_i}{m}$.
4. Il existe aussi un estimateur intermédiaire (formule de Laplace)

$$\widehat{\mathbf{P}(\omega_i)} = \frac{m_i + M/C}{m + M}$$

où M est un nombre arbitraire. Cette formule est employée quand m est petit, donc quand les estimations m_i/m sont très imprécises. M représente une augmentation virtuelle du nombre d'exemples, pour lesquels on suppose les classes équiprobables.

Le second cas s'applique par exemple à la reconnaissance des chiffres manuscrits sur les chèques ; en revanche, pour les codes postaux, la troisième méthode est préférable si la base de données a été bien constituée (la proportion de chiffres 0 y est supérieure à celle des autres).

Si le troisième cas semble plus naturel, il peut aussi être trompeur : dans certains problèmes, les classes ne sont pas représentées de la même manière dans l'ensemble d'apprentissage et dans les exemples qu'il faudra classer. Par exemple, un diagnostic médical peut s'apprendre à partir d'un ensemble d'apprentissage comportant un nombre équilibré d'exemples et de contre-exemples, alors que la maladie est rare. Il faudra alors corriger ce biais.

La question de la probabilité *a priori* des classes a déjà été abordée au paragraphe 6.1 du chapitre 3.

[2] 0.15 pour la lettre e, 0.08 pour la lettre s, etc.

[3] Rappelons que le nombre d'exemples total est noté m, alors que m_i désigne le nombre d'exemples supervisés par la classe i.

Copyright © 2010 Eyrolles.

L'estimation des densités conditionnelles a priori

Il reste donc à estimer les densités $\mathbf{p}(\mathbf{x} \mid \omega_i)$. Dans un problème d'apprentissage de règle de classification, on dispose d'un échantillon d'exemples supervisés : le problème se ramène donc à estimer chaque $\mathbf{p}(\mathbf{x} \mid \omega_i)$ à partir des échantillons d'apprentissage supervisés par la classe ω_i.

Indépendamment pour chaque classe, on se trouve donc finalement à devoir estimer une densité de probabilité à partir d'un nombre fini d'observations. C'est un problème tout à fait classique en statistiques. Il faut introduire des hypothèses supplémentaires pour le résoudre (un biais, en terme d'apprentissage artificiel). On a l'habitude de distinguer (voir le chapitre 2) :

- Les méthodes *paramétriques*, où l'on suppose que les $\mathbf{p}(\mathbf{x} \mid \omega_i)$ possèdent une certaine forme analytique ; en général, on fait l'hypothèse qu'elles sont des distributions gaussiennes. Dans ce cas, le problème se ramène à estimer la moyenne et la covariance de chaque distribution ; la probabilité d'appartenance d'un point $\mathbf{x}$ à une classe se calcule alors directement à partir des coordonnées de $\mathbf{x}$. Ce sera l'objet du paragraphe 2.
- Les méthodes *non paramétriques* (paragraphe 3), pour lesquelles on estime localement les densités $\mathbf{p}(\mathbf{x} \mid \omega_i)$ au point $\mathbf{x}$ en observant l'ensemble d'apprentissage autour de ce point. Ces méthodes sont implémentées par la technique des *fenêtres de Parzen* (paragraphe 3.2) ou l'algorithme des *k-plus proches voisins* (paragraphe 3.3).
- Les méthodes semi-paramétriques, (paragraphe 4) pour lesquelles nous ne connaissons pas non plus la forme analytique des distributions de probabilités. Nous supposons cependant que ces distributions appartiennent à des familles et que les « hyper-paramètres » qui les caractérisent à l'intérieur de cette famille peuvent être déterminés.

1.6 La règle bayésienne et la régression aux moindres carrés

Considérons maintenant le problème de la régression, c'est-à-dire de l'apprentissage d'une fonction $f : \mathcal{X} \longrightarrow \mathbb{R}$ à partir d'un ensemble d'exemples $\mathcal{S}$:

$$\mathcal{S} = \{(\mathbf{x}_1, u_1), \ldots, (\mathbf{x}_i, u_i), \ldots, (\mathbf{x}_m, u_m)\}$$

Nous sommes ici dans le cas où l'espace des hypothèses et celui des échantillons possibles sont tous deux infinis : nous emploierons donc des densités de probabilités.

Plaçons-nous d'autre part sous l'hypothèse que la partie supervision de chaque exemple s'écrit :

$$u_i = f(\mathbf{x}_i) + \mathbf{e}_i$$

où $\mathbf{e}_i$ est un tirage aléatoire d'une distribution gaussienne de moyenne nulle et de variance inconnue σ^2. En d'autres termes, on considère qu'il s'agit d'apprendre la fonction f à partir d'exemples bruités[4].

On cherche maintenant dans un ensemble de fonctions hypothèses $\mathcal{H}$ quelle est h^*, la plus probable d'être la fonction f. On sait que :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \ \mathbf{p}(h \mid \mathcal{S})\mathbf{p}(h)$$

Si on suppose toutes les fonctions équiprobables dans $\mathcal{H}$ et les exemples indépendants, on peut appliquer la règle du maximum de vraisemblance :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \ \mathbf{p}(\mathcal{S} \mid h) = \underset{h \in \mathcal{H}}{\text{ArgMax}} \prod_{i=1}^{m} \mathbf{p}(\mathbf{z}_i \mid h)$$

[4] Cette hypothèse est souvent réaliste et en particulier très employée dans le traitement du signal et des images.

Copyright © 2010 Eyrolles.

La relation :

$$u_i = f(\mathbf{x}_i) + \mathbf{e}_i$$

contraint la valeur u_i à être le résultat d'un tirage aléatoire d'une distribution gaussienne de centre $f(\mathbf{x}_i)$ et de variance σ^2. Par conséquent, $\mathbf{p}(\mathbf{u}_i \mid h)$ est le résultat d'un tirage aléatoire d'une distribution gaussienne de centre $h(\mathbf{x}_i)$ et de variance σ^2. Ces contraintes sur la relation entre les $\mathbf{x}_i$ et les $\mathbf{u}_i$ étant ainsi formalisées, l'apprentissage par maximum de vraisemblance se transforme en :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(\mathbf{u}_i - h(\mathbf{x}_i))^2}$$

d'où, en prenant le logarithme :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMin}} \sum_{i=1}^{m} \frac{1}{2\sigma^2}(u_i - h(\mathbf{x}_i))^2 = \underset{h \in \mathcal{H}}{\text{ArgMin}} \sum_{i=1}^{m} (\mathbf{u}_i - h(\mathbf{x}_i))^2 \tag{15.8}$$

Par conséquent, l'hypothèse h^* la plus probable dans $\mathcal{H}$ est celle qui minimise la somme sur tous les exemples $\mathbf{x}_i$ du carré de la différence entre la valeur $h^*(\mathbf{x}_i)$ au point $\mathbf{x}_i$ et la valeur bruitée de la fonction $f(\mathbf{x}_i)$, donnée par $u_i = f(\mathbf{x}_i) + \mathbf{e}_i$.

h^* est donc la meilleure fonction à la fois au sens de la minimisation des moindres carrés et au sens de la règle du maximum de vraisemblance, quand on émet l'hypothèse que le bruit de mesure sur les exemples est gaussien.

Un exemple classique est celui de la régression linéaire dans un plan : la meilleure droite pour approximer un ensemble de points du plan (si l'on suppose ces points issus d'une même droite, mais déplacés par un bruit gaussien) se calcule comme celle qui minimise la somme des distances des points à elle-même (figure 15.1).

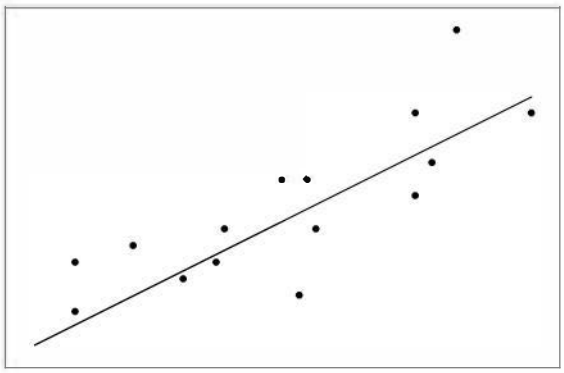


FIG. 15.1: *La meilleure droite pour approximer un ensemble de points.*

1.7 La règle bayésienne et la minimisation de l'entropie croisée

Plaçons-nous maintenant dans un autre cas d'apprentissage : celui d'une fonction aléatoire à deux valeurs $f : \mathbb{R} \longrightarrow \{0, 1\}$.

Les données d'apprentissage qui résument la connaissance sur f sont rassemblées dans l'ensemble $\mathcal{S} = \{(\mathbf{x}_1, u_1), \ldots, (\mathbf{x}_i, u_i), \ldots, (\mathbf{x}_m, u_m)\}$, avec $u_i = f(\mathbf{x}_i) = 0$ ou 1.

L'espace des hypothèses est un ensemble $\mathcal{H}$ dans lequel on cherche la meilleure fonction h^*, au sens bayésien, pour approcher f à partir de $\mathcal{S}$.

Copyright © 2010 Eyrolles.

La probabilité que les données aient été engendrées par la distribution de probabilités h s'écrit, en supposant les données indépendantes entre elles :

$$\mathbf{P}(\mathcal{S} \mid h) = \prod_{i=1}^{m} \mathbf{P}(\mathbf{z}_i \mid h)$$

En utilisant la règle des probabilités conditionnelles :

$$\mathbf{P}(\mathbf{z}_i \mid h) = \mathbf{P}(\mathbf{x}_i, u_i \mid h) = \mathbf{P}(u_i \mid h, \mathbf{x}_i)\,\mathbf{P}(\mathbf{x}_i)$$

$$\mathbf{P}(\mathcal{S} \mid h) = \prod_{i=1}^{m} \mathbf{P}(u_i \mid h, \mathbf{x}_i)\,\mathbf{P}(\mathbf{x}_i)$$

En notant que $\mathbf{P}(u_i \mid h, \mathbf{x}_i)$ vaut $h(\mathbf{x}_i)$ si $u_i = 1$ et $1 - h(\mathbf{x}_i)$ si $u_i = 0$, on peut écrire :

$$\mathbf{P}(u_i \mid h, \mathbf{x}_i) = h(\mathbf{x}_i)^{u_i}\,(1 - h(\mathbf{x}_i))^{1-u_i}$$

D'où :

$$\mathbf{P}(\mathcal{S} \mid h) = \prod_{i=1}^{m} h(\mathbf{x}_i)^{u_i}\,(1 - h(\mathbf{x}_i))^{1-u_i}\,\mathbf{P}(\mathbf{x}_i)$$

La règle bayésienne d'apprentissage h^* s'écrit donc :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}}\ \mathbf{P}(\mathcal{S} \mid h) = \underset{h \in \mathcal{H}}{\text{ArgMax}} \prod_{i=1}^{m} h(\mathbf{x}_i)^{u_i}\,(1 - h(\mathbf{x}_i))^{1-u_i}\,\mathbf{P}(\mathbf{x}_i) \tag{15.9}$$

$\mathbf{P}(\mathbf{x}_i)$ peut être supposé indépendant de h : il est raisonnable de supposer que la probabilité d'observer telle ou telle donnée est indépendante de l'hypothèse que l'on fait. Par conséquent :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \prod_{i=1}^{m} h(\mathbf{x}_i)^{u_i}\,(1 - h(\mathbf{x}_i))^{1-u_i} \tag{15.10}$$

D'où, en passant au logarithme :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \sum_{i=1}^{m} u_i\ \log[h(\mathbf{x}_i)) + (1 - u_i)\ (1 - h(\mathbf{x}_i)] \tag{15.11}$$

Le terme de droite est l'opposé de l'*entropie croisée* entre la distribution des exemples sur $\{0, 1\}$ et la distribution h. Par conséquent, la meilleure fonction aléatoire au sens bayésien est celle qui maximise ce terme. Rappellons que l'entropie croisée a été utilisée pour l'apprentissage des arbres de décision : chaque choix pendant la phase de construction est donc localement optimal au sens bayésien (voir le chapitre 13).

1.8 La règle bayésienne et la longueur minimale de description

Nous anticipons sur un aspect qui sera abordé au chapitre 21 : les rapports entre le principe *MDL* et l'apprentissage bayésien. On peut en effet interpréter la décision bayésienne comme un compromis optimal entre la qualité et la complexité de la solution. Pour cela, il faut raisonner en termes de codage. Rappelons le premier théorème de Shannon [CT91], qui assure le résultat suivant :

Copyright © 2010 Eyrolles.

Théorème 15.1

Soit un ensemble fini d'objets $\{O_1, \ldots, O_n\}$, chacun pouvant apparaître avec la probabilité $\mathsf{P}_1, \ldots, \mathsf{P}_n$. Un code binaire assigne à chaque objet un message sous la forme d'une séquence de bits, de manière bi-univoque. Le code binaire optimal, c'est-à-dire celui qui minimise en moyenne la longueur du message transmis, nécessite $-log_2\ \mathsf{P}_i$ bits pour transmettre le message signifiant que l'objet O_i est apparu.

Nous supposons ici $\mathcal{H}$ de taille finie et nous notons Γ le code optimal correspondant à la distribution des hypothèses dans $\mathcal{H}$. La transmission de l'hypothèse h_i par le code Γ nécessite un nombre de bits que nous notons L_Γ et qui vaut :

$$L_\Gamma(h_i) = -\log_2 \mathsf{P}(h_i)$$

De même, si nous voulons transmettre les données $\mathcal{S}$ en supposant que l'émetteur et le récepteur connaissent tous les deux l'hypothèse h_i, il faut choisir un code optimal correspondant à la distribution conditionnelle des événements $\mathsf{P}(\mathcal{S} \mid h_i)$. Notons ce code Λ. La longueur du message est alors :

$$L_\Lambda(\mathcal{S} \mid h_i) = -\log_2 \mathsf{P}(\mathcal{S} \mid h_i)$$

Maintenant, pour transmettre à la fois les données et une hypothèse h_i, il nous faut donc au mieux :

$$L(h_i, \mathcal{S}) = L_\Gamma(h_i) + L_\Lambda(\mathcal{S} \mid h_i) = -\log_2 \mathsf{P}(h_i) - \log_2 \mathsf{P}(\mathcal{S} \mid h_i)$$

On appelle souvent $L(h_i, \mathcal{S})$ la *longueur de description* de la solution h_i. Elle reflète d'une certaine façon la qualité de h_i, puisqu'elle s'interprète comme le coût de transmettre à la fois une hypothèse et les données connaissant cette hypothèse. Pour expliquer ceci, prenons un exemple.

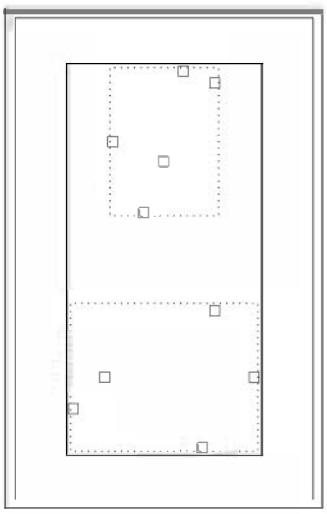


FIG. 15.2: *L'hypothèse constituée d'un seul rectangle (en trait continu) est courte à coder, mais sa précision n'est pas bonne : beaucoup de points sont loin de la frontière. Celle constituée de deux rectangles (en pointillés) est à la fois assez courte à coder et précise : seuls deux points ne sont pas sur la frontière. Une troisième hypothèse est celle constituée de rectangles entourant chaque point : elle est très précise, mais son codage est le plus long. Un bon compromis est donc l'hypothèse constituée de deux rectangles.*

Copyright © 2010 Eyrolles.

EXEMPLE

Imaginons de disposer de données $\mathcal{S}$ qui sont des points de $\mathbb{R}^2$ et supposons qu'une hypothèse h se décrive comme une union de rectangles contenant les données.

Il existe un code optimal Γ pour calculer la longueur de description $L_\Gamma(h)$ d'une hypothèse h, que nous ne connaissons pas. Il est cependant vraisemblable que coder chaque rectangle par les coordonnées de ses deux sommets opposés en diagonale soit une bonne approche. Le nombre de rectangles est donc sans doute relié directement à la complexité du codage de l'hypothèse.

Nous pouvons mesurer la qualité de la description d'une donnée en observant si elle est proche ou non de la frontière d'un rectangle qui la contient. Pour cela, faisons passer une droite horizontale et une droite verticale par ce point et mesurons à quelle distance à gauche, à droite, en haut et en bas elle rencontre la première frontière du rectangle. La plus faible de ces quatre valeurs nous donne une idée de la qualité de la description de ce point : il est satisfaisant qu'une frontière de rectangle passe près du point. Nous pouvons sommer une telle valeur sur tous les points et en prendre l'inverse : nous disposons d'une mesure de qualité d'une hypothèse connaissant les données. Le codage de cette hypothèse, à supposer qu'on connaisse le code optimal Λ, prend exactement la longueur de description $L_\Lambda(\mathcal{S} \mid h)$.

Regardons maintenant deux cas extrêmes :

- On choisit comme hypothèse un rectangle unique, couvrant juste les données. Son codage $L_\Gamma(h_i)$ sera très court, mais la valeur de $Q(\mathcal{S}, h)$ est en revanche élevée. Au total, $L(h_i, \mathcal{S}) L_\Gamma(h_i) + L_\Lambda(\mathcal{S} \mid h_i)$ sera forte.
- On choisit comme hypothèse l'union de tout petits rectangles, chacun entourant une donnée. Cette fois la valeur de $Q(\mathcal{S}, h)$ sera excellente, mais le codage de l'hypothèse sera long. Au final, $L(h_i, \mathcal{S})$ sera forte pour les raisons inverses.

En réalité, il existe un compromis pour lequel $L(h_i, \mathcal{S})$ est minimale. La figure 15.2 en donne une idée sur un exemple.

En revenant au cas général, l'hypothèse ayant la longueur de description minimale est donc en un certain sens la meilleure hypothèse. Elle est caractérisée par :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMin}} \left[-\log_2 \mathbf{P}(h) - \log_2 \mathbf{P}(\mathcal{S} \mid h) \right]$$

En changeant le signe et en prenant l'exponentielle, on constate que h^* est également la meilleure hypothèse bayésienne :

$$h^* = \underset{h \in \mathcal{H}}{\text{ArgMax}} \ \mathbf{P}(\mathcal{S} \mid h)\ \mathbf{P}(h) \tag{15.12}$$

1.9 L'apprentissage bayésien non supervisé

Un autre aspect de l'apprentissage bayésien est sa capacité à être utilisé en apprentissage non supervisé, c'est-à-dire à segmenter des données en familles homogènes. Nous traitons ce cas au chapitre 18.

2. Les méthodes paramétriques

L'une des approches classiques pour estimer des distributions de probabilités est de les représenter à l'aide de fonctions paramétrées. Il s'agit alors d'optimiser la valeur de ces paramètres pour que les fonctions s'adaptent aux données d'apprentissage. On peut alors appliquer la règle

Copyright © 2010 Eyrolles.

de décision bayésienne en remplaçant les probabilités vraies par leur estimation. Le chapitre 2 a déjà abordé ce sujet.

En classification, il s'agit donc d'estimer les grandeurs $\mathsf{P}(\omega_i)$ et $\mathsf{p}(\mathbf{x}|\omega_i)$ pour chaque classe ω_i. Nous avons déjà appris à estimer $\mathsf{P}(\omega_i)$, où m_i est le nombre de formes appartenant à la classe ω_i et m le nombre total de formes observées (paragraphe 1.5). Reste à estimer $\mathsf{p}(\mathbf{x}|\omega_i)$.

2.1 L'estimation par maximum de vraisemblance

Nous allons supposer que la fonction de densité de probabilité $\mathsf{p}(\mathbf{x}|\omega)$ dépend d'un ensemble de paramètres que nous notons $\theta = (\theta_1, \ldots, \theta_L)^\top$. L'ensemble des valeurs que peut prendre θ est noté Θ. Dans le cas d'un problème de classification, nous aurons une fonction par classe afin de représenter $\mathsf{p}(\mathbf{x}|\omega_i)$ pour chaque classe ω_i, ou plus précisément, dans cette approche paramétrique, $\mathsf{p}(\mathbf{x}|\theta_i)$.

Nous disposons d'un échantillon de données $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ de taille m dont nous extrayons le sous-ensemble de $\mathcal{S}^i$ de taille m_i relatif à une classe ω_i :

$$\mathcal{S}^i = \{\mathbf{x}_1, \ldots, \mathbf{x}_{m_i}\}$$

En supposant que ces données soient tirées indépendamment les unes des autres suivant la loi de distribution $\mathsf{p}(\mathbf{x}|\theta_i)$, la densité de probabilité de l'échantillon total s'écrit :

$$\mathcal{L}(\theta_i) \;=\; \mathsf{p}(\mathcal{S}^i|\theta_i) \;=\; \prod_{i=1}^{m_i} \mathsf{p}(\mathbf{x}_i|\theta_i)$$

où $\mathcal{L}(\theta_i)$ est la *vraisemblance* du vecteur de paramètres θ_i pour l'échantillon $\mathcal{S}^i$.

La méthode du maximum de vraisemblance consiste à prendre pour estimation $\hat{\theta}_i$ la valeur du vecteur de paramètres inconnu θ_i qui maximise la vraisemblance que les données aient été produites à partir de la distribution $\mathsf{p}(\mathbf{x}|\theta_i)$.

$$\boxed{\hat{\theta}_i \;=\; \underset{\theta_i \in \Theta}{\mathrm{ArgMax}}\; \mathcal{L}(\theta_i)}$$

Il est plus facile de minimiser l'opposé du logarithme de cette expression, d'où :

$$\begin{aligned} \hat{\theta}_i \;&=\; \underset{\theta_i \in \Theta}{\mathrm{ArgMin}}\; \{-\log \mathcal{L}(\theta_i)\} \\ &=\; \underset{\theta_i \in \Theta}{\mathrm{ArgMin}} \left\{ -\sum_{i=1}^{m_i} \ln \mathsf{p}(\mathbf{x}_i|\theta_i) \right\} \end{aligned} \tag{15.13}$$

Il est à noter que les estimateurs obtenus par la méthode du maximum de vraisemblance sont excellents[5]. Cependant cela n'en garantit pas la qualité pour des échantillons de taille réduite.

Pour la plupart des choix de fonctions de densité, l'optimum $\hat{\theta}_i$ devra être estimé en utilisant des procédures numériques itératives (voir par exemple le chapitre 3). Pour certaines formes fonctionnelles, notamment pour le cas particulier des densités normales (gaussiennes), la solution optimale peut être déterminée de manière analytique, en cherchant la valeur de θ qui annule la dérivée de l'équation 15.13.

[5] En termes techniques, asymptotiquement sans biais et de variance minimale.

Copyright © 2010 Eyrolles.

2.2 L'estimation des paramètres d'une distribution gaussienne

On suppose ici que chaque classe possède une distribution de probabilités de forme paramétrique. On traite en général uniquement le cas gaussien, c'est-à-dire que l'on suppose que la distribution de probabilités de chaque classe est une loi normale, entièrement définie par l'ensemble des paramètres $\theta = (\mu, Q)$ composé de son vecteur moyenne et de sa matrice de covariance.

Faire une telle hypothèse est un biais fort dont la validité peut éventuellement être contrôlée par un test statistique ; il faut garder à l'esprit que cette supposition est dépendante du choix de l'espace de représentation et ne possède aucune justification théorique *a priori*[6]. Mais elle permet d'obtenir une solution analytique simple et un algorithme peu complexe d'apprentissage inductif dans $\mathbb{R}^d$.

Rappelons que la *moyenne* μ d'une densité de probabilité $\mathbf{p}$ dans $\mathbb{R}^d$ est un vecteur de dimension d et sa covariance une matrice $Q(d \times d)$. Si $\mathbb{E}[\mathbf{p}]$ dénote l'*espérance mathématique* de la variable aléatoire $\mathbf{p}$, on a :

$$\begin{aligned} \mu &= \mathbb{E}[\mathbf{x}] \\ Q &= \mathbb{E}[(\mathbf{x}-\mu)(\mathbf{x}-\mu)^\top] \end{aligned}$$

Une distribution de probabilités gaussienne a pour caractéristique d'être entièrement définie par le paramètre θ composé de son vecteur moyenne et de sa matrice de covariance. En effet, en supposant donc la classe ω_i gaussienne[7], sa densité de probabilité s'écrit, dans un espace multidimensionnel :

$$\mathbf{p}(\mathbf{x} \mid \omega_i) = \mathcal{N}(\mu_i, Q_i) = \frac{\mid Q \mid^{-1/2}}{(2\pi)^{d/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_i)^\top Q_i^{-1}(\mathbf{x}-\mu_i)\right\} \tag{15.14}$$

Ce n'est qu'une généralisation de la définition en dimension $d = 1$, plus familière :

$$\mathbf{p}(\mathbf{x} \mid \omega_i) = \mathcal{N}(\mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\} \tag{15.15}$$

Rappellons aussi (voir figure 15.3) que la surface d'équidensité d'une distribution gaussienne est une quadrique (à deux dimensions, c'est une ellipse).

Compte tenu des m_i points d'apprentissage $\mathcal{S}^i = \{\mathbf{x}_1, ..., \mathbf{x}_j, ..., \mathbf{x}_{m_i}\}$, relatifs à la classe w_i (supposée gaussienne), il est démontré en annexe 4 que les meilleures estimations de sa moyenne μ_i et de sa matrice de covariance Q_i au sens du maximum de vraisemblance (c'est-à-dire celles qui maximisent la probabilité d'observer les données d'apprentissage) se calculent simplement par :

$$\widehat{\mu_i} = \frac{1}{m_i}\sum_{j=1}^{m_i} \mathbf{x}_j \tag{15.16}$$

$$\widehat{Q_i} = \frac{1}{m_i}\sum_{j=1}^{m_i} (\mathbf{x}_j - \widehat{\mu_i})(\mathbf{x}_j - \widehat{\mu_i})^\top \tag{15.17}$$

[6] La loi des grands nombres est souvent invoquée à tort pour justifier ce biais.

[7] Ou *normale*, d'où la notation classique $\mathcal{N}$.

Copyright © 2010 Eyrolles.

2.3 Le résultat de l'apprentissage

On peut facilement interpréter dans le cas gaussien la règle bayésienne de décision en termes de surfaces séparatrices ; en effet, le lieu des points où les probabilités d'appartenir aux deux classes ω_i et ω_j sont égales a pour équation :

$$\begin{aligned} \mathbf{p}(\mathbf{x} \mid \omega_i) &= \frac{\mid Q_i \mid^{-1/2}}{2\pi^{d/2}} \exp\left\{-1/2(\mathbf{x}-\mu_i)^\top Q_i^{-1}(\mathbf{x}-\mu_i)\right\} \\ &= \frac{\mid Q_j \mid^{-1/2}}{2\pi^{d/2}} \exp\left\{-1/2(\mathbf{x}-\mu_j)^\top Q_j^{-1}(\mathbf{x}-\mu_j)\right\} = \mathbf{p}(\mathbf{x} \mid \omega_j) \end{aligned}$$

Après simplification et passage au logarithme, on obtient une forme quadratique du type :

$$\mathbf{x}^\top \Phi \mathbf{x} + \mathbf{x}^\top \phi + \alpha = 0 \tag{15.18}$$

où la matrice Φ, le vecteur ϕ et la constante α ne dépendent que de μ_i, μ_j, Q_i et Q_j. On constate donc que faire une hypothèse gaussienne sur la répartition de chaque classe revient à supposer des surfaces de décision quadriques ; à deux dimensions, ce sont des coniques. La figure 15.3 montre un exemple de cette propriété.

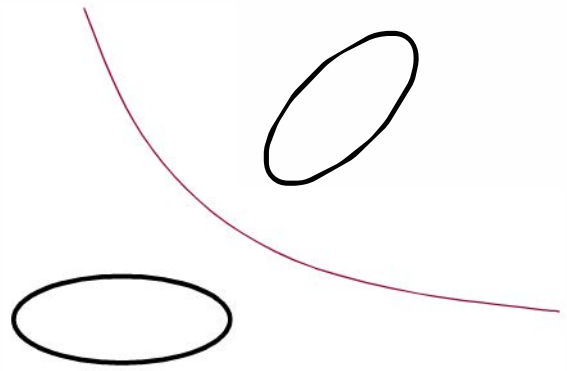


FIG. 15.3: *À deux dimensions, la surface séparatrice de deux classes gaussiennes bidimensionnelles est une conique (ici une branche d'hyperbole) et les surfaces d'équidensité d'une distribution gaussienne sont des ellipses. On a représenté pour chaque classe l'ellipse d'équidensité telle que la probabilité d'appartenir à la classe soit supérieure à 0.5 quand on est à l'intérieur de cette ellipse.*

EXEMPLE **Un exemple à deux dimensions**

Considérons l'ensemble d'apprentissage suivant, comportant huit exemples, quatre pour chacune des deux classes :

$$\mathcal{S} = \left\{ \left(\begin{pmatrix}0\\4\end{pmatrix}, \omega_1\right), \left(\begin{pmatrix}1\\1\end{pmatrix}, \omega_1\right), \left(\begin{pmatrix}3\\3\end{pmatrix}, \omega_1\right), \left(\begin{pmatrix}4\\0\end{pmatrix}, \omega_1\right), \left(\begin{pmatrix}4\\0\end{pmatrix}, \omega_2\right), \left(\begin{pmatrix}7\\1\end{pmatrix}, \omega_2\right), \left(\begin{pmatrix}8\\4\end{pmatrix}, \omega_2\right), \left(\begin{pmatrix}5\\3\end{pmatrix}, \omega_2\right) \right\}$$

La modélisation gaussienne de la classe ω_1 amène les paramètres suivants, en notant $\mathbf{x}_{1j}$ les vecteurs des exemples de cette classe :

$$\widehat{\mu_1} = \frac{1}{4}\Sigma_{j=1}^{4}\mathbf{x}_{1j} = \frac{1}{4}\begin{pmatrix}0+1+3+4\\4+1+3+0\end{pmatrix}\begin{pmatrix}2\\2\end{pmatrix}$$

Copyright © 2010 Eyrolles.

$$\widehat{Q_1} = \frac{1}{4}\sum_{j=1}^{4}(\mathbf{x}_{1j} - \widehat{\mu_1})(\mathbf{x}_{1j} - \widehat{\mu_1})^\top$$

Pour $j = 1$, par exemple, le terme de cette somme vaut :

$$\begin{pmatrix} 0-2 \\ 4-2 \end{pmatrix}(0-2\ 4-2) = \begin{pmatrix} 4 & -4 \\ -4 & 4 \end{pmatrix}$$

On trouve au total :

$$\widehat{Q_1} = \frac{1}{4}\left(\begin{pmatrix} 4 & -4 \\ -4 & 4 \end{pmatrix} + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} 4 & -4 \\ -4 & 4 \end{pmatrix}\right) = \begin{pmatrix} 5/2 & -3/2 \\ -3/2 & 5/2 \end{pmatrix}$$

D'où : $|\widehat{Q_1}| = 4$ et : $\widehat{Q_1}^{-1} = \begin{pmatrix} 5/8 & 3/8 \\ 3/8 & 5/8 \end{pmatrix}$

De même, on trouve :

$\widehat{\mu_2} = \begin{pmatrix} 6 \\ 2 \end{pmatrix}$, $\widehat{Q_2} = \begin{pmatrix} 5/2 & 3/2 \\ 3/2 & 5/2 \end{pmatrix}$, $|\widehat{Q_1}| = 4$ et $\widehat{Q_2}^{-1} = \begin{pmatrix} 5/8 & -3/8 \\ -3/8 & 5/8 \end{pmatrix}$

Puisque les deux classes ont une matrice de covariance de même déterminant, la surface discriminante entre ω_1 et ω_2 est simplement définie par l'équation :

$$(\mathbf{x} - \widehat{\mu_1})^\top \widehat{Q_1^{-1}}(\mathbf{x} - \widehat{\mu_1}) = (\mathbf{x} - \widehat{\mu_2})^\top \widehat{Q_2^{-1}}(\mathbf{x} - \widehat{\mu_2})$$

$$(x_1 - 2\ \ x_2 - 2)\begin{pmatrix} 5 & 3 \\ 3 & 5 \end{pmatrix}\begin{pmatrix} x_1 - 2 \\ x_2 - 2 \end{pmatrix} = (x_1 - 6\ \ x_2 - 2)\begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix}\begin{pmatrix} x_1 - 6 \\ x_2 - 2 \end{pmatrix}$$

Après développement, on trouve la surface d'équation :

$$(x_1 - 4)(x_2 + 4/3) = 0$$

Autrement dit, cette surface séparatrice est une hyperbole dégénérée en deux droites qui partage le plan en quatre zones, deux affectées à la classe ω_1 et deux à ω_2. Cet exemple est représenté sur la figure 15.4.

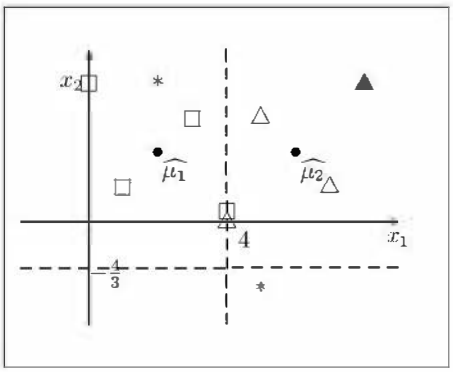


FIG. 15.4: *Deux classes supposées gaussiennes sont représentées chacune par quatre points d'apprentissage notés par* □ *et* △. *L'estimation des moyennes* $\widehat{\mu_1}$ *et* $\widehat{\mu_2}$ *est indiquée. La surface séparatrice entre les deux classes est l'hyperbole dégénérée composée des deux droites en pointillé. Par conséquent, les deux points* * *sont classés comme appartenant à la classe* □.

Copyright © 2010 Eyrolles.

2.4 Des hypothèses simplificatrices

Dans un espace de dimension d avec C classes, le nombre de paramètres évalués par les formules 15.16 et 15.17 est d pour chaque moyenne, et $d(d+1)/2$ pour chaque matrice de covariance, qui est symétrique ; soit au total : $(d^2 + 3d)/2$ paramètres par classe et $C(d^2 + 3d)/2$ au total. Si le nombre m_i de points de la classe courante est faible, la précision de ces estimations est mauvaise. On peut donc être amené à faire des hypothèses plus simples ; par exemple supposer que toutes les classes ont la même matrice de covariance, ou que celle-ci est diagonale pour chaque classe, ou même les deux à la fois. Voyons les conséquences de chacun de ces biais supplémentaires.

Toutes les classes sont supposées avoir la même matrice de covariance

Dans ce cas, l'estimation des valeurs de la matrice de covariance par la formule 15.17 peut se faire une seule fois sur tous les points de l'ensemble d'apprentissage et non plus classe par classe. Ceci conduit donc à une matrice unique que l'on note $\widehat{Q}$; les moyennes sont cependant estimées séparément pour chaque classe par une valeur $\widehat{m_i}$. Le nombre de paramètres à estimer vaut : $Cd + (d^2 + d)/2$.

La règle de décision peut être interprétée de la façon suivante : pour un point inconnu x, on mesure sa « distance de Mahalanobis » aux vecteurs moyens de chaque classe et on lui attribue la classe la plus proche au sens de cette distance. Celle-ci s'exprime par la formule :

$$D_M(\mathbf{x}, \omega_i) = (\mathbf{x} - \widehat{\mu_i})^T \widehat{Q} (\mathbf{x} - \widehat{\mu_i}) \tag{15.19}$$

Cette règle de décision revient implicitement à effectuer une transformation linéaire des coordonnées de chaque point et à prendre la classe dont le centre de gravité est alors le plus proche. Cette transformation « allonge » chaque axe proportionnellement à la valeur propre correspondante dans $\widehat{Q}$.

Les surfaces séparatrices sont des hyperplans : on se trouve donc ici à une intersection très simple des méthodes bayésiennes et des méthodes de surfaces séparatrices linéaires décrites au chapitre 9.

La classification bayésienne naïve

On suppose ici que chaque classe possède une matrice de covariance diagonale. Cette hypothèse revient à dire que les attributs sont statistiquement décorrélés. Ceci n'étant en général pas vrai, on introduit là un autre type de biais. Cette hypothèse mène à l'estimation de $2Cd$ paramètres et conduit à des séparatrices quadriques de formes particulières (mais pas des hyperplans). Cette hypothèse est souvent appelée la *méthode bayésienne naïve*.

Dans cette simplification, la probabilité d'observer $\mathbf{x}^\top = (x_1, \ldots, x_d)$ pour un point de n'importe quelle classe ω_i est la probabilité d'observer x_1 pour cette classe, multipliée par celle d'observer x_2 pour cette classe, etc. Donc, par hypothèse :

$$\omega^\star = \underset{i \in \{1,\ldots,C\}}{\text{ArgMax}} \; \mathsf{P}(\omega_i) \prod_{i=1}^{d} \mathsf{p}(x_i \mid \omega_i)$$

Le problème de trouver la classe

$$\omega^* = \underset{i \in \{1,\ldots,C\}}{\text{ArgMax}} \; [\mathsf{P}(\omega_i \mid \mathbf{x})]$$

se ramène donc ici à estimer pour chaque classe la valeur $\mathsf{p}(x_1, \ldots, x_d \mid \omega_i)\,\mathsf{P}(\omega_i)$ à partir des données d'apprentissage.

Copyright © 2010 Eyrolles.

Dans les espaces de grande dimension, la méthode naïve présente un énorme avantage, qu'il est facile d'illustrer sur un exemple. Supposons que les valeurs de chaque attribut soient bornées, disons entre 0 et 1 pour fixer les idées. Une façon d'estimer $\mathbf{p}(x_1, \ldots, x_d \mid \omega_i)$ est de diviser l'espace utile $[0, 1]^d$ en hypercubes et de compter le nombre de points d'apprentissage appartenant à ω_i qui se trouvent dans chacun. Si chaque segment $[0, 1]$ est divisé en k parties égales, l'espace sera composé de k^d éléments de volumes hypercubiques. Si dans l'un d'entre eux on compte n_i points d'apprentissage (pour un total de m_i), la probabilité $\mathbf{p}(\mathrm{x} \mid \omega_i)$ sera estimée par la valeur n_i/m_i pour tout point x de cet élément de volume. Pour obtenir une bonne précision à cette estimation, les valeurs n_i doivent être suffisamment importantes. D'un autre côté, on aura une description de la classe ω_i d'autant plus exacte que k est grand ; mais pour garder une précision d'estimation donnée quand on augmente k, le nombre de points d'apprentissage doit croître exponentiellement avec k. C'est ce que l'on appelle le *fléau de la dimensionnalité* ou encore la *malédiction de la dimensionnalité*.

Dans la méthode naïve, chaque estimation est réalisée indépendamment sur chaque axe. Pour garder une précision donnée quand on augmente k, le nombre de points d'apprentissage n'a besoin que d'augmenter linéairement avec k

Cette méthode, malgré sa simplicité, est donc très employée dans des espaces de grande dimension. Par exemple, en classification de textes, les attributs sont souvent les mots de la langue et leur valeur est le nombre de fois qu'ils sont présents dans un texte. La dimensionnalité est par conséquent de l'ordre de dix mille... ce qui n'empêche pas les méthodes bayésiennes naïves de fournir des résultats de bonne qualité.

Chaque classe a une matrice de covariance proportionelle à la matrice identité I

Cette hypothèse impose de plus une isotropie à l'espace de représentation. Elle permet de n'avoir à estimer que $C(d + 1)$ paramètres. Les surfaces qui discriminent chaque classe d'une autre ne sont des hyperplans que si les variances des deux classes sont égales, mais dans ce cas ils sont de plus parallèles aux axes.

2.5 Les cas non gaussiens et multigaussiens

On a dit plus haut que le seul cas paramétrique que l'on traitait analytiquement était celui de la distribution gaussienne. Ce n'est pas tout à fait exact : on sait en particulier aussi résoudre le problème en modélisant les classes par des distributions uniformes sur des volumes finis, ou par des distributions exponentielles ; mais ces solutions ne possèdent pas beaucoup d'intérêt pratique.

Une autre cas plus intéressant est celui où l'on suppose que les classes possèdent une distribution de probabilités qui est la somme pondérée de M distributions gaussiennes. Il est traité dans la section 4.2 ainsi que dans la section 3.1 du chapitre 18 dans le cadre des méthodes d'apprentissage non supervisé.

2.6 L'analyse discriminante

Les méthodes décrites précédemment s'appuient sur l'estimation des densités de probabilité conditionnelles à chaque classe $\mathbf{p}(\mathrm{x}|\omega_i)$ et sur la mise ne œuvre d'un principe de maximum de vraisemblance. Toujours dans les approches paramétriques, qui font des hypothèses sur la forme des densités de probabilité conditionnelles aux classes, on peut adopter un point de vue différent.

L'idée est de considérer le logarithme du rapport :

$$\log \left\{ \frac{\mathbf{P}(\omega = \omega_k | \mathbf{X} = \mathrm{x})}{\mathbf{P}(\omega = \omega_l | \mathbf{X} = \mathrm{x})} \right\} \tag{15.20}$$

Copyright © 2010 Eyrolles.

pour tous les couples de classes (ω_k, ω_l) et de retourner la classe pour laquelle ce logarithme est toujours positif.

La densité totale sur $\mathcal{X}$ est : $\mathbf{p}(\mathbf{x}) = \sum_{i=1}^{C} \mathbf{P}(\omega_i)\, \mathbf{p}(\mathbf{x}|\omega_i)$.

Pour chaque classe, on a :

$$\mathbf{P}(\omega = \omega_l | \mathbf{X} = \mathbf{x}) = \frac{\mathbf{P}(\omega_l)\, \mathbf{p}(\mathbf{x}|\omega_l)}{\sum_{i=1}^{C} \mathbf{P}(\omega_i)\, \mathbf{p}(\mathbf{x}|\omega_i)}$$

2.6.1 L'analyse discriminante linéaire

Dans le cas où l'on suppose que les distributions de toutes les classes sont des gaussiennes multivariées et égales, c'est-à-dire de même matrice de covariance : $\mathbf{\Sigma}_k = \mathbf{\Sigma}$ $(\forall k)$, le critère de décision (15.20) se simplifie.

En notant : $\mathbf{p}(\mathbf{x}|\omega_k) = \frac{1}{(2\pi)^{d/2}\,|\mathbf{\Sigma}_k|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu_k)^\top \mathbf{\Sigma}_k^{-1}(\mathbf{x} - \mu_k)\}$, on obtient :

$$\begin{aligned} \log\left\{\frac{\mathbf{P}(\omega = \omega_k|\mathbf{X} = \mathbf{x})}{\mathbf{P}(\omega = \omega_l|\mathbf{X} = \mathbf{x})}\right\} &= \log\frac{\mathbf{p}(\mathbf{x}|\omega_k)}{\mathbf{p}(\mathbf{x}|\omega_l)} + \log\frac{\mathbf{P}(\omega_k)}{\mathbf{P}(\omega_l)} \\ &= \log\frac{\mathbf{P}(\omega_k)}{\mathbf{P}(\omega_l)} - \frac{1}{2}(\mu_k - \mu_l)^\top \mathbf{\Sigma}^{-1}(\mu_k - \mu_l) + \mathbf{x}^\top \mathbf{\Sigma}^{-1}(\mu_k - \mu_l) \end{aligned} \tag{15.21}$$

qui est une *équation linéaire* en $\mathbf{x}$! Cela signifie que dans l'espace des entrées $\mathcal{X}$, les frontières de décision entre les classes ω_k sont des hyperplans, à l'instar des pavages de Voronoï.

On a ainsi pour chaque classe une fonction de décision :

$$\boxed{\delta_k(\mathbf{x}) = \mathbf{x}^\top \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2}\mu_k^\top \mathbf{\Sigma}^{-1} \mu_k + \log\big(\mathbf{P}(\omega_k)\big)} \tag{15.22}$$

et la classe décidée est : $\omega^\star = \mathrm{ArgMax}_k\, \delta_k(\mathbf{x})$.

Il faut alors estimer les paramètres $\mathbf{P}(\omega_k)$, μ_k et $\mathbf{\Sigma}_k$ pour toutes les classes k, comme dans les techniques présentées plus haut.

2.6.2 L'analyse discriminante non linéaire

Si maintenant, on suppose que les matrices de covariance ne sont pas égales, alors on obtient des fonctions de décision :

$$\boxed{\delta_k(\mathbf{x}) = -\frac{1}{2}\log|\mathbf{\Sigma}_k| - \frac{1}{2}(\mathbf{x} - \mu_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \mu_k) + \log\big(\mathbf{P}(\omega_k)\big)} \tag{15.23}$$

qui sont des *équations quadratiques* en $\mathbf{x}$.

Dans tous les cas, on rencontre le problème de l'estimation de très nombreux paramètres lorsque les données sont décrites en grande dimension. (Pour plus de détails, voir [HTF09], 106-113).

2.7 La prédiction bayésienne de la distribution des paramètres

Au lieu de chercher à identifier une distribution sous-jacente aux données par estimation d'une fonction paramétrée, on peut résoudre directement le problème de la prédiction de la valeur $\mathbf{y}_i$ correspondant à l'observation $\mathbf{x}_i$. Pour cela, il existe une approche conceptuellement très intéressante et idéalement optimale, même si elle est difficile à mettre en pratique et nécessite de nombreuses approximations.

Copyright © 2010 Eyrolles.

L'idée essentielle est la suivante. Au lieu de chercher la valeur optimale des paramètres[8] θ en maximisant leur vraisemblance sur les données, on décrit ces paramètres comme des distributions de probabilités. Celles-ci sont initialement fixées sous forme d'une distribution *a priori*, puis transformées en distribution *a posteriori* par l'utilisation du théorème de Bayes. Au lieu de chercher une valeur spécifique de θ, on cherche donc ici à trouver la distribution des valeurs s'adaptant le mieux aux données (voir figure 15.5). La prédiction pour l'événement x s'effectue alors en pondérant les valeurs prédites de θ par la probabilité *a posteriori* correspondante.

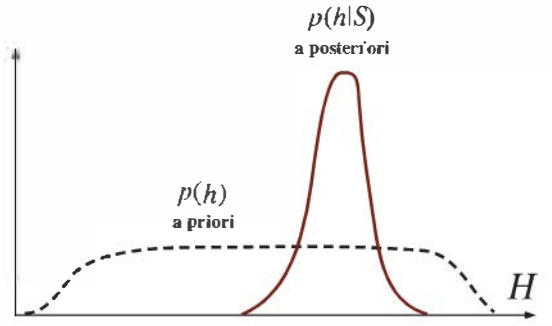


FIG. 15.5: *Illustration schématique de la méthode de prédiction bayésienne. Soit un paramètre θ caractérisant la dépendance fonctionnelle entre les entrées et les sorties. La distribution* a priori *représente notre connaissance initiale sur la distribution possible de θ qui est typiquement assez lâche. Une fois que les données d'apprentissage ont été prises en compte, la distribution* a posteriori *calculée par le théorème de Bayes est généralement beaucoup plus focalisée autour d'une ou plusieurs valeurs spécifiques qui sont les plus cohérentes avec les données. Il est alors possible d'utiliser cette distribution pour calculer une prédiction pour une observation nouvelle $\mathbf{x}_n$.*

Reprenons les notations précédentes, en remarquant que cette fois θ est un vecteur[9] aléatoire de densité de probabilité $\mathbf{p}(\theta)$ connue.

On cherche la densité du vecteur x étant donné l'échantillon $\mathcal{S}$:

$$\mathbf{p}(\mathbf{x}|\mathcal{S}) = \int \mathbf{p}(\mathbf{x}, \theta|\mathcal{S})\, d(\theta)$$

La formule de Bayes permet d'écrire :

$$\mathbf{p}(\mathbf{x}, \theta|\mathcal{S}) = \mathbf{p}(\mathbf{x}|\theta, \mathcal{S})\, \mathbf{p}(\theta|\mathcal{S})$$

Le premier facteur est indépendant de $\mathcal{S}$ puisque nous supposons que la valeur de la densité de x est entièrement fixée par la valeur du vecteur des paramètres θ.

Nous avons donc :

$$\boxed{\mathbf{p}(\mathbf{x}|\mathcal{S}) = \int \mathbf{p}(\mathbf{x}|\theta)\, \mathbf{p}(\theta|\mathcal{S})\, d\theta} \qquad (15.24)$$

Ainsi, au lieu de chercher une valeur spécifique de θ, la méthode de prédiction bayésienne calcule une moyenne pondérée sur toutes les valeurs de θ.

[8] Dans le cas de l'hypothèse d'une distribution gaussienne, ces paramètres sont la moyenne μ et la matrice de covariance Q.

[9] L'ensemble des paramètres est regroupé sous la forme d'un vecteur.

Copyright © 2010 Eyrolles.

Le facteur de pondération $\mathbf{p}(\theta|\mathcal{S})$, qui est la distribution *a posteriori* de θ, est déterminé en partant d'une distribution choisie *a priori* $\mathbf{p}(\theta)$ qui est ensuite mise à jour par utilisation de la règle de Bayes sur l'échantillon d'apprentissage $\mathcal{S}$. Comme les exemples de cette séquence sont supposés résulter d'un tirage aléatoire suivant une certaine distribution sous-jacente (tirage i.i.d.), on peut écrire :

$$\mathbf{p}(\mathcal{S}|\theta) = \prod_{i=1}^{m} \mathbf{p}(\mathbf{x}_i|\theta)$$

en utilisant à nouveau le théorème de Bayes :

$$\mathbf{p}(\theta|\mathcal{S}) = \frac{\mathbf{p}(\mathcal{S}|\theta)\,\mathbf{p}(\theta)}{\mathbf{p}(\mathcal{S})} = \frac{\mathbf{p}(\theta)}{\mathbf{p}(\mathcal{S})} \prod_{i=1}^{m} \mathbf{p}(\mathbf{x}_i|\theta) \tag{15.25}$$

où le facteur de normalisation $\mathbf{p}(\mathcal{S})$ est choisi pour que $\int \mathbf{p}(\theta|\mathcal{S})d\theta = 1$.

L'évaluation d'une intégrale comme celle de l'équation 15.24 n'est possible de manière analytique que pour une classe de fonctions de densité pour lesquelles la densité *a posteriori* $\mathbf{p}(\theta|\mathcal{S})$ a la même forme que la densité *a priori* $\mathbf{p}(\theta)$. Dans ce cas particulier, on parle de *densités autoreproductibles* [DH73]. L'exemple le plus commun de telles densités est celui de la distribution normale (gaussienne).

Illustration avec une loi normale unidimensionnelle

Supposons que les observations $\mathbf{x}$ soient décrites par une mesure unidimensionnelle qui suit une loi normale de moyenne inconnue μ et de variance σ connue. Le paramètre θ est donc ici simplement égal à μ

L'approche de la prédiction bayésienne nous dicte de chercher la densité de probabilité de la variable μ aléatoire en fonction des données d'apprentissage $\mathcal{S}$. Nous supposons que le paramètre μ suit également une loi normale de moyenne μ_0 et de variance σ_0. Pour exprimer notre ignorance *a priori* sur la valeur de μ, nous prenons une grande valeur pour la variance σ_0.

$$\mathbf{p}_0(\mu) = \frac{1}{(2\pi\sigma_0^2)^{\frac{1}{2}}} \exp\left\{-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right\}$$

La donnée d'une séquence d'apprentissage $\mathcal{S}$ permet de réviser cette densité de probabilité en utilisant le théorème de Bayes suivant l'équation (15.25) :

$$\mathbf{p}(\mu|\mathcal{S}) = \frac{\mathbf{p}_0(\mu)}{\mathbf{p}(\mathcal{S})} \prod_{i=1}^{m} \mathbf{p}(\mathbf{x}_i|\mu)$$

En utilisant le fait que :

$$\mathbf{p}(\mathbf{x}|\mu) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{(\mathbf{x}-\mu)^2}{2\sigma^2}\right\}$$

il est facile de montrer que la distribution *a posteriori* $\mathbf{p}(\mu|\mathcal{S})$ est également normale avec :

$$\begin{cases} \mu = \dfrac{m\sigma_0^2}{m\sigma_0^2+\sigma^2}\,\bar{\mathbf{x}} + \dfrac{\sigma^2}{m\sigma_0^2+\sigma^2}\,\mu_0 \\ \dfrac{1}{\sigma^2} = \dfrac{m}{\sigma^2} + \dfrac{1}{\sigma_0^2} \end{cases}$$

Copyright © 2010 Eyrolles.

où $\bar{\mathbf{x}}$ est la moyenne : $\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$.

Ces équations montrent qu'au fur et à mesure que le nombre m de données augmente, la moyenne μ de la distribution *a posteriori* approche la moyenne de l'échantillon d'apprentissage $\bar{\mathbf{x}}$. De même, l'écart type σ décroît vers zéro.

Ainsi l'approche de la prédiction bayésienne calcule une moyenne pondérée sur toutes les valeurs de θ au lieu de choisir une valeur spécifique. Cependant si la densité *a posteriori* $\mathbf{p}(\theta|\mathcal{S})$ présente un pic étroit centré sur une valeur $\hat{\theta}$, alors $\mathbf{p}(\theta|\mathcal{S}) \approx \mathbf{p}(h|\hat{\theta})$, et nous retrouvons le résultat donné par la méthode du maximum de vraisemblance. Cela arrive généralement pour les échantillons d'apprentissage de grande taille.

Bien que cela ne soit pas le sujet de ce chapitre, il est utile de noter dès à présent que le principe du maximum de vraisemblance et l'apprentissage bayésien ne se prêtent pas aux mêmes méthodes de calcul. Le premier se traite comme un problème d'optimisation : il faut chercher le minimum d'une fonction d'erreur. En revanche, dans le second, l'essentiel du calcul implique une intégration sur des espaces de grandes dimensions. Dans ce dernier cas, les méthodes classiques d'intégration ne conviennent pas et il faut se tourner vers des méthodes approchées, par exemple les méthodes de Monte-Carlo (voir le chapitre 3).

3. Les méthodes non paramétriques

Très généralement, on appelle *méthodes non paramétriques* les méthodes permettant à la complexité de la solution (l'hypothèse ou la fonction de décision construite) d'augmenter avec le nombre de données. Ces méthodes incluent les méthodes de plus proches voisins, les méthodes à noyaux (chapitre 14) et les réseaux connexionnistes dans la mesure où leur architecture peut évoluer automatiquement durant l'apprentissage.

3.1 Généralités : le problème de l'estimation locale d'une densité

Les méthodes non paramétriques traitent de l'estimation d'une densité de probabilités pour laquelle aucune régularité fonctionnelle n'est supposée *a priori*. Ces méthodes reposent cependant sur l'hypothèse fondamentale que les distributions ou fonctions recherchées sont localement régulières.

Soit une densité de probabilité inconnue $\mathbf{p}(\mathbf{x})$. La probabilité Q pour qu'une forme $\mathbf{x}$ issue de cette distribution soit observée dans la région $\mathcal{R} \in \mathcal{X}$ est :

$$Q = \int_{\mathcal{R}} \mathbf{p}(u)\,\mathrm{d}u$$

L'annexe 3 explique comment on obtient une bonne estimation de Q à partir de la moyenne des points observés dans la région $\mathcal{R}$:

$$Q \approx k/m. \tag{15.26}$$

Par ailleurs, en faisant l'hypothèse que la densité cherchée $\mathbf{p}(\mathbf{x})$ est continue et ne varie pas significativement dans la région $\mathcal{R}$, on peut faire l'approximation :

$$Q = \int_{\mathcal{R}} \mathbf{p}(u)\,\mathrm{d}u \approx \mathbf{p}(\mathbf{x})\,V \tag{15.27}$$

où V est le volume de la région $\mathcal{R}$.

Copyright © 2010 Eyrolles.

De (15.26) et (15.27) on déduit :

$$\mathbf{p}(\mathbf{x}) \approx \frac{k/m}{V} \tag{15.28}$$

Application à l'apprentissage d'une règle de classification

Dans le cas où l'on cherche à apprendre une règle de classification, la méthode bayésienne consiste à estimer en un point $\mathbf{x}$ donné la densité de probabilité de chaque classe afin de choisir celle qui possède la valeur la plus grande.

Nous omettons dans ce qui suit l'indice correspondant au numéro de classe, puisque le problème est le même pour chaque classe. Comme ce n'est qu'après l'estimation séparée pour chaque classe qu'on les compare, on peut agir comme s'il n'y avait qu'une seule densité de probabilité à estimer.

Néanmoins nous allons indexer les termes par m, la taille de l'échantillon : on verra que cette précision est nécessaire quand on étudie les propriétés de convergence.

On suppose donc être en possession de m points de $\mathbb{R}^d$ obtenus par tirages indépendants selon une densité qui caractérise la classe ω. Comment estimer $\mathbf{p}(\mathbf{x} \mid \omega)$ au point $\mathbf{x}$ à partir d'un ensemble d'apprentissage ? Le principe vient d'être expliqué : on définit autour de $\mathbf{x}$ une certaine région $\mathcal{R}_m$ (en pratique, une hypersphère ou un hypercube) et on compte le nombre k_m de points de l'échantillon d'apprentissage qui sont inclus dans ce volume (figure 15.6).

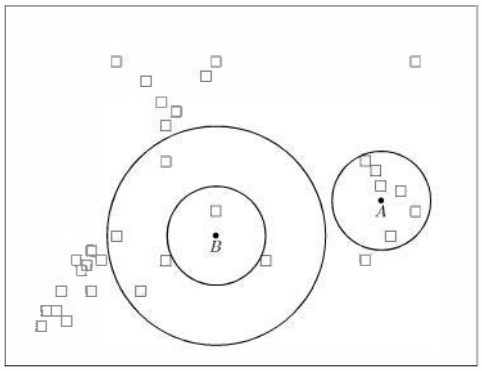


FIG. 15.6: *Les points □ sont des tirages indépendants selon une certaine distribution dans le plan* $\mathbb{R}^2$*, dont la densité est plus forte au point A qu'au point B. En effet, pour le même volume autour du point A et du point B,* k_m *vaut respectivement* 6 *et* 1*. Pour obtenir* $k_m = 6$ *autour du point B, il faut augmenter le volume.*

On a vu que l'estimateur de $\mathbf{p}(\mathbf{x} \mid \omega)$ pour un échantillon de taille m se définit par :

$$\widehat{\mathbf{p}_m}(\mathbf{x} \mid \omega) = \frac{k_m/m}{V_m} \tag{15.29}$$

où V_m est le volume de la région $\mathcal{R}_m$ considérée.

On peut démontrer (voir annexe 3) que, quand m augmente, cet estimateur converge vers la valeur cherchée $\mathbf{p}(\mathbf{x} \mid \omega)$, quand les conditions suivantes sont remplies :

$$\lim_{m \to \infty} V_m = 0$$

$$\lim_{m \to \infty} k_m = \infty$$

$$\lim_{m \to \infty} (k_m/m) = 0$$

Copyright © 2010 Eyrolles.

Il y a en pratique deux solutions pour remplir ces conditions :

- Soit définir V_m à partir d'une région $\mathcal{R}_0$ de forme et de volume V_0 fixés : par exemple un hypercube de côté unité, mais on verra que c'est loin d'être le seul cas possible. On prend alors :

$$V_m = V_0/f(m)$$

où f est une fonction croissante de m. Ceci conduit aux méthodes des fonctions noyau, en particulier aux *fenêtres de Parzen.*

- Soit fixer le nombre k_m, se donner une famille de volumes paramétrée par une variable (par exemple les hypersphères centrées en x, de rayon variable) et ajuster cette variable pour que le volume contienne exactement k_m points de l'ensemble d'apprentissage. Cette technique d'estimation est connue sous le nom de méthode des *k-plus proches voisins.* Utilisée dans le problème de la classification, elle se traduit par un algorithme qui ne nécessite pas l'estimation explicite de la densité de chaque classe au point à classer, mais en permet plus simplement un choix direct.

3.2 Les fonctions noyau et les fenêtres de Parzen

Rappelons quelques notions introduites dans le chapitre 14. Ici, cependant, les fonctions noyau seront définies sur $\mathcal{X}$ au lieu de l'être sur $\mathcal{X} \times \mathcal{X}$, mais le sens général de mesure d'une similarité persiste. Une fonction noyau (*kernel*) κ est donc ici une fonction bornée sur $\mathcal{X}$ d'intégrale égale à 1. On suppose en général que κ présente un pic centré en 0. Par conséquent, $\kappa(\mathbf{x}_i - \mathbf{x}_j)$ détermine une mesure de proximité entre les points $\mathbf{x}_i$ et $\mathbf{x}_j$. On impose aussi en général que cette fonction soit symétrique : $\kappa(-\mathbf{x}) = -\kappa(\mathbf{x})$.

Dans cette perspective, l'estimation locale de la densité $\mathbf{p}(\mathbf{x})$ se calcule comme une somme pondérée des exemples $\mathbf{x}_j$ pondérée par leur distance à $\mathbf{x}$:

$$\hat{p}(\mathbf{x}) = \frac{1}{m} \sum_{j=1}^{m} \kappa(\mathbf{x} - \mathbf{x}_j) \tag{15.30}$$

ce qui peut aussi être interprété comme une moyenne des fonctions noyau centrées sur chaque exemple.

Pour une classe ω_k donnée, on a :

$$\hat{\mathbf{P}}(\omega_k|\mathbf{x}) = \frac{\hat{\mathbf{P}}(\omega_k)\,\hat{p}(\mathbf{x}|\omega_k)}{\sum_{i=1}^{C} \hat{\mathbf{P}}(\omega_i)\,\hat{p}(\mathbf{x}|\omega_i)} = \frac{\frac{\hat{\mathbf{P}}(\omega_k)}{m_k} \sum_{\mathbf{x}_i \in \omega_k} \kappa(\mathbf{x} - \mathbf{x}_i)}{\sum_{i=1}^{C} \frac{\hat{\mathbf{P}}(\omega_i)}{m_i} \kappa(\mathbf{x} - \mathbf{x}_i)}$$

Si les probabilités des classes sont estimées par le rapport m_k/m où m_k est le nombre d'exemples de la classe ω_k et m le nombre total d'exemples, on en déduit :

$$\hat{\mathbf{P}}(\omega_k|\mathbf{x}) = \frac{\sum_{\mathbf{x}_i \in \omega_k} \kappa(\mathbf{x} - \mathbf{x}_i)}{\sum_{i=1}^{m} \kappa(\mathbf{x} - \mathbf{x}_i)} \tag{15.31}$$

ce qui revient à prendre la proportion pondérée d'exemples autour de $\mathbf{x}$ de la classe ω_k.

Une difficulté de ces méthodes est le choix de la fonction κ. Une autre difficulté provient de leur médiocre capacité à être utilisées dans des espaces de représentation $\mathcal{X}$ de grande dimension. En effet, les estimations sont fondées sur la détermination d'un volume dans l'espace des données. Or, dans les espaces de grande dimension, un volume qui couvre suffisamment de données n'est plus valide pour une estimation locale, car son rayon tend à devenir grand par rapport à l'intervalle des valeurs possibles pour les données.

Copyright © 2010 Eyrolles.

Le paragraphe suivant explore ces méthodes d'estimation par voisinage dans $\mathcal{X}$. Les cas des fonctions noyau définies comme des hypercubes ou des distributions gaussiennes y sont en particulier traités. La méthode qui en résulte s'appelle la méthode des fenêtres de Parzen.

3.2.1 Les fenêtres de Parzen : le cas élémentaire

Commençons en définissant la région élémentaire (le noyau) $\mathcal{R}_m$ comme un hypercube de coté h_m centré en $\mathbf{x}$. On a dans ce cas :

$$V_m = (h_m)^d$$

$$\widehat{\mathbf{p}_m}(\mathbf{x}) = \frac{1}{mV_m} \sum_{i=1}^{i=m} \frac{\phi(\mathbf{x} - \mathbf{x}_i)}{h_m}$$

où ϕ est la fonction caractéristique de l'hypercube unité :

$$\begin{cases} \phi(\mathbf{x}_i) = 1 & \text{si } \mathrm{x}_i \in [-1/2, 1/2] \\ \phi(\mathbf{x}_i) = 0 & \text{sinon} \end{cases}$$

Les conditions de convergence citées plus haut au paragraphe 3.1 sont remplies par exemple en prenant, pour une valeur h_0 fixée :

$$h_m = \frac{h_0}{m}$$

La formule 15.29 définit par conséquent l'estimateur de $\mathbf{p}(\mathrm{x} \mid \omega)$. k_m est le nombre de points d'apprentissage de la classe ω inclus dans l'hypercube centré en $\mathbf{x}$ de côté h_m. En pratique, pour estimer $\mathbf{p}_m(\mathrm{x} \mid \omega)$, il ne reste qu'à fixer la valeur h_0 : on en déduit la valeur de h_m, puis celle de $\widehat{\mathbf{p}_m}(\mathrm{x} \mid \omega)$ par comparaison des coordonnées des points de l'ensemble d'apprentissage avec celles de $\mathbf{x}$.

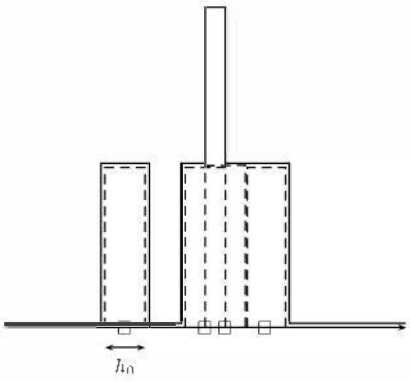


FIG. 15.7: *Estimation de densité par la méthode des fenêtres de Parzen. Il y a quatre points d'apprentissage, dans un espace à une dimension. Les hypercubes sont des segments de largeur h_0 centrés sur les points d'apprentissage. La densité (en trait plein) est calculée comme la somme des fenêtres centrées sur chaque point. Ici, cette fenêtre est étroite (h_0 est petit) : la densité résultante est peu lisse. La surface sous la courbe en trait plein est égale à 1.*

Le choix de h_0 influe sur le résultat de la façon suivante : si cette valeur est choisie petite, la probabilité estimée que le point $\mathbf{x}$ ait été engendré par le processus ω est nulle partout, sauf

Copyright © 2010 Eyrolles.

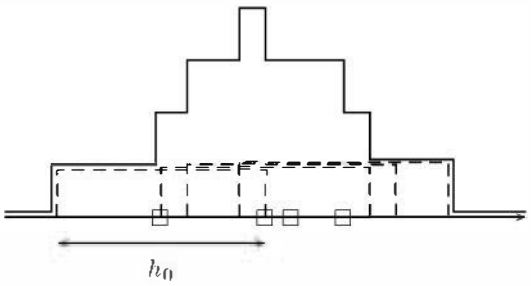


FIG. 15.8: *La même estimation pour h_0 plus grand : la densité est estimée de manière plus lisse.*

au voisinage immédiat des points de l'ensemble d'apprentissage ; on a donc dans ce cas modélisé $\mathbf{p}(\omega)$ comme un « peigne ». Si elle est choisie grande, $\mathbf{p}(\omega)$ est en revanche modélisée de manière « lisse » (figures 15.7 et 15.8).

3.2.2 Généralisation à des fonctions noyau gaussiennes

La technique précédente a l'avantage de se réduire à un algorithme simple, mais présente l'inconvénient de la sensibilité du choix de la valeur h_0. D'autre part, dès que l'on s'éloigne des points d'apprentissage, la densité est estimée comme nulle. On peut remédier à ce problème en s'arrangeant pour que l'appartenance au volume élémentaire V_m autour du point $\mathbf{x}$ ne soit plus une fonction caractéristique (à valeur binaire) d'appartenance à un volume, mais une probabilité.

C'est ce qui a été présenté ci-dessus en introduction : la densité est estimée comme la somme de noyaux, qui sont donc des densités élémentaires centrées sur les points d'apprentissage. Souvent, ces noyaux sont des distributions gaussiennes (voir la figure 15.9). Le résultat est plus ou moins lisse selon la valeur de la variance, mais il n'y a plus de point de l'espace où la densité soit estimée comme nulle.

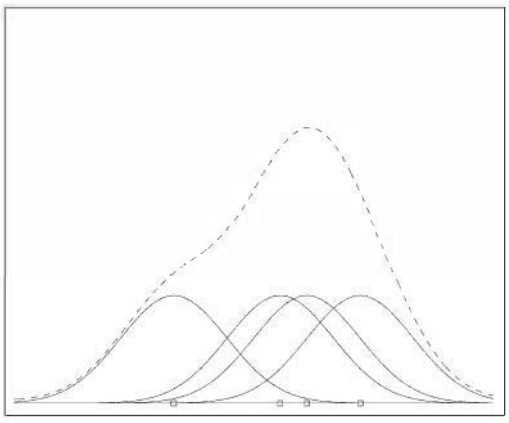


FIG. 15.9: *Fenêtres de Parzen : estimation avec un noyau gaussien.*

Sans entrer dans les détails, on peut voir intuitivement cette généralisation de la façon suivante : pour estimer la valeur $\mathbf{p}_m(\mathbf{x} \mid \omega)$, on additionne sur les m points de l'ensemble d'apprentissage des valeurs entre 0 et 1 qui sont fonction de la distance entre $\mathbf{x}$ et le point courant, avant de diviser cette somme par un coefficient normalisateur. Cette fonction n'est donc plus binaire comme dans

Copyright © 2010 Eyrolles.

le cas précédent, mais calculée de façon continue. On doit évidemment la choisir (ainsi que la normalisation finale) de façon que $\mathbf{p}_m(\mathbf{x} \mid \omega)$ soit effectivement une estimation de densité de probabilité. La figure 15.9 montre l'estimation réalisée à partir des mêmes points que dans les figures 15.8 et 15.7 pour des noyaux gaussiens.

3.2.3 Astuces de calcul

Reprenons l'équation 3.2.3 qui donne l'estimation de la probabilité de la classe ω_p au point $\mathbf{x}$:

$$\hat{\mathbf{P}}(\omega_k|\mathbf{x}) = \frac{\sum_{\mathbf{x}_i \in \omega_k} \kappa(\mathbf{x} - \mathbf{x}_i)}{\sum_{i=1}^{m} \kappa(\mathbf{x} - \mathbf{x}_i)}$$

En décision bayésienne, il faut calculer $\hat{\mathbf{P}}(\omega_k|\mathbf{x})$ pour chaque classe et choisir d'attribuer au point $\mathbf{x}$ la classe pour laquelle cette probabilité conditionnelle est la plus forte.

Le dénominateur n'a pas besoin d'être calculé, puisqu'il est identique pour toutes les classes. En revanche, il semble qu'il faille, à chaque décision, calculer le numérateur pour chaque classe, ce qui oblige au total à m estimations de la fonction noyau au point $\mathbf{x}$.

On peut éviter tous ces calculs en utilisant une fonction noyau κ cette fois définie sur $\mathcal{X} \times /calX$ et possédant une propriété supplémentaire : il faut qu'il existe une fonction Φ de $\mathbb{R}^d$ dans un espace vectoriel F_n de dimension n telle que :

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$$

où $\langle ., . \rangle$ désigne le produit scalaire dans F_n.

Pour chaque classe ω_k dont on possède m_k points d'apprentissage, le calcul du numérateur s'effectue alors ainsi, pour un noyau κ :

$$\begin{aligned}
\hat{\mathbf{P}}(\omega_k|\mathbf{x}) &= \frac{1}{m_k} \sum_{\mathbf{x}_i \in \omega_k} \kappa(\mathbf{x}, \mathbf{x}_i) \\
&= \frac{1}{m_k} \sum_{\mathbf{x}_i \in \omega_k} \langle \Phi(\mathbf{x}) \, , \, \Phi(\mathbf{x}_i) \rangle \\
&= \frac{1}{m_k} \langle \Phi(\mathbf{x}) \, , \, \sum_{\mathbf{x}_i \in \omega_k} \Phi(\mathbf{x}_i) \rangle
\end{aligned}$$

On peut précalculer une fois pour toutes $\sum_{\mathbf{x}_i \in \omega_k} \Phi(\mathbf{x}_i)$ à partir de l'ensemble d'apprentissage. La décision de classification d'un point $\mathbf{x}$ nécessite désormais seulement C produits scalaires en dimension n : c'est un calcul en temps constant, indépendant de la taille m de l'ensemble d'apprentissage.

Par exemple, pour $d = 2$, on peut définir le noyau κ_3 (*polynomial d'ordre 3*) en posant par définition[10] (voir aussi le chapitre 14) :

$$\kappa_3(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$$

On peut alors vérifier que la définition suivante de Φ_3 :

$$\Phi_3(\mathbf{x}) = (x_1^3 \, , \, \sqrt{3}\, x_1^2 x_2 \, , \, \sqrt{3}\, x_1 x_2^2 \, , \, x_2^3)^\top$$

vérifie bien :

$$\mathbf{x} \cdot \mathbf{y} = \langle \Phi_3(\mathbf{x}) \, , \, \Phi_3(\mathbf{y}) \rangle$$

[10] Le produit scalaire dans $\mathbb{R}^2$ est noté classiquement $\mathbf{x} \cdot \mathbf{y}$.

Copyright © 2010 Eyrolles.

Utiliser le noyau κ_3 revient à transformer les données de manière non linéaire en les projetant de la dimension $d = 2$ à la dimension $n = 4$ par la fonction Φ_3. Il est assez facile de généraliser cette propriété : au noyau polynomial d'ordre p est associée une fonction Φ_p qui transforme les données de la dimension d à la dimension $n = \binom{d+p-1}{p}$. La valeur n augmente rapidement : par exemple, pour un noyau polynomial d'ordre $p = 3$ et un espace de représentation initial de dimension $d = 6$, on projette dans un espace de dimension $n = 168$.

Si on veut utiliser une fonction noyau gaussienne, ce calcul ne peut plus se faire exactement. En effet, la fonction Φ correspondante est de dimension infinie (intuitivement, parce que $e^x = \sum_{i=1}^{\infty} \frac{1}{i!} x^i$). On peut cependant l'approcher par une somme pondérée de noyaux polynomiaux, d'aussi près que l'on veut en augmentant p, le plus grand de leurs ordres (et donc n).

3.3 Les k-plus proches voisins (k-ppv)

L'un des problèmes avec les méthodes par fonctions noyau provient de ce que leur taille est fixe. Si celle-ci est trop grande, l'approximation peut être trop « lissée » par rapport à la réalité. Si elle trop petite, l'estimation dans des régions de faible densité peut être nulle ou très approximative. Il faudrait donc que cette taille soit fonction de la position dans l'espace $\mathcal{X}$. C'est ce que réalise la méthode par plus proches voisins.

Dans celle-ci, le nombre k de points dans la région autour de $\mathbf{x}$ est fixé et on fait au contraire varier le volume V. On considère donc une hypersphère (en général on utilise une distance euclidienne) centrée en $\mathbf{x}$ et on modifie son rayon jusqu'à ce qu'elle contienne k points. L'estimation de la densité est alors donnée par le rapport k/mV où m est le nombre total de points dans l'échantillon de données.

D'un certain côté, cela revient à choisir une fonction noyau simple, constante sur l'hypersphère contenant les k points et nulle ailleurs. Mais cela permet de passer directement à une règle de décision : on classe une forme inconnue $\mathbf{x}$ en prenant la classe qui est majoritaire dans les k points d'apprentissage les plus proches. Cette règle est appelée règle des k-plus proches voisins (« *k-nearest-neighbour classification rule* ») où k est le nombre de voisins considérés.

Dans le cas où $k = 1$, la règle s'appelle la *règle de classification du plus proche voisin*. Elle assigne à $\mathbf{x}$ simplement la même étiquette que le point d'apprentissage le plus proche. Dans ce cas, les frontières de décision dans l'espace $\mathcal{X}$ prennent la forme d'un pavage convexe (voir la figure 15.12). Il est remarquable que cette règle extrêmement simple possède un comportement asymptotique excellent vis-à-vis du risque minimal de Bayes, comme on le verra au paragraphe 3.3.2.

Il est conseillé de se reporter à [DH73] et [DK82] pour une étude approfondie des méthodes de plus proches voisins. Le lecteur francophone lira avec profit [CL96].

3.3.1 Le principe

La règle de décision par k-ppv est facile à illustrer, comme sur la figure 15.10.

On y a représenté un problème à deux classes : les points à classer sont notés • et les points alentour sont les données d'apprentissage, appartenant soit à la classe notée □, soit à celle notée ◆. On cherche, au sens de la métrique choisie pour le problème (sur ce dessin, euclidienne), les k-plus proches voisins des points $\mathbf{x}$; pour $k = 1$, dans les deux cas, c'est un des points notés □. On affecte donc aux deux points • la classe □. Pour $k = 3$, le voisinage du premier point • compte deux points ◇ et un point □ : c'est la classe ◆ qui est majoritaire, et ce point est classé comme appartenant à la classe ◇. Pour l'autre point, la décision pour $k = 3$ confirme l'appartenence à la classe □.

Copyright © 2010 Eyrolles.

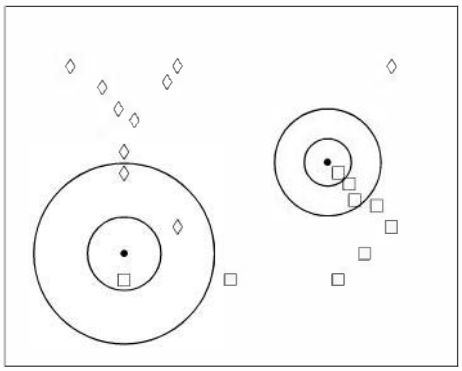


FIG. 15.10: *Décision par 1-ppv et 3-ppv dans un ensemble d'exemples appartenant à deux classes.*

La figure 15.11 représente la même opération pour un problème à trois classes. Pour $k = 1$, les points • sont classés comme □ ; pour $k = 3$, la règle de décision produit une ambiguïté pour le premier point : on ne peut pas se décider entre les trois classes.

Le déroulement de la méthode est donné dans l'algorithme 23.

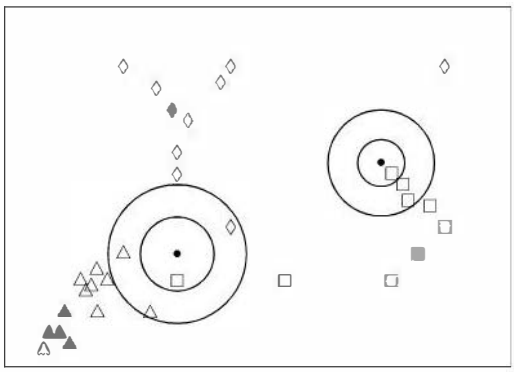


FIG. 15.11: *Décision par 1-ppv et 3-ppv dans un ensemble d'exemples appartenant à trois classes.*

3.3.2 La validité bayésienne

Quelle est la validité de cette règle en apparence naïve ? Elle est conforme aux règles de l'estimation bayésienne définies au paragraphe 3.1, sous l'hypothèse que les probabilités *a priori* des classes sont bien estimées par leur proportion d'échantillons d'apprentissage.

La règle des k-ppv fait implicitement une estimation comparative de toutes les densités de probabilités des classes apparaissant dans le voisinage de x et choisit simplement la plus probable : elle approxime donc la décision bayésienne.

Pour s'en convaincre, il suffit de supposer que les m points de l'ensemble d'apprentissage comportent m_i points de la classe ω_i et que sur les k-plus proches voisins de x, il y a k_{m_i} points de cette classe. On a, d'après l'équation 15.29 :

$$\widehat{\mathbf{p}_m}(\mathbf{x} \mid \omega_i) = \frac{k_{m_i}/m_i}{V_m}$$

Copyright © 2010 Eyrolles.

Algorithme 23 : Algorithme des k-plus proches voisins

début
On cherche à classer le point x
pour chaque *exemple* (y, ω) *de l'ensemble d'apprentissage* **faire**
calculer la distance $\mathcal{D}(y, x)$ entre y et x
fin
Dans les k points les plus proches de x
compter le nombre d'occurences de chaque classe
Attribuer à x la classe qui apparaît le plus souvent
fin

On formule maintenant l'hypothèse que m_i/m est un estimateur de $P(\omega_i)$, la probabilité *a priori* de la classe de rang i. On peut donc noter : $m_i/m = \widehat{P}_m(\omega_i)$.

On en déduit :

$$k_{m_i} = m\, V_m\, \widehat{p_m}(x \mid \omega_i)\, \widehat{P}_m(\omega_i)$$

Par conséquent, la classe qui a le plus de points d'apprentissage dans les k_m (celle pour laquelle la valeur k_{m_i} est maximale) est aussi celle qui maximise la valeur $p_m(x \mid \omega_i)\, \widehat{P}_m(\omega_i)$ qui est égale, par la règle de Bayes, à : $\widehat{P}_m(\omega_i \mid x)\, p(x)$. Cette classe est donc celle qui maximise la valeur $\widehat{P}_m(\omega_i \mid x)$. Son choix approxime par conséquent la règle de classification bayésienne.

Rappellons que tout ce calcul ne vaut que si m_i/m est un estimateur de $P(\omega_i)$. Il faut donc n'appliquer la règle des k-ppv qu'après s'être assuré de la validité de cette hypothèse.

3.3.3 Quelques propriétés de convergence

Il est assez facile de démontrer que la probabilité d'erreur R_{k-ppv} de la règle des k-ppv converge vers le risque bayésien R_B quand m, le nombre total d'échantillons, croît vers l'infini, et ceci pour tout k. Cette propriété est démontrée en annexe 5 pour $k = 1$.

On a de plus les propriétés suivantes, dans le cas de deux classes, toujours à la limite sur m :

$$R_B \leq R_{k-ppv} \leq R_{(k-1)-ppv} \cdots \leq R_{1-ppv} \leq 2R_B \tag{15.32}$$

avec :

$$R_{k-ppv} \leq R_B + R_{1-ppv}\sqrt{\frac{2}{\pi k}} \tag{15.33}$$

et pour un nombre quelconque C de classes :

$$R_{1-ppv} \leq R_B\left(2 - \frac{C}{C-1}R_B\right) \tag{15.34}$$

Ces formules valident donc l'intuition que l'augmentation de k améliore l'estimation réalisée ; en même temps, elles prouvent que la règle simple du plus proche voisin (1-ppv) est asymptotiquement efficace. On résume souvent plaisamment la formule 15.32 par l'expression : « la moitié de l'information sur la classification optimale d'un point inconnu est disponible dans son seul plus proche voisin ».

Mais les formules ci-dessus ne sont valables que pour m assez grand, ce qui est une remarque pratique importante. Pour plus de détails, on se reportera avec profit à [Rip96] (pp.192-197).

Copyright © 2010 Eyrolles.

3.3.4 Considérations pratiques

Bien sûr, dans la problématique de l'apprentissage, le nombre m est fini ; il faut alors trouver un compromis entre une valeur faible de k, qui semble moins favorable selon les formules ci-dessus, et une valeur exagérément grande[11]. Diverses considérations théoriques et expérimentales mènent à l'heuristique suivante : choisir k autour de $\sqrt{m/C}$ où m/C est le nombre moyen de points d'apprentissage par classe. On remarquera que d, la dimension de l'espace de représentation, n'apparaît pas dans cette formule.

Un autre problème pratique est : quelle décision prendre en cas d'égalité? On peut augmenter k de 1 pour trancher le dilemme, mais s'il y a plus de deux classes, l'ambiguïté peut subsister. Une autre solution consiste à tirer au hasard la classe à attribuer au point ambigu ; son analyse montre qu'elle n'est pas mauvaise.

Enfin, un grand nombre d'auteurs ont proposé des variantes de la règle du k-ppv ; par exemple, au lieu de compter simplement les points de chaque classe parmi les k (ce que l'on peut traduire par les faire voter avec une voix chacun), on a pensé pondérer ces votes par la distance au point x, qui est de toute façon calculée. On est dans ce cas dans des méthodes intermédiaires entre les k-plus proches voisins et les fenêtres de Parzen.

3.3.5 Les surfaces séparatrices de la règle de décision k-ppv

Il est courant d'appeller *zone de Voronoï* d'un exemple le lieu des points de $\mathbb{R}^d$ qui sont plus proches de cet exemple que de tout autre exemple.

Des considérations géométriques permettent de prouver que la zone de Voronoï d'un exemple est l'intersection de $m-1$ demi-espaces, définis par les hyperplans médiateurs entre cet exemple et tous les autres. La zone de Voronoï d'un exemple est donc un volume convexe (pour $d=2$, c'est un polygone convexe) et la frontière entre deux zones de Voronoï est un « polygone » en dimension $d-1$.

Pour $k=1$, la surface séparatrice entre deux classes est la surface séparatrice entre les deux volumes obtenus par l'union des surfaces de Voronoï des exemples de chaque classe (voir la figure 15.12). On peut montrer que pour $k>1$, les séparatrices sont encore des hyperplans par morceaux.

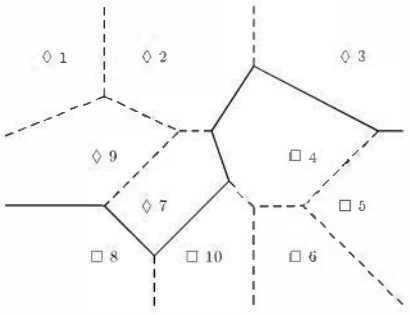


FIG. 15.12: *Un ensemble de points appartenant à deux classes et leurs zones de Voronoï. La séparatrice entre les deux classes par la règle de décision 1-ppv est en trait plein.*

[11] Prendre $k=m$ mène à au résultat suivant : un point sera toujours classé comme appartenant à la classe la plus nombreuse dans l'ensemble d'apprentissage.

Copyright © 2010 Eyrolles.

3.3.6 Algorithmique avancée pour les k-ppv

L'algorithme de base de la décision par k-ppv consiste à calculer les m distances du point x à classer aux m points d'apprentissage, et à identifier les k points les plus proches pour choisir la classe majoritaire dans ces k points. C'est à chaque fois un calcul en $\mathcal{O}(m \times d)$ qu'il faut effectuer. Ceci est à comparer à la quantité de calculs que requiert une décision quand l'ensemble d'apprentissage a été « compilé » par apprentissage paramétrique, par exemple sous la forme de distributions gaussiennes explicites : au plus en $\mathcal{O}(C\,d^2)$. En général, la comparaison n'est pas à l'avantage de la règle des k-ppv, en tout cas dès que l'on dispose d'un ensemble d'apprentissage un peu conséquent, comme il faut le souhaiter. C'est pourquoi une algorithmique particulière a été développée, visant soit à réduire l'ensemble d'apprentissage (méthodes de *nettoyage* et de *condensation*) sans changer le résultat des futures décisions par k-ppv, soit à l'organiser sous des structures de données permettant d'accélérer la décision (méthodes *rapides* de k-ppv).

Nettoyage et condensation de l'ensemble d'apprentissage

Le nettoyage d'un ensemble d'apprentissage est une technique très générale, reliée mathématiquement aux méthodes de validation statistique d'un apprentissage. On la présente ici comme une technique inséparable de l'algorithme de condensation : l'usage recommande en effet de ne pas utiliser le second sans le premier.

Algorithme 24 : Algorithme de nettoyage

début
  Diviser aléatoirement l'ensemble d'apprentissage en deux sous-ensembles S_1 et S_2
  tant que *la stabilisation de S_1 et S_2 n'est pas réalisée* **faire**
    Classer tous les points de S_1 sur S_2 par la règle du 1-ppv
    Eliminer de S_1 tous les points dont la classe n'est pas la même que celle de leur plus proche voisin dans S_2
    Classer tous les points de S_2 sur le nouveau S_1 par la règle du 1-ppv
    Eliminer de S_2 tous les points dont la classe n'est pas la même que celle de leur plus proche voisin dans S_1
  fin tant que
  L'ensemble d'apprentissage nettoyé est composé de $S_1 \cup S_2$
fin

L'algorithme de nettoyage n'est pas très utile en soi, dans la mesure où il réduit généralement assez peu la taille de l'ensemble d'apprentissage ; mais il constitue un prétraitement très efficace à la condensation. En effet, la même idée guide ces deux algorithmes : éliminer les points inutiles du point de vue de la décision par plus proche voisin. Mais ils ne s'intéressent pas à l'élimination des mêmes points : la condensation ne garde que les points proches de la frontière des classes ; le nettoyage élimine les points isolés et par conséquent définit ces frontières comme plus simples. Sur les exemples des figures 15.14, 15.15 et 15.16, on voit quatre versions du même ensemble d'apprentissage (à deux dimensions et à deux classes) : son état original, le résultat de son nettoyage, le résultat de sa condensation et celui de la condensation après le nettoyage.

Ces méthodes sont valides au sens où elles ne changent pas le résultat de la classification par k-ppv quand le nombre m d'exemples augmente indéfiniment. En pratique, il a été constaté que leur efficacité diminue considérablement quand la dimension d de l'espace de représentation augmente, même si le nombre des points de l'ensemble d'apprentissage augmente en proportion.

Copyright © 2010 Eyrolles.

Algorithme 25 : Algorithme de condensation

début
  Ordonner les m exemples d'apprentissage de x_1 à x_m
  Initialiser S par x_1 et G par x_2 à x_m
  tant que *S et G ne sont pas stabilisés* **faire**
    pour chaque *point x_i de G* **faire**
      si *Le 1 − ppv de x_i dans S n'a pas la même classe que x_i* **alors**
        Enlever x_i de G et le mettre dans S
      fin si
    fin
  fin tant que
  L'ensemble d'apprentissage condensé est S
fin

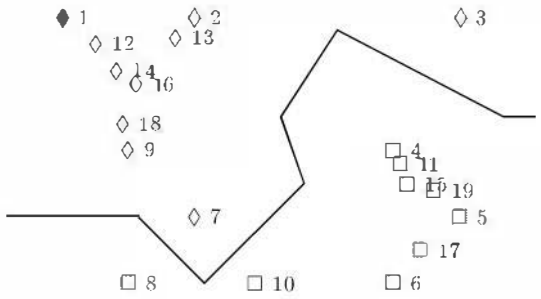


FIG. 15.13: *Un ensemble d'exemples appartenant à deux classes. La surface séparatrice par la règle du 1-ppv est une ligne brisée composée de segments de médiatrices entre couples de points de classe différente.*

Parcours accéléré de l'ensemble d'apprentissage

Les méthodes précédentes ont pour but de réduire la taille de l'ensemble d'apprentissage et donc en proportion le temps nécessaire au classement d'un point inconnu. Il existe d'autres techniques pour réduire ce temps tout en préservant exactement l'ensemble d'apprentissage. La plupart sont basées sur l'inégalité triangulaire de la distance euclidienne Δ. On suppose pour cette méthode que l'on a calculé par avance toutes les distances entre les points d'apprentissage.

L'idée est alors la suivante : soit $\mathbf{x}$ le point à classer ; on est en train de parcourir l'ensemble des points d'apprentissage et le plus proche voisin de $\mathbf{x}$ est pour le moment un certain point d'apprentissage $\mathbf{y}$ à la distance $\Delta(\mathbf{x},\mathbf{y}) = \delta$. Soit $\mathbf{z}$ le point suivant dans l'ensemble d'apprentissage. Si $\Delta(\mathbf{x},\mathbf{z}) \leq \delta$, on réactualise δ et $\mathbf{y}$. Sinon, on peut affirmer que parmi tous les points d'apprentissage restant à examiner, on doit définitivement éliminer deux catégories :

- ceux situés à l'intérieur de la boule de centre $\mathbf{z}$ et de rayon $\Delta(\mathbf{x},\mathbf{z}) - \delta$;
- ceux qui sont à l'extérieur de la boule de centre $\mathbf{z}$ et de rayon $\Delta(\mathbf{x},\mathbf{z}) + \delta$.

En effet, pour tout point $\mathbf{v}$, le triangle $(\mathbf{x},\mathbf{v},\mathbf{z})$ vérifie :

$$\Delta(\mathbf{x},\mathbf{z}) \leq \Delta(\mathbf{x},\mathbf{v}) + \Delta(\mathbf{v},\mathbf{z})$$

Copyright © 2010 Eyrolles.

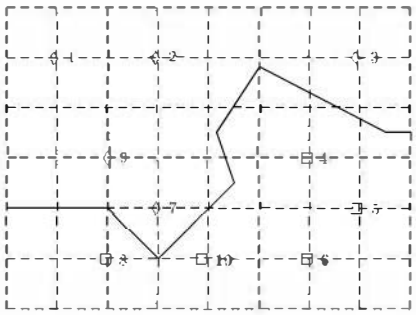


FIG. 15.14: *Un sous-ensemble de l'ensemble précédent, définissant la même surface séparatrice (ensemble simplifié pour pouvoir faire tourner les algorithmes « à la main »).*

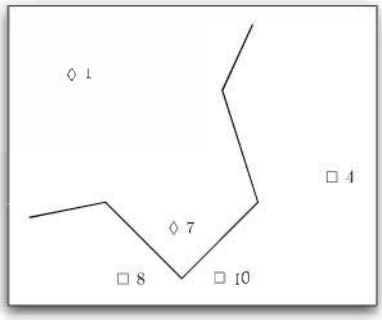


FIG. 15.15: *L'ensemble simplifié après condensation et la nouvelle surface séparatrice par la règle du 1-ppv.*

Si $\mathbf{v}$ est un point d'apprentissage restant appartenant à la première catégorie, on a :

$$\Delta(\mathbf{v},\mathbf{z}) \leq \Delta(\mathbf{x},\mathbf{z}) - \delta$$

soit :

$$\Delta(\mathbf{v},\mathbf{z}) + \delta \leq \Delta(\mathbf{x},\mathbf{z})$$

Donc en combinant les deux inégalités :

$$\Delta(\mathbf{v},\mathbf{z}) + \delta \leq \Delta(\mathbf{x},\mathbf{v}) + \Delta(\mathbf{v},\mathbf{z})$$

d'où :

$$\delta \leq \Delta(\mathbf{x},\mathbf{v})$$

ce qui prouve que le point $\mathbf{v}$ ne peut pas être le plus proche voisin de $\mathbf{x}$.

Un raisonnement analogue mène à éliminer les points appartenant à la seconde catégorie (voir la figure 15.17).

En utilisant ce principe brutalement, il faut pour commencer calculer $m(m-1)/2$ distances, ce qui prend évidemment du temps ; cependant, ce calcul n'est à mener qu'une seule fois, alors que la décision de classement d'un point inconnu est accélérée systématiquement.

On peut grandement améliorer cette technique. Une première manière est de sélectionner astucieusement le point suivant $\mathbf{z}$ dans l'ensemble d'apprentissage, au lieu de garder cet ensemble

Copyright © 2010 Eyrolles.

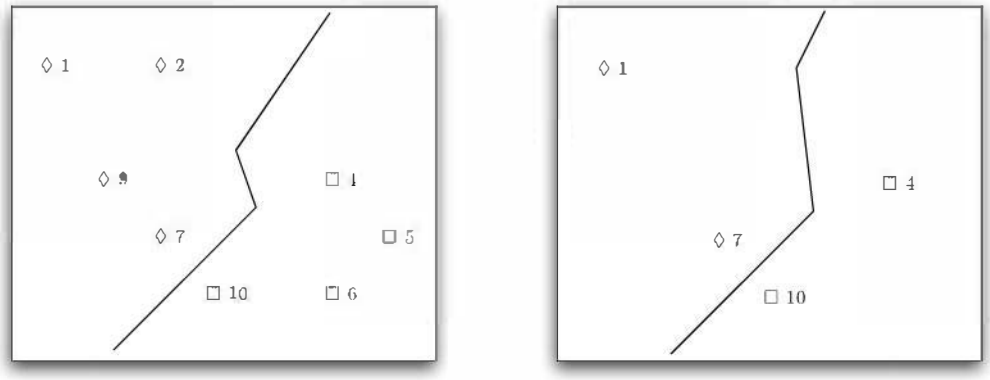


FIG. 15.16: *À gauche, l'ensemble simplifié après nettoyage et la nouvelle surface séparatrice par la règle du 1-ppv. À droite, l'ensemble simplifié après nettoyage puis condensation.*

dans un ordre fixe (arbitraire). Cette combinaison sélection élimination est connue comme l'algorithme AESA. Elle est plus efficace que la version élémentaire présentée, mais demande encore de mémoriser toutes les distances. Cette contrainte peut être levée en la combinant avec une organisation arborescente de l'ensemble d'apprentissage pour appliquer une technique de séparation évaluation *branch and bound*. On trouve en particulier dans [Vid94b] une méthode sophistiquée (LAESA) dont la complexité est en pratique indépendante de m.

— EXEMPLE

Plaçons nous dans $\mathbb{R}^2$, avec l'ensemble d'apprentissage suivant (la classe des exemples n'a pas besoin d'être précisée ici) :

$$\mathcal{S} = \left\{ \mathbf{y}_1 = \begin{pmatrix} 5 \\ 4 \end{pmatrix}, \mathbf{y}_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{y}_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \mathbf{y}_4 = \begin{pmatrix} 0 \\ 8 \end{pmatrix}, \mathbf{y}_5 = \begin{pmatrix} 3 \\ 4 \end{pmatrix} \right\}$$

On calcule d'abord la demi-matrice des distances entre les exemples. Ici, on a tabulé le carré de ces distances :

	y_1	y_2	y_3	y_4	y_5
y_1	0	41	34	41	4
y_2		0	1	64	25
y_3			0	49	18
y_4				0	25
y_5					0

Soit le point $\mathbf{x} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$ dont on cherche le plus proche voisin dans A.

On calcule $\Delta(\mathbf{x}, \mathbf{y}_1) = \sqrt{5}$, et $\mathbf{y}_1$ devient le plus proche voisin courant, à la distance $\delta = \sqrt{5}$. Le lecteur est invité ici à tracer le cercle de centre $\mathbf{x}$ et de rayon δ ainsi que les deux cercles de centre $\mathbf{y}_2$, tangents intérieurement et extérieurement au premier cercle, puis à placer les quatre autres points de $\mathcal{S}$ sur cette figure.

On calcule ensuite $\Delta(\mathbf{x}, \mathbf{y}_2) = \sqrt{18}$ qui est strictement supérieur à δ. $\mathbf{y}_2$ n'est donc pas le nouveau ppv. Quels points de $\mathcal{S}$ peut-on d'ores et déjà éliminer (*marquer*) ?

$\mathbf{y}_3$? Oui, car $\Delta(\mathbf{y}_3, \mathbf{y}_2) \leq \Delta(\mathbf{x}, \mathbf{y}_2) - \delta$ (puisque $\sqrt{1} \leq \sqrt{18} - \sqrt{5}$).

$\mathbf{y}_4$? Oui, car $\Delta(\mathbf{y}_4, \mathbf{y}_2) \geq \Delta(\mathbf{x}, \mathbf{y}_2) - \delta$ (puisque $\sqrt{64} \geq \sqrt{18} + \sqrt{5}$).

Copyright © 2010 Eyrolles.

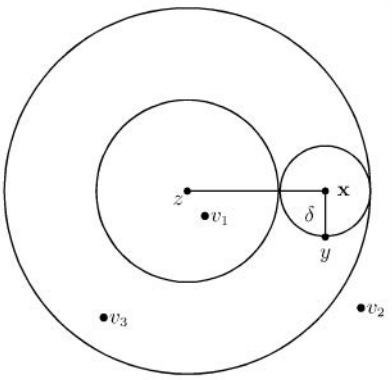


FIG. 15.17: *Méthode accélérée de recherche du plus proche voisin. Le ppv courant de* $\mathbf{x}$ *est* $\mathbf{y}$*, à la distance* δ*. Le point d'apprentissage suivant,* $\mathbf{z}$*, n'est pas à distance inférieure à* δ *de* $\mathbf{x}$*. Aucun point du type* $\mathbf{v}_1$ *ou* $\mathbf{v}_2$ *ne peut plus être le ppv de* $\mathbf{x}$*. En revanche, il faudra calculer la distance* $\Delta(\mathbf{x}, \mathbf{v}_3)$*.*

$\mathbf{y}_5$? Non[12], car on n'a :

- ni $\Delta(\mathbf{y}_5, \mathbf{y}_2) \leq \Delta(\mathbf{x}, \mathbf{y}_2) - \delta$ (puisque $\sqrt{25} > \sqrt{18} - \sqrt{5}$).
- ni $\Delta(\mathbf{y}_5, \mathbf{y}_2) \geq \Delta(\mathbf{x}, \mathbf{y}_2) + \delta$ (puisque $\sqrt{25} < \sqrt{18} + \sqrt{5}$).

On passe au point d'apprentissage suivant non marqué, $\mathbf{y}_5$ et on constate que $\Delta(\mathbf{x}, \mathbf{y}_5) = 1 < \delta$. $\mathbf{y}_5$ est donc le ppv de $\mathbf{x}$ dans $\mathcal{S}$.

4. Les méthodes semi-paramétriques

Nous supposons dans cette section que, contrairement aux méthodes d'estimation paramétriques, nous ne connaissons pas *a priori* la forme analytique des distributions de probabilités. En revanche, nous supposons cependant que ces distributions suivent des lois dont les « hyper paramètres » peuvent être déterminés de manière systématique.

Il n'existe pas à notre connaissance de catalogue exhaustif des méthodes semi-paramétriques. Il risquerait fort de toute façon d'être rapidement obsolète. Nous avons choisi ici de présenter succinctement trois méthodes pour leur intérêt dans les applications pratiques.

4.1 La discrimination logistique

La discrimination logistique s'intéresse aux problèmes à deux classes. Elle est issue de l'observation suivante : pour de nombreuses formes de distributions, la règle de décision bayésienne peut se ramener à une équation linéaire du type :

$$\log \frac{\mathbf{p}(\mathbf{x}|\omega_1)}{\mathbf{p}(\mathbf{x}|\omega_2)} = \theta^\top \mathbf{x} + \theta_0 \qquad (15.35)$$

[12] Noter que s'il existait dans l'ensemble d'apprentissage le point $\mathbf{y}_6 = \begin{pmatrix} -5 \\ -4 \end{pmatrix}$, il ne serait pas éliminé non plus, bien que très éloigné de $\mathbf{x}$.

Copyright © 2010 Eyrolles.

Algorithme 26 : Recherche rapide du plus proche voisin

```
début
    On cherche le ppv de x dans A = {y_1, ..., y_m}
    Calculer toutes les distances Δ(y_i, y_j) ; δ = +∞ ; i = 1
    tant que i < m et y_i non marqué faire
        Calculer Δ(x, y_i)
        si Δ(x, y_i) < δ alors
            δ = Δ(x, y_i) ; ppv y_i
        sinon
            j = i + 1
            tant que j ≤ m faire
                si Δ(y_i, y_j) ≤ Δ(x, y_i) − δ ou Δ(y_i, y_j) ≥ Δ(x, y_i) + δ alors
                    marquer y_j
                fin si
            fin tant que
        fin si
    fin tant que
fin
```

où θ est un vecteur de p paramètres et θ_0 une constante.

Cette approche ne suppose pas de forme analytique pour les distributions d'appartenance aux classes, mais seulement pour leur rapport $\mathbf{p}(\mathbf{x}|\omega_1)/\mathbf{p}(\mathbf{x}|\omega_2)$, ce qui est une hypothèse beaucoup moins forte. Comme un grand nombre de distributions statistiques courantes vérifient cette condition, elle est raisonnable et intéressante.

Par exemple, toutes les lois de probabilité de la famille exponentielle :

$$\mathbf{p}(\mathbf{x}|\theta_k) \;=\; \exp \theta_k^\top \mathbf{x} \;+\; a(\mathbf{x}) \;+\; b_k(\theta_k)$$

respectent cette hypothèse. Ceci inclut aussi bien des distributions de lois continues que des distributions de lois discrètes, par exemple les lois normales (avec égalité des matrices de covariance), la loi gamma, la loi bêta, la loi de Poisson, la loi binômiale, la loi multinômiale, etc.

Dans tous ces cas, la fonction de décision prend la forme suivante :

$$d(\mathbf{x}) \;=\; \log \frac{\mathbf{p}(\mathbf{x}|\omega_1)}{\mathbf{p}(\mathbf{x}|\omega_2)} \;+\; \log \frac{(l_{21} - l_{11})\,\mathbf{p}(\omega_1)}{(l_{12} - l_{22})\,\mathbf{p}(\omega_2)}$$

et l'on attribue la forme $\mathbf{x}$ à la classe ω_1 si $d(\mathbf{x}) \geq 0$ et à la classe ω_2 sinon. Soit, par rapprochement avec l'équation (15.35) :

$$\forall \mathbf{x}, \quad \theta^\top \mathbf{x} + \theta_0 - \log \frac{(l_{21} - l_{11})\,\mathbf{p}(\omega_1)}{(l_{12} - l_{22})\,\mathbf{p}(\omega_2)} \begin{cases} \geq 0 & \text{alors x est attribuée à } \omega_1, \\ < 0 & \text{alors x est attribuée à } \omega_2 \end{cases} \tag{15.36}$$

Cette règle de décision fait clairement ressortir que la fonction de décision est linéaire en $\mathbf{x}$. De plus, c'est la fonction de décision qui est paramétrée, et non, comme dans les approches paramétriques, les vraisemblances conditionnelles $\mathbf{p}(\mathbf{x}|\omega_i)$.

Par ailleurs, il est aisé de montrer que l'équation (15.35) est équivalente à une comparaison de

Copyright © 2010 Eyrolles.

fonctions de type sigmoïde :

$$\begin{aligned} \mathbf{p}(\omega_1|\mathbf{x}) &= \frac{e^{d(\mathbf{x})}}{1 + e^{d(\mathbf{x})}} \\ \mathbf{p}(\omega_2|\mathbf{x}) &= \frac{1}{1 + e^{d(\mathbf{x})}} \end{aligned} \tag{15.37}$$

Nous avons déjà rencontré des expressions de ce type dans le cadre des réseaux connexionnistes (chapitre 10). Ici, elles sont appelées *fonctions logistiques*. On voit d'après ces deux inégalités que si l'approche bayésienne exposé ci-dessus s'appuie sur les distributions conditionnelles $\mathbf{p}(\mathbf{x}|\omega_i)$, l'approche de la discrimination logistique s'appuie en revanche sur les probabilités des classes $\mathbf{P}(\omega_i|\mathbf{x})$.

Pour calculer les paramètres θ_0 et θ, il faut utiliser un échantillon d'apprentissage $\mathcal{S}$ constitué d'exemples de la classe ω_1 et d'exemples de la classe ω_2. Anderson, dans [And82], a montré que ces paramètres peuvent être obtenus par maximisation de la vraisemblance des paramètres conditionnellement aux exemples :

$$\begin{aligned} L &= \prod_{\mathbf{x}\in\omega_1} \mathbf{p}(\omega_1|\mathbf{x}) \prod_{\mathbf{x}\in\omega_2} \mathbf{p}(\omega_2|\mathbf{x}) \\ &= \prod_{\mathbf{x}\in\omega_1} \frac{e^{d(\mathbf{x})}}{1 + e^{d(\mathbf{x})}} \prod_{\mathbf{x}\in\omega_2} \frac{1}{1 + e^{d(\mathbf{x})}} \end{aligned}$$

Il a été de plus montré que sous des conditions très générales [Alb78], le maximum de L est unique.

Cette approche se généralise facilement au cas multi classes. Il faut alors considérer les fonctions :

$$\log \frac{\mathbf{p}(\mathbf{x}|\omega_i)}{\mathbf{p}(\mathbf{x}|\omega_j)} = \theta_{ij}^{\top}\mathbf{x} + \theta_{0ij}$$

4.2 Les mélanges de distributions

Dans la section précédente, nous avons vu qu'une approche, pour rendre plus souple le type de distributions qu'il est possible d'approximer, tout en conservant un moyen de contrôle sur l'espace d'hypothèses, est d'utiliser des fonctions de discrimination paramétrables, comme la fonction logistique. Une autre approche consiste à supposer que les distributions peuvent être décomposées en un produit de distributions plus simples. Dans les modèles de mélanges de distributions, une distribution complexe p est paramétrée comme une combinaison linéaire de distributions plus simples, souvent la distribution normale, sous la forme :

$$\mathbf{p}(\mathbf{x}) = \sum_{i=1}^{M} \lambda_i \, \mathbf{p}_i(\mathbf{x}) \tag{15.38}$$

où les $\lambda_i \geq 0$ sont appelés coefficients de mélange et satisfont à la condition : $\sum_i \lambda_i = 1$. Les distributions $\mathbf{p}_i$ sont appelées les *composantes de mélange* et ont leurs propres paramètres (moyenne, écart type, etc.). Remarquons le lien avec les fonctions noyau : ainsi on peut avoir $\mathbf{p}_i = \kappa(\mathbf{x} - \mathbf{x}_i)$ où $\mathbf{x}_i$ est le centre du noyau et $M = m$. Il y a dans ce cas autant de composantes au mélange que de points d'apprentissage.

Quelques points sont à noter :

Copyright © 2010 Eyrolles.

- Les mélanges de distributions ne prennent pas en compte directement l'étiquette des exemples. Ce sont des moyens d'exprimer des densités de probabilités. Leur estimation ressort donc des techniques d'apprentissage non supervisé (voir au chapitre 18). Il est cependant possible de les utiliser pour des tâches de classification en estimant la distribution de probabilités pour chaque classe tour à tour :

$$\mathbf{p}(\mathbf{x}|\omega_k) = \sum_{i=1}^{M} \lambda_i \, \mathbf{p}_i(\mathbf{x}|\omega_k) \tag{15.39}$$

- Une propriété importante des mélanges de distributions est que pour un large choix de fonctions de base, elles permettent d'approximer avec un degré arbitraire de précision n'importe quelle distribution continue, du moment que le mélange a un nombre suffisant de composantes et que les paramètres sont bien choisis [MB88a].
- Un choix usuel pour les distributions composantes ou fonctions de base est de prendre des fonctions gaussiennes représentant la probabilité conditionnelle d'observer $\mathbf{x}$ quand la classe est ω_k : $\mathbf{p}(\mathbf{x}|\omega_k)$. La plupart des ouvrages comportant une section sur ces méthodes traitent de ce cas (par exemple [Bis95]).
- L'idée des approches semi-paramétriques est de faire varier systématiquement le nombre de paramètres du modèle en fonction de la difficulté du problème traité. Dans le cas des mélanges de distributions, ce principe se traduit par une procédure de choix du nombre M de fonctions de base utilisées dans le mélange. Malheureusement, il semble que ce choix soit un problème notoirement difficile [MB88a, FL94].
- Concernant l'apprentissage d'un mélange de distributions, il est possible d'interpréter les coefficients de mélange λ_i comme des probabilités *a priori* des composantes du mélange. Dans ce cas, pour un point $\mathbf{x}_l$ donné, il est possible d'utiliser le théorème de Bayes pour évaluer la probabilité *a posteriori* correspondante :

$$R_{li} \equiv \mathbf{p}(i|\mathbf{x}_l) = \frac{\lambda_i \, \mathbf{p}(\mathbf{x}_l|i)}{\sum_j \lambda_j \, \mathbf{p}(\mathbf{x}_l|j)} \tag{15.40}$$

La valeur de $\mathbf{p}(i|\mathbf{x}_l)$ peut être vue comme la responsabilité que la composante i assume pour « expliquer » la donnée $\mathbf{x}_l$ (d'où la notation R_{li}). En prenant le logarithme de cette expression on obtient la vraisemblance :

$$\mathcal{L}(\{\pi_i, \mathbf{p}_i\}) = \sum_{l=1}^{m} \log\left\{\sum_{i=1}^{M} \pi_i \, \mathbf{p}(\mathbf{x}_l|i)\right\} \tag{15.41}$$

Malheureusement, la maximisation de cette vraisemblance est beaucoup plus complexe que pour une distribution à une seule composante à cause de la somme dans le logarithme. Un algorithme élégant et puissant pour réaliser cette optimisation est l'algorithme EM (voir l'annexe 7).

4.3 Le cas des réseaux connexionnistes et des arbres de décision

Les réseaux connexionnistes et les arbres de décision sont aussi des familles d'estimateurs semi-paramétriques. Rappellons que les premiers peuvent servir de mécanisme (sophistiqué) pour estimer une densité de probabilité (voir le chapitre 10, paragraphe 3.4).

Quant aux seconds, on peut considérer qu'ils réalisent une estimation à valeur constante dans chaque feuille de l'arbre. Un arbre de décision partitionne en effet l'espace $\mathcal{X}$ des observations

Copyright © 2010 Eyrolles.

possibles récursivement en sous-régions jusqu'à des sous-régions suffisamment homogènes correspondant aux feuilles de l'arbre et dont l'étiquette sert à caractériser les formes qui s'y trouvent.

Nous n'approfondissons pas ces aspects ici mais nous conseillons de consulter la référence [CM98].

Il est également possible d'organiser les mélanges de distributions de manière hiérarchique avec plusieurs composantes correspondant à une composante du niveau supérieur, et ainsi de suite (voir la figure 15.18). Certains réseaux de neurones peuvent être vus comme de tels modèles hiérarchiques.

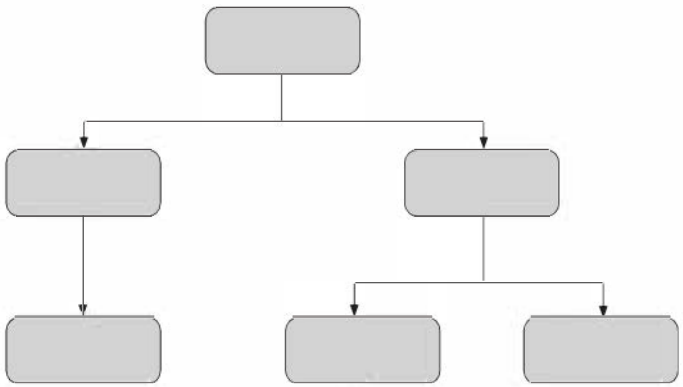


FIG. 15.18: *Un exemple de structure hiérarchique pour un mélange de distributions. Chaque niveau de la hiérarchie correspond à un modèle de distribution, chaque niveau inférieur donnant une représentation plus raffinée et détaillée.*

4.4 Le cas des réseaux connexionnistes

Sans revenir dans le détail du fonctionnement des réseaux de neurones artificiels qui font l'objet du chapitre 10, il existe une famille de réseaux qui peuvent naturellement se prêter à une interprétation sous forme de modèles paramétriques. Il s'agit des *réseaux à passe avant* dans lesquels le signal d'entrée, correspondant à la forme $\mathbf{x}$, est propagée d'une couche de neurones à la suivante jusqu'à la couche de sortie qui produit la sortie y.

Ces réseaux peuvent être utilisés pour une grande variété de tâches suivant les fonctions de base réalisées par chaque nœud du réseau. Nous distinguerons en particulier ici deux types de réseaux correspondant à deux types de fonctions.

1. *Les perceptrons multi-couches.* Ce sont des réseaux de neurones dans lesquels les neurones réalisent une sorte de fonction à seuil Ces réseaux sont généralement utilisés comme des fonctions de discrimination ou de classification permettant de distinguer les formes appartenant à une classe ou à une autre. L'intérêt de ces réseaux est qu'ils peuvent apprendre des fonctions de discrimination arbitrairement complexes dans l'espace des formes d'entrée $\mathcal{X}$. En réglant le nombre de couches de neurones ainsi que le nombre de neurones par couche, il est possible de contrôler en partie et de manière systématique la complexité des frontières de décision. D'où l'appartenance de ces systèmes à la famille des modèles semi-paramétriques.
2. *Les réseaux de neurones à fonctions de base radiales.* Ces résaux sont similaires aux perceptrons multi-couches sauf en ce qui concerne la couche d'entrée pour laquelle les fonctions réalisées par les neurones de la première couche sont des fonctions que l'on appelle à base

Copyright © 2010 Eyrolles.

radiale. Ces fonctions réalisent une sorte de voisinage autour de points particuliers, un peu comme des gaussiennes De ce fait, il est possible d'interpréter le fonctionnement de ces résaux comme des modèles hiérarchiques de distributions de probabilité à symétrie radiale.

4.5 Les méthodes par segmentation

Au lieu d'estimer une densité par un « pavage » sur l'espace $\mathcal{X}$, ce qui est l'essence de l'approche des fonctions noyau, ou de définir une nouvelle règle de décision par vote majoritaire dans un voisinage de la forme considérée, ce qui est réalisé par la méthode des plus proches voisins, il est aussi envisageable de segmenter récursivement l'espace $\mathcal{X}$ en sous-régions jusqu'à ce que soient obtenues des régions suffisamment homogènes du point de vue de l'étiquette des formes d'apprentissage s'y trouvant. Dans cette approche, on définit récursivement un pavage hiérarchique de l'espace des entrées $\mathcal{X}$ pour approximer au mieux la distribution des données.

La figure 4.5 montre un exemple très simple de segmentation d'un espace d'entrée à deux dimensions $\{a_1, a_2\}$. On peut représenter cette segmentation hiérarchique de l'espace par un *arbre de décision*, ou encore *arbre de segmentation* ou *arbre de classification*.

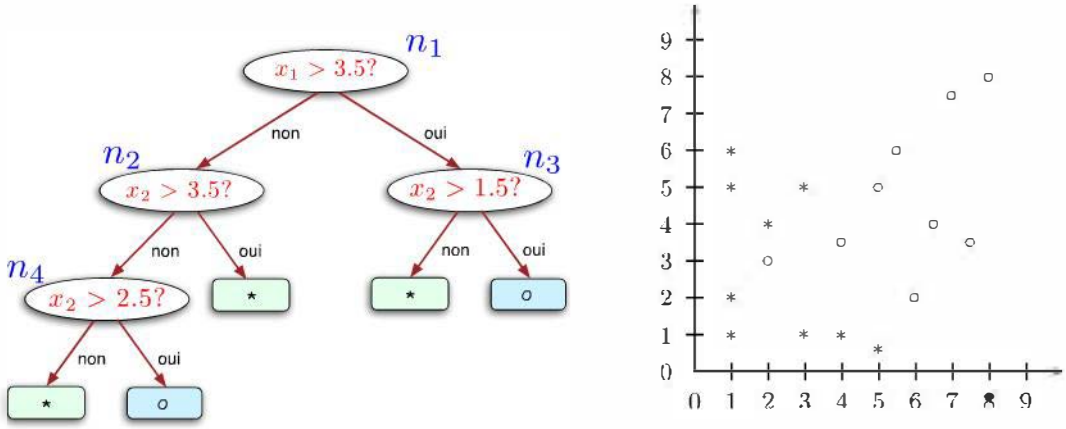


FIG. 15.19: Un arbre de décision utilisant des séparatrices linéaires perpendiculaires aux axes et la partition de l'espace $\mathcal{X}$ qu'il induit.

Un arbre de décision partitionne l'espace $\mathcal{X}$ des observations possibles récursivement en sous-régions jusqu'à des sous-régions suffisamment homogènes correspondant aux feuilles de l'arbre et dont l'étiquette sert à caractériser les formes qui s'y trouvent. Les arbres de décision sont donc un moyen de représenter une partition hiérarchique de l'espace d'entrée. L'apprentissage d'une telle partition s'opère donc naturellement sur l'arbre de décision qui est construit récursivement de la racine vers les feuilles. Comme dans la méthode des plus proches voisins, la segmentation de l'espace se déroule de manière adaptative en fonction des données d'apprentissage.

Dans cet ouvrage, le chapitre 13 leur est entièrement consacré.

Notes historiques et sources bibliographiques

Nous nous sommes contentés ici d'un survol. Plusieurs publications sont à recommander pour aller plus loin : [EH81, TSM85, RW84].

Copyright © 2010 Eyrolles.

E. Parzen a proposé en 1962 [Par62] d'estimer les densités de probabilités par des superpositions de « fenêtres ». Un grand nombre de travaux de probabilistes et de statisticiens ont approfondi cette approche. Les applications en apprentissage ont connu un renouveau avec l'application à certains types de réseaux connexionnistes [Web99], fondés sur des fonctions dites « radiales de base » qui sont des fenêtres de Parzen de type particulier.

Les méthodes des k-plus proches voisins pour la reconnaissance des formes remontent aux années 1950, mais les algorithmes et les preuves de convergence sont un peu plus récentes : T. Cover et P. Hart en ont posé les principes en 1967 [CH67]. Le livre de J. Kittler et P. Devijver [DK82] a apporté un grand nombre de résultats théoriques et pratiques supplémentaires. Les algorithmes rapides sont nombreux, comme on peut le voir en lisant la compilation de B. Dasarathy [Das90]. L'algorithme AESA et ses variantes, les meilleurs du genre, proviennent de E. Vidal, J. Oncina et de leurs équipes, en particulier [Vid94b, MO98]. Le domaine de recherche est toujours actif (voir par exemple [GBMO06]). Une revue remarquable des méthodes rapides de recherche du plus proche voisin est donnée dans [CNBYM01]. On lira aussi avec intérêt les articles recueillis par D. Aha [Aha97] où la technique « paresseuse » de la recherche de formes similaires dans un ensemble d'exemples est étendue à de nombreux domaines de l'apprentissage, y compris à des données symboliques.

Résumé

- L'apprentissage bayésien consiste à partir d'hypothèses *a priori* pour les réviser en fonction des données d'apprentissage.
- Cette opération est optimale au sens probabiliste : les hypothèses *a posteriori* obtenues de la sorte sont en effet les plus vraisemblables.
- L'apprentissage bayésien requiert d'une part une connaissance *a priori* sur la vraisemblance des hypothèses en concurrence et d'autre part celle la probabilité des données d'apprentissage conditionnellement à ces hypothèses. Ces valeurs doivent être estimées à partir de l'ensemble d'apprentissage.
- Les méthodes d'estimation paramétrique font l'hypothèse que la distribution à estimer possède une certaine forme analytique et trouvent les meilleurs paramètres.
- Les méthodes d'estimation non paramétrique estiment une densité conditionnelle en un point en examinant comment l'ensemble d'apprentissage se comporte au voisinage de ce point.
- Les méthodes des k-plus proches voisins ont l'avantage de la simplicité. Une algorithmique efficace existe pour les rendre rapides.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 16

L'apprentissage de réflexes par renforcement

L'un des problèmes les plus fascinants en apprentissage est celui de l'adaptation en ligne, par renforcement. Il est en effet omniprésent dans les systèmes naturels, y compris chez les plus simples organismes, et correspond à une large classe d'applications dans laquelle il n'est pas envisageable de fournir les informations détaillées nécessaires à l'apprentissage supervisé. Dans sa forme la plus simple, cet apprentissage implique un système agissant dans le monde et soumis de ce fait à une séquence de signaux correspondant aux états successifs du système. De temps en temps, un signal de renforcement positif ou négatif sanctionne la séquence de décisions prises par le système. La tâche du système est de chercher une stratégie de conduite, appelée « politique » dans ce contexte, qui maximise l'espérance de renforcement dans les situations à venir. Cela passe généralement par une estimation des espérances de renforcement soit en fonction des états, soit en fonction des actions du système dans le monde.

L'apprentissage par renforcement est difficile pour deux raisons principales. D'une part, le signal de renforcement fourni au système en retour de ses actions est très pauvre ; c'est généralement un simple scalaire et il n'apporte donc que peu d'informations sur le monde et sur les décisions à prendre. D'autre part, le délai qui sépare le signal de renforcement des décisions qui y ont conduit rend ardue l'attribution de mérite ou de blâme à chacune des décisions prises dans le passé.

Malgré ces difficultés, l'apprentissage par renforcement a suscité de nombreux travaux depuis plus de quarante ans en automatique et en apprentissage artificiel. La notion d'un système autonome interagissant directement avec l'environnement et tirant de cette expérience une connaissance du monde suffisante pour y « survivre » et prospérer est en effet très séduisante intellectuellement. Ensuite, c'est finalement, pour le moment, l'un des rares apprentissages artificiels étudiés qui s'inscrive dans la durée. Finalement, de très nombreuses applications potentielles existent dans les domaines du contrôle de processus, de la navigation, de la conduite de robot, de l'apprentissage dans les jeux, de la planification financière, etc.

Copyright © 2010 Eyrolles.

Sommaire

1 Description du problème 534
1.1 La modélisation d'un agent en action dans le monde 535
1.2 Les notions fondamentales 536
1.2.1 Les mesures de gain 536
1.2.2 La mesure de performance de l'apprentissage 537
1.3 Les processus décisionnels de Markov 538
1.4 Les problèmes et les grandes approches 538
1.5 Le dilemme exploration contre exploitation 539
2 Si tout est connu : l'utilité de la fonction d'utilité 541
3 L'apprentissage des fonctions d'utilité quand l'environnement est connu 543
3.1 L'évaluation d'une politique par propagation locale d'information 543
3.2 Un théorème conduisant à l'amélioration de politique 544
3.3 Processus itératif d'amélioration de politique 546
4 Sans modèle du monde : la méthode de Monte-Carlo 547
5 La méthode des différences temporelles 548
5.1 L'évaluation suivant la méthode des différences temporelles 548
5.2 L'amélioration de politique avec les différences temporelles 549
5.3 SARSA : une méthode d'amélioration « sur politique » 550
5.4 Le Q – *learning* : une méthode d'amélioration « hors politique » 551
5.5 TD(λ) : les méthodes de différences temporelles à plusieurs pas 551
6 La généralisation dans l'apprentissage par renforcement 552
6.1 Le problème 552
6.2 Généralisation par approximation de la fonction d'utilité 553
6.3 Méthodes de généralisation par partition de l'espace 555
6.4 Méthodes directes d'apprentissage de politique 556
6.5 Les méthodes « actor-critic » 557
6.6 Apprentissage par renforcement bayésien 557
7 Contrôle optimal par recherche arborescente et algorithme UCT . 558
8 Le cas des environnements partiellement observables 561
9 Exemples d'application 562
9.1 Le TD-Gammon 562
9.2 Le jeu de Go 563
9.3 Applications au contrôle et à la robotique 564
10 Bilan et perspectives 565

Copyright © 2010 Eyrolles.

UN CANARD automatique, un programme, un robot peuvent-ils apprendre à se comporter dans un environnement inconnu, ou en tout cas dont ils ne perçoivent l'existence que de l'étroite fenêtre de leurs sensations et par des réponses agréables ou désagréables à leurs actions ? On sait qu'un chien se dresse par punition-récompense et que des animaux généralement jugés moins intelligents, comme les oies ou les corbeaux, sont capables de modifier durablement leur comportement en fonction d'essais et de réponses du monde extérieur. Mais ce type d'apprentissage, fondamental dans le monde animal, est-il modélisable et transposable pour des programmes?

Il semble désormais que la réponse soit positive, grâce aux techniques de l'apprentissage par renforcement. Il est aujourd'hui possible d'imaginer un robot arrivant sur une planète inconnue où il doit tout découvrir : les éventuelles chausses-trappes, les endroits lui permettant de recharger ses batteries, l'action de ses roues dans cet environnement inconnu. Son objectif est de survivre le mieux possible dans ce monde. Pour cela, il doit identifier les différents *états* dans lesquels il peut se trouver, apprendre à associer des effets à ses actions pour chaque état possible et découvrir comment ce monde extérieur associe une réponse aux états.

Dans cette situation d'apprentissage, l'espace des hypothèses est particulier puisqu'il s'agit d'**apprendre deux fonctions** : l'une faisant passer d'état en état et l'autre faisant effectuer une action à partir d'un état, l'une et l'autre tenant compte de la récompense ou de la punition associée par le monde extérieur. Il s'agit donc d'un problème d'optimisation numérique, mais dans lequel les techniques par exploration définies au chapitre 3 ne sont que de peu d'utilité.

Naturellement, le robot de l'illustration ci-dessous n'est pas doté de facultés d'apprentissage ; il n'est pourtant pas impossible d'envisager des systèmes mécaniques améliorant leur comportement par apprentissage par renforcement, mais il est beaucoup plus facile d'utiliser des machines programmables pour cela.



FIG. 16.1: *Une réplique réalisée par Frédéric Vidoni en 1998 du canard de J. de Vaucanson (c. 1741). Musée des automates de Grenoble.*

Copyright © 2010 Eyrolles.

Notations utiles pour ce chapitre

$\mathcal{E}$	L'ensemble des états
$\mathcal{A}$	L'ensemble des actions
$\mathcal{R}$	L'ensemble des signaux de renforcement (souvent $\mathbb{R}$)
s	Un état
a	Une action
$\pi(s,a)$	La probabilité que l'action a soit choisie dans l'état s par la politique π
$Q(s,a)$	L'espérance de gain quand l'action a est prise dans l'état s
$V(s)$	L'espérance de gain dans l'état s
r_t	Le signal de renforcement reçu par l'agent à l'instant t
R_t	Le gain cumulé à partir de l'instant t
$E_\pi(\cdot)$	Espérance en suivant la politique π
$0 \leq \gamma \leq 1$	Le taux de diminution des renforcements

1. Description du problème

Nous nous intéressons à un agent situé dans un certain environnement qu'il ne connaît pas, ou du moins qu'il ne connaît qu'imparfaitement. Cet agent cherche à recevoir des récompenses et à éviter les punitions, qui dépendent de ses actions. Comment doit-il s'y prendre ? Quelle ligne de conduite doit-il suivre ?

Reprenons les choses plus formellement. Les récompenses et punitions, pour lesquelles nous utiliserons désormais le terme de *signal de renforcement*, dépendent de l'état présent du système, ou de la transition entre deux états. Plus précisément, nous supposons qu'à chaque état, ou à chaque transition, correspond un signal de renforcement, éventuellement nul. Si l'agent cherche à maximiser une certaine espérance de gain, il doit donc chercher à atteindre les états correspondant aux signaux les plus favorables (les récompenses). Pour cela, l'agent doit être capable de mesurer son état courant et d'identifier les actions qui sont les plus à même de le conduire vers les états favorables. Les problèmes à résoudre sont les suivants :

1. L'agent ne connaît pas son environnement, ce qui signifie :
   - qu'il ne connaît pas les signaux de renforcement associés à chaque état ;
   - qu'il ne connaît pas la topologie de l'espace des états, c'est-à-dire notamment les états accessibles à partir d'un état donné.
2. L'agent ne connaît pas l'effet de ses actions dans un état donné, c'est-à-dire la fonction qui associe à chaque couple (état, action) un état résultant.

Dans les méthodes d'apprentissage par renforcement, l'agent n'a pas de connaissances *a priori* sur le monde et il opère par une sorte de reconnaissance des états du monde et une interpolation de ses observations. Il est évident que ce type d'apprentissage par identification progressive nécessite un long apprentissage. En contrepartie de cette relative inefficacité, et parce qu'il s'accomode de présupposés très faibles, il s'agit d'un apprentissage applicable dans une grande variété de situations. En fait, la seule hypothèse est que le monde est de nature stochastique (les actions peuvent avoir des effets non déterministes) et *stationnaire* (les probabilités de transition entre états, et les signaux de renforcement, restent stables au cours du temps).

Copyright © 2010 Eyrolles.

1.1 La modélisation d'un agent en action dans le monde

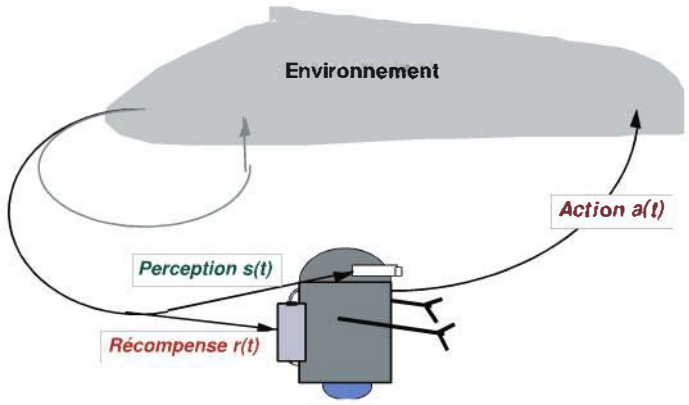


FIG. 16.2: *Le schéma abstrait d'un agent en interaction avec le monde suivant trois canaux : perception, renforcement immédiat et action instantanée.*

La théorie de l'apprentissage par renforcement s'appuie sur une modélisation des agents et des environnements qui est une idéalisation manipulable des situations réelles d'apprentissage, sans être triviale au point de trop s'éloigner de la réalité. Cette modélisation suppose que l'agent communique avec son environnement par trois canaux distincts :

- Un *canal perceptif* par lequel l'agent mesure l'état dans lequel il se trouve dans l'environnement. Ce canal correspond aux données fournies par un ensemble de capteurs, par exemple des caméras, des capteurs de proximité à ultrasons, des centrales inertielles, etc. Les informations fournies sont souvent partielles et parfois erronées. Nous notons $s(t)$ l'ensemble des informations passant par ce canal à l'instant t.
- Un canal spécifique aux *signaux de renforcement* renseignant l'agent sur la qualité de l'état courant. On suppose dans l'apprentissage par renforcement que ce canal ne transmet qu'un scalaire[1]. Nous notons $r(t)$ l'information transmise par ce canal. Ce signal n'est généralement pas disponible dans tous les états, mais seulement pour quelques états particuliers. Par exemple, c'est seulement à la fin d'une partie d'échecs que l'on dispose de la sanction : perte, gain ou nulle.
- Un canal qui transmet à l'environnement l'*action de l'agent*. Nous notons $a(t)$ l'information ainsi transmise de l'agent vers l'environnement. Généralement, ce signal déclenche une modification de l'état de l'environnement, comme quand un robot fait tourner ses roues ou quand un joueur d'échecs joue un coup. Ici aussi, la modification de l'état peut être non déterministe dans la mesure où l'agent n'a pas une connaissance parfaite de l'environnement.

Nous notons $\mathcal{E}$ l'*espace des états* dans lesquels peut se trouver l'agent, $\mathcal{R}$ l'*espace des signaux de renforcement*, c'est-à-dire un intervalle de la forme $[-a, +b]$ avec $a, b \in \mathbb{R}^+$, et $\mathcal{A}$ l'*espace des actions* disponibles pour l'agent. Dans ce cadre, nous posons donc qu'à chaque instant t, l'agent perçoit le monde comme étant dans l'état $s_t \in \mathcal{E}$. Il choisit alors d'effectuer l'action $a_t \in \mathcal{A}$ parmi les actions possibles dans l'état courant. À la suite de cette action prise dans cet état, il reçoit un signal de renforcement immédiat $r_t \in \mathcal{R}$.

L'**agent** peut donc être considéré comme réalisant une fonction de $\mathcal{E}$ dans $\mathcal{A}$: $s_t \mapsto a_t$. Suivant la terminologie en usage, nous appelons *politique* cette fonction de comportement, et nous notons

[1] Dans le cas des organismes naturels, un certain précâblage existe pour percevoir ce type de signal comme une douleur ou un plaisir plus ou moins forts.

Copyright © 2010 Eyrolles.

π_t la politique à l'instant t. Plus précisément, une politique est une fonction définie de $\mathcal{E} \times \mathcal{A}$ dans $\mathbb{R}$, qui associe à chaque état s et chaque action a possible dans s, la probabilité $\pi(s, a)$ associée de choisir l'action a dans s. Si les probabilités sont uniformément nulles sauf pour une action, l'agent est déterministe.

L'**environnement** peut être considéré pour sa part comme implémentant une fonction de $\mathcal{E} \times \mathcal{A}$ dans $\mathcal{E} \times \mathcal{R}$: $(s_t, a_t) \mapsto (s_{t+1}, r_t)$. Pour des raisons de clarté, il est utile de décomposer cette fonction en deux fonctions. La première est une *fonction de transition entre états* notée T. Elle traduit la dynamique du monde ; elle est définie de $\mathcal{E} \times \mathcal{A}$ dans $\mathcal{E}$: $(s_t, a_t) \mapsto s_{t+1}$. La seconde est une *fonction de renforcement immédiat* R de $\mathcal{E} \times \mathcal{A}$ dans $\mathcal{R}$: $(s_t, a_t) \mapsto r_t$. Chacune de ces fonctions est stochastique, soumise à des aléas imprévisibles, qu'on suppose issus d'une distribution stationnaire.

1.2 Les notions fondamentales

Dans cette section, nous allons formaliser les concepts introduits par la théorie actuelle de l'apprentissage par renforcement et décrire les problèmes et les grandes familles d'approches.

L'apprentissage par renforcement considère un apprenant plongé dans un environnement et devant essayer, par ses actions, de maximiser une mesure de gain dépendant des signaux de renforcement reçus tout au long de son existence dans le monde. L'une des premières questions consiste donc à spécifier cette mesure de gain.

1.2.1 Les mesures de gain

Précisons d'emblée qu'il n'y a pas de mesure de gain universelle valable pour toutes les situations. Chaque domaine d'application est susceptible de posséder une mesure adaptée. Ainsi, le joueur d'échec est sans doute sensible au compte des pièces gagnées ou perdues en cours de partie, à certains critères tels que le contrôle du centre, mais ce qui l'intéresse avant tout est l'issue ultime de la partie : gain, perte ou nulle. En revanche, pour un fournisseur d'énergie électrique qui essaie de réguler sa production en fonction de la demande, laquelle dépend de la météorologie, de l'heure de la journée, de la situation économique, etc., il est important de mesurer les gains et les coûts tout au long du processus de production. La mesure de gain doit donc être différente dans ce cas. En général, on s'intéresse à une mesure de gain cumulée dans le temps. Ainsi, que l'on ne tienne compte, comme aux échecs, que du du gain ultime, ou bien que l'on moyenne les gains réalisés en cours d'action, ou encore que l'on tienne compte, comme en économie, de gains pondérés par des taux d'intérêt, toutes les options sont envisageables et dépendent de l'application concernée. Cependant, trois mesures ont été plus particulièrement distinguées dans les recherches sur l'apprentissage par renforcement (on suppose que l'on part de l'état s_0).

- *Gain cumulé avec horizon fini* :

$$R_T = r_0 + r_1 + r_2 + \ldots + r_{T-1} \,|s_0 = \sum_{t=0}^{T-1} r_t \,|s_0 \tag{16.1}$$

- *Gain cumulé avec intérêt et horizon infini* :

$$R = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \,|s_0 = \sum_{t=0}^{\infty} \gamma^t \, r_t \,|s_0 \tag{16.2}$$

où γ joue le rôle d'un taux d'intérêt : $0 \leq \gamma \leq 1$.

Copyright © 2010 Eyrolles.

- *Gain en moyenne* :

$$R_T = \frac{1}{T-1} \sum_{t=0}^{T-1} r_t \,|s_0 \tag{16.3}$$

Il faut noter que le choix de la mesure de gain a une grande influence sur le choix de la meilleure politique par le système (voir la figure 16.3). Il est donc essentiel de peser soigneusement sa définition avant de se lancer dans le processus d'apprentissage.

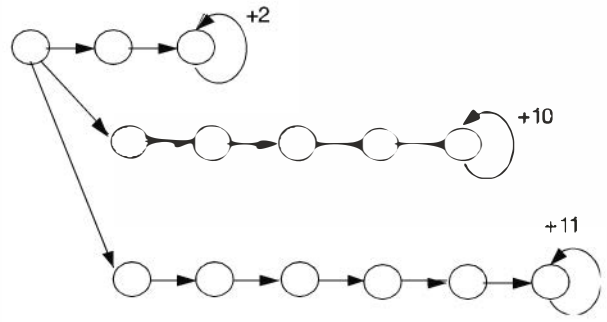
Avec $T = 4$ et $\gamma = 0.9$, quelle politique choisir ?	$\sum_{t=0}^{T} r_t$	$\sum_{t=0}^{\infty} \gamma^t r_t$	$\text{Lim}_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} r_t$
	**6**	16	2
	0	**59**	10
	0	58.4	**11**

FIG. 16.3: *Dans l'exemple ci-dessus, le choix de la meilleure stratégie, parmi les trois possibles, dépend du critère de gain adopté. Avec $k = 4$ et $\gamma = 0.9$, quelle est la meilleure politique dans l'état initial ? Si l'on vise la première formule de gain, il faut choisir la première politique puisqu'elle conduit au meilleur gain. Pour la deuxième formule de gain, il faut opter pour la seconde politique. Et pour la troisième formule, il faut choisir la troisième politique (d'après un exemple dû à [KLM96]).*

Par ailleurs, l'une des caractéristiques essentielles de ces critères de gains est qu'ils sont additifs. Cela permet d'appliquer le *principe d'optimalité de Bellman* [Bel57a] (« Les sous-politiques d'une politique optimale sont elles-mêmes optimales ») qui est à la base des algorithmes de programmation dynamique permettant de résoudre efficacement les problèmes de décision markoviens [Bel57b].

1.2.2 La mesure de performance de l'apprentissage

Dans ce qui précède, nous avons introduit les éléments de mesure des performances de l'agent dans son univers. Une autre question concerne la mesure de performance de l'apprentissage lui-même. Sans entrer dans les détails, il est clair qu'il s'agit d'un paramètre important. En effet, l'apprentissage par renforcement suppose un agent apprenant par essais et erreurs. Les erreurs peuvent coûter cher : soit parce qu'elles correspondent à des dégradations de l'environnement (par exemple si un robot heurte des objets ou un autre robot durant son apprentissage) ou à des dégradations de l'agent lui-même, ou encore parce qu'ils exigent de très longues sessions d'apprentissage. C'est pourquoi il est intéressant de caractériser l'apprentissage en fonction de ses propriétés de convergence vers une politique optimale, ou bien aussi en fonction de la complexité en calculs ou encore en nombre d'expériences (paires *situation, action*) nécessaires. Par exemple, un ordre de grandeur pour l'apprentissage du jeu d'Othello par renforcement est du million de parties d'essai à jouer par l'agent avant d'avoir un bon niveau de jeu. Ce qui est tolérable pour

Copyright © 2010 Eyrolles.

ce type de tâche, facile à simuler, peut l'être beaucoup moins lorsque l'apprentissage implique un agent réel, par exemple une sonde d'exploration planétaire.

On cherche donc autant que possible à analyser les propriétés de convergence asymptotique vers une politique optimale, ainsi que les complexités calculatoire et en nombre d'essais (mesurés en nombre d'actions ou de séquences d'actions par exemple) de cette convergence.

1.3 Les processus décisionnels de Markov

L'apprentissage par renforcement implique un problème de *décision séquentielle*, ou de décisions répétées, *dans l'incertain*. En effet, chaque décision de l'agent a une influence potentielle sur les problèmes de décision subséquents. De ce fait, les entrées ne peuvent plus être considérées comme indépendantes et identiquement distribuées (i.i.d.), comme dans le cadre dominant de l'apprentissage statistique. On retrouve ici les problèmes de planification connus depuis longtemps en intelligence artificielle, et, particulièrement, les problèmes de recherche de chemin les plus courts dans un graphe. Par ailleurs, les conséquences des décisions possibles sont incertaines, soit que le monde soit insuffisamment connu (ou qu'il comporte des agents étrangers), soit que le calcul parfait des conséquences soit trop coûteux, particulièrement dans des contextes de (quasi) temps réel.

Les problèmes de décision dans l'incertain font l'objet de nombreuses formalisations et méthodes de résolution. Les *processus décisionnels de Markov* (PDM) sont l'une de ces approches, la plus classique. Ils généralisent les méthodes de recherche de plus court chemin dans un environnement stochastique. Lorsque l'on suppose que l'on peut rendre compte des situations où se trouve l'agent par des *états*, des transitions entre ces états par des *actions* et que l'on peut associer à ces transitions d'états des *récompenses* ou *renforcements*, on peut alors formaliser un problème de décision markovien comme une recherche dans l'espace des chaînes de Markov visitant des états sous l'effet des actions et valuées par les renforcements. La solution d'un problème de décision markovien n'est pas une séquence de décisions, qui dépend d'aléas incontrôlables, mais une *politique* ou une stratégie qui spécifie l'action à prendre en chacun des états rencontrés pour maximiser l'espérance de gain.

1.4 Les problèmes et les grandes approches

Le problème d'optimisation ayant été défini, plusieurs approches sont envisageables pour le résoudre.

La première consiste à chercher à *apprendre directement un modèle de l'environnement* en estimant d'une part la *fonction de renforcement* immédiat associée à chaque état, ou à chaque couple (état, action) R définie de $\mathcal{E}$ dans $\mathcal{R}$: $s_t \mapsto r_t$, et, d'autre part, la *fonction de transition* T caractérisant la dynamique de l'environnement définie de $\mathcal{E} \times \mathcal{A}$ dans $\mathcal{E}$: $(s_t, a_t) \mapsto s_{t+1}$. Le problème est alors celui d'un apprentissage supervisé s'appuyant sur les exemples glanés en cours d'expérience. L'inconvénient de cette approche, outre qu'elle nécessite de chercher à tester toutes les situations possibles, et cela plusieurs fois, est de ne pas tenir compte des interactions entre les états.

Une autre approche prend en compte ces interactions par l'introduction de *fonctions d'utilité*. Il s'agit de fonctions traduisant l'espérance de gain à partir d'un état ; fonction notée $V(s)$ et définie sur $\mathcal{E}$, ou à partir d'un couple (*état, action*) ; fonction notée $Q(s, a)$ et définie sur $\mathcal{E} \times \mathcal{Z}$. Ces fonctions estiment sur le long terme la qualité des états ou des couples (*état, action*). Elles sont donc à différencier des fonctions de renforcement immédiat. Dans ce cas, l'apprentissage consiste à agir dans le monde et à calculer pour chaque *état* ou couple (*état, action*) l'espérance

Copyright © 2010 Eyrolles.

de gain associée. Un apprentissage de ces fonctions d'utilité peut tirer profit des interdépendances entre états.

Finalement, il est possible d'envisager de *travailler directement dans l'espace des politiques* plutôt que de passer par l'intermédiaire de fonctions locales aux états. C'est par exemple ainsi qu'opère le mécanisme de sélection darwinienne dans la nature (voir chapitre 8). Chaque agent correspond à une certaine politique, et en moyenne les agents de performances médiocres sont éliminés au profit d'agents supérieurs. Ce type d'approche a également fait l'objet d'expériences dans le cadre de modèles d'évolution simulée (voir chapitre 8). Le succès dépend en grande partie de la structuration de l'espace des politiques. Si celle-ci est faible, il faut avoir recours aux méthodes d'apprentissage de fonctions locales, c'est-à-dire aux méthodes d'apprentissage par renforcement proprement dites que nous allons désormais examiner.

L'apprentissage par un mécanisme local d'évaluation d'une fonction d'utilité requiert l'exploration de l'espace des possibilités par l'agent et un mécanisme d'essais/erreurs. La fonction d'utilité traduit les circonstances les plus favorables ou non, et, implicitement, permet d'identifier les séquences d'actions conduisant aux meilleurs gains. Il y a donc une tension entre l'activité d'exploration, nécessaire pour obtenir une information fiable et précise, et l'exploitation de ces informations pour atteindre le meilleur gain possible. Cela se traduit par le fameux compromis exploration contre exploitation.

1.5 Le dilemme exploration contre exploitation

Plaçons-nous dans le cas d'un agent ne connaissant pas son environnement et cherchant à maximiser une mesure de gain cumulée. Pour ce faire, il doit naturellement commencer à explorer son univers pour en découvrir les potentialités, mais très vite il se trouve devant un dilemme. Ayant découvert que certains comportements semblent plus profitables que d'autres, doit-il chercher à les reproduire au maximum afin de maximiser le gain recherché au risque de passer à côté d'opportunités non encore identifiées, ou doit-il continuer son exploration au risque de perdre du temps et de ne pas réaliser un bon gain cumulé ? Ce problème est classique dans tous les scénarios où un agent doit prendre des décisions « en ligne » et optimiser une mesure de performance le long de sa trajectoire. Une modélisation simple de ce genre de situations est celle d'un joueur placé devant l'une de ces machines que l'on appelle[2] le « bandit à deux bras » (voir figure 16.4).

Le principe est le suivant. Un joueur dispose de m jetons avec lesquels il peut jouer avec cette machine. Pour chaque jeton inséré dans la fente, le joueur peut tirer sur l'un ou l'autre des bras. Il reçoit alors un certain nombre de pièces correspondant à son gain. Les bras sont notés A_1 et A_2, ils ont une espérance de gain respective de μ_1 et μ_2 avec une variance respective de σ_1^2 et σ_2^2. Cela signifie que le gain associé à chaque bras est aléatoire, avec une certaine moyenne et un certain écart-type stationnaires. Les tirages aléatoires sont supposés indépendants. Le joueur ne connaît ni les moyennes ni les variances associées à chaque bras et doit donc les estimer en cours de jeu. Le joueur ne sait donc pas quel est le bras dont l'espérance de gain est la meilleure, et il doit essayer de maximiser son gain avec ses m jetons. Quelle doit alors être sa stratégie de tirage des bras étant donnée son estimation courante des moyenne et variance de chaque bras ?

Une stratégie extrême est de tirer une fois chaque bras, de noter celui qui a donné le meilleur résultat, puis de jouer désormais systématiquement celui-ci. Cela correspond à une *stratégie d'exploitation pure* : ne plus explorer plus dès qu'on possède des éléments d'information minimaux. Bien sûr, il existe un risque que les deux tirages initiaux n'aient pas révélé le meilleur bras à cause de la variance des résultats et qu'en conséquence ce soit le plus mauvais bras qui ait été tiré systématiquement à partir de là. Plus généralement, on appelle stratégie d'exploitation

[2] La machine à sous classique de casino est appellée en argot américain « bandit manchot » (*one-armed bandit*).

Copyright © 2010 Eyrolles.

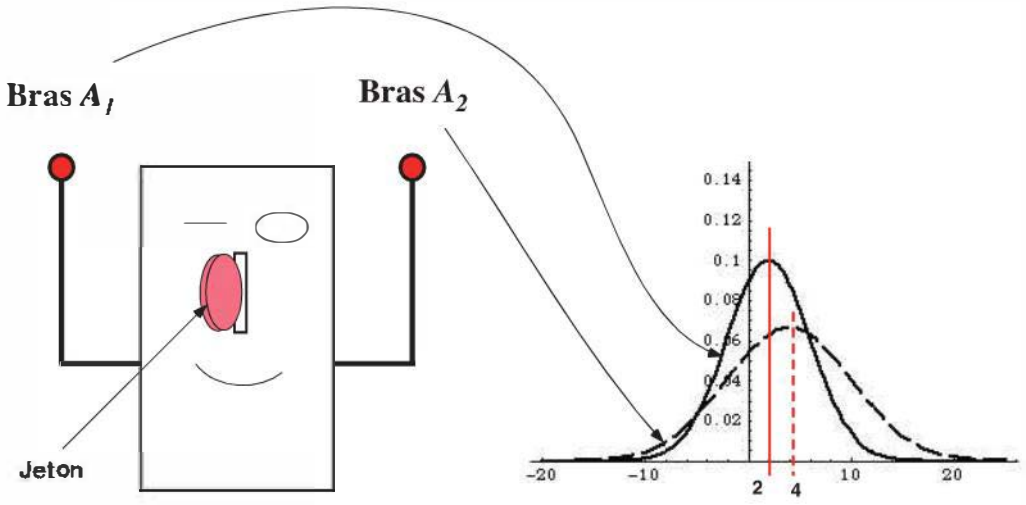


FIG. 16.4: *Une machine à sous appelée « bandit à deux bras ». Chacun des bras est associé à un gain aléatoire suivant une distribution normale. Par exemple, ici, le bras A_1 correspond à une loi de moyenne 4 et de variance 6, alors que le bras A_2 correspond à une loi de moyenne 2 et de variance 4. Il peut bien sûr arriver que sur un tirage le bras A_1 donne un résultat inférieur au résultat obtenu avec un tirage de A_2.*

toute stratégie qui choisit l'action dont l'estimation courante de gain est la plus élevée.

La stratégie extrême inverse consiste à tirer $\lfloor \frac{m-1}{2} \rfloor$ fois sur le bras gauche et $\lfloor \frac{m-1}{2} \rfloor$ fois sur le bras droit[3], puis à tirer le dernier ou les deux derniers coups sur le bras dont la moyenne observée est la meilleure. Cela correspond à une *exploration pure*, dans laquelle on alloue quasiment toutes les décisions à l'exploration de l'environnement avant de choisir la décision ultime. Celle-ci est alors prise avec une connaissance aussi grande que possible, mais au prix de n'avoir pas cherché à optimiser le gain durant la phase d'exploration.

On sent bien que la stratégie optimale doit se situer entre ces deux types de politiques et qu'elle correspond à la résolution d'un compromis entre exploitation et exploration. Sa résolution analytique (voir par exemple [Hol75]) montre que, dans le cas du bandit à deux bras, la meilleure stratégie consiste, au fur et à mesure de l'acquisition d'informations sur les probabilités de gain de chaque bras, à accroître exponentiellement le nombre de tirages du bras paraissant le meilleur par rapport à l'autre (on peut trouver également une analyse simplifiée du bandit à deux bras dans [Mit96]).

Nous verrons dans la suite que la tâche d'apprentissage par renforcement implique également la résolution d'un conflit entre exploration et exploitation. Si les leçons générales du problème des bandits à deux bras restent valables, elles doivent être affinées pour chaque type d'environnement. En général, il est d'ailleurs impossible d'obtenir une solution analytique par insuffisance de connaissances sur le modèle sous-jacent de l'environnement.

La résolution du compromis exploitation contre exploration se traduit par une méthode de sélection de l'action a_t à un instant donné t en fonction des évaluations associées à chaque action a_i possible. Plusieurs méthodes ont été proposées. Les méthodes dites *non dirigées* reposent sur les évaluations des actions, alors que les méthodes dites *dirigées* utilisent en plus des heuristiques exploitant des informations acquises lors de l'apprentissage.

[3] La notation $\lfloor x \rfloor$ indique l'entier immédiatement inférieur à x.

Copyright © 2010 Eyrolles.

1. **Méthodes non dirigées**
   - Prendre la meilleure action pendant N_1 pas de temps, puis tirer uniformément a_t dans $\mathcal{A}$ pendant N_2 pas de temps.
   - Méthode *ε-greedy*. Tirer la meilleure action connue (selon la fonction d'utilité courante) avec une probabilité $1-\varepsilon$, ou tirer uniformément dans $\mathcal{A}$ avec une probabilité ε ($\varepsilon \in [0,1]$).
   - Méthode *softmax*. Tirer l'action a_t selon une distribution de Boltzmann :

$$p_t(a) = \frac{\exp(-\frac{Q_t(s,a)}{T_t})}{\sum_{a'} \exp(-\frac{Q_t(s,a')}{T_t})}$$

   où T_t est la « température » à l'instant t, et $\lim_{t\to\infty} T_t = 0$, et où $Q_t(s,a)$ est supposée être la fonction d'utilité associée à l'action a dans l'état s.

2. **Méthodes dirigées**
   - *Méthode basée sur la récence*. La fonction d'utilité associée aux actions prend un compte un *bonus* favorisant les actions qui n'ont pas été choisies depuis longtemps. Par exemple, $\varepsilon\sqrt{\delta n_{s,a}}$ où $\delta n_{s,a}$ est le nombre d'itérations depuis la dernière fois que l'action a a été sélectionnée dans l'état s. ($\varepsilon < 1$).
   - *Méthode de l'estimation d'incertitude*. Le bonus est $\frac{c}{n_{s,a}}$, où c est une constante, et $n_{s,a}$ est le nombre de fois où l'action a a été utilisée dans l'état s.

Ces différentes méthodes impliquent des paramètres arbitraires, qui doivent être choisis par l'utilisateur afin d'améliorer l'efficacité de l'apprentissage. Cela dépend de chaque problème. En revanche, nous verrons que la théorie indique que, si l'hypothèse de Markov est vérifiée et si tous les états sont parcourus un nombre infini de fois, alors les valeurs des fonctions d'utilité convergent vers leur valeur optimale. Toutes les méthodes présentées ci-dessus sont donc *a priori* correctes.

Des travaux récents [ACBF02] ont relancé une grande activité de recherche sur le problème des bandits et sur son application à des problèmes d'apprentissage et à des problèmes de jeu en particulier. Nous en donnons un bref aperçu dans la section 10.

2. Si tout est connu : l'utilité de la fonction d'utilité

L'idée profonde sous-jacente à l'apprentissage par renforcement est de permettre à l'agent d'optimiser sa conduite sur le long terme (celle qui maximise son gain cumulé) sur la base de décisions locales ne nécessitant pas de recherche en avant. Il faut donc que l'information disponible localement, au moment et au lieu où se trouve l'agent, reflète l'espérance de gain à long terme. Cette information locale, résumant les potentialités à long terme des actions possibles, est traduite par une valeur numérique appelée *utilité*. Plus généralement, on parle de *fonction d'utilité* pour désigner la fonction associant à chaque *état*, ou à chaque paire (*état, action*), sa valeur d'utilité. Nous étudierons deux fonctions d'utilité particulières : la première associe à chaque état s l'espérance de gain à partir de cet état si l'on suit la politique π :

$$V^\pi(s) = E_\pi\{R_t | s_t = s\} \tag{16.4}$$

La seconde associe à chaque couple (*état, action*) (s,a) l'espérance de gain en partant de l'état s, en effectuant l'action a, puis en suivant la politique π :

$$Q^\pi(s,a) = E_\pi\{R_t | s_t = s, a_t = a\} \tag{16.5}$$

Copyright © 2010 Eyrolles.

Dans le cas d'un gain cumulé avec intérêt et horizon infini, les deux formules deviennent :

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k}\, r_{t+k+1} \mid s_t = s\right\} \\ Q^{\pi}(s,a) &= E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k}\, r_{t+k+1} \mid s_t = s, a_t = a\right\} \end{aligned}$$

Nous venons de définir l'espérance de gain, donc l'utilité, en fonction d'une politique donnée. Il se trouve que l'on peut définir une relation d'ordre partiel sur les politiques en fonction des valeurs d'utilité associées[4]. Plus précisément :

Définition 16.1 (Ordre sur les politiques)

Une politique π est dite supérieure *à une autre politique π' si et seulement si l'espérance de gain suivant π est supérieure ou égale à l'espérance de gain suivant π' pour tous les états $s \in \mathcal{E}$. En d'autres termes, $\pi \geq \pi'$ ssi $V^{\pi}(s) \geq V^{\pi'}(s), \forall s \in \mathcal{E}$.*

Cet ordre partiel permet de définir le concept de *politique optimale* :

Définition 16.2 (Politique optimale π^*)

Une politique optimale *est une politique supérieure ou égale à toutes les autres politiques. Dans le cas des processus markoviens, il en existe toujours une. On note π^* cette politique optimale, ou les politiques optimales s'il y en a plusieurs.*

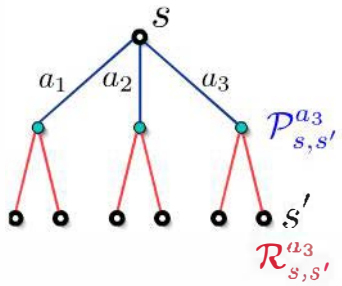
On a alors les fonctions d'utilité correspondantes :

$$\begin{aligned} V^{*}(s) &= \max_{\pi} V^{\pi}(s), \qquad \forall s \in \mathcal{E} \\ Q^{*}(s,a) &= \max_{\pi} Q^{\pi}(s,a), \qquad \forall s \in \mathcal{E} \text{ et } \forall a \in \mathcal{A} \end{aligned} \tag{16.6}$$

En les utilisant, il devient facile d'obtenir un comportement optimal.

Supposons que l'agent dispose des valeurs $V^*(s), \forall s \in \mathcal{E}$. Placé dans l'état s, l'agent n'a qu'à faire une recherche en avant à un pas pour déterminer l'action à prendre. Pour chaque action a disponible, il examine les états accessibles s' (il peut y en avoir plusieurs pour une même action si l'environnement est non déterministe, chacun avec une certaine probabilité d'obtention) et note pour chacun d'eux sa valeur $V^*(s')$. L'action a^* associée avec la meilleure espérance de gain est l'action qu'il faut choisir :

$$a_t^* = \underset{a \in \mathcal{A}}{\text{ArgMax}} \sum_{s'} \mathcal{P}_{ss'}^{a}\left[\mathcal{R}_{ss'}^{a} + \gamma V^*(s')\right] \tag{16.7}$$



TAB. 16.1: *Passage de s à s' en choisissant une action a (ici a_3).*

[4] Plus précisément, cette relation d'ordre existe dans le cas des *processus markoviens*, pour lesquels on suppose que la donnée d'un état suffit à résumer toute l'histoire antérieure du système (voir le chapitre 12). En d'autres termes, les décisions peuvent alors être prises sans connaître l'histoire passée du système. Cette hypothèse est toujours citée dans la théorie de l'apprentissage par renforcement.

Copyright © 2010 Eyrolles.

Supposons maintenant que l'agent dispose des valeurs $Q^*(s,a), \forall s \in \mathcal{E}$ et $\forall a \in \mathcal{A}$. Dans ce cas, la détermination de la conduite optimale dans la situation courante s est encore plus simple, il suffit de choisir l'action a maximisant $Q^*(s,a)$:

$$a_l^* = \underset{a \in \mathcal{A}}{\text{ArgMax}}\ Q^*(s,a) \tag{16.8}$$

Dans les deux cas, la décision peut donc être obtenue très facilement. Si l'on dispose des valeurs d'utilité $V^*(s)$, il faut effectuer une recherche en avant d'un pas, tandis que cela n'est même pas nécessaire dans le cas où l'on dispose des valeurs d'utilité $Q^*(s,a)$. Pourtant, la différence entre les deux méthodes est plus profonde qu'il n'y paraît. L'équation (16.7) nécessite en effet la connaissance de l'environnement sous la forme des probabilités de transition $\mathcal{P}^a_{ss'}$, et des renforcements associés $\mathcal{R}^a_{ss'}$ reçus lorsque l'agent passe de l'état s à l'état s' sous l'action a. Cette connaissance n'est pas nécessaire dans le cas de l'équation (16.8). Bien sûr, cette énorme simplification se paie par le fait qu'il faut maintenant travailler dans l'espace produit $\mathcal{E} \times \mathcal{A}$ au lieu de l'espace $\mathcal{E}$.

3. L'apprentissage des fonctions d'utilité quand l'environnement est connu

Dans cette section, nous allons supposer que l'agent a une connaissance de l'environnement sous la forme des probabilités de transition $\mathcal{P}^a_{ss'}$ et des renforcements associés $\mathcal{R}^a_{ss'}$. En revanche, il ne connaît pas les fonctions d'utilité. Cela signifie qu'il a une connaissance locale de son environnement, mais qu'il n'a pas d'information sur l'impact à long terme des décisions qu'il pourrait prendre. Comment peut-il alors les apprendre?

Nous allons avoir deux problèmes à résoudre. D'une part, nous l'avons vu, une fonction d'utilité dépend de la politique suivie. Il nous faut donc voir comment apprendre ces fonctions dans le cadre d'une politique donnée. D'autre part, nous cherchons à obtenir, ou du moins à approcher, une politique optimale. Il nous faut donc voir comment passer de fonctions d'utilité associées à des politiques *a priori* sous-optimales à la détermination d'une politique optimale. Nous allons étudier ces deux problèmes l'un après l'autre.

3.1 L'évaluation d'une politique par propagation locale d'information

Dans un premier temps, nous cherchons à évaluer une politique π en déterminant les valeurs $V^\pi(s), \forall s \in \mathcal{E}$.

Intuitivement, une approche simple serait de placer l'agent en chaque état s et de lui faire suivre à partir de là la politique π, éventuellement de nombreuses fois pour moyenner sur toutes les trajectoires possibles si le monde est non déterministe. Il suffirait alors de calculer la moyenne des gains cumulés obtenus pour connaître une estimation de l'espérance de gains à partir de s. Nous allons développer la formule correspondante dans le cas du gain cumulé avec intérêt et horizon infini, mais ce qui importe est l'idée générale qui se transfère sans problème à d'autres

Copyright © 2010 Eyrolles.

formules de gains.

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\left\{ R_t \mid s_t = s \right\} \\ &= E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1} \;\middle|\; s_t = s \right\} \\ &= E_{\pi}\left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+2} \;\middle|\; s_t = s \right\} \\ &= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma \, E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+2} \;\middle|\; s_{t+1} = s' \right\} \right] \\ &= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma \, V^{\pi}(s') \right] \end{aligned} \tag{16.9}$$

Ce résultat est remarquable car il met en valeur le fait que l'on peut ramener un calcul de gain prenant en compte toute une trajectoire (ou un ensemble de trajectoires) à partir de s à un calcul de gain s'appuyant sur les estimations $V^{\pi}(s')$ des états s' accessibles à partir de l'état courant s. Il permet d'exploiter ainsi une dépendance ou corrélation entre les espérances de gain des états. L'utilité d'un état suivant la politique π est égale à une somme pondérée suivant les probabilités de transition aux états successeurs des utilités de ces états successeurs plus le signal de renforcement reçu lors de la transition de s à s'. C'est l'*équation de Bellman*[5] pour V^{π}.

Il se trouve que l'on peut démontrer que la valeur $V^{\pi}(s)$ est l'unique solution de l'équation de Bellman. Par ailleurs, l'équation de Bellman peut être utilisée dans une procédure d'approximation itérative (voir algorithme 27) :

$$\begin{aligned} V_{k+1}(s) &= E_{\pi}\{ r_{t+1} + \gamma \, V_k(s_{t+1}) \mid s_t = s \} \\ &= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma \, V_k(s') \right] \end{aligned} \tag{16.10}$$

pour tout $s \in \mathcal{E}$. Il est clair que $V_k = V^{\pi}$ est un point fixe de cette règle itérative. Par ailleurs, il est possible de montrer que la séquence $\{V_k\}$ converge vers V^{π} lorsque $k \to \infty$ si $\gamma < 1$ ou si les gains sont calculés sur un horizon limité. Intuitivement, cette convergence se justifie par le fait que chaque mise à jour d'une valeur $V_k(s)$ s'appuie sur d'autres estimations $V_k(s')$, mais aussi sur le signal de renforcement observé r_{t+1}. Il y a donc bien un gain d'information sur l'environnement à chaque itération.

3.2 Un théorème conduisant à l'amélioration de politique

Nous avons vu comment il était possible en principe d'approcher l'espérance de gain en chaque état *pour une politique donnée*. Cela permet à l'agent de prendre des décisions rapides sur la base d'informations locales. Cependant, l'agent cherche aussi à améliorer sa politique, et éventuellement à trouver la politique optimale pour un environnement donné. Il existe un théorème qui arrange bien les choses car il relie les valeurs d'utilité locales $V^{\pi}(s)$ et les valeurs relatives de politiques entre elles. Il va donc être possible de s'appuyer sur les premières pour savoir dans quelle direction modifier les secondes afin de les améliorer. Nous ne saurions trop souligner l'importance de ce théorème, qui fonde la plupart des méthodes d'apprentissage par renforcement.

[5] Cette équation fait intervenir le principe de Bellman, qui est à la base des méthodes de programmation dynamique. Ce principe a été appliqué par exemple au chapitre 12 dans l'algorithme de Viterbi.

Copyright © 2010 Eyrolles.

Algorithme 27 : Algorithme d'évaluation itérative d'une politique

Données : la politique π à évaluer
Résultat : $V \approx V^{\pi}$
début
  Initialisation : $V(s) = 0$, pour tous $s \in \mathcal{E}^{+}$ (les états accessibles depuis s)
  répéter
    $\Delta \leftarrow 0$
    pour tous les $s \in \mathcal{E}$ **faire**
      $v \leftarrow V(s)$
      $V(s) \leftarrow \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} [\mathcal{R}^{a}_{ss'} + \gamma V(s')]$
      $\Delta \leftarrow max(\Delta, |v - V(s)|)$
    fin
  jusqu'à $\Delta < \theta$ (un petit nombre réel positif)
fin

Théorème 16.1 (Relation d'ordre sur les politiques)

Soient π et π' deux politiques déterministes, telles que, pour tout état $s \in \mathcal{E}$:

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s) \tag{16.11}$$

Alors la politique π' doit être au moins aussi bonne que la politique π, ce qui signifie que, pour tout état $s \in \mathcal{E}$:

$$V^{\pi'}(s) \geq V^{\pi}(s) \tag{16.12}$$

De plus, si l'inégalité large dans l'équation (16.11) est remplacée par un inégalité stricte, alors il en est de même dans l'équation (16.12).

Ce théorème indique donc que si l'on trouve une modification π' de la politique π qui vérifie l'équation (16.12), alors on obtient une meilleure politique. Concrètement, cela peut se traduire par une procédure d'amélioration de politique. Prenons en effet une politique déterministe π, et une autre politique déterministe π' identique à π, sauf pour un état s pour lequel : $\pi'(s) = a \neq \pi(s)$. Alors, pour tous les états autres que s, l'équation (16.11) est vérifiée. Si de plus nous avons $Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s)$, alors la nouvelle politique π' est meilleure que π.

Il est facile d'étendre cette procédure à une procédure qui améliore une politique π sur tous les états s. Pour cela, il suffit de prendre une politique π' qui pour chaque état s sélectionne l'action qui semble la meilleure selon la fonction d'utilité $Q^{\pi}(s,a)$:

$$\begin{aligned} \pi'(s) &= \underset{a \in \mathcal{A}}{\text{ArgMax}}\ Q^{\pi}(s,a) \\ &= \underset{a \in \mathcal{A}}{\text{ArgMax}}\ E\{ r + \gamma V^{\pi}(s') \} \\ &= \underset{a \in \mathcal{A}}{\text{ArgMax}} \sum_{s'} \mathcal{P}^{a}_{ss'} [\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')] \end{aligned} \tag{16.13}$$

où r est le renforcement obtenu en passant de l'état s à l'état s' par l'action a.

Cette procédure de type gradient[6] choisit donc pour chaque état l'action qui semble la meilleure à un pas en avant, selon la fonction d'utilité V^{π} associée à la politique π. Il est facile de montrer

[6] Voir le chapitre 3.

Copyright © 2010 Eyrolles.

que si la nouvelle politique π' choisie selon cette procédure n'est pas meilleure que la politique π, c'est que l'on a : $V^{\pi'} = V^{\pi} = V^*$, donc que l'on a atteint la politique optimale.

Tout ce qui précède a été obtenu pour le cas de politiques déterministes, mais peut être étendu sans problème au cas des politiques non déterministes. Nous reportons le lecteur intéressé à [SB98] par exemple.

3.3 Processus itératif d'amélioration de politique

La section précédente a montré comment passer d'une politique π décrite par sa fonction d'utilité V^{π} à une meilleure politique. Il est facile de voir comment on peut itérer ce processus. L'idée est d'alterner les phases d'évaluation de politique (section 3.1) avec les phases d'amélioration de politique (section 3.2). Nous pouvons alors obtenir une séquence de politiques en amélioration monotone :

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{A} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{A} \pi_2 \xrightarrow{E} \ldots \xrightarrow{A} \pi^* \xrightarrow{E} V^*$$

où $\xrightarrow{E}$ dénote une phase d'*évaluation* et $\xrightarrow{A}$ une phase d'*amélioration*.

Cette procédure itérative converge en un nombre fini d'itérations vers la politique optimale si la politique est représentée par un processus markovien à nombre d'états fini[7]. Par ailleurs, la convergence observée empiriquement est généralement très rapide. Cependant, la phase d'évaluation de politique est très coûteuse puisqu'elle requiert de nombreux passages sur chaque état $s \in \mathcal{E}$ qui peuvent être très nombreux. Mais est-il possible de faire l'économie de la détermination précise des valeurs d'utilité relative à chaque politique π et π' que semble requérir théorème 16.1 ?

Il est heureusement envisageable de ne pas attendre la convergence de la phase d'*évaluation de politique* avant de lancer une nouvelle phase d'*amélioration de politique*. De fait, on dispose de théorèmes démontrant qu'il y aura convergence ultime sur la politique optimale même si l'alternance entre évaluation et amélioration se fait sur des granularités beaucoup plus fines que le processus alternatif décrit plus haut. Par exemple, on peut alterner une seule itération du processus d'évaluation de politique entre chaque phase d'amélioration, et dans ce cas, on obtient l'*algorithme d'itération de valeur* :

$$\begin{aligned} V_{k+1}(s) &= \max_{a \in \mathcal{A}} E\{r + \gamma V_k(s')\} \\ &= \max_{a \in \mathcal{A}} \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V_k(s')] \end{aligned} \tag{16.14}$$

Cette équation de mise à jour à réaliser pour tout $s \in \mathcal{E}$, est en fait une version itérative de l'équation de Bellman (16.9) qui suppose que cette règle itérative converge vers un point fixe correspondant à la fonction d'utilité optimale $V^*(s)$. On notera que l'équation 16.14 suppose connu le modèle du monde (c'est-à-dire les fonctions de transition $\mathcal{P}^a_{ss'}$ et de renforcement $\mathcal{R}^a_{ss'}$).

L'action choisie à chaque instant est alors l'action maximisant le gain indiqué par la fonction d'utilité $V_k(s)$ courante :

$$\pi(s) = \operatorname*{ArgMax}_{a \in \mathcal{A}} \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V_k(s')]$$

L'algorithme d'itération de valeur converge lorsque l'on entrelace chaque phase d'évaluation avec une phase d'amélioration. Cependant, on obtient généralement une meilleure vitesse de convergence lorsque plusieurs phases d'évaluation ont lieu entre chaque phase d'amélioration.

[7] Dans ce cas, en effet, le nombre de politiques est fini.

Copyright © 2010 Eyrolles.

Tel quel, cet algorithme requiert néanmoins que tous les états soient visités à chaque phase. Dans la plupart des applications intéressantes, cela est irréaliste[8]. Heureusement, il est possible d'utiliser une mise à jour asynchrone des évaluations, visitant certains états plusieurs fois tandis que d'autres ne le sont qu'une fois. La convergence requiert cependant qu'aucun état ne cesse d'être visité. Il s'agit là d'une condition de convergence universelle pour les règles de mise à jour par propagation locale des informations. Cette remarque ouvre la possibilité d'apprendre en ligne en cours d'expérience, en mettant à jour les valeurs d'utilité des états rencontrés.

4. Sans modèle du monde : la méthode de Monte-Carlo

Les méthodes précédentes supposent que l'on connaisse l'environnement à travers des probabilités de transition d'états et de renforcement : $\mathcal{P}^a_{ss'}$ et $\mathcal{R}^a_{ss'}$. Si ce n'est pas le cas, le principe général est d'estimer ces valeurs par un échantillonnage obtenu en ligne, au cours des actions.

Les méthodes de Monte-Carlo supposent l'observation d'*épisodes* complets[9] et le calcul de gains moyens. Leur principe est d'estimer directement les valeurs d'utilité en calculant des moyennes de gain pour chaque état ou chaque paire (*état, action*) en fonction des expériences de l'agent.

Supposons que l'on cherche la valeur $V^\pi(s)$ pour l'état s et la politique π. Il suffit de considérer toutes les séquences d'états qui ont suivi l'état s et de calculer la moyenne des gains observés. On réalise alors empiriquement l'approximation de l'espérance : $V^\pi(s) = E_\pi\{R_t | s_t = s\}$. Sans entrer dans les détails, notons que cette méthode ne s'appuie plus sur la corrélation entre les états mise en évidence par l'équation de Bellman. Cette perte d'information nuit à la convergence du processus, mais permet en revanche de ne pas avoir à échantillonner uniformément tous les états et de s'apesantir sur les états les plus importants en ignorant les états inintéressants.

Afin d'améliorer la politique suivie, il est possible d'utiliser le même principe d'alternance de phase d'évaluation (selon la méthode ci-dessus) et de phase d'amélioration. Cependant, comme l'environnement est inconnu, il n'est plus possible de travailler avec l'équation (16.14). Il faut avoir recours à une procédure itérative portant sur les fonctions d'utilité $Q^\pi(s,a)$. On aura donc :

$$\pi(s) = \underset{a \in \mathcal{A}}{\text{ArgMax}}\ Q(s,a) \tag{16.15}$$

et une procédure d'amélioration itérative de politique :

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}(s, \underset{a \in \mathcal{A}}{\text{ArgMax}}\ Q^{\pi_k}(s,a)) \\ &= \max_{a \in \mathcal{A}} Q^{\pi_k}(s,a) \\ &\geq Q^{\pi_k}(s, \pi_k(s)) = V^{\pi_k}(s) \end{aligned} \tag{16.16}$$

Pour plus de détails sur les conditions d'application de cette procédure, nous reportons le lecteur à [SB98].

[8] Par exemple, le backgammon est associé à un espace d'environ 10^{20} états.

[9] Des séquences d'états et d'actions s'arrêtant dans des états terminaux.

Copyright © 2010 Eyrolles.

5. La méthode des différences temporelles

La méthode des différences temporelles (*temporal-difference learning*) combine des idées des méthodes issues de la programmation dynamique et des méthodes de Monte-Carlo. Comme les premières, elles prennent en compte les corrélations entre les états pour mettre à jour leur évaluation. Comme les secondes, elles n'ont pas besoin d'une connaissance *a priori* sur l'environnement. La méthode des différences temporelles s'appuie également sur une alternance de *phases d'évaluation* et de *phases d'amélioration*. Décrivons tour à tour les deux phases.

5.1 L'évaluation suivant la méthode des différences temporelles

Nous avons vu que l'estimation de la valeur d'utilité selon la méthode de Monte-Carlo repose sur une approximation de la formule :

$$V^{\pi}(s) \;=\; E_{\pi}\{R_t \,|\, s_t = s\}$$

qui se traduit par une opération itérative de mise à jour :

$$\boxed{V(s) \;\leftarrow\; V(s) + \alpha\,[R_t - V(s)]}$$

où R_t est le gain mesuré après l'instant t en partant de l'état s, et α est un paramètre constant. Cela se démontre par la série d'égalités suivante. On suppose que l'espérance définissant la valeur de $V^{\pi}(s)$ est estimée par moyennage sur $m+1$ épisodes observés après passage par l'état s :

$$\begin{aligned}
V^{\pi}_{m+1}(s) &= \frac{1}{m+1}\sum_{i=1}^{m+1} R_i \\
&= \frac{1}{m+1}\left(R_{m+1} + \sum_{i=1}^{m} R_i\right) \\
&= \frac{1}{m+1}\left(R_{m+1} + m\,V^{\pi}_{m}(s) + V^{\pi}_{m}(s) - V^{\pi}_{m}(s)\right) \\
&= \frac{1}{m+1}\left(R_{m+1} + (m+1)\,V^{\pi}_{m}(s) - V^{\pi}_{m}(s)\right) \\
&= V^{\pi}_{m}(s) + \frac{1}{m+1}\left[R_{m+1} - V^{\pi}_{m}(s)\right]
\end{aligned}$$

L'intérêt de cette méthode de mise à jour incrémentale est qu'elle ne nécessite que la mémorisation de $V^{\pi}_{m}(s)$ et m et un calcul simple après chaque observation d'un épisode. On peut généraliser cette procédure à la forme générale suivante :

$$NouvelleEstimation \;\leftarrow\; AncienneEstimation + \alpha\,[Cible - AncienneEstimation] \quad (16.17)$$

dans laquelle $[Cible - AncienneEstimation]$ est une *erreur* sur l'estimation courante qui est réduite en allant d'un pas vers la *Cible*. Cette cible indique donc la direction dans laquelle aller. Elle peut être sujette à des variations aléatoires. Le pas α peut être une constante ou une variable décroissant lentement, ce qui est souvent utilisé pour stabiliser la valeur estimée.

La méthode de la programmation dynamique (voir l'équation (16.13)) utilise la formule :

$$V^{\pi}(s) \;=\; E_{\pi}\{r + \gamma\,V^{\pi}(s')\}$$

Copyright © 2010 Eyrolles.

où r est le renforcement obtenu en passant de l'état s à l'état s'.

La méthode des différences temporelles est basée sur une mise à jour au coup par coup de cette estimation :

$$V_{t+1}^{\pi}(s) \leftarrow V_t^{\pi}(s) + \alpha \left[r + \gamma V_t^{\pi}(s') - V_t^{\pi}(s) \right] \tag{16.18}$$

On parle dans ce cas de *retour échantillonné* (*sample backup*) parce que la mise à jour s'effectue à partir d'observations individuelles obtenues durant l'action. Par contraste, les méthodes de *retour complet* (*full backup*) s'appuient sur une distribution complète des successeurs possibles. C'est le cas de l'équation d'estimation utilisée dans la programmation dynamique. La possibilité d'utiliser un retour échantillonné pour estimer incrémentalement les valeurs d'utilité est cruciale pour la faisabilité d'un apprentissage en ligne durant l'expérience de l'agent dans le monde. Cela conduit à l'algorithme 28.

Algorithme 28 : Évaluation par la méthode des différences temporelles

début
  Initialiser $V(s)$ arbitrairement, et π à la politique à évaluer.
  répéter
    pour chaque *épisode* **faire**
      Initialiser s
      répéter
        pour chaque *étape de l'épisode* **faire**
          $a \leftarrow$ l'action donnée par π pour l'état s
          Exécuter l'action a ; recevoir le renforcement r ; et mesurer l'état suivant s'
          $V^{\pi}(s) \leftarrow V(s) + \alpha \left[r + \gamma V^{\pi}(s') - V^{\pi}(s) \right]$
          $s \leftarrow s'$
        fin
      jusqu'à s *est terminal*
    fin
  jusqu'à *critère d'arrêt* (convergence suffisante)
fin

Cet algorithme offre la possibilité de mettre à jour les valeurs d'utilité tout en agissant dans le monde et en tirant parti des observations ainsi réalisées. Il faut cependant, pour en assurer la convergence théorique, que tous les états soient visités infiniment souvent sur un temps infini. Les propriétés et conditions de cette convergence sont encore du domaine de la recherche (pour les détails, on peut se se reporter à [SB98]).

5.2 L'amélioration de politique avec les différences temporelles

Comme pour les méthodes de programmation dynamique et de Monte-Carlo, il faut préciser comment passer de l'évaluation de la politique à l'amélioration de la politique. Nous allons considérer deux approches typiques dont les principes sont de portée générale. La première est fondée sur une approche similaire à celles évoquées plus haut reposant sur une alternance de phases d'évaluation de politique et de phases d'amélioration. La seconde court-circuite d'une certaine manière l'idée même de politique.

Copyright © 2010 Eyrolles.

5.3 SARSA : une méthode d'amélioration « sur politique »

L'idée de base de l'algorithme SARSA ([SB98] section 6.4) est la suivante : à chaque choix d'action dans l'état courant s_t, l'agent suit approximativement (nous verrons pourquoi et comment) la politique courante π. Après observation du nouvel état courant s' et du renforcement reçu r, il met à jour la valeur d'utilité de la situation rencontrée et est alors prêt à choisir l'action suivante. Il s'agit donc d'une méthode itérative alternant évaluation et amélioration. On qualifie ce genre d'approche de méthode *sur politique* (*on-policy*) car elle suppose que l'agent s'inspire à chaque instant de la politique courante π pour le choix de ses actions.

Afin de s'affranchir de la nécessité de connaître un modèle de l'environnement, l'approche SARSA estime la fonction d'utilité $Q^\pi(s,a)$ pour la politique courante et pour toutes les paires (*état, action*). La méthode des différences temporelles conduit alors à la mise à jour suivante :

$$Q^\pi(s,a) \leftarrow Q^\pi(s,a) + \alpha\left[r + \gamma Q^\pi(s',a') - Q^\pi(s,a)\right] \tag{16.19}$$

Cette mise à jour est effectuée après chaque transition partant d'un état s non terminal. Si s' est un état terminal, alors $Q(s',a')$ est défini comme égal à zéro. Comme cette règle de mise à jour utilise les valeurs de s, a, r, s', a', elle a été baptisée algorithme SARSA.

L'amélioration de politique se fait alors grâce à la procédure suivante. À chaque choix d'action, l'agent utilise les valeurs estimées $Q^\pi(s,a)$ pour sélectionner l'action à exécuter. Pour négocier le compromis exploitation contre exploration, l'agent utilise une procédure dite *ε-gloutonne* c'est-à-dire une méthode de gradient bruité : l'agent sélectionne en général l'action a associée à la meilleure valeur d'utilité $Q^\pi(s,a)$. De temps en temps cependant, avec une probabilité ε, il sélectionne aléatoirement une autre action, ce qui permet ainsi d'explorer les politiques possibles.

Algorithme 29 : Algorithme SARSA d'amélioration de politique

début
  Initialiser $Q(s,a)$ arbitrairement.
  répéter
    pour chaque *épisode* **faire**
      Initialiser s
      Choisir l'action a par une procédure ε-gloutonne dérivée des valeurs de $Q^\pi(s,a)$
      répéter
        pour chaque *étape de l'épisode* **faire**
          Exécuter l'action a ; recevoir le renforcement r ; et mesurer l'état suivant s'
          Choisir l'action a' à partir de s' en utilisant une procédure ε-gloutonne dérivée des valeurs de $Q^\pi(s,a)$
          $Q^\pi(s,a) \leftarrow Q^\pi(s,a) + \alpha\left[r + \gamma Q^\pi(s',a') - Q^\pi(s,a)\right]$
          $s \leftarrow s'$; $a \leftarrow a'$
        fin
      jusqu'à *s est terminal*
    fin
  jusqu'à *critère d'arrêt* (convergence suffisante)
fin

La procédure SARSA converge avec une probabilité de 1 vers une politique optimale si toutes les paires (*état, action*) sont visitées infiniment souvent sur une durée infinie et si le coefficient ε est bien réglé (par exemple, en posant $\varepsilon = \frac{1}{t}$).

Copyright © 2010 Eyrolles.

5.4 Le Q – *learning* : une méthode d'amélioration « hors politique »

L'algorithme du *Q-learning* [Wat89] utilise la dépendance explicite de la fonction d'utilité $Q(s,a)$ sur les actions, pour à la fois mettre à jour ces valeurs, donc converger vers la fonction optimale $Q^*(s,a)$, et aussi pour déterminer l'action à prendre dans la situation courante. Dans cette technique, la mise à jour des valeurs d'utilité se fait selon l'équation suivante :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a' \in \mathcal{A}} Q(s',a') - Q(s,a) \right] \tag{16.20}$$

tandis que l'action a sélectionnée dans l'état courant s est déterminée par une politique du genre ε-gloutonne assurant le compromis exploitation contre exploration. Il faut noter qu'il n'existe pas pour le moment de méthode générique pour résoudre ce compromis dans le cas du *Q-learning* et que la pratique est d'utiliser des règles *ad hoc* déterminées empiriquement pour chaque situation.

Comme pour toutes les procédures de mise à jour stochastiques dans des processus markoviens, la convergence vers la valeur optimale $Q^*(s,a)$ nécessite que chaque état soit visité infiniment souvent et que le paramètre α décroisse de manière appropriée.

Il est intéressant de noter que la convergence de la méthode est assurée quelle que soit la manière dont les actions sont sélectionnées à chaque instant, pourvu que tous les états soient visités infiniment souvent. C'est pourquoi on parle de méthode *hors politique* (*off-policy*). En revanche, les vitesses de convergence observées sont souvent plus lentes que celles d'autres méthodes. Cela semble le prix à payer pour l'utilisation d'une méthode qui a révolutionné l'apprentissage par renforcement grâce à sa très grande aisance d'emploi et à la facilité d'établir des preuves à son propos.

5.5 TD(λ) : les méthodes de différences temporelles à plusieurs pas

Pour présenter les méthodes TD(λ), le mieux est de présenter d'abord le cas particulier TD(0). Cette méthode effectue ses mises à jour des valeurs d'utilité en ne regardant qu'un seul pas en avant. Même si cela suffit à garantir la convergence selon les conditions déjà soulignées, celle-ci peut être lente. Les méthodes TD(λ) la généralisent en effectuant des mises à jour selon un horizon plus lointain.

L'idée est la suivante : lorsque le renforcement r_{t+1} entre l'état courant s_t et le suivant s_{t+1} atteint avec l'action a_t a été mesuré, on peut mettre à jour la valeur $V(s_t)$ mais aussi, par ricochet, les valeurs des états antérieurement visités $V(s_{t-i})$. En général, on ne met pas à jour uniformément les valeurs des états visités dans le passé, mais on utilise une pondération diminuant l'effet de la mise à jour au fur et à mesure que l'on remonte dans la séquence des états.

La formule générale de mise à jour s'écrit :

$$V(u) \leftarrow V(u) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right] e(u) \tag{16.21}$$

où u est n'importe quel état pour lequel $e(u) \neq 0$. La *trace d'éligibilité* $e(u)$ détermine ainsi les états sujets à mise à jour. Une expression habituelle de trace d'éligibilité est :

$$e(s) = \sum_{k=1}^{t} (\lambda\gamma)^{t-k} \delta_{s,s_k}, \text{ où } \delta_{s,s_k} = \begin{cases} 1 & \text{si } s = s_k \\ 0 & \text{sinon} \end{cases} \tag{16.22}$$

avec $0 \leq \lambda < 1$ pour assurer la convergence.

L'éligibilité d'un état s définit ainsi le degré auquel il doit être sensible à la mise à jour actuelle. Quand $\lambda = 0$, on retrouve la méthode TD(0). Quand $\lambda = 1$, on retrouve à peu près la méthode

Copyright © 2010 Eyrolles.

de Monte-Carlo, c'est-à-dire que l'on met à jour chaque état à raison du nombre de fois où il a été visité durant l'épisode servant de base à la mise à jour.

Il est aussi possible de modifier en ligne la trace d'éligibilité :

$$e(s) \leftarrow \begin{cases} \gamma\lambda e(s) + 1 & \text{si } s = \text{l'état courant} \\ \gamma\lambda e(s) & \text{sinon} \end{cases}$$

L'algorithme TD(λ) est évidemment nettement plus coûteux à mettre en œuvre que TD(0), mais il converge en général beaucoup plus rapidement pour des valeurs de λ assez grandes (mais toujours < 1). Plusieurs travaux ont été consacrés à l'analyse des traces d'éligibilité. Elles peuvent être appliquées aux algorithmes de Q-learning, conduisant aux méthodes appelées Q(λ)-learning.

6. La généralisation dans l'apprentissage par renforcement

6.1 Le problème

Jusqu'à présent nous avons implicitement supposé que les états et les actions étaient énumérables et qu'il était possible de représenter les différentes fonctions (par exemple la fonction d'utilité V) par des tables de valeurs. Or, excepté pour des environnements très simples, cela implique des tailles de mémoire irréalistes. De plus, dans de nombreuses applications, l'espace des états, et parfois aussi celui des actions, est continu, rendant impossible l'usage direct de tables. Finalement, les algorithmes de mise à jour de tables font une utilisation assez peu efficace de l'information glanée en cours d'expérience dans la mesure où, alors qu'il est fréquent que des états similaires aient des valeurs d'utilité et des actions optimales attachées similaires, la modification d'une valeur dans la table n'entraîne pas celle des autres. Dans ces conditions, l'utilisation de tables semble peu opportune et il faut chercher une représentation des politiques à la fois plus compacte et permettant une utilisation plus efficace de l'information.

Une idée assez immédiate est d'avoir recours à des techniques de généralisation dans l'espace $\mathcal{E}$ des états, et éventuellement dans l'espace $\mathcal{A}$ des actions. De cette manière, lorsqu'une situation impliquant un état et une action donnés est observée, l'information obtenue à cette occasion peut être transférée par généralisation aux états et actions similaires. Il devient alors envisageable d'appliquer l'ensemble des techniques d'induction supervisée à ce problème.

Les fonctions que l'on peut vouloir apprendre incluent :

- l'apprentissage direct d'une politique $\pi : \mathcal{E} \rightarrow \mathcal{A}$;
- l'apprentissage de la fonction d'utilité $V : \mathcal{E} \rightarrow \mathbb{R}$;
- l'apprentissage de la fonction d'utilité $Q : \mathcal{E} \times \mathcal{A} ; \rightarrow \mathbb{R}$
- l'apprentissage de l'environnement :
  - la fonction de transition :
    - déterministe : $\mathcal{E} \times \mathcal{A} \rightarrow \mathcal{E}$,
    - ou non déterministe : $\mathcal{E} \times \mathcal{A} \times \mathcal{E} \rightarrow [0,1]$;
  - la fonction de renforcement : $\mathcal{E} \times \mathcal{E} \rightarrow \mathbb{R}$.

Certaines de ces fonctions, à savoir la fonction de transition et la fonction de récompense, sont du ressort direct de nombreuses méthodes classiques d'apprentissage supervisé, telles des méthodes connexionnistes (chapitre 10), l'induction d'arbres de décision (chapitre 13) ou des méthodes par k-plus proches voisins (chapitre 15). Il est en effet facile d'obtenir de nombreux

Copyright © 2010 Eyrolles.

exemples, et la fonction cible est généralement fixe ou peu changeante (sauf si l'environnement est gravement perturbé). D'autres posent plus de problèmes, comme la fonction de politique car il est difficile d'obtenir des exemples à partir desquels apprendre. Dans tous les cas, il est souhaitable de disposer de méthodes d'apprentissage qui permettent la prise en compte incrémentale des exemples au fur et à mesure que l'exploration du monde les rend disponibles et qui ont la capacité de suivre une fonction cible changeante.

6.2 Généralisation par approximation de la fonction d'utilité

Nous supposons dans un premier temps que le problème de prédiction porte sur la fonction d'utilité sur les états : V. L'idée est d'affiner une estimation de la fonction optimale V^* par des itérations successives d'une estimation V_t au temps t en fonction de l'expérience courante de l'agent dans le monde. On suppose ici que les fonctions estimées V_t ne sont plus des tables de valeurs mais font partie d'une classe de fonctions que l'on supposera paramétrée par un vecteur $\boldsymbol{\theta}$. Cela signifie que la fonction estimée V_t dépend entièrement de $\boldsymbol{\theta}_t$ variant d'un instant t au suivant. L'une des nouveautés par rapport à un problème de régression classique est que l'apprentissage s'opère ici *en ligne* avec des exemples issus d'une distribution dépendant de l'action de l'agent et non d'une distribution d'exemples indépendamment et identiquement distribués (i.i.d.) comme c'est le cas général en induction.

Typiquement, l'estimation V_t peut être réalisée par un réseau connexionniste dont les poids sont réglés en cours d'apprentissage, ou par un arbre de décision. Dans ce dernier cas, le vecteur $\boldsymbol{\theta}_t$ décrit les branchements de l'arbre appris.

Les exemples servant à l'apprentissage dépendent des méthodes de prédiction de gain utilisées. Par exemple, il est courant de prendre l'estimation de gain calculée selon la méthode des différences temporelles : $s \mapsto r + \gamma V_t(s')$.

L'utilisation de méthodes de généralisation pose plusieurs questions :

1. L'application d'une méthode de généralisation dans l'espace des états signifie que l'observation d'une situation ou d'une séquence de situations particulière entraîne la modification de l'estimation de l'utilité non seulement pour la situation concernée mais aussi pour d'autres situations. Existe-t-il des garanties que cette méthode converge ? Et si oui, converge-t-elle vers la fonction d'utilité optimale, V^* dans le cas de la fonction d'utilité définie sur les états ?
2. Les méthodes itératives d'amélioration de politique entremêlant phases d'évaluation et phases d'amélioration peuvent-elles encore s'appliquer avec des garanties de convergence vers la politique optimale ?
3. Les techniques d'exploration ε-gloutonnes continuent-elles à être efficaces pour, d'une certaine manière, échantillonner l'espace des exemples ?
4. Quelle mesure d'erreur utiliser pour évaluer la performance de l'approximation de la fonction d'utilité ? Est-il encore approprié d'utiliser la mesure d'erreur quadratique qui est employée pour la régression ?

Au moment de la rédaction de cet ouvrage, ces questions font l'objet de recherches et n'ont pas encore de réponses définitives. Répondre aux deux premières questions est d'autant moins facile que l'apprentissage par renforcement implique souvent à la fois un apprentissage de type incrémental capable de prendre en compte les exemples au fur et à mesure de leur arrivée, mais aussi un environnement qui peut évoluer au cours du temps. De plus, sans même avoir affaire à un environnement changeant, les exemples eux-mêmes évoluent du fait qu'ils correspondent souvent à des évaluations de gain qui sont adaptatives, comme c'est le cas par exemple pour la

Copyright © 2010 Eyrolles.

méthode des différences temporelles. Les problèmes de convergence et de vitesse de convergence sont donc encore plus aigus que pour l'induction classique.

De nombreux travaux ont été publiés concernant des expériences d'apprentissage par renforcement avec généralisation : Boyan et Moore [BM95a] ont utilisé des méthodes de plus proches voisins dans une approche d'itération de valeur ; Lin [Lin91] a mis en œuvre un réseau connexionniste avec apprentissage par rétropropagation de gradient pour apprendre la fonction d'utilité $Q(s, a)$ dans le Q-learning ; Watkins [Wat89], toujours dans le cadre du Q-learning, a utilisé la technique de CMAC (*Cerebellar Model Articulatory Controller*) due à Albus [Alb75, Alb81]) (voir la section 6.3) et a été suivi par de nombreux autres chercheurs ; Tesauro [Tes95] a utilisé un réseau connexionniste pour apprendre la fonction $V(s)$ dans le cas du jeu de Backgammon et Zhang et Dietterich [ZD95] ont utilisé un réseau connexionniste dans une technique de différence temporelle TD(λ) pour apprendre des stratégies d'ordonnancement de tâches pour des ateliers.

Dans l'ensemble, même si des résultats spectaculaires ont été rapportés, des interférences pernicieuses sont observées entre la mise à jour adaptative des valeurs d'utilité et l'apprentissage par généralisation. En effet, alors que dans les environnements discrets avec maintien de tables de valeurs, il existe des garanties que toute opération qui met à jour la valeur d'utilité (selon les équations de Bellman) ne peut que réduire l'erreur entre la valeur courante estimée et la valeur optimale, de telles garanties n'existent plus dans le cas continu avec des techniques de généralisation. Il semble en particulier que les méthodes de différences temporelles à plusieurs pas ne soient pas appropriées dans les méthodes de généralisation, et qu'il faille réexaminer les méthodes d'évaluation locales de valeur d'utilité dérivées des équations de Bellman. Boyan et Moore, par exemple, ont les premiers attiré l'attention sur ce problème en 1995 [BM95a] en donnant des exemples de fonctions d'utilité dont l'erreur croissait de manière arbitrairement grande par l'utilisation de techniques de généralisation. Certaines solutions *ad hoc* pour des classes de problèmes particulières ont été proposées, mais elles ne conduisent généralement qu'à des optima locaux. La question des conditions nécessaires et suffisantes pour la convergence dans le cas de l'utilisation de méthodes de généralisation de fonctions d'utilité reste donc ouverte.

Parmi les causes potentielles de problèmes figure le fait que lors de l'apprentissage par renforcement, les exemples utilisés pour l'apprentissage résultent d'un échantillonnage qui reflète la politique courante, laquelle n'est pas la politique optimale cible. Il semble que les méthodes d'exploration classiques, telles que la méthode ε-gloutonne, ne soient pas adaptées à cette situation. Là encore, des recherches sont nécessaires pour mieux comprendre les interactions entre échantillonnage, apprentissage et amélioration de politique.

Finalement, la question de la mesure d'erreur à utiliser pour la généralisation a été soulevée. En effet, contrairement à la tâche de régression pour laquelle l'écart quadratique est approprié, l'apprentissage par renforcement vise moins à approximer correctement les fonctions d'utilité qu'à fournir la meilleure politique. Ce qui compte n'est pas l'écart à la fonction optimale d'utilité, mais le fait que l'estimation d'utilité conduise bien à ce que la meilleure politique soit choisie. Il s'agit donc de respecter l'ordre relatif des politiques afin que la meilleure, π^*, soit en tête. Nous verrons dans la section 6.4 que cela a conduit récemment à réexaminer l'ensemble du problème.

Étant données toutes ces interrogations, les deux approches par estimation de fonction les plus employées actuellement reposent toutes les deux sur une technique de descente de gradient pour réduire l'écart quadratique entre les valeurs estimées à l'instant t et la cible courante, par exemple celle rapportée par la méthode de différences temporelles.

- La première approche consiste à utiliser un *réseau connexionniste* (généralement un perceptron multicouche) comme réalisation d'un vecteur de paramètre θ_t en le modifiant par une technique de rétropropagation de gradient selon une formule telle que la suivante (ici pour

Copyright © 2010 Eyrolles.

la fonction d'utilité V) :

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t \, \alpha \left[v_t - V_t(s_t) \right] \nabla_{\boldsymbol{\theta}_t} V_t(s_t) \tag{16.23}$$

où v_t représente la cible, c'est-à-dire l'estimation courante de $V^{\pi}(s_t)$, et α est un pas d'apprentissage décroissant.

- La seconde approche consiste à utiliser une *combinaison linéaire de n fonctions de base* pour approximer la fonction cible. L'ensemble des fonctions de base $\Phi(s) = \{\phi_1(s), \ldots, \phi_n(s)\}$ est défini sur l'espace $\mathcal{E}$ des états. L'approximation de la fonction d'utilité correspond à la formule :

$$V_t(s) = {\boldsymbol{\theta}_t}^{\mathsf{T}} \Phi(s) = \sum_{i=1}^{n} \theta_t(i) \, \phi_i(s) \tag{16.24}$$

Dans ce cas, le gradient de la fonction V_t par rapport à $\boldsymbol{\theta}_t$ devient : $\nabla_{\boldsymbol{\theta}_t} V_t(s) = \Phi(s)$. Cette règle a l'avantage d'être simple et de conduire à un seul optimum. Comme de plus elle est assez efficace en termes de données et de calculs, elle a la faveur de nombreux chercheurs et praticiens.

Une approche duale consiste à apprendre non pas la combinaison de fonctions de base, mais à apprendre ces fonctions de base elles-mêmes. Souvent cela correspond à apprendre des voisinages dans l'espace. Ces méthodes font l'objet de la section suivante.

6.3 Méthodes de généralisation par partition de l'espace

L'idée de ces méthodes est de décider de la réponse en un point (par exemple l'utilité associée à un état, ou à un couple (état, action)) en fonction des valeurs connues dans son voisinage. L'apprentissage consiste dans ce cas à définir le voisinage en chaque point. Le chapitre 15 traite ce problème en général, nous indiquons donc ici seulement les grandes approches testées en apprentissage par renforcement.

- *Techniques par couverture de voisinages uniformes.* L'idée est d'associer à chaque point dont la valeur est connue un voisinage, par exemple une boule (voir la figure 16.5). La valeur d'un point inconnu est décidée en fonction des voisinages, par exemple des boules, dont elle fait partie. On utilise souvent une fonction linéaire des valeurs des boules dont le point fait partie. On peut modifier la densité des boules, ou bien leur forme, ce qui revient alors à rendre non linéeaire l'approximation.

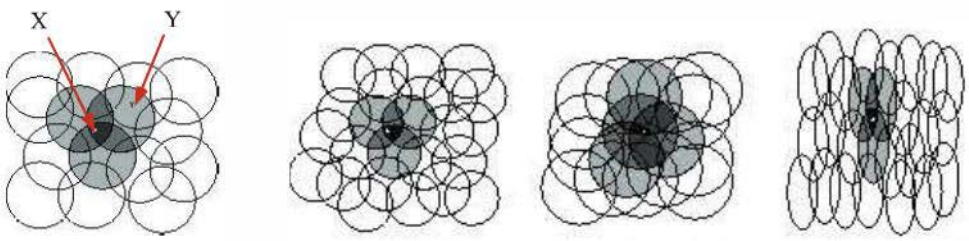


FIG. 16.5: *La valeur cherchée au point X (par exemple $V(X)$) dépend des valeurs des points Y dont les voisinages incluent X (d'après [SB98], p.203).*

- *Techniques par partition adaptative de l'espace.* Une idée qui a fait l'objet de travaux récents consiste à définir adaptativement une partition de l'espace. Dans ce cas, chaque partition est exclusive des autres, et la valeur d'un point est celle associée à la partition à laquelle il

Copyright © 2010 Eyrolles.

appartient (voir figure 16.6). La granularité de la partition est d'autant plus fine que la valeur estimée connaît des variations importantes, dans un esprit similaire à une décomposition en éléments finis. Le lecteur est invité à se reporter par exemple à [MA95, MM99, Rey00].

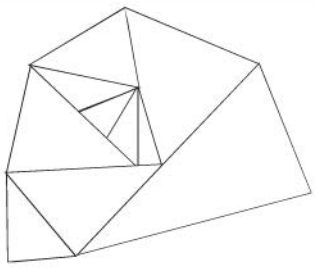


FIG. 16.6: *Sur cet exemple où l'on suppose que l'espace des états est bidimensionnel, la partition de l'espace est à granularité variable, d'autant plus fine que les variations de la fonction estimées sont importantes.*

- *Techniques par utilisation de fonctions noyau.* Les fonctions noyau permettent de définir un voisinage variable autour des points. Une fonction radiale typique est la fonction gaussienne dont la variable est la distance entre le centre c et le point s :

$$\phi_i(s) = exp\left(-\frac{\|s - c_i\|^2}{2\sigma_i^2}\right) \tag{16.25}$$

  Ces fonctions de voisinage présentent l'avantage d'être adaptatives et différentiables, d'où leur emploi fréquent.

- *Approches hiérarchiques.* Une approche pour essayer de faire face à la malédiction de la dimensionalité est de les traiter comme des hiérarchies de problèmes d'apprentissage avec des granularités croissantes. Par exemple, l'une des méthodes employées consiste à utiliser une hiérarchie sous forme de porte logique (voir figure 16.7). Chaque boîte reçoit une description de l'environnement et prescrit un comportement, la porte choisit lequel est effectivement appliqué. L'apprentissage peut alors consister à apprendre les boîtes ou bien le critère de sélection de la porte.
  En approfondissant, il devient intéressant de voir s'il est possible de décomposer un comportement en tâches indépendantes, ou quasi indépendantes, et de structurer un problème en fonction des interrelations entre sous-problèmes. Cela ouvre la voie à l'intégration de plusieurs comportements, qu'ils soient issus de plusieurs types ou niveaux de descriptions, ou qu'ils résultent d'univers multiagents. L'article [Die00] constitue un bon point d'entrée pour la littérature consacrée à cette approche.

6.4 Méthodes directes d'apprentissage de politique

Les difficultés liées aux méthodes de généralisation pour approximer les fonctions d'utilité, concernant en particulier la convergence, non garantie, et la qualité de l'approximation, parfois très mauvaise, ont relancé l'idée d'opérer une recherche directe dans l'espace des politiques plutôt que de passer par l'apprentissage de fonctions d'utilité. Au lieu de chercher à minimiser un écart quadratique par rapport à la fonction d'utilité vraie, le principe consiste à exprimer les politiques comme des fonctions paramétrées, par un vecteur θ, et à chercher une valeur de θ correspondant à une politique optimale, c'est-à-dire maximisant l'espérance de gain $E_\pi(R_t)$.

Copyright © 2010 Eyrolles.

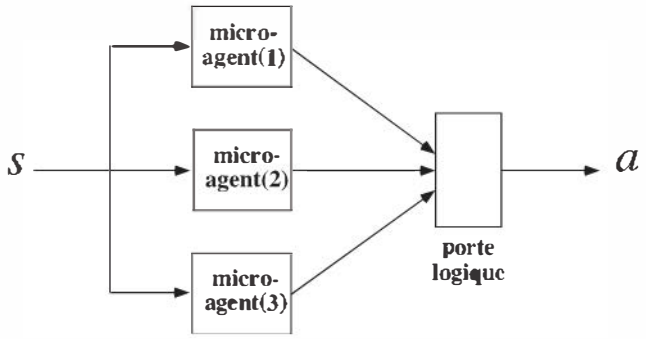


FIG. 16.7: *Une hiérarchie organisée sous forme de porte logique.*

Pour ce faire, on passe généralement par une expression paramétrée par $\boldsymbol{\theta}$ de l'estimation de gain, noté $\eta(\boldsymbol{\theta})$, et on opère ensuite une descente de gradient pour trouver la valeur de $\boldsymbol{\theta}$ maximisant cette expression. Les difficultés de cette approche sont nombreuses. La première est d'évaluer précisément l'espérance de gain $\eta(\boldsymbol{\theta})$. La seconde consiste à trouver une paramétrisation pratique de la politique, et donc du gain. Un réseau connexionniste multicouche est souvent utilisé à cet effet. La troisième concerne la possibilité d'utiliser une technique de descente de gradient sur $\eta(\boldsymbol{\theta})$. Cette fonction est rarement différentiable, et il faut donc avoir recours à des astuces permettant de reformuler le problème. Finalement se pose l'éternel problème de la combinaison entre évaluation de l'espérance de gain et amélioration de la politique. L'approche par recherche directe de politique n'est donc pas évidente, mais elle connaît actuellement un vif intérêt (voir par exemple [BM99, Bax00, GU00, Wil92b]).

Nous donnons ici des éclairages ponctuels sur des pistes étudiées actuellement.

6.5 Les méthodes « actor-critic »

Les méthodes « actor-critic » séparent la partie fonctions d'évaluation et de prédiction (*critic*) de la partie politique ou choix de l'action (*actor*). La partie *actor* associe une action à un état d'une manière probabiliste. Le *critic* correspond à la fonction d'utilité conventionnelle qui associe le gain espéré aux états. Ainsi, le *critic* résout un problème de prédiction, tandis que l'*actor* résout un problème de contrôle.

Lorsque le *critic* et l'*actor* utilisent la même fonction d'utilité, pour prédire le gain et pour sélectionner la prochaine action, on retrouve les méthodes d'apprentissage par renforcement telles que TD-learning. Mais il est possible d'envisager que l'*actor* utilise un espace de politique différent, ne reposant pas sur les critères utilisés par le *critic*. Dans ce cas, celui-ci est utilisé pour fournir une évaluation de la qualité instantanée de la politique, et surtout, de fournir des informations permettant de mettre en œuvre une méthode d'optimisation par gradient de la politique.

L'un des intérêts de ce découplage est de permettre l'utilisation d'un espace de politiques choisi pour ses qualités propres.

6.6 Apprentissage par renforcement bayésien

Dans l'apprentissage par renforcement bayésien développé récemment, le but est de trouver une approche bien fondée au problème de l'exploration en représentant explicitement l'incertitude des renforcements, des modèles de transitions entre états et des fonctions de valeurs. Une distribution de probabilité est ainsi définie sur l'espace des modèles décisionnels de Markov (MDP), au lieu de

Copyright © 2010 Eyrolles.

ne maintenir qu'une seule estimation de vraisemblance maximale. Cette distribution est utilisée pour choisir l'action d'utilité attendue maximale.

La difficulté principale de cette approche est de définir une distribution *a priori* sur l'espace des MDP.

Pour plus de détails, nous renvoyons le lecteur à un papier fondateur sur ce sujet [WFRT00].

7. Contrôle optimal par recherche arborescente et algorithme UCT

Le problème du contrôle optimal consiste à déterminer à chaque instant l'action qui maximise le gain à venir. Une approche, motivée en particulier par des applications sur les jeux, vise à résoudre le problème du contrôle optimal en effectuant une recherche arborescente dans un arbre dont la racine correspond à l'état courant du système et où les nœuds sont les états accessibles tandis que les arcs représentent les différentes actions possibles. Une branche de l'arbre relie le nœud racine à un nœud feuille de l'arbre et correspond à une séquence de coups des joueurs depuis une position initiale (la racine) jusqu'à une position finale. La valeur de cette branche est simplement la somme (éventuellement actualisée avec γ) des renforcements associés aux arcs traversés. On cherche naturellement à identifier la branche dont la valeur est la plus élevée afin de déterminer l'action à prendre à la racine (première action de la branche jugée optimale).

On résout généralement ce genre de problèmes en utiliseant un algorithme de recherche dans l'arbre des actions possibles. L'algorithme *alpha-beta* par exemple [RN06], développe l'arbre des coups possibles jusqu'à une profondeur seuil qui dépend des ressources calculatoires disponibles entre deux coups. Cette approche, qui est à la base des algorithmes utilisés pour les jeux tels que le jeux d'échec, est inopérante dans les cas où le facteur de branchement – nombre d'actions possibles en chaque état – est grand (supérieur à quelques dizaines). Il devient alors nécessaire d'avoir une stratégie de choix des actions à envisager qui favorise des parties de l'arbre, que l'on espère les plus prometteuses, au dépend des parties jugées les moins intéressantes. On se trouve à nouveau devant le fameux *compromis exploration vs. exploitation.*

L'algorithme UCB (pour *Upper Confidence Bounds*), introduit en 2002 par Peter Auer, Nicolo Cesa-Bianchi et Paul Fischer [ACBF02], résout ce compromis en choisissant à chaque tour, parmi toutes les actions possibles, celle qui maximise la somme d'un terme d'exploitation (moyenne des gains obtenus en choisissant cette action) et d'un terme d'exploration (terme de variance d'autant plus élevé que cette action a été choisie peu souvent dans le passé). On obtient ainsi un compromis tel que lorsque l'on joue un grand nombre de fois, la probabilité de choisir un autre bras que le bras optimal tend vers zéro avec une vitesse de convergence optimale (en $\ln n/n$ où n est le nombre de parties jouées). Contrairement à l'algorithme *alpha-beta* (ou à *MinMax* dont il découle), UCB choisit l'action qui peut être la meilleure (selon la moyenne augmentée de la largeur de l'intervalle de confiance). On parle de stratégie optimiste dans l'incertain.

Plus précisément, ces algorithmes dérivent de recherches sur le problème du bandit multi-bras (extension du bandit à deux bras présenté en section 1.5). Un problème de bandit à K bras est défini par des variables aléatoires $X_{i,n}$ avec $i \leq i \leq K$ et $n \geq 1$, où i est le numéro d'un « bras » et n dénote l'instant considéré. Les tirages successifs du bras i rapportent des renforcements $X_{i,1}, X_{i,2}, \ldots$ indépendants et identiquement distribués selon une loi de moyenne μ_i. Une stratégie d'allocation est un algorithme déterminant le bras à jouer à l'instant $n+1$ en se basant sur la séquence des bras déjà tirés et des renforcements observés. Soit $T_i(n)$ le nombre de fois où le bras i a été tiré durant les n premiers instants. Le *regret* associé à la stratégie est

Copyright © 2010 Eyrolles.

alors défini par :

$$\mu^* n - \mu_j \sum_{j=1}^{K} \mathbb{E}[T_j(n)] \qquad \text{où} \quad \mu^* \stackrel{\text{def}}{=} \max_{1 \leq i \leq K} \mu_i$$

Le *regret* est ainsi l'espérance de perte due au fait que la stratégie d'allocation ne joue pas systématiquement le meilleur bras. Lai et Robbins [LR85] ont proposé en 1985 une stratégie d'allocation selon laquelle le bras optimal est joué exponentiellement plus souvent que les autres. Ils ont montré de plus que le regret obtenu à l'aide de cette stratégie est le meilleur possible. Cette stratégie associe une valeur appelée *indice de confiance supérieur* à chaque bras. Des travaux ultérieurs ont simplifié le calcul de cet indice qui se base sur l'histoire passée des tirages. L'algorithme UCB (dans sa version la plus simple) est esquissé dans l'algorithme 30.

Algorithme 30 : Algorithme UCB

début
  Initialisation : Jouer chaque bras une fois
  répéter
    Jouer le bras j qui maximise $\bar{x}_j + \sqrt{\frac{2 \ln n}{T_j(n)}}$ où $\bar{x}_j$ est le renforcement moyen obtenu en jouant le bras j, $T_j(n)$ le nombre de fois où le bras j a été joué et n le nombre total de tirages jusque là.
  jusqu'à *Fin du jeu*
fin

D'autres stratégies de type UCB ont été proposées [ACBF02].

L'algorithme UCB ne peut cependant pas être utilisé tel quel dans le cas où les gains peuvent subvenir bien après l'action choisie. L'algorithme UCT (pour *Upper Confidence bounds in Trees*) [KS06] utilise UCB de façon récursive pour estimer la valeur des nœuds et de la racine dans l'arbre des coups possibles. À chaque nœud exploré de l'arbre est associé un algorithme de type UCB sélectionnant la prochaine action à entreprendre. Le choix d'une séquence d'actions est ainsi décomposé en un ensemble de problèmes plus simples selon une méthode de type Monte-Carlo dans laquelle les branches les plus prometteuses sont explorées en priorité.

Dans ses grandes lignes, l'algorithme UCT développe et explore l'arbre de manière asymétrique en réalisant un compromis entre l'exploration des branches qui semblent les meilleures compte tenu des feuilles déjà explorées et l'exploration des branches encore mal connues. Des épisodes sont lancés depuis la racine jusqu'à des feuilles. En chaque nœud rencontré, s'il reste des successeurs non explorés, l'algorithme en choisit un aléatoirement. Si tous les nœuds fils ont déjà été visités au moins une fois, l'algorithme choisit le nœud n maximisant la valeur UCB de j. Quand une feuille est atteinte, sa valeur est intégrée dans la valeur moyenne des nœuds traversés (voir algorithme 32).

Plus précisément, l'algorithme UCT engendre itérativement des épisodes et retourne l'action associée au plus haut gain moyen observé. Les épisodes sont engendrés par la fonction *Explorer* qui sélectionne et effectue des actions récursivement jusqu'à ce qu'une condition de terminaison soit satisfaite : soit que l'état atteint est une feuille, soit que la profondeur limite ait été atteinte. Dans la fonction *MiseAJour*, le gain total $Q_{UCT}(s, a)$ est utilisé pour ajuster la valeur estimée pour la paire (*état, action*) à la profondeur donnée, et le nombre de visites de la paire (*état, action*) est augmenté de 1. Soit $s_1, a_1, s_2, a_2, \ldots, s_T$ la séquence d'états et d'actions correspondant à un

Copyright © 2010 Eyrolles.

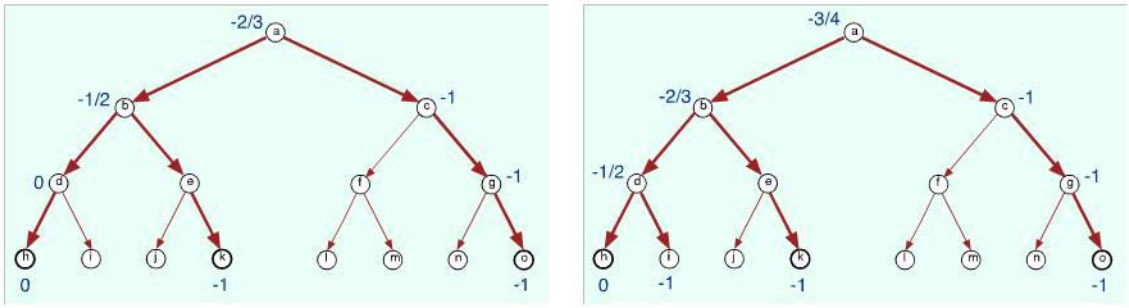


FIG. 16.8: *On suppose ici que l'arbre exploré à l'instant t est celui de gauche. Les feuilles h, k et o ont été atteintes. De ce fait le nœud b est affecté d'une valeur UCB de* $-1/2 + \sqrt{(2\ln 4)/2} \approx 0.677$. *De même, on a* $UCB(c) = -1 + \sqrt{2(\ln 4)/1} \approx 0.665$. *Donc* $UCB(b) > UCB(c)$ *et b est par conséquent choisi pour exploration ultérieure. Comme* $UCB(d) = 0 + \sqrt{2(\ln 4)/1} > UCB(e) = -1 + \sqrt{2(\ln 4)/1}$, *le nœud d est choisi. Puisque la branche passant par i n'a pas encore été explorée, elle est sélectionnée. La valeur de la feuille correspondante, -1, est remontée pour mettre à jour les valeurs de ses nœuds ancêtres :* $V(d) = (0-1)/2 = -1/2$; $V(b) = [2(-1/2) - 1]/3 = -2/3$; $V(a) = [3(-2/3) - 1]/4 = -3/4$. *On voit sur ce petit exemple, que l'algorithme UCT a tendance, au début, à se comporter comme un algorithme d'exploration en largeur d'abord [RN06], puis à se concentrer petit à petit vers les branches ayant mené à des parties (simulées) gagnées. Globalement, les valeurs remontées convergent vers les valeurs MinMax des nœuds correspondants.*

Algorithme 31 : Algorithme UCT (*état*)

Fonction *MeilleureAction(état, profondeur) : Booléen*
Résultat : MeilleureAction(*état*,0)
début
  répéter
    Explorer (état,0)
  jusqu'à *Temps épuisé*
fin

épisode exploré par UCT à partir de s_1. Chaque paire (*état, action*) est alors mise à jour selon :

$$n(s_t, a_t) \leftarrow n(s_t, a_t) + 1 \tag{16.26}$$

$$Q_{UCT}(s_t, a_t) \leftarrow Q_{UCT}(s_t, a_t) + \frac{1}{n(s_t, a_t)}[R_t - Q_{UCT}(s_t, a_t)] \tag{16.27}$$

où R_T est le gain cumulé mesuré depuis (s_t, a_t). Pour les paires (*état, action*) jamais rencontrées jusque-là, on initialise par : $n(s_t, a_t) = 1$ et $Q_{UCT}(s_t, a_t) = R_t$.

La performance de l'algorithme est essentiellement déterminée par la fonction *selectAction* qui choisit les actions à essayer. Pour ce choix, l'algorithme UCT utilise la technique UCB.

Chaque choix d'action, pour chaque nœud s exploré, est traité comme un problème de bandit multi-bras indépendant. Un bras correspond à une action a possible et le gain associé au gain cumulé mesuré (jusque-là) des chemins explorés partant du nœud : $Q_{UCT}(s_t, a_t, d)$ (où d est la profondeur du nœud s). Le bras (action) associé à la plus grande valeur de $Q_{UCT}(s_t, a_t, d) + C\sqrt{\frac{\ln n(s_t)}{n(a_t, s_t)}}$ est sélectionné, où $C > 0$ est une constante appropriée, $n(s_t)$ est le nombre de fois où le nœud s a été exploré à l'instant t, et $n(a_t, s_t)$ le nombre de fois où l'action a_t a été sélectionnée dans l'état s_t à t.

Copyright © 2010 Eyrolles.

Algorithme 32 : Fonction Explorer (*état*, *profondeur*)

Résultat : q (valeur de l'état)
début
  si Terminal(*état*) **alors retourner** 0
  si Feuille(*état*) **alors retourner** `Evalue` (*état*)
  action := `selectAction` (*état*, *profondeur*)
  (*étatSuivant*, *renforcement*) := `ActionSimulée` (*état*, *action*)
  $q := renforcement + \gamma$ `Explorer` (*étatSuivant*, *profondeur* + 1)
  `MiseAJour` (*état*, *action*, q, *profondeur*)
fin

L'algorithme UCT pose certains problèmes théoriques. Ainsi, l'indépendance des gains d'un coup au suivant n'est pas assurée puisque les valeurs des trajectoires issues d'un nœud dépendent des coups joués ensuite. Or cette indépendance est requise pour la convergence de l'algorithme UCB vers le choix optimal. Kocsis et Szepesvàri ont cependant prouvé que les valeurs des nœuds calculées par UCT convergent vers la valeur optimale, c'est-à-dire la valeur MinMax. De plus Munos et Szepesvàri ont établi que la vitesse de convergence de l'algorithme UCT dépend de la profondeur effective de l'arbre, c'est-à-dire du nombre de coups à partir duquel les estimations selon l'heuristique Monte-Carlo rejoignent la valeur Min-Max exacte. En pratique, cela signifie que la convergence sera assez rapide si les valeurs des nœuds sont corrélées à l'intérieur de l'arbre. En revanche, lorsque l'arbre ne présente pas de régularités, c'est-à-dire que le meilleur coup est caché dans une séquence qui semble peu prometteuse de prime abord, alors l'algorithme UCT est *a priori* peu efficace.

8. Le cas des environnements partiellement observables

Dans de nombreuses situations issues du monde réel, l'agent ne peut pas avoir une perception parfaite et complète de l'état du monde, mais doit faire face à toute une variété d'incertitudes sur cet état. On dit alors que l'état est « partiellement observable ». Souvent, ces observations incomplètes se traduisent par une ambiguïté sur l'état réel du monde dans la mesure où plusieurs états peuvent correspondre aux mêmes observations (le terme *aliasing* est fréquemment employé en anglais, d'où le terme « situations d'alias perceptuel » parfois utilisé en français, voir figure 16.9).

Le modèle des processus markoviens est alors inapplicable tel quel. Une idée est d'essayer de remédier au manque instantané d'information par la prise en compte d'une mémoire des événements passés. Des modèles à base de réseaux connexionnistes récurrents ont été expérimentés, mais sans qu'ils soient concluants au-delà de problèmes simples. Une autre approche consiste à prendre explicitement en compte l'incertitude sur les états. C'est ce que permettent les modèles de Markov cachés ou POMDPs (*Partially Observable Markov Decision Processes*).

Dans ces modèles, on introduit un nouvel espace Ω, celui des observations, avec les probabilités $O_t(o|s)$ associées d'observer o quand l'état est s, et b_o une distribution de probabilité initiale sur les états. Afin de calculer, et d'optimiser, l'espérance de gain associée à une politique, il faut relier les états, les renforcements (correspondants aux transitions entre états) et les probabilités de transition entre états aux observations ou plutôt aux informations dont dispose l'agent. En

Copyright © 2010 Eyrolles.

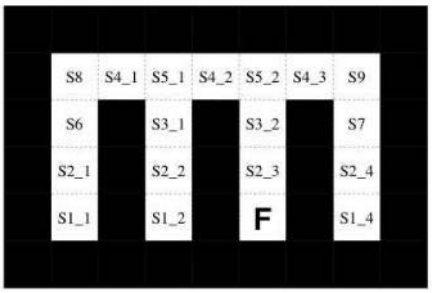


FIG. 16.9: *Un labyrinthe, dit de Lanzi, dans lequel un agent qui ne perçoit que les 8 cases l'entourant ne peut distinguer que les états S6, S7, S8 et S9. En revanche, les états S3_1 et S3_2 ne sont pas distinguables.*

général, il n'est pas possible de trouver la politique optimale pour un POMDP en utilisant uniquement l'observation courante o_t. Il faut extraire suffisamment d'information des observations passées pour définir des informations dites suffisantes (au sens des statistiques suffisantes) et des états estimés, et, ensuite, pouvoir définir une politique optimale. En pratique cependant, les algorithmes existants sont limités à des espaces d'états très restreints et à des horizons temporels très limités. L'espace des états estimés croît en effet très vite, et, plus encore, la fonction de valeur à estimer, linéaire par morceaux, demande un nombre exponentiel de paramètres pour sa représentation.

Parmi les pistes explorées pour dépasser ces limitations, on peut citer l'apprentissage directement dans l'espace des politiques, sans passer par des fonctions de valeur. On peut ainsi explorer un espace de politiques paramétré en utilisant une descente de gradient. Une autre approche récente se base sur la notion de *représentations par états prédictifs* (PSR ou *Predictive State Representation* en anglais). Ces représentations sont liées aux probabilités de réalisation de certaines trajectoires dans le futur. Des algorithmes d'apprentissage sont en cours de développement pour découvrir ces PSR (souvent associées à des états critiques (e.g. portes, obstacles) de l'environnement) et pour apprendre leurs probabilités de réalisation [SLJ+03, ABT07].

Pour de plus amples détails, nous reportons le lecteur à l'ouvrage récent en deux volumes [SB08, BS08].

9. Exemples d'application

Il n'est pas question de fournir un panorama complet des domaines d'application des méthodes d'apprentissage par renforcement. Nous nous contentons donc de présenter rapidement quelques exemples pour lesquels il est facile de trouver davantage de détails dans la littérature scientifique.

9.1 Le TD-Gammon

Bien que l'apprentissage par renforcement ne se prête pas naturellement au domaine du jeu car l'environnement, commandé par l'adversaire, n'est pas stationnaire, il y a eu de nombreuses tentatives de systèmes d'apprentissage pour des jeux divers. La plus ancienne application réussie

Copyright © 2010 Eyrolles.

est celle du jeu de dames américain[10] due à Samuel en 1959 [Sam59]. Le programme apprenait une fonction d'évaluation $V(s)$ représentée par une fonction linéaire d'un certain nombre de facteurs déterminés par Samuel. Il employait un mécanisme d'apprentissage similaire à celui de l'algorithme d'itération de valeur, des différences temporelles et du Q-learning.

Une autre succès plus récent est celui de Tesauro dans le domaine du backgammon [Tes92, Tes94, Tes95]. Ce jeu comporte environ 10^{20} états, ce qui rend impossible une méthode basée sur une table d'états. Il faut donc utiliser une méthode de généralisation dans l'espace des états. Tesauro a employé un perceptron multicouche (voir chapitre 10) à une couche cachée avec apprentissage par rétropropagation de gradient, pour réaliser un système d'estimation de la fonction de valeur :

Position sur le jeu $\longrightarrow$ *Probabilité de victoire pour le joueur courant*

Une première version de base de l'algorithme appelé TD-GAMMON ne comportait aucune connaissance spécifique du domaine, tandis que les versions ultérieures (TD 1.0, TD 2.0, TD 2.1) utilisaient des connaissances propres à certaines positions de jeu. Pour toutes ces versions, l'apprentissage fut réalisé par simulation de jeu de l'ordinateur contre lui-même. Remarquablement, aucune stratégie d'exploration n'était utilisée et l'algorithme choisissait toujours le coup apparemment le meilleur. Cela ne pose pas de problème au backgammon car les situations de jeu obtenues dépendent en partie d'un tirage aux dés ce qui suffit à garantir que tous les états seront visités. De plus, il s'agit d'un jeu dans lequel les parties terminent en un temps fini, ce qui assure que des renforcements sont reçus assez fréquemment. Les résultats obtenus par les différentes versions de TD-GAMMON sont résumés dans le tableau suivant :

	Parties jouées en apprentissage	Cellules sur la couche cachée	Résultats
Version de base			Médiocre
TD 1.0	300 000	80	Battu de 13 points en 51 matches
TD 2.0	800 000	40	Battu de 7 points en 38 matches
TD 2.1	1 500 000	80	Battu d'un point en 40 matches

TD-GAMMON se place parmi les meilleurs joueurs mondiaux.

9.2 Le jeu de Go

Le Go est un jeu à deux joueurs d'origine chinoise. Deux adversaires placent à tour de rôle des pions blancs ou noirs sur les intersections d'un tableau appelé *go-ban* de dimension normale 19 × 19, mais qui se joue aussi sur des plateaux de dimensions plus réduites (9 × 9 ou 13 × 13). Il s'agit d'un jeu à information complète car il n'y a pas de hasard et chaque joueur connaît à chaque instant toutes les possibilités de jeu. Le vainqueur est le joueur ayant réussi à délimiter un territoire plus vaste que celui de son adversaire (voir figure 16.2).

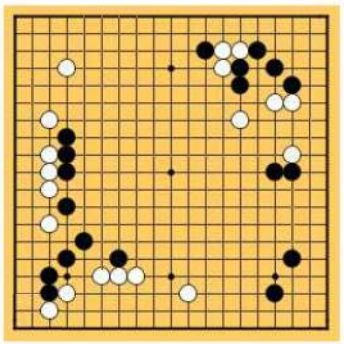
Ce jeu est l'un des derniers grands jeux (avec le pocker par exemple) pour lesquels les experts humains restent très supérieurs à la machine. En ce qui concerne le Go, deux raisons au

[10] Contrairement au jeu de dames joué en Europe continentale, le jeu de dames nord-américain (*checkers*) se joue sur un damier de 8 × 8 cases. De plus certaines règles de prise sont différentes.

Copyright © 2010 Eyrolles.

moins peuvent être invoquées. D'une part, on n'a pas, jusqu'à présent (en 2008), trouvé de fonction d'évaluation d'un *Go-ban* satisfaisante, qui reflète assez fidèlement la force d'une position. D'autre part, le facteur de branchement du jeu, de l'ordre de 200 en milieu de partie, excède le facteur de branchement de jeux comme les échecs (de l'ordre de 30 environ) et ne se prête pas facilement à des approches traditionnelles comme l'algorithme alpha-beta qui suppose une exploration complète (même si des élagages sont recherchés) de l'arbre de jeux jusqu'à une profondeur seuil.

À ces deux obstacles, l'approche UCT permet de donner une réponse. La première idée est de remplacer l'élusive fonction d'évaluation de position par une exploration par échantillonnage de Monte-Carlo de parties possibles jusqu'à leur conclusion. Ainsi, chaque partie jouée permet d'obtenir un résultat sûr (perte ou gain). C'est la moyenne de gain de ces parties qui joue alors le rôle de fonction d'évaluation. Par ailleurs, c'est la deuxième idée, au lieu d'une exploration complètement aléatoire, l'approche UCT (voir section 7) permet de concentrer les explorations vers les régions les plus prometteuses de l'arbre de jeu. Finalement, un avantage de cette technique est d'être intrinsèquement *anytime*, c'est-à-dire de pouvoir fournir une réponse à n'importe quel instant, cette réponse s'affinant avec le temps imparti.



TAB. 16.2: *Exemple de situation de jeu de Go.*

Les systèmes de jeu de Go les plus performants actuels utilisent la technique UCT. C'est le cas par exemple de MoGo [GS07], qui a remporté le 26 mars 2008 la première victoire homologuée, non-blitz, opposant une machine à un maître du Go !

9.3 Applications au contrôle et à la robotique

Si les applications de l'apprentissage par renforcement sont de plus en plus nombreuses, elles restent encore du domaine de l'art autant que de la science. C'est pourquoi il est intéressant avant de se lancer dans un problème d'étudier ce qui a été fait pour des problèmes connexes. Sans chercher, ni pouvoir, être exhaustifs, nous citons ici des travaux remarquables dans le domaine du contrôle et de la robotique. Les références associées fournissent un point de départ pour explorer la littérature sur ce thème.

Le contrôle et la robotique se prêtent bien à l'apprentissage par renforcement car il s'agit d'applications souvent difficiles à programmer complètement, dans lesquelles l'information n'est disponible que lors du fonctionnement de manière incrémentale et dans un environnement parfois changeant. Parmi les applications les plus spectaculaires, on compte le robot jongleur de Schaal et Atkeson [SA94]. Ce robot comporte deux bras commandés par un système à trois degrés de liberté. Il prend une décision d'action toutes les 200 ms et doit tenter de maintenir en l'air une sorte de pendule inversé.

Une autre application concerne des robots mobiles devant pousser des boîtes d'un endroit à un autre dans des pièces [MC91]. Ce problème se caractérise par de grandes incertitudes sur l'effet des actions. L'approche utilisée employait une décomposition hiérarchique des tâches en sous-tâches. Une application similaire [Mat94] impliquait quatre robots mobiles devant rassembler des disques. Outre l'immense espace d'états impliqués, la tâche se caractérise par des problèmes de contrôle distribué, de communication et, éventuellement, par le partage des connaissances apprises.

Copyright © 2010 Eyrolles.

L'apprentissage par renforcement a été également employé avec succès dans une tâche de contrôle de plusieurs ascenseurs dans un immeuble de dix étages. L'objectif est de minimiser la moyenne du temps d'attente des utilisateurs. L'approche utilisée avec du Q-learning et un estimateur dans l'espace des états à l'aide d'un réseau connexionniste a donné d'excellents résultats comparés à l'état de l'art.

D'autres applications incluent l'optimisation du remplissage de conteneurs, l'allocation dynamique de canaux pour les téléphones cellulaires [SB97] et l'ordonnancement de tâches pour l'installation et le test de charges pour la navette spatiale [ZD95, Zha96].

10. Bilan et perspectives

L'apprentissage par renforcement s'intéresse au problème général se posant à un agent devant apprendre à choisir ses actions dans le but d'accroître son espérance de gain à long terme. La structure de son environnement étant généralement supposée inconnue, l'agent doit apprendre à partir de ses interactions avec le monde.

Dans les approches fondées sur les fonctions d'utilité, l'agent cherche à apprendre l'utilité de chaque *état* ou de chaque paire (*état, action*). Il sélectionne alors l'action associée à l'utilité maximale. Si la fonction d'utilité estimée est exacte, cette approche conduit à la politique optimale sous des conditions très générales [SB98, BT96]. Cependant, pour la plupart des problèmes du monde réel, il est impossible de représenter les fonctions d'utilité exactement, en particulier avec des tables de valeurs. L'agent doit alors chercher une bonne approximation de la fonction d'utilité au sein d'une classe restreinte de fonctions (par exemple sous la forme d'un réseau connexionniste ou d'une classe de fonctions noyau). Cette approche a permis l'obtention de succès remarquables dans l'apprentissage de jeux (jeu de dames [Sam59], backgammon [Tes92, Tes94], jeu d'échecs [BTW00]), dans le domaine de l'ordonnancement de tâches [ZD95] et dans l'allocation dynamique de canaux de communication [SB97]).

La combinaison entre évaluation et apprentissage par généralisation pose de sérieux problèmes non encore résolus. C'est pourquoi la recherche directe dans l'espace des politiques est une option qui retient l'attention des chercheurs. Il faut signaler également les efforts visant à rendre moins empiriques les méthodes d'exploration utilisées pour échantillonner les situations [Str00].

Prolongeant l'apprentissage hors politique, certains travaux actuels se penchent sur l'apprentissage lorsque l'expérience porte sur un environnement qui diffère de l'environnement cible. C'est le cas par exemple de l'apprentissage de la bicyclette qui se fait avec une bicyclette à stabilisateurs pour lequel les contraintes sont différentes [Ran00]. Ce sera le cas éventuellement de robots d'exploration planétaire.

Par ailleurs, une question essentielle porte sur l'intégration de l'apprentissage par renforcement, qui est de fait un apprentissage de réflexes, avec l'activité de planification qui implique un raisonnement de nature beaucoup plus stratégique.

Notes historiques et sources bibliographiques

L'apprentissage par renforcement a une longue histoire et plusieurs ascendances. L'une d'entre elles concerne les théories behavioristes de l'apprentissage par essais et erreurs, association et

Copyright © 2010 Eyrolles.

punitions-récompenses. Une autre réside dans les théories du contrôle optimal et des approches par programmation dynamique, en particulier dues à Bellman. Une autre est directement liée aux efforts en intelligence artificielle pour simuler des souris cybernétiques, apprendre à jouer au tic-tac-toe ou aux dames, ou encore modéliser certains apprentissages au niveau neuronal.

L'idée d'utiliser une fonction associant à chaque couple (état, action) une estimation de sa valeur remonte à Shannon [Sha50] en 1950 qui la proposa dans le cadre du jeu d'échec. L'un des premiers articles influents en intelligence artificielle est celui de Minsky en 1961 [Min61] dans lequel il introduisit le problème du *credit assignment problem* central en apprentissage par renforcement. Donald Michie, un disciple de Turing, explora lui aussi plusieurs méthodes d'apprentissage par renforcement dont le système BOXES [MC68] qui, dans le cadre du pendule inversé, associe une action à chaque « boîte » dans l'espace des états. Nous citerons également Harry Klopf, qui dans les années 1970 fut l'un de ceux qui insistèrent sur la différence entre apprentissage supervisé et apprentissage par renforcement. Il introduisit les premiers éléments de l'idée d'apprentissage par différence temporelle, idée qui fut reprise et développée par Barto et Sutton dans les années 1980 et 1990. Watkins en 1989 [Wat89] réunit les approches de la théorie du contrôle et de l'apprentissage par différence temporelle dans le Q-learning qui eut un grand impact sur le domaine et aida à la propagation des idées d'apprentissage par renforcement dans des cercles plus larges d'utilisateurs et de théoriciens, aidé en cela par le succès de Tesauro sur le back-gammon. La dernière décennie a connu un développement exceptionnel de ce domaine de recherche qui est certainement appelé à connaître d'autres révolutions.

Le compromis exploitation contre exploration est connu depuis longtemps et a été modélisé par le scénario du bandit à deux bras par exemple par [Bel61] qui en a fait une analyse extensive dans le contexte de la théorie de la décision et du contrôle adaptatif. Holland [Hol75] l'a étudié également dans le cadre de l'analyse des algorithmes génétiques afin de montrer que ceux-ci réalisent spontanément une allocation optimale entre exploration et exploitation. On peut aussi se référer à [BF85].

Le livre de Barto et Sutton [SB98] est un ouvrage irremplaçable pour l'étude de l'apprentissage par renforcement et pour les sources historiques et bibliographiques le concernant. Un excellent ouvrage très à jour en français en deux tomes émanant de la communauté PDMIA (Processus Décisionnels Markoviens et Intelligence Artificielle) devrait paraître prochainement [SB08, BS08].

Copyright © 2010 Eyrolles.

Résumé

L'apprentissage par renforcement concerne l'apprentissage par un agent autonome d'une politique optimale, c'est-à-dire de l'action la mieux adaptée à chaque situation envisageable pour le système décisionnel considéré. La structure de son environnement étant généralement supposée inconnue, l'agent doit apprendre à partir de ses interactions avec le monde. En particulier, aucun professeur ne lui dit quelle action est la meilleure à prendre dans une situation donnée et seul un signal de renforcement assez pauvre (un scalaire) l'informe de temps en temps de sa performance liée à ses décisions passées.

Classiquement, l'apprentissage par renforcement est basé sur une fonction d'utilité. Divers algorithmes et structures de représentations de l'environnement ont été proposés pour apprendre cette fonction d'utilité dans le cadre formel des processus décisionnels markoviens (PDM).

Dans ces approches, l'agent cherche à apprendre l'utilité de chaque *état* ou de chaque paire (*état, action*). Il sélectionne alors l'action associée à l'utilité maximale. Si la fonction d'utilité estimée est exacte, cette approche conduit à la politique optimale sous des conditions très générales [SB98, BT96]. Cependant, pour la plupart des problèmes du monde réel, il est impossible de représenter les fonctions d'utilité exactement, notamment avec des tables de valeurs. L'agent doit alors chercher une bonne approximation de la fonction d'utilité au sein d'une classe restreinte de fonctions (par exemple sous la forme d'un réseau connexionniste ou d'une classe de fonctions noyau). La difficulté est de trouver une représentation compacte assurant la convergence des méthodes d'apprentissage pour les PDM. C'est pourquoi de nouvelles approches d'apprentissage plus direct de la politique sont aussi explorées.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Cinquième partie

Au-delà de l'apprentissage supervisé

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 17

Apprentissage de combinaisons d'experts

Ce chapitre répond à la question de savoir comment combiner plusieurs classificateurs pour qu'ils s'entraident à résoudre le même problème d'apprentissage supervisé. Plutôt que de chercher à savoir quel est le meilleur classificateur pour un problème donné, il est en effet naturel de se demander si une combinaison de plusieurs classificateurs pourrait être plus efficace que le meilleur d'entre eux. Une approche différente et complémentaire, le boosting*, applique plusieurs fois le même classificateur, mais change le poids des exemples dans l'ensemble d'apprentissage. D'autre part, certains algorithmes d'apprentissage ne fonctionnent en principe que pour la séparation de deux classes (l'apprentissage de concept). Ce chapitre aborde aussi le problème de leur passage à la résolution de problèmes d'apprentissage pour un nombre quelconque de classes.*

Copyright © 2010 Eyrolles.

Sommaire

1 Principes des méthodes par combinaison 572
1.1 Introduction 572
1.1.1 Le méta-apprentissage 573
1.1.2 Application à la classification 574
1.2 Rappels et notations 575
2 Le vote de plusieurs classificateurs 576
2.1 Le principe du vote 576
2.2 L'apprentissage à deux étages (*stacked generalization*) 576
3 Les codes correcteurs de classes 577
3.1 Présentation 577
3.2 Deux codages naturels 577
3.2.1 Une classe contre toutes les autres 577
3.2.2 Une classe contre une seule autre 578
3.2.3 Le cas $C = 4$ 579
3.3 Codes binaires 579
3.4 Décodage par marges 580
4 Le *boosting* d'un algorithme d'apprentissage 582
4.1 Le premier algorithme : boosting par sous-ensembles 582
4.2 Le boosting probabiliste et l'algorithme AdaBoost 583
4.3 Quelques interprétations de AdaBoost 585
4.3.1 AdaBoost généralise le premier algorithme de boosting . . . 585
4.3.2 AdaBoost est un algorithme d'optimisation 586
4.3.3 AdaBoost est (presque) un algorithme bayésien 586
4.4 Les propriétés de l'algorithme AdaBoost 587
4.4.1 Bornes sur l'erreur apparente et sur l'erreur réelle 587
4.4.2 AdaBoost et les marges 587
4.4.3 AdaBoost est (presque) un SVM 588
4.5 Généralisation à une classe, à plus de deux classes et à la régression . . 588
4.6 L'utilisation du boosting 589
4.7 Boosting et théorie *PAC* 589
5 Le *bagging* 590
6 Les forêts aléatoires (*random forests*) 591
7 L'apprentissage en cascade (*cascading*) 591

Copyright © 2010 Eyrolles.

1. Principes des méthodes par combinaison

1.1 Introduction

Il est rare qu'un décideur ait sous la main un expert omniscient et incontesté lui permettant d'opérer le meilleur choix. Il n'a souvent d'autres ressources que de consulter un comité d'experts

plus ou moins compétents puis de combiner leurs avis pour prendre sa décision. Mais cette combinaison de décisions est-elle meilleure que la décision qu'aurait pris le meilleur expert du comité ? Peut-on s'arranger pour rendre ce comité d'experts de plus en plus performant ?

Prenons un exemple non technique[1]. Soit un joueur de tiercé cherchant à maximiser ses gains. Il connaît un certain nombre d'« experts » des courses de chevaux. Aucun d'eux n'est capable d'expliciter complètement son expertise, mais, interrogé à propos d'un ensemble de courses, chacun d'eux peut fournir des règles grossières (par exemple : « au trot attelé, il faut parier sur le cheval ayant gagné le plus grand nombre de courses », ou : « en courses d'obstacles, il faut parier sur le cheval ayant la plus grande cote »). Prise isolément, chacune de ces règles est peu performante. On peut cependant raisonnablement penser qu'elles sont un peu meilleures que le hasard. De plus, si on interroge chaque expert sur des ensembles de courses différents, on peut obtenir plusieurs règles de ce type. Le joueur a maintenant deux questions à résoudre. D'abord, quels ensembles de courses devrait-il présenter à chaque expert en vue d'extraire les règles les plus intéressantes? Ensuite, comment doit-il combiner les avis des experts pour atteindre la meilleure décision ? La première question concerne le choix des exemples d'apprentissage soumis à l'apprenant. La deuxième concerne la manière de combiner l'avis d'apprenants (potentiellement différents) entraînés sur des échantillons différents.

Les recherches en apprentissage artificiel initiées au début des années 1990 montrent qu'il est possible d'atteindre une décision aussi précise que souhaitée par une combinaison judicieuse d'experts imparfaits mais correctement entraînés.

1.1.1 Le méta-apprentissage

Supposons que nous ayons une décision à prendre, par exemple une décision de classification : cet oiseau est-il une oie ou un cygne ? Nous savons qu'il existe plusieurs méthodes qui permettent d'élaborer une règle de classification par apprentissage, à partir de l'expérience donnée par des ensembles d'exemples supervisés. Disons que nous avons appris et testé un classificateur bayésien d'une part et un SVM d'autre part. La réponse de chacun des deux classificateurs sur un oiseau est un nombre réel : la probabilité P_1 qu'il soit une oie pour le premier classificateur (celle qu'il soit un cygne vaut $1 - \mathsf{P}_1$) et la marge M pour le second (positive quand le SVM décide que c'est une oie, négative sinon).

Pour combiner les deux décisions et tirer profit de notre double apprentissage, nous pouvons par exemple calculer la valeur $V = \mathsf{P}_1 - \frac{1}{2} + \alpha M$, avec α positif[2]. Si les deux classificateurs se décident pour une oie (respectivement pour un cygne), V sera certainement positive (respectivement négative), quelle que soit la valeur choisie pour α. Mais si les deux classificateurs prennent des décisions contradictoires, c'est en fonction de la valeur de α que sera prise la décision finale. Il est donc important de choisir α avec efficacité. On pourrait tenter de fixer sa valeur empiriquement, mais le mieux est certainement de ... l'apprendre ! Une idée simple pour le réaliser est par exemple de la fixer à une première valeur arbitraire, puis de la faire évoluer par descente de gradient en mesurant son efficacité sur un ensemble de validation.

L'apprentissage de α est un exemple de *méta-apprentissage*, puisque α n'est un paramètre ni du premier classificateur, ni du second, mais de leur combinaison linéaire pour obtenir un troisième classificateur. On dit que α est un *méta-paramètre* d'apprentissage. Remarquons que rien ne nous dit qu'il est effectivement possible de combiner les deux classificateurs pour en constituer un nouveau qui soit meilleur. Il est possible que le SVM soit tellement bien adapté au problème

[1] Repris de Freund et Schapire, les concepteurs du boosting, dans [FS99].

[2] Il serait certainement plus judicieux de calculer une telle combinaison linéaire entre des valeurs comparables et donc de normaliser la marge M entre 0 et 1. Mais ce n'est ici qu'un exemple d'école.

Copyright © 2010 Eyrolles.

et le classificateur bayésien tellement mal que le second ne puisse que dégrader le premier.

D'une manière générale, le méta-apprentissage peut se définir comme l'apprentissage du réglage de la combinaison de programmes d'apprentissage, ou comme l'apprentissage du choix entre plusieurs programmes d'apprentissage. Nous traiterons dans la suite de classificateurs, mais il est possible également de procéder au méta-apprentissage en régression. Comme nous le verrons, pour un problème à C classes, on peut soit combiner des classificateurs partiels, qui ne savent par exemple que distinguer une classe d'une autre, soit combiner des classificateurs complets. On sait aussi multiplier les apprentissages avec le même algorithme, mais en changeant les données (plus exactement en changeant la distribution de probabilités des exemples dans l'ensemble d'apprentissage), avant de combiner leurs résultats. Le regroupement des résultats se fait généralement par *vote pondéré*, c'est-à-dire par une combinaison linéaire des décisions suivie d'une décision ferme.

Le méta-apprentissage pose des problèmes techniques et théoriques très intéressants (voir par exemple la référence [VD02]). Nous le voyons dans ce chapitre essentiellement sous un aspect opérationnel.

1.1.2 Application à la classification

Dans le problème de la classification à C classes, on peut disposer d'un certains nombre d'experts spécialisés dans des tâches élémentaires consistant à prendre une décision sur deux classes, sans rien décider sur les autres. On peut imaginer de construire un classificateur très performant pour distinguer les caractères manuscrits 'o' des caractères 'p', mais qui soit incompétent sur les 's' ou les 'z'. De plus, techniquement parlant, il existe des méthodes d'apprentissage qui par nature ne savent traiter que des problèmes à deux classes : c'est le cas des *SVM* (en tout cas dans leur version classique) et des hyperplans, par exemple, comme de la plupart des méthodes symboliques. Apprendre à combiner efficacement des classificateurs à deux classes (entraînés sur des données différentes) pour résoudre un problème à C classes est un des problèmes de base du méta-apprentissage.

Une autre manière d'aborder le problème est de mener l'expérience suivante : sur un ensemble S, supervisé par C classes, apprenons un arbre de décision et un réseau connexionniste. Ensuite, pour un objet de classe inconnue, observons les décisions prises par ces deux classificateurs et par une troisième classification par k plus proches voisins. Que conclure si les trois classes prédites sont différentes ? En revanche, peut-on considérer qu'une décision unanime est très fiable ? Combiner des classificateurs différents, mais ayant acquis leurs connaissances sur les mêmes données, est un autre aspect du méta-apprentissage.

Le méta-apprentissage peut aussi consister à trouver la meilleure valeur de k pour traiter un problème par k plus proches voisins ou à régler l'élagage d'un arbre de décision. Comme on le sait, ces problèmes peuvent être abordés par l'utilisation d'un ensemble de validation (voir le chapitre 3).

Un autre aspect du méta-apprentissage se présente quand les données peuvent être décrites par plusieurs jeux d'attributs différents. Un exemple classique est de décrire une page Web soit par la fréquence des mots qu'elle utilise, soit par les adresses des pages auxquelles elle se refère (ou de celles qui la référencent ou des deux). Ce cas est traité sous le nom de co-apprentissage dans le chapitre 19. Il est en effet un peu différent des précédents et se rapproche de l'apprentissage non supervisé ou semi-supervisé.

Les paragraphes suivants mettent l'accent sur plusieurs techniques de méta-apprentissage :

- le *vote* de différents classificateurs appris sur les mêmes données et son extension à l'*apprentissage à deux étages* (*stacked generalization*) ;

Copyright © 2010 Eyrolles.

- une extension du cas précédent, le *codage correcteur de classes*, qui permet en particulier de combiner des classificateurs à deux classes en un classificateur à C classes ;
- le *dopage*, ou *boosting*, qui réutilise le même classificateur plusieurs fois en pondérant différemment les données d'apprentissage avant de combiner les résultats ;
- le *bagging*, qui en est une variante ;
- l'*apprentissage en cascade* (*cascading*) qui utilise différents classificateurs à la suite les uns des autres.

1.2 Rappels et notations

Rappelons d'abord quelques notations générales qui seront utiles dans ce chapitre. Nous disposons d'un ensemble d'apprentissage supervisé composé de m exemples, $\mathcal{S} = \{(\mathbf{x}_i, u_i)\}_{i=1,m}$.

La supervision d'un exemple est ici une classe parmi C (nous traitons seulement du problème de la classification ; l'extension à la régression est en général assez simple) : $u_i \in \{1, C\}$.

Les classes sont notées $\omega_1, \ldots \omega_C$. Pour exprimer que $u_i \in \omega_j$, on a deux méthodes. La première consiste à définir la supervision comme un nombre entier, et dans ce cas $u_i = j$ équivaut à $u_i \in \omega_j$; cette méthode ressemble à la supervision d'un problème de régression. Cependant, dans le cas de la classification, il est plus pratique de définir u_i comme un vecteur de taille C, dont seule la composante j vaut $+1$, toutes les autres prenant la valeur 0 (on utilise aussi la valeur -1 au lieu de 0).

Un classificateur $h \in \mathcal{H}$ produit un résultat $h(\mathbf{x}_i)$ sur un exemple, ou plus généralement un résultat $h(\mathbf{x})$ sur tout élément de l'espace de représentation $\mathcal{X}$. Ce résultat se représente en général par un vecteur $\boldsymbol{h}$ de $\mathbb{R}^C$, qui peut prendre deux aspects.

- Il peut être *binaire*, composé entièrement de 0 (ou de -1) sauf pour une coordonnée à $+1$, ce qui signifie que le classificateur a choisi la classe de $\mathbf{x}$ comme celle qui est désignée par cette coordonnée. C'est par exemple de cette manière que l'on peut écrire le résultat d'une décision par k plus proches voisins ou par un arbre de décision. Dans certains cas, il peut y avoir plusieurs coordonnées à $+1$, ce qui indique une décision ambigüe, ou toutes les coordonnées à 0 (ou à -1), ce qui indique le refus de décision (ou *rejet*).
- Il peut être *numérique*, composé de valeurs réelles. C'est par exemple le cas d'un classificateur bayésien (les valeurs sont alors comprises entre 0 et 1) ou d'un réseau connexionniste. Dans ce cas, la décision se fait en choisissant la classe de valeur maximale.

Dans le cas particulier où $C = 2$, que nous avons appelé apprentissage de *concept*, il est intéressant d'introduire la notion de *marge*, à condition que le vecteur de classification $\boldsymbol{h}$ soit numérique.

On dispose dans ce cas d'ensembles d'exemples et de contre-exemples $\mathcal{S} = \{(\mathbf{x}_i, u_i)\}_{i=1,m}$, avec $u_i \in \{+1, -1\}$. Un classificateur h est en général une application $\mathcal{X} \to \mathbb{R}$ qui attribue donc à tout point de l'espace de représentation $\mathcal{X}$ une valeur numérique. La marge d'un exemple $(\mathbf{x}, u)$ vis-à-vis de h est par définition la valeur $\text{marge}(\mathbf{x}, h) = u.h(\mathbf{x})$, qui doit être positive si et si seulement l'exemple est bien classé par l'hypothèse h. Il est souhaitable par ailleurs qu'il existe une relation monotone entre la valeur absolue de la marge et la distance à la surface séparatrice que définit h. De la sorte, la marge d'un exemple reflète non seulement sa classification par h, mais aussi la qualité de cette classification. Un exemple ayant une grande marge est classé de manière fiable par h.

Copyright © 2010 Eyrolles.

2. Le vote de plusieurs classificateurs

2.1 Le principe du vote

La manière la plus simple de combiner des classificateurs élémentaires est de les faire voter, c'est-à-dire de calculer une valeur sur les classes qui tienne compte de l'avis de chacun.

Supposons disposer de L classificateurs h_k, $k = 1, L$, chacun produisant sur un point à classer $\mathbf{x}$ un vecteur $\boldsymbol{h}_k$ de dimension C, dont la coordonnée courante est notée $\boldsymbol{h}_{kj}$, $j = 1, C$. Leur vote pondéré consiste à créer un vecteur de décision $\boldsymbol{H}$ dont chaque coordonnée est calculée par :

$$\boldsymbol{H}_j = \sum_{k=1}^{L} w_k \, \boldsymbol{h}_{kj}$$

où les valeurs w_k, $k = 1, L$ sont strictement positives et de somme égale à 1. Chacune d'elle quantifie le poids relatif de la décision du classificateur correspondant. La décision finale est donc une combinaison linéaire des décisions élémentaires.

Il est raisonnable de supposer que tous les classificateurs élémentaires produisent un vecteur de décision de même nature, binaire ou numérique. Dans le premier cas, chacun a pris une décision ferme et ce sont ces décisions qui sont combinées dans $\boldsymbol{H}$. Dans le second cas, la décision est entièrement reportée sur $\boldsymbol{H}$.

On connaît des résultats théoriques sur l'intérêt du vote, pour le cas $C = 2$. Le plus simple à énoncer, que nous donnons informellement, est conforme à l'intuition : si chaque classificateur est meilleur que le hasard, et si chacun est indépendant de chaque autre, alors l'efficacité du classificateur qui les fait voter avec des poids égaux augmente avec le nombre L de classificateurs élémentaires ([HS90]).

2.2 L'apprentissage à deux étages (*stacked generalization*)

La question restée en suspens dans le vote est évidemment celle du réglage des valeurs w_k. Elle est traitée dans ce paragraphe de manière élégante : dans l'apprentissage à deux étages, les sorties des classificateurs élémentaires sont en effet combinées par une fonction éventuellement non linéaire (un *méta-classificateur*) dont les paramètres sont eux-mêmes appris.

Par exemple, pour réaliser un classificateur à C classes, on peut utiliser $C(C-1)/2$ hyperplans destinés à séparer chaque paire de classes, produisant en sortie une valeur de marge, puis utiliser comme méta-classificateur un réseau connexionniste à $C(C-1)/2$ entrées et à C sorties, avec une couche cachée de taille intermédiaire (disons $3C$ pour fixer les idées). Le nombre total de paramètres à entraîner vaudra $dC(C-1)/2$ pour les hyperplans, plus environ $3\,C^3/2$ pour les poids des connexions menant à la couche cachée, plus $3\,C^2$ pour les poids menant à la couche de sortie. Si le nombre de classes n'est pas trop important, la recombinaison des marges se fera de manière efficace, puisqu'apprise par optimisation.

Cette technique peut s'appliquer au vote par combinaison linéaire du paragraphe précédent. Le méta-classificateur est alors un hyperplan dont les poids w_k peuvent être ajustés par gradient ou par toute technique d'optimisation de manière à minimiser l'erreur apparente ou une fonction de perte dépendant des marges des exemples (voir ci-dessous au paragraphe 3.4).

En pratique, la procédure d'apprentissage à deux étages se déroule en quatre étapes [Wol92b] :

1. Diviser l'échantillon d'apprentissage $\mathcal{S}$ en deux parties $\mathcal{S}_1$ et $\mathcal{S}_2$.
2. Réaliser l'apprentissage des L classificateurs élémentaires sur $\mathcal{S}_1$.

Copyright © 2010 Eyrolles.

3. Faire fonctionner ces classificateurs sur $\mathcal{S}_2$. Le résultat est, pour chaque objet de $\mathcal{S}_2$, un ensemble de L vecteurs binaires ou numériques.

4. Apprendre les paramètres du méta-classificateur sur un nouvel ensemble d'apprentissage dont les objets sont les résultats de l'étape précédente, supervisés par leur vraie classe.

3. Les codes correcteurs de classes

3.1 Présentation

Ce paragraphe décrit une implantation générale pour transformer un classificateur à deux classes en classificateur à C classes. Plus globalement, il montre comment on peut éclater un problème d'apprentissage à C classes en L sous-problèmes à deux classes avant de recombiner leurs décisions pour résoudre le problème initial.

Nous avons dans le chapitre 9, paragraphe 3.5, évoqué comment combiner plusieurs hyperplans pour résoudre un problème multiclasse. Un hyperplan sépare en effet l'espace de représentation en deux classes, par nature.

Nous avons fait allusion à deux solutions : la première consiste à définir C hyperplans en apprenant à chaque fois une classe contre la réunion des autres. Dans la seconde solution, on apprend $C(C-1)/2$ hyperplans, autant que de couples de classes. Dans le premier cas, les exemples du problème d'apprentissage sont les représentants d'une classe et les contre-exemples sont l'ensemble des représentants de toutes les autres classes. Dans le second, les exemples sont les représentants d'une classe et les contre-exemples sont les représentants d'une seule autre classe. Ensuite, dans un cas comme dans l'autre, une procédure de vote est mise en route.

Supposons donc disposer d'un ensemble de classificateurs élémentaires, qui s'intéressent à tout ou partie des données. Nous allons définir une *matrice de codage des classes*, qui va permettre de transformer l'ensemble des résultats obtenus sur les classificateurs élémentaires en une classification globale. Pour illustrer cela, nous reprenons l'exemple précédent dans les deux cas que nous avons décrits.

3.2 Deux codages naturels

3.2.1 Une classe contre toutes les autres

Notons H_j l'hyperplan obtenu par apprentissage en utilisant les exemples de la classe ω_j contre l'union des exemples des autres classes. Notons par la valeur $+1$ le fait qu'un point x soit classé par H_j comme appartenant à ω_j et par -1 la décision contraire.

Nous pouvons écrire l'ensemble des décisions sous la forme d'un vecteur v de taille C. Par exemple, le point indiqué par un carré sur la figure 17.1 produira le vecteur $\begin{pmatrix} +1 \\ +1 \\ -1 \end{pmatrix}$ puisqu'il est classé comme ω_1 par H_1, comme ω_2 par H_2 et comme non ω_3 par H_3. De même, le point indiqué par un triangle produira le vecteur $\begin{pmatrix} +1 \\ -1 \\ -1 \end{pmatrix}$.

La matrice W de codage des classes sera définie dans ce cas de manière simple par la matrice

Copyright © 2010 Eyrolles.

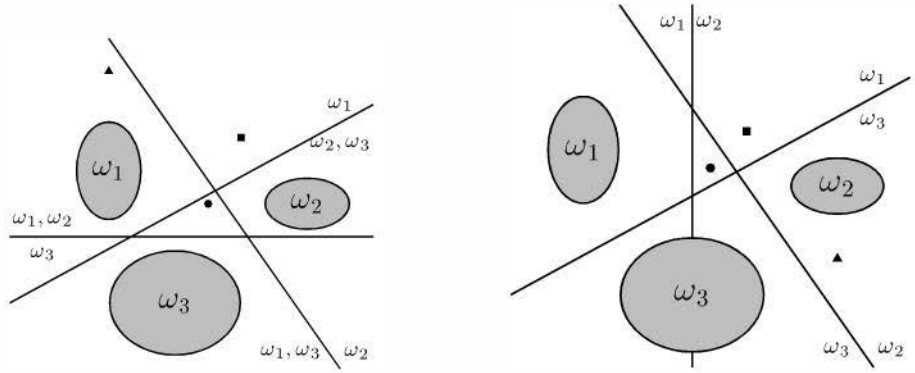


FIG. 17.1: ***Séparation linéaire à plus de deux classes.*** *À gauche, on sépare chaque classe de toutes les autres : il y a C hyperplans. Le point en triangle est attribué à la classe ω_1, le point en carré est ambigu entre ω_1 et ω_2, le point central est ambigu entre les trois classes. Sur les sept zones, quatre sont ambiguës. À droite, on sépare chaque classe de chaque autre : il y a $\frac{C(C-1)}{2}$ hyperplans. Le point en triangle et le point en carré sont attribués à la classe ω_2, le point central est ambigu entre les trois classes. Sur les sept zones, une seule est ambiguë.*

diagonale $C \times C$:

$$\begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{pmatrix}$$

Les valeurs de la troisième ligne indiquent par exemple que le premier classificateur doit répondre -1 sur les éléments des classes ω_1 et ω_2 et $+1$ sur ceux de ω_3.

L'attribution d'une classe au vecteur $\boldsymbol{v}$ se fait en calculant le vecteur $\boldsymbol{\omega}$ de taille C comme la multiplication de la matrice $\boldsymbol{W}$ par le vecteur $\boldsymbol{v}$. On obtient ainsi un vecteur $\boldsymbol{\omega} = \boldsymbol{W}\boldsymbol{v}$ dont la coordonnée de plus grande valeur indique la classe à choisir. Dans le cas du point indiqué par un triangle sur la figure 17.1, le calcul donne :

$$\boldsymbol{\omega} = \begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{pmatrix} \begin{pmatrix} +1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} +1 \\ -1 \\ -1 \end{pmatrix}$$

Le résultat indique que ce point doit être attribué sans ambiguïté à la classe ω_1.

Dans le cas du point indiqué par un carré, le calcul donne :

$$\boldsymbol{\omega} = \begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ +1 & +1 & -1 \end{pmatrix} \begin{pmatrix} +1 \\ +1 \\ -1 \end{pmatrix} = \begin{pmatrix} +1 \\ +1 \\ -1 \end{pmatrix}$$

Le résultat indique que ce point peut être attribué à la classe ω_1 ou à la classe ω_2, mais pas à la classe ω_3. La décision est ambigüe.

3.2.2 Une classe contre une seule autre

Notons maintenant H_{ij} l'hyperplan obtenu par apprentissage en utilisant les exemples de la classe ω_i contre ceux de la classe ω_j. Notons par la valeur $+1$ le fait qu'un point x soit classé par H_{ij} comme appartenant à ω_i et par -1 le fait qu'il est classé comme appartenant à ω_j.

Copyright © 2010 Eyrolles.

Nous pouvons écrire l'ensemble des décisions sous la forme d'un vecteur v de taille $C(C-1)/2 = 3$, en les rangeant dans l'ordre H_{12}, H_{13}, H_{23}. Par exemple, le point indiqué par un carré sur la figure 17.1 produira le vecteur $\begin{pmatrix} -1 \\ +1 \\ +1 \end{pmatrix}$ puisqu'il est classé comme ω_2 par H_{12}, comme ω_3 par H_{13} et comme ω_2 par H_{23}. De même, le point indiqué par un triangle produira le vecteur $\begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix}$.

La matrice $\boldsymbol{W}$ (de taille $C \times C(C-1)/2$) de codage des classes sera définie dans ce cas par :

$$\begin{pmatrix} +1 & +1 & 0 \\ -1 & 0 & +1 \\ 0 & -1 & -1 \end{pmatrix}$$

Par exemple, l'élément de la troisième ligne et deuxième colonne, de valeur -1, indique que la troisième classe ω_3 est classée du côté négatif par le second classificateur H_{13}. Les valeurs 0 indiquent que le classificateur ne fournit pas d'information sur la classe.

Dans le cas du point indiqué par un triangle, le calcul donne ici :

$$\omega = \begin{pmatrix} +1 & +1 & 0 \\ -1 & 0 & +1 \\ 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix} = \begin{pmatrix} -2 \\ 2 \\ 0 \end{pmatrix}$$

Le résultat indique que ce point doit être attribué sans ambiguïté à la classe ω_2.

Dans le cas du point indiqué par un carré :

$$\omega = \begin{pmatrix} +1 & +1 & 0 \\ -1 & 0 & +1 \\ 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ +1 \\ +1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ -2 \end{pmatrix}$$

Ce qui indique que ce point doit aussi être attribué à la classe ω_2.

3.2.3 Le cas $C = 4$

Pour éviter une éventuelle confusion, dûe au fait que pour $C = 3$ on a aussi $C(C-1)/2 = 3$, regardons le cas $C = 4$. Dans le premier cas (une classe contre toutes les autres) il y a alors quatre hyperplans, le vecteur v est de dimension 4, la matrice $\boldsymbol{W}$ reste diagonale de taille $C \times C = 4 \times 4$.

Dans le second cas (une classe contre chaque autre), il y a $4(4-1)/2 = 6$ hyperplans, le vecteur v est de dimension 6 et la matrice $\boldsymbol{W}$ est de taille 4×6. Dans sa colonne correspondant à H_{ij}, il y a la valeur $+1$ à la ligne de rang i, -1 à celle de rang j et 0 partout ailleurs :

$$\begin{pmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

3.3 Codes binaires

Les deux cas particuliers précédents sont extrêmes et présentent des avantages et des inconvénients inverses. Dans le premier cas, les ambiguïtés de classement sont nombreuses, mais le calcul est rapide. Ajoutons un autre inconvénient, moins apparent : les apprentissages déséquilibrent le

Copyright © 2010 Eyrolles.

nombre d'exemples et de contre-exemples, ce qui peut avoir une influence néfaste si les probabilités *a priori* des classes sont estimées à partir des fréquences dans l'échantillon d'apprentissage. On peut se reporter aux paragraphes 6.1 du chapitre 3 et 1.5 du chapitre 15 pour approfondir ce point. En revanche, pour l'apprentissage du type « une classe contre chaque autre », le nombre de classificateurs à apprendre est important, mais le classement est en revanche rarement ambigu.

Il existe plusieurs techniques pour généraliser les deux approches précédentes. Une première idée est de fixer *a priori* le nombre L de classificateurs élémentaires. On peut supposer qu'ils produisent comme dans le cas précédent un résultat valant seulement -1 ou $+1$. Cette hypothèse sera relâchée dans le paragraphe suivant.

On va donc construire une matrice $\boldsymbol{W}$ de taille $C \times L$, composée de -1 et de $+1$ (le cas où certaines valeurs de cette matrice sont égales à 0 sera lui aussi traité dans le paragraphe suivant) et procéder comme précédemment : la classification d'un objet $\mathbf{x}$ par les classificateurs élémentaires H_i, $i = 1, L$) fournit un vecteur v de dimension L. On calcule ensuite $\omega = \boldsymbol{W}v$ et on choisit la classe de plus grande valeur dans le vecteur ω de dimension C.

Et maintenant, comment choisir la matrice $\boldsymbol{W}$? Elle doit répondre à deux exigences.

1. Il faut que ses colonnes soient aussi différentes que possible les unes des autres, pour que les tâches d'apprentissage réalisées par les classificateurs élémentaires soient également aussi différentes que possible.

2. Il faut aussi que ses lignes soient à distance maximale, de façon à ce qu'un classificateur élémentaire qui prend une décision erronée puisse être corrigé par les autres.

Autrement dit, il faut maximiser la somme des distances inter-lignes et des distances inter-colonnes. La distance entre deux vecteurs peut être simplement calculée comme la distance de Hamming : la somme sur les composantes de la distance élémentaire entre deux composantes, qui est elle-même fixée à 1 si elles sont différentes, à 0 dans le cas contraire.

Ces constatations rapprochent la construction de $\boldsymbol{W}$ de celle des matrices utilisées dans les techniques de *codage correcteur* en communications numériques, ce qui explique le titre de ce paragraphe. C'est dans cette discipline que l'on a en effet étudié, pour des raisons analogues, la construction de matrices binaires sous les mêmes contraintes. Différents algorithmes ont été proposés pour résoudre ce problème d'optimisation particulier.

L'inventeur de la technique des codes correcteurs de classes, T. Dietterich [DB95], propose sur son site Internet un grand nombre de matrices $\boldsymbol{W}$ répondant aux contraintes ci-dessus, pour des valeurs diverses de L et C. R. Schapire *et al.* [ASS00] ont présenté cette méthodologie sous le nom de *décodeur de Hamming*, avant d'en donner l'extension présentée au paragraphe suivant.

3.4 Décodage par marges

Le défaut de la méthode précédente est de n'utiliser pour chaque élément de la matrice $\boldsymbol{W}$ qu'une réponse binaire ($+1$ ou -1). Comme on l'a vu, il est d'abord souhaitable d'ajouter des valeurs 0 : une telle valeur signifie que le classificateur élémentaire de la colonne n'a pas d'avis sur la classe correspondant à la ligne. Un autre point à considérer est que la plupart des classificateurs binaires fournissent une réponse numérique non binaire. Par exemple, un classificateur bayésien H_i estime la probabilité $P_i = P(\mathbf{x}|h_i)$ qu'un objet $\mathbf{x}$ réponde à l'hypothèse h_i.

La valeur m_j est interprétable en terme de *marge*, une notion que nous avons rappelée au début de ce chapitre. Pour un ensemble d'exemples et de contre-exemples $\mathcal{S} = \{(\mathbf{x}_i, u_i)\}_{i=1,m}$, avec $u_i \in \{+1, -1\}$ et $h : \mathcal{X} \rightarrow \mathbb{R}$ un classificateur, la marge d'un exemple $(\mathbf{x}, u)$ vis-à-vis de h est, rappelons-le, la valeur $\text{marge}(\mathbf{x}, h) = u.h(\mathbf{x})$.

Copyright © 2010 Eyrolles.

Dans cette optique, l'erreur apparente de h sur $\mathcal{S}$ s'écrit :

$$\frac{1}{m}\sum_{i=1}^{m} \text{VRAI}(\text{marge}(\mathbf{x}_i, h) \leq 0)$$

avec VRAI(P) = 1 si le prédicat P est vrai, 0 sinon.

Le principe *ERM* (voir les chapitres 2 et 21) énonce qu'il faut minimiser l'erreur apparente pour chercher le meilleur h dans son ensemble $\mathcal{H}$. Cependant, comme il est difficile de le faire exactement (l'expression mathématique de l'erreur apparente n'est pas dérivable), les algorithmes minimisent donc plutôt en pratique des critères fondés sur une *fonction de perte* calculée à partir d'une fonction $\mathcal{L}$ de la marge, continue et monotone croissante de $\mathbb{R}$ dans $\mathbb{R}^+$. Sur un exemple $\mathbf{x}_i$, pour une hypothèse h, cette fonction de perte vaut $\mathcal{L}(\text{marge}(\mathbf{x}_i, h)).h(\mathbf{x}_i)$. La perte moyenne apparente vaut donc, pour une hypothèse h donnée :

$$\frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\text{marge}(\mathbf{x}_i, h)).h(\mathbf{x}_i)$$

Ceci posé, une matrice de codage peut maintenant être définie comme étant de dimension $C \times L$ et composée d'éléments valant -1, 0 ou $+1$. Les composantes du vecteur $\boldsymbol{v}$ prennent désormais des valeurs continues (ce sont des marges) et le calcul $\boldsymbol{\omega} = \boldsymbol{W}\boldsymbol{v}$ fournit également un vecteur avec une valeur réelle pour chaque classe. C'est naturellement la classe de valeur maximale qui sera choisie par la règle de classification. Cette approche est appelée *décodage par marges*, ou *décodage par fonction de perte*.

On peut associer une perte moyenne apparente à une matrice de codage, puisqu'elle est composée par un ensemble de L hypothèses $h_1, \ldots, h_L$, par la formule :

$$p(\boldsymbol{W}) = \frac{1}{mL}\sum_{i=1}^{m}\sum_{j=1}^{L} \mathcal{L}(m(\mathbf{x}_i, h_j)).h_j(\mathbf{x}_i)$$

Il s'agit maintenant de construire une matrice $\boldsymbol{W}$ en suivant les mêmes principes que précédemment : maximiser la distance entre ses lignes comme entre ses colonnes. La distance de Hamming n'est plus utilisable, puisque $\boldsymbol{W}$ n'est plus une matrice binaire. Il est cependant facile de la généraliser. Posons que la distance entre deux 0 vaut 1/2, celle entre deux $+1$ ou deux -1 vaut 0 et celle entre $+1$ et -1 (ou le contraire) vaut 1. On peut aisément vérifier que la distance entre deux vecteurs $\boldsymbol{u}$ et $\boldsymbol{v}$ de dimension d composés de -1, de 0 et de $+1$, définie comme la somme des distances de leurs composantes, s'exprime alors par :

$$\Delta(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{d} \frac{1 - \boldsymbol{u}_i.\boldsymbol{v}_i}{2} = \frac{d - \boldsymbol{u}.\boldsymbol{v}}{2}$$

La puissance de correction $\rho(\boldsymbol{W})$ d'une matrice $\boldsymbol{W}$ peut alors se mesurer comme la distance minimale entre deux lignes distinctes de cette matrice. On peut calculer que cette valeur vaut 2 dans le cas « une classe contre toutes les autres » et $\frac{C^2-C+2}{4}$ dans le cas « chaque classe contre chaque autre ». L'utilité de cette notion est qu'elle est reliée à l'erreur apparente commise par le classificateur : il a été montré [ASS00] que l'erreur apparente est en effet bornée supérieurement par l'expression :

$$\frac{L}{\mathcal{L}(0)}\frac{p(\boldsymbol{W})}{\rho(\boldsymbol{W})}$$

Copyright © 2010 Eyrolles.

où $\mathcal{L}(0)$ est une constante qui désigne la perte moyenne apparente de la matrice entièrement composée de valeurs 0.

La borne est d'autant plus petite (et le codage d'autant meilleur) que $p(\boldsymbol{W})$ est petit, ce qui est naturel. Cependant, le facteur $\rho(\boldsymbol{W})$ croît malheureusement dans le même sens que $p(\boldsymbol{W})$: la puissance de correction d'une matrice $\boldsymbol{W}$, comme sa perte moyenne, est d'autant plus grande que la proportion de 0 est grande. Il faut donc établir un compromis.

4. Le *boosting* d'un algorithme d'apprentissage

Le mot *boosting*[3] s'applique à des méthodes générales capables de produire des décisions très précises (au sens d'une fonction de perte) à partir d'un ensemble de règles de décision « faibles », c'est-à-dire dont la seule garantie est qu'elles soient un peu meilleures que le hasard. Ces méthodes s'appliquent aussi bien à l'estimation de densité qu'à la régression ou à la classification. Pour simplifier, nous nous concentrons ici sur la tâche de classification binaire.

Dans sa version « par sous-ensembles », cette technique fait produire à l'algorithme trois résultats selon la partie de l'ensemble d'apprentissage sur laquelle il apprend, puis combine les trois apprentissages réalisés pour fournir une règle de classification plus efficace. Examinons d'abord cette technique avant de voir comment la généraliser à l'aide de distributions de probabilités sur les exemples.

4.1 Le premier algorithme : boosting par sous-ensembles

Schapire [Sch90] a développé le premier algorithme de boosting pour répondre à une question de Kearns : est-il possible de rendre aussi bon que l'on veut un algorithme d'apprentissage « faible », c'est-à-dire un peu meilleur que le hasard ? Shapire montra qu'un algorithme faible peut toujours améliorer sa performance en étant entraîné sur trois échantillons d'apprentissage bien choisis. Nous ne nous intéressons ici qu'à des problèmes de classification binaire.

L'idée est d'utiliser un algorithme d'apprentissage qui peut être de nature très diverse (un arbre de décision, une règle bayésienne de classification, une décision dépendant d'un hyperplan, etc.) sur trois sous-ensembles d'apprentissage.

1. On obtient d'abord une première hypothèse h_1 sur un sous-échantillon $\mathcal{S}_1$ d'apprentissage de taille $m_1 < m$ (m étant la taille de $\mathcal{S}$ l'échantillon d'apprentissage disponible).
2. On apprend alors une deuxième hypothèse h_2 sur un échantillon $\mathcal{S}_2$ de taille m_2 choisi dans $\mathcal{S} - \mathcal{S}_1$ dont la moitié des exemples sont mal classés par h_1.
3. On apprend finalement une troisième hypothèse h_3 sur m_3 exemples tirés dans $\mathcal{S} - \mathcal{S}_1 - \mathcal{S}_2$ pour lesquels h_1 et h_2 sont en désaccord.
4. L'hypothèse finale est obtenue par un vote majoritaire des trois hypothèses apprises :

$$H = \text{vote majoritaire}(h_1, h_2, h_3)$$

Le théorème de Schapire sur la « force de l'apprentissage faible » prouve que H a une performance supérieure à celle de l'hypothèse qui aurait été apprise directement sur l'échantillon $\mathcal{S}$.

Une illustration géométrique du boosting selon cette technique de base est donnée dans les figures 17.2, 17.3 et 17.4.

[3] La traduction littérale de ce mot est « stimulation » ou « amplification » (pourquoi pas « dopage » ?) ; le terme anglais n'est jamais traduit dans le contexte de l'apprentissage.

Copyright © 2010 Eyrolles.

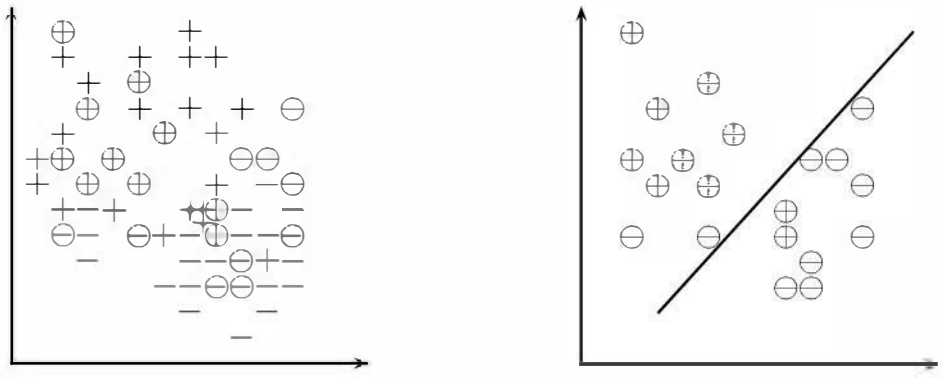


FIG. 17.2: *À gauche : l'ensemble d'apprentissage $\mathcal{S}$ et le sous-ensemble $\mathcal{S}_1$ (points entourés). À droite : l'ensemble $\mathcal{S}_1$ et la droite $\mathcal{C}_1$ apprise sur cet ensemble.*

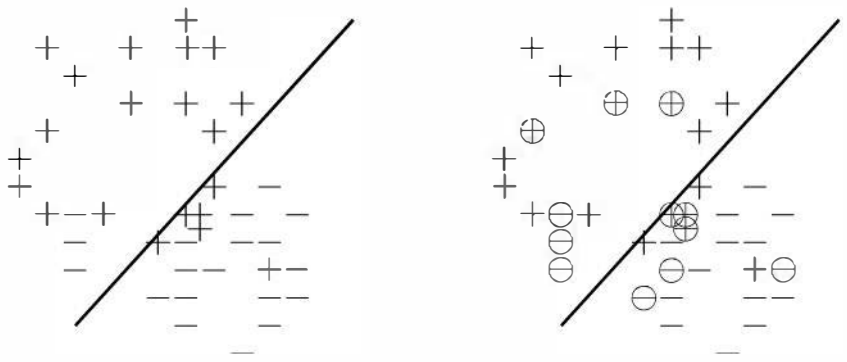


FIG. 17.3: *À gauche : l'ensemble $\mathcal{S} - \mathcal{S}_1$ et la droite $\mathcal{C}_1$ apprise sur $\mathcal{S}_1$. À droite : un ensemble $\mathcal{S}_2$ inclus dans $\mathcal{S} - \mathcal{S}_1$ parmi les plus informatifs pour $\mathcal{C}_1$ (points entourés).*

Idéalement, les trois ensembles d'exemples extraits de $\mathcal{S}$ devraient le vider de tous ses exemples, ce qui revient à dire que la somme des valeurs m_1, m_2 et m_3 doit approcher m. C'est la façon de tirer un profit maximal de $\mathcal{S}$. Mais on conçoit que ce réglage ne soit pas forcément facile à obtenir en pratique : si l'algorithme $\mathcal{A}$ est performant sur $\mathcal{S}$, m_2 pourra être pris bien inférieur à m_1, alors que la proportion pourrait être inverse si $\mathcal{A}$ est seulement un peu meilleur qu'un tirage de classe au hasard. En général, on règle empiriquement les proportions des trois ensembles en effectuant plusieurs essais, jusqu'à ce que tous les éléments de $\mathcal{S}$ ou presque participent au processus.

On peut utiliser récursivement la méthode et procéder avec neuf sous-ensembles, vingt-sept sous-ensembles, etc. Mais la meilleure généralisation est de faire glisser la notion de fonction caractéristique (qui vaut 1 sur les points d'un sous-ensemble et 0 partout ailleurs) vers celle de distribution de probabilités sur les points de l'ensemble d'apprentissage. Cette technique a déjà été employée pour les fenêtres de Parzen (chapitre 15). C'est ce que réalise l'algorithme que nous présentons maintenant.

4.2 Le boosting probabiliste et l'algorithme ADABOOST

Trois idées fondamentales sont à la base des méthodes de boosting probabiliste :

1. L'utilisation d'un comité d'experts spécialisés que l'on fait voter pour atteindre une déci-

Copyright © 2010 Eyrolles.

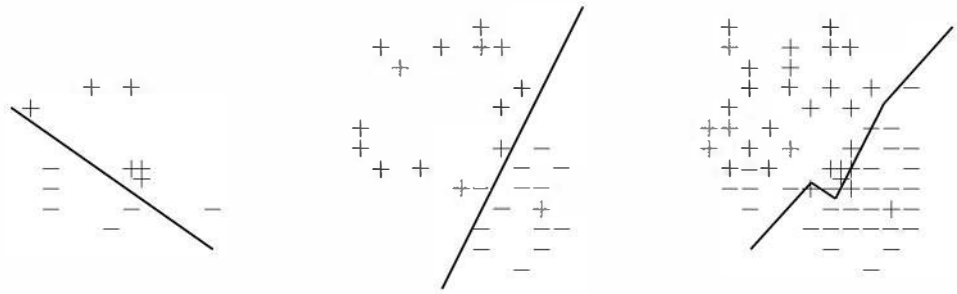


FIG. 17.4: *À gauche : l'ensemble $\mathcal{S}_2$ et la droite séparatrice $\mathcal{C}_2$ apprise sur cet ensemble. Au centre : l'ensemble $\mathcal{S}_3 = \mathcal{S} - \mathcal{S}_1 - \mathcal{S}_2$ et la droite séparatrice $\mathcal{C}_3$ apprise sur cet ensemble. À droite : l'ensemble $\mathcal{S}$ et la combinaison des 3 droites séparatrices apprises sur cet ensemble.*

sion.

2. La pondération adaptative des votes par une technique de mise à jour multiplicative.
3. La modification de la distribution des exemples disponibles pour entraîner chaque expert, en surpondérant au fur et à mesure les exemples mal classés aux étapes précédentes.

L'algorithme le plus pratiqué s'appelle ADABOOST (pour *adaptive boosting*). L'une des idées principales (voir l'algorithme 33) est de définir à chacune de ses étapes $1 \leq t \leq T$, une nouvelle distribution D_t de probabilités *a priori* sur les exemples d'apprentissages en fonction des résultats de l'algorithme à l'étape précédente. Le poids à l'étape t d'un exemple $(\mathbf{x}_i, u_i)$ d'indice i est noté $p_t(i)$. Initialement, tous les exemples ont un poids identique, puis à chaque étape, les poids des exemples mal classés par l'apprenant sont augmentés, forçant ainsi l'apprenant à se concentrer sur les exemples difficiles de l'échantillon d'apprentissage.

Algorithme 33 : ADABOOST **dans le cas d'un apprentissage de concept**

début
  $\mathcal{S} = \{(\mathbf{x}_1, u_1), \ldots, (\mathbf{x}_m, u_m)\}$, avec $u_i \in \{+1, -1\}, i = 1, m$
  pour tous les i=1,m **faire** $p_0(\mathbf{x}_i) \leftarrow 1/m$ $t \leftarrow 0$
  pour $t \leq T$ **faire**
    Tirer un échantillon d'apprentissage $\mathcal{S}_t$ dans $\mathcal{S}$ selon les probabilités p_t
    Apprendre une règle de classification h_t sur $\mathcal{S}_t$ par l'algorithme $\mathcal{A}$
    Soit ε_t l'erreur apparente de h_t sur $\mathcal{S}_t$. Calculer $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$
    pour tous les $i = 1, m$ **faire**
      $p_{t+1}(\mathbf{x}_i) \leftarrow \frac{p_t(\mathbf{x}_i)}{Z_t} e^{-\alpha_t}$ si $h_t(\mathbf{x}_i) = u_i$ (bien classé par h_t)
      $p_{t+1}(\mathbf{x}_i) \leftarrow \frac{p_t(\mathbf{x}_i)}{Z_t} e^{+\alpha_t}$ si $h_t(\mathbf{x}_i) \neq u_i$ (mal classé par h_t).
        (Z_t est une valeur de normalisation telle que $\sum_{i=1}^{m} p_t(\mathbf{x}_i) = 1$)
    fin
    $t \leftarrow t + 1$
  fin pour
  Fournir en sortie l'hypothèse finale : $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$
fin

Copyright © 2010 Eyrolles.

À chaque étape t, l'apprenant cherche une hypothèse h_t $\mathcal{X} \rightarrow \{-1, +1\}$ bonne pour la distribution D_t sur $\mathcal{X}$. La performance de l'apprenant est mesurée par l'erreur apparente[4] :

$$\varepsilon_t = p_t[h_t(\mathbf{x}_i) \neq u_i] = \sum_{i\,:\,h_t(\mathbf{x}_i)\neq u_i} p_t(i)$$

On note que l'erreur est mesurée en fonction de la distribution D_t sur laquelle l'apprenant est entraîné. En pratique, soit les poids des exemples sont effectivement modifiés, soit c'est la probabilité de tirage des exemples qui est modifiée et l'on utilise un tirage avec remise (*bootstrap*).

Chaque hypothèse h_t apprise est affectée d'un poids α_t mesurant l'importance qui sera donnée à cette hypothèse dans la combinaison finale. Ce poids est positif si $\varepsilon_t \leq 1/2$ (on suppose ici que les classes « + » et « − » sont équiprobables et donc que l'erreur d'une décision aléatoire est de 1/2). Plus l'erreur associée à l'hypothèse h_t est faible, plus celle-ci est dotée d'un coefficient α_t important[5].

L'examen des formules de mise à jour des poids des hypothèses dans l'algorithme 33 suggère que vers la fin de l'apprentissage, le poids des exemples difficiles à apprendre devient largement dominant. Si une hypothèse performante sur ces exemplespeut être trouvée (c'est-à-dire avec $\varepsilon_t \approx 0$), elle sera alors dotée d'un coefficient α_t considérable. L'une des conséquences possibles est que les exemples bruités, sur lesquels finit par se concentrer l'algorithme, perturbent gravement l'apprentissage par boosting. C'est en effet ce qui est fréquemment observé.

À la fin de cet algorithme, chaque règle de classification h_t est pondérée par une valeur α_t calculée en cours de route. La classification d'un nouvel exemple (ou des points de $\mathcal{S}$ pour obtenir l'erreur apparente) s'opère en utilisant la règle :

$$H(\mathbf{x}) = \text{signe}\left(\sum_{t=1}^{t=T} \alpha_t\, h_t(\mathbf{x})\right) \tag{17.1}$$

En un sens, on voit que le boosting construit l'hypothèse finale comme une combinaison linéaire d'une base de fonctions, dont les éléments sont les hypothèses h_t. On retrouve là un thème fréquent dans les techniques d'apprentissage (par exemple les SVM, les méthodes d'approximation bayésiennes, etc.).

4.3 Quelques interprétations de AdaBoost

4.3.1 AdaBoost généralise le premier algorithme de boosting

Le premier algorithme de boosting décrit au paragraphe 4.1 utilisait un vote sur trois classificateurs appris sur trois sous-ensembles d'apprentissage. D'une certaine façon, il effectuait trois itérations d'AdaBoost avec des valeurs binaires pour les poids p_1, p_2 et p_3. De plus, il choisissait le second sous-ensemble comme des exemples pour lesquels le premier classificateur avait la performance d'un tirage aléatoire et le troisième comme des exemples pour lesquels le premier et le second classificateurs étaient en désaccord. AdaBoost généralise cette idée : à chaque étape, le calcul de la nouvelle pondération est mené de manière à ce que le nouvel ensemble d'apprentissage soit mal classé par la combinaison linéaire des classificateurs précédents.

[4] Nous reprenons provisoirement la notation ε_t des auteurs de AdaBoost pour l'erreur apparente au lieu de R_{Emp}.

[5] Le terme AdaBoost vient du fait que contrairement aux algorithmes de boosting antérieurs, il n'est pas nécessaire de fournir la borne d'amélioration γ *a priori*. AdaBoost s'adapte à l'erreur de chaque hypothèse faible.

Copyright © 2010 Eyrolles.

Plus précisément, après t étapes, le classificateur est défini par : $H_t(\mathbf{x}) = \text{signe}\left(\sum_{r=1}^{Q} \alpha_r h_r(\mathbf{x})\right)$. Ensuite, l'algorithme calcule une nouvelle distribution de probabilités $\mathbf{D}_{t+1}$, qui affecte à chaque élément de $\mathcal{S}_t$ la probabilité $p_{t+1}(\mathbf{x}_i)$, selon la formule donnée dans l'algorithme 33.

Schapire et Singer [SS98a, FHT98] ont montré le résultat suivant, qui prouve en effet que les poids sont réactualisés d'une manière bien particulière :

> h_t classerait $\mathcal{S}$ pondéré par $\mathbf{D}_{t+1}$ comme le hasard.

4.3.2 AdaBoost est un algorithme d'optimisation

Rappelons que d'une manière générale, l'apprentissage par minimisation du risque empirique (ERM) consiste à trouver l'hypothèse (dans une certaine famille) qui minimise une fonction de perte sur l'ensemble d'apprentissage. Il est cependant souvent difficile d'utiliser directement l'erreur apparente comme fonction de perte, car elle varie de manière non continue.

L'erreur apparente (ou *risque empirique*) à la fin de AdaBoost est égale à[6] $\varepsilon_T = \frac{1}{m}|\{i : H(\mathbf{x}_i) \neq u_i\}|$. En notant $f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$, il a été montré qu'elle est bornée de la façon suivante (ce résultat vient aussi de [SS98a]) :

$$\varepsilon_T \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-u_i f(\mathbf{x}_i)) = \sum_{t=1}^{T} Z_t$$

Nous n'avons pas encore explicité Z_t, le facteur de normalisation introduit dans l'algorithme. Il est facile de calculer que :

$$Z_t = \sum_{i=1}^{m} \mathbf{D}_t(i) \exp(-\alpha_t u_i h_t(\mathbf{x}_i))$$

La borne sur l'erreur apparente suggère donc de minimiser cette erreur par un algorithme glouton, en choisissant à chaque itération α_t et h_t de façon à minimiser Z_t. Et on peut démontrer que c'est bien ce que fait AdaBoost, en choisissant α_t égal à $\frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$; cet algorithme minimise donc une expression qui borne supérieurement l'erreur apparente et qui vaut, rappelons-le :

$$\frac{1}{m} \sum_{i=1}^{m} \exp\left(-u_i \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right)$$

4.3.3 AdaBoost est (presque) un algorithme bayésien

Il est également possible de relier AdaBoost à une décision bayésienne. Notons $\mathbf{P}_f[u = +1|\mathbf{x}]$ la probabilité qu'un vecteur $\mathbf{x}$ soit classé $+1$. On peut montrer qu'une bonne estimation de cette probabilité s'écrit :

$$\mathbf{P}_f[u = +1|\mathbf{x}] = \frac{\exp(f(\mathbf{x}))}{\exp(f(\mathbf{x})) + \exp(-f(\mathbf{x}))}$$

avec comme ci-dessus $f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$.

On peut pousser le raisonnement plus loin et modifier l'algorithme AdaBoost pour qu'il produise exactement une valeur de probabilité d'appartenance aux deux classes. C'est ce qui a été fait dans l'algorithme LogiBoost [FHT98].

[6] $|E|$ dénote ici le nombre d'éléments de l'ensemble fini E.

Copyright © 2010 Eyrolles.

4.4 Les propriétés de l'algorithme AdaBoost

4.4.1 Bornes sur l'erreur apparente et sur l'erreur réelle

Commençons l'analyse de l'erreur apparente de AdaBoost. Écrivons cette erreur ε_t de h_t sous la forme : $\frac{1}{2} - \gamma_t$, où γ_t mesure l'amélioration apportée par l'hypothèse h_t par rapport à l'erreur de base 1/2. Freund et Shapire ont montré dans [FS97] que l'erreur apparente ε_T de l'hypothèse finale H est bornée par :

$$\prod_{t=1}^{T} \left[2\sqrt{\varepsilon_t(1-\varepsilon_t)} \right] = \prod_{t=1}^{T} \sqrt{1-4\gamma_t{}^2} \leq exp\left(-2\sum_t \gamma_t{}^2 \right)$$

Ainsi, si chaque hypothèse faible est légèrement meilleure que le hasard, ($\gamma_t \geq \gamma > 0$), alors l'erreur apparente diminue exponentiellement avec t.

L'erreur en généralisation de l'hypothèse finale H peut être bornée par une expression faisant intervenir son erreur apparente ε_T, le nombre d'exemples d'apprentissage m, la dimension de Vapnik-Chervonenkis $d_{\mathcal{H}}$ de l'espace d'hypothèses (voir les chapitres 2 et 21 et le nombre T d'étapes de boosting [FS97] :

$$R_{\text{Réel}}(H) = \varepsilon_T + \mathcal{O}\left(\sqrt{\frac{T \cdot d_{\mathcal{H}})}{m}} \right)$$

Cette borne suggère que le boosting devrait tendre à surapprendre lorsque T devient grand, puisque le deuxième terme devient grand. Si cela arrive effectivement parfois, il a été observé empiriquement que souvent cela ne se produit pas. De fait, il apparaît même fréquemment que le risque réel tend à diminuer même lontemps après que le risque empirique soit devenu stable, voire nul. Cette observation *a priori* énigmatique s'éclaircit si l'on établit un lien entre le boosting et les *méthodes à vaste marge* (voir le chapitre 9 sur les SVM).

4.4.2 AdaBoost et les marges

À la fin de AdaBoost, la *marge* d'un exemple $(\mathbf{x}, u)$, avec $u = \pm 1$ désignant la classe, s'exprime par :

$$\text{marge}(\mathbf{x}, u) = \frac{u \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})}{\sum_{t=1}^{T} \alpha_t}$$

Ce nombre, sous sa forme normalisée donnée ici, est compris dans l'intervalle $[-1, +1]$ et est positif seulement si H classifie correctement l'exemple. Nous savons que la marge peut être interprétée comme une mesure de confiance dans la prédiction. Il a été prouvé que l'erreur en généralisation peut être bornée avec une grande probabilité, pour tout $\theta > 0$, par :

$$R_{\text{Réel}}(H) \leq \mathbf{P}[\text{marge}(\mathbf{x}, u) \leq \theta] + \mathcal{O}\left(\sqrt{\frac{d_{\mathcal{H}}}{m\theta^2}} \right)$$

On note que cette borne est maintenant indépendante de T, le nombre d'étapes de boosting. De plus, il a été montré que le boosting cherche effectivement à augmenter la marge sur les exemples ; la raison en est qu'il se concentre sur les exemples difficiles à classer, c'est-à-dire sur les exemples dont la marge est la plus faible. Le fait que le risque réel tend à diminuer même lontemps après que le risque empirique soit devenu stable signifie que les marges continuent à croître, même si le classement de l'ensemble d'apprentissage reste inchangé.

Copyright © 2010 Eyrolles.

4.4.3 AdaBoost est (presque) un SVM

AdaBoost produit un classificateur qui s'écrit $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{t=T} \alpha_t h_t(\mathbf{x})\right)$. Il est intéressant d'interpréter ce classificateur comme un hyperplan en dimension T, dont les paramètres sont le vecteur des valeurs α_t, pour t de 1 à T. Les T axes de ce nouvel espace de représentation sont calculés par la transformation de X par les T fonctions h_t.

Sous cet angle, AdaBoost ressemble à un classificateur SVM. On a vu au chapitre 14 qu'en effet les SVM calculaient un hyperplan optimal dans un espace transformé (de grande dimension en général). Le type de transformation n'est pas le même, certes, mais l'hyperplan calculé par AdaBoost a-t'il quelque chose à voir avec le séparateur SVM ?

Comme on l'a vu au paragraphe précédent, AdaBoost est, comme les SVM, un algorithme de maximisation des marges. Plus précisément, en notant $\boldsymbol{h}$ le vecteur de dimension T composé des hypothèses $(h_1, \ldots, h_T)$ et $\boldsymbol{\alpha}$ le vecteur transposé de dimension T composé des valeurs $(\alpha_1, \ldots, \alpha_T)$, on peut établir AdaBoost cherche la quantité :

$$\max_{\boldsymbol{\alpha}} \min_{i} \frac{(\boldsymbol{\alpha}.\boldsymbol{h}(\mathbf{x}_i))u_i}{||\boldsymbol{\alpha}||_1.||(\boldsymbol{h})(\mathbf{x})_i)||_\infty}$$

avec $||\boldsymbol{\alpha}||_1 = \sum_{i=1}^{T} |\alpha_i|$ et $||(\boldsymbol{h})(\mathbf{x})_i)||_\infty = \max_{i=1}^{T} |(h_t)(\mathbf{x})_i)|$.

La quantité recherchée par un SVM peut s'écrire de manière analogue :

$$\max_{\boldsymbol{\alpha}} \min_{i} \frac{(\boldsymbol{\alpha}.\boldsymbol{h}(\mathbf{x}_i))u_i}{||\boldsymbol{\alpha}||_2.||(\boldsymbol{h})(\mathbf{x})_i)||_2}$$

La ressemblance, au moins formelle, est donc finalement très grande. Les différences sont les suivantes :

- Comme les normes utilisées (L_2 pour les SVM et L_1 et L_∞ pour AdaBoost) sont différentes, les espaces explorés le sont donc aussi.
- Les contraintes de l'optimisation sont quadratiques pour les SVM et linéaires pour le boosting.
- La recherche est globale pour les SVM ce qui est rendu possible par l'astuce des fonctions noyau permettant de faire des calculs virtuels simples dans des espaces de très grande dimension. Le boosting effectue une recherche locale gloutonne : il traite une coordonnée $h(\boldsymbol{x})$ à la fois, cette coordonnée devant avoir une corrélation non négligeable (meilleure que le hasard) avec l'étiquette u.

Si des liens ont ainsi pu être établis de manière prometteuse entre le boosting et les méthodes à large marge, il reste encore à les explorer de manière plus complète et il y a là encore de beaux sujets de recherche en perspective (voir par exemple [FS99] pour des références).

4.5 Généralisation à une classe, à plus de deux classes et à la régression

Quand toutes les données d'apprentissage ont la même classe, l'apprentissage se ramène à la classification non supervisée (voir le chapitre18), qui peut s'aborder aussi comme un problème d'estimation de densité de probabilité à partir d'observations. Une approche consiste à estimer par séparation les *quantiles* de la distribution multidimensionnelle cherchée. Dans ce cas, on cherche dans une certaine famille une surface séparatrice H_μ telle que la probabilité d'appartenir à la classe unique soit supérieure à μ d'un côté de la surface et inférieure de l'autre côté. Pour AdaBoost, on se place du point de vue de l'espace de dimension T et on y cherche un hyperplan, comme indiqué au paragraphe 4.4.3.

Copyright © 2010 Eyrolles.

Quand on pose formellement le problème, on peut le ramener à un problème d'optimisation sous contraintes assez semblable à celui décrit au même paragraphe.

Il est également possible d'étendre AdaBoost à l'apprentissage d'une règle de classification pour un nombre quelconque C de classes. Plusieurs solutions ont été exprimées. L'une des plus récentes propose de remplacer le calcul

$$\alpha_t \leftarrow \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$$

opéré dans l'algorithme 33 par l'affectation suivante :

$$\alpha_t \leftarrow \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t} + Log(C - 1)$$

L'introduction de ce terme n'est pas qu'un artefact, mais a effectivement une signification en terme d'optimisation [ZRZH06]. D'autres propositions ont été faites, par exemple dans [ASS00, SS98a].

L'algorithme AdaBoost peut également être utilisé pour la régression : à partir d'une méthode de régression de base, on construit itérativement une combinaison pondérée d'occurences de cette méthode en modifiant l'échantillon d'apprentissage. Une bibliographie est donnée dans [MR03].

4.6 L'utilisation du boosting

Le boosting, et particulièrement l'algorithme AdaBoost, a été employé avec succès avec de nombreux algorithmes d'apprentissage « faibles ». On l'utilise souvent par exemple avec des arbres de décision à profondeur volontairement limitée (dans le cas extrême, les *decision stumps*, ou « souches de décision » ne font qu'un seul test pour choisir une classe). On utilise aussi par exemple C4.5, un système d'apprentissage d'arbre de décision [Qui93] ou Ripper, un système d'apprentissage de règles). Le boosting a été testé sur des domaines d'application variés. En général, l'utilisation du boosting a pour résultat d'améliorer souvent sensiblement les performances en apprentissage.

Les avantages du boosting et de AdaBoost en particulier sont qu'il s'agit d'une méthode facile à programmer et aisée d'emploi. Elle ne nécessite pas de connaissance *a priori* sur l'algorithme d'apprentissage « faible » utilisé, et elle peut s'appliquer de fait à n'importe quel algorithme d'apprentissage faible. Les seuls paramètres à régler sont la taille de l'ensemble d'apprentissage m et le nombre total d'étapes T, qui peuvent être fixés par l'utilisation d'un ensemble de validation (voir le chapitre 3). De plus, des garanties théoriques sur l'erreur en généralisation permettent de contrôler l'apprentissage. Une autre propriété intéressante du boosting est qu'il tend à détecter les exemples aberrants (*outliers*) puisqu'il leur donne un poids exponentiellement grand en cours d'apprentissage. Cependant, la contrepartie de ce phénomène est que le boosting est sensible au bruit et ses performances peuvent être grandement affectées lorsque de nombreux exemples sont bruités. Récemment des algorithmes ont été proposés pour traiter ce problème (comme *Gentle AdaBoost* [HTF01] ou BrownBoost [Fre99]).

Nous avons vu au passage que l'adaptation aux problèmes multiclasses n'est pas immédiate, mais qu'elle a cependant fait l'objet d'études menant aussi à des algorithmes efficaces. De même, il existe des applications de boosting à la régression.

4.7 Boosting et théorie *PAC*

Les premiers travaux sur le boosting sont historiquement issus du cadre de l'apprentissage *PAC* (*probably approximately correct*, voir les chapitres 2 et 21). Dans ce cadre, un algorithme *PAC* au sens *fort* (*strong PAC*) est défini ainsi :

Copyright © 2010 Eyrolles.

- pour toute distribution de probabilité $\mathcal{D}_{\mathcal{X}} \times \mathcal{D}_{\mathcal{U}}$ sur l'espace des exemples $(\mathbf{x}, \boldsymbol{u})$,
- $\forall \varepsilon > 0, \delta > 0$,
- étant donné un nombre polynomial (fonction de $1/\varepsilon$ et de $1/\delta$) d'exemples i.i.d. suivant $\mathcal{D}_{\mathcal{X}} \times \mathcal{D}_{\mathcal{U}}$,
- l'algorithme trouve une hypothèse d'erreur $\leq \varepsilon$ avec une probabilité $\geq 1 - \delta$.

Les algorithmes d'apprentissage dits *faibles* ont une définition analogue, mais on leur demande seulement de trouver une hypothèse d'erreur $\varepsilon \geq \frac{1}{2} - \gamma$, avec γ strictement positif, donc éventuellement juste un peu meilleure que le hasard, en supposant une tâche de classification binaire avec la même proportion d'exemples positifs et négatifs.

La question qui est à l'origine du boosting est la suivante : « est-il est possible d'utiliser un algorithme faible pour obtenir un apprentissage de type fort? »

Comme on l'a vu, Shapire [FS99]a prouvé que la réponse à cette question est positive et a conçu le premier algorithme de boosting par sous-ensembles. Freund [Fre99] a ensuite produit un algorithme beaucoup plus efficace, également optimal, mais difficile à appliquer. En 1995, Freund et Shapire [FS97] ont proposé l'algorithme AdaBoost, efficace et pratique, qui est désormais la technique la plus employée pour améliorer les performances de n'importe quel algorithme d'apprentissage supervisé.

Dans le même temps, d'autres chercheurs ont analysé comment il est possible d'identifier les bons experts au sein d'une grande collection d'experts ou bien les bons attributs quand on a un grand nombre d'attributs (ces deux problèmes sont reliés). Les algorithmes développés, tels que *Winnow* [LW94] (voir au chapitre 18) ont révélé l'intérêt dans ces problèmes de la mise à jour multiplicative des pondérations d'experts, comme le réalise le boosting pour la classification.

5. Le *bagging*

Le *bagging* (de *bootstrap aggregation*) est une méthode qui, comme le boosting, combine des hypothèses pour obtenir une hypothèse finale. Cependant la méthode est plus simple et généralement moins performante. L'idée de base est d'entraîner un algorithme d'apprentissage élémentaire sur plusieurs bases d'apprentissage obtenues par tirage avec remise[7] de m' (avec $m' < m$) exemples d'apprentissage dans l'échantillon d'apprentissage $\mathcal{S}$. Pour chaque tirage b (pour *bag*), une hypothèse h_b est obtenue. L'hypothèse finale est simplement la moyenne des hypothèses obtenues sur B tirages au total :

$$H(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} h_b(\mathbf{x})$$

L'une des justifications de cette méthode est que si les hypothèses h_b calculées pour chaque tirage b ont une variance importante (donc sont sensibles à l'échantillon des m' exemples d'apprentissage), alors leur moyenne H aura une variance réduite.

[7] Méthode de tirage que l'on appelle *bootstrap*, voir le chapitre 3.

Copyright © 2010 Eyrolles.

6. Les forêts aléatoires (*random forests*)

Le *bagging* semble performant quand il est appliqué à des fonctions d'estimation de faible biais mais de forte variance. C'est le cas en particulier des arbres de décision ou de régression. La *méthode des forêts aléatoires* (*random forests*) modifie l'algorithme du *bagging* appliqué aux arbres en ajoutant un critère de *dé-corrélation* entre les arbres. Les performances de cette méthode sont souvent très bonnes, et le fait que l'algorithme soit facile à mettre en œuvre le rend populaire.

La justification de cette approche est la suivante. Lorsque que l'on calcule la moyenne de B variables aléatoires, chacune de variance σ^2, la variance globale est de : $\frac{1}{B}\sigma^2$. Si les variables sont seulement identiquement distribuées mais non nécessairement indépendantes, et de corrélation par paires ρ, alors la variance globale est : $\rho\,\sigma^2 + \frac{1-\rho}{B}\sigma^2$, dans laquelle le second terme tend vers 0 quand B augmente.

L'idée de la méthode des forêts aléatoires est ainsi de réduire la corrélation ρ sans augmenter trop la variance. La technique utilisée consiste à sélectionner aléatoirement un sous-ensemble de n variables à considérer à chaque étape de choix du meilleur nœud de l'arbre (voir chapitre 13). Typiquement, n a comme valeur $\sqrt{d}$ quand d est la taille de la dimension de l'espace d'entrée, c'est-à-dire le nombre de descripteurs.

7. L'apprentissage en cascade (*cascading*)

Dans l'apprentissage en cascade, les classificateurs élémentaires ne sont pas placés en parallèle comme dans le vote ou le boosting, mais en série. Autrement dit, l'organisation est la suivante : le premier classificateur élémentaire est lancé sur la donnée. Si le résultat produit est considéré comme fiable, il est conservé. Sinon, c'est le second classificateur élémentaire qui traite la donnée. Si son résultat est fiable, on le conserve, sinon on passe la main au classificateur élémentaire suivant, et ainsi de suite.

Par exemple, pour $C = 2$, le premier classificateur est, disons un hyperplan. Si la marge qu'il produit en résultat est trop faible, on passe la main à un classificateur bayésien paramétrique. Si la différence des probabilités de prédiction entre les deux classes n'est pas suffisante, on lance un classificateur par k plus proches voisins.

Cette technique a un avantage évident : elle optimise le temps de décision, en particulier si on ordonne les classificateurs élémentaires par complexité de calcul croissante. Un classificateur élémentaire complexe ne sera alors lancé que sur des objets particulièrement difficiles à classer. D'une manière générale, les problèmes simples seront résolus rapidement et les problèmes difficiles seront examinés avec plus de soin.

En revanche, elle nécessite une évaluation de la confiance de chaque décision, ce qui est un problème analogue à celui de la *classification avec rejet*, que nous n'avons pas encore abordé dans ce livre. Les exemples ci-dessus sont réalistes : la marge ou l'évaluation d'une probabilité de classement sont des indicateurs utilisables. On peut aussi évaluer le seuil de fiabilité d'un classificateur par un ensemble de validation. Cette seconde technique peut d'ailleurs être intégrée dynamiquement au processus de cascade pour éviter de fixer des seuils indépendants du problème particulier que l'on est en train de traiter.

Dans cette optique, l'apprentissage en cascade peut alors se décrire par le schéma suivant. On commence par diviser l'ensemble d'apprentissage $\mathcal{S}$ en deux parties $\mathcal{A}_1$ et $\mathcal{V}_1$. On apprend le

Copyright © 2010 Eyrolles.

premier classificateur sur $\mathcal{A}_1$. On classe les éléments de $\mathcal{V}_1$. Cette classification permet de trouver les éléments de $\mathcal{V}_1$ qui sont non fiables : ils forment un nouvel ensemble $\mathcal{S}_2$, que l'on divise en deux parties $\mathcal{A}_2$ et $\mathcal{V}_2$. On apprend le second classificateur sur $\mathcal{A}_2$. On classe les éléments de $\mathcal{V}_2$. Cette classification permet de trouver les éléments de $\mathcal{V}_2$ qui sont non fiables, et ainsi de suite.

Notes historiques et sources bibliographiques

L'histoire du boosting est évoquée au paragraphe 4.7. De remarquables développements théoriques ont été effectués sur ces méthodes en particulier sur la capacité de généralisation et les liens avec les *SVM* (chapitre 9). Le mot anglais *arcing* (de *adaptive reweighting and combining*) est employé pour désigner toutes les méthodes qui sélectionnent ou repondèrent les données pour améliorer la classification. Les deux méthodes de boosting que nous avons vues en font partie. Le *bagging* en fait aussi partie (voir [HTF01] et le chapitre 3).

Résumé

- On a d'abord vu dans ce chapitre comment combiner plusieurs classificateurs différents pour qu'ils s'entraident à résoudre un problème de classification supervisée. L'apprentissage des paramètres de ces combinaisons est appelé méta-apprentissage.
- On a également exposé les codes correcteurs de classes, qui utilisent une matrice de codage des décisions. Ceci permet en particulier d'appliquer des méthodes d'apprentissage de concept (deux classes) à un nombre quelconque de classes.
- On a abordé ensuite le *boosting*, qui est une technique d'apprentissage qui vise à rendre plus performant un système d'apprentissage « faible ». Pour cela, le système d'apprentissage est entraîné successivement sur des échantillons d'apprentissage surpondérant les exemples difficiles à apprendre. À chaque fois, une hypothèse h_t est produite, et l'hypothèse finale est une combinaison linéaire de ces hypothèses pondérées par des coefficients liés à leur performance.
- Le boosting est d'un emploi très large et fait l'objet de nombreux travaux et applications.

Copyright © 2010 Eyrolles.

Chapitre 18

La classification non supervisée et la fouille de données

Ce chapitre aborde un nouvel aspect de l'apprentissage : le cas où les données ne sont pas supervisées par une classe. Dans cette optique, la connaissance que l'on peut en extraire est de plusieurs types. On peut chercher comment les données se regroupent entre elles, et l'on apprend de la sorte quelle est leur division « naturelle » en classes. C'est ce qu'on appelle la classification non supervisée. *On peut aussi chercher des corrélations entre les valeurs des attributs, et on est dans le domaine de la « fouille de données », une discipline que la multiplication des fichiers géants de données fait actuellement prospérer.*

L'apprentissage non supervisée concerne aussi la recherche de variables latentes, que ce soit, au sens large, les axes d'inertie sous-jacents aux données ou bien des composantes indépendantes ou encore des composantes non négatives.

Copyright © 2010 Eyrolles.

Sommaire

1 Introduction 596
1.1 Le problème général 596
1.2 Les difficultés 596
2 Les méthodes de classification fondées sur les distances 597
2.1 Distances entre objets 597
2.2 Finesse d'une partition, qualité d'une partition 598
2.3 L'algorithme des k-moyennes 599
2.4 Les k-moyennes « en ligne » 600
2.5 Les cartes auto-organisatrices de Kohonen 601
2.5.1 Présentation. Un exemple 601
2.5.2 La méthode générale 602
2.6 La classification hiérarchique de données numériques 603
2.6.1 Généralités 603
2.6.2 Un algorithme général de classification hiérarchique 606
2.6.3 L'indice du lien simple 606
2.6.4 L'indice de la distance entre centres de gravité 607
2.6.5 L'indice de Ward 607
2.7 L'évaluation de la partition obtenue 607
2.8 Les grandes dimensions : effets étranges et propriétés utiles 608
3 Les méthodes de classification par des modèles probabilistes 609
3.1 L'estimation d'une somme pondérée de distributions gaussiennes 609
4 Méthodes spectrales de catégorisation 610
5 La classification de données symboliques 612
5.1 Les données binaires et catégorielles 612
5.2 Les attributs nominaux : la représentation attribut-valeur 613
5.3 Les données logiques 615
6 La fouille de données 615
6.1 La recherche de motifs fréquents 618
6.1.1 Vocabulaire, définitions et propriétés 618
6.1.2 Algorithmes de recherche de motifs fréquents 620
6.1.3 Généralisations 622
6.2 Les règles d'association 622
6.2.1 Définition et concepts essentiels 623
6.2.2 Calcul des règles d'association à partir des motifs fréquents . . 623
6.2.3 L'évaluation des règles d'association 624
6.3 Autres problèmes en fouille de données 625
7 Les analyses en composantes 625
7.1 Analyse en composantes principales (ACP) 626
7.2 Analyse en composantes indépendantes (ACI) 626
7.2.1 Principe des méthodes d'ACI 627
7.2.2 Mesures d'indépendance statistique 627
7.2.3 Algorithmes et applications 628
7.2.4 Lien avec la technique de « projection pursuit » 629
7.3 Analyse en composantes non négatives (NMF) 630
7.4 Analyses en variables latentes 632

Copyright © 2010 Eyrolles.

7.4.1 Analyse sémantique latente (LSA) 632
7.4.2 Modèles graphiques 632
7.5 Propriétés des bases de primitives de représentation 632

LES ANIMAUX se divisent en : a) appartenant à l'empereur, b) embaumés, c) apprivoisés, d) cochons de lait, e) sirènes, f) fabuleux, g) chiens en liberté, h) inclus dans la présente classification, i) qui s'agitent comme des fous, j) innombrables, k) dessinés avec un pinceau très fin en poil de chameau, l) *et cætera*, m) qui viennent de casser la cruche, n) qui de loin semblent des mouches.

Cette classification que M. Foucault qualifie de « déconcertante »[1] est due à l'imagination de J.-L. Borges[2]. Son étrangeté provient d'abord de ce qu'il ne s'agit pas vraiment d'une « division » (en terme formel : d'une partition) des espèces : un animal peut être à la fois un cochon de lait, apprivoisé et une possession de l'empereur. Surtout, peut-être, la variété des concepts utilisés pour regrouper les espèces est absurde : « s'agiter » ne s'oppose ni ne se compare en rien à « être dessiné », par exemple. Et que dire de « inclus dans la présente classification » ou de « *et cætera* » ? Par contraste, une classification raisonnable des animaux devrait utiliser une description comparable de chaque animal (vertébré ou non, et si vertébré, mammifère ou non, etc.) et *in fine* fournir une partition opératoire (par exemple, deux espèces sont différentes si elles ne sont pas interfécondes, pour des raisons génétiques ou géographiques).

Nous avons parlé juqu'ici dans ce livre de classification supervisée, pour laquelle les éléments de l'ensemble d'apprentissage sont pourvus d'une étiquette ou d'une valeur numérique fournies par un oracle. Dans ce chapitre, la problématique de la classification est différente : étant donné un certain nombre d'objets décrits par des attributs, est-il possible d'identifier les familles dans lesquelles se regroupent ces objets ? C'est pour ainsi dire le problème auquel l'oracle a été confronté avant d'être capable de réaliser un étiquetage. Ce problème s'appelle la classification non supervisée ou le *clustering*[3]. Il y a deux manières de l'aborder : soit en construisant une hiérarchie de classes ; soit directement en cherchant un nombre fixé de classes.

Ces deux méthodes sont assez aisées à pratiquer pour des attributs numériques, quand la notion de distance est naturelle. Dans le cas d'attributs binaires ou nominaux, un certain nombre d'extensions sont cependant possibles.

Pour finir, ce chapitre abordera une autre problématique de l'apprentissage non supervisé : celle de la recherche de corrélations entre attributs et de l'extraction automatique de connaissances de bases de données non supervisées, ou *fouille de données*.

Nous traitons ici l'apprentissage non supervisé et de la fouille de données de manière rapide. Le lecteur désirant approfondir ses connaissances dans ce domaine peut se reporter aux références proposées dans la section 7.5.

Copyright © 2010 Eyrolles.

[1] « On sait ce qu'il y a de déconcertant dans la proximité des extrêmes ou tout bonnement le voisinage soudain des choses sans rapport ; l'énumération qui les entrechoque possède à elle seule un pouvoir d'enchantement. » *Les mots et les choses*. Gallimard (1966).

[2] *La langue analytique de John Wilkins*, dans *Enquêtes*, Gallimard (1957).

[3] On dit parfois « la coalescence » ou simplement « la classification ».

1. Introduction

1.1 Le problème général

On considère dans ce chapitre des données non supervisées et dans la première partie, on cherche à trouver une partition de ces données en classes « naturelles ». Ce problème général est souvent appelé *classification automatique*. Une partition des données en classes signifie simplement que chaque donnée se voit attribuer une classe et une seule, autrement dit que les données sont *partitionnées* en classes.

Définition 18.1 (Partition)

Soit un ensemble fini S. Une partition π *de S est un ensemble de parties de S, non vides et disjointes deux à deux, dont l'union est S. Si s désigne un élément de S, il existe donc un unique élément,* ou bloc, *de π comprenant s.*

Dans ce chapitre, compte tenu du contexte, nous emploierons souvent le mot « classe » au lieu de « bloc ». La classification automatique est fondée sur l'hypothèse qu'il existe une *distance* entre les objets, ou qu'une distribution de probabilités inconnue les a engendrés. Dans le premier cas, les classes vont regrouper les objets proches et seront aussi éloignées entre elles que possible. Dans le second, on fera des hypothèses sur la famille à laquelle appartient la distribution ; le cas le plus général est celui où les données sont supposées avoir été engendrées par un mélange de lois normales, ou *mélange de gaussiennes*.

Ce chapitre aborde aussi très brièvement le problème de l'apprentissage multi-vues et du co-apprentissage. Il traite ensuite plus largement de la fouille de données et des analyses en composantes. Dit de manière très générale, ces derniers sujets s'intéssent non pas au regroupement des objets, mais plutôt à celui des attributs qui les décrivent. La fouille de données cherche les attributs communs aux objets et les « motifs » (ensembles d'attributs) fréquents dans la description des données par ces attributs. L'analyse en composantes cherche les combinaisons linéaires des attributs qui « expliquent » le mieux les données. Cette dernière problématique a été déjà abordée au chapitre 3, paragraphe 1.3, qui traitait de la réduction de la dimension de l'espace d'entrée. C'est en effet une de ses applications possibles, dans le cadre de l'apprentissage supervisé cette fois.

1.2 Les difficultés

Les difficultés de la classification automatique sont de deux ordres : d'abord, on ne connaît pas, bien sûr, le nombre de classes (le nombre de blocs de la partition). Il est facile de faire converger un algorithme vers une des deux classifications triviales : celle avec un seul bloc (tous les objets sont regroupés dans la même classe) et celle avec autant de classes que de données. Il est plus difficile de lui faire choisir le nombre de classes intrinsèque aux données, à supposer qu'il existe. De plus, il faut supposer que la représentation en attributs est correcte et bien normalisée. La plupart des méthodes donneront un résultat différent si l'un des attributs (ici, numériques) est, disons, divisé par dix, alors que la nature des données n'est évidemment pas changée par une telle opération. Cela provient du fait que ces méthodes sont basées sur l'utilisation d'une distance, qui donne la même importance à toutes les coordonnées, ici les attributs. Changer l'échelle d'un des attributs modifie la distance entre objets. Diverses techniques de normalisation, hors du propos de cet ouvrage, peuvent être envisagées pour traiter ce problème.

Dans le cas des méthodes de classification fondées sur des hypothèses probabilistes, souvent rien

Copyright © 2010 Eyrolles.

n'assure *a priori* que ces hypothèses soient valables. Une hypothèse fréquente est que les données de chaque classe viennent d'une distribution de probabilités, que toutes les distributions sont du même type (en général gaussien), mais que leurs paramètres sont différents. Une donnée est alors affectée à la classe la plus probable. L'hypothèse gaussienne est loin d'être une évidence, contrairement à ce qu'une mauvaise interprétation de la loi des grands nombres peut laisser supposer. Par exemple, tracer un histogramme, sur mille ou dix mille personnes, de l'âge où ils ont passé leur permis de conduire ne donnera jamais une courbe de type gaussien. Il existe des tests sur le caractère gaussien, qui permettent de juger de la qualité de cette hypothèse, et des transformations vers une répartition gaussienne, qui sont eux aussi en dehors du propos de ce livre.

Quant à la fouille de données, les difficultés viennent littéralement de ce qu'on ne sait pas ce que l'on cherche. On est donc obligé d'éviter une explosion combinatoire par des heuristiques de recherche, et ce d'autant plus qu'on s'intéresse en général à des volumes énormes de données.

2. Les méthodes de classification fondées sur les distances

2.1 Distances entre objets

Définition 18.2 (Distance)

Une distance Δ *sur un espace* E *est une application de* $E \times E$ *dans* $\mathbb{R}^+$ *qui vérifie les propriétés suivantes :*

1. $\Delta(x,y) = 0 \iff x = y$;
2. $\forall\ x,y \in \Sigma, \Delta(x,y) = \Delta(y,x)$ *(symétrie)* ;
3. $\forall x,y,z \in \Sigma\ \ \Delta(x,y) \leq \Delta(x,z) + \Delta(z,y)$ *(inégalité triangulaire).*

Une relation qui vérifie les deux premières propriétés mais pas l'inégalité triangulaire est souvent appelée une *mesure de dissimilarité*. Quand on travaille sur des objets qui sont des vecteurs réels, c'est-à-dire quand $E = \mathbb{R}^d$, la *p-distance* entre deux objets $\mathbf{x} = (x_1, \ldots, x_d)^\top$ et $\mathbf{y} = (y_1, \ldots, y_d)^\top$ s'écrit :

$$\Delta_p(\mathbf{x},\mathbf{y}) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$$

En général, on prend $p = 2$ pour calculer la *distance euclidienne* entre deux objets :

$$\Delta(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

Quand les objets considérés sont des vecteurs binaires (quand $E = \mathbb{B}^d$), on peut utiliser la distance de Hamming, qui est égale au nombre d'éléments binaires différents entre les deux objets (éventuellement divisé par d). On verra au paragraphe 5 d'autres distances entre objets symboliques.

Dans $\mathbb{R}^d$, on appelle *centre de gravité* ou *moyenne* de m objets $\mathbf{x}_1 = (x_{11}, \ldots, x_{1d})^\top, \ldots,$

Copyright © 2010 Eyrolles.

$\mathbf{x}_m = (x_{m1}, \ldots, x_{md})^\top$ l'objet $\boldsymbol{\mu}$ de $\mathbb{R}^d$ défini par :

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{m}^{m} \mathbf{x}_i = \left(\frac{1}{m} \sum_{i=1}^{m} x_{i1}, \ldots, \frac{1}{m} \sum_{i=1}^{m} x_{id} \right)^\top$$

Rappelons que ce nouvel objet $\boldsymbol{g}$ possède la propriété suivante :

$$\boldsymbol{\mu} = \underset{\boldsymbol{y} \in \mathbb{R}^d}{\text{ArgMin}} \sum_{i=1}^{m} \Delta(\mathbf{y}, \mathbf{x}_i)$$

Autrement dit, le centre de gravité $\boldsymbol{\mu}$ d'un ensemble de m points est le point de $\mathbb{R}^d$ qui minimise la somme des distances euclidiennes aux m points considérés.

2.2 Finesse d'une partition, qualité d'une partition

Soit un E ensemble de m objets. Nous allons d'abord définir une relation d'ordre partiel entre les partitions de E.

Définition 18.3

Une partition π_i est plus fine *qu'une partition π_j si et seulement si tout bloc de π_j est un bloc de π_i ou est l'union de plusieurs blocs de π_i.*

Selon la relation d'ordre précédente, la partition la plus fine de $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ est constituée de m blocs et s'écrit :

$$\mathcal{P}_m(\mathcal{S}) = (\mathbf{x}_1), \ldots, (\mathbf{x}_m)$$

alors que la partition la moins fine n'a qu'un bloc et s'écrit :

$$\mathcal{P}_1(\mathcal{S}) = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$$

Par exemple, pour $\mathcal{S} = \{a, b, c, d, e, f\}$, les deux partitions

$$(a, b, c), (d), (e, f) \text{ et } (a, b), (c, d), (e, f)$$

sont toutes deux plus fines que la partition

$$(a, b, c, d), (e, f)$$

mais n'ont pas de relation de finesse entre elles.

Nous nous plaçons maintenant dans $\mathbb{R}^d$, avec m objets partitionnés en C blocs et nous utilisons la distance euclidienne Δ. Notons $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_C$ les centres de gravité des C blocs et δ_i^j le symbole de Kronecker, qui vaut 1 quand $i = j$ et 0 sinon. On définit T, la *somme des variances intra classes* (ou plus simplement la *variance*) de la partition considérée, par :

$$T = \frac{1}{m} \sum_{j=1}^{C} \sum_{i=1}^{m} \delta_i^j \, (\mathbf{x}_i - \boldsymbol{\mu}_j)^2$$

Le produit mT est donc égal à la somme sur toutes les classes de la distance de tous les points de cette classe au centre de gravité de celle-ci.

Pour deux partitions différentes, mais de même nombre de classes, T est un bon indice de la qualité de la classification effectuée, du point de vue du regroupement en terme de distance euclidienne. Par exemple, dans la figure 18.1, la partition dessinée par des traits pleins est meilleure que celle dessinée par des pointillés : les deux blocs sont plus regroupés et mieux séparés.

Les notions de variance et de finesse sont reliées dans $\mathbb{R}^d$ par la propriété suivante :

Copyright © 2010 Eyrolles.

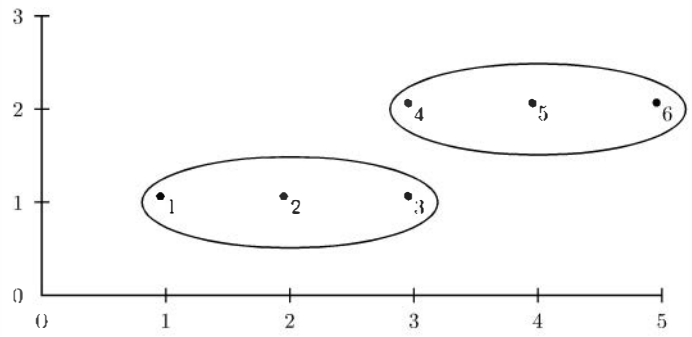


FIG. 18.1: *Deux partitions à deux classes comme résultats possibles de la classification non supervisée de quatre points.*

Propriété 18.1

Soit une partition π de variance $T(\pi)$ plus fine qu'une partition σ de variance $T(\sigma)$. Alors :

$$T(\pi) \leq T(\sigma)$$

2.3 L'algorithme des k-moyennes

L'algorithme dit *des k-moyennes* (ou *k-means*) cherche à trouver la partition à k blocs qui minimise la variance. C'est une technique empirique, qui n'assure pas la découverte de la partition optimale.

Donnons un exemple emprunté à Webb [Web99] pour présenter cette méthode. Soient six objets numérotés de 1 à 6 dans $\mathbb{R}^2$, comme sur la figure 18.2. On cherche à les regrouper en deux classes, autrement dit à en faire une partition à deux blocs. Pour cela, on commence par en tirer deux au hasard, disons les points 5 et 6.

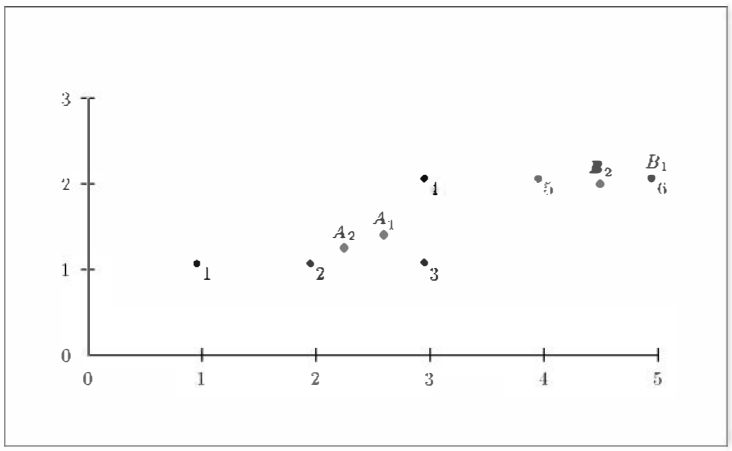


FIG. 18.2: *Un ensemble de points à partitionner à deux classes et le résultat de la méthode des 2-moyennes en partant des points 5 et 6.*

Copyright © 2010 Eyrolles.

La première étape est constituée de deux phases. La première phase va allouer les six points aux deux classes sur le critère suivant : ceux qui sont plus près (au sens de la distance euclidienne) du point 5 que du point 6 sont dans la première classe, les autres dans la seconde. La première classe est donc pour le moment le bloc $(1, 2, 3, 4, 5)$ et la seconde le bloc (6). La seconde phase consiste à calculer le centre de gravité de ces deux classes. Le premier, appelé A_1 a pour coordonnées $(1.6, 2.4)$, le second, B_1 est le point 6, aux coordonnées $(5, 2)$.

On recommence les deux phases dans la seconde étape : les six points sont alloués aux deux classes en fonction de A_1 et de B_1. On obtient les blocs $(1, 2, 3, 4)$ et $(5, 6)$. Les centres de gravité de ces deux blocs sont A_2 et B_2, aux coordonnées $(2.25, 1.25)$ et $(4.5, 2)$.

Une nouvelle étape ne change plus la partition : l'algorithme a convergé.

Il est intéressant de calculer une qualité globale de la classification en utilisant T, la somme des variances intra classes. Les valeurs successives de ce critère sur les partitions obtenues sont les suivantes :

(1,2,3,4,5), (6)	6.4
(1,2,3,4), (5,6)	4.0

On peut montrer que l'algorithme des k-moyennes fait en effet diminuer la valeur T, mais rien n'assure que le mimimum global soit atteint : la convergence peut en effet mener à un minimum local.

Si on initialise l'algorithme avec les points 2 et 5, la convergence vers la partition $(1, 2, 3), (4, 5, 6)$ est réalisée d'entrée pour la valeur $T = 3.3$ du critère. La première initialisation, avec les points 5 et 6, converge donc vers un minimum local.

L'algorithme des k-moyennes est un cas particulier des algorithmes de classification par *allocation-recentrage*. La phase d'allocation est ici le calcul des classes à partir des centres de gravité provisoires, et la phase de recentrage le calcul des nouveaux centres de gravité des classes que l'on vient d'établir. Cette technique peut se voir de manière encore plus générale comme une application particulière de l'algorithme EM, qui sera abordé au paragraphe 3.1.

Il existe une grande variété d'algorithmes du type k-moyennes, permettant en particulier de faire naître et mourir des classes au cours des calculs, ce qui donne de la souplesse à la méthode : son inconvénient évident est en effet que l'utilisateur doit donner au départ le nombre de classes, ce qui n'est pas très réaliste.

Il est aussi possible de choisir le nombre de classes en faisant plusieurs essais avec des valeurs différentes et en calculant pour chacune un critère fondé sur un compromis entre la variance et le nombre de classes, que nous verrons pour les méthodes hiérarchiques au paragraphe 2.7.

2.4 Les k-moyennes « en ligne »

L'algorithme des k-moyennes produit donc après convergence k classes et les k centres de gravité correspondants. Appelons ici les centres de gravité simplement des « centres », pour donner une version différente de l'algorithme des k-moyennes dans laquelle chaque vecteur est présenté tour à tour une seule fois.

Au départ, k centres sont choisis au hasard. Quand un nouveau point se présente, le centre le plus proche de ce point est déplacé, et c'est le seul centre qui est modifié. Ce déplacement se fait de manière à ce que la variance de la classification diminue.

Cette technique peut être formalisée comme une descente de gradient pour minimiser la variance. Nous ne la présentons pas en détail, mais nous la mentionnons comme une version primitive de la technique du paragraphe suivant, dans laquelle plusieurs centres sont modifiés à l'arrivée d'un nouveau point.

Copyright © 2010 Eyrolles.

2.5 Les cartes auto-organisatrices de Kohonen

2.5.1 Présentation. Un exemple

Dans cette méthode, les « centres » appartiennent toujours à l'espace de représentation, mais il sont souvent en grand nombre et surtout, ils sont contraints selon une certaine topologie. Cela veut dire que pour chaque centre il existe un certain nombre d'autres centres désignés comme devant appartenir à son voisinage. En général, cette topologie est définie en dimension faible : un, deux ou trois. Nous prenons pour illustrer dans la suite la dimension deux et nous supposons que nous utilisons cent centres.

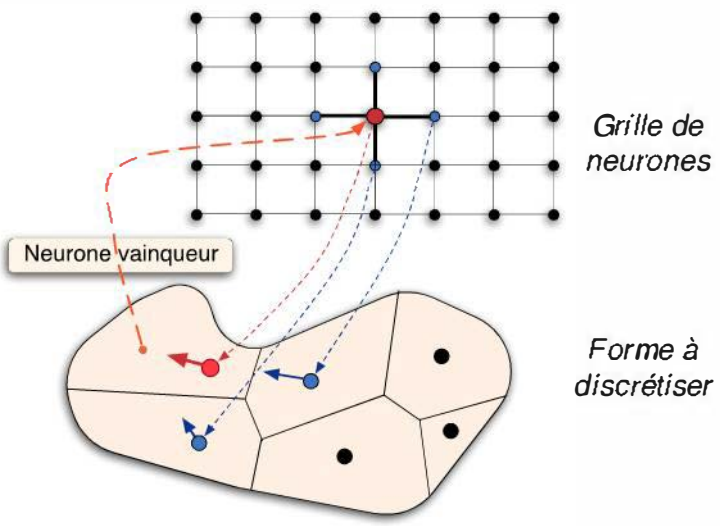


FIG. 18.3: *Méthode des cartes auto-organisatrices. La topologie du réseau de neurones est ici définie par une grille où chaque neurone a 4 voisins au premier degré. La forme à discrètiser produit des stimuli. Pour chacun d'eux le vecteur de référence le plus proche (le vainqueur) est sélectionné et sa position est modifiée ainsi que celle de ses voisins.*

En pratique, définir une topologie en dimension deux revient à donner à chaque centre un nom μ_{ij} différent, avec $1 \leq i, j \leq 10$, et à définir leur voisinage à partir des indices qui apparaissent dans leurs noms : ainsi, deux centres μ_{ij} et μ_{kl} tels que $Max(|i-k|, |j-l|) = 1$ sont considérés comme des voisins directs. Notons $VD(\mu_{ij})$ l'ensemble des voisins directs de μ_{ij}. Alors : $VD(\mu_{34}) = \{\mu_{23}, \mu_{33}, \mu_{43}, \mu_{45}, \mu_{35}, \mu_{25}, \mu_{24}, \mu_{44}\}$.

L'algorithme se déroule alors de la manière suivante. On tire d'abord au hasard la position des cent centres dans l'espace de représentation. À cette étape, naturellement, la topologie des centres n'est pas respectée : les voisins (du point de vue de leurs indices) n'ont pas de raison d'être physiquement proches.

Ensuite, on examine tous les points de l'ensemble d'apprentissage plusieurs fois, souvent un grand nombre de fois. Quand un point $\mathbf{x}$ est traité, on commence par identifier le centre qui lui est le plus proche en distance euclidienne. Disons que c'est μ_{34}. On déplace μ_{34} dans la direction de $\mathbf{x}$ d'une certaine quantité et on déplace aussi tous les éléments de $VD(\mu_{34})$ dans la direction de $\mathbf{x}$, d'une quantité plus petite voir figure 18.3).

Au début de l'algorithme, il se peut que, même si μ_{31} est très proche de $\mathbf{x}$, en revanche μ_{44} en soit très éloigné. Mais cela n'a pas d'importance. Après suffisamment d'étapes, la topologie

Copyright © 2010 Eyrolles.

arbitrairement imposée aux centres va se retrouver dans l'espace de représentation par convergence, ce que l'auteur de cette méthode, T. Kohonen, a appelé *l'auto-organisation*. À la fin, les centres μ_{34} et μ_{11} seront à faible distance euclidienne.

Après convergence, on trouve donc les centres dans l'espace de représentation $\mathbb{R}^d$ organisés selon des contraintes qui ont été imposées en dimension deux. Il est alors intéressant de les visualiser dans un espace de dimension deux, par exemple après une analyse en composantes principales (voir le paragraphe 7.1). La densité locale des centres reflète celle des données de départ. En fin de compte, on peut considérer cette méthode davantage comme une technique de visualisation des données que comme une méthode de classification proprement dite (voir figure 18.4).

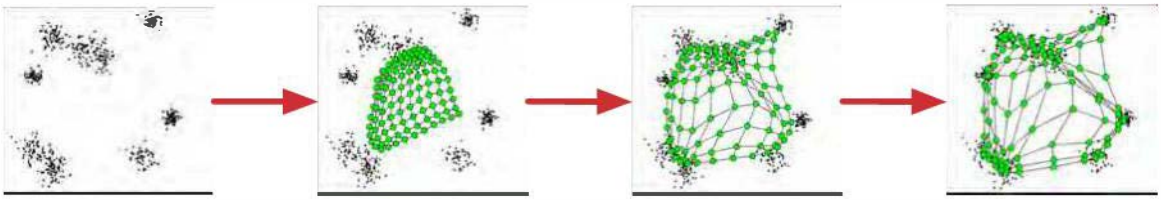


FIG. 18.4: *Évolution d'une carte auto-organisatrice à deux dimensions sur des données définies également en deux dimensions. À la fin, les centres de la carte reflètent la densité des points (merci à Younes Bennani pour cette figure).*

Algorithme 34 : La construction d'une carte auto-organisatrice

début
  Prendre au hasard les centres μ_i, pour $i = 1, C$ et choisir η positif quelconque
  $t \leftarrow 0$
  tant que $t \leq T$ **faire**
    Tirer au hasard une donnée d'apprentissage $\mathbf{x}$
    Déterminer l'indice i du centre le plus proche de $\mathbf{x}$
    pour *chaque centre* μ_j *avec* $j = 1, C$ **faire**
      $\delta_j \leftarrow \eta\, e(\mu_j, \mu_i)\, \Delta(\boldsymbol{x}, \mu_i)$
      $\mu_j \leftarrow \mu_j + \delta_j$
    fin pour
  fin tant que
  $t \leftarrow t + 1$
fin

2.5.2 La méthode générale

D'une manière générale, une topologie sur les centres est définie comme une fonction capable de calculer leur proximité dans l'espace de dimension réduite. Prenons C centres $\mu_1, \ldots, \mu_C$ et donnons-nous une fonction $e(\mu_i, \mu_j)$ qui prend une valeur de plus en plus petite au fur et à mesure que les centres μ_i et μ_j s'éloignent dans la topologie que l'on veut leur imposer. Par exemple, $e(\mu_i, \mu_j)$ peut être une exponentielle décroissante de la distance entre les centres dans l'espace de dimension réduite. Au lieu de définir pour chaque centre qui est un voisin et qui ne l'est pas, on peut alors considérer que tous les centres sont voisins et utiliser la fonction e pour

Copyright © 2010 Eyrolles.

leur mise à jour. La formule de réactualisation de la valeur des centres devient alors, si le centre le plus proche de x est μ_i :

$$\mu_j \leftarrow \mu_j + \delta(\mu_j) = \mu_j + \eta\, e(\mu_j, \mu_i)\, \Delta(\mathbf{x}, \mu_i)$$

La distance euclidienne Δ est prise dans l'espace de représentation, alors que la fonction e décrit la topologie imposée aux centres. Le paramètre η règle le compromis entre la vitesse et la qualité de la convergence. La valeur T est choisie à partir de la quantité T/m (m est la taille de l'ensemble d'apprentissage), qui indique le nombre de fois en moyenne que chaque exemple est utilisé. Cette quantité T/m peut être de l'ordre de la centaine, car la convergence est souvent lente.

Il faut noter la méthode de Kohonen est sujette à aboutir à des minima locaux. Par ailleurs, comme la plupart des méthodes non supervisées, elle est difficile à valider car il n'y a pas de critère objectif associé à la partition obtenue.

2.6 La classification hiérarchique de données numériques

2.6.1 Généralités

Soit un ensemble $\mathcal{S} = \{\mathbf{x}_1 \ldots \mathbf{x}_m\}$ de m objets. Nous allons définir formellement une hiérarchie sur $\mathcal{S}$, de deux manières différentes, mais finalement équivalentes. Nous utilisons la relation d'ordre de finesse entre partitions en commençant par une définition préliminaire.

Définition 18.4

Une chaîne *dans l'ensemble des partitions de $\mathcal{S}$ est un ensemble de partitions $\{\pi_1, \ldots, \pi_r\}$ tel que pour $i = 1, r-1$ on a : π_i est plus fine que π_{i+1}.*

Nous pouvons maintenant définir une hiérarchie de partitions de deux manières (leur équivalence est facile à montrer).

Définition 18.5 (Hiérarchie, 1)

Une hiérarchie *sur $\mathcal{S}$ est une chaîne de partitions de $\mathcal{S}$ dont la plus fine est $\mathcal{P}_m$ et la moins fine est $\mathcal{P}_1$.*

L'autre définition est la suivante :

Définition 18.6 (Hiérarchie, 2)

Une hiérarchie *H sur $\mathcal{S}$ est un sous-ensemble des parties de $\mathcal{S}$ tel que :*

- *pout tout élément $\mathbf{x}$ de $\mathcal{S}$, $\{\mathbf{x}\} \in H$;*
- *pour tout couple d'éléments h et h' de H avec $h \neq h'$, on a :*
  - *soit $h \cap h' = \emptyset$,*
  - *soit $h \cap h' \neq \emptyset$, alors soit $h \subset h'$, soit $h' \subset h$.*

Par exemple, la hiérarchie représentée à la figure 18.5 peut être vue soit comme la chaîne de partitions

(a, b, c, d, e, f)
$(a, b, c, d), (e, f)$
$(a, b, c), (d), (e, f)$
$(a, b, c), (d), (e), (f)$
$(a), (b), (c), (d), (e), (f)$

Copyright © 2010 Eyrolles.

soit comme l'ensemble de parties de $\mathcal{S}$: $H = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\}$, avec

$$\begin{array}{ll} h_1 = \{a\} & h_7 = \mathcal{S} = \{a, b, c, d, e, f\} \\ h_2 = \{b\} & h_8 = \{a, b, c, d\} \\ h_3 = \{c\} & h_9 = \{e, f\} \\ h_4 = \{d\} & h_{10} = \{a, b, c\} \\ h_5 = \{e\} & \\ h_6 = \{f\} & \end{array}$$

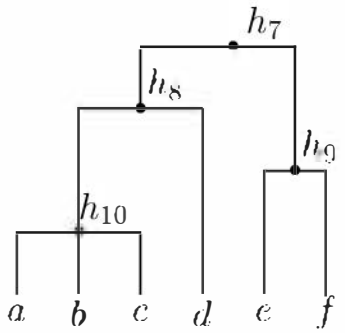


FIG. 18.5: *Une hiérarchie sur* $\mathcal{S} = \{a, b, c, d, e, f\}$.

Définition 18.7 (Hiérarchie indicée monotone)

Une hiérarchie indicée *est une hiérarchie H sur un ensemble fini à laquelle on associe une suite de nombres réels r_i. Une hiérarchie indicée est* monotone *si pour deux éléments h_i et h_{i+1} consécutifs dans H, avec h_i plus fine que h_{i+1}, on a $r_i \leq r_{i+1}$.*

Par exemple, sur la figure 18.6, la hiérarchie de la figure 18.5 a été indicée de manière monotone sur l'axe vertical selon l'association :

$$\begin{array}{rl} (a, b, c, d, e, f) & 4.2 \\ (a, b, c, d) & 3 \\ (e, f) & 1.9 \\ (a, b, c) & 1.1 \\ (a), (b), (c), (d), (e), (f) & 0 \end{array}$$

Construire une hiérarchie sur un ensemble d'exemples est donc équivalent à trouver une chaîne de partitions sur cet ensemble. Si l'on construit une hiérarchie indicée monotone sur cet ensemble, on peut obtenir les partitions de la chaîne en « coupant » la hiérarchie pour une valeur de l'indice. Par exemple, sur la figure 18.6, la coupure pour la valeur 2.5 de l'indice fournit la partition $(a, b, c), (d), (e, f)$.

On peut associer à toute hiérarchie indicée H une mesure de dissimilarité entre ses éléments de la façon suivante : la dissimilarité $\delta(h_i, h_j)$ entre les parties h_i et h_j de H prend la valeur de l'indice de la plus petite partie h de H qui contient h_i et h_j.

On peut démontrer que cette mesure de dissimilarité possède une propriété forte : elle est une *distance ultramétrique*.

Copyright © 2010 Eyrolles.

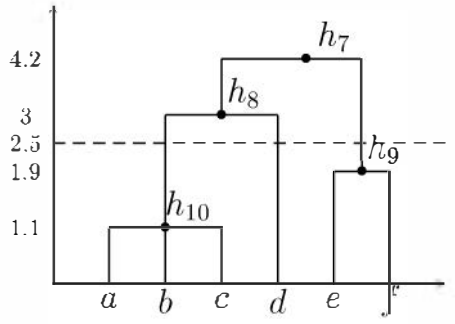


FIG. 18.6: *Une hiérarchie indicée sur* $\mathcal{S} = \{a, b, c, d, e, f\}$. *La coupure pour la valeur* 2.5 *de l'indice produit la partition* $(a, b, c), (d), (e, f)$.

Définition 18.8 (Ultramétrique)

Une ultramétrique *(abrégé de distance ultramétrique) sur un ensemble* Z *est une application de* $Z \times Z \rightarrow \mathbb{R}$ *qui vérifie les trois conditions suivantes pour tous les* z_i, z_j *et* z_k *de* Z :

- $\delta(z_i, z_i) = 0$;
- $\delta(z_i, z_j) = \delta(z_j, z_i)$;
- $\delta(z_i, z_k) \leq Max(\delta(z_i, z_j), \delta(z_j, z_k))$.

Par exemple, la hiérarchie indicée de la figure 18.6 place les éléments $\{a\}$ et $\{d\}$ à la distance ultramétrique $\delta(\{a\}, \{d\}) = 3$, puisque la plus petite partie de H qui contient $\{a\}$ et $\{d\}$ est h_8. De même, $\delta(\{d\}, \{e, f\}) = 4.2$ puisque la plus petite partie de H qui contient $\{d\}$ et $\{e, f\}$ est h_7. On peut vérifier la propriété ultramétrique :

$$\delta(\{a\}, \{e, f\}) \leq Max(\delta(\{a\}, \{d\}), \delta(\{d\}, \{e, f\})) = Max(3, 4.2) = 4.2$$

En résumé, nous sommes désormais en possession des notions suivantes :

- Nous avons un ensemble d'exemples $\mathcal{S}$ composé de m objets.
- Cet ensemble est muni d'une distance Δ, en général la distance euclidienne.
- Nous cherchons à construire une hiérarchie indicée H de partitions de $\mathcal{S}$.
- Nous savons que construire une telle hiérarchie indicée est équivalent à donner une ultramétrique δ sur les parties de $\mathcal{S}$ présentes dans H.
- Si la hiérarchie indicée construite est monotone, la hiérarchie de partitions trouvée est complètement satisfaisante, puisque l'agrégation de deux sous-ensembles d'une partition produit une partition d'indice supérieur.

Il nous faut donc utiliser la distance Δ sur $\mathcal{S}$ pour en déduire une hiérarchie indicée, si possible monotone. Pour cela, il faut définir une autre mesure D de dissimilarité, celle-là entre tous les couples de parties de $\mathcal{S}$. La première étape est facile : si l'on considère les objets de $\mathcal{S}$ comme des parties de $\mathcal{S}$ comportant un seul élément, on posera naturellement :

$$D(\{\mathrm{x}\}, \{\mathrm{y}\}) = \Delta(\mathrm{x}, \mathrm{y})$$

Il nous faut ensuite trouver comment définir la valeur de D sur n'importe quel couple de sous-ensembles de $\mathcal{S}$. Nous présentons dans les paragraphes suivants quelques indices D qui permettent

Copyright © 2010 Eyrolles.

de construire une hiérarchie sur $\mathcal{S}$. Auparavant, donnons un algorithme constructif qui utilise D et dont le résultat est une hiérarchie indicée, mais pas forcément monotone.

2.6.2 Un algorithme général de classification hiérarchique

Quand on dispose d'une mesure de similarité D entre les parties de $\mathcal{S}$, la construction d'une hiérarchie indicée se fait de manière simple par l'algorithme 35.

Chaque étape de cet algorithme produit un élément de la chaîne de partitions. Le nombre maximal d'étapes est m, le nombre d'objets.

Il nous reste maintenant à présenter quelques mesures de dissimilarité ou *indices* D classiques qui produisent par cet algorithme des hiérarchies indicées.

Algorithme 35 : Algorithme de classification hiérarchique

Établir la table T_D des valeurs de $D(\mathbf{x}, \mathbf{y})$ pour $\mathbf{x}$ et $\mathbf{y}$ parcourant $\mathcal{S}$.

répéter

1. Choisir les deux sous-ensembles h_i, h_j de $\mathcal{S}$ tels que $D(h_i, h_j)$ est le plus petit nombre réel dans la table T_D
2. Supprimer h_j de la table, remplacer h_i par $h_i \cup h_j$
3. Calculer les mesures de similarité D entre $h_i \cup h_j$ et les autres éléments de la table.

jusqu'à *ce que la table T_D n'ait plus qu'une colonne*

2.6.3 L'indice du lien simple

Cet indice produit l'ultramétrique dite *sous-dominante*. Il mesure la distance euclidienne entre les deux points les plus proches, l'un dans un bloc de partition, l'autre dans le second. Quand les blocs sont réduits à un élément, cet indice mesure la distance euclidienne entre ces deux éléments. La hiérarchie indicée trouvée est monotone.

Reprenons un ensemble à six exemples et donnons la table des distances euclidiennes entre les exemples. Cette dernière est aussi la table T_D, qui est identique sur les couples d'objets. Comme cette table est symétrique, seule une moitié est représentée.

	a	b	c	d	e	f
a	0	**1.1**	1.1	3	6	5
b		0	1.1	4	5.5	4.2
c			0	3	6.5	5.3
d				0	9	8
e					0	1.9
f						0

L'algorithme de classification hiérarchique produit successivement les tables ci-dessous (la plus petite valeur non nulle de chaque table est en gras). Notons qu'à la première étape, ce ne sont pas trois points qui sont regroupés, bien que a, b et c soient deux à deux à la même distance, mais deux, comme le veut l'algorithme. Dans un cas pareil, on tire au hasard deux points sur les trois.

D	$h_{11} = \{a, b\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$
h_{11}	0	**1.1**	3	5.5	4.2
$\{c\}$		0	3	6.5	5.3
$\{d\}$			0	9	8
$\{e\}$				0	1.9
$\{f\}$					0

Copyright © 2010 Eyrolles.

D	$h_{10} = \{a, b, c\}$	$\{d\}$	$\{e\}$	$\{f\}$
h_{10}	0	3	5.5	4.2
$\{d\}$		0	9	8
$\{e\}$			0	**1.9**
$\{f\}$				0

D	h_{10}	$\{d\}$	$h_9 = \{e, f\}$
h_4	0	**3**	4.2
$\{d\}$		0	8
h_3			0

D	$h_8 = h_{10} \cup \{d\}$	h_9
h_2	0	**4.2**
h_3		0

D	$h_7 = h_{10} \cup h_9$
h_1	0

On retrouve donc la hiérarchie indicée des figures 18.5 et 18.6.

Cette indice est très simple et semble naturel. Il peut donner des résultats surprenants en raison d'un certain « effet de chaîne », qui regroupe parfois des points de manière non naturelle. On utilise de préférence pour cette raison des indices un peu plus sophistiqués, comme ceux qui sont proposés aux paragraphes suivants.

2.6.4 L'indice de la distance entre centres de gravité

Prendre pour indice D la distance euclidienne entre les centres de gravité des blocs fournit une hiérarchie non forcément monotone, ce qui n'est en général pas souhaitable.

Il est facile de la remarquer sur l'exemple suivant : soient les points $\{a, b, c\}$ de $\mathbb{R}^2$ de coordonnées $a = (0,0)$, $b = (9,0)$ et $c = (4.5,8.5)$. La distance la plus petite est entre a et b et vaut 9 (les deux autres valent environ 9.6). Le centre de gravité du bloc (a, b), de coordonnées (4.5,0) est à une distance de c qui vaut 8.5, strictement inférieur à 9.

Par conséquent, il se produit ce qu'on appelle une *inversion* dans la construction de la hiérarchie : le résultat n'est pas une hiérarchie indicée monotone.

2.6.5 L'indice de Ward

Pour tenir compte de la variance des classes et pour éviter l'effet de chaîne dans la classification, Ward [Sim85] a proposé d'utiliser l'indice suivant, donné ici sous la forme de « formules de réactualisation ». Il fournit une hiérarchie indicée monotone.

Nous sommes à l'étape courante de l'algorithme 35, nous avons choisi les deux blocs les plus proches h_i et h_j et nous cherchons à calculer l'indice entre le nouveau bloc $h_i \cup h_j$ et un autre bloc h_k. Le nombre d'éléments du bloc h_i (respectivement : h_j, h_k) vaut m_i (respectivement : m_j, m_k).

$$D(h_i \cup h_j, h_k) = \frac{n_k + n_i}{n_k + n_i + n_j} D(h_i, h_k) + \frac{n_k + n_j}{n_k + n_i + n_j} D(h_j, h_k) - \frac{n_i + n_j}{n_k + n_i + n_j} D(h_i, h_j)$$

2.7 L'évaluation de la partition obtenue

La classification hiérarchique fournit potentiellement autant de classifications qu'il y a de données. Donner une valeur de l'index correspond exactement à donner une partition, donc une classification. Mais le choix de cette valeur n'est pas facile. On peut utiliser deux types de méthodes pour effectuer ce choix. Premièrement, étudier la progression des valeurs de l'index correspondant aux changement du nombre de classes. Dans notre exemple, ces valeurs sont : 1.1, 1.9, 3.0 et 4.2. Si on observe un intervalle nettement plus élevé que les autres, on peut couper la

Copyright © 2010 Eyrolles.

hiérarchie par une valeur située dans cet intervalle. Une seconde technique consiste à mesurer la somme T' des variances intra-classes et à calculer une valeur du type

$$\frac{1}{T'}\frac{m-C}{C-1}$$

qui combine les deux termes T' et C variant en sens inverse. T' est nul quand le nombre de classes C est égal au nombre m d'objets et est maximum quand C vaut 1. Ce critère passe donc par un extremum pour un nombre de classes intermédiaire qui représente un bon compromis. D'autre critères plus justifiés du point de vue statistique ont été proposés. On peut se reporter au livre [DHS01] pour plus de précisions.

2.8 Les grandes dimensions : effets étranges et propriétés utiles

L'analyse de données en grande dimension, i.e. au-delà de 30, pose de grands défis. En effet :

- De nombreux estimateurs statistiques requièrent un nombre de points exponentiel en la dimension de l'espace d'entrée. Ainsi, certains modèles, comme les histogrammes, présentent des taux d'erreur en $\mathcal{O}(n^{-1/d})$, ce qui signifie que pour diviser l'erreur par deux, il faut 2^d fois plus de données!!
- Une autre difficulté est d'ordre calculatoire. Le cas du problème de 2-moyenne en est un exemple typique. Il s'agit de répartir les m points des données en deux groupes, et donc en deux centres, de telle manière à minimiser la moyenne des distances des points à leur centre le plus proche. Des méthodes naïves sont de complexité calculatoire en $\mathcal{O}(m^d)$, ce qui est astronomique, même pour des petites valeurs de d. De ce fait, des méthodes heuristiques sont employées, qui sont malheureusement souvent fondées sur une intuition de ce qui se passe en 2D ou en 3D, alors que les espaces de grande dimension présentent de nombreuses propriétés contre-intuitives.

On parle de *malédiction ou de fléau de la dimensionnalité* (*curse of dimensionality*) pour souligner l'aspect supposé maléfique apporté par un espace de grande dimension. Pourtant des travaux récents commencent à clarifier ces effets et même à proposer d'en tirer parti pour simplifier l'analyse des données. Nous citerons ici deux propriétés étranges liées aux espaces de grande dimension.

1. En grande dimension, le volume d'un solide, disons une hypersphère $\mathcal{B}_d$, tend à se concentrer dans sa « peau ». Plus précisément, si l'on tire au hasard un point dans une sphère de petite dimension d et de rayon 1, il existe une grande chance qu'il soit à une distance inférieure à $r < 1 - \varepsilon$ du centre de son centre. En revanche, en grande dimension, cette probabilité tend vers 0. La raison en est que la sphère de rayon r de même centre ne contient qu'une fraction r^d du volume de la sphère unité, et cette fraction tend rapidement vers 0 quand d augmente. Pour des valeurs de d assez grandes, la plus grande partie du volume se trouve dans une « peau » $\{x \in \mathbb{R}^d : 1 - \varepsilon \leq ||x|| \leq 1\}$ pour $\varepsilon = \mathcal{O}(1/d)$.
2. En espace de dimension d, par définition, il existe au plus d vecteurs orthogonaux. Mais supposons que l'on soit moins exigeant et que l'on cherche des vecteurs *presque* orthogonaux, à $90^\circ \pm \varepsilon$, alors on peut en trouver un nombre exponentiel. En effet, un ensemble de $\exp\big(\mathcal{O}(\varepsilon^2 d)\big)$ vecteurs pris aléatoirement entre le centre et un point de la boule $\mathcal{B}_d$ satisfera presque sûrement cette propriété.

On comprend que ces propriétés puissent paraître mettre en péril de nombreuses méthodes de recherche de régularités entre les données puisque la plupart sont fondées ultimement sur des comparaisons de distances. Si la distance entre deux points pris au hasard est la même que pour

Copyright © 2010 Eyrolles.

n'importe quelle autre paire de points, ce que tend à impliquer la propriété sur la concentration du volume dans la peau d'une boule, alors aucune régularité valide ne pourra être découverte.

Cependant, ces propriétés concernent des points très particuliers, ceux qui sont tirés aléatoirement dans l'espace. Les données réelles peuvent être sensiblement différentes. Le péril serait donc circonscrit. Mais, ne peut-on pas imaginer de tirer parti, au contraire, des propriétés mentionnées plus haut pour simplifier l'analyse de données en grande dimension ? L'idée de méthodes récentes consiste à *ajouter* du bruit pour utiliser ces propriétés.

Ainsi, un résultat très important est le *théorème de Johnson-Lindenstrauss*. En substance, il affirme que si on prend n'importe quel ensemble de m points dans un espace de grande dimension, et qu'on les projette sur un sous-espace de dimension $\mathcal{O}(\log m)$ choisi arbitrairement, alors la distance entre ces m points sera très probablement presque parfaitement préservée. Il s'agit là d'un résultat remarquable qui permet par exemple de réduire à $\mathcal{O}(\log m)$ la complexité calculatoire de tâches de clustering ou d'autres types d'analyse qui dépendent du calcul de distances entre m points.

3. Les méthodes de classification par des modèles probabilistes

3.1 L'estimation d'une somme pondérée de distributions gaussiennes

On peut faire une analogie raisonnée entre la méthode des k-moyennes et la technique non paramétrique du plus proche voisin (chapitre 15, paragraphe 3.3). De même, il est possible de faire des hypothèses paramétriques sur la distribution des objets. Souvent, on considère qu'ils sont des tirages i.i.d. d'une distribution multigaussienne, encore appelée un mélange de gaussiennes (*mixture of normal distributions model*), c'est-à-dire une somme pondérée de gaussiennes de moyennes et de matrices de covariance inconnues. En pratique, cela revient à supposer que chaque classe est une gaussienne avec ses caractéristiques particulières et une probabilité *a priori* plus ou moins forte[4].

Nous avons donné au chapitre 15 la formule d'une distribution gaussienne dans $\mathbb{R}^d$. Nous la rappelons ici en notant $p(\mathcal{N}(\boldsymbol{\mu}, Q))$ la densité de cette distribution, avec $\boldsymbol{\mu}$ sa moyenne et Q sa matrice de covariance.

$$p(\mathcal{N}(\boldsymbol{\mu}, Q)) = \frac{|Q|^{-1/2}}{(2\pi)^{d/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top Q^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{18.1}$$

Un mélange de C gaussiennes s'écrit donc :

$$p(\mathcal{N}(\boldsymbol{\mu}, Q)) = \sum_{j=1,C} k_i \frac{|Q_i|^{-1/2}}{(2\pi)^{d/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^\top Q_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right\} \tag{18.2}$$

avec

$$\sum_{j=1,C} k_i = 1$$

En supposant C connu, il faut estimer les C scalaires k_i, les C matrices symétriques Q_i de dimension $d \times d$ et les C vecteurs $\boldsymbol{\mu}_i$ de dimension d pour tout i entre 1 et C, soit $\frac{d^2+2d+2}{2}$ paramètres.

[4] Sous ce point de vue l'algorithme des k-moyennes revient à faire l'hypothèse que les classes proviennent de distributions gaussiennes ayant toutes la même matrice de covariance ; les probabilités *a priori* sont quelconques.

Copyright © 2010 Eyrolles.

On utilise pour cela l'algorithme *EM*, que nous avons déjà rencontré au chapitre 12 pour l'apprentissage des HMM. Une explication de la méthode *EM* et de son application à l'estimation des paramètres des mélanges de gaussiennes est donnée en annexe 7. Cette méthode souffre aussi de ce que C doit être fixé à l'avance. Là encore, diverses techniques sont applicables pour trouver la meilleure valeur, mais on peut s'appuyer ici sur l'hypothèse paramétrique émise sur la distribution des objets. Une technique éprouvée est de maximiser le « critère bayésien d'information » (*BIC*), comme présenté dans [JW98].

— EXEMPLE

La figure 18.7 illustre les performances de cette méthode par comparaison avec celle des k-moyennes. On y voit 600 points, 287 tirés selon une première loi normale de moyenne $\mu_1^T = (1 \quad 1)$ et de matrice de covariance $Q_1 = \begin{pmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{pmatrix}$ et 313 selon une seconde de moyenne $\mu_1^T = (0.8 \quad 0.8)$ et de matrice de covariance $\begin{pmatrix} 0.02 & 0 \\ 0 & 0.02^2 \end{pmatrix}$. Les deux distributions sont supposées équiprobables. Les surfaces d'équidensité de ces distributions sont les cercles en trait plein ; la probabilité qu'un point soit tiré à l'intérieur est de 90 %. L'erreur bayésienne (le mieux que l'on puisse faire, voir le chapitre 15) vaut ici 13 %.

Les programmes qui implantent l'algorithme *EM* et celui des k-moyennes, connaissent les 400 points et le fait qu'il y ait deux classes à trouver, mais ils ignorent la classe de chaque point et la probabilité *a priori* des classes.

Sur la figure de gauche est donné le résultat de *EM*. Il trouve deux distributions gaussiennes assez proches des originales. Une probabilité *a priori* de 0.57 est attribuée à la première classe et de 0.43 à la seconde. Les distributions sont représentées par les ellipses en pointillés, qui correspondent aux valeurs suivantes :

$$\hat{\mu}_1^T = (1.043 \quad 1.018) \quad \hat{Q}_1 = \begin{pmatrix} 0.097 & -0.005 \\ -0.005 & -0.093 \end{pmatrix}$$

$$\hat{\mu}_2^T = (0.792 \quad 0.808) \quad \hat{Q}_2 = \begin{pmatrix} 0.025 & 0.002 \\ 0.002 & 0.019 \end{pmatrix}$$

La matrice de confusion de la classification des données initiales est la suivante ; elle donne une erreur de 27 % :

	classe 1	classe 2
classe 1	199	88
classe 2	19	294

Sur la figure de droite, l'algorithme des 2-moyennes sépare les points en deux classes. La surface séparatrice est la droite médiatrice des centres de gravité des classes trouvées (en pointillés). La matrice de confusion de la classification des données initiales est la suivante (ce qui donne une erreur de 34 %) :

	classe 1	classe 2
classe 1	162	125
classe 2	10	308

4. Méthodes spectrales de catégorisation

Comme les sections précédentes l'ont montré, l'une des approches les plus utilisées pour la catégorisation est fondée sur une approche générative, c'est-à-dire dans laquelle on cherche à identifier

Copyright © 2010 Eyrolles.

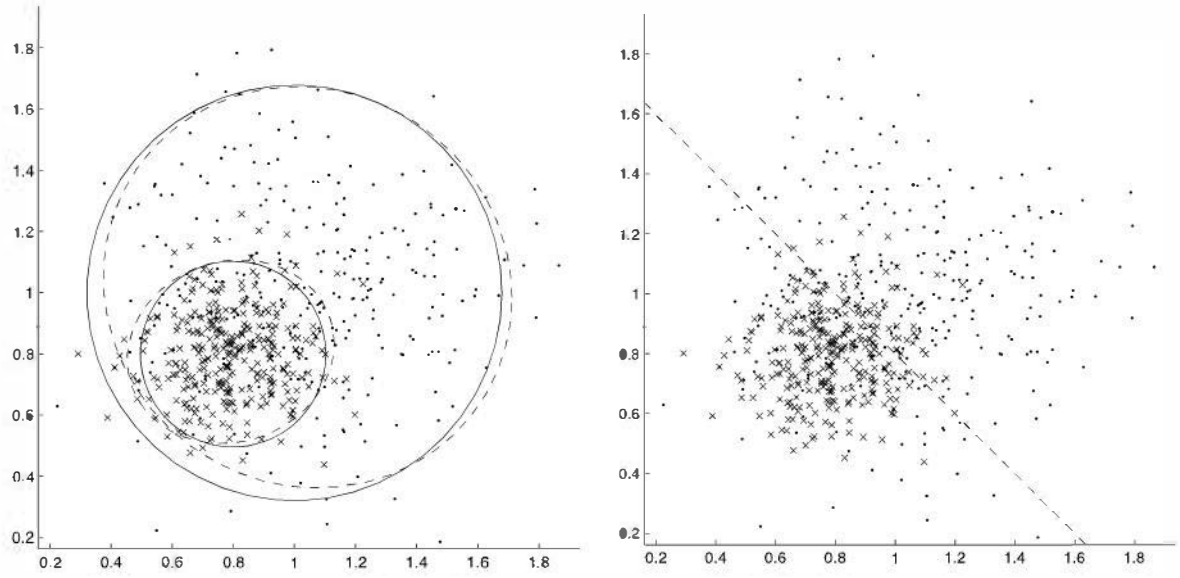


FIG. 18.7: *L'algorithme EM retrouve un mélange de distributions gaussiennes, mais l'algorithme des 2-moyennes est plus en difficulté.*

la distribution sous-jacente des données $\mathbf{P}_{\mathcal{X}}$. Une méthode standard est d'utiliser un algorithme tel que EM (*Expectation-Maximisation*), (voir l'annexe 7) qui recherche un mélange de distributions. Ce type d'approches souffre de plusieurs inconvénients. D'abord, l'estimation de densités paramétriques requiert la plupart du temps de faire des hypothèses (trop) simplificatrices, par exemple, que la distribution de probabilité est gaussienne. Ensuite, la log-vraisemblance peut présenter de nombreux minima locaux, ce qui implique d'utiliser une stratégie de recherche à partir de plusieurs initialisations. Les algorithmes tels que la k-moyenne connaissent les mêmes problèmes. Une approche différente a été proposée qui utilise des méthodes spectrales pour la catégorisation. L'idée générale est d'employer les premiers vecteurs propres d'une matrice dérivée des distances entre les points représentant les données.

Les méthodes spectrales de catégorisation résolvent le problème classique : étant donné un ensemble de points $\{\mathbf{x}_i\}_{i=1..m}$, avec, souvent, $\mathbf{x}_i \in \mathbb{R}^d$, et une mesure de similarité entre toutes les paires de points $(\mathbf{x}_i, \mathbf{x}_j)$, partitionner les points en groupes (*clusters*) tels que les points à l'intérieur d'un groupe soient similaires, tandis que les points de groupes différents soient très peu similaires.

Les algorithmes de catégorisation spectrale prennent comme point de départ un graphe de similarité pondéré $G(V, E)$, où les sommets correspondent aux points et les poids aux similarités entre paires de points. Elles calculent alors le Laplacien du graphe (voir le chapitre 14) et trouvent une décomposition en valeurs propres de ce Laplacien. Certains algorithmes utilisent directement les vecteurs propres trouvés pour calculer une catégorisation *k-way*. La plupart des algorithmes forment une partition binaire des données à partir du deuxième vecteur propre (associé à la deuxième plus grande valeur propre). En cas de nécessité, la décomposition se poursuit de manière récursive sur les sous-groupes identifiés.

L'algorithme 36 est un exemple de catégorisation spectrale binaire dans lequel la mesure de similarité $\exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_k^2})$ est utilisée.

Copyright © 2010 Eyrolles.

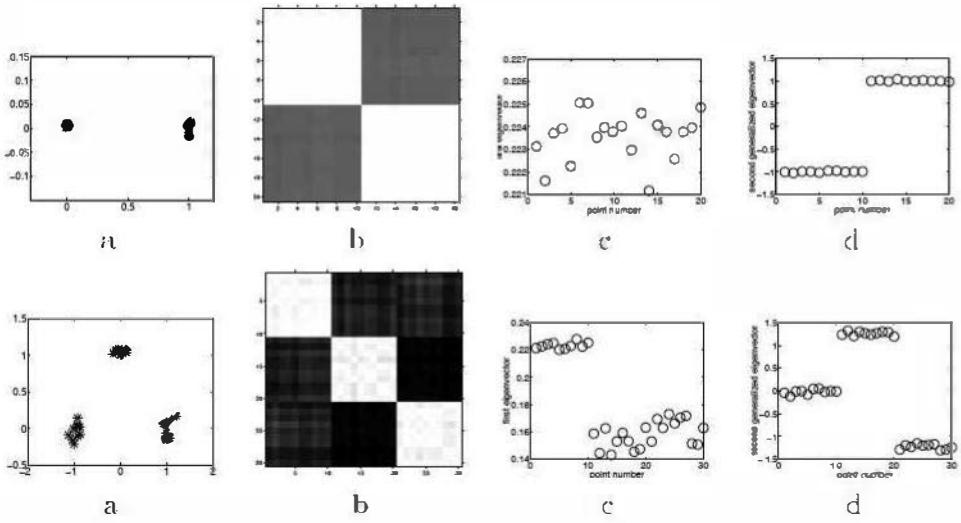


FIG. 18.8: (a) Un problème simple de catégorisation. (b) la matrice d'affinité. (c) le premier vecteur propre. (d) le second vecteur propre (d'après [Wei99], p.976).

Algorithme 36 : Algorithme de catégorisation spectrale bipartie

Données : Échantillon S de m points $\{\mathbf{x}_i\}_{i=1..m}$ avec $\mathbf{x}_i \in \mathbb{R}^d$
Résultat : Une bi partition S' et S'' des données.
début

1. Calcul de la matrice de similarité K : $K_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_k^2}\right), \quad \forall \mathbf{x}_i, \mathbf{x}_j$
2. Calcul de la matrice D : $D_i = \sum_{j=1}^{m} K_{ij}$
3. Calcul de la matrice laplacienne normalisée : $L = I - D^{-1}K$
4. Trouver le deuxième vecteur propre $\mathbf{v}_2$ de L
5. Calcul des deux parties de S en utilisant $\mathbf{v}_2$:
   $S' = \{[i] : v_{2i} > 0\}, \quad S'' = \{[i] : v_{2i} \leq 0\}$

fin

5. La classification de données symboliques

5.1 Les données binaires et catégorielles

Quand on quitte l'espace de représentation $\mathbb{R}^d$, de nouveaux problèmes apparaissent pour la classification automatique. Prenons d'abord le cas des données binaires, dans lequel chaque exemple objet est un vecteur de $\mathbb{B}^d$ composé de d bits. Quelles mesures de distance peut-on proposer dans une telle représentation ?

La distance la plus simple est celle de Hamming qui mesure le nombre de bits différents entre deux objets, divisé par d. Sa valeur est donc toujours comprise entre 0 et 1. Mais on peut proposer d'autres mesures qui ne vérifient pas forcément les propriétés de la distance[5]. Soient deux objets

[5] Rappellons la définition d'une distance, donnée en 3.2 : une distance Δ sur un espace $E \times E$ est une application de $E \times E$ dans $\mathbb{R}^+$ si et seulement si elle vérifie les propriétés :
- $\Delta(x, y) = 0 \iff x = y$;

Copyright © 2010 Eyrolles.

x et y de $\mathbb{B}^d$. Notons :

- a le nombre d'attributs qui sont $VRAI$ pour x et y ;
- b le nombre de ceux qui sont $VRAI$ pour x et $FAUX$ pour y ;
- c le nombre de ceux qui sont $FAUX$ pour x et $VRAI$ pour y ;
- d le nombre d'attributs qui sont $FAUX$ pour x et y.

Les mesures de dissimilarité suivantes entre x et y sont classiques :

Nom	Formule	Commentaire
Hamming	$\frac{c+d}{a+b+c+d}$	est une distance
Russel et Rao	$1 - \frac{a}{a+b+c+d}$	ne vérifie pas la première propriété de la distance
Jaccard	$1 - \frac{a}{a+b+c}$	est une distance
Yule	$1 - \frac{ad-bc}{ad+bc}$	

Il est difficile d'appliquer l'algorithme non hiérarchique des k-moyennes à des données binaires, car il fait appel à la notion de centre de gravité *a priori* intraduisible dans $\mathbb{B}^d$. Certains auteurs ont proposé des techniques empiriques pour contourner la situation. Nous en verrons un exemple au paragraphe suivant (dans le cas de la représentation attribut-valeur) qui peut s'appliquer en particulier à celui de la logique des propositions.

En revanche, les algorithmes hiérarchiques peuvent toujours s'appliquer, à condition de savoir calculer un indice ultramétrique entre deux ensembles d'objets binaires. Les indices du lien simple et de Ward sont en particulier calculables, car ils ne font pas appel au centre de gravité des ensembles. Nous verrons aussi au chapitre 21 que l'on ne peut pas mesurer la ressemblance entre deux objets binaires avec le nombre de propositions logiques qui ne valent par exemple $VRAI$ sur les deux que si on limite volontairement l'ensemble des propositions disponibles par un biais (par exemple, si on décide de n'y mettre que les conjonctions).

5.2 Les attributs nominaux : la représentation attribut-valeur

Dans le cas où les attributs sont catégoriels (nominaux), un grand nombre de méthodes hiérarchiques ont été proposées. Elles sont fondées sur le calcul d'une distance entre objets catégoriels, puis sur son extension à un indice ultramétrique. Prenons l'exemple suivant. Un oiseau est défini par trois attributs :

- le fait que son bec soit *aplati* ou non ;
- sa *taille* qui peut prendre trois valeurs : *petite*, *moyenne* ou *grande* ;
- la *couleur* de son cou qui peut prendre quatre valeurs *roux*, *orange*, *gris* ou *noir*.

Soient deux oiseaux définis par :

- $\forall\ x, y \in \Sigma, \Delta(x,y) = \Delta(y,x)$ (symétrie) ;
- $\forall x, y, z \in \Sigma\ \ \Delta(x,y) \leq \Delta(x,z) + \Delta(z,y)$ (inégalité triangulaire).

Copyright © 2010 Eyrolles.

	aplati	taille	couleur	nom
x_1	$VRAI$	moyenne	roux	nette rousse
x_2	$FAUX$	moyenne	noir	corneille noire

On peut par exemple généraliser la distance de Hamming par le calcul suivant[6] :

$$\Delta(x_1, x_2) = 1 - (1 + 0 + 3) = 1 - \frac{1}{3}(2\frac{1}{2} + 3\frac{0}{3} + 4\frac{1}{4}) = \frac{1}{3}$$

Cette formule peut aussi être remplacée par la suivante, qui considère que $aplati = VRAI$ et $aplati = FAUX$ sont en quelque sorte deux fois moins différents que $couleur = roux$ et $couleur = noir$, puisqu'il y a deux modalités pour la première variable et quatre pour la seconde :

$$\Delta(x_1, x_2) = 1 - \frac{1}{3}(\frac{1}{2} + \frac{0}{3} + \frac{1}{4}) = \frac{1}{3} = 0.25$$

Cette dernière expression varie entre 0 et $\frac{1}{3}(\frac{1}{2} + \frac{1}{3} + \frac{1}{4}) = 0.36$, mais il est facile de la ramener entre 0 et 1 si nécessaire.

Il est intéressant de noter qu'un certain nombre de méthodes fondées sur les concepts de l'espace des versions (voir le chapitre 4) ont été proposées pour construire des classifications non hiérarchiques. On a remarqué plus haut que la méthode des k-moyennes n'est pas applicable, car la notion de centre de gravité n'existe plus. Mais il est possible de construire des algorithmes analogues comme $CLUSTER/2$ [MS83] dont l'algorithme est schématiquement donné ci-dessous (algorithme 37).

Algorithme 37 : Un algorithme de classification non hiérarchique pour données symboliques

Fixer le nombre de classes : k.
répéter
  Choisir k objets amorces
  pour chaque *amorce* **faire**
    Apprendre un ensemble de concepts discriminants vis-à-vis des autres amorces
  fin
  Modifier les concepts pour en déduire un ensemble de partitions sur les objets.
  Choisir la meilleure
jusqu'à *Le test de fin est satisfait*

Un certain nombre de commentaires sont nécessaires pour préciser cet algorithme.

- Le choix des amorces n'est effectué au hasard que la première fois. Quand la partition trouvée est meilleure que la meilleure partition courante, on cherche à l'améliorer encore en choisissant des objets « au centre » de chaque bloc. Sinon, on cherche à la corriger en choisissant des objets « au bord » de chaque bloc.
- La qualité d'une partition est évaluée par un mélange de critères parmi lesquels on peut citer :
  1. l'adéquation d'un concept, c'est-à-dire le nombre d'objets qu'il peut couvrir par rapport au nombre d'objets qu'il couvre dans l'ensemble à classer ;
  2. la simplicité d'un concept, que l'on peut mesurer par le nombre d'attributs qui y sont présents ;

[6] Nous ne donnons pas la formule générale, qui n'ajoute rien à la compréhension.

Copyright © 2010 Eyrolles.

3. l'intersection des concepts : elle est nulle dans l'ensemble des objets à classer, mais peut se mesurer dans l'absolu.
4. etc.

5.3 Les données logiques

Que devient l'apprentissage non supervisé dans des données décrites par des structures symboliques complexes comme les formules de la logique des prédicats ou les arbres ? Ce sujet est difficile, puisque la notion de distance ou de similarité dans ces espaces de représentation n'est pas naturelle. Pourtant, il est la clé du développement de la fouille de données pour la découverte de concepts évolués.

Il existe un certain nombre de travaux de conceptualisation et de réalisation dans ce domaine. Il faudrait introduire de nouveaux concepts et de nouvelles définitions pour aborder le sujet, même rapidement. Nous préférons donc ici renvoyer le lecteur à l'article de synthèse de G. Bisson dans [DKBM00] sur la notion de similarité dans les domaines statistiques et symboliques. C'est une excellente mise en lumière des problèmes et des solutions actuelles.

6. La fouille de données

La fouille de données (*data mining*) est un ensemble de méthodes et un domaine de recherche issu de préoccupations sur l'exploitation des bases de données qui se font de plus en plus importantes et de plus en plus répandues dans tous les champs d'activités. Jusque vers le milieu des années 1990, les recherches en bases de données portaient essentiellement sur l'organisation des données, leur représentation, leur stockage, les langages d'interrogation, leur distribution, etc. En un mot, sur le stockage et sur l'exploitation directe de toutes les données que l'on jugeait utile de conserver. Cependant, ces technologies ne permettaient que de retrouver une information stockée, et non pas de découvrir éventuellement des tendances, des régularités, implicitement contenues dans les bases de données. Avec l'avènement d'entrepôts de données[7] (*data warehouses*), il devenait impossible d'envisager que ces tâches de découverte soient entièrement réalisées manuellement. Tant le volume de données à traiter que l'impossibilité d'être prêt à traquer tout type de régularités potentiellement intéressantes rendaient nécessaire le recours à des techniques automatiques. Pour désigner celles-ci, on parle d'*extraction de connaissances à partir de données* (ECD) ou de fouille de données. Le sens de ce dernier vocable est cependant souvent restreint à l'étape de recherche de régularités dans les données, indépendamment des étapes de prétraitement et de post-traitement des données.

Les définitions de l'ECD sont nombreuses. Par exemple : c'est un *« processus complexe permettant l'identification, au sein des données, de motifs valides, nouveaux, potentiellement intéressants et les plus compréhensibles possible »* [FPSSU96]. C'est aussi *« le processus non trivial d'extraction d'informations valides, implicites, potentiellement utiles et ultimement compréhensibles à partir de grandes bases de données »* [HK06].

[7] Un *entrepôt de données* regroupe des informations hétérogènes en un seul lieu (logique) sous un format de données unifié, permettant le maintien facile de ces données et leur exploitation. Les opérations possibles incluent le « nettoyage » des données, leur intégration et leur traitement analytique en-ligne (*On-Line Analytical Processing* (OLAP)), c'est-à-dire des techniques permettant le résumé, l'agrégation de données, ou leur visualisation sous différents angles.

Copyright © 2010 Eyrolles.

L'ECD a donc pour but de fournir à un utilisateur final de nouvelles connaissances, disséminées dans un vaste ensemble de données, qui soient à la fois compréhensibles et utilisables vis-à-vis d'un contexte applicatif particulier. Elle met en jeu un processus interactif de préparation des données (sélection de descripteurs, constitution d'une table, discrétisation), d'extraction de connaissances à l'aide d'algorithmes de calcul, de visualisation et d'interprétation des résultats, lors d'interactions avec l'expert (voir figure 18.9).

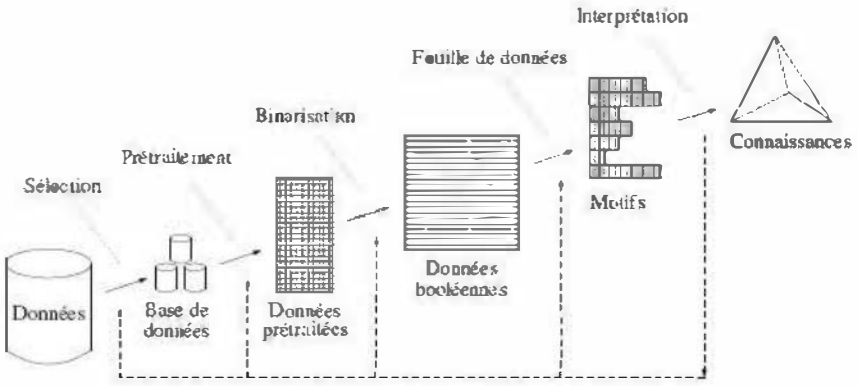


FIG. 18.9: *Cycle d'extraction de connaissances à partir de données.*

Les *principales étapes* d'un processus de fouille de données (ou *Knowledge Discovery from Databases* (KDD)) sont les suivantes :

1. **Le nettoyage des données**. Afin d'éliminer le bruit et les données incohérentes.
2. **L'intégration de données**. Lorsque les données proviennent de sources différentes.
3. **La sélection de données**. Recherche des données pertinentes dans la base de données.
4. **La transformation des données**. Afin de les mettre dans une forme appropriée pour les opérations de recherche de régularités. Cela peut inclure des opérations de résumé ou d'agrégation de données.
5. **La fouille de données**. Extraction de régularités dans les données.
6. **L'évaluation des régularités**. Afin de ne garder que les régularités intéressantes.
7. **La représentation des connaissances extraites.** Des techniques de visualisation et de représentation sont utilisées pour transmettre le plus clairement possible les régularités extraites aux utilisateurs finaux.

Les applications concernées sont toutes celles qui font appel à la technologie de grandes bases de données. Elles sont donc très largement répandues. Elles incluent par exemple l'analyse de marchés afin d'affiner des campagnes de marketing, la détection de fraudes, le contrôle de production ou encore l'exploitation de données scientifiques, particulièrement en astronomie ou en biologie. Les applications récentes se tournent vers les données disponibles sous forme de flux (*data streams*) ou accessibles sur le Web et dans les réseaux sociaux, ouvrant de nouvelles pistes et de nouvelles directions de recherche.

La fouille de données recouvre ainsi l'ensemble des techniques qui permettent de chercher de manière automatique des régularités intéressantes dans les bases de données (on parle abusivement d'« extraction des connaissances »). Le champ couvert est à la fois plus vaste que celui

Copyright © 2010 Eyrolles.

de l'apprentissage artificiel, en ceci qu'il met par exemple davantage l'accent sur les étapes de prétraitements ainsi que sur celles de mise en valeur des tendances mises à jour, par exemple par des outils sophistiqués de visualisation, mais aussi plus restreint, car il est essentiellement centré sur des techniques d'apprentissage non supervisé, ou à partir d'exemples positifs seuls.

En effet, il y a rarement de notion d'étiquettes dans les bases de données. Tous les champs d'un enregistrement se valent (en termes descriptifs) et l'on est intéressé à détecter des corrélations entre ces champs, ou descripteurs, sans référence à une classe.

Comme l'introduction de ce chapitre l'a abondamment souligné, il est impossible de chercher des régularités à l'aveuglette, sans avoir d'idées préconçues sur ce qui est cherché. Et, bien sûr, toute idée préconçue est empreinte d'arbitraire.

Le domaine de la fouille de données étant très actif, comme en témoignent l'importance des conférences qui lui sont annuellement consacrées (KDD[8], ECML-PKDD[9], ICDM[10], EGC[11] pour n'en citer que quelques unes), il n'est pas possible de fournir un catalogue exhaustif des types de régularités recherchées dans les bases de données. Cependant, l'essentiel des applications est concerné par les types de régularités suivants :

- **Description de classes ou concepts.** L'idée est de chercher une description des objets (enregistrements) associés à une classe, c'est-à-dire à une valeur de l'un (ou plusieurs) des descripteurs (champs). Par exemple, quel est le profil des auditeurs de France-Culture le samedi matin de 9h à 10h ? Ou bien, quel est le profil des produits dont les ventes ont augmenté de plus de 10 % l'année dernière ? Les techniques d'apprentissage supervisé peuvent naturellement fournir des outils pour découvrir ce type de régularité. Il faut noter que le but peut être de décrire et comprendre les données disponibles, comme il peut être de prédire quels objets pourraient faire partie de la classe étudiée.
- **Discrimination.** Il est également possible que le but soit de trouver ce qui permet de distinguer deux classes d'objets dans une base de données. Par exemple, ce qui distingue les acheteurs réguliers de produits informatiques des acheteurs très occasionnels, ou bien ce qui distinguer les produits dont les ventes ont augmenté de 10 % l'année dernière de ceux dont les ventes ont baissé de 30 %. Les techniques d'apprentissage supervisé de concepts sont naturellement *a priori* pertinentes pour cette tâche.
- **Recherche de motifs fréquents, de règles d'association.**
  - Les *motifs fréquents* sont, comme leur nom l'indique, des formes qui apparaissent fréquemment dans les données. Cela peut concerner des conjonctions d'attributs, des sous-séquences, ou des sous-structures. Ce que l'on appelle les *frequent itemset* ou FIS, désignent généralement des ensembles d'attributs qui sont fréquemment vrais ensemble dans les données. Par exemple, *pain*, *moutarde* et *saucisse*.
  - Les *règles d'association* (*association rules*) expriment sous forme de règles certaines corrélations en espérant ainsi identifier des causalités par exemple. Ainsi :

    `achète(X, ordinateur)` → `achète(X, logiciel)`.

    Comme il est possible de trouver un très grand nombre de telles règles dans toute base de données, seules celles qui passent un test de validité, à définir, sont proposées par le système.

[8] (KDD) *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, née en 1995. L'une des toutes premières conférences dans le domaine.

[9] (ECML-PKDD) *European Conference on Machine Learning & European Conference on Principles and Practice of Knowledge Discovery*. Née en 1986 pour la partie apprentissage, et en 1997 pour la partie fouille de données.

[10] (ICDM) *International Conference on Data Mining*, née en 2001.

[11] (EGC) *Journées francophones « Extraction et Gestion des Connaissances »*. La première conférence francophone dans le domaine, née en 2001.

Copyright © 2010 Eyrolles.

- **Les cas aberrants** (*outliers*). Parfois, ce qui est recherché est ce qui sort de la norme. Par exemple, si l'on veut identifier les fraudeurs au fisc. Dans ce cas, on utilisera des techniques dédiées, comme par exemple certains types de SVM (voir chapitre 14).

Les méthodes que nous décrivons dans la suite sont donc dédiées à l'exploitation des bases de données. Le vocabulaire utilisé est de ce fait largement influencé par ce dernier domaine. Les données traitées sont en général booléennes et non pas continues. L'accent en fouille de données a été en particulier mis sur la mise au point d'algorithmes performants pour traiter de (très) grosses bases de données. En revanche, les mesures d'intérêt sur les régularités extraites sont d'ordre essentiellement heuristique et peu de travaux ont considéré l'aspect statistique de l'extraction de régularités, central en induction (mais voir par exemple [PLT04]).

6.1 La recherche de motifs fréquents

Les méthodes de fouille de données peuvent être envisagées sous deux angles. Soit comme appartenant à la famille des méthodes d'apprentissage non supervisé, puisque, en général, l'apprentissage ne tient pas compte d'une classification cible, mais cherche des régularités présentes dans les données. Soit comme des méthodes d'apprentissage à partir d'exemples positifs seuls puisque, généralement, seuls des exemples positifs sont décrits dans les données (e.g. les clients d'une banque ou les complexes connus de protéines). Dans les deux cas, il est naturel de vouloir découvrir des motifs ou *patterns* communs aux données, ou, au moins, communs à une partie significative d'entre elles. C'est ce que l'on appelle les *motifs fréquents* ou *Frequent ItemSets* (FIS) pour des raisons que nous allons voir. Une grande partie des méthodes de fouille de données se base sur leur découverte.

6.1.1 Vocabulaire, définitions et propriétés

La recherche de motifs fréquents a pour but d'identifier des caractéristiques communes aux exemples ou enregistrements $\mathcal{S} = \{\mathbf{x}_1, .., \mathbf{x}_m\}$ présents dans une base de données et décrits par un ensemble d'attributs [12] $A = \{a_1, ..., a_d\}$.

Ainsi, un exemple $\mathbf{x}_i \in \mathcal{S}$, qui correspond à un certain nombre d'attributs de A à valeur « vrai », peut être vu comme un sous-ensemble de A et vérifiera donc aussi $\mathbf{x}_i \subseteq A$. Un attribut dont la valeur est « 1 » ou « vrai » est appelé un *item*. Un ensemble d'items est appelé *itemset*, d'après l'appellation originale américaine. Un itemset est donc un sous-ensemble de tous les attributs binaires valant 1 pour un exemple donné.

Définition 18.1 (Couverture d'un exemple)

On dira qu'un ensemble d'items $\mathcal{I}$ couvre *un exemple $\mathbf{x}_i$ si et seulement si $\mathcal{I} \subseteq \mathbf{x}_i$. Cela signifie que $\mathbf{x}_i$ vérifie toutes les conditions booléennes exprimées par les attributs de $\mathcal{I}$.*

—— Exemple ——

Pour illustrer ces notions, l'exemple de la table 18.1 représente le résultat d'un sondage fictif réalisé auprès de touristes étrangers en visite à Paris. Les objets correspondent à des touristes anonymes ($\mathbf{x}_i$) et les attributs représentent les lieux visités : les bateaux mouches (BM), le

[12] Parce que c'est un cas très répandu, et par souci de simplicité, nous ne considérons ici que le cas des attributs booléens (descripteur présent ou non. Par exemple « a acheté du pain »). Cependant, les concepts et méthodes décrits s'étendent sans problème au cas des attributs ordinaux (c'est-à-dire à valeur entière, par exemple « a acheté 2 baguettes »).

Copyright © 2010 Eyrolles.

	Attributs					
Observations	BM	CP	LO	MO	ND	TE
$\mathbf{x}_1$	×		×	×	×	
$\mathbf{x}_2$	×		×		×	
$\mathbf{x}_3$	×		×	×		×
$\mathbf{x}_4$	×		×	×	×	
$\mathbf{x}_5$	×		×	×	×	×
$\mathbf{x}_6$	×	×	×		×	
$\mathbf{x}_7$	×				×	
$\mathbf{x}_8$			×		×	
$\mathbf{x}_9$			×		×	
$\mathbf{x}_{10}$	×	×	×	×	×	

TAB. 18.1: *Base de données sur des visites de Paris.*

centre Pompidou (CP), le musée du Louvre (LO), le musée d'Orsay (MO), la cathédrale Notre-Dame (ND) et la tour Eiffel (TE).
Ainsi, en s'appuyant sur le vocabulaire précédent, on peut dire que l'exemple $\mathbf{x}_1$ est égal à $\{BM, LO, MO, ND\}$ et que l'item CP couvre les exemples $\mathbf{x}_6$ et $\mathbf{x}_{10}$.
$\{BM, LO\}$ et $\{MO, TE\}$ sont des itemsets de $\mathbf{x}_5$.

La notion de *motif* étend celle d'item. Celle de *motif fréquent* repose sur les notions de *support* et de *fréquence*, pour lesquelles il existe quelques légères variantes mathématiques sans grandes conséquences sémantiques.

Définition 18.2 (Motif)

Un motif *est un ensemble d'items. C'est donc un ensemble d'attributs dont la valeur doit être vraie.*

Définition 18.3 (Couverture d'un motif)

La couverture *d'un motif est l'ensemble des exemples couverts à la fois par tous les items du motif.*

Définition 18.4 (Support d'un motif)

Le support *d'un motif est le cardinal de la couverture de ce motif.*

Définition 18.5 (Fréquence d'un motif)

La fréquence *d'un motif est égale à son support divisé par le nombre total d'exemples.*

Remarque : Il faut noter que la fréquence est parfois appelée support par certains auteurs.

La taille t d'un motif m_i est définie comme le cardinal de ses items, c'est-à-dire $t(m_i) = |m_i|$. Ainsi, la taille du motif $\{BM, TE\}$ est égale à 2.

On dira qu'un motif est fréquent si sa fréquence est supérieure à un seuil défini *a priori* noté f_min.

Copyright © 2010 Eyrolles.

— EXEMPLE —

Ainsi, en reprenant la base de données de la table 18.1, et en considérant le motif $\{BM, LO\}$, nous avons :

$$\text{Couverture}(\{BM, LO\}) = \{x_1, x_2, x_3, x_5, x_{10}\};$$
$$\text{Support}(\{BM, LO\}) = 5;$$
$$\text{Fréquence}(\{BM, LO\}) = 0.5.$$

Une tâche centrale en fouille de données consiste à **trouver tous les motifs fréquents** (ou *Frequent ItemSets* FIS), c'est-à-dire dont la fréquence est supérieure à f_min.

Or, d'après la définition d'un motif, M_A, qui représente l'ensemble des motifs que l'on peut construire à partir d'un ensemble d'attributs A, est exactement l'ensemble des parties de A (sauf la partie vide). En effet, il est égal à l'ensemble des motifs de taille 1, plus l'ensemble des motifs de taille 2, ..., plus l'ensemble des motifs de taille D (nombre d'attributs).

$$\sum_{i=1}^{D} \binom{D}{i} = 2^D - 1$$

Ainsi, dans le cas de $D = 100$ le nombre total de motifs est d'environ 1.27×10^{30}. Il est clair que l'on arrive très vite à des ensembles qu'aucun ordinateur ne peut traiter directement.

— EXEMPLE —

Dans la base de données de la table 18.1, les motifs fréquents au niveau 0.7 (couverture d'au moins 70 %) sont $\{BM, LO, ND\}$ et tous leurs sous-motifs de taille 2 $\{BM, LO\}$, $\{BM, ND\}$, $\{LO, ND\}$ et de taille 1 $\{BM\}$, $\{LO\}$, $\{ND\}$.

On observera sur cet exemple que tous les sous-motifs (définis à partir des sous-ensembles des items du motif considéré) d'un motif ont un support au moins aussi élevé que celui du motif. Cette propriété est essentielle pour concevoir des algorithmes de recherche des motifs fréquents efficaces malgré l'immensité de l'espace des motifs à explorer.

Propriété 18.2 (Antimonotonicité de la couverture et du support)

Puisque tout sous-ensemble non vide m_1 d'un motif fréquent m_2 (au niveau f_min) est nécessairement également fréquent, nous avons :

$$\begin{array}{rcl} \text{Couverture}(m_2) & \subseteq & \text{Couverture}(m_1) \\ \text{Support}(m_2) & \leq & \text{Support}(m_1) \end{array}$$

*On parle d'*antimonotonicité *de la couverture et du support car leurs relations d'ordre sont inversées par rapport à celle de l'inclusion des motifs entre eux.*

En conséquence de cette propriété, nous avons pour tous motifs m_1 et m_2 de M :

$$\begin{array}{rcl} \text{Couverture}(m_2) & = & \text{Couverture}(m_1) \cap \text{Couverture}(m_2) \\ \text{et : } \text{Support}(m_1 \cup m_2) & \leq & \min(\text{Support}(m_1), \text{Support}(m_2)) \end{array}$$

6.1.2 Algorithmes de recherche de motifs fréquents

Les algorithmes discutés ici cherchent les (ou des) motifs fréquents au sens d'un seuil de fréquence f_min prédéfini. Fréquemment, les applications étudiées impliquent des matrices de

Copyright © 2010 Eyrolles.

données {0,1} très grandes, avec un nombre de transactions (exemples) allant de 10^5 à 10^8, voire plus, et un nombre d'attributs allant de 10^2 à 10^6. Ces matrices sont souvent très clairsemées, car les transactions ne comportent souvent que très peu d'items (par exemple, moins de 0.1%).

La méthode la plus directe de recherche de motifs fréquents serait de calculer tous les motifs ainsi que leur fréquence associée. En raison du nombre énorme des motifs possibles, cette méthode est vouée à l'échec. Heureusement, la propriété d'antimonotonicité du support permet d'organiser la recherche de manière efficace. En effet, un motif ne peut être fréquent que si tous ses sous-motifs sont fréquents. Il est donc possible d'élaguer l'espace de recherche en n'examinant pas les sur-motifs (sur-ensemble d'items) d'un motif non fréquent.

Ainsi, une méthode de génération ascendante (*bottom-up*) est possible. On cherche d'abord tous les motifs de taille 1 fréquents. Une fois ceux-ci identifiés, on engendre les motifs candidats de taille 2 : ensembles de la forme $\{m_A, m_B\}$ où m_A et m_B sont de taille 1 et fréquents. Il faut alors ne retenir que les motifs fréquents parmi eux. À partir de là, on peut construire les motifs de taille 3, puis sélectionner parmi eux les motifs fréquents. Et ainsi de suite. D'où l'algorithme `Apriori` (voir algorithme 38) décrit pour la première fois, et indépendamment, par [AS94] et [MTV94].

Algorithme 38 : `Apriori` : recherche des motifs fréquents

Données : $\mathcal{S}$ collection des exemples x décrits par les attributs A, f_{min} le seuil de fréquence minimale fixé
Résultat : M_f l'ensemble des motifs fréquents avec leur fréquence associée
début
  $i = 0$
  $C_i = \{\{A_j\} | A_j$ est un item$\}$
  tant que $C_i \neq \emptyset$ **faire**
    `/* Etape 1 : Passe sur la base de données */`
    pour $m \in C_i$ **faire**
      `/* Tester si` m `est fréquent. Si oui, l'ajouter à` M_f `*/`
      si Fréquence$(m) \geq f_{min}$ **alors**
        $L_i \longleftarrow L_i \cup \{m\}$
      fin si
    fin pour
    `/* Etape 2 : Engendrer les motifs candidats d'ordre supérieur */`
    $i \longleftarrow i + 1$
    $C_i \longleftarrow$ motifs de taille i dont tous les sous-motifs sont fréquents
  fin tant que
  $M_f \longleftarrow \bigcup(L_i)$
  retourner M_f
fin

— EXEMPLE **Fonctionnement de l'algorithme `Apriori`** —

En supposant une fréquence minimale seuil de $f_{min} = 0.2$, à la quatrième étape du traitement de notre exemple, on a les motifs fréquents de taille 4 :

$$L_4 = \{ \{BM, LO, MO, ND\}, 0.4), (\{BM, LO, MO, TE\}, 0.2), (\{BM, CP, LO, ND\}, 0.2) \}$$

Copyright © 2010 Eyrolles.

Par exemple le 4-itemset $(\{BM, LO, MO, ND\}, 0.4)$ couvre les exemples x_1, x_4, x_5 et x_{10}.
On construit alors l'ensemble C_5 des motifs candidats de taille 5 à partir des motifs de L_4.
$C_5 = \{(\{BM, LO, MO, ND, TE\}, 0.1), (\{BM, CP, LO, MO, ND\}, 0.1)\}$
Aucun de ces motifs n'est fréquent au seuil $f_{min} = 0.2$ et le processus de recherche s'arrête donc là, en retournant $M_{0.2} = \bigcup_{i=1..4}(L_i)$.

La complexité de l'algorithme `Apriori` est fonction des deux étapes principales : la *formation des motifs* candidats par niveau croissant (algorithme en largeur-d'abord, aussi appelé parfois *levelwise algorithm*), et le *calcul de leur fréquence* dans la base de données.

La formation des candidats se fait simplement. Supposons que nous ayons la collection L_i des motifs fréquents au niveau i (c'est-à-dire de taille $\leq i$). Nous cherchons les motifs candidats de taille $i+1$. Cela peut se faire en considérant tous les couples de motifs $\{m_A, m_B\}$ appartenant à L_i dont l'union est de taille $i+1$. Il y a donc de l'ordre de $|L_i|^2$ ensembles à considérer au maximum. Et pour chacun d'eux l'opération d'union est de complexité de l'ordre de $|L_i|$. La complexité en pire cas est donc en $\mathcal{O}(|L_i|^3)$. En pratique, la complexité est souvent linéaire en $|L_i|$ car, dans le cas de matrices très clairsemées, il y a peu d'intersection entre les motifs.

Pour chaque candidat m_k, il est aisé de calculer sa fréquence par une passe dans la base de données pour compter son support. Le temps requis est de l'ordre de la taille de la base de données $\mathcal{S}$. Des structures de données permettent d'accélérer cette étape si besoin est.

La complexité totale en pire cas est donc en $\mathcal{O}(K|L_i|^3|\mathcal{S}|)$ (où K est la taille maximale des motifs fréquents) et, de fait, souvent en $\mathcal{O}(|L_i||\mathcal{S}|)$, voire moins avec des techniques de calcul appropriées aux caractéristiques de la matrice de données.

6.1.3 Généralisations

La méthode générale utilisée dans l'algorithme `Apriori` peut facilement s'étendre à d'autres types de données et de motifs à partir du moment où des propriétés essentielles de monotonicité ou d'antimonotonicité sont vérifiées.

En particulier, une propriété important est que les motifs, quelle que soit leur structure (séquence, arbre, ...), soient des conjonctions de motifs atomiques pris dans un ensemble $\mathcal{A}$, c'est-à-dire que l'ensemble possible $\mathcal{M}$ des motifs contienne les motifs du type : $m_i \wedge m_k \wedge \ldots \wedge m_r$ où $m_l \in \mathcal{A}, \forall l$. Ces conjonctions pouvant aussi être considérées comme des conjonctions de contraintes sur les exemples les vérifiant, on a alors naturellement une relation d'ordre antimonotonique de la couverture et donc du support et de la fréquence au sens où :

$$\forall m_1, m_2 \in \mathcal{M} : \quad m_1 \subseteq m_2 \quad \Rightarrow \quad \text{Support}(m_1) \geq \text{Support}(m_2)$$

Ces propriétés sont immédiates pour tout un ensemble de types de motifs : motifs ordinaux (attributs à valeur entière), séquences, arbres, etc. Et elles permettent qu'une recherche ascendante en largeur d'abord élague aussitôt que possible les branches stériles.

6.2 Les règles d'association

Les règles d'association sont l'une des représentations les plus populaires pour l'expression de corrélations locales en fouille de données.

Supposons que le monde qui nous intéresse soit décrit à l'aide de transactions (lignes) dans une base de données. Il est fréquent d'en tirer un ensemble de variables booléennes dénotant la présence (`vrai`) ou l'absence (`faux`) d'items de chaque type. Par exemple, les achats d'un client dans un supermarché peuvent être décrits par les types d'items qu'ils contiennent. On peut alors

Copyright © 2010 Eyrolles.

être intéressé à découvrir que les clients qui achètent tel type d'ordinateur achètent également un antivirus. On pourrait ainsi obtenir, par recherche dans la base de données des achats effectués dans le supermarché, une règle de la forme :

$$\texttt{ordinateur} \Rightarrow \texttt{antivirus} \quad [support = 3\%, confiance = 60\%]$$

ou encore, en supposant que la base de donnée comporte également des données sur les acheteurs :

$$\texttt{age(x,[50, 75])} \wedge \texttt{revenu(x,<20 K€)} \Rightarrow \texttt{achète(x,pain)} \quad [support = 2\%, confiance = 50\%]$$

Le *support* et la *confiance* sont deux mesures, parmi d'autres, pour estimer l'intérêt des règles considérées. Elles sont destinées à qualifier respectivement l'utilisabilité et la certitude des règles découvertes. Par exemple, un support de 3 % pour la première règle ci-dessus signifie que dans 3 % des transactions analysées, un ordinateur et un antivirus ont été achetés ensemble. Une confiance de 60 % signifie que 60 % des acheteurs qui ont acquis un ordinateur ont également acheté un antivirus.

Typiquement, on cherche des règles qui vérifient un support et une confiance qui sont au-dessus de seuils minimaux fixés *a priori*. On dit que ces règles d'association sont *fortes*.

6.2.1 Définition et concepts essentiels

Définition 18.6 (Couverture, support d'une règle d'association)

On appelle couverture *(ou* support*) de* $a_1 \Rightarrow a_2$ *la probabilité* $P(a_1, a_2)$ *que* a_1 *et* a_2 *soient Vrai en même temps. Comme* $P(a_1, a_2) = P(a_2, a_1)$*, la couverture de* $a_1 \Rightarrow a_2$ *est la même que celle de* $a_2 \Rightarrow a_1$*.*

Définition 18.7 (Confiance d'une règle d'association)

On appelle confiance *de* $a_1 \Rightarrow a_2$ *la probabilité* $P(a_2|a_1)$ *que* a_2 *soit vérifiée quand* a_1 *l'est.*

6.2.2 Calcul des règles d'association à partir des motifs fréquents

La recherche de règles d'association peut être décomposée en deux étapes principales :

1. La *recherche de motifs fréquents*. Par définition, ces motifs, ou itemsets, ont une fréquence au moins aussi grande que le seuil de support minimum : f_min.
2. La *recherche de règles d'association fortes à partir des motifs fréquents* qui vérifient à la fois un support et une confiance minimales.

Nous avons vu dans la section 6.1 que la recherche des motifs fréquents pouvait se faire efficacement en tenant compte de l'antimonotonicité du support.

Une fois que les motifs fréquents (satisfaisant un support minimal) ont été extraits, il est relativement facile d'engendrer les règles d'association fortes. On cherche en effet les règles de confiance supérieure à un seuil donné, sachant, encore une fois, que la confiance s'exprime comme :

$$confiance(a_1 \Rightarrow a_2) = P(a_2|a_1) = \frac{support(a_1 \cup a_2)}{support(a_1)} = \frac{n\_support(a_1 \cup a_2)}{n\_support(a_1)}$$

où $n\_support(a)$ dénote le nombre de transactions contenant a.

On peut alors engendrer les règles d'association fortes comme suit :

- pour tout motif fréquent ℓ, engendre tous les sous-ensembles non vides de ℓ ;

Copyright © 2010 Eyrolles.

- pour tout sous-ensemble non vide s de ℓ, produire en sortie la règle « $s \Rightarrow (\ell \setminus s)$ » si $\frac{n\_support(\ell)}{n\_support(s)} \geq min\_confiance$, où $min\_confiance$ est le seuil minimal de confiance exigé.

Comme les règles sont engendrées à partir de motifs fréquents, elles satisfont automatiquement au seuil de support minimal. Les motifs fréquents peuvent être stockés à l'avance dans des tables de hachage avec leur nombre d'occurrence associé afin d'accélérer leur manipulation.

EXEMPLE **Génération des règles d'association**

Soit une table de transactions (table 18.2.

	Attributs				
Observations	$a1$	$a2$	$a3$	$a4$	$a5$
x_1	a1	a2			a5
x_2		a2		a4	
x_3		a2	a3		
x_4	a1	a2		a4	
x_5	a1		a3		
x_6		a2	a3		
x_7	a1		a3		
x_8	a1	a2	a3		a5
x_9	a1	a2	a3		

TAB. 18.2: *Une table de transactions.*

Supposons que la table contienne le motif fréquent $\ell = \{a1, a2, a5\}$. Quelles règles d'association peuvent être engendrées?

Les sous-ensembles non vides de $\{a1, a2, a5\}$ sont : $\{a1, a2\}, \{a1, a5\}, \{a2, a5\}, \{a1\}, \{a2\}$ et $\{a5\}$.

Les règles d'association correspondantes sont les suivantes, avec leur confiance associée :

$a1 \wedge a2 \Rightarrow a5$ $\quad confiance = 2/4 = 50\%$
$a1 \wedge a5 \Rightarrow a2$ $\quad confiance = 2/2 = 100\%$
$a2 \wedge a5 \Rightarrow a1$ $\quad confiance = 2/2 = 100\%$
$a1 \Rightarrow a2 \wedge a5$ $\quad confiance = 2/6 = 33\%$
$a2 \Rightarrow a1 \wedge a5$ $\quad confiance = 2/7 = 29\%$
$a5 \Rightarrow a1 \wedge a2$ $\quad confiance = 2/2 = 100\%$

6.2.3 L'évaluation des règles d'association

L'un des gros avantages des notions de *support* et de *confiance* est leur grande intelligibilité ce qui est très séduisant pour l'utilisateur non expert. Leur utilisation exclusive présente cependant certains inconvénients et limites.

D'abord, elles ont tendance à engendrer un grand nombre de règles, dont beaucoup n'ont pas d'intérêt car elles n'apportent pas d'information nouvelle. Par exemple, il se peut que l'on soit intéressé par des règles exceptionnelles (cas des services des impôts qui cherchent les fraudeurs). On peut alors être tenté d'abaisser les seuil de support minimal, mais cela provoque généralement une explosion du nombre de règles produites.

De même, on peut avoir des règles $a \Rightarrow b$ avec $P(b|a) = P(b)$, ce qui signifie que a et b sont indépendants.

Copyright © 2010 Eyrolles.

Il est donc nécessaire d'avoir recours à d'autres mesures d'intérêt. Nous reportons le lecteur intéressé à [LT04, LTP07] pour une revue et une analyse d'un grand nombre de mesures existantes.

Par ailleurs, il faut être conscient, que, à l'instar du test multiple (voir section 4.1 du chapitre 2) ou de la recherche d'hypothèses rendant compte au mieux des données d'apprentissage, il existe le risque qu'une règle d'association soit trouvée intéressante, alors qu'elle présente accidentellement les propriétés recherchées. Étant donné le grand nombre de règles produites, il faut pouvoir quantifier la proportion de ces règles dues à des corrélations fortuites. Une analyse statistique, par exemple du type de celle de Vapnik est alors nécessaire (voir par exemple [TL01]).

6.3 Autres problèmes en fouille de données

La recherche de règles d'association est une opération de base en fouille de données. Cependant, d'autres tâches sont également importantes, dont certaines sont encore à la frontière de la recherche.

Ainsi, dans les techniques décrites, on a ignoré la possibilité que les bases de données soient entachées d'imperfections et qu'en conséquence les *données* soient *bruitées*. L'analyse de la qualité des règles produites en présence d'un certain taux de bruit demande d'étendre les techniques d'analyse évoquées plus haut et qui sont elles-mêmes ignorées la plupart du temps des praticiens.

Un problème qui provoque de nombreux travaux depuis quelques années est celui de la découverte des « sous-groupes » (*subgroup discovery*). Il s'agit essentiellement de découvrir des sous-ensembles de variables en inter-relation, vérifiant à la fois chacun des critères d'intérêt de nature statistique, mais ayant également des propriétés globales, telles que par exemple ces sous-ensembles soient peu nombreux, se recouvrent peu, mais couvrent au mieux l'ensemble des transactions de la base de données. De manière intéressante, ces travaux redécouvrent de nombreux concepts caractérisant l'apprentissage dans des espaces d'hypothèses structurés par une relation de généralité (chapitre 4 en particulier).

Un autre problème important est celui de la recherche de motifs fréquents et de règles d'associations quand les descripteurs ne sont plus seulement des variables booléennes, mais peuvent prendre des *valeurs numériques*. Ainsi, on peut être intéressé non seulement au fait qu'un acheteur ait acquis des actions d'une certaine entreprise, mais aussi combien. On parle alors de recherche dans des multi-ensembles, par contraste aux techniques décrites qui traitent les ensembles d'items.

Plus généralement, le type des motifs ou des régularités recherchés peut inclure les *motifs séquentiels* (ou épisodes), les *motifs structurés* ou *hiérarchiques*, etc. Si les techniques de base restent similaires, il faut cependant les adapter avec soin pour qu'elles conservent leur faible coût calculatoire et qu'elles puissent être employées dans des grosses bases de données.

Un autre problème est celui de l'extraction automatique de *liens causaux*. Il s'agit là d'un problème notoirement difficile, déjà discuté dans le cadre des réseaux bayésiens (chapitre 11).

Pour finir, nous signalerons aussi les travaux pour réaliser des *fouilles de données incrémentales* qui puissent s'adapter aux flux de données (voir aussi le chapitre 20).

7. Les analyses en composantes

Les méthodes utilisant un dictionnaire de fonctions de base présupposent la sélection de ces fonctions.

Copyright © 2010 Eyrolles.

Deux démarches existent. Soit utiliser un code linéaire choisi *a priori*, comme les transformées de Fourier ou les représentations en ondelettes. Soit utiliser des représentations dépendantes des données, qui sont ajustées automatiquement aux statistiques des entrées. De telles représentations sont apprises directement à partir des données en optimisant des mesures qui quantifient leurs propriétés désirables. Cette classe de méthodes inclue l'*analyse en composantes principales* (ACP), l'*analyse en composantes indépendantes* (ICA) [Hyv99, HO00, HKO01b] et la *factorisation non négative en matrices* (NMF) [LS99].

7.1 Analyse en composantes principales (ACP)

L'analyse en composantes principales (ACP) est une technique linéaire optimale de réduction de dimension au sens des moindres carrés. À partir d'un ensemble $\mathcal{S}$ de m exemples (vecteurs) d'apprentissage $\mathcal{S} = \langle \mathbf{x}_1, \ldots, \mathbf{x}_m \rangle$ définis dans l'espace d'entrée $\mathcal{X}$, on calcule la moyenne ($\mu = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i$) et la matrice de covariance ($\Sigma = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^\top$). On peut alors définir une matrice de projection P composée des p vecteurs propres de Σ associés aux valeurs propres les plus grandes. La représentation d'un vecteur $\mathbf{x}$ dans le sous-espace projeté de $\mathcal{X}$ s'obtient par : $\mathbf{x}_p = P^\top(\mathbf{x} - \mu)$.

L'idée de base de l'analyse en composantes principales (ACP) est de trouver les composantes $\phi_1, \phi_2, \ldots, \phi_p$ qui rendent compte au mieux de la variance du signal par p composantes transformées linéairement. La méthode consiste à chercher les p vecteurs propres correspondant aux p plus grandes valeurs propres de la matrice de covariance. Mais cette méthode suppose que le signal est fortement (totalement gaussien) car seule cette partie du signal est prise en compte.

L'analyse en composantes principales produit des représentations globales et non locales de l'espace d'entrée : des sous-espaces propres.

7.2 Analyse en composantes indépendantes (ACI)

L'analyse en composantes indépendantes (ACI) (*Independent Component Analysis* ou ICA, en anglais) et la séparation de sources (*Blind Source Separation* ou BSS) concernent la recherche de signaux indépendants à partir de mesures qui en sont des combinaisons linéaires. Par exemple, dans le cas des sonars passifs qui tapissent les océans en certains lieux stratégiques, les microphones enregistrent des signaux qui sont des combinaisons provenant de plusieurs sources telles que bancs de crevettes, hélices de sous-marins, bateaux de pêche, etc. Chaque senseur mesure une combinaison différente puisque placé en un endroit distinct des autres. Le but est alors de séparer et d'identifier les différentes sources de sons.

On trouve ce problème à chaque fois qu'il existe un ensemble de signaux engendrés indépendamment (que l'on appelle *sources* ou encore *variables latentes*), $\mathbf{s} = [s_1, \ldots, s_n]^\top$, un milieu assurant un mélange linéaire $\mathbf{A}$ et un certain nombre d de mesures : $\mathbf{x} = [x_1, \ldots, x_d]^\top$, avec $\mathbf{x} = \mathbf{A}\,\mathbf{s}$. Ainsi, chaque mesure x_j résulte d'une combinaison linéaire de signaux : $x_j = a_{j1}s_1 + \ldots + a_{jn}s_n$.

Le vecteur *source* $\mathbf{s} \in \mathbb{R}^n$ est supposé tiré d'une densité de distribution $\mathbf{p}(\mathbf{s})$, dont les coordonnées sont indépendantes, c'est-à-dire :

$$\mathbf{p}(s_1, \ldots, s_n) = \mathbf{p}_1(s_1)\mathbf{p}_2(s_2)\ldots\mathbf{p}_n(s_n) \tag{18.3}$$

où $\mathbf{p}_i(s_i)$ représente la densité marginale de s_i.

Les sources $\mathbf{s}$ ne sont pas observées directement, et la forme des distributions de probabilités $\mathbf{p}_i(s_i)$ est inconnue. Seul le mélange $\mathbf{x} \in \mathbb{R}^d$ est observé.

La tâche de l'analyse en composantes indépendantes est alors de trouver la matrice de mélange $\mathbf{A}$, et donc les sources indépendantes $\mathbf{s}$, à partir de l'observation d'un ensemble de vecteurs $\mathbf{x}$.

Copyright © 2010 Eyrolles.

Il faut noter que, dans ce modèle linéaire, les sources ne peuvent être identifiées qu'à une permutation et une normalisation près.

7.2.1 Principe des méthodes d'ACI

Les méthodes d'analyse en composantes indépendantes reposent sur un théorème central limite classique en théorie des probabilités selon lequel *la distribution d'une somme de variables aléatoires indépendantes tend vers une distribution gaussienne*, sous certaines conditions.

Ainsi, supposons que le vecteur de données $\mathbf{x}$ ait une distribution vérifiant le modèle $\mathbf{x} = \mathbf{A}\,\boldsymbol{s}$, c'est-à-dire soit un mélange de composantes indépendantes. Nous supposerons de plus, pour simplifier, que ces composantes sont toutes de même distribution. Nous allons maintenant voir comment estimer chaque composante indépendante s_i.

Soit une combinaison linéaire des mesures x_j que nous noterons $y = \mathbf{w}^\top \mathbf{x} = \sum_i \mathbf{w}_i x_i$ où $\mathbf{x}$ est un vecteur de poids à déterminer. Si $\mathbf{w}$ était une des lignes de la matrice inverse de $\mathbf{A}$, cette combinaison linéaire serait en fait égale à l'une des composantes indépendantes recherchées s_i puisque $\mathbf{s} = \mathbf{A}^{-1}\mathbf{x}$. Dans l'ignorance de la matrice de mélange $\mathbf{A}$, il est cependant possible d'estimer les vecteurs $\mathbf{w}$.

En effet, on peut poser $y = \mathbf{w}^\top \mathbf{x} = \mathbf{w}^\top \mathbf{A}\mathbf{s} = \mathbf{z}^\top \mathbf{s}$. Donc y est une combinaison linéaire de s_i dont les poids sont donnés par z_i. En conséquence du théorème central limite cité plus haut, cette combinaison est davantage gaussienne que chacune de ces composantes, sauf dans le cas où y est en fait égale à l'une seulement des composantes. Il faut donc chercher à minimiser le caractère gaussien de $\mathbf{w}^\top \mathbf{x}$ en jouant sur le vecteur $\mathbf{w}$. Le paysage d'optimisation associé dans l'espace à n dimensions des vecteurs $\mathbf{w}$ présente $2n$ minima locaux, deux par composantes (correspondant à s_i et $-s_i$). La recherche de ces minima est grandement facilitée par le fait que les composantes sont décorrélées. On peut en effet contraindre la recherche à trouver des estimations décorrélées avec les précédentes.

Il reste à préciser la mesure du caractère gaussien utilisée.

7.2.2 Mesures d'indépendance statistique

Pour simplifier, on supposera que les variables y sont centrées (de moyenne nulle) et de variance unité. Il est facile de normaliser y pour se trouver dans ce cas.

Plusieurs méthodes de mesure d'indépendance statistique ont été proposées.

- **Kurtosis**. La mesure la plus classique du caractère non gaussien d'une distribution est le moment d'ordre 4, encore appelé *kurtosis*. Pour une variable aléatoire y :

$$\mathrm{Kurt}(y) = \mathrm{E}\{y^4\} - 3(\mathrm{E}\{y^2\})^2$$

  Pour une variable de distribution gaussienne, $\mathrm{E}\{y^4\} = 3(\mathrm{E}\{y^2\})^2$, le kurtosis est nul. Le kurtosis, ou plutôt sa valeur absolue, est très utilisé dans l'analyse en composantes indépendantes, en raison de sa simplicité à la fois théorique et calculatoire. Malheureusement, le kurtosis est très sensible aux valeurs aberrantes et n'est donc pas robuste.

- **Néguentropie**. La *néguentropie* est une quantité d'information liée à l'entropie. L'entropie d'une variable aléatoire peut être interprétée comme une mesure de l'information apportée par cette variable. Plus la variable prend des valeurs d'allure aléatoire, imprévisible et non structurée, plus grande est son entropie. De fait, l'entropie est liée à la longueur de codage minimale des valeurs prises par la variable (voir la section 11). Pour une variable discrète Y :

$$H(Y) = -\sum_i P(Y = a_i) \log P(Y = a_i)$$

Copyright © 2010 Eyrolles.

où les a_i sont les valeurs possibles de Y. Dans le cas d'une variable continue y de densité $f(\mathbf{y})$, l'entropie (différentielle) est :

$$H(\mathbf{y}) = -\int f(\mathbf{y}) \log f(\mathbf{y}) \mathrm{d}\mathbf{y}$$

Un résultat fondamental de la théorie de l'information est qu'une variable gaussienne est d'entropie maximale parmi toutes les distributions de même variance. L'entropie peut donc être employée pour mesurer le caractère gaussien d'une distribution. En fait, la néguentropie J est utilisée car elle est nulle pour une variable gaussienne et positive pour les autres :

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y})$$

où $\mathbf{y}_{gauss}$ est une variable aléatoire gaussienne de même matrice de covariance que y. Si la néguentropie est un estimateur optimal du caractère gaussien du point de vue de la théorie statistique, elle est malheureusement difficile à calculer. C'est pourquoi on lui préfère souvent des approximations.

- **Information mutuelle**. L'information mutuelle entre n variables aléatoires $y_i, i = 1 \ldots n$ est définie comme :

$$I(y_1, \ldots, y_n) = \sum_{i=1}^{n} H(y_i) - H(\mathbf{y})$$

où y est le vecteur dont les composantes sont les y_i. L'information mutuelle est équivalente à la divergence de Kullback-Leibler entre la densité jointe $f(\mathbf{y})$ et le produit de ses densités marginales, ce qui est une mesure naturelle d'indépendance. Cette quantité est positive, sauf dans le cas de variables indépendantes.

D'autres mesures d'indépendance ont été proposées, dont par exemple le principe *Infomax* utilisé dans des perceptrons multi-couches (voir le chapitre 10). Pour davantage de détails sur ces mesures et sur l'analyse en composantes indépendantes en général, voir par exemple [HO00].

7.2.3 Algorithmes et applications

Plusieurs algorithmes existent pour l'analyse en composantes indépendantes, dont l'algorithme FastICA réputé pour sa rapidité et disponible sous forme de bibliothèque Matlab[13]. L'analyse en composantes indépendantes a été appliquée aux problèmes de séparation de sources (identification de m sources produisant par mélange linéaire un signal mesuré en m points) et d'extraction de composantes d'un signal.

—— EXEMPLE **Analyse d'images par ACI** ——

L'analyse d'images entre dans ce dernier cadre. L'idée est alors d'extraire un code, c'est-à-dire un catalogue, constitué d'« images de base », dans lequel toute image du domaine considéré (correspondant à une certaine famille de distributions de probabilités) peut être représentée fidèlement.

Étant données la complexité des calculs et la taille limitée souhaitée pour le catalogue d'images de base, les expériences réalisées ne travaillent pas directement sur les images, mais sur des « imagettes » (par exemple 8×8 ou 12×12). Les figures 18.10 et 18.11 montrent respectivement la décomposition linéaire d'une imagette 12×12 et une base d'imagettes obtenues par ICA à partir d'images en couleurs[14].

[13] Le paquage Matlab FastICA est disponible sur `http ://www.cis.hut.fi/projects/ica/fastica/`.

[14] Transparents fournis par Patrick Hoyer et Aapo Hyvarïnen.

Copyright © 2010 Eyrolles.

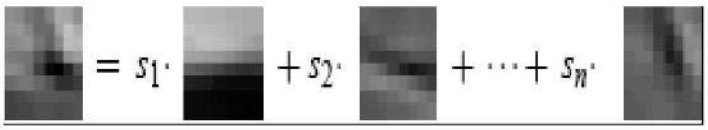


FIG. 18.10: *Principe de décomposition linéaire d'une imagette*

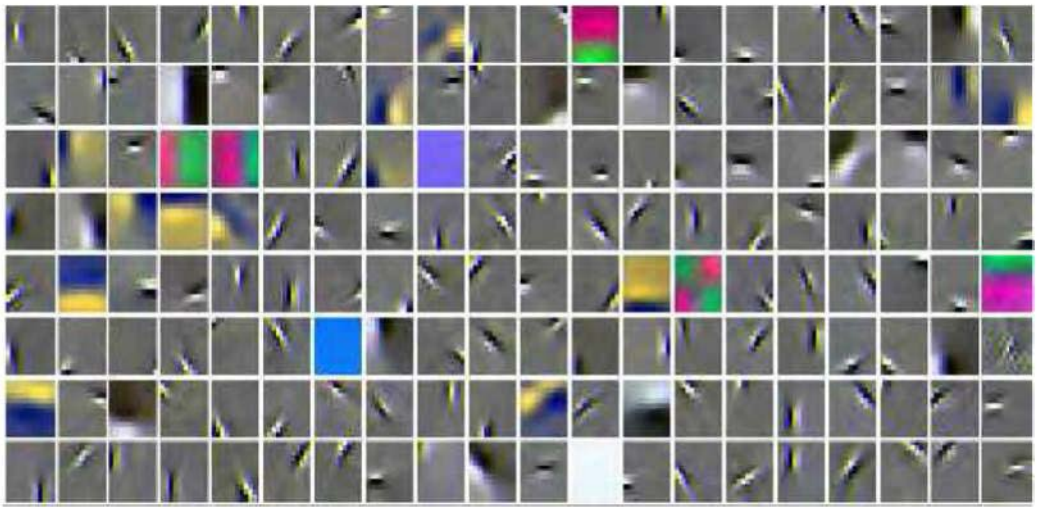


FIG. 18.11: *Base d'imagettes* 12 × 12 *de base obtenue par Patrick Hoyer et Aapo Hyvrïnen sur une base d'images en couleurs (signal RGB).*

Les résultats obtenus de cette manière sont intéressants, car ils sont en particulier confrontables avec les champs perceptifs des neurones tels qu'on peut essayer de les interpréter. D'un autre côté, la limitation au traitement d'imagettes oblitère peut-être des corrélations spatiales à plus grande échelle importantes pour le domaine. C'est pourquoi, en s'inspirant de l'analyse ICA (analyse en composantes indépendantes) mais en employant la technique récemment développée en fouille de données des FIS (*Frequent Item Sets*), une attaque directe peut être essayée. C'est la méthode FISICA (voir [Jou02, JCS+03, CSM04]).

7.2.4 Lien avec la technique de « projection pursuit »

La technique appelée *projection pursuit* (ou *poursuite de projections* en français) vise à découvrir des projections linéaires intéressantes (c'est-à-dire révélant une structure) de données multidimensionnelles sur des sous-espaces de petite dimension [FT74, Fri87]. Ces projections peuvent être utilisées pour permettre des visualisations révélatrices des données, ou bien comme étape pour de l'estimation de densité ou la régression. Le concept d'axes intéressants sur lesquels projeter les données dépend naturellement des applications, mais il est en général admis que les axes sur lesquels les données ont une distribution gaussienne sont les moins intéressants. L'algorithme de *poursuite de projection* cherche donc à identifier les axes sur lesquels les données se projettent en ayant une distribution la moins gaussienne possible. C'est exactement ce que l'on fait dans l'estimation en analyse en composantes indépendantes, et l'ACI peut être considérée comme un cas particulier de *poursuite de projection*. Encore une fois, la différence est notable avec l'objectif de l'analyse en composantes principales qui vise à conserver au maximum l'information au sens des moindres carrés (voir figure 18.12).

Copyright © 2010 Eyrolles.

La technique de *poursuite de projection* s'applique aussi lorsque le nombre de sources est inférieur au nombre de mesures. De plus, *poursuite de projection* ne fait pas d'hypothèse sur le modèle génératif des données, ce qui le rend moins sensible à des *a priori* erronés.

Pour une analyse de cette technique et des approches algorithmiques, nous reportons le lecteur en particulier à [JL95].

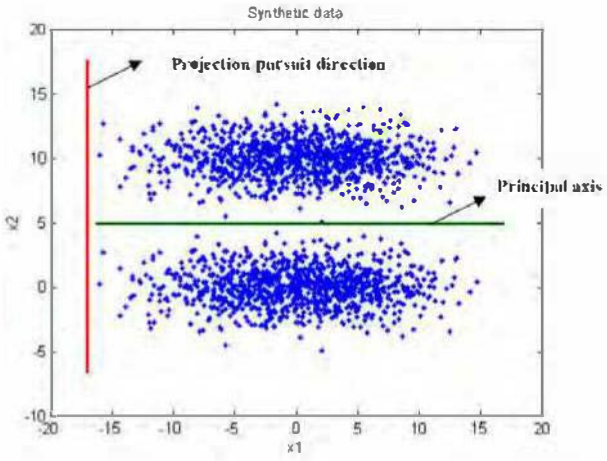


FIG. 18.12: ***Poursuite de projection.*** *Les données sont clairement séparées en deux groupes. L'analyse en composantes principales privilégie l'axe d'inertie principale, sur lequel la projection des deux groupes se confond. La poursuite de projection, en revanche, privilégie l'axe sur lequel la projection des données sépare au mieux les deux nuages de points.*

7.3 Analyse en composantes non négatives (NMF)

L'analyse en composantes non négatives (NMF) (*non-negative matrix factorization*, en anglais) impose que la décomposition des formes ou exemples s'opère par *addition* uniquement de primitives. En d'autres termes, dans l'expression des exemples, de dimension d, sous la forme :

$$\mathbf{x}_i = [\sum_{a=1}^{r} \phi_{1a} w_{ai}, \sum_{a=1}^{r} \phi_{2a} w_{ai}, \ldots, \sum_{a=1}^{r} \phi_{da} w_{ai}]^\top$$

dans laquelle les ϕ_{ja} sont des primitives de base, tous les coefficients w_{jk} doivent être ≥ 0.

Étant donnée une base d'exemples $\mathcal{S}$ exprimée sous la forme d'une matrice $[\mathcal{S}]$ de taille $d \times m$ (où d est la dimension de l'espace d'entrée $\mathcal{X}$ et m le nombre d'exemples d'apprentissage), il est possible de trouver une matrice Φ de formes de base ϕ_{jk} et une matrice de codage $\mathbf{W}$ telles que :

$$[\mathcal{S}]_{ij} \approx (\Phi \mathbf{W})_{ij} = \sum_{a=1}^{r} \phi_{ia} w_{aj} \tag{18.4}$$

Les matrices Φ et $\mathbf{W}$ sont respectivement de dimensions $d \times r$ et $r \times m$. On choisit généralement r de telle manière que $(d + m)r < dm$. Chaque colonne de la matrice Φ contient un vecteur de base, et chaque colonne de la matrice $\mathbf{W}$ contient les poids permettant l'approximation de la colonne correspondante de $[\mathcal{S}]$ en utilisant la base spécifiée par Φ.

Copyright © 2010 Eyrolles.

Dans l'analyse en composantes principales (ACP), chaque colonne de la matrice $\mathbf{\Phi}$ représente un vecteur propre et la matrice $\mathbf{W}$ représente les projections des exemples sur ces vecteurs propres. Contrairement à l'ACP, la factorisation non négative (NMF) contraint les poids à être ≥ 0. Cela conduit à des primitives de base qui représentent des formes locales dans l'espace $\mathcal{X}$, et dont la combinaison linéaire additive permet la description des exemples (voir la figure 18.13).

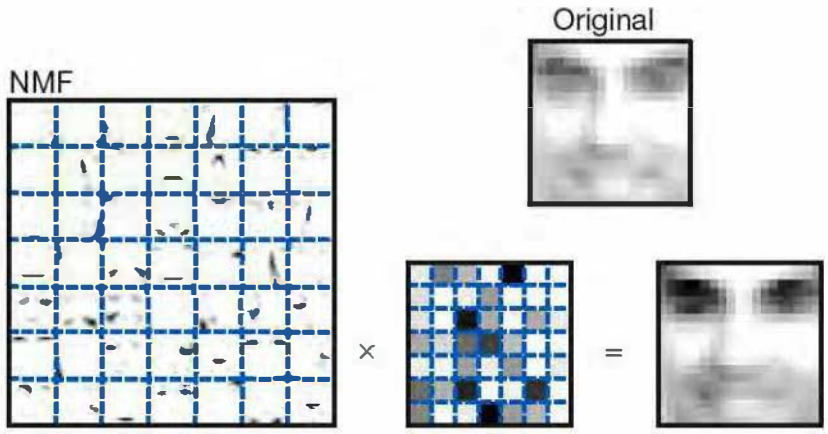


FIG. 18.13: *Un exemple de codage de visages en utilisant une factorisation non négative (tiré de [LS99])*

Afin de permettre l'estimation des matrices de factorisation $\mathbf{\Phi}$ et $\mathbf{W}$, il faut définir une fonction objectif. Une fonction possible et souvent utilisée est :

$$F = \sum_{i=1}^{d} \sum_{j=1}^{m} [[\mathcal{S}]_{ij} \log(\mathbf{\Phi}\,\mathbf{W})_{ij} - (\mathbf{\Phi}\,\mathbf{W})_{ij}]$$

Cette fonction objectif est liée à la vraisemblance d'engendrer les exemples de $\mathcal{S}$ à partir de la base de primitives $\mathbf{\Phi}$ et des codages spécifiés dans $\mathbf{W}$ (on suppose que chaque composante des exemples est obtenue par addition d'un bruit de Poisson au produit $(\mathbf{\Phi}\,\mathbf{W})_{ij}$). Une approche itérative permettant d'atteindre un maximum local de cette fonction est prescrite grâce aux règles suivantes :

$$\mathbf{\Phi}_{ia} \leftarrow \mathbf{\Phi}_{ia} \sum_{j} \frac{[\mathcal{S}]_{ij}}{(\mathbf{\Phi}\,\mathbf{W})_{ij}} \mathbf{W}_{aj}$$

$$\mathbf{\Phi}_{ia} \leftarrow \frac{\mathbf{\Phi}_{ia}}{\sum_k \mathbf{\Phi}_{ka}}$$

$$\mathbf{W}_{aj} \leftarrow \mathbf{W}_{aj} \sum_{i} \mathbf{\Phi}_{ia} \frac{[\mathcal{S}]_{ij}}{(\mathbf{\Phi}\,\mathbf{W})_{ij}}$$

Les matrices $\mathbf{\Phi}$ et $\mathbf{W}$ sont initialisées par des matrices positives aléatoires. La procédure converge (voir [LS99, LS00b] pour plus d'information) et produit généralement un codage parcimonieux (peu de coefficients h_{aj} non nuls). La factorisation non négative a été utilisée pour le codage de visages, mais aussi pour l'analyse de textes.

Copyright © 2010 Eyrolles.

7.4 Analyses en variables latentes

L'analyse en variables latentes est un outil statistique pour décrire et modéliser la structure sous-jacente de données multivariées. Elle est utilisée lorsque l'on suppose que les données peuvent être expliquées par un petit nombre de facteurs non observables. Le principe sous-jacent est simple. Si une variable latente gouverne un certain nombre de variables observées, alors le conditionnement sur cette variable explicative rend les variables observées statistiquement indépendantes. Le problème de l'analyse en variables latentes est donc d'identifier un ensemble de variables explicatives satisfaisant cette condition pour un ensemble de variables observées.

7.4.1 Analyse sémantique latente (LSA)

L'analyse sémantique latente (*Latent Semantic Analysis* ou LSA en anglais) a été motivée à l'origine par des applications de recherche d'information (*information retrieval*) dans des bases de documents textuels à partir de requêtes également exprimées à l'aide de mots. Un des problèmes rencontrés dans ces applications est que les mots de la requête peuvent être totalement différents de ceux présents dans les documents alors même que les concepts couverts sont proches. Par exemple, on aimerait qu'une requête portant sur la « découverte de connaissances » puisse ramener des documents dans lesquels il est question de « fouille de données ». L'idée est alors de passer par un espace intermédiaire dans lequel les termes exprimeraient davantage les distinctions sémantiques sous-jacentes aux documents (et aux requêtes). La projection des termes de la requête et de ceux des documents dans cet espace permettrait un meilleur rapprochement entre les concepts évoqués dans la requête et ceux évoqués par les documents.

Cette projection est calculée en décomposant la matrice N de termes-documents par décomposition en valeurs singulières (*Singular Value Decomposition* ou SVD en anglais) : $\mathbf{N} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top$, où $\mathbf{U}$ et $\mathbf{V}$ sont des matrices orthogonales $\mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$ et la matrice diagonale $\mathbf{\Sigma}$ contient les valeurs singulières de $\mathbf{N}$. L'approximation LSA de $\mathbf{N}$ est calculée en mettant à 0 toutes les valeurs singulières de $\mathbf{\Sigma}$ sauf celles de plus hautes valeurs. On obtient ainsi $\tilde{\mathbf{\Sigma}}$, qui est de rang K optimal au sens de la norme L_2 sur les matrices, et l'approximation $\tilde{\mathbf{N}} = \mathbf{U}\tilde{\mathbf{\Sigma}}\mathbf{V}^\top \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{N}$. Pour plus de détails et de références, nous reportons le lecteur à [LFL98].

7.4.2 Modèles graphiques

Les modèles graphiques, dont les réseaux bayésiens sont l'exemple le plus notable, font partie des méthodes à variables latentes. Nous renvoyons le lecteur au chapitre 11 pour plus de précisions.

7.5 Propriétés des bases de primitives de représentation

Lorsque l'on utilise une base de primitives pour représenter des données, il est intéressant de considérer un certain nombres de propriétés afin de déterminer si cette base est adéquate pour l'application en vue.

- **Complétude.** La représentation à l'aide des primitives permet-elle de reconstruire complètement l'entrée ? On parle alors de représentation complète. Il faut noter que la notion de représentation sur-complète (*over-complete basis* en anglais) est distincte puisqu'elle désigne une représentation dans laquelle le nombre de primitives de base excède (de beaucoup) la dimension de l'espace d'entrée.
- **Parcimonie.** Un codage est parcimonieux (*sparse coding* en anglais) si chaque forme d'entrée est codée par un petit sous-ensemble des primitives de représentation. Il est fréquent que la parcimonie d'un codage soit liée à son caractère sur-complet.

Copyright © 2010 Eyrolles.

- **Redondance**. Une représentation est redondante s'il est possible de reconstruire une forme d'entrée, même en l'absence de certaines parties du codage. Dans ce cas, le codage d'une forme d'entrée n'est pas unique. En général, la parcimonie d'un codage est associée à sa non-redondance.

Notes historiques et sources bibliographiques

Les sources philosophiques de la classification sont anciennes et nombreuses et la classification automatique est toujours l'objet de débats de fond. L'argument de D. Hume, relayé par J.-L. Borges, est en effet simple et confondant : « Il n'existe pas de classification de l'univers qui ne soit arbitraire et conjecturale. La raison en est fort simple : nous ne savons pas ce qu'est l'univers ». Sur un plan opérationnel, s'il ne s'agit que d'opposer les classes par leur nature géométrique et statistique (et non pas de découvrir leur nature cachée), les travaux remontent à Pearson (1894). Les mesures statistiques et les algorithmes ont été développés en particulier dans le cadre des sciences naturelles, mais aussi en reconnaissance de formes et en statistique appliquée. Les besoins actuels de la fouille de données ont fourni une nouvelle impulsion à ce domaine, en particulier par l'introduction des techniques de découverte des associations entre attributs binaires, l'étude des mélanges supervisés et non supervisés et l'invention du co apprentissage. L'ouvrage de Jain et Dubes [JD88] est une somme théorique et pratique constamment citée en classification automatique « classique ». Mais les ouvrages en français sont nombreux et reflètent la vigueur de ce domaine en France : [Cel89, Ler81, Sap90, Jam89, Leb95, BD00].

On peut dire que le concept de fouille de données est né avec le livre édité par Piatetsky-Shapiro et Frawley en 1991 [PSF91] qui recensait des articles portant sur ce sujet émergent. Celui édité par Fayyad, Piatetsky-Shapiro, Smyth et Uthurusamy en 1995 [FPSSU96], a définitivement installé le domaine dans les préoccupations des chercheurs. Depuis, de nombreux ouvrages ont paru sur la fouille de données. On citera ici des ouvrages généraux tels que [HMS01, HK06, CPSK07].

Les analyses en composantes ont fait l'objet d'un nombre grandissant de publications, de conférences et de tutoriels ces dernières années. On citera ici en particulier les deux excellents ouvrages [HKO01a, Sto04] principalement dédiés à l'analyse en composantes indépendantes, mais exposant des principes généraux importants et s'appliquant plus largement.

Finalement, on ne pourra que fortement conseiller la lecture d'un excellent ouvrage de vulgarisation portant sur le développement de l'analyse en ondelettes, mais qui, ce faisant re-situe de manière très claire les fondements de l'analyse [Hub95].

Résumé

- Il existe des méthodes pour séparer en classes un ensemble d'apprentissage non supervisé.
- Ces méthodes peuvent induire une hiérarchie de partitions sur l'ensemble ou une partition avec un nombre donné de classes.
- Ces méthodes s'appliquent naturellement aux données numériques, mais peuvent s'étendre aux données binaires ou symboliques.

Copyright © 2010 Eyrolles.

- D'autres techniques, à la base de la fouille de donnés, permettent d'extraire des associations logiques entre les attributs. Leur raffinement autorise la prise en compte d'intervalles.
- Les analyses en composantes cherchent à réduire la dimensionnalité des donnés en extrayant des combinaisons linéaires des attributs.

Copyright © 2010 Eyrolles.

Chapitre 19

L'apprentissage semi-supervisé

Les techniques d'apprentissage décrites jusqu'ici supposaient essentiellement que les données étaient soit étiquetées, cas de l'apprentissage supervisé, soit non étiquetées, cas de l'apprentissage non supervisé. Cependant un nombre croissant d'applications comportent à la fois des exemples étiquetés et des exemples non étiquetés. En est-on alors réduit à adopter l'une ou l'autre des deux familles de techniques d'apprentissage en se résignant à perdre les informations présentes dans le type de données que l'on ne pourra pas traiter ?

Les travaux récents essaient de répondre à ce défi par le développement de méthodes pouvant s'accommoder et tirer parti des deux types de données. C'est ce que l'on appelle l'apprentissage semi-supervisé. Il est encore loin d'être totalement compris et demande de prendre des précautions. Mais il est essentiel à la fois pour faire face aux nouvelles applications, mais aussi pour le développement conceptuel du domaine de l'apprentissage.

Copyright © 2010 Eyrolles.

Sommaire

1 Présentation 636
1.1 Définition et illustration 637
1.2 Quelques notions de base 639
1.3 Les principes des méthodes 639
2 Les modèles génératifs : apprentissage dans l'espace joint $\mathcal{X} \times \mathcal{Y}$ . . 640
2.1 L'utilisation de l'algorithme *EM* 640
2.2 Extensions 642
2.3 Quelques leçons 642
2.4 La méthode *cluster and label* 643
3 L'auto-apprentissage 643
4 Le co-apprentissage 644
4.1 L'apprentissage multi-vue et le co-apprentissage 644
4.2 Le cas de deux jeux indépendants d'attributs : le co-apprentissage . . . 645
5 L'utilisation d'hypothèses fondamentales sur les données 646
5.1 Éviter les zones de forte densité 647
5.1.1 Les SVM transductifs et semi-supervisés (S3VM) 647
5.2 Méthodes à base de graphes 649
5.2.1 Le principe 649
5.2.2 Critères transductifs régularisés sur graphe 650
5.2.3 Méthodes itératives 654
5.2.4 Optimisation directe 654
5.2.5 Extension à l'apprentissage inductif 655
6 Quelques directions pour l'apprentissage semi-supervisé 655
6.1 Apprentissage actif pour le cadre semi-supervisé 655
6.2 Régression semi-supervisée 656
6.3 Apprentissage semi-supervisé en ligne 656
6.4 Analyse PAC de l'apprentissage semi-supervisé 656
7 Conclusion et petite discussion 657

Copyright © 2010 Eyrolles.

1. Présentation

Supposons à nouveau dans ce chapitre que la tâche d'apprentissage vise à savoir étiqueter des exemples pour lesquels il n'y a pas d'étiquette connue, soit parce qu'il est difficile et coûteux d'obtenir ces étiquettes, soit parce qu'il s'agit d'exemples non encore observés. On peut envisager, et il existe, des situations dans lesquelles certains exemples ont été étiquetés, on parlera d'*exemples supervisés*, et d'autres exemples qui ne l'ont pas été, ce sont les *exemples non supervisés*. Ainsi, pour entraîner un trieur de spams, on dispose souvent d'une base d'exemples de courriels étiquetés, et d'une énorme masse de courriels non étiquetés.

Le cadre classique de l'induction supervisée ne prévoit que l'utilisation des exemples supervisés pour guider la recherche d'une règle de prédiction. Pourtant, *a priori*, il semble qu'à tout le moins la considération des exemples non supervisés ne peut pas nuire à l'induction, et qu'éventuellement, elle pourrait apporter des informations sur la tâche. C'est de cette intuition que procède l'étude de l'apprentissage semi-supervisé (*semi-supervised learning*). Plus précisément, bien souvent les régularités du problème n'impliquent pas seulement la *forme* de la fonction cible, mais aussi *la relation de cette fonction avec la distribution des données*. Or les exemples non supervisés apportent une information sur celle-ci. D'où leur utilité potentielle si l'on sait les utiliser.

De fait, quelques travaux pionniers ont proposé dans les années quatre-vingt-dix des méthodes attrayantes, tandis que certaines expériences montraient l'intérêt de ce type d'apprentissage. Cependant, d'autres expériences nous ont également averti que la prise en compte d'exemples non supervisés pouvaient parfois dégrader les performances par rapport à un apprentissage purement supervisé. Il est donc devenu clair qu'une analyse plus fondamentale était nécessaire afin de mieux comprendre les conditions permettant non seulement d'espérer, mais de garantir, des améliorations de performances.

Ce chapitre présente à la fois des algorithmes d'apprentissage semi-supervisé et les prolégomènes d'une analyse fondamentale. Il s'agit là d'un des champs de recherche les plus intéressants pour une évolution du domaine de l'apprentissage artificiel au-delà du cadre supervisé à partir de données i.i.d. classiques et pour le traitement de nouvelles applications. Il peut également éclairer l'apprentissage humain qui fonctionne fréquemment en mode semi-supervisé.

1.1 Définition et illustration

L'apprentissage semi-supervisé cherche à extraire une règle de décision ou de régression d'un ensemble d'apprentissage, avec une particularité : cet ensemble contient à la fois des objets supervisés, c'est-à-dire étiquetés, et d'autres qui ne le sont pas.

On pourrait penser que la présence supplémentaire d'objets non supervisés ne peut rien apporter à l'apprentissage supervisé ; en réalité, il est souvent possible de les utiliser pour améliorer la règle de classification apprise sur les exemples supervisés.

—— EXEMPLE **Apprentissage semi-supervisé sur $\mathbb{R}$** ——

Soient des points d'apprentissage placés sur la droite des réels, certains étant supervisés, avec une étiquette prise dans $\{0, 1\}$, d'autres étant non supervisés (voir la figure 19.1). En supposant que tous les points aient le même poids, le seuil de décision calculé pour les deux exemples supervisés passe au milieu (ligne en pointillés). En revanche, en prenant en compte les points non supervisés, et en supposant qu'ils soient issus de deux gaussiennes, le seuil est déplacé afin de ne pas passer au milieu de nuages de points (ligne continue). Intuitivement, la seconde règle de décision est meilleure que la première.

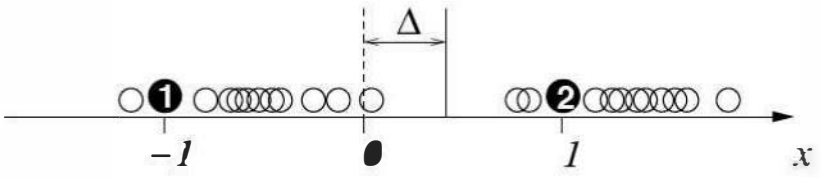


FIG. 19.1: *Les points (1) et (2) sont supervisés, les autres non. Δ est la distance entre le seuil de décision calculé à partir des exemples supervisés seuls (trait en pointillés) et le seuil obtenu en tenant compte des exemples non supervisés (trait continu).*

Copyright © 2010 Eyrolles.

L'écart entre les règles de décision, et l'amélioration éventuelle de la performance en généralisation, dépendent évidemment de la représentativité des points supervisés et des points non supervisés. Une question est d'ailleurs de déterminer à partir de quelle taille d'échantillon supervisé il devient inutile de tenir compte des exemples non supervisés.

EXEMPLE **Apprentissage semi-supervisé sur $\mathbb{R}^2$ (auto-apprentissage)**

Prenons un autre exemple simple. Les objets d'apprentissage sont ici des éléments de $\mathbb{R}^2$. Les exemples supervisés sont notés par l'étiquette de leur classe (+ ou −) et les objets non supervisés sont indiqués par des points.

La première opération consiste à apprendre une règle de classification sur les exemples supervisés. Ensuite, on classe les objets non supervisés selon cette règle. C'est ce qui est montré à la figure 19.2 (à droite). Les objets non supervisés sont indiqués après cette opération avec leur classe (⊕ ou ⊖). Dans un troisième temps, puisque tous les objets sont désormais supervisés, on peut recommencer l'apprentissage sur la totalité (figure 19.3). La règle de décision est modifiée et l'utilisation d'un ensemble de test montrerait qu'elle a été effectivement améliorée.

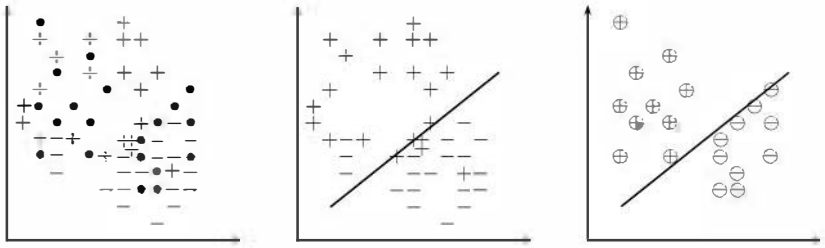


FIG. 19.2: *À gauche : l'ensemble d'apprentissage, composé d'exemples supervisés et d'objets non supervisés. Au centre : la règle de classification calculée sur les exemples supervisés. À droite : la classification des objets non supervisés selon cette règle.*

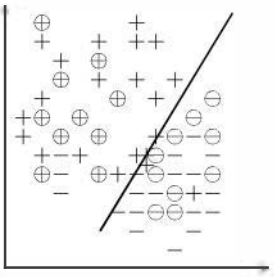


FIG. 19.3: *L'apprentissage d'une règle de décision sur les exemples supervisés et les objets non supervisés désormais classés par la règle précédente. La nouvelle règle de classification obtenue est meilleure.*

Cet exemple est exagérément simplifié à deux titres. Le premier est que l'utilisation des objets non supervisés est trop naïve. La plupart du temps, dans ce type de méthode, on progresse par étapes plus prudentes en étiquetant des petites fractions successives des données non supervisées (cette méthode est appelée *auto-apprentissage* ; nous y reviendrons dans la suite de ce chapitre.) Le second est que la méthode semble à juste titre peu fiable : il

Copyright © 2010 Eyrolles.

est facile d'imaginer des échantillons qui dégraderaient le résultat plutôt que de l'améliorer, quelques précautions que l'on prenne. Cependant, cet exemple montre que l'apprentissage semi-supervisé peut parfois être opératoire. Peut-on donner d'une manière générale les conditions de l'efficacité de ce type d'apprentissage ?

En première approche, il semble nécessaire que les objets non supervisés apportent une information complémentaire mais non contradictoire aux exemples supervisés. Pour être moins vague, il faudra poser des hypothèses sur la répartition des exemples et sur celle des objets : des propriétés de convergence pourront alors en être induites. Il est donc nécessaire de donner une modélisation statistique de ce type d'apprentissage pour espérer en formuler des propriétés. Nous revenons sur cette question au paragraphe 1.3.

1.2 Quelques notions de base

La définition du paragraphe précédent peut se formaliser comme suit. On dispose d'un ensemble d'apprentissage composé de m objets, parmi lesquels l sont supervisés. On a donc, en les ordonnant :

$$\mathcal{S} = \mathcal{S}_s \cup \mathcal{S}_{ns} = \{(\mathbf{x}_i, u_i)\}_{i=1,l} \cup \{\mathbf{x}_i\}_{i=l+1,m}$$

Les éléments supervisés seront, selon le vocabulaire établi, appelés des *exemples* et les autres seront appelés des *objets* (on précisera *non supervisés* si nécessaire). Nous nous intéressons au cas de la classification, c'est-à-dire que les éléments u_i appartiennent à un ensemble fini $\mathcal{C}$.

L'apprentissage semi-supervisé peut avoir deux buts un peu différents.

1. Soit chercher à classer uniquement les objets de l'ensemble $\{\mathbf{x}_i\}_{i=l+1,m}$. On parle alors d'apprentissage *transductif*.
2. Soit trouver une fonction capable de classer tout objet de l'espace de représentation. On est alors dans le cas classique de l'apprentissage *inductif*.

L'apprentissage transductif pourrait être traité comme une problématique en soi, mais il est désormais traditionnel de le classer dans l'apprentissage semi-supervisé. L'apprentissage semi-supervisé inductif, qui correspond à l'exemple du paragraphe précédent, vise *a priori* plus de généralité et semble donc plus difficile. On peut déjà remarquer qu'après avoir réalisé un apprentissage transductif, on se retrouve apparemment dans une situation d'apprentissage inductif classique. Nous verrons sur un exemple, au paragraphe 5.2.5 comment on peut aller un peu plus loin que cette idée simple.

1.3 Les principes des méthodes

Plaçons-nous d'abord dans le cas de l'apprentissage inductif. Pour que les objets non supervisés puissent aider les exemples à apprendre une règle de classification, il faut, dans un modèle statistique, que leur distribution puisse apporter de l'information sur les classes, bien que leur classe soit inconnue. Autrement dit, il faut que leur classification non supervisée (voir chapitre 18) produise des classes corrélées statistiquement avec celles des exemples.

Comme on traite un mélange entre données supervisées et non supervisées, on peut envisager deux approches, selon que l'on aborde le problème comme une induction compliquée par les données non supervisées, ou comme une classification non supervisée facilitée par des contraintes exprimées sous la forme d'exemples. La première approche inclut l'auto-apprentissage, mais aussi d'autres algorithmes, dits *génératifs*, comme l'identification au maximum de vraisemblance des paramètres des classes (modélisées en général comme des processus gaussiens). La seconde inclut

Copyright © 2010 Eyrolles.

une sorte de symétrique de l'auto-apprentissage (algorithme *cluster and label*), et des méthodes basées sur la théorie des graphes.

Quant aux méthodes transductives, elles sont fondées sur la théorie des graphes ou sur les techniques de maximisation des marges qui sont des extensions des SVM que nous avons étudiées au chapitre 14.

Les paragraphes suivants décrivent brièvement les méthodes en question.

2. Les modèles génératifs : apprentissage dans l'espace joint $\mathcal{X} \times \mathcal{Y}$

Rappelons que les modèles génératifs supposent que l'information acquise permet d'engendrer de nouvelles données dont la distribution ne peut pas être distinguée des données réelles. Le modèle cherché est de la forme $\mathsf{p}(\mathbf{x}, y) = \mathsf{p}(y)\,\mathsf{p}(\mathbf{x}|y)$, où $\mathsf{p}(\mathbf{x}|y)$ est un mélange de distributions identifiable, par exemple un mélange de gaussiennes. À partir d'échantillons de données non supervisées de grande taille, dépendant de la dimension de l'espace d'entrée, les composantes du mélange peuvent être identifiées. Il suffit alors d'un exemple étiqueté par composante pour déterminer complètement le modèle génératif. En cela, les modèles génératifs constituent l'une des plus anciennes méthode d'apprentissage semi-supervisé.

De plus, la technique de base pour identifier les composantes de mélange est l'algorithme *EM* qui fonctionne à partir d'exemples supervisés et non supervisés (voir l'annexe 7).

2.1 L'utilisation de l'algorithme *EM*

D'une manière générale, comment utiliser la partie supervisée $\mathcal{S}_{sup}$ des exemples pour étiqueter $\mathcal{S}_{nonsup}$, la partie non supervisée ? Nous allons décrire une procédure fondée sur l'algorithme *EM*, qui a été instanciée sur un certain nombre d'applications diverses. Rappellons que l'algorithme *EM*, dont l'annexe 7 donne une description, a déjà été présenté comme utile pour l'apprentissage des HMM (chapitre 12) et pour l'estimation des paramètres des mélanges de distribution gaussiennes, au paragraphe 3.1 de ce même chapitre.

Pour simplifier, plaçons-nous dans le cas de deux classes, ce qui n'est en aucune façon limitatif, et donnons une version informelle du déroulement de cette méthode.

Il faut d'abord disposer d'un algorithme d'apprentissage fondé sur l'estimation de paramètres d'un modèle statistique. Typiquement, cet algorithme suppose par exemple que la distribution *a priori* de chacune des classes est gaussienne : les paramètres à estimer sont alors la moyenne et la matrice de covariance de la distribution de chaque classe. Ce problème a été traité dans le cas supervisé au chapitre 15. Par conséquent, une fois l'apprentissage par estimation des paramètres réalisé sur les données étiquetées $\mathcal{S}_{sup}$, on peut calculer pour chacune des deux classes la probabilité estimée d'avoir engendré chaque exemple de $\mathcal{S}_{sup}$ et de $\mathcal{S}_{nonsup}$.

Il est maintenant possible d'étiqueter chaque exemple par la règle *MAP* du maximum à posteriori de la classification bayésienne. Chaque exemple se verra donc attribuer l'étiquette de la classe qui a la plus grande probabilité de l'avoir engendré.

On peut désormais apprendre deux nouveaux modèles de classes sur l'ensemble des données, en utilisant l'étiquetage que l'on vient de réaliser. Ces deux nouveaux modèles permettront un nouvel étiquetage en deux classes de l'ensemble des données par le principe *MAP*, ce qui permettra de calculer deux modèles modifiés, et ainsi de suite. Cette boucle sera poursuivie jusqu'à ce qu'aucun exemple ne change plus d'étiquette.

Plus largement, on peut considérer la classe de chaque exemple comme une variable cachée et

Copyright © 2010 Eyrolles.

utiliser l'algorithme EM pour l'estimer, avec pour résultat l'étiquetage de chaque exemple par une classe. Ceci conduit à l'algorithme général 39.

Cette méthode générale a été particularisée pour l'apprentissage dans des problèmes variés où l'étiquetage est coûteux, comme en traitement de la langue naturelle écrite ou orale. On a ainsi réalisé l'apprentissage de modèles statistiques de séquences pour la la syntaxe et le vocabulaire en reconnaissance de la parole, celui de modèles gaussiens naïfs pour la classification des sites web, etc. Dans le cas où les modèles statistiques utilisés sont les modèles bayésiens naïfs, l'algorithme se réécrit sous une forme qui permet d'y ajouter des sophistications [BM98].

Algorithme 39 : Algorithme supervisé non supervisé

Apprendre les modèles initiaux des classes sur $\mathcal{S}_{sup}$
$\mathcal{S} \leftarrow \mathcal{S}_{sup} \cup \mathcal{S}_{nonsup}$
répéter
  Étape E (estimation) : étiqueter $\mathcal{S}$ par le principe MAP selon les modèles courants des classes
  Étape M (maximisation) : estimer les modèles des classes à partir de cet étiquetage
jusqu'à *ce que la convergence des modèles des classes soit réalisée*

Nous allons traiter ces méthodes en présentant d'abord un exemple simple. Cependant, à partir des principes sous-jacents, on peut construire des systèmes d'apprentissage aussi sophistiqués que les modèles statistiques le permettent. Nous ferons donc ensuite quelques commentaires sur ces extensions.

EXEMPLE **Application de l'algorithme EM en apprentissage semi-supervisé**

Nous prenons un problème à deux classes ($C = 2$), chacune d'elles étant supposée être modélisée correctement par une distribution gaussienne. Nous savons depuis le chapitre 15 comment estimer au maximum de vraisemblance les paramètres de ces distributions quand tous les exemples sont étiquetés. La question est d'étendre cette technique à un ensemble d'apprentissage semi-supervisé.

Nous disposons de l'ensemble d'apprentissage $\mathcal{S} = \{(\mathbf{x}_i, u_i)\}_{i=1,l} \cup \{\mathbf{x}_i\}_{i=l+1,m}$ et nous prenons le modèle suivant : nous supposons que la première classe est engendrée par un processus gaussien $\mathcal{N}(\mu_1, \Sigma_1)$ et la seconde par $\mathcal{N}(\mu_2, \Sigma_2)$. Nous cherchons à estimer l'ensemble des paramètres $\theta = (\mu_1, \mu_2, \Sigma_1, \Sigma_2)$. Rappelons que μ_1 désigne le vecteur moyenne dans $\mathbb{R}^d$ de la première classe et que Σ_1 est sa matrice $(d \times d)$ de covariance. Ces deux éléments suffisent à définir la première classe, sous l'hypothèse gaussienne. De même pour la seconde classe.

L'expression $P(\mathbf{x}|u, \theta)$ dénote donc la probabilité que, dans le modèle de paramètres θ, l'objet $\mathbf{x}$ ait pour supervision la classe de valeur u. On sait estimer au maximum de vraisemblance θ de façon à ce que $P(\mathbf{x}|u, \theta)$ soit la meilleure (c'est un résultat que nous avons donné au chapitre 15). Mais ce résultat n'est applicable que si tous les exemples sont étiquetés. Notons maintenant la probabilité *a priori* de la classe d'étiquette u par π_u et introduisons le nouvel ensemble de paramètres $\pi = \{\pi_i\}_{i=1,2}$ (rappelons que nous traitons un problème à deux classes).

Pour n'importe quelles valeurs $\widehat{\theta}$ et $\widehat{\pi}$ de θ et de π, on peut écrire, par la règle de Bayes :

$$P(u|\mathbf{x}, \widehat{\theta}, \widehat{\pi}) \frac{\widehat{\pi}_u P(\mathbf{x}|u, \widehat{\theta})}{\sum_{\omega=1}^{2} \widehat{\pi}_\omega P(\mathbf{x}|\omega, \widehat{\theta})} \tag{19.1}$$

Par conséquent, si l'on possède une estimation $\widehat{\pi}$ et $\widehat{\theta}$ des inconnues de notre problème, on est capable de fournir une règle de classification.

À défaut de pouvoir agir directement, on peut considérer les étiquettes absentes dans l'ensemble $\mathcal{S}_{n.s}$ comme des *données manquantes* et utiliser l'algorithme EM (voir l'annexe 7)

Copyright © 2010 Eyrolles.

pour les estimer itérativement. La technique sera comparable à l'utilisation de cet algorithme dans la classification non supervisée, mais elle fera usage de l'information supplémentaire qui est donnée par les étiquettes de l'ensemble $\mathcal{S}_s$.

Informellement, l'algorithme EM fonctionne comme suit. D'abord on initialise π et θ par un calcul direct sur $\mathcal{S}_s$. Puis on démarre le cycle suivant : premièrement (étape E), on utilise la formule 19.1 pour affecter des étiquettes à tous les objets de $\mathcal{S}_{ns}$. Deuxièmement (étape M), on met à jour les estimations de π et de θ sur l'ensemble de toutes les données, qui sont maintenant étiquetées. On revient à l'étape E et on boucle jusqu'à convergence. Rappelons que les deux classes sont supposées gaussiennes. Les paramètres à apprendre sont $\theta = (\mu_1, \mu_2, \Sigma_1, \Sigma_2)$, les caractéristiques des distributions, et $\pi = \{\pi_i\}_{i=1,2}$, la probabilité *a priori* des classes.

Algorithme 40 : Algorithme EM pour l'apprentissage semi-supervisé

Données : un ensemble d'apprentissage semi-supervisé $\mathcal{S} = \{(\mathbf{x}_i, u_i)\}_{i=1,l} \cup \{\mathbf{x}_i\}_{i=l+1,m}$
Résultat : une estimation de π et de θ
début
  Initialiser π et θ par estimation sur $\mathcal{S}_s$
  tant que *la convergence n'est pas réalisée* **faire**
    Étape E : Affecter des étiquettes à tous les objets de $\mathcal{S}_{ns}$ par la formule 19.1
    Étape M : Mettre à jour les estimations de π et de θ sur l'ensemble de toutes les données (qui sont maintenant toutes étiquetées).
  fin tant que
fin

2.2 Extensions

Ce qui a été présenté ci-dessus peut directement être étendu à un nombre de classes supérieur à deux et à des gaussiennes multidimensionnelles. D'autre part, l'algorithme EM permet d'estimer des données cachées sous des hypothèses très variées. Il n'y a donc pas de raison de se limiter à des classes gaussiennes. Plusieurs extensions sont donc envisageables.

On peut par exemple supposer que chaque classe est elle-même un mélange de distributions gaussiennes. Dans ce cas, le nombre de variables cachées augmente et leur nature change, mais la méthode de résolution est toujours applicable. On peut aussi supposer que la distribution des classes suit une autre loi, plus simple comme un modèle bayésien naïf (voir le chapitre 15, paragraphe 1.1) ou différente comme un processus dépendant du temps. C'est ainsi qu'on peut réaliser l'apprentissage semi-supervisé de modèles de Markov cachés, par extension de l'apprentissage supervisé présenté au chapitre 12. Il est à noter que la version de base de cet apprentissage utilise déjà l'algorithme EM.

On peut aussi observer que si EM est une manière efficace de maximiser la valeur $P(\mathbf{x}|u, \theta)$ quand il y a des variables cachées, ce n'est pas la seule. On peut alternativement utiliser des méthodes d'optimisation comme les algorithmes génétiques, par exemple.

2.3 Quelques leçons

La méthode EM fonctionne sur la base d'hypothèses *a priori* sur les modèles des données, par exemple, qu'elles soient issues de gaussiennes. Sans ces hypothèses, la méthode ne peut pas fonctionner. Mais si ces hypothèses ne sont pas en accord avec la vraie distribution, inconnue,

Copyright © 2010 Eyrolles.

des données, les résultats peuvent alors être mauvais, voire complètement erronés. Il faut donc être très attentif à l'adéquation des hypothèses *a priori* par rapport au monde.

Par ailleurs, l'estimation par EM peut mener à des optima locaux, et, encore une fois, à des erreurs si le choix du modèle n'est pas le bon.

Finalement, la convergence est plus aléatoire si le nombre de variables cachées est trop important, autrement dit quand le nombre d'objets non étiquetés est trop grand relativement à celui des exemples. Dans ce cas, il est parfois utile de sous-pondérer les premiers.

2.4 La méthode *cluster and label*

Une méthode quelque peu différente des approches génératives consiste à utiliser une étape de classification non supervisé (*clustering*) sur l'ensemble des données, supervisées et non supervisées, avant d'étiqueter les nuages de points ainsi identifiés à l'aide des exemples supervisés.

Ici encore, ces approches fonctionnent bien si le biais des méthodes de clustering employées épouse bien la vraie distribution des données.

D'un certain côté, cette méthode est symétrique de l'auto-apprentissage (voir section 3) : elle part des objets non étiquetés de $\mathcal{S}_{ns}$ avant d'utiliser l'étiquetage des exemples de $\mathcal{S}_s$. La technique de base est décrite dans l'algorithme 41.

Algorithme 41 : Algorithme *cluster and label* pour l'apprentissage semi-supervisé

Données : un ensemble d'apprentissage semi-supervisé
début
  $\mathcal{S} = \{(\mathbf{x}_i, u_i)\}_{i=1,l} \cup \{\mathbf{x}_i\}_{i=l+1,m}$
  Étape *cluster*. Utiliser un algorithme de classification non supervisé pour classer $\mathcal{S}$ en ignorant les étiquettes de $\mathcal{S}_s$.
  Étape *label*. Donner à chaque classe une étiquette : celle qui est majoritaire parmi des points de $\mathcal{S}_s$ qui se trouvent dans cette classe.
fin

L'un des avantages de la méthode *cluster and label* est qu'elle ne requiert pas d'hypothèses fortes sur la distribution des données. Il faut cependant ne pas oublier que tout algorithme de clustering s'appuie sur des présupposés sur les données, même s'ils ne sont pas aussi forts que dans les méthodes génératives. Par ailleurs, un biais *a priori* plus faible se paye nécessairement soit par la nécessité d'un plus grand échantillon d'apprentissage, soit par un apprentissage moins performant.

Nous avons décrit ici le plus simple des algorithmes de *cluster and label*. Il est clair qu'il est possible de faire appel à des algorithmes de clustering plus complexes, par exemple par catégorisation hiérarchique, permettant de rendre compte de structures de données plus riches, tout en conservant le principe du mariage de l'approche non supervisée et d'un étiquetage grâce aux données supervisées.

3. L'auto-apprentissage

La méthode de principe (algorithme 42) est très simple : par une méthode d'apprentissage quelconque, on induit une règle de classification à partir des l exemples de l'ensemble $\{(\mathbf{x}_i, u_i)\}_{i=1,l}$. Ensuite, on prend le point suivant (ou au hasard) dans l'ensemble des objets $\{\mathbf{x}_i\}_{i=l+1,m}$ et on le

Copyright © 2010 Eyrolles.

classe par la règle courante. On apprend alors une nouvelle règle de classification sur l'ensemble des exemples augmenté d'une unité. Tout ceci est effectué $(m - l)$ fois, jusqu'à disparition des objets non supervisés.

Algorithme 42 : Algorithme d'auto-apprentissage

Données : un ensemble d'apprentissage semi-supervisé $\mathcal{S} = \{(\mathbf{x}_i, u_i)\}_{i=1,l} \cup \{\mathbf{x}_i\}_{i=l+1,m}$
Résultat : une règle de classification h_m
début
  $j \leftarrow l$
  pour $j < m$ **faire**
    Apprendre une règle de classification h_j sur $\{(\mathbf{x}_i, u_i)\}_{i=1,j}$
    Classer $\mathbf{x}_{j+1}$ par h_j
    $j \leftarrow j + 1$
  fin pour
  Apprendre la règle de classification finale h_m sur les m exemples désormais tous supervisés.
fin

Un certain nombre de variantes sont possibles. En particulier, on peut classer systématiquement tous les objets restants et ajouter à l'ensemble supervisé celui dont la classification est considérée comme la plus fiable. On peut aussi ajouter les objets par groupes et non pas un par un.

L'avantage de cette méthode est qu'elle permet d'inclure[1] avec simplicité n'importe quelle technique d'apprentissage supervisé pour opérer un l'apprentissage semi-supervisé. Elle fonctionne bien dans nombre de cas et elle est employée par exemple pour construire des trieurs de spams dont la qualité augmente avec l'expérience. En revanche, sa convergence n'est assurée ni théoriquement, ni pratiquement. Elle dépend beaucoup des premiers étiquetages : c'est la raison de la première variante donnée ci-dessus.

4. Le co-apprentissage

4.1 L'apprentissage multi-vue et le co-apprentissage

Jusqu'ici, comme il est usuel en apprentissage artificiel, nous avons généralement supposé que chaque exemple ou donnée était représenté comme un unique vecteur dans un espace de descripteurs ou comme un unique graphe lorsque l'information était de type relationnel. Cependant, dans de nombreux problèmes réels, **un même exemple peut être décrit de différentes manières**, par exemple par différents vecteurs ou par différents graphes ou par un mélange de ces deux représentations. Une question est alors de savoir comment apprendre à partir de ces représentations « multi-vues » et d'exploiter leur richesse.

Un exemple typique est celui de la catégorisation de pages web. En effet, une même page peut être décrite soit par les hyperliens qui la lient à d'autres pages, soit par les mots qui la composent. La première description conduit à considérer un graphe dirigé dans lequel chaque sommet représente une page web et chaque arc orienté un hyperlien. La seconde description se prête à considérer les pages comme des vecteurs dans un espace euclidien où chaque élément

[1] Le terme anglais pour cette propriété *wrapper method*, évoque un emballage.

Copyright © 2010 Eyrolles.

d'un vecteur correspond par exemple au nombre d'occurrences d'un mot dans la page (approche « sac de mots »).

Une manière de combiner ces deux représentations pourrait alors être de pondérer les arcs par une mesure de similarité entre les vecteurs de mots représentant les pages liées. Cependant, cette approche met trop l'accent sur les liens puisque la similarité entre pages non directement liées n'est pas prise en compte.

Une autre approche consiste à définir des fonctions noyau (voir le chapitre 14) pour chacune des représentations, puis à les combiner de manière convexe pour obtenir une fonction noyau générale [JCST01, ZPD06].

Nous présentons dans ce paragraphe deux techniques récentes qui permettent, sous certaines hypothèses, de superviser totalement un ensemble d'apprentissage composé d'une partie supervisée et d'une partie non supervisée. C'est particulièrement utile dans le cas où l'étiquetage par expert, qui coûte cher, doit être fait sur des grandes quantités de données, comme pour le traitement de la langue écrite ou orale.

4.2 Le cas de deux jeux indépendants d'attributs : le co-apprentissage

Une première technique peut s'appliquer quand on dispose sur les données d'assez d'attributs pour les partager en deux sous-ensembles statistiquement indépendants. Blum et Mitchell [BM98] proposent l'exemple de la classification des pages Web des sites universitaires en classes telles que *pages personnelles des étudiants*, *page d'accueil d'un département*, *page de description d'un cours*, etc. Les deux jeux de descripteurs sont les suivants :

- Un vecteur entier de dimension égale à la taille du dictionnaire utilisé (le nombre de mots différents possibles), dont la valeur d'une coordonnée est le nombre d'apparitions dans la page Web. Cette description en « sac de mots » est très utilisée en linguistique automatique et donne en général des résultats plutôt bons[2] si on considère qu'elle ne tient pas compte de l'ordre des mots.
- Un autre vecteur sac de mots qui ignore le texte de la page et décrit ses références à d'autres pages (ses hyperliens).

Ces deux descriptions sont indépendantes et pourtant les mots qui composent les hyperliens d'une page donnée sont d'une certaine façon une bonne description de cette page[3].

Notons d'une manière générale ces deux ensembles d'attributs indépendants $\mathcal{X}_1$ et $\mathcal{X}_2$ et notons $\mathcal{S}_{sup}$ la partie supervisée de l'ensemble d'apprentissage et $\mathcal{S}_{nonsup}$ la partie non supervisée. La technique dite de *co-apprentissage* (*co-training*) se déroule comme indiqué dans l'algorithme 43. Elle y est décrite pour l'apprentissage d'un concept (deux classes), mais son extension est immédiate.

Le cœur de la méthode repose sur le choix des exemples en nombre $p_1 + n_1 + p_2 + n_2$ que l'on rajoute à chaque étape à $\mathcal{S}_{sup}$. Il faut pour les choisir classer par A et B tout l'ensemble $\mathcal{S}_{nonsup}$. On retient alors les p_1 exemples pour lesquels A est « le plus sûr » qu'ils sont positifs. De même pour les n_1 négatifs : ce sont ceux pour lesquels la décision de A est la plus sûre. C'est également ainsi que p_2 et n_2 autres exemples sont sélectionnés par B. Un algorithme de classification peut en effet avoir une mesure naturelle de « sûreté », par exemple une probabilité, dans le cas d'une classification bayésienne, une distance à l'hyperplan appris dans le cas d'une décision linéaire, etc.

[2] Pour la tâche d'apprentissage du concept « page de description d'un cours », un classificateur bayésien naïf a été entraîné sur douze pages Web avec une description en sac de mots. Il a une performance de l'ordre de 87 % de bonne classification sur un ensemble de test de deux cent cinquante pages.

[3] Les résultats expérimentaux montrent que la classification obtenue sur ce jeu d'attributs est presque équivalente à celle obtenue sur la même tâche avec l'autre jeu d'attributs.

Copyright © 2010 Eyrolles.

Algorithme 43 : Algorithme de co-apprentissage

répéter

Apprendre un classificateur A sur $\mathcal{S}_{sup}$ Apprendre un classificateur B sur $\mathcal{S}_{sup}$ Classer $\mathcal{S}_{nonsup}$ par A ;
Classer $\mathcal{S}_{nonsup}$ par B ;
Choisir les p_1 exemples positifs et n_1 exemples négatifs de $\mathcal{S}_{nonsup}$ les plus sûrs pour A
Choisir par B p_2 exemples positifs et n_2 exemples négatifs de $\mathcal{S}_{nonsup}$ les plus sûrs pour B Ajouter ces $p_1 + n_1 + p_2 + n_2$ nouvellement classés à $\mathcal{S}_{sup}$

jusqu'à *la convergence est réalisée*;

La justification empirique de cette méthode peut s'énoncer ainsi : si le classificateur A trouve dans les données non supervisées un exemple très proche d'un des ses exemples d'apprentissage, il a de bonnes chances de le classer correctement. Mais cela ne signifie en rien que le classificateur B l'aurait classé correctement, puisque les jeux d'attributs $\mathcal{X}_1$ et $\mathcal{X}_2$ sont indépendants : être proches dans le premier espace n'implique pas que l'on soit proches dans le second. Par conséquent, A a ajouté à $\mathcal{S}_{sup}$ un exemple supervisé qui va apporter de l'information à B.

Les expériences montrent que cette technique est efficace[4]. Son analyse théorique dans le cadre PAC prouve sa convergence sous certaines conditions. Cette méthode possède aussi des liens statistiques avec les méthodes de rééchantillonnage (voir le chapitre 17). En pratique, il est important de noter que l'indépendance effective des deux jeux d'attributs est déterminante pour son succès.

5. L'utilisation d'hypothèses fondamentales sur les données

Pour que les exemples non étiquetés puissent fournir une information complémentaire à l'information apportée par les exemples supervisés, il faut qu'ils se traduisent par des contraintes supplémentaires sur le choix des fonctions de décision candidates. Pour cela, il faut, d'une part, que des hypothèses *a priori* soient faites sur la distribution des données non supervisées, et, d'autre part, que ces hypothèses aient un lien avec la vraie distribution des données.

L'une ou l'autre de deux hypothèses fondamentales est généralement utilisée pour fonder les méthodes proposées :

- Dans la première, on suppose que la distribution $\mathbf{p}_{\mathcal{X}}$ des données correspond à un mélange de distributions convexes (par exemple un mélange de gaussiennes) et que chaque composante de ce mélange correspond à une étiquette. Cela implique que *les changements d'étiquettes se font* dans les régions entre les composantes, et donc, essentiellement *dans des zones de moins forte densité*.
- Dans la seconde, on suppose qu'à l'intérieur de l'espace des entrées $\mathcal{X}$, les données sont définies sur un sous-espace (*manifold*) de dimension inférieure à celle de $\mathcal{X}$. On représente ce sous-espace à l'aide d'un graphe mettant en jeu les exemples connus, supervisés et non supervisés, et on suppose que *la fonction d'étiquetage de ces points vérifie une certaine continuité*.

[4] Le même concept appris sur un ensemble de deux cent cinquante pages étiquetées par co-apprentissage (dont douze étaient étiquetés au début) classe un ensemble de test indépendant en moyenne avec 95 % de succès. Les attributs sont maintenant l'union des ensembles $\mathcal{X}_1$ et $\mathcal{X}_2$ [BM98].

Copyright © 2010 Eyrolles.

Ces hypothèses, quoique intuitivement attrayantes et raisonnables, ne sont pas, pour le moment, étayées par une analyse formelle les justifiant comme suffisantes ou nécessaires pour garantir le succès de l'apprentissage semi-supervisé. Il semble clair que si $\mathbf{p}(\mathbf{x})$ et $\mathbf{p}(y|\mathbf{x})$ ne partagent pas de paramètres, l'apprentissage semi-supervisé ne peut pas améliorer l'apprentissage supervisé classique, mais ce n'est là qu'une condition minimale.

Par ailleurs, de nombreuses méthodes d'apprentissage cherchent directement à optimiser une fonction de décision liée à la probabilité conditionnelle $\mathbf{p}(y|\mathbf{x})$ sans utiliser $\mathbf{p}(\mathbf{x})$ qui est la composante sur laquelle les données non supervisées peuvent apporter de l'information. Comment alors ces méthodes discriminatives peuvent elles profiter des données non supervisées ?

La plupart d'entre elles s'appuient sur l'hypothèse que la fonction de décision (la frontière) ne passe pas dans les régions de haute densité dans $\mathcal{X}$. C'est le cas, par exemple, de la méthode des séparateurs à vastes marges transductif (TSVM ou parfois S^3VM pour *Semi-Supervised SVM*).

5.1 Éviter les zones de forte densité

5.1.1 Les SVM transductifs et semi-supervisés (S3VM)

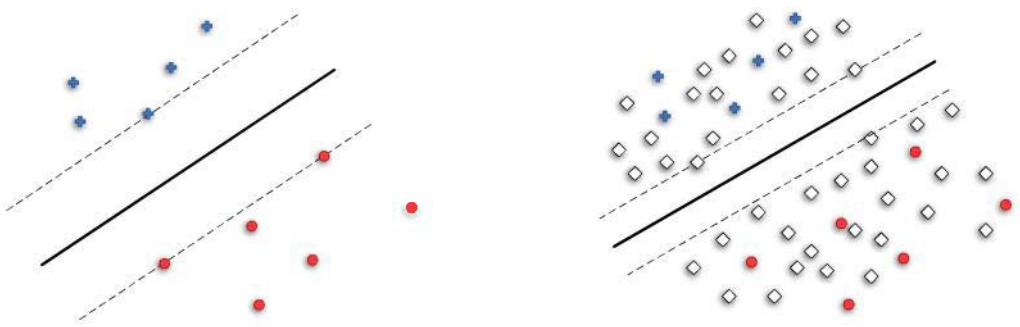


FIG. 19.4: *SVM calculé avec les exemples supervisés (à gauche) ; SVM calculé en tenant compte des exemples non supervisés, en supposant que les nuages de points des deux classes sont séparés (à droite).*

Nous nous intéressons ici au cas de la classification binaire, c'est-à-dire à deux classes. Nous avons vu au chapitre 14 que la méthode des séparatrices à vaste marge (SVM) permet de traiter de tels problèmes de manière bien fondée et adaptée à une large classe de problèmes. Supposons maintenant qu'au lieu de ne disposer que d'un échantillon d'exemples supervisés, $\mathcal{S}_s$, nous disposions également de la connaissance d'exemples non supervisés, $\mathcal{S}_{ns}$, ceux sur lesquels nous attendons d'être interrogés. De quelle manière peut-on modifier la méthode des SVM pour tirer parti de la connaissance de $\mathcal{S}_{ns}$?

La figure 19.4 fournit une intuition de ce que l'on pourrait souhaiter. À la vue en effet des exemples non supervisés, il semble que la séparatrice optimale calculée à partir des exemples supervisés gagnerait à être modifiée pour mieux séparer les deux « nuages » de points. Cette stratégie repose naturellement sur un présupposé qui est que les classes + et − sont séparables dans l'espace des entrées $\mathcal{X}$ ou dans l'espace de redescription $\Phi(\mathcal{X})$ (voir chapitre 14). Nous reviendrons sur ce point.

Formellement, rappelons que la recherche de la séparatrice à marge maximale peut s'exprimer par le problème d'optimisation portant sur le vecteur directeur de l'hyperplan séparateur $\mathbf{w}$ de telle manière qu'il permette de classer au mieux les exemples supervisés, tout en autorisant que

Copyright © 2010 Eyrolles.

quelques exemples ne soient pas bien placés, ce que permet l'introduction de variables ressort ξ :

$$\begin{cases} \min\limits_{\mathbf{w},b,\xi} & \sum\limits_{i=1}^{l} \xi_i + \lambda ||\mathbf{w}||^2 \\ \text{tel que} & u_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \qquad i = 1 \ldots, l \end{cases} \tag{19.2}$$

Le paramètre λ permet de contrôler l'importance relative de la recherche de la marge maximale (terme $\mathbf{w}||^2$) et de minimiser la tolérance aux exemples mal placés (terme $\sum_{i=1}^{l} \xi_i$).

Le problème d'optimisation ainsi défini est quadratique, avec un optimum global et est appelé forme primale.

Il est possible de montrer que ce problème est équivalent à :

$$\min_{\mathbf{w},b} \sum_{i=1}^{l} \max(1 - u_i(\mathbf{w}^\top \mathbf{x}_i + b), 0) + \lambda ||\mathbf{w}||^2 \tag{19.3}$$

lorsque l'on utilise la fonction de coût en coude (*hinge loss*) (voir chapitre 14) :

$$\ell\big(\mathbf{x}, u, h(\mathbf{x})\big) = \max\big(1 - u(\mathbf{w}^\top \mathbf{x} + b), 0\big) \tag{19.4}$$

Comment alors incorporer une contrainte sur le problème d'optimisation correspondant au présupposé que les deux classes soient bien séparées?

Il nous faut une fonction de coût s'appliquant aux exemples non supervisés et favorisant la séparation des nuages de points. Une traduction de ces contraintes consiste à prendre pour étiquette putative $\hat{y}$ associée à un exemple non supervisé $\mathbf{x}$ la prédiction qui serait faite par un séparateur linéaire, à savoir : $\hat{y} = \text{signe}(h(\mathbf{x}))$. On peut alors prendre pour fonction de perte :

$$\begin{aligned} \ell(\mathbf{x}, \hat{y}, h(\mathbf{x})) &= \max\big(1 - \hat{y}(\mathbf{w}^\top \mathbf{x} + b), 0\big) \\ &= \max\big(1 - \text{signe}(\mathbf{w}^\top \mathbf{x} + b)(\mathbf{w}^\top \mathbf{x} + b), 0\big) \\ &= \max(1 - |\mathbf{w}^\top \mathbf{x} + b|, 0) \end{aligned} \tag{19.5}$$

où nous utilisons le fait que : $\text{signe}(z) \cdot z = |z|$.

Cette fonction de coût a une forme de « chapeau » (voir figure 19.5). Elle ne pénalise évidemment pas les exemples pour être du mauvais côté de la séparatrice puisque ce ne peut être le cas pour des exemples étiquetés putativement par cette même fonction. En revanche, elle pénalise les exemples proches de la séparatrice.

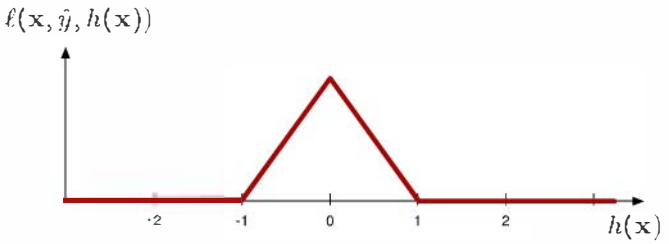


FIG. 19.5: *La fonction de coût en chapeau.*

Copyright © 2010 Eyrolles.

On obtient ainsi le **risque empirique régularisé adapté à l'apprentissage semi-supervisé :**

$$R_{\text{Emp}}(h) = \underset{\mathbf{w},b}{\text{ArgMin}}\left\{ \underbrace{\sum_{i=1}^{l} \max\left(1 - u_i(\mathbf{w}^\top \mathbf{x}_i + b), 0\right) + \lambda_1 ||\mathbf{w}||^2}_{\text{Risque empirique supervisé}} + \underbrace{\lambda_2 \sum_{j=l+1}^{m} \max\left(1 - |\mathbf{w}^\top \mathbf{x}_j + b|, 0\right)}_{\text{Risque empirique non supervisé}} \right\} \tag{19.6}$$

où h est déterminé par les coefficients $\mathbf{w}$ et b du séparateur linéaire.

En pratique, il arrive parfois que ce critère conduise au choix d'une séparatrice mettant une grande partie des exemples non supervisés, sinon tous, dans la même classe. Un moyen de corriger ce déséquilibre, généralement inadapté, est d'imposer que la proportion des classes soient approximativement la même que celle des exemples supervisés. On complète alors le critère inductif 19.6 par la contrainte :

$$\frac{1}{m-l} \sum_{j=l+1}^{m} h(\mathbf{x}) = \frac{1}{l} \sum_{i=1}^{l} u_i \tag{19.7}$$

Il faut noter que l'utilisation de la fonction de perte en chapeau conduit à un critère inductif non convexe (puisqu'il comporte une addition de critères non convexes) et de ce fait difficile à optimiser. Une partie des travaux sur les S3VMs porte donc sur la recherche de méthodes efficaces pour trouver de bons optima locaux de ce critère. D'autres fonctions de perte ont également été proposées, mais elles conduisent toutes à des critères non convexes (voir [Zhu08], pour plus de détails).

Par ailleurs, les S3VMs peuvent conduire à de mauvais résultats si le présupposé selon lequel les classes sont bien séparées est en défaut.

Finalement, nous sommes restés ici dans le cadre de SVM opérant directement dans l'espace de description des exemples $\mathcal{X}$. Mais l'approche se transpose sans problème à la recherche d'un séparatrice linéaire dans l'espace de redescription $\Phi(\mathcal{X})$.

5.2 Méthodes à base de graphes

5.2.1 Le principe

Dans ces méthodes, l'idée est de représenter les données sous forme d'un graphe (V, E), dont l'ensemble V des sommets sont les objets d'apprentissage et les arcs (les éléments de E) représentent la similarité entre deux objets. Le graphe est donc pondéré par une matrice $\mathbf{W}$ dont chaque élément $\mathbf{W}_{ij}$ mesure la similarité entre l'objet i et l'objet j dans $\mathcal{S}$, l'échantillon d'apprentissage supervisé et non supervisé.

Les valeurs $\mathbf{W}_{ij}$ peuvent être simplement binaires : une valeur nulle efface l'arc entre les sommets, alors que la présence d'un arc indique par exemple que $\mathbf{x}_i$ est parmi les k plus proches voisins de $\mathbf{x}_j$. Un tel graphe est orienté (i et j ne jouent pas des rôles symétriques). Mais ces valeurs peuvent aussi être continues, par exemple calculées (étant donné une certaine norme et un paramètre σ) de la manière suivante :

$$\mathbf{W}_{ij} = e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}} \tag{19.8}$$

Dans ce dernier cas, le graphe est complètement connecté. La construction du graphe est en soi une étape importante qui doit bénéficier de toute connaissance préalable du domaine.

Dans cette représentation, les étiquettes des objets de $\mathcal{S}_s$ n'apparaissent pas encore. On les place comme une information supplémentaire sur les sommets correspondants du graphe G.

Copyright © 2010 Eyrolles.

La technique de base est d'utiliser la structure du graphe pour étendre cet étiquetage à tous les sommets, en partant des points supervisés pour lesquels l'étiquette est donnée. La plupart des algorithmes cherchent à optimiser un compromis entre la *fidélité aux données d'apprentissage* (les points initialement étiquetés devraient conserver leur étiquette au cours de la propagation) et une notion de *continuité* dans l'étiquetage (intuitivement, un sommet non étiqueté proche de sommets étiquetés de la classe ω_i et éloigné de tous les autres recevra l'étiquette ω_i et pourra la propager à son tour) (voir figure 19.6). C'est l'essence des méthodes inductives régularisées, ici traduites par des critères spécifiques dans le cadre de données situées sur un graphe, comme nous le verrons ci-dessous.

Évidemment, certains conflits apparaîtront dans ce processus. Savoir les régler est l'essentiel des problèmes posés par ces méthodes. Nous verrons comment traiter ce problème par des algorithmes itératifs, puis par des méthodes d'optimisation globale.

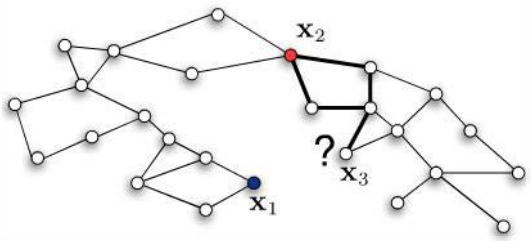


FIG. 19.6: *Les points* x_1 *et* x_2 *sont supervisés avec des étiquettes différentes, les autres non. L'étiquette de* x_3 *est plus proche de celle de* x_2 *que de celle de* x_1 *puisque* x_2 *est plus proche dans le graphe que* x_3.

Il est à noter qu'il s'agit ici *a priori* d'apprentissage transductif, puisque seuls les objets d'apprentissage non étiquetés peuvent être classés par un tel processus. Cependant, certaines techniques peuvent être appliquées au cas de l'apprentissage semi-supervisé, c'est-à-dire pour calculer l'étiquette de points non observés. Nous reviendrons sur ce point au paragraphe 5.2.5.

5.2.2 Critères transductifs régularisés sur graphe

Plusieurs critères transductifs ont été proposés pour exprimer le compromis entre la *fidélité aux données supervisées* et le biais vers *une continuité des étiquetages sur le graphe*. Nous en présentons succinctement trois.

Le critère MinCut

Le critère MinCut suppose que les étiquettes des données, et donc des nœuds du graphe, sont prises dans $\{+1, -1\}$. Dans ce cadre, les exemples positifs sont considérés comme des « sources » et les exemples négatifs comme des « puits ». L'objectif est de bloquer le flot entre sources et puits en coupant un ensemble d'arcs de poids total minimal. Cela conduit à une partition du graphe, chaque partie étant étiquetée par $+1$ ou -1 selon l'étiquette du seul nœud étiqueté qu'elle contient (voir la figure 19.7).

On cherche donc une fonction $h : \mathcal{X} \to \{-1, +1\}$, telle que les étiquettes des exemples supervisés soient respectées et telle que les étiquettes des autres exemples soient décidées sur la base d'une partition MinCut du graphe.

L'un des défauts du critère MinCut est qu'il conduit facilement à des minima de même valeur. Par exemple, dans la figure 19.6, s'il y avait un seul lien entre les nœuds x_2 et x_4, il y aurait

Copyright © 2010 Eyrolles.

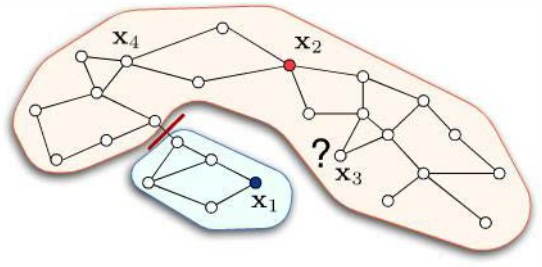


FIG. 19.7: *Les points x_1 et x_2 sont supervisés avec des étiquettes différentes, les autres non. En supposant que les arcs sont tous affectés d'un poids unitaire, la partition du graphe se fait sur l'arc coupé. De ce fait, l'étiquette de x_3 sera celle de x_2.*

deux partitions apparemment également bonnes du graphe (celle de la figure et une autre avec séparation entre x_2 et x_4).

Critère avec fonction harmonique

Le critère utilisant une fonction harmonique permet en grande partie d'éliminer le problème des minima globaux multiples en assouplissant le problème d'optimisation. On utilise en effet une fonction harmonique h, donc à valeur dans $\mathbb{R}$, au lieu de la fonction indicatrice à valeur dans $\{-1, 1\}$.

La fonction h est définie de telle manière que les étiquettes des exemples supervisés soient respectées, tandis que les étiquettes des exemples non supervisés résultent d'une moyenne harmonique, pondérée par les poids des liens, des valeurs des étiquettes des points voisins.

$$\begin{cases} h(\mathbf{x}_i) = y_i & i = 1, \dots l \\ h(\mathbf{x}_i) = \dfrac{\sum_{k=l+1}^{m} w_{jk} h(\mathbf{x}_k)}{\sum_{k=l+1}^{m} w_{jk}} & j = l+1, \dots m \end{cases} \tag{19.9}$$

Le problème d'optimisation est alors :

$$\begin{cases} \text{ArgMin}_{h : h(\mathbf{x}) \in \mathbb{R}} & \sum_{i,j=1}^{m} w_{ij} \left(h(\mathbf{x}_i) - h(\mathbf{x}_j)\right)^2 \\ \text{tel que :} & h(\mathbf{x}_i) = y_i \quad \text{pour } i = 1 \dots l \end{cases} \tag{19.10}$$

ce qui exprime que les étiquettes supervisées doivent être respectées (deuxième ligne) et que l'étiquette d'un point doit être proche des étiquettes des points proches au sens des poids w_{ij} (première ligne).

Ce problème d'optimisation a une solution analytique qui est unique et est un optimum global. En revanche, les étiquettes produites sont des réels. Il faut donc les transformer en étiquettes valides ($\in \{-1, +1\}$), ce que l'on réalise souvent grâce à une fonction seuil (par exemple si $h(\mathbf{x}) > 0$ alors prédire $y = 1$, et si $h(\mathbf{x}) < 0$ alors prédire $y = -1$).

Remarque (Analogie avec une marche aléatoire)

La fonction harmonique peut aussi être interprétée comme une marche aléatoire, ce qui est à la source de certaines fonctions noyau définies sur les graphes (voir chapitre 14). En effet, imaginons une particule

Copyright © 2010 Eyrolles.

placée sur le point $\mathbf{x}_i$ qui est non supervisé. Soit la marche aléatoire spécifiée par la probabilité qu'une particule en $\mathbf{x}_i$ aille en $\mathbf{x}_j$ proportionnelle à w_{ij} :

$$\mathbf{P}(j|i) = \frac{w_{ij}}{\sum_k w_{ij}}$$

La marche aléatoire continue jusqu'à ce que la particule parvienne en un point supervisé $\mathbf{x}_k$. La valeur de la fonction harmonique en $\mathbf{x}_i$, $h(\mathbf{x}_i)$, est alors égale à la probabilité que la particule atteigne un point supervisé d'étiquette positive.

Une procédure itérative (dite de propagation des étiquettes) permet de calculer la fonction harmonique en chaque point. Pour cela, on initialise les points $\{\mathbf{x}_i\}_{i=1,\ldots,l}$ avec leur valeur supervisé, et les autres avec une valeur tirée aléatoirement dans $[0, 1]$. On réévalue alors itérativement les valeurs des points non supervisées par la moyenne pondérée de leurs voisins :

$$h(\mathbf{x}_i) \leftarrow \frac{\sum_{j=1}^{m} w_{ij} h(\mathbf{x}_j)}{\sum_{j=1}^{m} w_{ij}}$$

Il est prouvé que cette procédure itérative produit des valeurs convergeant vers la fonction harmonique, quelles que soient les valeurs choisies initialement pour les points non supervisés.

Plus formellement, la méthode standard de *propagation des étiquettes* est donnée dans l'algorithme 44. Elle fonctionne pour deux classes, notées + et −. Les valeurs $\widehat{U}_s = \{\widehat{u}_i\}_{i=1,m}$ affectées aux éléments d'apprentissage prennent des valeurs continues au cours du calcul et c'est le signe de la valeur finale (après convergence) qui détermine la classe attribuée à $\mathbf{x}_i$. On remarque que l'étiquette des exemples de $\mathcal{S}_s$ est délibérement préservée.

Algorithme 44 : Algorithme de propagation des étiquettes.

début
  Calculer la matrice $\mathbf{W}$ d'après l'équation 19.8
  Calculer la matrice diagonale $\mathbf{D}$ par $\mathbf{D}_{ii} \leftarrow \sum_j \mathbf{W}_{ij}$
  Initialiser les étiquettes : $\widehat{U}^0(u_1, \ldots, u_l, \text{rand}[0,1], \ldots, \text{rand}[0,1])$
  $t \leftarrow 0$
  tant que *la convergence n'est pas réalisée* **faire**
    $\widehat{U}^{t+1} \leftarrow \mathbf{D}^{-1}\mathbf{W}\widehat{U}^t$
    $\widehat{U}_s^{t+1} \leftarrow U_s$
    $t \leftarrow t + 1$
  fin tant que
  Affecter à $\mathbf{x}_i$ le signe de $\widehat{u}_i^t$
fin

Dans la variante dite d'*extension des étiquettes* (*label spreading*), en revanche, on ne préserve pas nécessairement l'étiquette des points supervisés. La méthode est détaillée par l'algorithme 45. Elle fait appel à une matrice classique en théorie des graphes, le *laplacien* normalisé (voir le chapitre 14). Sa convergence est assurée ; elle se produit expérimentalement en un nombre d'itérations entre m^2 et m^3.

Critère avec régularisation sur variété

Les méthodes MinCut et par fonction harmonique décrites ci-dessus présentent deux inconvénients. Elles sont transductives, c'est-à-dire qu'elles ne permettent pas d'extrapoler à des points

Copyright © 2010 Eyrolles.

Algorithme 45 : Algorithme d'extension des étiquettes

début
Calculer la matrice $\mathbf{W}$ d'après l'équation 19.8
Calculer la matrice diagonale $\mathbf{D}$ par $\mathbf{D}_{ii} \leftarrow \sum_j \mathbf{W}_{ij}{}^*$
Calculer le laplacien $\mathcal{L} \leftarrow \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$
Initialiser les étiquettes : $\widehat{U}^0(u_1, \ldots, u_l, 0, \ldots, 0)$
$t \leftarrow 0$
Choisir la valeur du paramètre α entre 0 et 1.
tant que *la convergence n'est pas réalisée* **faire**
  $\widehat{U}^{t+1} \leftarrow \alpha \mathcal{L}\widehat{U}^t + (1-\alpha)\widehat{U}^0$
  $t \leftarrow t+1$
fin tant que
Affecter à $\mathbf{x}_i$ le signe de $\widehat{u}_i^t$
fin

non vus dans $\mathcal{X}$. Par ailleurs, elles ne tolèrent pas facilement des données supervisées incorrectement étiquetées, ce qui arrive pourtant fréquemment. L'approche par régularisation sur variété (*manifold regularization*) répond à ces deux limitations.

Il s'agit d'une approche inductive produisant une fonction h définie sur $\mathcal{X}$ ($h : \mathcal{X} \rightarrow \mathbb{R}$), qui, de plus, doit satisfaire un critère régularisé prenant en compte à la fois une certaine continuité sur le graphe des points d'apprentissage, mais aussi une condition de régularité sur $\mathcal{X}$. La régularité sur $\mathcal{X}$ est fréquemment traduite par une condition sur la norme ℓ_2 de la fonction h : $||h||^2 = \int_{x \in \mathcal{X}} h(\mathbf{x})^2 d\mathbf{x}$.

Le critère inductif régularisé devient alors :

$$\hat{h}_{\mathcal{S}} = \underset{h:\mathcal{X}\rightarrow\mathbb{R}}{\text{ArgMin}} \sum_{i=1}^{l} \big(y_i - h(\mathbf{x}_i)\big)^2 + c_1 ||h||^2 + c_2\, \mathbf{h}^\top L\mathbf{h} \tag{19.11}$$

où $\mathbf{h}$ est le vecteur des étiquettes $\big(h(\mathbf{x}_1), \ldots, h(\mathbf{x}_l)\big)$, L est la matrice du laplacien de graphe (voir section 5.3.7 du chapitre 14), et les coefficients $c_1, c_2 \geq 0$ contrôlent l'importance des termes de régularisation.

Le théorème de représentation (voir chapitre 14) assure qu'il existe une représentation de dimension finie (au plus de taille m) de l'hypothèse optimale h. Toute fonction h définie sur le graphe peut être décomposée en :

$$h = \sum_{i=1}^{m} a_i \phi_i \tag{19.12}$$

où les a_i sont les coefficients à valeur réelle, et les ϕ_i sont les vecteurs propres de la matrice $\mathbf{L} = \mathbf{D} - \mathbf{W}$. Le terme de régularisation de h sur le graphe s'écrit alors :

$$\mathbf{h}^\top \mathbf{L}\mathbf{h} = \sum_{i=1}^{m} a_i^2 \lambda_i \tag{19.13}$$

où les λ_i sont les valeurs propres de $\mathbf{L}$.

Le terme $\mathbf{h}^\top \mathbf{L}\mathbf{h}$ est petit si les valeurs a_i ou λ_i sont petites pour tous les i. Or puisque les fonctions ϕ_i sont d'autant plus régulières sur le graphe que leur valeur propre λ_i associée est proche de 0, intuitivement, cela signifie que $\mathbf{h}$ doit prendre ses grands coefficients a_i sur les ϕ_i

Copyright © 2010 Eyrolles.

associées aux petites valeurs λ_i, c'est-à-dire être décomposable en termes de fonctions régulières, ce qui est exactement ce que l'on attend (voir les figures 19.8 et 19.9 pour des exemples de graphes et de spectre associé).

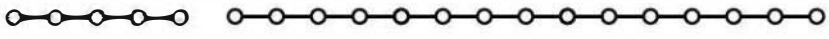


FIG. 19.8: *Un graphe correspondant à deux chaînes déconnectées.*

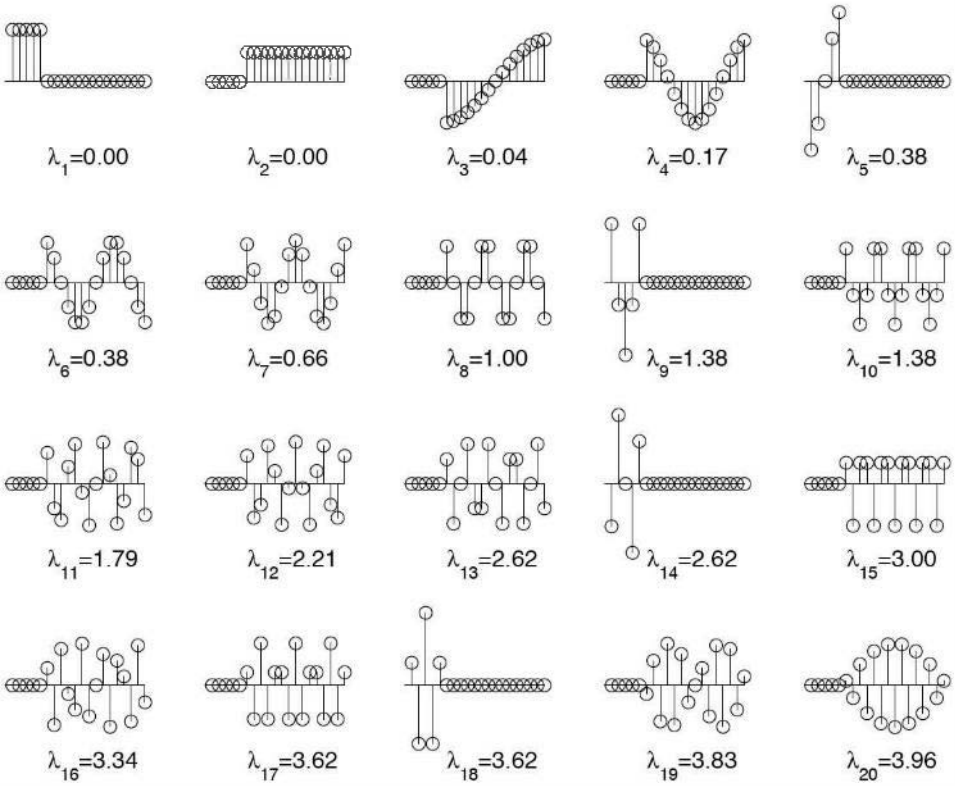


FIG. 19.9: *Le spectre du graphe de la figure 19.8 (laplacien de graphe non normalisé) avec les valeurs propres et les fonctions de base associées. Plus la valeurs propre a une valeur élevées, plus la fonction de base associée est irrégulière.*

5.2.3 Méthodes itératives

Dans ce paragraphe, l'ensemble d'apprentissage s'écrira

$$\mathcal{S} = \mathcal{S}_s \cup \mathcal{S}_{ns}\{(\mathbf{x}_i, \widehat{\mathbf{u}}_i)\}_{i=1,l} \cup \{(\mathbf{x}_i, \widehat{u}_i)\}_{i=l+1,n}$$

pour indiquer que des étiquettes $\widehat{u}_i$ sont attribuées à tous les éléments de $\mathcal{S}$ au cours du processus. En principe, on doit avoir $\widehat{U_s} = \{\widehat{u}_i\}_{i=1,l}$ exactement égal à $U_s = \{u_i\}_{i=1,l}$, mais il peut être utile de relâcher (prudemment) cette contrainte, par exemple si les classes se recouvrent.

5.2.4 Optimisation directe

L'affectation d'étiquettes aux sommets de $\mathcal{S}_{ns}$ doit respecter la géométrie du graphe et l'étiquetage des exemples de $\mathcal{S}_s$. La conservation de cette seconde contrainte peut s'exprimer de la

Copyright © 2010 Eyrolles.

manière suivante. Soit $\widehat{U} = \widehat{U}_s \cup \widehat{U}_{ns}$ un étiquetage calculé sur tous les sommets. La quantité suivante doit être minimale, nulle si possible :

$$||\widehat{U}_s - U_s|| = \sum_{i=1}^{l} (\widehat{u}_i - u_i)^2$$

La consistance avec la géométrie du graphe et une hypothèse naturelle sur une certaine régularité des données impliquent qu'il faut que des points proches aient une forte probabilité d'avoir la même étiquette. Ceci peut se traduire en écrivant qu'il faut dans la mesure du possible minimiser une autre quantité :

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \mathbf{W}_{ij} (\widehat{u}_i - \widehat{u}_j)^2$$

Cette dernière expression peut se développer et se recalculer comme $\widehat{U}^\top (\mathbf{D} - \mathbf{W}) \widehat{U}$. Les éléments $\mathbf{D}$ et $\mathbf{W}$ ont été définis dans l'algorithme 44 ; l'expression $(\mathbf{D} - \mathbf{W})$ est le laplacien *non normalisé* du graphe. Le problème revient donc à trouver le jeu d'étiquettes $(\widehat{u}_1, \ldots, \widehat{u}_m)$ qui minimise le critère suivant (une somme pondérée de ces deux termes) :

$$C = \alpha ||\widehat{U}_s - U_s|| + (1 - \alpha) \widehat{U}^\top (\mathbf{D} - \mathbf{W}) \widehat{U} \tag{19.14}$$

Différentes méthodes sont possibles pour cela, dans les détails desquelles nous ne rentrons pas. Les commentaires bibliographiques de ce chapitre orienteront le lecteur vers des références adéquates.

5.2.5 Extension à l'apprentissage inductif

Nous sommes partis d'un ensemble d'apprentissage $\mathcal{S} = \mathcal{S}_s \cup \mathcal{S}_{ns} = \{(\mathbf{x}_i, u_i)\}_{i=1,l} \cup \{\mathbf{x}_i\}_{i=l+1,m}$ qui, après apprentissage transductif, est devenu : $\{(\mathbf{x}_i, \widehat{u}_i)\}_{i=1,l} \cup \{(\mathbf{x}_i, \widehat{u}_i)\}_{i=l+1,m}$ avec des étiquettes $\widehat{u}_i$ attribuées à tous les éléments de $\mathcal{S}$ au cours du processus. Rappelons que certaines étiquettes des exemples de $\mathcal{S}_s$ peuvent avoir changé.

Soit un nouvel objet $\mathbf{x}$ dont nous ignorons l'étiquette. Pour la déterminer, il est évidemment possible d'utiliser n'importe quelle méthode d'apprentissage de règle de classification sur $\mathcal{S}_s$: par exemple calculer un SVM ou utiliser la règle des k plus proches voisins. Une autre solution, en principe meilleure, est d'enlever les étiquettes des éléments de $\mathcal{S}_{ns}$, de lui ajouter $\mathbf{x}$ et de recommencer l'apprentissage semi-supervisé.

Le coût de la seconde méthode peut être trop élevé pour étiqueter seulement un objet. Une manière intermédiaire de procéder est de réécrire le critère précédent en lui ajoutant l'étiquette inconnue u de $\mathbf{x}$, mais de garder les valeurs $\widehat{\mathbf{u}}_1$ à $\widehat{u}_m$ qui le minimisaient sans cette étiquette. On n'a plus qu'à le minimiser par rapport à u, ce qui se fait en le dérivant par rapport à cette variable. L'expression analytique est facile à établir et le calcul exact est immédiat.

6. Quelques directions pour l'apprentissage semi-supervisé

6.1 Apprentissage actif pour le cadre semi-supervisé

Une partie des travaux en apprentissage actif concerne l'apprentissage à partir d'un ensemble fourni *a priori* de données non étiquetées. Le problème de l'apprenant est alors de trier ces données pour limiter le nombre de requêtes à l'oracle afin de converger le plus rapidement possible sur une bonne hypothèse.

Copyright © 2010 Eyrolles.

Il existe donc une parenté avec l'apprentissage semi-supervisé. Cependant, les travaux rapprochant ces deux problèmes sont encore très rares. L'article de [Zhu08] fournit quelques références.

6.2 Régression semi-supervisée

On peut noter que les méthodes à base de graphe estiment d'abord une fonction continue sur les points non supervisés avant d'utiliser un seuillage pour déterminer la classe des exemples. Ces méthodes peuvent donc naturellement servir pour une tâche de régression. Nous n'entrons pas davantage dans les détails ici, mais reportons le lecteur à [Zhu08] pour des références aux quelques travaux existants dans ce domaine.

6.3 Apprentissage semi-supervisée en ligne

L'apprentissage en ligne est capable de prendre en compte les données au fur et à mesure de leur arrivée pour affiner ou pour adapter la fonction de décision calculée. La plupart des travaux portent soit sur l'apprentissage en ligne supervisé, soit sur l'apprentissage en ligne non supervisé. Pourtant, il est fréquent que les données arrivent en flux et que, parmi celles-ci, la plupart soient non étiquetées. C'est le cas par exemple pour un robot autonome. Lorsqu'il est impossible de stocker toutes ces données, comment utiliser au mieux le flux de données ?

Très peu de travaux traitent de ce problème à ce jour[5]. Une idée utilisée dans [GLZ08] est d'adapter les approches batch de l'apprentissage semi-supervisé qui utilisent un critère inductif convexe (ce qui exclut les S3VM comme nous l'avons vu). En effet, dans ce cas, il est possible de recourir à des méthodes d'optimisation par programmation convexe en ligne.

On peut dire qu'il s'agit là d'un domaine complètement ouvert pour des approches innovantes.

6.4 Analyse PAC de l'apprentissage semi-supervisé

Alors que la théorie de l'induction supervisée classique peut maintenant être considérée comme mûre, la théorie de l'apprentissage semi-supervisée en est encore à un stade exploratoire. Les travaux dans ce domaine sont récents et encore peu nombreux. Parmi ceux-ci, l'analyse de Balcan et Blum [BB09], dessine un cadre à la fois général et fidèle à l'intuition qui a fondé l'apprentissage semi-supervisé.

L'une des limites, et des forces, de l'analyse classique est qu'elle ne considère pas la distribution des exemples $\mathsf{p}_{\mathcal{X}}$. Il s'agit d'une analyse contre toute distribution. Mais ce sont justement des hypothèses émises *a priori* sur la forme de cette distribution qui autorisent l'apprentissage semi-supervisé.

L'idée de la nouvelle théorie est d'ajouter un critère traduisant l'espoir que la distribution des exemples est telle que celle qui est attendue. Par exemple, que les frontières entre les classes passent bien par les zones de basse densité. Les auteurs définissent le critère de *compatibilité* qui estime la compatibilité de l'hypothèse candidate avec la distribution des données non supervisées. En supposant que les données respectent cette compatibilité, il est alors possible de réduire la taille de l'espace des hypothèses à considérer. Deux volets sont à considérer. D'une part, le degré auquel la fonction cible obéit effectivement aux hypothèses *a priori* sur la distribution. D'autre part, le degré auquel la distribution des données permet d'éliminer des hypothèses candidates. Ainsi, par exemple, si les données ne forment pas de clusters (sorte de distribution uniforme sur un sous-espace de $\mathcal{X}$), alors toutes les fonctions candidates sont également (in-)compatibles avec les données.

[5] Octobre 2009.

Copyright © 2010 Eyrolles.

La traduction de cette mesure de compatibilité se fait par l'intermédiaire d'une fonction de coût prenant en compte des données non étiquetées. Une des possibilités étant la fonction de coût en chapeau utilisée pour les S3VM (section 5.1.1).

Le modèle théorique obtenu permet de calculer le nombre d'exemples supervisés et non supervisés pour apprendre, presque sûrement, avec un certain niveau de performance. On peut ainsi montrer que, dans des circonstances favorables, le nombre d'exemples supervisés requis peut être très petit.

7. Conclusion et petite discussion

L'apprentissage semi-supervisé consiste à savoir quand et comment utiliser les données non supervisées pour mieux apprendre la dépendance entre un espace d'entrée $\mathcal{X}$ et un espace de sortie $\mathcal{Y}$. Les techniques développées visent souvent d'abord à étiqueter les données non supervisées avant, éventuellement, de généraliser à l'espace $\mathcal{X}$ tout entier. C'est en raison de ce possible découplage en deux étapes que l'apprentissage semi-supervisé a été souvent comparé à l'*apprentissage transductif* qui, lui, ne vise qu'à prédire les étiquettes d'exemples non supervisés connus à l'avance.

L'apprentissage transductif a été mis en avant par Vapnik en 1998 pour des raisons « philosophiques » très attrayantes. Puisque dans de nombreux cas, on connaît à l'avance les points de $\mathcal{X}$ sur lesquels on va être interroger, est-il judicieux dans ces cas là d'apprendre une fonction h définie sur l'ensemble de l'espace d'entrée $\mathcal{X}$? Cela semble faire appel à une information plus importante, et donc plus difficile à acquérir, que ce que nous avons besoin. De même qu'il vaut mieux apprendre directement une fonction de décision plutôt que de faire le détour par l'apprentissage de la distribution génératrice des données $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$, de même pourrait-on imaginer d'apprendre à ne prédire que sur les points sur lesquels nous serons interrogés.

Vapnik argumente que la transduction fonctionne si les points sur lesquels on va être interrogé permettent une factorisation non triviale de l'espace des hypothèses et donc une réduction de l'espace à explorer. De plus les bornes de généralisation peuvent être beaucoup plus précises car ne portant pas sur une distribution de probabilité $\mathbf{p}_{\mathcal{X}}$ inconnue.

Sans que les algorithmes d'apprentissage transductif semblent encore très convaincants sur leur supériorité, l'argument philosophique sur lequel ils s'appuient est très intéressant et mérite considération. On peut attendre des progrès sur cet apprentissage, y compris dans des directions non envisagées par Vapnik. C'est un souhait que l'on peut formuler.

Résumé

- L'apprentissage semi-supervisé traite un ensemble d'apprentissage partiellement supervisé et cherche à tirer parti de la partie supervisée comme de la partie non supervisée pour apprendre une loi générale de classification.

Copyright © 2010 Eyrolles.

- Le co-apprentissage est l'une des premiers exemples d'apprentissage semi-supervisé. Il consiste à construire deux modèles prédictifs fondés sur deux vues différentes des données, et d'utiliser chaque modèle pour sélectionner les données non supervisées dont l'étiquette prédite semble la plus sûre et fournir ces données à l'autre méthode d'apprentissage.
- La méthode de SVM transductif utilise à la fois les données supervisées et les données non supervisées pour maximiser la marge.
- Les méthodes récentes incluent des méthodes basées sur des graphes qui expriment les relations entre les données et utilisent ce graphe pour propager une information sur les étiquettes des points non supervisés.
- Pour que les données non supervisées puissent être utiles à l'apprentissage, il est nécessaire de formuler des suppositions sur les données. Deux suppositions ont été essentiellement énoncées jusqu'ici. La première appelée « hypothèse de cluster » ou de manière équivalente « hypothèse de lissage » suppose que deux données appartenant à un même cluster ont la même étiquette. Elle est à la base des SVM transductifs. La seconde, « hypothèse de sous-variété », suppose que les données définies en grande dimension appartiennent en fait à une variété de petite dimension, et que sur cette variété deux points proches ont des étiquettes proches. Elle est à la base des méthodes utilisant des graphes.
- L'apprentissage transductif cherche à classer seulement la partie non supervisée de l'ensemble d'apprentissage. L'apprentissage inductif cherche une loi générale valable dans tout l'espace de représentation.

Copyright © 2010 Eyrolles.

Chapitre 20

Vers de nouvelles tâches et de nouvelles questions

Ce chapitre fournit un panorama non exhaustif mais évocateur de nouvelles directions de recherche visant à dépasser certaines des limitations du paradigme dominant en apprentissage artificiel.

Celui-ci concerne essentiellement l'apprentissage supervisé dans lequel on cherche une fonction associant une étiquette à une entrée. Il suppose que les données sont tirées aléatoirement suivant une distribution stationnaire. De plus, les étiquettes sont simples, correspondant à une classe ou un réel.

Un grand mouvement se développe pour sortir de ce cadre classique. Ainsi, dans l'apprentissage actif, on étudie un apprenant pouvant sélectionner ses exemples pour apprendre plus vite. De ce fait, les données ne sont plus tirées aléatoirement. C'est également souvent le cas dans l'apprentissage en ligne et à partir de flux. À l'exigence de pouvoir fournir une décision à n'importe quel moment, s'ajoute dans ces cas la possibilité que l'environnement puissent changer. Cela renouvelle profondément les problèmes à résoudre et les concepts à mettre en œuvre. De même, on commence à envisager l'apprentissage produisant des étiquettes structurées, comme des arbres ou des séquences par exemple. Finalement, de nouvelles tâches d'apprentissage apparaissent sous la pression de nouvelles applications, comme c'est le cas pour l'apprentissage collaboratif.

Ces nouveaux cadres et leurs questions s'accompagnent d'un glissement de paradigme en apprentissage artificiel. Ce chapitre de fin d'ouvrage est ainsi une invitation à explorer d'autres horizons.

Copyright © 2010 Eyrolles.

Sommaire

1 Apprentissage actif 661
1.1 Protocoles d'apprentissage actif 662
1.2 Méthodes et algorithmes de sélection des exemples 663
1.2.1 Méthodes considérant l'hypothèse courante : $h_t \times \mathcal{S}_{ns} \rightarrow \mathbf{x}_{t+1}$ 664
1.2.2 Méthodes considérant l'ensemble des hypothèses candidates : $\mathcal{H}_t \times \mathcal{S}_{ns} \rightarrow \mathbf{x}_{t+1}$ 666
1.3 Analyse théorique 668
2 Apprentissages en ligne, incrémental et par transferts 669
2.1 Apprentissage(s) en ligne 670
2.2 Les contraintes calculatoires de l'apprentissage en ligne 671
2.2.1 Méthodes de gradient stochastique 671
2.2.2 Vers une analyse théorique de l'apprentissage à grande échelle 672
2.2.3 Algorithmes pour les flux de données 672
2.3 Déplacement covarié (*Covariate shift*) : $\mathbf{p}_{\mathcal{X}}$ varie 672
2.4 Dérive de concept : $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}$ varie 674
2.5 L'apprentissage par transfert entre tâches 674
2.6 Quelle analyse théorique? Le cas de l'apprentissage en ligne 675
3 Apprentissage à partir de flux de données 677
3.1 Échantillonnage et comptage 678
3.1.1 Algorithme du *reservoir sampling* 678
3.1.2 Algorithme du *min-wise sampling* 679
3.1.3 Calcul d'entropie sur les flux 679
3.1.4 Calcul de résumés sur les flux 679
3.2 Recherche de règles d'association 679
3.3 Détection de ruptures 680
3.4 Classification non supervisé 681
4 Apprentissage de sorties structurées 682
4.1 Les problèmes à résoudre 683
4.2 La fonction de perte 683
4.3 Les approches 683
4.3.1 Le cas de l'apprentissage multi-instance 685
5 Apprentissage pour le filtrage collaboratif 686
5.1 Les approches 686
5.1.1 Les approches par modèle de l'utilisateur 686
5.1.2 Les approches par similarité et plus proches voisins 687
5.2 Les difficultés 689
5.2.1 Traitement des matrices 689
5.2.2 Mesures de similarité 690
5.2.3 Évaluation des algorithmes de filtrage collaboratif 691
5.2.4 Limitations générales 692

Copyright © 2010 Eyrolles.

1. Apprentissage actif

Excepté pour le cas de l'apprentissage par renforcement, cet ouvrage a essentiellement rendu compte de protocoles d'apprentissage dans lesquels l'apprenant est passif, recevant les données, supposées tirées aléatoirement, que veut bien lui fournir la nature ou l'expert. La différence est notable avec les agents cognitifs naturels qui agissent sur le monde et sont en partie responsables du flot de données leur parvenant. Ne passe-t-on pas ainsi à côté d'une source d'information, l'expérimentation, qui pourrait modifier la puissance de l'apprentissage, et peut-être rendre compte de la différence entre les exigences des théorèmes d'apprenabilité et l'efficacité souvent constatée des apprentissages effectifs ?

Il se pourrait que, d'une part, il existe des concepts apprenables *seulement* dans un protocole actif et non dans le protocole passif. Cela signifierait que le caractère passif de l'apprentissage impliquerait une limite d'ordre fondamental sur le transfert d'information en apprentissage ; et, d'autre part, que les concepts apprenables dans le protocole passif s'apprennent *plus vite* (requérant moins d'exemples et/ou moins de ressources calculatoires) en utilisant une stratégie d'apprentissage actif.

Quatre questions fondamentales se posent ainsi à propos de l'apprentissage actif :

1. Est-il possible d'*apprendre des régularités qui seraient non apprenables en utilisant un apprentissage passif* ?
2. *Peut-on apprendre plus rapidement*, c'est-à-dire avec moins d'exemples d'apprentissage, des régularités apprenables avec un apprentissage passif?
3. *Comment procéder pour sélectionner* au mieux les exemples d'apprentissage ?
4. *Comment évaluer l'apprentissage* alors que les données ne sont plus distribuées aléatoirement et identiquement ? Cette question a naturellement un impact sur le critère inductif à utiliser dans l'apprentissage actif.

La réponse à la *première question* est négative. La possibilité de poser des questions ou de réaliser des expériences ne permet pas d'apprendre des concepts cibles qui ne le seraient pas apprenables par l'apprentissage passif.

En revanche, en réponse partielle à la *seconde question*, il est facile d'exhiber un exemple dans lequel l'apprentissage actif est nettement plus efficace que l'apprentissage passif.

— EXEMPLE **Différence d'échantillonnage entre apprentissages passif et actif** —

On suppose que les données sont disposées sur la droite des réels suivant une distribution $\mathsf{p}_{\mathcal{X}}$, et que l'espace des hypothèses est constitué des fonctions seuil, $\mathcal{H} = \{h_w\}$:

$$h_w(x) = 1 \text{ si } x > w, \text{ et } 0 \text{ autrement.}$$

$$h_w(x) = \begin{cases} 1 & \text{if } x \geq w \\ 0 & \text{if } x < w \end{cases}$$

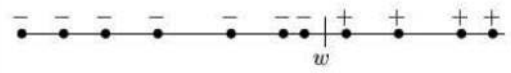


FIG. 20.1: *Apprentissage d'un concept seuil sur la droite.*

La théorie statistique de l'apprentissage prédit que si la fonction cible appartient à $\mathcal{H}$, alors il suffit de tirer un échantillon aléatoire de données, suivant la distribution $\mathsf{p}_{\mathcal{X}}$, de taille $\bullet(1/\varepsilon)$ pour obtenir une hypothèse de taux d'erreur d'au plus ε (suivant $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$).

Copyright © 2010 Eyrolles.

Supposons maintenant que m points non étiquetés soient tirés suivant la distribution $\mathbf{p}_{\mathcal{X}}$. Sur la droite des réels, leurs étiquettes inconnues forment une séquence de '0' suivie d'une séquence de '1'. Le but est de trouver un point w pour lequel se produit la transition entre les deux séquences. Une simple recherche binaire permet de trouver un tel point en demandant l'étiquette de seulement $\mathcal{O}(\log m)$ exemples. L'apprentissage actif peut donc fournir un avantage exponentiel sur le nombre d'étiquettes nécessaires. On notera cependant que cette efficacité s'appuie sur une connaissance *a priori* du type de concept cible.

Dans le cas général, néanmoins, la supériorité de l'apprentissage actif n'est pas garantie. Les travaux de recherche tant expérimentaux que théoriques essaient de cerner les conditions sous lesquelles cette supériorité existe et son importance. Nous y reviendrons dans la section 1.3.

1.1 Protocoles d'apprentissage actif

Les scénarios d'apprentissage actif se distinguent suivant que :

1. L'apprenant est libre de ses « questions » à la nature. On parle de ***requêtes constructives*** car l'apprenant construit lui-même ses questions ou, plus largement, ses expériences.
2. L'apprenant est contraint de choisir les exemples d'apprentissage dans un ensemble prédéterminé d'exemples. Ce scénario, appelé ***apprentissage actif par sélection*** se subdivise lui-même en deux sous-scénarios :
   - Le scénario dans lequel il pré-existe un ensemble d'exemples non étiquetés dans lequel l'apprenant peut sélectionner ceux pour lesquels il désire connaître l'étiquette (*pool-based active learning*)
   - Le scénario dans lequel les exemples non étiquetés arrivent en séquence ou flux et l'apprenant doit décider pour chaque nouvel exemple s'il en demande l'étiquette ou non (*stream-based active learning*).

Dans le premier scénario, l'apprenant peut demander l'étiquette de n'importe quel exemple possible dans l'espace d'entrée $\mathcal{X}$. Ainsi, dans le cas de l'apprentissage des mouvements possibles d'un robot, il est possible pour l'apprenant de demander quelles sont les coordonnées de la main du robot étant donnée en entrées n'importe quelle valeur des commandes de chaque degré de liberté du bras mécanique [CGJ96]. Ce type de scénario peut cependant être délicat lorsque l'étiqueteur est un expert humain. En effet, il se peut que la requête ainsi construite ne soit pas étiquetable. Par exemple, un système d'apprentissage pour la reconnaissance de chiffres manuscrits pourrait trouver qu'un exemple particulièrement intéressant est celui correspondant à un mélange de '4' et de '8'. Malheureusement, l'exemple ne pourra pas être étiqueté et une requête aura été dépensée pour rien [BK92]. En revanche, [KWJ+04] décrit une application récente prometteuse dans laquelle un « robot scientist » décide lui-même quelles expériences biologiques réaliser. Il peut s'agir par exemple d'un mélange de solutions chimiques constituant un milieu nutritif et d'un mutant de levure. Une étiquette correspond alors à savoir si le milieu est très favorable à la croissance de la levure ou non. Les essais de ce scénario, qui utilise une méthode de programmation logique inductive (voir chapitre 5) pour l'apprentissage, montre un gain très significatif par rapport à un scénario expérimental naïf. Nous ne nous intéresserons cependant pas davantage à ce scénario dans la suite de cette section, même s'il est très intéressant, car il est moins traité pour le moment, en partie pour des raisons théoriques[1].

[1] Dans ce cadre en effet, par contraste avec l'apprentissage actif par sélection, on ne peut plus rien dire sur la distribution sous-jacente $\mathbf{p}_{\mathcal{X}}$.

Copyright © 2010 Eyrolles.

Un autre facteur important dans la définition du scénario d'apprentissage concerne le *type de questions que l'apprenant peut poser* à la nature. Les théoriciens de l'apprentissage ont principalement étudié les protocoles suivants :

- Le **protocole de requêtes binaires arbitraires** (*arbitrary binary queries*) : l'apprenant peut poser des questions arbitraires binaires (oui / non) sur les concepts et sur la distribution sous-jacente des exemples. Plus précisément, l'apprenant peut soumettre une requête qui est un sous-ensemble de $\mathcal{F} \times \mathbf{p}_{\mathcal{X}}$ et demander à l'oracle si le concept cible f et la distribution appartiennent à ce sous-ensemble. Ainsi, par ses questions, l'apprenant peut éliminer une partie des concepts cibles et des distributions possibles [KMT93].
- Le **protocole de requêtes d'appartenance** (*membership query*) : l'apprenant peut choisir n'importe quel exemple dans $\mathcal{X}$ et demander à l'oracle s'il appartient ou non au concept cible. Il s'agit donc d'un protocole d'apprentissage actif constructif. La question fondamentale est alors de calculer le nombre minimal de questions nécessaires pour identifier la meilleure hypothèse. ([Ang88])
- Le **protocole de requêtes d'équivalence** (*equivalence query*) : « Ce concept est-il équivalent au concept cible ? ». L'apprenant peut proposer une hypothèse h, et l'oracle, soit l'informe que l'hypothèse est logiquement équivalente à la fonction cible, soit lui fournit un contre-exemple infirmant l'hypothèse. [Ang88]
- Le **protocole de requêtes statistiques** (*statistical query model*) : l'apprenant ne peut avoir accès directement aux exemples étiquetés, mais peut poser des questions sur les statistiques des exemples étiquetés (par exemple 3/4 des 52 exemples sont positifs). Ce modèle est particulièrement utile dans le cas de données dont l'étiquette peut être erronée (bruit de classification) [KV94].

1.2 Méthodes et algorithmes de sélection des exemples

Les algorithmes existants d'apprentissage actif opèrent tous selon le même schéma. Que ce soit parmi un ensemble d'exemples non étiquetés ($\mathcal{S}_{ns}$) (approche *pool-based*) ou face à chaque nouvel exemple d'un flux (approche *stream-based*), ils évaluent l'information que peut apporter l'exemple courant et décident s'ils le retiennent pour requête de son étiquette ou pas. L'algorithme générique est donc le suivant :

Algorithme 46 : Algorithme générique d'échantillonnage actif

Notations :

- h : une hypothèse prédictive munie d'un algorithme d'apprentissage
- $\mathcal{S}_s$ et $\mathcal{S}_{ns}$: des ensembles d'exemples étiquetés (supervisés) et non étiquetés
- n : le nombre d'exemples d'apprentissage souhaité
- $\mathcal{S}_s$: échantillon d'apprentissage courant (avec $|\mathcal{S}_s| < n$)
- Une fonction **Utile** : $\mathcal{X} \times \mathcal{H} \rightarrow \mathbb{R}$ qui estime l'utilité d'un exemple $\mathbf{x}$ pour l'apprentissage d'une hypothèse

tant que $|\mathcal{S}_s| < n$ **faire**

(A) Apprendre : explorer $\mathcal{H}$ grâce à $\mathcal{S}_s$ (et éventuellement $\mathcal{S}_{ns}$)

(B) Rechercher l'exemple $q = \text{ArgMax}_{u \in U}$ **Utile**$(u, \mathcal{H})$

(C) Retirer q de $\mathcal{S}_{ns}$ et demander son étiquette $f(q)$ à l'oracle

(D) Ajouter $(q, f(q))$ à $\mathcal{S}_s$

fin tant que

Les méthodes diffèrent par la définition de la fonction `Utile`. L'apprentissage actif vivant encore sa période exploratoire dans laquelle manque un socle théorique solide et général, de nombreuses approches de nature heuristique ont été proposées. On peut grossièrement distinguer :

Copyright © 2010 Eyrolles.

- Les approches qui *considèrent l'hypothèse courante* h_t entretenue par l'apprenant et qui s'appuient sur certaines mesures prenant en compte cette hypothèse et les données pour choisir la prochaine donnée à examiner : $h_t \times \mathcal{S}_{ns} \rightarrow \mathrm{x}_{t+1}$.
- Les approches qui *prennent en compte l'ensemble des hypothèses candidates* (par exemple l'espace des versions) et cherchent à en réduire le « volume » : $\mathcal{H}_t \times \mathcal{S}_{ns} \rightarrow \mathrm{x}_{t+1}$

1.2.1 Méthodes considérant l'hypothèse courante : $h_t \times \mathcal{S}_{ns} \rightarrow \mathrm{x}_{t+1}$

Les méthodes qui considèrent l'hypothèse courante h_t pour évaluer le mérite des exemples non étiquetés s'appuient généralement sur une mesure de l'incertitude de l'hypothèse par rapport à ces exemples potentiels. L'idée intuitive sous-jacente est d'examiner en priorité les exemples pour lesquels l'hypothèse est la plus incertaine sur l'étiquette à leur attribuer.

Échantillonnage par mesure d'incertitude

L'une des techniques d'apprentissage actif les plus employée consiste à sélectionner l'exemple pour lequel l'hypothèse courante est la plus incertaine. Cette approche proposée par [LG94] se traduit facilement dans le cas où l'hypothèse associe un degré de certitude à l'étiquette calculée comme dans les modèles probabilistes. On testera alors en priorité les exemples pour lesquels le degré de probabilité est le plus proche de 0.5 ou bien ceux pour lesquels l'*incertitude* est la plus grande :

$$\mathrm{x}^{\star}_{\mathrm{INC}} = \underset{\mathrm{x} \in \mathcal{S}_{ns}}{\mathrm{Argmax}} \left\{ \frac{1}{\underset{y \in \mathcal{Y}}{\mathrm{ArgMax}}\ \hat{p}(y|\boldsymbol{x})} \right\}$$

Une méthode plus générale utilise l'*entropie* des étiquettes possibles comme mesure d'incertitude :

$$\mathrm{x}^{\star}_{\mathrm{ENT}} = \underset{\mathrm{x} \in \mathcal{S}_{ns}}{\mathrm{Argmax}} \left\{ - \sum_i \mathsf{P}(y_i|\mathrm{x}; h)\ \log \mathsf{P}(y_i|\mathrm{x}; h) \right\}$$

Les deux mesures sont équivalentes dans le cas de la classification binaire.

Une troisième approche apparentée consiste à utiliser la *proximité des exemples avec la frontière de décision* comme estimation de l'incertitude de leur classification.

Échantillonnage par réduction de la variance ou par information de Fisher

Une méthode liée à la précédente pour le cas de la régression a été proposée dans [CGJ96]. L'idée est de minimiser l'erreur en généralisation. En s'appuyant sur la décomposition de l'erreur par [GBD92] qui fait ressortir le *bruit*, le *biais* et la *variance* (voir section 4.2 du chapitre 2), ils proposent de réduire le seul terme contrôlable avec les exemples qui est la variance.

Pour cela, la distribution des sorties de l'hypothèse courante h_t est estimée quand l'exemple candidat est étiqueté et ajouté à l'ensemble d'apprentissage $\mathcal{S}_s$. L'exemple sélectionné est alors celui qui réduit au maximum la variance future. L'article [CGJ96] décrit cette approche dans le cas de perceptrons multicouches. Elle est assez coûteuse en calculs.

Une approche analogue pour la classification, basée sur l'information de Fisher, a été proposée dans [ZO00].

L'*information de Fisher*(1890-1962) est une mesure de la quantité d'informations qu'une variable observable X apporte sur un paramètre inconnu θ dont la fonction de vraisemblance dépend. Formellement, l'information de Fisher est la variance du score, soit encore le second

Copyright © 2010 Eyrolles.

moment du *score*, c'est-à-dire la dérivée de la log-vraisemblance par rapport à θ. On a donc :

$$\mathcal{I}(\theta) = \mathbb{E}\left\{ [\frac{\partial}{\partial\theta} \log f(\mathbf{x};\theta)]^2 | \theta \right\}$$

L'information de Fisher peut aussi s'écrire :

$$\mathcal{I}(\theta) = -\int_{\mathbf{x}} \mathsf{p}(\mathbf{x}) \int_{y} \mathsf{p}(y|\mathbf{x};\theta) \frac{\partial^2}{\partial\theta^2} \log \mathsf{p}(y|\mathbf{x};\theta)$$

et peut donc s'interpréter comme l'incertitude sur la distribution $\mathsf{p}(\mathbf{x})$ par rapport aux paramètres de l'hypothèse considérée. Pour des hypothèses à plusieurs paramètres, l'information de Fisher prend la forme d'une matrice de covariance. On sélectionne alors l'exemple candidat qui minimise le ratio de l'information de Fisher :

$$\mathbf{x}^*_{\text{FIS}} = \underset{\mathbf{x}\in\mathcal{S}_{ns}}{\text{Argmin}}\ tr\big(\mathcal{I}_{\mathbf{x}}(\theta)^{-1}\mathcal{I}_{\mathcal{S}_{ns}}(\theta)\big)$$

où $\mathcal{I}_{\mathbf{x}}(\theta)$ est la matrice d'information de Fisher pour un exemple candidat non étiqueté $\mathbf{x} \in \mathcal{S}_{ns}$, et $\mathcal{I}_{\mathcal{S}_{ns}}$ est la matrice analogue intégrée sur l'ensemble des exemples non étiquetés. La fonction $tr(\cdot)$ est la somme des termes diagonaux de la matrice. L'équation correspond donc au produit de la matrice inverse de $\mathcal{I}_{\mathbf{x}}(\theta)$ et de $\mathcal{I}_{\mathcal{S}_{ns}}$.

L'intérêt du ratio de l'information de Fisher est que $\mathcal{I}_{\mathbf{x}}(\theta)$ n'indique pas seulement à quel point l'hypothèse courante est incertaine à propos de l'étiquette de l'exemple $\mathbf{x}$ (i.e. la magnitude des éléments diagonaux de la matrice), mais qu'on peut aussi en tirer une information sur les paramètres de l'hypothèse qui sont le plus liés à cette incertitude. De même, la matrice $\mathcal{I}_{\mathcal{S}_{ns}}$ fournit la même information à propos de l'ensemble des exemples non étiquetés.

En minimisant le ratio de Fisher, on tend à sélectionner les exemples associés à une variance de l'hypothèse la plus proche possible de la variance induite par l'ensemble des exemples non étiquetés $\mathcal{S}_{ns}$. L'article [ZO00] fournit une justification théorique plus détaillée.

Les méthodes par réduction de variance ou par minimisation du ratio de Fisher sont très apparentées dans leur objectif : la minimisation d'une variance. Elles sont malheureusement aussi toutes les deux *coûteuses en termes de calcul* car elles requièrent l'inversion de matrices dont la taille K est le nombre de paramètres du modèle dont sont issues les hypothèses. Le coût général de ces stratégies de sélection active des exemples est de complexité en $\mathcal{O}(|\mathcal{S}_{ns}|\,K^3)$. C'est pourquoi des méthodes approchées ont été développées.

Échantillonnage par estimation de réduction de l'erreur

Plutôt que de chercher à réduire indirectement l'erreur en généralisation en minimisant le terme de variance, des méthodes directes ont été explorées. L'idée est alors d'estimer l'erreur de généralisation qui résulterait de la prise en compte d'un nouvel exemple d'apprentissage $\mathbf{x} \in \mathcal{S}_{ns}$ qui serait étiqueté, et de sélectionner l'exemple conduisant à l'erreur estimée la plus faible.

Ainsi les auteurs de [RM01] ont été les premiers à proposer une telle stratégie pour de la classification de texte en utilisant un classifieur bayésien naïf. Ceux de [ZLG03] ont combiné cette approche avec de l'apprentissage semi-supervisé et ont ainsi obtenu des gains très significatifs par rapport à un échantillonnage aléatoire ou basé sur l'incertitude (voir plus haut).

Malheureusement, cette approche est aussi *la plus coûteuse*. Non seulement elle requiert que soit estimée la future erreur de généralisation sur $\mathcal{S}_{ns}$ pour chaque exemple possible, mais une nouvelle hypothèse doit être calculée pour chaque étiquette possible de chaque exemple, et, pour l'estimation de l'erreur en généralisation, il doit être fait de même pour tous les exemples de $\mathcal{S}_{ns}$.

Copyright © 2010 Eyrolles.

Heureusement, pour quelques classes de modèles (espaces d'hypothèses $\mathcal{H}$), les calculs peuvent être rapides.

Échantillonnage par maximisation de l'espérance de changement d'hypothèse

Une technique possible d'apprentissage actif consiste à sélectionner l'exemple qui conduirait au plus grand changement d'hypothèse si on connaissait son étiquette. Afin de mesurer cette variation de l'hypothèse courante, une technique est de mesurer le gradient qui résulterait de la prise en compte de cet exemple. Les expériences réalisées ont montré qu'un gain pouvait ainsi être obtenu sur le nombre d'exemples à prendre pour l'apprentissage, mais la technique est cependant coûteuse en termes de calcul.

Il est à noter que l'approche par mesure d'incertitude peut avoir tendance à sélectionner des exemples aberrants (*outliers*), ce qui n'est pas le cas des approches par réduction de variance ou par réduction de l'erreur.

Ce n'est pas non plus le cas des approches qui considèrent l'ensemble des hypothèses candidates que nous examinons maintenant.

1.2.2 Méthodes considérant l'ensemble des hypothèses candidates : $\mathcal{H}_t \times \mathcal{S}_{ns} \rightarrow \mathbf{x}_{t+1}$

Plutôt que de ne s'appuyer que sur l'hypothèse courante considérée par l'apprenant, une autre méthode d'apprentissage actif consiste à examiner directement l'objectif de l'apprentissage qui est de réduire au plus vite le nombre d'hypothèses qui peuvent rendre compte du monde. C'est ce que le joueur de Mastermind, par exemple, essaie de faire en proposant des exemples et en en demandant l'étiquette par l'oracle[2].

Nous avons vu au chapitre 4 que l'*espace des versions* est l'ensemble des hypothèses de $\mathcal{H}$ qui sont cohérentes avec les données d'apprentissage, c'est-à-dire dont le risque empirique est nul. Une approche pour l'apprentissage actif est donc de sélectionner les exemples d'apprentissage qui conduisent à la plus grande réduction possible de cet espace des versions. C'est ce que Tom Mitchell avait déjà évoqué lorsqu'il avait défini l'espace des versions et l'algorithme d'élimination des candidats (voir chapitre 4). Idéalement, avait-il souligné, dans le cas de l'apprentissage de concept (où il n'y a que deux classes), il faudrait choisir les exemples qui permettent d'éliminer la moitié de l'espace des versions une fois que leur étiquette est connue. Cependant, il ne donnait pas de méthode pour déterminer ces exemples maximalement discriminants.

Échantillonnage par mesure de réduction de l'espace des versions

Cohn, Atlas et Ladner ont proposé en 1994, [CAL94], le concept de *région d'incertitude* dans $\mathcal{X}$ pour désigner l'ensemble des exemples pour lesquels les hypothèses de l'espace des versions courant ne sont pas d'accord sur leur étiquette. Il est clair qu'en choisissant l'un de ces exemples et en demandant son étiquette, on est certain d'éliminer des hypothèses de l'espace des versions, à savoir toutes celles qui n'avaient pas prédit l'étiquette révélée par la requête.

La région d'incertitude est recalculée après chaque requête jusqu'à ce que le budget en nombre de requêtes ait été atteint ou jusqu'à ce que l'espace des versions ait été suffisamment réduit. Ce protocole est compatible avec l'approche constructive des exemples comme avec l'approche sélective.

Le principal obstacle à sa mise en œuvre vient de la difficulté de calculer la région d'incertitude qui peut rapidement être prohibitive quand l'espace des hypothèses est complexe. Ainsi, la figure 20.2 montre que même dans le cas simple de concepts définis comme des hyperrectangles

[2] La notion d'étiquette est cependant plus complexe dans le cas du Mastermind puisqu'elle est composée et prend en compte la couleur et la position des pions de la combinaison testée par le joueur.

Copyright © 2010 Eyrolles.

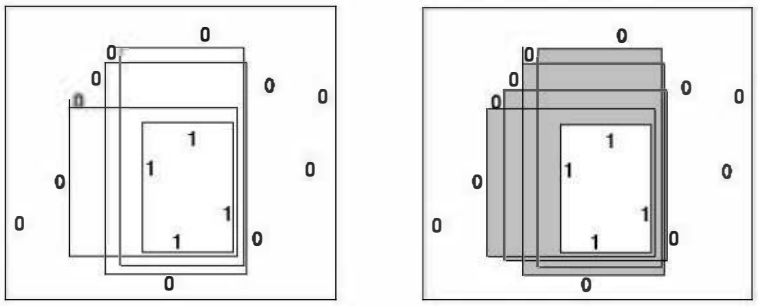


FIG. 20.2: *On suppose que l'espace des concepts consiste en les rectangles définis sur l'espace des exemples. À gauche sont représentés quelques exemples positifs (étiquetés '1') et quelques exemples négatifs (étiquetés '0'), ainsi que certains concepts candidats cohérents avec ces données. À droite est représentée la région d'incertitude associée, c'est-à-dire l'ensemble des points de $\mathcal{X}$ pour lesquels les concepts candidats ne sont pas tous d'accord sur leur étiquette (tiré de [CAL94]).*

de $\mathcal{X}$, la représentation de la région d'incertitude peut impliquer un grand nombre de paramètres (les coins des hyperrectangles maximaux et minimaux cohérents avec les données d'apprentissage vus jusque-là).

[CAL94] montre comment implémenter cette technique dans le cas du perceptron multicouches.

Échantillonnage par mesure de désaccord au sein d'un comité

Une autre approche [SOS92] visant à réduire l'espace des versions consiste à échantillonner des hypothèses dans l'espace des versions, afin d'obtenir un *comité d'hypothèses* dont on espère qu'il est représentatif, et de faire ensuite voter ce comité pour déterminer l'étiquette des exemples candidats. Les exemples pour lesquels le comité est le plus en désaccord sont alors sélectionnés.

Cependant, cette approche, très séduisante sur le plan théorique, pose des problèmes pratiques et calculatoires. Il faut en effet être capable d'échantillonner des hypothèses dans l'espace des versions qui soient représentatives de toutes ses régions. Il faut de plus définir et calculer une mesure de désaccord entre les hypothèses du comité.

Plusieurs méthodes ont été proposées pour l'échantillonnage des hypothèses (voir [Set09] pour des références). De même, pour des mesures de désaccord, pour lesquelles deux grandes approches existent.

La première utilise une *mesure d'entropie sur le vote* :

$$\mathbf{x}^{\star}_{\mathrm{VE}} = \underset{\mathbf{x}}{\mathrm{Argmax}} \left\{ -\sum_{i} \frac{V(y_i)}{C} \log \frac{V(y_i)}{C} \right\} \tag{20.1}$$

où C est le nombre d'hypothèses dans le comité, y_i décrit toutes les étiquettes possibles de $\mathbf{x}_i$, et $V(y_i)$ est le nombre de votes correspondant à une étiquette possible y_i.

Une deuxième mesure est définie par une *divergence de Kullback-Leibler* entre la distribution des étiquettes prédites pour les exemples non étiquetés par une hypothèse donnée $\mathbf{p}_{h_c}$ et par l'ensemble du comité $\mathbf{p}_{\mathcal{C}}$.

$$\mathbf{x}^{\star}_{\mathrm{KL}} = \underset{\mathbf{x}}{\mathrm{Argmax}} \frac{1}{C} \sum_{c=1}^{C} \mathbf{D}(\mathbf{p}_{h_c} || \mathbf{p}_{\mathcal{C}}) \tag{20.2}$$

Copyright © 2010 Eyrolles.

L'exemple jugé le plus informatif étant celui qui maximise le désaccord entre une hypothèse du comité et le consensus général.

La méthode de sélection par comité (*query-by-committee*) a été analysée par Freund et ses collègues [FSST97]. Dans le cas où les données sont supposées placées uniformément sur la surface d'une sphère unité dans $\mathbb{R}^d$ centrée à l'origine, et où le concept cible est supposé être un hyperplan passant par l'origine, il a été montré qu'il était possible d'apprendre une hypothèse d'erreur de généralisation de ε après avoir observé $\Theta(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$ dans un flux de données et avoir demandé $\Theta(d\log\frac{1}{\varepsilon})$ étiquettes, ce qui constitue un gain exponentiel par rapport à l'échantillonnage aléatoire normalement requis de $\mathcal{O}(\frac{d}{\varepsilon})$ exemples. Il peut être montré qu'il s'agit là d'un résultat optimal : il suffit de calculer le nombre de disques sphériques de rayon ε qui peuvent être réparties sur une sphère unité dans $\mathbb{R}^d$. Ce résultat remarquable doit cependant être tempéré par la difficulté de mise en œuvre de la méthode.

Tong et Koller [TK00] ont proposé une méthode adaptée pour l'apprentissage actif de SVM (Séparateurs à Vastes Marges, voir chapitre 14) dans laquelle plusieurs critères géométriques sont utilisés pour estimer le « volume » de l'espace des versions et pour choisir l'exemple conduisant à sa plus grande diminution.

Les méthodes cherchant à diminuer le plus rapidement l'espace des versions sont fondées sur l'hypothèse que le concept cible se trouve dans l'espace des versions, c'est-à-dire en particulier que les données ne sont pas bruitées. Il est possible de relâcher ces contraintes, mais cela pose des problèmes. Cependant, malgré ses difficultés pratiques, ce point de vue sur l'apprentissage actif est celui qui conduit aux analyses théoriques actuelles car il se prête aux approches de type PAC (voir chapitres 2 et 21).

Position par rapport aux « plans d'expériences »

En statistique, les plans d'expérience sont des organisations séquentielles de tests, ou des tests séquentiels, destinés à examiner la dépendance d'une variable de sortie sur des facteurs externes, par exemple, la dépendance du risque cardiovasculaire à la prise d'aspirine, et dont le déroulement dépend des résultats courants du test. À la différence de l'apprentissage actif, le système ne choisit pas lui-même les expériences à réaliser, mais se contente de déterminer quand une série de tests est suffisamment concluante par rapport à une question. On citera en particulier à cet égard le test séquentiel de Wald.

1.3 Analyse théorique

Une question fondamentale est celle de savoir si l'apprentissage actif est efficace, c'est-à-dire s'il permet d'apprendre plus rapidement (avec la même précision) que l'apprentissage passif. De fait, la question devrait même être plus précise. Il ne s'agit en effet pas seulement de compter le *nombre d'exemples d'apprentissage requis* dans chacun des scénarios : actif ou passif. Idéalement, il faudrait aussi *tenir compte du coût calculatoire* qu'implique une méthode d'apprentissage actif pour sélectionner les exemples les plus informatifs.

Empiriquement, de nombreuses études indiquent que l'apprentissage actif peut réduire notablement le nombre d'exemples d'apprentissage. Cependant, certaines expériences montrent *a contrario* que ce n'est pas nécessairement le cas. De plus, il faut se méfier du biais de publication qui pousse à publier surtout les résultats positifs.

Que dit alors l'analyse théorique ? On aimerait qu'elle nous permette de calculer des bornes sur le nombre de requêtes nécessaires pour apprendre approximativement et presque certainement les concepts cibles (cadre de l'apprentissage PAC). Cependant, une première difficulté est que la théorie classique de l'apprentissage s'appuie sur le fait que la distribution des exemples

Copyright © 2010 Eyrolles.

d'apprentissage est supposée la même que la distribution des exemples test, ceux sur lesquels la vraie valeur de l'apprentissage sera mesurée. De plus, il est supposé que les exemples sont indépendants les uns des autres, ce qui est essentiel pour invoquer les théorèmes de type central limite qui sont au cœur de tous les calculs de bornes. Or ces deux suppositions sur le tirage des données se trouvent invalidées dans l'apprentissage actif. Comment alors procéder?

Les premiers travaux théoriques sur l'apprentissage actif se sont placés dans le cadre de l'apprentissage par requêtes constructives (voir [Ang03] pour une revue récente). Toute une série de travaux récents ont exploré l'apprentissage actif sélectif dans lequel l'apprenant dispose d'une information sur la vraie distribution des exemples grâce à l'ensemble des exemples non étiquetés $\mathcal{S}_{ns}$. Les résultats obtenus dépendent d'hypothèses *a priori* sur les données ou sur l'espace des hypothèses $\mathcal{H}$:

- Espace $\mathcal{H}$ de taille finie ou non (dans ce dernier cas, on tient compte de la dimension de Vapnik-Chervonenkis $d_{\mathcal{H}}$, voir chapitre 21).
- Cas *réalisable* (la fonction cible f appartient à $\mathcal{H}$) ou *agnostique* (f peut ne pas appartenir à $\mathcal{H}$).
- Distribution des exemples uniforme ou pas dans $\mathcal{X}$.
- La tâche concerne l'apprentissage de concept, de classification multi-classe ou la régression.

La plupart des études portent sur un cadre particulier défini par un choix des possibilités décrites ci-dessus et sur un type particulier d'apprentissage (e.g. apprentissage de perceptron linéaire).

L'un des résultats les plus généraux obtenu [BHW08] montre que l'apprentissage actif surpasse l'apprentissage passif (en terme d'économie en nombre d'exemples requis pour apprendre) pour presque toutes les classes de concepts de $d_{\mathcal{H}}$ finie, avec un gain exponentiel : $\mathcal{O}\left(\frac{1}{\log(1/\varepsilon)}\right)$ au lieu de $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$. Il existe cependant toujours des cas très particuliers pour lesquels le gain est nul.

Il faut noter que les méthodes et surtout les analyses théoriques diffèrent beaucoup entre l'apprentissage pour la classification et l'apprentissage pour la régression.

Par ailleurs, les méthodes et analyses supposent que le concept cible est fixe et la distribution des exemples stationnaires. Lorsque ce n'est pas le cas, on entre dans le cadre de l'apprentissage en ligne pour lequel des méthodes spécifiques d'apprentissage actif restent à imaginer.

2. Apprentissages en ligne, incrémental et par transferts

Les premiers systèmes d'apprentissage étaient presque tous incrémentaux. Ils apprenaient au fur et à mesure que les exemples étaient disponibles. C'était ainsi le cas du perceptron (voir chapitre 9), du programme Checker de Samuel [Sam59], du programme ARCH de Winston [Win70], ou de l'algorithme d'élimination des candidats (voir chapitre 4) pour n'en citer que quelques-uns. Plusieurs raisons y concourraient. D'abord, nombre de ces programmes visaient à modéliser l'apprentissage humain, qui se fait essentiellement de manière incrémentale. Ensuite, les ressources calculatoires des premiers systèmes ne permettaient pas la mémorisation et le traitement de grosses bases d'exemples. Cet état de fait a été entièrement bouleversé à partir des années 1980, en partie sous la pression d'une justification de la recherche par les applications et en raison des capacités accrues des ordinateurs. Les nouveaux algorithmes développés : arbres de décision, réseaux connexionnistes, Séparatrices à Vastes Marges, systèmes d'inférence grammaticaux, et bien d'autres, sont tous des systèmes dits *batch learners* en anglais, ce qui signifie qu'ils traitent

Copyright © 2010 Eyrolles.

les données d'un seul coup en optimisant un critère inductif sur l'*ensemble* de la base d'exemples. Lorsque des données nouvelles deviennent disponibles, ces systèmes reprennent leur apprentissage de zéro en incorporant ces nouveaux exemples dans la base d'apprentissage.

Cela ne pose pas de vrais problèmes tant que les ressources permettent ce retraitement à chaque fois que de nouvelles données arrivent et surtout tant que l'on suppose que les données sont générées par un processus stationnaire et indépendamment les unes des autres (distribution i.i.d. pour identiquement et indépendamment distribuée). À partir du moment où l'on envisage des apprentissages à partir de flux de données rapides et à horizon lointain (*long-life learning*), en revanche, il faut réexaminer non seulement les algorithmes et leurs propriétés computationnelles, mais aussi les fondements des théories statistiques de l'apprentissage sous-jacents à ces algorithmes. Cette section décrit certaines des directions étudiées actuellement pour développer une nouvelle science de l'apprentissages.

2.1 Apprentissage(s) en ligne

Si l'apprentissage en ligne est maintenant évoqué dans de nombreux contextes, il recouvre en fait des concepts et des aspects différents.

1. **Aspect computationnel**. On parle d'apprentissage en ligne pour désigner des algorithmes d'apprentissage capables de traiter les données au fur et à mesure de leur arrivée sans avoir à traiter à nouveau les anciens exemples.
2. **Cadre théorique de l'apprentissage en ligne**. Il désigne un scénario dans lequel l'algorithme reçoit séquentiellement des données non étiquetées et doit calculer leur étiquette avant de recevoir la « vraie » étiquette. Le critère à minimiser est la perte cumulée de l'algorithme sur la séquence d'apprentissage.
3. **Cadre théorique de l'apprentissage en ligne contre toute séquence**. Au cadre théorique précédent on peut ajouter l'objectif d'une minimisation du critère contre toute séquence d'apprentissage possible, et comparer l'algorithme au meilleur algorithme d'apprentissage « batch » pris dans un ensemble d'apprenants de référence.
4. **Environnement changeant**. Les scénarios précédents supposent en général que l'environnement est stationnaire. Si l'environnement change au cours du temps, alors l'apprentissage doit être adaptatif. L'environnement peut être formalisé à l'aide de la distribution sur les exemples $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$ qui peut être décomposée en : $\mathsf{p}_{\mathcal{X}}\,\mathsf{p}_{\mathcal{Y}|\mathcal{X}}$, une composante décrivant la distribution des descriptions et l'autre la dépendance entre la sortie et l'entrée. De ce fait, on distingue souvent les variations qui peuvent affecter l'une, l'autre ou les deux composantes :
   - Lorsque $\mathsf{p}_{\mathcal{X}}$ varie, ce que les statisticiens appellent la distribution des variables *explicatives*, mais pas $\mathsf{p}_{\mathcal{Y}|\mathcal{X}}$, on parle de *déplacement covarié* (*covariate shift*). Comme nous l'avons vu à la section 1, l'apprentissage actif conduit naturellement au déplacement co-varié.
   - Lorsque la dépendance fonctionnelle entre l'espace d'entrée et l'espace de sortie varie : $\mathsf{p}_{\mathcal{Y}|\mathcal{X}}$, on parle de *dérive de concept* (*concept drift*).
5. **Apprentissage à horizon infini** (*long-life learning* ou *continuous learning*). Lorsque le système apprenant est censé s'adapter sur une longue durée, il faut également envisager que les tâches d'apprentissage, et les environnements (e.g. les capteurs) puissent changer. On parle alors d'*apprentissage multitâche* et d'*apprentissage par transfert*.

Nous examinons rapidement ces divers aspects dans la suite.

Ajoutons cependant d'autres distinctions. On parle souvent indifféremment d'*apprentissage en ligne* et d'*apprentissage incrémental*. Nous proposons de restreindre la signification de ce

Copyright © 2010 Eyrolles.

dernier à l'apprentissage d'un concept ou d'une compétence cible par un apprentissage progressif, éventuellement cumulatif, l'accent étant alors mis sur la séquence de sous-tâches d'apprentissage et sur la trajectoire suivie par l'apprenant. L'apprentissage en ligne est plus général au sens où le concept cible peut évoluer au cours du temps.

Par ailleurs, dans la suite, nous nous concentrons essentiellement sur l'apprentissage supervisé. Rappelons cependant que l'apprentissage non supervisé, pour des raisons de complexité calculatoire, et l'apprentissage par renforcement, par nature, sont des apprentissages incrémentaux ou en ligne suivant les cas.

2.2 Les contraintes calculatoires de l'apprentissage en ligne

Ces dernières années ont vu l'émergence dans tous les domaines de bases de données énormes et en croissance rapide (e.g. santé, biologie, télécommunications, physique des particules, Web, ...). Pour les exploiter, des méthodes de recherche de régularité, de détection de rupture ou de suivi de tendance sont nécessaires (*large-scale learning*), avec des coûts en calcul et en mémoire faibles. Idéalement, ces coûts doivent être de l'ordre de $\mathcal{O}(m)$ c'est-à-dire linéaires avec le nombre d'exemples m. De fait, afin de rendre possible un apprentissage en ligne, il faut que le coût soit constant avec chaque nouvel exemple. Il s'agit là d'une contrainte très forte sur l'apprentissage.

2.2.1 Méthodes de gradient stochastique

Les méthodes de gradient stochastique mettent à jour incrémentalement l'hypothèse courante en minimisant une approximation stochastique du gradient ne tenant compte que de l'exemple d'apprentissage courant. Bien que ces méthodes requièrent davantage d'itérations pour converger que les méthodes déterministes traditionnelles (*batch*), chaque itération est en elle-même plus simple, ne retraitant pas l'ensemble des données d'apprentissage.

Un algorithme *hors ligne* ou en *batch* minimise le risque empirique $R_m(h)$ selon :

$$h_{k+1} = h_k - \gamma_k \nabla_h R_m(h_k) = h_k - \gamma_k \frac{1}{m} \sum_{i=1}^{m} \nabla_h \ell\big(h_k(\mathbf{x}_i), y_i\big) \tag{20.3}$$

où $\nabla_h expr$ dénote le gradient de $expr$ en fonction de h, et où γ_k, le taux d'apprentissage, est > 0. Des études ont montré que $(h_k - \hat{h})^2$ converge en e^{-k}, où k dénote le $k^{\text{ème}}$ passage de l'échantillon d'apprentissage.

En revanche, une procédure de *gradient stochastique en ligne* met à jour l'hypothèse courante sur la base d'un exemple unique $(\mathbf{x}_t, y_t)$, tiré généralement aléatoirement à chaque itération.

$$h_{t+1} = h_t - \gamma_t \nabla_h \ell\big(h_t(\mathbf{x}_t), y_t\big) \tag{20.4}$$

La convergence est rapide jusque vers la zone d'un bon optimum, mais lente après, en raison des fluctuations introduites par la considération de chaque exemple.

Lorsque les données sont abondantes, redondantes ou issues de flux de données potentiellement non stationnaires, les méthodes de gradient stochastiques sont plus efficaces que les méthodes d'optimisation classiques. En effet, bien que les méthodes hors ligne (*batch*) convergent plus rapidement vers un bon minimum du risque empirique que le gradient stochastique, celui-ci est beaucoup moins coûteux en calculs. Cela signifie qu'avec les mêmes ressources, il peut traiter un échantillon d'apprentissage plus important et donc optimiser un risque empirique mieux informé.

Copyright © 2010 Eyrolles.

Les études [BB08, BL05] ont ainsi montré que des méthodes de gradient stochastique permettent d'examiner de l'ordre de $\mathcal{O}(m \log \log m)$ exemples quand une méthode hors ligne peut en traiter m. Par exemple, si $m = 10\,000$, le gradient stochastique peut traiter 22000 exemples. Même si l'optimisation en ligne est de moins bonne précision, elle s'appuie sur un risque nettement mieux informé, ce qui la rend finalement plus performante.

L'essentiel des travaux de recherche dans ce domaine portent sur le contrôle du pas d'apprentissage ou pas de gradient durant l'apprentissage.

2.2.2 Vers une analyse théorique de l'apprentissage à grande échelle

Afin de rendre compte du compromis *complexité calculatoire - précision* des algorithmes d'apprentissage, Léon Bottou et Olivier Bousquet [BB08] ont proposé d'étendre la formule classique exprimant la compromis biais-variance. L'erreur totale ε_T résulterait ainsi de trois termes :

$$\begin{aligned} \varepsilon_T \;=\;& \mathbb{E}\big[R_{\text{Réel}}(h^*) - R_{\text{Réel}}(f)\big] && \text{(erreur d'approximation)} \\ +\;& \mathbb{E}\big[R_{\text{Réel}}(\hat{h}) - R_{\text{Réel}}(h^*)\big] && \text{(erreur d'estimation)} \\ +\;& \mathbb{E}\big[R_{\text{Réel}}(\hat{h}_{\text{opt}}) - R_{\text{Réel}}(\hat{h})\big] && \text{(erreur d'optimisation)} \end{aligned} \tag{20.5}$$

où f est la fonction cible, h^* est l'hypothèse optimale dans $\mathcal{H}$, $\hat{h}$ est l'hypothèse minimisant le risque empirique, et $\hat{h}_{\text{opt}}$ est l'hypothèse trouvée en tenant compte des ressources allouée à l'optimisation du risque empirique.

Si l'erreur d'optimisation est plus élevée dans le cas du gradient stochastique en ligne, en revanche l'erreur d'estimation serait plus faible car s'appuyant sur un risque empirique mieux informé avec les mêmes ressources calculatoires. Plus généralement, Bousquet et Bottou définissent le régime de l'apprentissage à grande échelle (*large-scale learning*) comme étant celui où le terme d'erreur d'optimisation, donc les contraintes calculatoires, prennent le pas sur les deux autres termes.

Ce type d'analyse prenant en compte les ressources calculatoires est certainement à développer.

2.2.3 Algorithmes pour les flux de données

Partant de prémisses différentes, les chercheurs en bases de données ont également étudié comment traiter les flux de données à ressources constantes. Une question essentielle concerne l'établissement de « résumés » des données en taille mémoire constante, en s'adaptant au mieux, s'il y a lieu, aux variations des régularités dans le flux de données.

Comme il s'agit d'un champ de recherche important et un peu à part, nous le traitons à la section 3.

2.3 Déplacement covarié (*Covariate shift*) : $\mathbf{p}_{\mathcal{X}}$ varie

Pour diverses raisons, les données utilisées en apprentissage peuvent différer de celles qui seront rencontrées en phase de généralisation. Même si les dépendances qui lient les entrées aux sorties pour un phénomène étudié restent stationnaires, c'est-à-dire que les étiquettes des exemples restent constantes (en supposant une dépendance déterministe), la distribution des entrées peut changer au cours du temps et donc être différente entre la phase d'apprentissage et la phase de généralisation. On parle alors de *déplacement covarié*.

Par exemple, même si les caractéristiques de certaines maladies sont fixes (e.g. gastro-entérites), leur prévalence peut varier en cours d'année selon les saisons. De même, l'échantillon d'apprentissage peut être biaisé pour faciliter l'apprentissage, par exemple en rééquilibrant des classes de fréquences très différentes, ou bien en adoptant un protocole d'apprentissage actif. Dans tous

Copyright © 2010 Eyrolles.

ces cas, l'échantillon d'apprentissage ne peut plus être considéré comme résultant d'un tirage indépendant et identiquement distribué selon une vraie distribution $\mathsf{p}_{\mathcal{X}}$ également représentative de la distribution en généralisation. Il en résulte que le risque empirique classique n'est plus un substitut valable du risque réel.

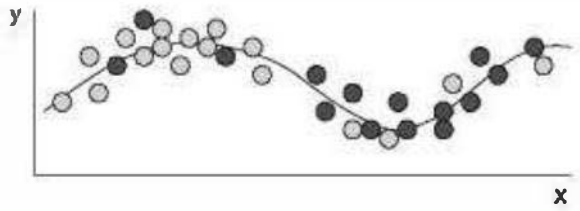


FIG. 20.3: *Dans un monde dans lequel les dépendances cibles* $\mathsf{p}(y|\mathbf{x})$ *sont stationnaires, il peut arriver que l'échantillon d'apprentissage n'ait pas la même distribution dans l'espace des entrées* $\mathsf{p}_{\mathcal{X}}$ *que l'échantillon de test caractérisée par une distribution* $\mathsf{p}_{\mathcal{X}'}$ *(respectivement correspondant à des points en gris clair et en gris foncé). Dans cette figure, l'espace des entrées est l'axe des* x*, tandis que l'espace des sorties est l'espace des* y*. (emprunté de [QCSSL09]).*

Soit $\mathsf{p}_{\mathcal{X}}$ la distribution de l'échantillon d'apprentissage et $\mathsf{p}_{\mathcal{X}'}$ la distribution en généralisation. La performance en généralisation dépend *a priori* de :

1. la performance de l'hypothèse apprise mesurée par rapport à $\mathsf{p}_{\mathcal{X}}$
2. la similarité entre les distributions $\mathsf{p}_{\mathcal{X}}$ et $\mathsf{p}_{\mathcal{X}'}$.

Une solution qui a été proposée pour récupérer la consistance de l'apprentissage (voir chapitre 21) est de pondérer les exemples d'apprentissage par leur *importance*, c'est-à-dire le rapport entre la densité en généralisation et la densité en apprentissage : $\mathsf{p}_{\mathcal{X}'}/\mathsf{p}_{\mathcal{X}}$. On obtient alors le principe inductif de *minimisation du risque empirique pondéré par l'importance* (*importance weighted* ERM) [SKM07] :

$$\hat{h}^* = \underset{h\in\mathcal{H}}{\text{Arg\_Min}}\left[\frac{1}{m}\sum_{i=1}^{m}\frac{\mathsf{p}_{\mathcal{X}'}(\mathbf{x})}{\mathsf{p}_{\mathcal{X}}(\mathbf{x})}\ell(h(\mathbf{x}_i),y_i)\right] \tag{20.6}$$

En dehors de considérations sur la stabilité de ce critère qui impose certaines modifications, ce nouveau critère inductif nécessite l'estimation de l'*importance* $\mathsf{p}_{\mathcal{X}'}/\mathsf{p}_{\mathcal{X}}$. Cependant l'approche naïve consistant à estimer d'abord les densités en apprentissage et en test, puis à calculer leur rapport est de fait généralement irréalisable puisque l'estimation de densité est notoirement difficile, spécialement dans le cas d'espace d'entrée de grande dimension. C'est pourquoi des approches plus récentes cherchent à court-circuiter ces estimations de densité pour calculer plus directement l'importance (voir par exemple [SNK+07]).

Cependant, il est important de réaliser que ces approches, aussi attrayantes qu'elles paraissent, ne s'accompagnent d'aucunes garanties quant à leur performance. De fait, il a été montré par Shai Ben-David[3] que les approches actuelles peuvent échouer totalement et que nous manquons pour le moment de concepts nous permettant de mesurer et de caractériser l'adaptation de domaine, à la fois dans le cadre du déplacement co-varié, qui est restreint, et dans un cadre plus large qui inclue la dérive de concept.

[3] Dans le workshop *Learning from non-IID Data : Theory, Algorithms and Practice* à ECML-PKDD-2009, qui peut être visionné à http://videolectures.net/ecmlpkdd09_ben_david_ttu/.

Copyright © 2010 Eyrolles.

2.4 Dérive de concept : $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}$ varie

Une des raisons de l'intérêt croissant pour l'étude de l'apprentissage en milieu non stationnaire est le développement de sources de données arrivant en flux sur de longues périodes de temps. C'est le cas par exemple lorsque l'on cherche à suivre les changements de goûts ou d'habitudes de clients, de même que pour les systèmes de recherche d'informations ou de recommandations qui doivent s'adapter aux variations d'intérêts, parfois brusques, des utilisateurs. Le système apprenant doit donc être capable de réviser continûment son modèle du monde.

Ici encore, nous nous plaçons principalement dans le cadre de l'apprentissage supervisé. Le problème d'apprentissage peut alors être défini comme suit. On suppose que les données arrivent séquentiellement une par une ou en petits lots. Dans chacun de ces lots, les données sont identiquement et indépendamment distribuées selon une distribution « locale » $\mathbf{p}_{\mathcal{X}\mathcal{Y}}(t)$. On dit qu'il y a *dérive de concept* (*concept drift*) lorsque la distribution $\mathbf{p}_{\mathcal{Y}|\mathcal{X}}$ varie avec le temps t. Le but de l'apprenant est de prédire l'étiquette de chaque nouvel exemple dans le flot de données et de minimiser sa perte cumulée, c'est-à-dire, souvent, le nombre total d'erreurs de prédiction.

Afin de pouvoir analyser l'apprentissage dans ce cadre, il faut ajouter des présupposés sur le processus sous-jacent de génération des données. Généralement, on suppose que ce processus évolue continûment, ou, du moins, que les données courantes ressemblent davantage aux données récentes qu'à des données plus anciennes. Cette sorte de *cohérence temporelle* a plusieurs expressions formelles qui traduisent que les données récentes apportent davantage d'information que les données plus anciennes.

D'un point de vue algorithmique, il existe deux grandes approches tirant parti de ce présupposé et visant à la fois à maintenir constantes les ressources calculatoires nécessaires pour traiter chaque nouvel exemple, et à s'adapter aux variations de l'environnement. L'une qui maintient une *fenêtre temporelle*, de taille fixe ou variable, des données observées le plus récemment et au-delà de laquelle les données sont oubliées. L'autre qui *pondère les données* en fonction de leur âge et de leur utilité estimée pour la tâche de prédiction.

Quelle que soit l'approche adoptée, le même compromis doit être résolu entre la tentation d'être très réactif aux changements de l'environnement, ce qui implique d'être prêt à oublier rapidement les données même peu anciennes, et la tentation d'être robuste aux variations accidentelles et également d'estimer le plus précisément possible les régularités sous-jacentes, ce qui implique au contraire de conserver le plus possible de données. La plupart des travaux portant sur la dérive de concept cherchent à optimiser ce compromis, souvent par des heuristiques, en **contrôlant la mémoire du passé**.

Il faut noter qu'une stratégie possible consiste à apprendre directement les variations qui caractérisent l'environnement plutôt que de chercher à adapter, toujours avec retard, un modèle courant. Cela modifie profondément le problème du contrôle de la mémoire et pose d'intéressants problèmes pour modifier en conséquence le critère inductif. (Voir [SKS07] pour un cadre d'apprentissage des variations appelé *tracking*. Voir également [Cor09] pour plus d'informations sur les approches en dérive de concept et en apprentissage en ligne en général).

2.5 L'apprentissage par transfert entre tâches

Dans les organismes naturels, l'apprentissage se construit toujours sur d'autres apprentissages. Cela se traduit par des effets parfois bénéfiques, parfois néfastes, mais on ne peut étudier un apprentissage isolément du passé du système apprenant. On parle de transfert (aussi *priming effect*, en sciences cognitives) lorsqu'un apprentissage influence un autre apprentissage. C'est le cas, par exemple, quand quelqu'un apprend à jouer aux échecs après avoir appris à jouer aux dames, ou apprend à jouer du piano après avoir appris à jouer de la flûte, ou encore lorsque l'on apprend

Copyright © 2010 Eyrolles.

une seconde langue. Comme l'a dit Stuart Russell dans un cours sur l'intelligence artificielle, l'apprentissage cumulatif ou l'apprentissage par transfert est peut-être le problème fondamental que doit résoudre l'apprentissage artificiel. C'est pourtant un sujet extraordinairement peu étudié par comparaison avec l'apprentissage isolé ou en une fois (*one-shot learning*) à l'exception de quelques papiers pionniers [Car97b, Sch94, TM95].

Récemment, cependant, un regain d'intérêt s'est manifesté pour ce sujet et plusieurs ateliers (*workshops*) de grandes conférences lui ont été dédiés. Il faut néanmoins noter que ce qui est appelé *apprentissage par transfert* (*transfert learning*) désigne encore essentiellement le transfert entre tâches d'induction supervisée avec les mêmes espaces d'entrée et de sortie, ce qui est loin d'épuiser toutes les situations de transfert en apprentissage. Un vocable a été inventé pour désigner ce cadre plus général, celui d'apprentissage continu (*continuous learning*, ou aussi *lifelong learning*). Par ailleurs, l'apprentissage par transfert est naturellement lié à *la question de la meilleure organisation d'une séquence d'apprentissages*, et donc à l'*apprentissage cumulatif*, voire *incrémental*, qui s'appuie sur l'apprentissage de tâches simples avant celui de tâches plus complexes. Il entretient également des liens évidents avec le raisonnement par analogie. Peut-être l'un des signes du manque de maturité de ce domaine, mais aussi de sa richesse et de sa nécessité, est la multiplicité des appellations sous lesquelles il apparaît : apprendre à apprendre (*learning to learn*), apprentissage multi-tâche (*multi-task learning*), transfert de connaissances (*knowledge transfer*), adaptation de domaine (*domain adaptation*), transfert inductif (*inductive tranfer*), méta apprentissage (*meta learning*), apprentissage incrémental (*incremental learning*), etc.

Étant donné qu'un problème d'apprentissage est défini en partie par le triplet $(\mathcal{X}, \mathcal{Y}, \mathsf{p}_{\mathcal{X}\mathcal{Y}})$, dont le dernier élément peut lui-même être décomposé en $\mathsf{p}_{\mathcal{X}} \cdot \mathsf{p}_{\mathcal{Y}|\mathcal{X}}$, ainsi que par la disponibilité d'exemples étiquetés ou non étiquetés dans chacun des problèmes, tout un ensemble de situations de transfert différentes peuvent être distinguées. Nous renvoyons le lecteur à [PY08] pour une typologie, sans doute encore incomplète et imparfaite.

L'apprentissage par transfert pose deux grandes questions :

1. Comment *mesurer le degré de relation* (*relatedness*) entre deux problèmes d'apprentissage ?
2. *Comment et quoi transférer* entre les problèmes ?

Ces deux questions sont encore compliquées par le fait qu'elles ne sont pas indépendantes. La mesure de la ressemblance dépend de ce qui peut être transféré, et ce qui peut être transféré dépend de la mesure de ressemblance.

De très nombreuses propositions ont été faites ces dernières années pour répondre à ces questions. Elles sont généralement très liées au domaine d'application étudié, et il reste à identifier des principes fondamentaux et des concepts théoriques pour commencer à maîtriser ce grand chantier.

Parmi les obstacles à une étude générale et une synthèse figurent l'absence de critères d'évaluation bien définis et, corrélativement, le manque de corpus de test permettant de comparer les différentes approches. Il y a là un champ de recherche largement ouvert et dont les résultats seront importants tant pour les applications en attente de méthodes que pour le développement conceptuel de l'apprentissage artificiel.

2.6 Quelle analyse théorique ? Le cas de l'apprentissage en ligne

Comme nous l'avons vu, il n'existe pas pour le moment de cadre théorique général permettant de situer et de caractériser les apprentissages par transfert. Cependant, dans le cas de l'apprentissage en ligne, les théoriciens ont essayé de se débarrasser de l'hypothèse fondamentale que

Copyright © 2010 Eyrolles.

les données d'apprentissage (et de test) étaient engendrées par un tirage indépendamment et identiquement distribué.

Dans l'apprentissage en ligne (*on-line learning*), on suppose que le processus se décompose en épisodes. Dans chacun de ceux-ci, un nouvel exemple est fourni à l'apprenant qui, selon le modèle étudié, doit soit prédire son étiquette, soit émettre une hypothèse. Il reçoit alors une pénalité qui est fonction de l'écart entre la réponse produite et la réponse désirée. Un cas typique est celui dans lequel chaque erreur vaut 1. On évalue l'apprentissage par le nombre d'erreurs produites par l'apprenant avant d'avoir identifié le concept cible. C'est le modèle *mistake-bound* [Blu94].

Deux problèmes liés se posent pour définir ce cadre. D'une part, de quel lien suppose-t-on l'existence entre le passé et l'avenir ? D'autre part, comment évalue-t-on la qualité d'un apprenant ?

Dans la théorie statistique de l'apprentissage, on suppose qu'une même distribution fixe régit les exemples d'apprentissage et les exemples à venir. Si on conserve cette hypothèse dans le cadre de l'apprentissage en ligne, alors il n'y a aucune différence fondamentale entre l'apprentissage hors ligne ou *batch* et l'apprentissage en ligne. La distribution sur l'espace des exemples induit une distribution sur les séquences d'apprentissage possibles, qui, par exemple, rend très peu probables les séquences dans lesquelles tous les exemples négatifs seraient fournies avant les exemples positifs.

Mais les théoriciens formés à l'informatique fondamentale sont obsédés par les études de pire cas. Ils souhaiteraient donc caractériser un apprenant par son comportement dans le cas des séquences d'apprentissage les plus défavorables. Or il est facile de montrer que si l'ordre des exemples est choisi par un adversaire tenant compte de l'état de l'apprenant, alors cet adversaire peut forcer des performances de l'ordre de celles qui seraient obtenues en répondant au hasard. Il faut donc avoir recours à des *critères de performances relatifs* et des bornes associées (*relative loss bounds*).

Principalement deux types de scénarios ont été explorés.

- Le premier s'appelle *apprentissage incrémental avec nombre d'erreurs* (*mistake-bound learning model*). Il est assez naturel. On se demande combien d'erreurs peut-on être amené à commettre dans le pire des cas (le pire concept cible et la séquence d'exemples la plus désavantageuse) pour identifier un concept cible. Plus formellement, à chaque étape, un exemple $x \in \mathcal{X} = \{0,1\}^d$ est fourni à l'algorithme qui doit prédire sa classe 0 ou 1 avant de recevoir sa véritable étiquette. L'algorithme est pénalisé pour chaque erreur de prédiction commise. Le but est d'avoir un apprenant produisant le moins d'erreurs possible. On suppose généralement que la présentation des exemples est sous le contrôle d'un adversaire. Dans ce modèle, on s'intéresse particulièrement aux algorithmes qui pour tout concept c dans un espace de concepts cible $\mathcal{C}$ et pour toute séquence d'exemples, ne font pas plus de $poly(d, taille(c))$[4] erreurs avec un temps de calcul par étape en $poly(d, taille(c))$. On dit alors que l'algorithme apprend $\mathcal{C}$ dans le modèle incrémental avec nombre d'erreurs.
- Le second est le *modèle d'apprentissage incrémental avec perte relative* (*relative loss bound model*). Il se rapporte à toute une lignée de travaux dans des domaines connexes comme l'*apprentissage de stratégies mixes optimales* dans les jeux itérés, le problème du *codage universel* en théorie de l'information, celui des *portefeuilles universels* en prédiction financière, et plus généralement le problème de la *prédiction universelle*. Dans cette approche, on s'intéresse aux propriétés de la prédiction d'une séquence (et non d'une population de séquences générée par un modèle probabiliste, ce qui mène à des espérances de pertes). La mesure de performance d'un algorithme de prédiction ne peut plus alors se faire dans l'absolu et doit

[4] C'est-à-dire une fonction polynomiale de d et de la taille de description du concept c.

Copyright © 2010 Eyrolles.

être établie par comparaison avec une population $\mathcal{F}$ de prédicteurs, que l'on appelle aussi des *experts*. On cherche alors quelle est la perte ou *regret* maximal de la stratégie d'apprentissage par rapport au meilleur expert de $\mathcal{F}$. Notons que la notion de regret renvoie aussi à ce qui est perdu par rapport à un apprenant (expert) qui aurait eu toutes les données d'un coup (apprentissage *batch*). Parfois, on parle d'apprentissage à partir d'avis d'experts (*using expert advices*). Les résultats connus sont encore parcellaires, mais il faut retenir que les pertes calculées dépendent de certaines propriétés métriques de la population d'experts $\mathcal{F}$.
L'idée est de supposer l'existence d'un ensemble de n d'experts (ou d'hypothèses). Pour chaque nouvel exemple $\mathbf{x}$, chaque expert fournit une prédiction y. L'algorithme d'apprentissage $\mathcal{A}$ n'a accès qu'à ces « avis » des experts pour former sa propre prédiction $y_{\mathcal{A}}$. Il apprend alors la bonne prédiction $y*$. On cherche à ce que l'algorithme $\mathcal{A}$ obtienne une performance presque aussi bonne que le meilleur expert dans l'ensemble. Si $L(e)$ est la perte de l'expert e, et $L(\mathcal{A})$ la perte de l'algorithme d'apprentissage, il est souvent possible de prouver :

$$L(\mathcal{A}) \leq \min_{e} L(e) + \log(n) \tag{20.7}$$

sur toutes les séquences d'apprentissage possibles.

Il faut noter le caractère nouveau de ce type d'étude. Il n'y a plus d'hypothèse selon laquelle les données sont indépendamment et identiquement distribuées (i.i.d.), et il n'y a pas de supposition sur la valeur des experts : ceux-ci peuvent être médiocres. L'inégalité (20.7) ne tient pas en probabilité seulement, elle tient toujours, quelle que soit la séquence d'apprentissage rencontrée.
Ce cadre, s'il se révèle intéressant pour toutes les applications de gestion de portefeuille par exemple, dans lesquelles il est possible de combiner les avis d'experts et de quantifier la performance à chaque épisode, est cependant limité et ne conduit pas à la conception de nouvelles méthodes d'apprentissage performantes. Sa généralité, l'apprentissage contre toute séquence d'exemples, est aussi une source de faiblesse. Si ce n'est sans doute pas le dernier mot sur ce problème, cette approche est cependant intéressante (voir [CBL06] pour la monographie la plus complète à ce jour sur ce sujet).

Un intérêt fondamental de ces travaux prenant en compte le caractère généralement incrémental des apprentissages, est qu'ils vont sans doute conduire à renouveler profondément l'approche théorique de l'apprentissage dans la mesure où l'on s'y affranchit d'hypothèses sur la distribution des exemples et en particulier sur l'hypothèse i.i.d. (distribution indépendante et identique) [AW01, Blu97, CBFH+97, CBL01, CT91, MF98].

3. Apprentissage à partir de flux de données

Ces dernières années, les progrès en matières de technologie ont permis la mesure automatique de données produites en continu dans des expériences ou des observations scientifiques (accélérateurs de particules, observations satellitaires), dans les réseaux de senseurs comportant des milliers de nœuds, ou bien liées aux quantités de méta-données disponibles sur le trafic dans les réseaux de communication ou de distribution d'énergie et dans les grilles de calcul. Ces développements apportent de grandes quantités de données qui croissent à un rythme lui-même en augmentation. On parle de *flux de données* (*data streams*). Formellement, un flux de données

Copyright © 2010 Eyrolles.

est une séquence ordonnées de *points* ou observations $\langle x_1, x_2, \ldots, x_m \rangle$ qui sont observés dans cet ordre, et qui ne peuvent l'être qu'une seule fois, ou un petit nombre de fois.

Ces données doivent être traitées et analysées pour des raisons diverses : contrôle de système, découverte de régularités ou de tendances dans les données ou détection de situations alarmantes. Cependant, la quantité massive de données généralement en jeu ne permet pas d'en envisager un stockage préalable. Il faut au contraire opérer en utilisant des ressources de calcul et de mémoire sub-linéaires par rapport à la taille des entrées[5], en cherchant malgré tout à garantir la qualité des résultats obtenus. De plus, seuls des accès mémoire linéaires sont possibles, les accès aléatoires aux données étant trop coûteux. Ces contraintes d'espace mémoire et de coût en calculs rendent nécessaire la conception de nouvelles techniques d'analyse des données, en particulier la capacité de « résumer » les données afin de laisser suffisamment d'espace mémoire pour traiter les données à venir.

Les applications et algorithmes liés à l'apprentissage en présence de flux sont en plein développement et il n'est pas possible d'en décrire le paysage, même approximativement, dans cet ouvrage. Nous nous concentrons donc sur quelques questions typiques qui permettent de mettre en évidence les difficultés nouvelles et des techniques de résolution propres à ce nouveau domaine d'applications. Il est ainsi évident que l'échantillonnage et des comptages divers sont des opérations fondamentales pour l'apprentissage. Par ailleurs, les tâches de découverte de règles d'association, de détection de rupture de tendance et de classification non supervisée (*clustering*) sont particulièrement présentes dans les applications.

3.1 Échantillonnage et comptage

Une opération fondamentale en apprentissage concerne la constitution d'un échantillonnage de données représentatif, par exemple un tirage aléatoire et uniforme dans l'univers des données. Cela est facile lorsque que l'ensemble des données est fini, par exemple lorsqu'il s'agit de tirer 100 éléments parmi 1 000, mais comment opérer lorsque le flux de données croît continuellement ? Supposons que l'on tire 100 éléments parmi les 1 000 premiers éléments du flux, que faire lorsqu'un million d'éléments arrivent en plus ? Conserver les mêmes 100 éléments au risque de ne plus du tout être représentatif du flux de données ?

Plusieurs solutions ont été proposées pour conserver un échantillon uniforme à partir d'un flux de données.

3.1.1 Algorithme du *reservoir sampling*

L'idée sous-jacente à l'algorithme du *reservoir sampling* [Vit85] remonte aux années 1980 et même sans doute avant.

Supposons que l'on doive constituer un échantillon de m éléments tirés uniformément dans un flux de données dont la taille instantanée est n, alors l'algorithme est le suivant. Tant que $n \leq m$, tirer tous les éléments du flux. Par la suite, lorsque $n > m$, tirer aléatoirement le i-ème élément du flux avec une probabilité de $1/i$ et s'il est retenu remplacer au hasard un élément de l'échantillon précédent.

Lorsque $m = 1$, il est facile de montrer que la probabilité que le i-ème élément du flux a une probabilité d'être retenu de $1/n$. En effet, cette probabilité est égale à la probabilité que le i-ème élément soit tiré multiplié par la probabilité que i survive jusqu'à la fin.

$$\frac{1}{i} \times \frac{i}{i+1} \times \frac{i+1}{i+2} \times \ldots \times \frac{n-2}{n-1} \times \frac{n-1}{n} = \frac{1}{n}$$

[5] Idéalement en complexité poly-logarithmique, c'est-à-dire en $\mathcal{O}((\log m \log n)^c)$, avec m le nombre d'éléments, n le nombre de types d'éléments, et c une constante

Copyright © 2010 Eyrolles.

Le cas de $m \neq 1$ s'obtient par m répétitions indépendantes de cet algorithme.

Dans le contexte d'applications mettant en jeu plusieurs senseurs (e.g. nœuds d'un réseau, machines d'une grille de calcul), l'inconvénient de cet algorithme est qu'il est difficile à paralléliser et à appliquer sur plusieurs flux simultanément. D'où l'invention d'un autre algorithme.

3.1.2 Algorithme du *min-wise sampling*

Pour chaque élément dans un des flux, tirer aléatoirement un nombre compris dans $[0, 1]$. Retenir les m éléments qui ont les plus petits nombres associés. Il est clair que chaque élément a la même probabilité d'être associé à l'un des m plus petits nombres. La distribution qui résulte de ce tirage est donc uniforme [NGSA04].

3.1.3 Calcul d'entropie sur les flux

L'entropie empirique d'une séquence S de caractères s'obtient en déterminant la fréquence empirique f_i de chaque caractère i et en calculant $H(S) = \sum_{i=1}^{k} \frac{f_i}{n} \log_2 \frac{f_i}{n}$, où k est le nombre de caractères différents et n la longueur de la séquence.

On utilise souvent une telle estimation d'entropie pour détecter les anomalies dans les flux de données.

Quand le nombre de types de caractères m est très grand, il faut utiliser une autre approche. Une méthode consiste à choisir aléatoirement une position i dans le flux, puis à compter le nombre r d'occurrences du caractère a_i dans la suite des n caractères du flux. La quantité $X = r \log_2 \frac{n}{r} - (r-1) \log_2 \frac{n}{r-1}$ est alors un estimateur non biaisé de $H(S)$: $E[X] = H(S)$ dont la variance est $\mathrm{Var}[X] = \mathcal{O}(\log^2 n)$. La qualité de l'estimation peut être améliorée en répétant plusieurs fois la procédure et en prenant la moyenne des résultats. En invoquant les bornes de Chebyshev, on montre que le nombre de répétitions nécessaires pour garantir une erreur bornée par ε / H^2 dans plus de 75 % des cas est $\mathrm{Var}[X]/\mathbb{E}^2[X]$, ce qui dans le cas de l'estimation de l'entropie signifie $\log^2 n / H^2(S)$. Il existe donc un problème potentiel pour le cas d'une entropie très faible.

Heureusement, l'analyse du cas d'une entropie faible montre qu'elle doit résulter d'un flux dans lequel un caractère est très largement prépondérant. Il suffit alors de l'enlever du flux pour pouvoir obtenir une estimation plus précise de $H(S)$ [CCM07].

3.1.4 Calcul de résumés sur les flux

Tous les problèmes d'estimation ne peuvent pas être résolus à l'aide d'échantillonnages. Par exemple, le calcul du nombre d'éléments distincts dans un flux requiert l'examen de tous les éléments du flux. On appelle *résumé* (*sketch* en anglais), une structure de données compacte qui résume le flux par rapport à une requête. Le fait que l'algorithme de traitement *voit* tous les éléments du flux même s'il ne les retient pas permet de calculer des résumés, éventuellement de manière approchée seulement.

3.2 Recherche de règles d'association

Un problème classique en fouille de données est celui de la recherche de *règles d'association*. Rappelons qu'il s'agit de chercher des règles de la forme $X \rightarrow y$ (un ensemble X d'items associé à un item y) dans une collection de transactions t_i, chacune d'entre elles étant un sous-ensemble d'items possibles, par exemple un ensemble de produits achetés dans un supermarché. Le *support*

Copyright © 2010 Eyrolles.

d'une règle est défini comme la fraction des transactions contenant tous les items de la règle :

$$\frac{|\{i|X \cup \{y\} \subseteq t_i\}|}{|t_i|}$$

La *confiance* d'une règle est définie comme le rapport du nombre de transactions contenant tous les membres de la règle sur le nombre de transactions contenant les items prémisses de la règle :

$$\frac{|\{i|X \cup \{y\} \subseteq t_i\}|}{|t_i|X \subseteq t_i|}$$

En général, l'objectif consiste à trouver (toutes) les règles de support et de confiance excédant des seuils spécifiques, d'où le terme de recherche de *Frequent Items Sets (FIS)*. Comme le nombre de telles règles est exponentiel par rapport au nombre d'items possibles, des stratégies de recherche spécialement conçues doivent être utilisées. Le problème est rendu encore plus difficile lorsque les transactions sont observées dans un flux et que les ressources en calcul et en mémoire sont limitées.

Le problème d'identifier les 1-itemsets fréquents (ensembles de taille 1) est déjà ardu lorsque l'ensemble des items possibles est très grand. Plusieurs approches ont été proposées. À des fins d'illustration, nous n'en évoquons qu'une ici : l'algorithme *Space-Saving* [MAA05]. Cet algorithme peut fournir un ensemble de 1-itemsets fréquents avec une erreur de moins de ε en utilisant une mémoire de taille $\mathcal{O}(\frac{1}{\varepsilon})$. Le principe consiste à maintenir un ensemble de $k = 1/\varepsilon$ items avec leur nombre d'occurrences associé. Pour chaque item, s'il a déjà été rencontré, son compteur est incrémenté de un, sinon l'algorithme remplace l'item associé au compteur le plus petit en prenant pour nombre d'occurrences la valeur de ce compteur plus un.

Une méthode pour trouver les ensembles d'items fréquents est décrite dans [MM02]. Pour chaque nouvelle transaction observée, l'ensemble de tous les sous-ensembles d'items est produit et stocké dans une structure compacte basée sur les tris. Lorsque l'espace mémoire est rempli, un algorithme d'élagage est utilisé afin de retirer les itemsets les moins fréquents. L'algorithme calcule l'erreur commise sur les nombres associés à chaque itemset. Cette technique est efficace, mais il n'existe pas, pour le moment, de borne d'erreur en pire cas sur l'espace mémoire requis pour atteindre une précision donnée.

Les recherches sur ces questions sont en plein essor et concernent aussi la recherche d'itemsets dans les séquences ordonnées et la recherche de formes séquentielles dans les flux de transactions.

3.3 Détection de ruptures

Un problème important dans le traitement des flux de données est celui de détecter les changements dans la distribution sous-jacente du processus engendrant les données. Une approche consiste à utiliser la divergence de Kullback-Leibler (ou entropie relative) pour mesurer la distance entre distributions. L'avantage de cette méthode est qu'elle est bien fondée, et qu'elle ne requiert pas de s'appuyer sur des modèles paramétrés des distributions.

L'idée est de comparer les distributions correspondant à une fenêtre temporelle de référence et à la fenêtre courante, toutes deux comprenant n points. Afin de ne pas avoir à faire d'hypothèse sur la distribution sous-jacente, la méthode décrite dans [DKVY06] propose de partitionner l'espace des entrées en sous-régions de poids (nombre d'éléments) à peu près égaux. Il est alors possible de calculer la probabilité discrète pour la fenêtre de référence $p(i)$, et de même pour la fenêtre courante $q(i)$. La distance de Kullbak-Leibler est alors calculée par :

$$D(p\|q) = \sum_i p(i) \log_2 \frac{p(i)}{q(i)}$$

Copyright © 2010 Eyrolles.

Afin de déterminer si la distance observée est statistiquement significative, une méthode de *bootstrapping* [Efr82] peut être utilisée. Ici, elle consiste à engendrer deux fenêtres virtuelles en tirant aléatoirement les points des fenêtres de référence et courante et à calculer la divergence de Kullback-Leibler résultante. Si la distance mesurée pour les vraies fenêtres dépasse la distance calculée pour des couples de fenêtres virtuelles engendrées aléatoirement plus d'un certain nombre de fois qui peut être calculé pour un certain intervalle de confiance, alors on conclut que la distribution sous-jacente a changé dans le flux.

Comme il est possible de calculer la divergence de Kullback-Leibler de manière incrémentale, l'ensemble de la procédure peut être implanté de manière efficace. D'autres approches utilisant des histogrammes ou des méthodes à noyaux sont également étudiées actuellement.

3.4 Classification non supervisé

Lorsque les données sont très mal connues, en particulier lorsqu'il n'est pas encore question de leur attribuer des étiquettes, une des premières approches pour en cerner les caractéristiques est d'en chercher une description par classification non supervisée ou *clustering*. Grossièrement, le clustering consiste à trouver une partition (généralement disjointe) des données telle que, suivant une mesure de similarité à définir, des objets similaires soient rassemblés dans une même partie (ou groupe) et que les parties soient aussi dissimilaires que possible (voir chapitre 18).

À l'exception de quelques méthodes incrémentales, les algorithmes classiques de classification non supervisée calculent la partition en s'appuyant sur la base entière des données. Cependant, les flux de données ne peuvent être appréhendés de cette manière. En effet, d'une part, le volume des données ne permet pas de les traiter toutes ensemble, d'autre part, un flux devant se considérer comme un processus non fini, évoluant avec le temps, la structure sous-jacente en termes de regroupements peut elle-même évoluer. Les algorithmes de classification non supervisée sur les flux de données doivent donc permettre de calculer une partition en fonction d'un certain empan temporel (par exemple, clusters sur une semaine, un mois, une année), ainsi que d'adapter la partition calculée en fonction des évolutions temporelles.

La classification non supervisée présente un défi dans le cas des flux de données, car il faut trouver une manière adaptée de représenter les données de manière économique, tout en préservant la capacité à représenter fidèlement la structure des données et sa possible évolution. Plusieurs approches ont été proposées depuis la fin des années 1990. La plupart reposent sur le calcul et le maintien de points représentatifs, les k-moyennes, qui sont les « centres » des clusters (voir par exemple [Gon85]). Le problème est alors de choisir le nombre adéquat de centres, et de savoir éventuellement le faire évoluer en fonction des variations de la distribution des données au cours du temps. De plus, cette méthode ne permet pas facilement de répondre à des requêtes portant sur des périodes de temps d'ampleurs différentes (quels étaient les clusters sur le dernier mois ou la dernière année ?). D'autres méthodes, telles que *CluStream* [AHWY03], défendent l'idée d'une hiérarchie de représentations. Les données sont d'abord résumées par des micro-clusters en nombre limité, mais cependant plus important que le nombre naturel de clusters. Ces micro-clusters s'adaptent ainsi à la micro-structure des données, et maintiennent une information sur leur évolution temporelle. Ils permettent ensuite, lors d'une requête, de calculer les macro-clusters d'intérêt pour l'utilisateur.

Quelle que soit l'approche, il faut idéalement pouvoir quantifier le degré d'approximation garanti en fonction de la taille mémoire autorisée (nombre de centres, écarts-type ou seuil, statistiques temporelles, etc.) et de la dynamique du signal. C'est ce que s'efforcent de faire les auteurs des méthodes proposées dans la littérature scientifique.

Copyright © 2010 Eyrolles.

4. Apprentissage de sorties structurées

Les tâches d'apprentissage supervisé classiques concernent la prédiction de sorties « simples » : $\{-1, 1\}$ (apprentissage de concept), une valeur parmi un petit ensemble, par exemple $\{a, b, \ldots, z\}$ (apprentissage multi-classe), un réels $\mathbb{R}$ (régression). Pourtant, dans de nombreux cas, on aimerait prédire un objet structuré en sortie et non une étiquette à une dimension.

Ainsi, dans l'analyse de texte, la tâche d'étiquetage grammatical (« *part-of-speech tagging* ») consiste à associer à une portion de texte (e.g. « le chien poursuivait le chat ») la séquence d'étiquette grammaticale correspondante (e.g. `Déterminant Nom Verbe Déterminant Nom`). La tâche d'analyse (*parsing*) consiste à associer l'arbre de dérivation grammatical à un texte (figure 20.4).

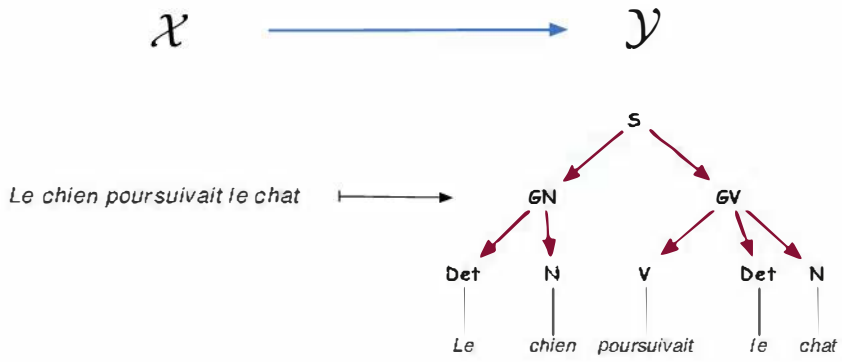


FIG. 20.4: *Analyse d'un texte en termes d'arbre de dérivations.*

De même, la tâche de prédiction de structure secondaire d'ARN consiste à associer à une séquence de nucléotides une structure secondaire présumée (figure 20.5).

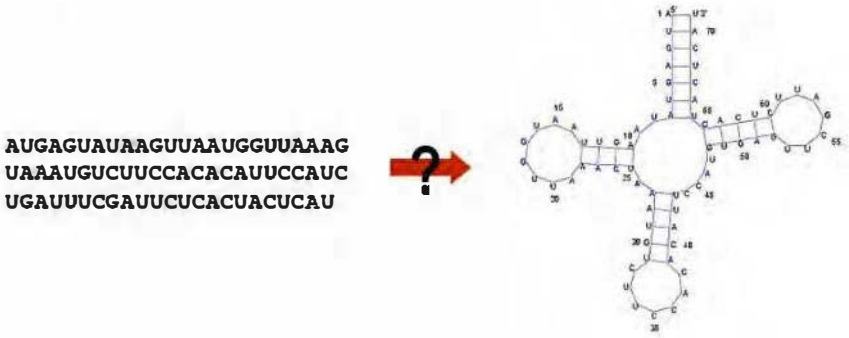


FIG. 20.5: *Prédiction de la structure secondaire associée à une séquence de nucléotides.*

Ces quelques exemples montrent que les méthodes classiques d'apprentissage supervisé ne sont pas bien adaptées à ce type de tâche. Même dans le cas de l'étiquetage grammatical, qui est le plus simple, apprendre à associer à chaque mot (ou morceau de texte) une étiquette impliquerait d'ignorer une source importante d'information, à savoir le voisinage de chaque étiquette. Ainsi, la séquence `Det Det Verbe` est impossible, mais un système qui opérerait sur chaque mot indépendamment ne pourrait tirer profit de cette connaissance. Il est donc nécessaire de considérer les relations qui lient les éléments des formes $\mathbf{y}$ de l'espace de sortie $\mathcal{Y}$.

Copyright © 2010 Eyrolles.

4.1 Les problèmes à résoudre

Soit donc la tâche d'inférer une étiquette complexe ou structurée $\mathbf{y} \in \mathcal{Y}$ à partir d'une entrée $\mathbf{x} \in \mathcal{X}$, elle-même éventuellement structurée. On suppose que l'on dispose d'un échantillon d'apprentissage $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$ engendré à partir d'une distribution inconnue $\mathbf{p}_{\mathcal{XY}}$. On cherche à estimer la fonction $f : \mathcal{X} \to \mathcal{Y}$ entre l'espace d'entrée $\mathcal{X}$ et l'espace de sortie $\mathcal{Y}$.

Trois problèmes sont alors à résoudre :

1. *L'espace de sortie est généralement immense*, constitué de toutes les structures possibles. De ce fait, il est peu probable, sinon impossible, d'avoir observé au moins un exemple d'apprentissage pour chaque sortie possible. En revanche, il existe souvent des interdépendances entre les éléments de l'espace de sortie. L'approche classique consistant à apprendre une fonction $h_y : \mathcal{X} \to \mathbb{R}$ pour chaque classe y indépendamment, puis à combiner ces fonctions pour obtenir une fonction de décision générale, n'est donc pas réalisable.
2. La *fonction de perte* doit maintenant pouvoir refléter la distance *entre structures* et non entre deux étiquettes univariées.
3. Il faut pouvoir *capturer les dépendances complexes entre les entrées et les sorties, mais aussi dans l'espace de sortie lui-même* puisque, d'une part, il existe des contraintes qui doivent informer l'apprentissage (e.g. on ne peut avoir `Det Det Verbe`), et, d'autre part, toutes les sorties ne peuvent figurer dans l'échantillon d'apprentissage et il faut donc pouvoir réaliser une sorte d'extrapolation entre sorties.

4.2 La fonction de perte

La fonction de prédiction recherchée h est évaluée en fonction d'une *fonction de perte* ℓ quantifiant une distance, en termes de coût de mauvaise décision, entre deux formes de sortie : $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Les fonctions de perte les plus courantes incluent :

- la fonction classique $0-1$ perte : $\ell(\mathbf{y}, \mathbf{y}) = 0$ et $\ell(\mathbf{y}, \mathbf{y}') = 1$ si $\mathbf{y} \neq \mathbf{y}'$;
- la *distance de Hamming* entre séquences : elle mesure la fréquence d'éléments différents entre deux séquences (de même longueur) ;
  On peut généraliser cette distance à la *distance d'édition*, qui fait intervenir un calcul de programmation dynamique pour sa détermination ;
- la *F-mesure* (pour l'analyse en dérivation grammaticale par exemple ou la recherche de documents). Elle mesure une moyenne harmonique entre la *précision* et le *rappel* (chapitre 3) :

$$F(h) = 2 \frac{precision \cdot rappel}{precision + rappel}$$

Plus généralement, les fonctions de perte entre structures opèrent par combinaison de mesures de distance entre sous-éléments des formes $\mathbf{y}$ et $\mathbf{y}'$ comparées. C'est justement ce que permettent les *fonctions noyau* entre structures développées ces dernières années (voir le chapitre 14), et une partie importante des recherches sur l'apprentissage de sorties structurées porte sur l'emploi et la conception de ces fonctions noyau (voir [BHS+07] pour un panorama de recherches récentes sur ce sujet).

4.3 Les approches

À partir du moment où les fonctions noyau font leur apparition dans le paysage conceptuel, il devient naturel d'invoquer également les **méthodes utilisant ces fonctions noyau** (voir chapitre 14).

Copyright © 2010 Eyrolles.

Rappelons très brièvement que dans ces approches, des méthodes linéaires de décision (e.g. SVM) ou d'analyse (e.g. ACP) sont utilisées grâce au passage par un espace de redescription des entrées $\Phi(\mathcal{X})$. Par exemple, dans le cas des SVM, la fonction de décision recherchée prend la forme :

$$h(\mathbf{x}) = \text{sign}\big(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle\big)$$

L'utilisation de l'astuce des noyaux permet de ne pas avoir à manipuler directement l'espace de redescription, qui n'intervient en fait que dans des produits scalaires. Ainsi, en utilisant une fonction noyau $\kappa(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$, la fonction de décision prend la forme :

$$h(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \, \kappa(\mathbf{x}, \mathbf{x}_i)$$

où les α_i sont les multiplicateurs de Lagrange apparaissant dans l'expression duale du problème d'optimisation associé au problème d'apprentissage.

Il est alors tentant d'étendre cette approche au cas de l'apprentissage de sortie structurées, en utilisant les fonctions noyau pour rendre compte des interdépendances entre sorties et/ou entre entrées et sorties. Deux grandes approches ont été proposées dans ce contexte.

1- La première, par **estimation d'un noyau de dépendance dans $\mathcal{Y}$**, consiste à traduire les similarités dans l'espace de sortie grâce à une fonction noyau $\kappa_{\mathcal{Y}}(\cdot, \cdot) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Comme nous l'avons vu au chapitre 14, cette fonction noyau induit un espace de redescription $\Phi(\mathcal{Y})$ dans un espace de Hilbert à noyau auto-reproduisant $\mathcal{H}_{\mathcal{Y}}$.

Dans cette approche, l'apprentissage consiste à apprendre une fonction $h' : \mathcal{X} \to \Phi(\mathcal{Y})$ ainsi qu'une fonction de « décodage » $h'' : \Phi(\mathcal{Y}) \to \mathcal{Y}$, de telle manière que la fonction recherchée soit donnée par $h = h'' \bullet h'$ (figure 20.6).

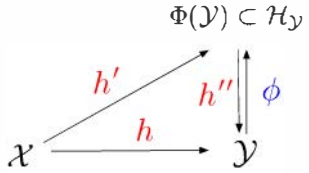


FIG. 20.6: *Fonctions entre les espaces $\mathcal{X}$, $\mathcal{Y}$ et $\Phi(\mathcal{Y})$*

Il est à noter que la détermination de la fonction de décodage est un problème en soi et n'est résolu pour le moment que pour certaines fonctions noyau.

2- La deuxième approche, plus générale, dite par **mesure de compatibilité**, considère une fonction de discrimination définie conjointement sur les espaces d'entrée et de sortie : $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, où $F(\mathbf{x}, \mathbf{y})$ peut être interprétée comme une mesure de compatibilité de $\mathbf{x}$ et $\mathbf{y}$. Une telle fonction induit la fonction hypothèse $h : \mathcal{X} \to \mathcal{Y}$ par :

$$\boxed{\mathbf{y}_i = h(\mathbf{x}) = \underset{\mathbf{y} \in \mathcal{Y}}{\text{argmax}} \, F(\mathbf{x}, \mathbf{y})} \qquad (20.8)$$

Par analogie avec les méthodes à noyau, l'espace des fonctions F est souvent restreint aux fonctions linéaires sur un espace de redescription Ψ défini sur l'espace conjoint $\mathcal{X} \times \mathcal{Y}$:

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle \qquad (20.9)$$

Copyright © 2010 Eyrolles.

L'espace de redescription Ψ doit refléter les interdépendances entre $\mathcal{X}$ et $\mathcal{Y}$ et au sein de $\mathcal{Y}$ afin de permettre la généralisation à des étiquettes non vues en apprentissage.

On cherche alors à obtenir une fonction noyau $\kappa : (\mathcal{X} \times \mathcal{Y})^2 \to \mathbb{R}$, exprimant une similarité sur l'espace conjoint $\mathcal{X} \times \mathcal{Y}$, c'est-à-dire telle que $\kappa(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}') = \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{y}, \mathbf{y}') \rangle$.

Il est alors possible d'envisager d'exprimer la fonction de compatibilité F comme :

$$F(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} \alpha_i \, \kappa(\mathbf{x}, \mathbf{y}, \mathbf{x}_i, \mathbf{y}_i) \tag{20.10}$$

En fonction de la tâche étudiée (e.g. classification hiérarchique, analyse grammaticale, ...), il est possible de décomposer la fonction noyau $\kappa(\mathbf{x}, \mathbf{y}, \mathbf{x}_i, \mathbf{y}_i)$ de diverses manières afin de rendre les calculs réalisables.

Par exemple, dans la classification hiérarchique qui consiste à ranger des éléments dans une hiérarchie de concepts, on peut utiliser la décomposition : $\kappa(\mathbf{x}, \mathbf{y}, \mathbf{x}_i, \mathbf{y}_i) = \kappa(\mathbf{x}, \mathbf{x}_i) \, \kappa(\mathbf{y}, \mathbf{y}_i)$, c'est-à-dire un découplage partiel entre les espaces d'entrée et de sortie.

Plus généralement, ces approches essaient de ré-exprimer des modèles adaptés pour rendre compte de distributions sur l'espace conjoint $\mathcal{X} \times \mathcal{Y}$ en termes de fonctions noyau. C'est le cas des modèles graphiques tels que champs markoviens aléatoires (*Markov Random Fields*), champs conditionnels aléatoires (*Conditional Random Fields*) et de modèles linéaires de mélanges de distributions (*Genaralized Linear Models in Multiple Views*).

Nous renvoyons le lecteur à l'ouvrage [BHS+07] et en particulier à son chapitre 5 pour des exemples de telles fonctions noyau et de leur décomposition. La thèse de Ulf Brefeld (2007) est également intéressante à consulter.

4.3.1 Le cas de l'apprentissage multi-instance

Il se peut que l'on veuille apprendre à reconnaître des « objets » ayant certaines propriétés quand ces propriétés sont dues à certaines conformations ou manifestations de ces objets, mais pas nécessairement à toutes. Ainsi, par exemple, on voudra apprendre que telle molécule dont on connaît la formule brute a un caractère cancérigène parce qu'au moins l'une de ses conformations est cancérigène. De même, un trousseau de clés est utile parce qu'il contient une clé au moins permettant d'ouvrir la porte. La difficulté de ce type d'apprentissage vient du fait que l'apprenant ne sait pas quelle conformation est responsable de l'étiquette de l'objet et qu'il doit cependant apprendre à prédire ces étiquettes. Il faut noter que l'apprentissage multi-instance introduit une dissymétrie entre les exemples négatifs et les exemples positifs. En effet, toutes les sous-structures (e.g. clés) d'un exemple négatif (e.g. trousseau) peuvent être étiquetées comme négatives. En revanche, un exemple positif implique qu'une des sous-structures au moins est positive, mais on ne sait pas laquelle.

Ce type d'apprentissage a été décrit par [DLLP97] dans le cadre de la reconnaissance de molécules cancérigènes. Il correspond à bien d'autres situations pratiques. Conceptuellement, cet apprentissage est intéressant à étudier car il est associé à des langages de représentation dont le pouvoir expressif doit être intermédiaire entre celui de la logique des propositions et celui de la logique des prédicats. Il est ainsi envisageable d'échapper aux limites de l'un et aux problèmes de calculabilité de l'autre. La thèse de Yann Chevaleyre [Che01] est une bonne introduction à l'apprentissage multi-instance (*multi-instance learning*).

Copyright © 2010 Eyrolles.

5. Apprentissage pour le filtrage collaboratif

Il semble que le nombre de choix qui nous sont ouverts augmente constamment. Films, livres, recettes, nouvelles du monde, autant d'ensembles dans lesquels nous devons opérer une sélection sans avoir la possibilité de considérer toutes les informations nécessaires. Comment alors choisir ? Sur une base individuelle, seule l'expérience accumulée sur la valeur de choix passés pourrait nous aider à guider nos choix futurs. Un processus d'amélioration au mieux linéaire, en supposant que les critères et ensemble de choix restent constants. Heureusement, nous ne sommes pas seuls face aux mêmes choix. Ainsi, si quelqu'un a des goûts similaires aux nôtres et qu'il a aimé tel film récent, les chances que nous aimions également ce film semblent plus grandes qu'en l'absence de cette information. Il est donc possible de tirer profit des informations disponibles sur les choix des autres agents pour induire des préférences sur nos propres choix. Avec maintenant la disponibilité d'Internet et de grandes bases de données sur les préférences des utilisateurs, il devient envisageable d'étendre à grande échelle, la notion de bouche à oreille. La formalisation et l'exploitation de cette intuition sont l'objet des travaux en filtrage collaboratif.

5.1 Les approches

Comme dans l'induction supervisée, on retrouve les approches utilisant un espace d'hypothèses et les approches opérant directement dans l'espace des utilisateurs ou des items.

Le but du filtrage collaboratif est de suggérer de nouveaux items ou de prédire l'utilité d'items inconnus pour un utilisateur donné, en se fondant sur les évaluations déjà exprimées (éventuellement implicitement) par cet utilisateur à propos d'autres items. On supposera ainsi qu'il existe une liste de m utilisateurs $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ et une liste de n items $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$. Chaque utilisateur u_l a exprimé une évaluation sur un sous-ensemble $\mathcal{I}_{u_l}$ d'items. On notera X_l^j l'évaluation de l'item i_j par l'utilisateur u_l, et $Pred_{l,j}$ la prédiction d'utilité pour l'utilisateur u_l de l'item i_j.

5.1.1 Les approches par modèle de l'utilisateur

L'approche classique pour les systèmes de recommandation consiste à construire des modèles des utilisateurs en fonction d'informations les concernant, comme leur intervalle de revenus, leur niveau culturel, leur âge, leurs habitudes culturelles, etc., et à utiliser ces modèles pour prédire la réponse d'un utilisateur ainsi caractérisé face à un objet ou item (livre, restaurant, ...).

Une technique courante est ainsi de construire un modèle régressif linéaire afin de prédire l'appréciation d'un utilisateur u_l pour un item i_j, par exemple :

$$pred(u_l, i_j) = \alpha_1(\hat{a}ge(u_l)) + \alpha_2(revenu(u_l)) + \alpha_3(intervalle\_prix(i_j)) + \ldots$$

Il est également possible d'apprendre la fonction *pred* en utilisant des SVM (voir chapitre 14) ou des méthodes de classification bayésienne.

Cependant cette approche comporte plusieurs difficultés. Outre le problème du choix du modèle (par exemple comment prendre en compte le paramètre date dans l'année dans un modèle linéaire ?), et celui du passage à l'échelle lorsque le nombre d'utilisateurs et d'items est énorme, il peut être très **difficile d'incorporer dans le modèle des variables cruciales pour l'appréciation.** Par exemple, il se peut qu'une recette de cuisine soit préférée à une autre, décrivant pourtant la confection du même plat, parce qu'elle est plus lisible, plus facile à comprendre. Mais comment un système va-t-il pouvoir mesurer cette facilité ? Faudra-t-il prévoir de décrire tous ces facteurs critiques souvent implicites : lisibilité, atmosphère, effet de mode, ... Cela devient

Copyright © 2010 Eyrolles.

vite inenvisageable. C'est pourquoi on va plutôt utiliser des *approches avec variables latentes*, non explicitées, mais permettant d'organiser les utilisateurs ou les items.

5.1.2 Les approches par similarité et plus proches voisins

On peut distinguer les *approches centrées utilisateurs* et les *approches centrées items*.

Dans l'**approche centrée utilisateurs**, un utilisateur est caractérisé par les appréciations qu'il a déjà émises à propos de certains items. Par exemple, la note qu'il a attribuée aux restaurants dans lesquels il a déjà dîné. Lorsque cet utilisateur va chercher l'appréciation probable qu'il porterait sur un nouveau restaurant, le système de recommandation va identifier les autres utilisateurs dont le profil est le plus proche de l'utilisateur en fonction des notes exprimées, et va ensuite à partir de ces « plus proches voisins », utiliser leurs notes (s'il y en a) sur le restaurant visé pour calculer une recommandation.

La correspondance entre les utilisateurs et les items se représente généralement sous la forme d'une table.

	Item 1	*Item* 2	*Item* 3	*Item* 4	...
Jean	-	2	7	8	...
Marie	4	1	-	7	...
Christian	3	8	-	4	...

On suppose que les appréciations vont de 0 à 10 (*très mauvais* à *excellent*) et que le tiret « - » représente l'absence d'évaluation. En effet, la plupart des utilisateurs ne vont noter qu'un très petit nombre d'items (par exemple Amazon vend des millions de livres, un lecteur même assidu ne peut en noter au plus que quelques milliers). Dans de nombreux cas, la matrice sera donc creuse, voire très creuse.

Supposons que nous voulions maintenant utiliser cette matrice pour calculer une similarité entre utilisateurs puis pour induire $Pred(u_i, i_j)$ pour un utilisateur u_i et un item i_j donnés. Il faut alors étudier les questions suivantes :

- Quelle mesure de similarité utiliser ?
- Combien de « voisins » prendre pour référence ?
- Quelle méthode de combinaison d'avis employer pour inférer une nouvelle évaluation ?

Mesure de similarité entre utilisateurs

Chaque utilisateur peut être considéré comme un vecteur incomplet dont nous ne connaissons que quelques composantes. Il est cependant possible de calculer une similarité entre de tels vecteurs en se restreignant aux seules composantes qu'ils ont en commun.

Si l'on suppose que les notes attribuées par les utilisateurs U_i et U_j aux items sont des variables aléatoires X_i et X_j qui suivent une distribution conjointe inconnue, il est possible de définir le coefficient de corrélation entre X_i et X_j par la *formule de Bravais-Pearson* :

$$\rho = \frac{Cov(X_i, X_j)}{\sqrt{Var(X_i)Var(X_j)}}$$

avec $Cov(X_i, X_j) = \mathbb{E}\left[(X_i - \mathbb{E}(X_i))\,(X_j - \mathbb{E}(X_j))\right]$.

Cette corrélation prend ses valeurs dans $[-1, +1]$, une valeur positive indiquant que les variables varient dans le même sens, tandis qu'une valeur négative signifie que les individus qui ont des

Copyright © 2010 Eyrolles.

scores élevés pour la première variable auront tendance à avoir des scores faibles pour la deuxième et inversement.

En disposant d'un échantillon de taille n, $(X_i^1, X_j^1), (X_i^2, X_j^2), \ldots, (X_i^n, X_j^n)$ tiré d'une distribution conjointe, la quantité :

$$r = \frac{\sum_k (X_i^k - \bar{X}_i)(X_j^k - \bar{X}_j)}{\sqrt{\sum_k (X_i^k - \bar{X}_i)^2} \sqrt{\sum_k (X_j^k - \bar{X}_j)^2}}$$

est une estimation de ρ.

EXEMPLE Calcul de la similarité

Selon cette mesure, la similarité entre *Jean* et *Marie* est :

$$r(\text{Jean, Marie}) = \frac{(2-5)(1-4)+(8-5)(7-4)}{\sqrt{(2-5)^2+(8-5)^2}\sqrt{(1-4)^2+(7-4)^2}} = \frac{(-3)(-3)+(3)(3)}{\sqrt{9+9}\sqrt{9+9}} = \frac{18}{18} = 1$$

avec $X_{\text{Jean}} = (2+8)/2 = 5$ et $X_{\text{Marie}} = (1+7)/2 = 4$ quand on prend les articles notés en commun.

Entre *Jean* et *Christian*, elle est : $r(\text{Jean, Christian}) = \frac{-12}{\sqrt{18}\cdot\sqrt{8}} = -12/(3 \cdot 2 \cdot 2) = -1$

Et entre *Marie* et *Christian* : $r(\text{Marie, Christian}) = \frac{-12}{\sqrt{18}\cdot\sqrt{14}} = -4/(\sqrt{2} \cdot \sqrt{14}) \approx -0.756$

Il apparaît ainsi que *Jean* et *Marie* sont positivement corrélés, tandis que *Jean* et *Christian*, de même que *Marie* et *Christian*, sont des paires corrélées négativement. Il faut noter que s'il peut sembler étrange que *Jean* et *Christian* soient parfaitement négativement corrélés, il faut prendre en compte leur moyenne sur les composantes 2 et 4 : à savoir 5 pour *Jean* et 6 pour *Christian*. Par rapport à ces moyennes, les notes de *Jean* pour les *items* 2 et 4 sont -3 et +3, tandis que pour *Christian* elles sont 2 et -2. Il y a bien tendance exactement inverse.

De fait, il a été empiriquement observé que la corrélation de Pearson n'est pas la mesure de corrélation la mieux adaptée pour le filtrage collaboratif, un **meilleur choix étant de prendre la corrélation de Pearson à la puissance 2.5** [BHK98].

Calcul d'une recommandation

La plupart des algorithmes de recommandations utilisent une combinaison des évaluations d'utilisateurs *proches* au sens de la mesure de similarité employée. Par exemple, l'approche la plus simple consiste à retenir les k plus proches voisins au sens de cette similarité r et de calculer une moyenne pondérée de leurs évaluations pour fournir une prédiction d'utilité.

$$Pred(u_l, i_j) = \bar{X}_l + \frac{\sum_{v=1}^{k} r(u_l, u_v)\,(X_v^j - \bar{X}_v)}{\sum_{v=1}^{k} |r(u_l, u_v)|} \qquad (20.11)$$

où $\bar{X}_l$ (resp. $\bar{X}_v$) est la moyenne des notes attribuées par l'utilisateur u_l (resp. u_v) à *tous* les items.

Copyright © 2010 Eyrolles.

EXEMPLE **Calcul d'une recommandation**

Supposons que nous voulions prédire la note que donnerait *Jean* à l'*item 1* et que nous prenions *Marie* et *Christian* comme voisins.

$$Pred(\text{Jean, item 1}) = \frac{17}{3} + \frac{(1 \cdot (4-4)) + (-1 \cdot (3-5))}{1+1} = 5.67 + 1 \approx 6.67$$

Intuitivement, puisque *Christian* n'a pas aimé l'*item 1*, et qu'il est négativement corrélé avec *Jean*, alors cela devrait amener à penser que *Jean* va plutôt aimer *item 1*, d'où une valeur accordée à *item 1* supérieure à la moyenne de *Jean*. L'évaluation de *Marie*, quant à elle, ne modifie rien car elle évalue l'*item 1* à sa moyenne : 4.

L'**approche centrée items** est duale de l'approche centrée utilisateurs. Elle consiste en effet à utiliser une *mesure de similarité entre items* pour déterminer les k items les plus similaires à l'item i_j pour lequel on cherche à calculer $Pred(u_l, i_j)$. On utilise alors une formule de pondération qui, dans le cas le plus simple, est :

$$Pred(u_l, i_j) = \bar{X}^i + \frac{\sum_{v=1}^{k} r(i_j, i_v)\,(X_l^v - \bar{X}^v)}{\sum_{v=1}^{k} |r(i_j, i_v)|} \tag{20.12}$$

EXEMPLE

Supposons que nous voulions prédire la note que donnerait *Jean* à l'*item 1* et que nous prenions *Item 2* et *Item 4* comme voisins (*Item 3* ne peut pas être voisin de *Item 1* car il ne partage aucune composante utilisateur).

En utilisant la mesure de corrélation de Pearson, on trouve que r(Item 1, Item 2) $= -1$ et r(Item 1, Item 4) $= 1$. On a alors :

$$Pred(\text{Jean, Item 1}) = \frac{4+3}{2} + \frac{(-1 \cdot (2 - 11/3)) + (1 \cdot (8 - 19/3))}{1+1} \approx 3.5 + 1.67 = 5.17$$

On observe que, en utilisant ces formules de similarité et de combinaison, le résultat n'est pas le même que dans l'approche centrée utilisateur.

5.2 Les difficultés

Les difficultés liées au filtrage collaboratif proviennent essentiellement :

- de la *dimension de la matrice* utilisateurs-items qui peut éventuellement être énorme (typiquement de l'ordre de 10^4 lignes $\times 10^5$ colonnes) ;
- du *caractère creux de cette matrice* (typiquement avec moins de 1 % d'entrées) ;
- du choix à effectuer de la *mesure de similarité*, en particulier si les données comportent des relations.

Par ailleurs, il n'est pas facile de *mesurer la performance* d'un système de filtrage-collaboratif, ce qui est nécessaire pour identifier la technique la mieux adaptée à une tâche particulière.

5.2.1 Traitement des matrices

Afin de traiter de très grandes matrices, dans lesquelles la liste des items est relativement peu variable comparée à la liste des utilisateurs, plusieurs techniques sont employées.

L'une d'elles consiste à précalculer les voisins dans l'espace des items (peu variables) afin de limiter les calculs en ligne de $pred(u_l, i_j)$. Une autre consiste à effectuer une classification non

Copyright © 2010 Eyrolles.

supervisée préalable des items (ou des utilisateurs) afin de limiter effectivement la dimensionalité du problème. Cette technique entraîne cependant souvent une dégradation des performances, mais elle n'a sans doute pas dit son dernier mot. Finalement, des techniques de réduction de dimensionalité classiques sont également utilisées en prétraitement (voir chapitres 3 et 18).

5.2.2 Mesures de similarité

Le choix de la mesure de similarité est naturellement crucial dans l'approche par voisinage. Elle doit d'abord s'appuyer sur une sélection avisée des descripteurs utilisés. Généralement, cet ensemble de descripteurs découle directement de la base de données disponible (par exemple, pour une base de films : *auteur, titre, acteurs, producteur* ...). Il se peut cependant que l'ensemble d'attributs pertinents ne soit pas facile à déterminer. Par exemple, dans le cas d'items décrits par des textes (articles, recettes ...), les mots ne doivent pas prendre tous le même poids. Il est alors fréquent de les pondérer grâce au score tf-idf (*Term Frequency* × *Inverse Document Frequency*) .

- Étant donnée la fréquence f_{ij} du terme t_i dans le document d_j, $TF_{ij} = \frac{f_{ij}}{\text{Max}_k f_{kj}}$.
- De même, étant donné n_i le nombre de documents mentionnant le terme i et N le nombre total de documents, $IDF_i = \log \frac{N}{n_i}$.
- Le score tf-idf pour le terme t_i et le document d_j est $w_{ij} = TF_{ij} \times IDF_i$.
- Le profil d'un document (item) est alors caractérisé par l'ensemble des termes de score tf-idf les plus élevés avec leur score.

Les mesures de similarité les plus populaires [HKR02] sont le coefficient de corrélation de Pearson exposé plus haut (ou sa version simplifiée cosinus) et le **coefficient de corrélation de rang de Spearman**. Celui-ci compare deux variables X_i et X_j après transformation de leurs réalisations $(X_i^1, X_i^2, \ldots, X_i^n)$ et $(X_j^1, X_j^2, \ldots, X_j^n)$ en liste de rangs $(rg_i^1, rg_i^2, \ldots, rg_i^n)$ et $(rg_j^1, rg_j^2, \ldots, rg_j^n)$ dans lesquelles rg_l^m est le rang de l'élément X_l^m dans la liste X_l ordonnée selon une certaine relation d'ordre (par exemple l'ordre de préférence de films).

Le coefficient de corrélation est basé sur la différence des rangs obtenus par les réalisations sur les deux variables selon la formule :

$$r_s = 1 - \frac{6 \sum_{m=1}^{n} D^2}{n(n^2 - 1)} \tag{20.13}$$

où D représente la différence de rang sur les deux variables pour une observation donnée.

—— EXEMPLE **Coefficient de corrélation de rang de Spearman** ——

$$\begin{aligned} r_s((1.8, 3.4, 2.5, 4.1), (6.0, 1.2, 2.2, 3.7)) &= r_s((1, 3, 2, 4), (4, 1, 2, 3)) \\ &= 1 - \frac{6\,(3^2 + 2^2 + 0 + 1^2)}{4\,(4^2 - 1)} = -0.4 \end{aligned}$$

Les deux variables sont donc négativement corrélées.

L'avantage du coefficient de Spearman est qu'il est non paramétrique, c'est-à-dire qu'il ne fait aucune présupposition sur la distribution de fréquence des variables. Il ne suppose pas non plus, à l'inverse du coefficient de Pearson, que la relation entre les variables est linéaire.

Il est possible que les données disponibles impliquent l'existence de relations au-delà du graphe bipartites des utilisateurs et des items. Ainsi, dans la figure 20.7, on suppose que les items, outre qu'ils sont liés avec les utilisateurs, appartiennent à des catégories (C_1 ou C_2).

Dans ce cas, des mesures de similarité plus sophistiquées permettent de tenir compte de la structure de graphe entre les éléments du domaine (voir section 4 du chapitre 18).

Copyright © 2010 Eyrolles.

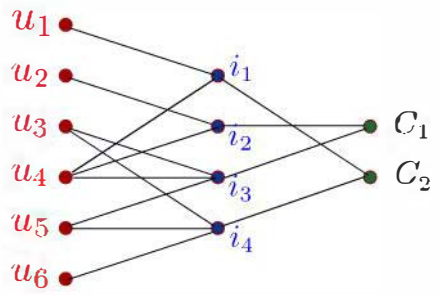


FIG. 20.7: *Graphe montrant les relations entre utilisateurs, items et catégories dans une hypothétique base de données. On voudrait pouvoir exprimer le fait que l'évaluation d'un utilisateur à propos d'un item devrait propager une information sur son évaluation potentielle d'autres items de la même catégorie. Pour cela il faut une mesure de similarité prenant en compte la structure de graphe.*

5.2.3 Évaluation des algorithmes de filtrage collaboratif

L'évaluation des algorithmes de prédiction en filtrage collaboratif procède généralement par validation croisée. On découpe les données disponibles en sous-ensembles et on utilise tous ces sous-ensembles sauf un pour l'apprentissage, et le dernier pour la mesure de performance, et on répète ce procédé plusieurs fois en changeant le sous-ensemble utilisé pour le test (voir chapitre 3).

En pratique, la démarche peut être la suivante :

1. On choisit, aléatoirement, un certain nombre d'utilisateurs, par exemple, la moitié de ceux-ci. Ces utilisateurs sont employés par l'algorithme d'apprentissage. Les autres utilisateurs constituent un ensemble test.
2. Les utilisateurs de l'ensemble test sont alors considérés un à un. Pour chaque utilisateur u, on ne fournit à l'algorithme qu'une partie des évaluations, par exemple, toutes les évaluations sauf une. On note l'utilisateur ainsi amputé de son évaluation sur l'item i par u'_i.
3. On mesure alors l'erreur commise par l'algorithme lorsqu'il tente de prédire la note accordée par l'utilisateur à l'item i : $pred(u', i)$, par rapport à la vraie évaluation x_u^i : $|pred(u', i) - x_u^i|$.
4. En calculant la moyenne sur tous les utilisateurs de l'ensemble test et tous les articles qu'ils ont notés, on obtient la mesure *All-But 1 Mean Average Error* (MAE).

La mesure MAE est la plus employée, mais |HKTR04| propose une revue des méthodes d'évaluation existantes.

Il est à noter qu'il n'est pas facile de comparer plusieurs algorithmes de recommandation entre eux, en particulier lorsque l'évaluation a lieu en ligne, c'est-à-dire alors même que le système est employé par les utilisateurs. Il exite en effet des boucles de rétroaction entre les recommandations produites par le système et les préférences exprimées par les utilisateurs. Par exemple, un item populaire sera souvent recommandé, donc souvent évalué par les utilisateurs, et son poids dans les mesures de similarité et le calcul des recommandations ultérieures sera de ce fait augmenté, ce qui biaisera le système.

Finalement, la précision des prédictions n'est pas le seul critère important d'évaluation d'un système. D'abord, les évaluations des utilisateurs varient généralement avec le temps. Une grande précision de prédiction est donc illusoire. Ensuite, le temps de calcul est souvent un facteur déterminant. Les utilisateurs ne sont pas prêts à attendre beaucoup plus qu'une seconde (trois

Copyright © 2010 Eyrolles.

secondes semble le maximum bien toléré) les recommandations. Par ailleurs, ce qui compte le plus n'est pas la précision des évaluations prédites, mais le classement et le fait qu'il n'y ait pas de faux positifs dans les items recommandés. Or, finalement, cette contrainte entre en conflit avec une autre attente des utilisateurs. En effet, il ne suffit pas de recommander aux utilisateurs ce qu'ils attendent. Supposons que l'utilisateur soit un admirateur de Jacques Brel; si le système lui recommande 10 disques de Jacques Brel, il aura l'impression que le système ne lui apporte rien. Il faut donc que le système soit capable de suggérer des « nouveautés ». Dans le cadre musical, des systèmes tels que *Music Access* ou *Music Browser*, dus à François Pachet, représentent des tentatives très intéressantes de prise en compte de la similarité entre morceaux musicaux et des goûts exprimés de l'utilisateur pour soit organiser une liste de morceaux musicaux, soit suggérer l'écoute de morceaux inconnus.

5.2.4 Limitations générales

D'autres difficultés sont à considérer :

- Le filtrage collaboratif court le risque de la sur-adaptation, ne recommandant jamais d'items qui ne figurent pas dans le profil initial de l'utilisateur. Si l'on ne peut reprocher au système de ne pas deviner les centres d'intérêt potentiellement multiples de l'utilisateur, il est cependant intéressant de chercher à élargir les recommandations offertes.
- Une difficulté liée à la précédente est celle de la construction d'un profil pour les nouveaux utilisateurs. Comment intégrer un nouvel utilisateur rapidement sans lui demander de renseigner une fiche qui peut vite sembler trop longue à remplir ?

Une approche qui peut répondre en partie à ces problèmes est d'utiliser des évaluations implicites des utilisateurs. Ainsi, au lieu de demander explicitement à un utilisateur de noter des items, il devient possible avec les technologies actuelles d'évaluer ces notes en mesurant des indices, tels que les achats effectués sur un site, le temps passé sur des pages du site, etc.

Notes historiques supplémentaires

La notion fondamentale d'**apprentissage actif** a une longue histoire en apprentissage artificiel. À notre connaissance, mais il conviendrait de mener une recherche plus approfondie, les premiers à parler d'apprentissage actif sont Simon et Lea [SL74] et Winston [Win75]. Simon et Lea argumentent ainsi que l'apprentissage artificiel est différent d'autres tâches de résolution de problème dans la mesure où l'apprentissage implique à la fois une recherche dans l'espace des hypothèses et dans l'espace des exemples. De ce fait, l'exploration de l'espace des hypothèses peut avoir un impact sur la recherche dans l'espace des exemples. Winston, quant à lui, détermine que les meilleurs exemples d'apprentissage sont les nuances critiques (*near-misses*) dans le cadre de son système ARCH basé sur une représentation des exemples et des hypothèses par réseau sémantique. Plus tard, des résultats théoriques ont montré qu'une réduction substantielle du nombre d'exemples d'apprentissage requis pouvait être obtenue par un échantillonnage bien informé [Ang88, Val84]. Le terme d'apprentissage actif a été utilisé plus récemment par Cohn et ses collègues [CAL94] pour décrire les protocoles dans lesquels c'est l'apprenant lui-même qui sélectionne les exemples potentiellement les plus informatifs.

Le terme **collaborative filtering** fut proposé par David Golberg et ses collaborateurs chez Xerox en 1992 [GNOT92].

Deux ans plus tard, en 1994, Paul Resnick du MIT (*Massachusetts Institute of Technology*)

Copyright © 2010 Eyrolles.

et ses collaborateurs de l'Université du Minnesota proposèrent l'architecture GroupLens[6] pour recommander des articles dans les newsgroup.

La librairie Amazon a popularisé le filtrage collaboratif avec « sa fonction : les utilisateurs qui ont aimé ce livre ont aussi aimé tel autre livre ». L'ingénieur responsable de ce projet, Greg Linden, a d'ailleurs un blog très intéressant (en anglais).

En 1998, Brin et Page publièrent leur algorithme PageRank et lancèrent Google. La même année, chez Microsoft, John S. Breese et ses collaborateurs publièrent un article charnière, *Empirical Analysis of Predictive Algorithms for Collaborative Filtering* [BHK98] dans lequel figure une comparaison détaillée des divers algorithmes de filtrage collaboratif.

Avant 2001, les algorithmes de filtrage collaboratif étaient soit basés sur les réseaux bayésiens, les réseaux de neurones, etc., soit sur une approche utilisateur-utilisateur. En 2001, Amazon innovait avec la publication d'un brevet introduisant le filtrage collaboratif basé sur l'article ; la même année, le groupe GroupLens publiait aussi, indépendamment, le même type d'algorithme [SKKR01].

En 2006, la compagnie Netflix a annoncé qu'elle accorderait un prix d'un million de dollars à celui qui améliorerait de 10 % leur outil de recommandation. La compagnie rend ainsi disponible un ensemble de données qui permettent de tester des systèmes. La compétition prendra fin en 2011.

Résumé

- Un *apprenant actif* a l'initiative du choix des exemples d'apprentissage. De nombreuses approches heuristiques ont été développées. Même si l'avantage en termes de nombre d'exemples requis pour apprendre n'est pas garanti, il s'observe cependant généralement. Les études théoriques doivent élargir le cadre i.i.d. classique pour en rendre compte. C'est encore un champ de recherche ouvert et... actif.
- L'*apprentissage en ligne* concerne l'apprentissage à partir de flux de données, éventuellement issus de distributions non stationnaires. Là aussi, le cadre classique i.i.d. doit être dépassé. Il y a là l'occasion d'un renouvellement important du paradigme de l'apprentissage artificiel qui s'ajoute à l'enjeu de pouvoir traiter de nombreux nouveaux domaines d'application.
- Dans le cadre de l'apprentissage à partir de *flux de données*, l'un des problèmes les plus importants est de calculer des résumés des données passées qui s'adaptent aux variations du processus génératif. Il y a là de très intéressantes questions de calcul et d'informatique qui ont produit de nombreux algorithmes très astucieux.
- L'apprentissage supervisé peut viser à calculer des *étiquettes complexes*, telles que des arbres ou des séquences. Dans ce cadre, il faut en particulier savoir mesurer des distances entre structures et rendre compte des dépendances à l'intérieur de l'espace de sortie $\mathcal{Y}$ et avec l'espace des entrées $\mathcal{X}$. Les méthodes à noyaux sont actuellement les plus étudiées pour cette tâche.

[6] voir http://www.inf.ed.ac.uk/teaching/courses/tts/papers/resnick.pdf

Copyright © 2010 Eyrolles.

- L'apprentissage pour le filtrage collaboratif introduit un nouveau type d'apprentissage. Il requiert le calcul de similarité entre items et entre utilisateurs à partir d'une matrice énorme et très creuse. L'estimation des performances est un problème en soi, demandant une métrique spécifique, rendant compte du caractère en ligne de la tâche et évitant les boucles de rétro-action trompeuses. Il s'agit d'un champ de recherche promis à un bel avenir.

Copyright © 2010 Eyrolles.

Chapitre 21

Analyse de l'induction : approfondissements et ouvertures

Le chapitre 2 a exposé les notions et les principes de base permettant d'aborder les méthodes développées en apprentissage artificiel et présentées dans cet ouvrage. Certaines d'entre elles cependant se réfèrent à une analyse théorique plus poussée, s'appuyant en particulier sur les travaux de Vapnik. Ceux-ci sont présentés ici ainsi que les approches relevant de principes de contrôle de l'espace d'hypothèses : principe de minimisation du risque structurel, théorie de l'estimation bayésienne, théorie de la régularisation, principe de compression de l'information. Le lecteur trouvera ainsi un complément d'informations utiles pour aller plus loin dans l'étude de ces méthodes.

Copyright © 2010 Eyrolles.

Sommaire

1 Généralisation de l'analyse du principe MRE 696
1.1 Le cas où $\mathcal{H}$ est infini 697
1.2 Mesurer la capacité grâce à un échantillon virtuel 698
1.3 Le cas d'un espace de fonctions réelles 700
2 Principes inductifs contrôlant l'espace des hypothèses 702
2.1 Mesures de pénalité liées aux espaces d'hypothèses considérés 704
2.1.1 Le critère AIC (*An Information Criterion* dû à Akaike) 704
2.1.2 Le critère BIC (*Bayesian Information Criterion*) 705
2.1.3 Le principe de minimisation de la longueur de description (MDLP) 706
2.1.4 Conclusion partielle 707
2.2 La théorie de la régularisation 708
2.2.1 Le principe général 708
2.2.2 La méthode des multiplicateurs de Lagrange 709
2.2.3 Le réglage du poids de la pénalisation 709
2.2.4 Applications de la méthode de régularisation 710
2.3 L'analyse bayésienne et le point de vue de la régularisation 710
2.4 La minimisation du risque structurel (SRM) 712
2.5 La marge comme mesure de capacité et outil de régularisation 713
3 Prise en compte de l'algorithme d'apprentissage dans la théorie . . 715
3.1 Sélection de modèle à partir des données (*the luckiness framework*) . . . 715
3.2 Analyse par la capacité à comprimer l'échantillon d'apprentissage 717
3.3 Les analyses par stabilité du risque empirique 717
3.4 Sélection de modèle et complexité calculatoire 718
3.5 Erreur induite par les imperfections de l'algorithme de recherche dans $\mathcal{H}$ 718
3.6 Induction guidée par gradient et phénomènes de transition de phase . . 719
4 Autres types d'analyses 720
4.1 Analyse de la physique statistique 720
4.2 Apprentissage et analyse des systèmes dynamiques 723
4.3 Discussion sur l'analyse classique : variantes et perspectives 723

Copyright © 2010 Eyrolles.

1. Généralisation de l'analyse du principe MRE

Le chapitre 2 a introduit les grands principes inductifs : principe de minimisation du risque empirique, principe de la décision bayésienne, principe de compression de l'information, et les approches dérivées de sélection de modèles qui prennent en compte l'espace d'hypothèses. Ces principes, intuitivement raisonnables, offrent-ils des garanties de performance ? Pour ce qui est du principe de minimisation du risque empirique (*Empirical Risk Minimization : ERM*), l'étude théorique rapportée dans le chapitre 2 avait montré l'intérêt d'une « analyse dans le pire cas »,

valable quelle que soit la distribution des exemples et pour n'importe quelle fonction cible. Cette analyse *PAC* (apprentissage Probablement Approximativement Correct), menée dans le cas d'espaces de fonctions indicatrices de cardinal fini et pour des fonctions de perte comptant le nombre d'erreurs de classification, a conduit à l'utilisation d'une convergence uniforme, s'appuyant sur la preuve que pour toutes les hypothèses de l'espace $\mathcal{H}$, il y a convergence du risque empirique mesuré sur l'échantillon d'apprentissage et du risque réel. Est-il possible de généraliser cette étude ? C'est ce que des travaux, dus en grande partie à Vladimir Vapnik, ont permis de réaliser. Il est important de les connaître car ils ont débouché sur des concepts de pouvoir heuristique puissant (la dimension de Vapnik-Chervonenkis) et sur de nouveaux algorithmes généraux et performants (les séparateurs à vastes marges et leurs dérivés (SVM), voir le chapitre 9).

Nous allons aborder successivement le cas d'espaces d'hypothèses indicatrices de cardinal infini, puis celui d'espaces de fonctions quelconques avec des fonctions de perte également quelconques.

1.1 Le cas où $\mathcal{H}$ est infini

Le chapitre 2 a montré que l'analyse de la consistance du principe de minimisation du risque empirique dans le cas d'un espace d'hypothèses fini ($|\mathcal{H}| < \infty$), faisait intervenir le cardinal de $\mathcal{H}$. Ainsi, le risque réel associé à une hypothèse (et donc en particulier à l'hypothèse minimisant le risque empirique : $h_S^\star$) est borné, en probabilité, par le risque empirique de cette hypothèse plus un terme qui est fonction de la richesse de $\mathcal{H}$, et que l'on nomme souvent *capacité* de $\mathcal{H}$:

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \underbrace{g(\log |\mathcal{H}|, \delta, m)}_{\text{Terme exprimant la richesse de } \mathcal{H}} \right] > 1 - \delta \quad (21.1)$$

Comment obtenir une preuve dans le cas d'un espace $\mathcal{H}$ de taille infinie ? Nous allons voir que le principe général est de *réduire l'infini au cas fini par une sorte de discrétisation.*

Il est intéressant de revenir à l'essence du problème de la prédiction à partir d'un échantillon de données d'apprentissage. Idéalement, l'espace des fonctions hypothèses considérées coïnciderait avec les régularités pouvant régir le monde. Il ne resterait plus qu'à obtenir quelques observations sur le monde pour déterminer la bonne régularité. Une faible variation de l'échantillon d'apprentissage ne devrait pas changer fondamentalement l'hypothèse faite sur le monde. Le cas inverse est celui dans lequel de nombreuses hypothèses s'accordent bien aux données d'apprentissage mais ne produisent pas du tout les mêmes prédictions sur le reste des points de $\mathcal{X}$ (voir figure 21.1). En un sens, étant donné l'espace de fonctions $\mathcal{H}$, la donnée des points d'apprentissage ne fournit aucune information sur la valeur des autres points, non vus, de $\mathcal{X}$. Toute prédiction est alors éminemment fragile et il n'est plus possible de garantir une quelconque cohérence entre la performance en apprentissage et la performance en généralisation. C'est l'écart maximal possible entre ces deux performances que cherche à borner la théorie statistique de l'apprentissage.

Il faudrait donc pouvoir évaluer la variabilité des prédictions des hypothèses qui s'accordent bien aux données d'apprentissage afin de mesurer le potentiel de différence de prédiction. C'est ce que cherchent à faire les différentes mesures de *capacité* de l'espace $\mathcal{H}$. Dans une autre optique, les approches par *régularisation* cherchent à limiter la possibilité de grande variabilité des prédictions possibles.

Différentes mesures ont été proposées pour estimer la capacité d'adaptation de $\mathcal{H}$ en fonction d'un échantillon de données. Nous citons rapidement ici quelques grandes familles d'approches, renvoyant le lecteur à l'abondante littérature sur ce sujet (voir par exemple [AB96, BvLR, Her02, MS, STC04, SC08, Vap95, Vid03] pour ne citer que des livres).

Copyright © 2010 Eyrolles.

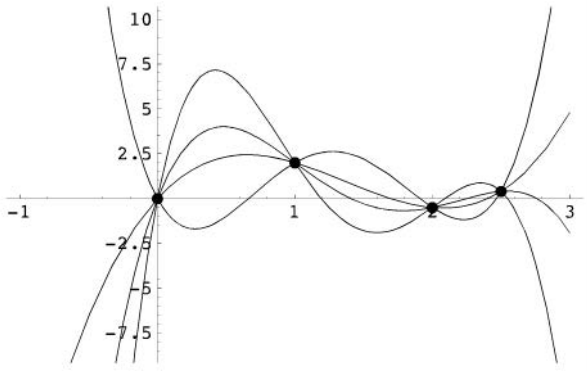


FIG. 21.1: *Par plusieurs points passent une infinité de polynômes (ici, seuls quatre ont été tracés). Lequel doit-on choisir pour interpoler à des données inconnues ?*

1.2 Mesurer la capacité grâce à un échantillon virtuel.

Vapnik et Chervonenkis ont proposé de ramener le cas d'un espace d'hypothèses infini au cas fini en introduisant un échantillon virtuel (ou échantillon « fantôme ») sur lequel on mesure la variabilité des étiquettes qui peuvent être produites par l'ensemble des hypothèses de $\mathcal{H}$. Cela ne s'applique directement qu'au cas de la classification, et spécialement de la classification binaire dans laquelle les étiquettes ne peuvent prendre que les valeurs -1 ou +1.

L'idée est que, même si $\mathcal{H}$ contient un nombre infini d'hypothèses, le nombre d'étiquetages différents que peuvent induire les hypothèses de $\mathcal{H}$ est fini. Par exemple, sur un échantillon de m points, il ne peut y avoir que 2^m étiquetages binaires possibles. Il s'agit alors de vérifier que la croissance du nombre d'étiquetages, appelé *coefficient de pulvérisation* (*shattering coefficient*) et noté $\mathcal{N}(\mathcal{H}, m)$, ne reste pas exponentielle lorsque la taille m de l'échantillon virtuel croît. En effet, intuitivement, en se rapportant au cas où $|\mathcal{H}|$ est fini, on voit que le terme équivalent à $\log|\mathcal{H}|$ dans les équations 2.61 ou 2.65 croîtrait sans limite, ne permettant pas de borner l'écart entre risque empirique et risque réel.

Théorème 21.1

Une ***condition nécessaire et suffisante*** *pour que le principe MRE soit consistant pour l'espace d'hypothèses $\mathcal{H}$ est que :*

$$\frac{\log\{\mathcal{N}(\mathcal{H}, m)\}}{m} \underset{m\to\infty}{\longrightarrow} 0 \tag{21.2}$$

Dans ce cas en effet, la probabilité suivante tend vers 0 :

$$P^m\left(\sup_{h\in\mathcal{H}} |R_{\text{Réel}}(h) - R_{\text{Emp}}(h)| > \varepsilon\right) \leq 2\mathcal{N}(\mathcal{H}, 2m)\exp\left(-m\varepsilon^2/4\right) \tag{21.3}$$

Et en utilisant cette inégalité pour borner, en probabilité, le risque réel, on obtient, avec une probabilité d'au moins $1-\delta$, et pour n'importe quelle fonction $h \in \mathcal{H}$:

$$R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \sqrt{\frac{4}{m}\left(\log\{2\mathcal{N}(\mathcal{H}, 2m)\} - \log\{\delta\}\right)} \tag{21.4}$$

Il suffit pour cela d'égaler le terme de droite de l'équation 21.3 avec $\delta > 0$.

Copyright © 2010 Eyrolles.

La dimension de Vapnik-Chervonenkis

L'un des inconvénients du coefficient de pulvérisation est qu'il est très difficile à évaluer. La dimension de Vapnik-Chervonenkis permet de caractériser par un seul nombre le comportement de la croissance du coefficient de pulvérisation. Dans le cas de la classification binaire, la *dimension de Vapnik-Chervonenkis* est égale à la taille du plus grand échantillon de points $\{\mathbf{x}_i\}_{i=1\ldots d_{VC}}$ tel que l'on puisse trouver une hypothèse $h \in \mathcal{H}$ pouvant s'accorder à chaque étiquetage possible de ces d_{VC} points (c'est-à-dire que $\mathcal{H}$ permet de réaliser les $2^{d_{VC}}$ étiquetages possibles). Si $d_{VC} = \infty$, alors l'induction devient impossible, c'est-à-dire qu'il ne peut plus y avoir de garantie sur la performance en généralisation.

Théorème 21.2

*Une **condition nécessaire et suffisante** pour que le principe MRE soit consistant pour l'espace d'hypothèses $\mathcal{H}$ est que $d_{VC}(\mathcal{H}) < \infty$.*

Nous renvoyons à la section 10 des annexes pour le détail de l'analyse de Vapnik.

Une propriété importante du coefficient de pulvérisation et de la dimension de Vapnik-Chervonenkis est qu'ils ne dépendent pas de la distribution sous-jacente des données $\mathbf{p}_{\mathcal{X}\mathcal{Y}}$, ni de la dimension de $\mathcal{X}$. Ils ne dépendent que de la classe d'hypothèses $\mathcal{H}$. C'est à la fois une force, par la généralité du résultat, et une faiblesse car les bornes obtenues sont très larges, valables dans le pire des cas. Une connaissance plus précise de la distribution des données permettrait d'obtenir des bornes plus serrées, et, souvent, plus réalistes.

La complexité de Rademacher

Soit $\sigma_1, \sigma_2, \ldots$ une collection de variables aléatoires indépendantes prenant leur valeur dans $\{-1, +1\}$ avec une probabilité de 0.5. On parle de *variables de Rademacher*.

Définition 21.1 (Complexité de Rademacher)

La complexité de Rademacher *d'une classe de fonctions indicatrices $\mathcal{H}$ est définie par :*

$$\mathcal{R}(\mathcal{H}) = \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i \, h(\mathbf{x}_i)\right] \tag{21.5}$$

L'expression de la complexité de Rademacher[1] peut paraître compliquée, mais son interprétation est en fait simple et intuitive. Le terme de somme $\sum_{i=1}^{m} \sigma_i \, h(\mathbf{x}_i)$ mesure de fait la corrélation entre les variables σ_i et les étiquettes $h(\mathbf{x}_i)$. Plus la corrélation est élevée, plus cette somme prend une grande valeur. Le sup correspond à l'idée que l'on recherche la fonction h dans $\mathcal{H}$ qui est la mieux corrélée aux variables σ_i. Finalement, on prend l'espérance de cette corrélation maximale sur l'ensemble des valeurs possibles des variables σ_i et des points $\mathbf{x}_i$. Cela signifie que l'on cherche à quel point on peut trouver une hypothèse de $\mathcal{H}$ qui s'accorde aux σ_i quelles que soient leurs valeurs, et quels que soient les points dans $\mathcal{X}$. Si la complexité de Rademacher est élevée, cela signifie que l'espace des hypothèses $\mathcal{H}$ est prêt à s'adapter à tout étiquetage de n'importe quelle collection de points $\mathbf{x}_i$.

La complexité de Rademacher mesure en un certain sens la capacité de $\mathcal{H}$ à rendre compte de données bruitées. Il est clair que l'on veut contrôler cette capacité. De fait, on a un théorème bornant en probabilité, le risque empirique pour n'importe quelle hypothèse $h \in \mathcal{H}$:

[1] Prononcer Rademaker.

Copyright © 2010 Eyrolles.

Théorème 21.3

Pour tout $\delta > 0$, on a avec une probabilité d'au moins $1 - \delta$:

$$\forall h \in \mathcal{H} : \quad R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + 2\mathcal{R}(\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{2m}} \tag{21.6}$$

on a également :

$$\forall h \in \mathcal{H} : \quad R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + 2\mathcal{R}(\mathcal{H}) + \sqrt{\frac{2\log(2/\delta)}{m}} \tag{21.7}$$

La borne de l'équation 21.7 est particulièrement intéressante car elle utilise une estimation empirique $\mathcal{R}_{\text{Emp}}(\mathcal{H})$ de la complexité de Rademacher, mesurée sur un échantillon d'apprentissage. Elle est donc assez facilement calculable, et, de plus, elle permet ainsi d'introduire une information sur la distribution $\mathbf{p}_{\mathcal{X}}$ des données.

La complexité de Rademacher, et, encore plus, la complexité empirique, mieux informée, permettent d'obtenir des bornes beaucoup plus serrées que les mesures de capacité classiques vues précédemment.

L'astuce de la symétrisation

Notre but n'est pas, dans ce chapitre, d'entrer dans les détails techniques des démonstrations utilisées pour prouver des convergences uniformes et obtenir des bornes de généralisation. Il peut cependant être éclairant de connaître les grandes idées utilisées dans ces preuves. La symétrisation est l'une d'elles.

Vapnik et Chervonenkis ont eu recours à un échantillon virtuel pour réduire le calcul de la capacité d'espaces d'hypothèses infinis à un calcul fini. La symétrisation va plus loin.

D'une part, elle relie la déviation de la moyenne empirique (le risque empirique) à l'espérance (le risque réel) à la *déviation de deux moyennes empiriques* sur deux échantillons indépendants et de même taille. D'autre part, elle relie cette dernière déviation à la probabilité d'obtenir une complexité de Rademacher élevée par permutation des exemples entre les échantillons.

Le principe est toujours de borner la queue d'une distribution correspondant à la probabilité qu'une hypothèse apparemment bonne sur les données (et donc pouvant être sélectionnée par un algorithme guidé par la minimisation du risque empirique) soit de fait « mauvaise », c'est-à-dire encourant un risque réel supérieur au risque empirique plus une certaine déviation. Cette idée est sous-jacente à la plupart des analyses décrites dans ce chapitre.

Les détails à propos de cette technique de preuve peuvent être trouvés dans de nombreux ouvrages et publications, par exemple dans [HW02, Her02, SS02].

1.3 Le cas d'un espace de fonctions réelles

Lorsque $\mathcal{H}$ est un espace de fonctions indicatrices (prenant leurs valeurs dans $\{-1, 1\}$), $|\mathcal{H}_{|\mathcal{S}}|$, l'espace des fonctions hypothèses réduites à l'espace de départ $\mathcal{S}$, est fini pour tout ensemble fini de points $\mathcal{S}$. Lorsque $\mathcal{H}$ est un espace de fonctions définies sur $\mathbb{R}$, même pour $\mathcal{S}$ fini, $|\mathcal{H}_{|\mathcal{S}}|$ peut être infini. Cependant, nous avons seulement besoin de pouvoir assurer que l'ensemble $|\mathcal{H}_{|\mathcal{S}}|$ ne contienne pas trop d'éléments qui *prennent des valeurs trop différentes*. Au lieu de la cardinalité de $|\mathcal{H}_{|\mathcal{S}}|$, on utilise alors la notion de couverture pour mesurer cette diversité.

Copyright © 2010 Eyrolles.

Les nombres de couverture

Les *nombres de couverture* (*covering numbers*) jouent le rôle du coefficient de pulvérisation dans le cas de la régression pour lequel h prend ses valeurs dans $\mathbb{R}$ et non dans $\{-1, 1\}$. L'idée est d'estimer la taille $\mathcal{N}$ d'un échantillon de fonctions « représentatives » permettant d'approcher un nombre infini de fonctions hypothèses.

En effet, en notant $\mathcal{H}_{|\mathcal{S}}$ les fonctions hypothèses réduites à l'espace de départ $\mathcal{S}$, la fonction de croissance $\Pi(\mathcal{H})$ correspond à :

$$\Pi(\mathcal{H}) = \max_{\mathcal{S}_m} \{|\mathcal{H}_{|\mathcal{S}_m}| \ : \ \mathcal{S}_m \subseteq \mathcal{X}\} \tag{21.8}$$

L'*ϵ-couverture* d'un ensemble S dans X est un ensemble de points de X tels que l'union de toutes les ϵ-boules autour de ces points contiennent S.

Il faut d'abord définir une métrique entre fonctions. Ici, cette métrique est aléatoire car elle dépend de l'échantillon de points sur laquelle on « projette » ces fonctions. Couramment, les métriques utilisées sont :

$$d_m(f, f') = \frac{1}{m}|\{f(\mathbf{x}_i) \neq f'(\mathbf{x}_i) \ : \ i = 1, \ldots, m\}| \tag{21.9}$$

ou bien :

$$d_m^{\infty}(f, f') = \max_{1 \leq i \leq m} |f(\mathbf{x}_i) - f'(\mathbf{x}_i)| \tag{21.10}$$

qui correspond à la norme l^{∞}.

Il faut bien noter que c'est une distance *entre vecteurs*, ici le vecteur de points induits par l'échantillon $\mathcal{S}_m$.

On dit alors qu'un ensemble de fonctions $f_1, f_2, \ldots, f_N$ couvre l'espace de fonctions $\mathcal{H}$ avec un rayon de ε si :

$$\mathcal{H} \subset \cup_{i=1}^{N} B(f_i, \varepsilon) \tag{21.11}$$

où $B(f_i, \varepsilon)$ est la boule de rayon ε autour de la fonction f_i. Ce qui signifie que pour toute fonction $h \in \mathcal{H}$, il existe une fonction f_i de la couverture telle que : $d_m(h, f_i) \leq \varepsilon$.

On peut alors définir le nombre de couverture empirique $\mathcal{N}(\mathcal{H}, \varepsilon, \mathcal{S}_m)$:

Définition 21.2 (Nombre de couverture empirique)

Le nombre de couverture empirique $\mathcal{N}(\mathcal{H}, \varepsilon, \mathcal{S}_m)$ *défini pour un rayon* ε, *pour un échantillon de taille* m *et pour une distance* d_m *est la taille minimale d'un ensemble de fonctions permettant de couvrir* $\mathcal{H}$ *à l'échelle* ε.

Le nombre de couverture découle de la notion précédente en éliminant la dépendance sur un échantillon de points particuliers, c'est-à-dire en prenant le supremum sur tous les échantillons possibles de taille m. On peut alors définir le nombre de couverture $\mathcal{N}^{\infty}(\mathcal{H}, \varepsilon, m)$:

Définition 21.3 (Nombre de couverture)

Le nombre de couverture $\mathcal{N}^{\infty}(\mathcal{H}, \varepsilon, m)$ *défini pour un rayon* ε *et pour une distance* d_m *est :*

$$\mathcal{N}^{\infty}(\mathcal{H}, \varepsilon, m) = \sup_{\mathcal{S}_m} \mathcal{N}(\mathcal{H}, \varepsilon, \mathcal{S}_m) \tag{21.12}$$

On notera que, contrairement au concept de coefficient de pulvérisation ou de dimension de Vapnik-Chervonenkis, cette mesure de capacité de $\mathcal{H}$ est dépendante d'une échelle, notée ici ε. Le nombre de couverture mesure la richesse de la classe de fonctions $\mathcal{H}$ à l'échelle ε.

Copyright © 2010 Eyrolles.

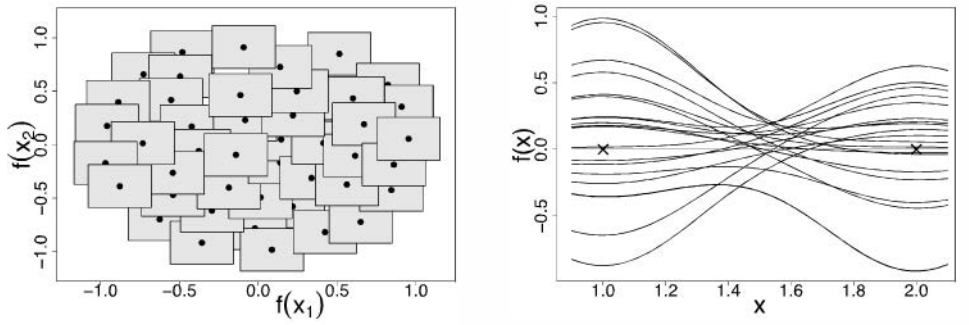


FIG. 21.2: *À gauche : 20 fonctions à valeurs réelles sur* $\mathbb{R}$*, et deux points d'apprentissage* x_1 *et* x_2 *(croix). À droite : une couverture (pas la plus petite) de l'espace* $\mathcal{F}$ *des fonctions. Dans le cas présent où* $m = 2$ *(deux points d'apprentissage), chaque fonction* $f \in \mathcal{F}$ *est caractérisée par deux nombres* $f(x_1)$ *et* $f(x_2)$*, et peut donc être représentée par un point du plan. Chaque point noir correspond à une fonction* $\hat{f}$ *de la couverture* $\mathcal{F}_\gamma((x_1,x_2))$ *et les carrés en grisés de côté* $2\,\gamma$ *correspondent aux fonctions couvertes (figure aimablement fournie par Ralf Herbrich, tiré de son livre [Her02], p.142).*

Un résultat typique de convergence uniforme utilisant la mesure du nombre de couverture a la forme :

$$P^m\left(\sup_{h\in\mathcal{H}} |R_{\text{Réel}}(h) - R_{\text{Emp}}(h)| > \varepsilon\right) \;\leq\; c_1(m)\,\mathcal{N}^{\infty}(\mathcal{H},\varepsilon,m)\,\exp(-\varepsilon^{\beta}m/c_2) \tag{21.13}$$

où $c_1(m)$, c_2 et β dépendent des caractéristiques de la tâche d'apprentissage.

À l'instar du coefficient de pulvérisation, le nombre de couverture est très difficile à évaluer. De la même manière qu'avec la dimension de Vapnik-Chervonenkis, on cherche donc à avoir recours à un nombre permettant de caractériser le comportement du nombre de couverture lorsque la taille m de l'échantillon d'apprentissage croît. Plusieurs quantités ont été proposées, telle que la *pseudo-dimension* due à Pollard, la *dimension de Vapnik-Chervonenkis pour les fonctions réelles* ou sa généralisation dépendante de l'échelle : la *fat-shattering dimension*. Nous renvoyons le lecteur intéressé à [AB96] ou à [Her02], entre autres nombreuses publications, pour avoir plus de détails sur ces mesures.

2. Principes inductifs contrôlant l'espace des hypothèses

L'analyse a montré que le critère de minimisation du risque empirique ne peut être employé sans précaution, il est nécessaire de contrôler l'espace des hypothèses considéré par le système apprenant. Sous l'influence des statisticiens, on parle souvent de *sélection automatique de modèle*.

Le mot « modèle » est particulièrement polysémique en apprentissage artificiel. Il peut désigner en effet :

1. *L'espace des fonctions hypothèses* considéré par l'algorithme d'apprentissage à un moment donné.
2. *Une famille d'algorithmes d'apprentissage* paramétrée par un ou plusieurs hyper-paramètres.

Copyright © 2010 Eyrolles.

3. *Le résultat de l'apprentissage* (la fonction apprise) qui est effectivement un modèle du monde.

Le problème de la sélection de modèle concerne le choix de l'espace d'hypothèses permettant de minimiser l'erreur totale, c'est-à-dire l'erreur d'approximation plus l'erreur d'estimation. Cette optimisation peut s'opérer « à la main », ou bien de manière automatique, grâce à des critères ou des techniques particulières. C'est de cela qu'il est question dans cette section.

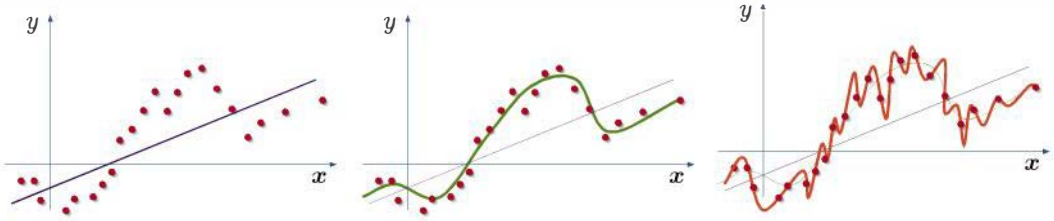


FIG. 21.3: *Trois hypothèses rendant compte d'une série de points (x_i, y_i). La première hypothèse est choisie dans l'espace des droites du plan, la deuxième dans un espace de fonctions plus riche (e.g. des polynômes de degré 5 au plus) et la troisième dans un espace encore plus riche.*

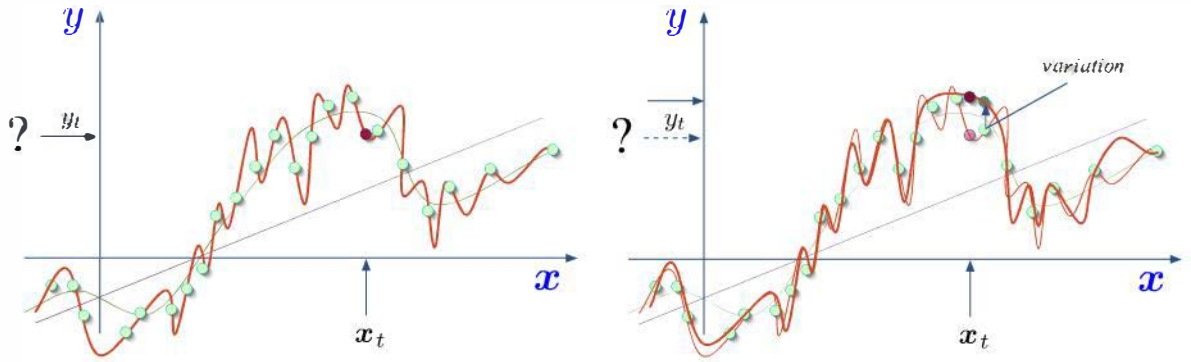


FIG. 21.4: *Le problème de l'induction vise, in fine, à pouvoir faire des prédictions pour des valeurs x_t non vues en apprentissage. Si l'hypothèse retenue après apprentissage est issue d'un espace de fonction trop riche, elle dépend trop de l'échantillon d'apprentissage et cela peut conduire à de grandes variations dans les prédictions.*

La figure 21.3, à propos d'une tâche de régression, montre comment des hypothèses différentes vont être trouvées en fonction de la richesse de l'espace des fonctions $\mathcal{H}$ considéré. Plus cet espace est riche et plus faible sera l'erreur empirique de la meilleure hypothèse. Malheureusement, la prédiction de la valeur y_t pour un exemple x_t non vu en apprentissage va devenir de plus en plus dépendante de l'échantillon d'apprentissage (figure 21.4). L'erreur d'approximation décroît au détriment de l'erreur d'estimation qui augmente. La question de la sélection de modèle consiste à choisir le bon espace $\mathcal{H}$ ou bien la meilleure régularité pour l'hypothèse sélectionnée $\hat{h}$.

Nous commençons par décrire des mesures de pénalité proposées pour corriger le degré d'adaptation aux données par la capacité de l'espace des hypothèses. Ces mesures peuvent être utilisées dans une procédure de sélection manuelle. D'autres procédures réalisent un ajustement automatique de l'espace des hypothèses.

Copyright © 2010 Eyrolles.

Finalement, il faut souligner qu'une approche très employée pour opérer le choix de l'espace des hypothèses est celle de la *validation croisée* ou celle du *bootstrap*. Ces approches sont décrites dans le chapitre 3.

2.1 Mesures de pénalité liées aux espaces d'hypothèses considérés

En statistique, le problème de la sélection de modèle est connu depuis longtemps, au moins de manière empirique. Dans les années 70, la question de la correction du biais introduit par le principe du maximum de vraisemblance a été posée explicitement et divers *critères d'information* ont été introduits pour pénaliser le sur-apprentissage possible avec les modèles plus complexes. On peut les distinguer en fonction des présupposés sur lesquels ils se fondent. Deux approches, en particulier, ont été développées. La première, incluant le critère AIC, est fondée sur la distance de Kullback-Leibler entre la densité vraie et la densité estimée, la seconde, incluant le critère BIC, sur l'analyse bayésienne.

2.1.1 Le critère AIC (*An Information Criterion* dû à Akaike)

Solomon Kullback (1907-1994) et Richard Leibler (1914-2003), deux cryptanalystes américains, ont publié en 1951 [KL51] un article devenu célèbre montrant comment mesurer une information liée au concept de statistique suffisante de Fisher. L'information dite de Kullback-Leibler est une quantité fondamentale en science avec des racines remontant au concept d'entropie de Boltzmann (1844-1906).

Soit f la densité de probabilité vraie ; f n'est pas paramétrée puisqu'il ne s'agit pas d'un modèle mais de la réalité. Soit h le modèle utilisé pour approcher cette densité. L'information de Kullback-Leibler $I_{\mathrm{KL}}(f,h)$, ou $d_{\mathrm{KL}}(f,h)$ pour *divergence de Kullback-Leibler*, est définie comme la perte d'information lorsque l'on utilise h au lieu de f. Pour les fonctions continues, elle est définie comme :

$$d_{\mathrm{KL}}(f,h) = \int_{\mathcal{X}} f(\mathbf{x}) \log\left(\frac{f(\mathbf{x})}{h_\theta(\mathbf{x})}\right) d\mathbf{x} \tag{21.14}$$

où l'on a souligné la dépendance de h sur un vecteur de paramètres θ. Il est à noter que cette mesure n'est pas symétrique (malgré le nom de distance de Kullback-Leibler qui lui est parfois donnée). Il s'agit bien d'une perte d'information qui est donc « dirigée » de la densité estimée à la densité vraie.

Selon cette mesure, la meilleure hypothèse est celle qui perd le moins d'information par rapport aux autres hypothèses dans l'espace des hypothèses considéré. Il faut donc minimiser $d_{\mathrm{KL}}(f,h)$ par rapport à h indexé par θ.

Le problème de la sélection de modèle (c'est-à-dire d'espace d'hypothèses) est cependant différent. D'une part, on ne connaît pas la densité vraie f, d'autre part, on ne connaît pas la valeur de h (plus précisément $h^\star_\theta$, la meilleure hypothèse dans l'espace des hypothèses indexé par θ), puisque celle-ci doit être estimée en fonction des données. On est donc amené à comparer des modèles (espaces d'hypothèses) sur la base d'estimations de la meilleure hypothèse dans chacun de ces modèles, estimations qui conduisent à des incertitudes. On est donc amené à minimiser une espérance d'information de Kullback-Leibler, plutôt qu'une information de Kullback-Leibler connue.

On peut ré-écrire l'equation 21.14 en :

$$\begin{aligned} d_{\mathrm{KL}}(f,h) &= \int_{\mathcal{X}} f(\mathbf{x}) \log(f(\mathbf{x}))\, d\mathbf{x} - \int_{\mathcal{X}} f(\mathbf{x}) \log h_\theta(\mathbf{x})\, d\mathbf{x} \\ &= C - \mathbb{E}_f[\log(h_\theta(\mathbf{x}))] \end{aligned} \tag{21.15}$$

Copyright © 2010 Eyrolles.

Ainsi, c'est seulement l'espérance d'information relative $\mathbb{E}_f[\log(h_\theta(\mathbf{x}))]$ qui doit être évaluée pour chaque modèle.

Akaike [Aka73, Aka74, Aka81, Aka83, Aka85] a montré qu'il fallait estimer

$$\mathbb{E}_y \mathbb{E}_{\mathbf{x}}[\log(h_{\hat{\theta}(\mathcal{S})}(\mathbf{x}))] \tag{21.16}$$

où l'espérance interne est juste $\mathbb{E}_f[\log(h_\theta(\mathbf{x}))]$ avec θ remplacée par son estimation par maximum de vraisemblance à partir des données $\mathcal{S}$.

En étudiant cette équation, Akaike a trouvé une relation entre l'information de Kullback-Leibler et la théorie de la vraisemblance. Ainsi, la log-vraisemblance maximale conduit à une estimation biaisée de $\mathbb{E}_y \mathbb{E}_{\mathbf{x}}[\log(h_{\hat{\theta}(\mathcal{S})}(\mathbf{x}))]$, mais ce biais est de l'ordre de K le nombre de paramètres à estimer dans le modèle utilisé. Cette observation fondamentale permet d'obtenir un estimateur presque non biaisé de $\mathbb{E}_y \mathbb{E}_{\mathbf{x}}[\log(h_{\hat{\theta}(\mathcal{S})}(\mathbf{x}))]$ pour des échantillons grands, qui est : $\log(\mathcal{L}(\hat{\theta}|\mathcal{S})) - K$.

Il est ainsi possible d'unifier l'évaluation de la qualité de l'estimation (ici par maximum de vraisemblance ou par minimisation du risque empirique) et la sélection de modèle.

Définition 21.4 (Critère d'information de Akaike, AIC)

$$AIC = -2\log(\mathcal{L}(\hat{\theta}|\mathcal{S})) + 2K \tag{21.17}$$

Les coefficients 2 étant présents pour des raisons historiques.

En pratique, le critère d'information de Akaike doit être utilisé pour trier les modèles possibles à partir des données $\mathcal{S}$ disponibles. On retient le modèle le meilleur selon ce classement. Il ne faut cependant pas oublier que l'estimation de la meilleure hypothèse dans une classe d'hypothèses (un modèle) est incertaine, et, en toute rigueur, il faudrait tenir compte de la variance associée dans la sélection de modèle.

Si les modèles considérés n'incluent pas la vraie densité f, le critère AIC tend à choisir des modèles de plus en plus complexes avec la taille de l'échantillon d'apprentissage $\mathcal{S}$ et donc à favoriser le sur-apprentissage. Cette observation a conduit à des pénalisations de second ordre (voir par exemple [BA04] et les références citées pour plus de détails).

2.1.2 Le critère BIC (*Bayesian Information Criterion*)

Le critère BIC [Sch78] peut être dérivé d'une analyse bayésienne. Soit un échantillon de données $\mathcal{S}$ et un ensemble de modèles $\{\mathcal{H}_i\}$ dont les paramètres associés sont $\boldsymbol{\theta}_i$. Selon la formulation bayésienne, on compare les modèles via la probabilité *a posteriori* $\mathsf{P}(\mathcal{H}|\mathcal{S})$ du modèle $\mathcal{H}$, ce qui requiert la connaissance de la distribution *a priori* $\mathsf{P}_{\mathcal{H}}$ sur les modèles. On a alors :

$$\mathsf{P}(\mathcal{H}|\mathcal{S}) \propto \mathsf{p}(\mathcal{S}|\mathcal{H})\,\mathsf{P}_{\mathcal{H}} \quad \text{avec :} \quad \mathsf{p}(\mathcal{S}|\mathcal{H}) = \int \mathsf{p}(\mathcal{S}|\mathcal{H},\boldsymbol{\theta})\,\mathsf{p}(\boldsymbol{\theta})\,d\boldsymbol{\theta}$$

où θ est la valeur possible du vecteur de paramètres associé à une hypothèse du modèle $\mathcal{H}$. Nous utilisons ici la terminologie des statistiques, car le critère BIC vient de ce domaine, mais le praticien de l'apprentissage artificiel peut aussi penser à θ comme à une hypothèse $h \in \mathcal{H}$ paramétrée par θ. Chaque espace d'hypothèses $\{\mathcal{H}_i\}$ ou modèle implique un vecteur de paramètre associé $\boldsymbol{\theta}_i$.

La comparaison des modèles $\mathcal{H}_i$ et $\mathcal{H}_j$ peut se faire par le calcul du rapport $\mathsf{p}(\mathcal{S}|\mathcal{H}_i)/\mathsf{p}(\mathcal{S}|\mathcal{H}_j)$ aussi appelé *facteur de Bayes*.

Copyright © 2010 Eyrolles.

En supposant que $\mathbf{p}(\theta|\mathcal{S})$ suit approximativement une loi normale de moyenne $\tilde{\theta}$ et de matrice de covariance Σ, on peut estimer $\tilde{\theta}$ par maximum de vraisemblance θ_{MAP} et Σ par $\mathbf{A}$ l'inverse de l'Hessien de $-\log \mathbf{p}(\theta_{\text{MAP}}|\mathcal{S})$ (puisque c'est la matrice de covariance pour une densité normale). On a alors :

$$\log \mathbf{p}(\mathcal{S}) \simeq \log \mathbf{p}(\mathcal{S}|\theta_{\text{MAP}}) + \underbrace{\log \mathbf{p}(\theta_{\text{MAP}}) + \frac{m}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{A}|}_{\text{facteur d'Occam}} \tag{21.18}$$

En supposant que la loi normale sur les paramètres est large (densité de probabilité *a priori* diffuse) et que la matrice hessienne est de rang complet, alors l'équation 21.18 peut se simplifier en :

$$\log \mathbf{p}(\mathcal{S}) \simeq \log \mathbf{p}(\mathcal{S}|\theta_{\text{MAP}}) - \frac{1}{2}K \log m \tag{21.19}$$

où K est le nombre de paramètres de θ et m la taille de l'échantillon d'apprentissage $\mathcal{S}$.

Il est apparent que ce critère BIC pénalise davantage la complexité des modèles que le critère AIC.

Le critère BIC suppose que le vrai modèle (ce qui est une abstraction) appartient aux modèles considérés. Lorsque l'échantillon de données est relativement réduit, l'utilisation du critère BIC conduit à du sous-apprentissage.

2.1.3 Le principe de minimisation de la longueur de description (MDLP)

Un principe inductif intuitif consiste à préférer parmi les hypothèses capables d'expliquer les données d'apprentissage celle qui est la plus simple, c'est-à-dire dont l'expression est la plus économique. Cela correspond au principe du rasoir d'Occam (voir la page 124). Il repose sur deux idées fondamentales. La première selon laquelle apprendre quelque chose à partir de données signifie identifier des régularités sous-jacentes. La seconde que l'identification de régularités permet de comprimer l'expression des données. On en conclut donc que plus il est possible de comprimer l'expression de données (sans perdre d'information), mieux on connaît ces données. Une troisième idée relève de l'induction, à savoir que lorsque l'on sait quelque chose à propos d'un échantillon de données, il est possible d'utiliser cette connaissance pour prédire de futures données. Ainsi, la compression et la prédiction, c'est-à-dire l'induction, semblent intrinsèquement liées.

—— Exemple **Induction d'automate** ——

Supposons que nous étudiions un phénomène caractérisé par des séquences d'observations : 0 avec une étiquette '+' ou '-' fournie par un oracle. Nous décidons d'en rendre compte à l'aide d'un automate déterministe à états finis (*DFA : Deterministic Finite state Automaton*). La simplicité (ou plutôt sa complexité) d'un automate sera mesurée par son nombre d'états.

Les séquences suivantes sont positives, ce qui correspond au fait qu'elles sont reconnues par l'automate recherché : 0, 000, 00010, 000000000 ; les séquences suivantes sont négatives, donc rejetées par l'automate : ϵ, 00, 0000, 000000.

Il existe une infinité d'automates cohérents avec ces séquences. La figure 21.5(a) montre l'automate trivial qui code simplement ces séquences. La figure 21.5(b) montre l'automate le plus simple permettant de rendre compte de ces données. Lequel des deux devons-nous préférer pour prédire l'étiquette de nouvelles séquences d'observations? Notre intuition nous souffle que le second est meilleur. Avons-nous raison?

Nous avons vu au chapitre 2 que le principe de minimisation de description minimale (MDLP) se traduit par :

Copyright © 2010 Eyrolles.

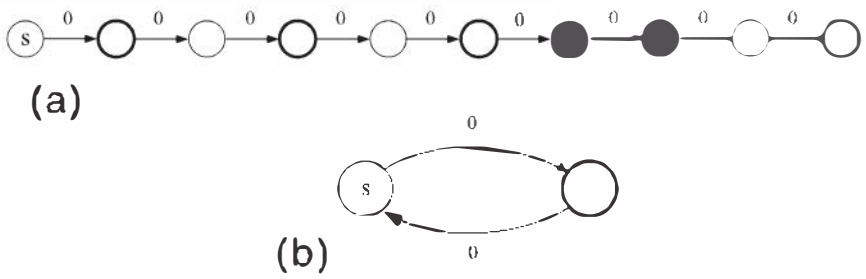


FIG. 21.5: En (a) l'automate trivialement cohérent avec les données. En (b) l'automate le plus simple cohérent avec les données.

Définition 21.1 (Principe de longueur minimale de description, *MDLP*)

La meilleure théorie, ou hypothèse, ou le meilleur modèle, rendant compte d'un échantillon d'apprentissage minimise la somme de deux quantités :

1. *la longueur, mesurée en bits, de la* description de la théorie *;*
2. *la longueur, mesurée en bits, de la* description des données *lorsqu'elles sont* décrites à l'aide de la théorie.

Formellement, cela signifie que, selon ce principe, l'hypothèse optimale $h^\star$ vérifie :

$$h^\star = \underset{h\in\mathcal{H}}{\mathrm{ArgMin}}\{K(h) + K(\mathcal{S}_m|h)\} \tag{21.20}$$

où $K(h)$ mesure la longueur de description de h, et $K(\mathcal{S}_m|h)$ mesure la longueur de description des données $\mathcal{S}_m$ en utilisant l'hypothèse h (la théorie) pour les coder.

2.1.4 Conclusion partielle

Doit-on conclure des résultats qui précèdent qu'il faut toujours mieux choisir une hypothèse succincte pour rendre compte de données ? La réponse est non. Plusieurs études se voulant provocatrices ont d'ailleurs montré qu'une hypothèse succincte pouvait se révéler moins performante qu'une hypothèse aussi bonne sur les données d'apprentissage, mais plus complexe. De fait, la préférence pour la simplicité des hypothèses s'assimile à un biais *a priori*, qui peut, ou non, être approprié. Pourquoi alors est-ce un biais naturel chez les êtres humains et que l'on trouve souvent satisfaisant ?

La réponse à cette question comporte au moins deux volets. Le premier est qu'à côté du pouvoir prédictif d'une hypothèse ou d'une théorie, nous recherchons souvent son caractère explicatif et donc compréhensible. Une hypothèse s'accordant parfaitement aux données, mais compliquée, est souvent moins satisfaisante qu'une hypothèse moins parfaite mais intelligible. À partir du moment où l'on parle d'intelligibilité, il faudrait aussi faire intervenir le reste des connaissances préalables dans lesquelles s'inscrit la nouvelle connaissance apprise. Les théorèmes de compression ne disent évidemment rien sur cet aspect des choses.

Le deuxième volet nous ramène au sens profond des théorèmes de pertinence du principe *ERM*. Pourquoi en effet une hypothèse simple serait meilleure en prédiction qu'une hypothèse plus complexe s'accordant aussi bien aux données ? Rien dans les théorèmes de Vapnik ne permet de l'expliquer. Rien, sauf ceci. Que la classe des hypothèses simples à exprimer dans un langage, dans tout langage, est forcément restreinte, quel que soit le langage utilisé. Si donc l'on trouve

Copyright © 2010 Eyrolles.

une hypothèse « simple » qui s'accorde bien aux données d'apprentissage, c'est que, dans un espace $\mathcal{H}$ limité, on a trouvé une bonne hypothèse au sens du risque empirique. Les théorèmes de Vapnik, qui prennent en compte la richesse de l'espace des hypothèses, affirment alors que la probabilité est grande que cette hypothèse se comporte bien à l'avenir. La simplicité d'une hypothèse est relative au langage utilisé pour l'exprimer, mais ce qui compte vraiment c'est la richesse de la classe des hypothèses à laquelle elle appartient. Si par chance on trouve une bonne hypothèse dans une classe restreinte, alors l'espoir est grand qu'elle soit bonne en général. C'est ce que Judea Pearl avait déjà remarqué dans un article de 1978 [Pea78] injustement oublié et qui préemptait bien des travaux ultérieurs sur ce sujet.

2.2 La théorie de la régularisation

Nous avons vu que l'analyse du problème de l'induction montre que le principe naïf de minimisation du risque empirique (*ERM*) est insuffisant et qu'il faut l'amender pour tenir compte de la « capacité » de l'espace d'hypothèses $\mathcal{H}$ à décrire des fonctions quelconques. La théorie de la régularisation prescrit le même remède mais en partant d'un souci différent.

D'abord, il faut noter que l'induction d'une fonction f à partir d'un échantillon $\mathcal{S}$ de données en nombre limité peut être vue comme un *problème inverse*. Le problème direct correspondrait à chercher l'image inconnue u_i d'une valeur x_i par une fonction f connue. C'est généralement un problème simple. Le problème inverse consiste à chercher une fonction f inconnue qui rende compte des couples de valeurs $\mathcal{S} = \{\mathbf{z}_1 = (\mathrm{x}_1, u_1), ..., \mathbf{z}_m = (\mathrm{x}_m, u_m)\}$. Il s'agit, selon les mathématiciens, d'un *problème mal posé*. Rappelons qu'un problème bien posé (au sens de Hadamard) présente les propriétés suivantes :

1. *Existence* : pour toute fonction cible f ayant engendré les données, il existe une fonction h dans l'espace $\mathcal{H}$ de fonctions considéré solution du problème inverse.
2. *Unicité* : la solution h est unique.
3. *Continuité* : la solution h dépend continûment de f.

L'induction est un problème mal posé dans la mesure où la solution obtenue par minimisation du risque empirique n'est en général pas unique. Par exemple, il existe une infinité de polynômes (de degré suffisamment grand) passant par un nombre fixe de points, et annulant donc le risque empirique (mesuré par exemple par un écart quadratique) (voir figure 21.1). Il peut également être mal posé dans le cas où les données sont bruitées ou engendrées par un mécanisme si complexe que la classe $\mathcal{H}$ des hypothèses ne permet pas de trouver une fonction rendant parfaitement compte des données (non existence).

La théorie de la régularisation, initiée par Tikhonov et Arsenin (1977) et développée en particulier par Poggio et Girosi [GJP95] dans le cadre de l'apprentissage, consiste à transformer le problème de l'induction en un problème bien posé (et si possible facile à résoudre effectivement) en utilisant des connaissances *a priori* pour contraindre l'espace des hypothèses.

2.2.1 Le principe général

La théorie de la régularisation suggère de transformer le problème mal posé de la recherche d'une hypothèse h rendant compte des données d'apprentissage $\mathcal{S}$ en un problème de recherche d'une hypothèse h soumise à des contraintes additionnelles, c'est-à-dire :

1. minimisant le risque empirique $R_{\mathrm{Emp}}(h)$;
2. et soumis à une contrainte $\Phi(f) \leq \mu$ où Φ est une fonctionnelle incorporant des connaissances *a priori* sur la solution recherchée et μ est un paramètre.

Copyright © 2010 Eyrolles.

Sous des conditions assez larges sur Φ, il est possible de montrer que la solution au problème de minimisation ci-dessus existe, est unique et dépend continûment des données.

La question est alors de savoir *quelle forme de connaissance* a priori, *traduite par la fonctionnelle* Φ, *il faut imposer?*

Intuitivement, l'idée est encore une fois de contraindre l'espace des hypothèses en pénalisant la classe des hypothèses si complexes qu'elles peuvent s'accorder à n'importe quel échantillon de données de taille m. Deux approches sont utilisées :

1. L'*approche paramétrique* dans laquelle on cherche à contraindre le nombre de paramètres des hypothèses. Par exemple, on cherchera des réseaux connexionnistes à petit nombre de connexions.
2. L'*approche non paramétrique* qui caractérise la complexité d'une fonction hypothèse h par une mesure de sa dynamique dans le domaine fréquentiel. On parle alors de la régularité de la fonction (le terme anglais utilisé est *smoothness*). En un sens, il s'agit de préférer les fonctions les plus « lisses » parmi toutes celles qui rendent compte des données (par exemple, dans le cas des polynômes, on favorisera les polynômes de degré plus faible).

2.2.2 La méthode des multiplicateurs de Lagrange

La minimisation des problèmes sous contrainte du type :

$$\begin{cases} \text{minimiser une fonctionnelle : } F(h) \\ \text{sous la contrainte : } G(h) \leq \mu \end{cases} \tag{21.21}$$

se résout en faisant appel à la méthode des multiplicateurs de Lagrange.

On construit d'abord le problème d'optimisation sous-contraint :

$$R_{\text{Reg}}(h) \;=\; F(h) \;+\; \lambda\, G(h) \tag{21.22}$$

où λ est un paramètre portant le nom de multiplicateur de Lagrange. Le point selle de cette fonctionnelle fournit alors la solution du problème d'optimisation. La fonctionnelle doit être minimisée en fonction de h et maximisée en fonction de λ.

La solution canonique du problème d'optimisation ci-dessus passe par deux étapes :

1. Pour chaque valeur de $\lambda > 0$, trouver le minimum $m(\lambda)$ du problème sous-contraint (21.22)
2. Trouver la valeur $\lambda = \hat{\lambda}$ pour laquelle : $G(m(\lambda)) = \mu$.
   Le minimum du problème contraint est : $m(\hat{\lambda})$.

2.2.3 Le réglage du poids de la pénalisation

Cependant, on applique généralement la théorie de la régularisation différemment. On choisit une fonctionnelle de pénalisation $G(h)$ correspondant à des connaissances *a priori* sur les hypothèses souhaitables, et compatible avec les techniques efficaces d'optimisation. On cherche alors une hypothèse h minimisant :

$$R_{\text{Reg}}(h) \;=\; R_{\text{Emp}}(h) \;+\; \lambda\, G(h) \tag{21.23}$$

λ agit comme un paramètre de contrôle permettant de régler le compromis entre la fidélité aux données d'apprentissage mesurée par le premier terme de l'équation, et la régularité de la solution h mesurée par le second terme. Cela définit un nouveau principe inductif : **choisir une hypothèse h minimisant le risque pénalisé.**

Copyright © 2010 Eyrolles.

Normalement, tant la fonctionnelle de pénalisation G que le paramètre λ devraient traduire des connaissances *a priori*, c'est-à-dire *externes* aux données d'apprentissage. Cependant, en général, il est difficile d'avoir suffisamment d'information *a priori* sur la bonne classe d'hypothèses, et l'on ajuste donc également en partie le terme de pénalisation, ce qui permet de corriger une mauvaise estimation *a priori* mais constitue aussi une erreur en terme d'induction. On risque en effet à nouveau un phénomène de surapprentissage. Le plus souvent, la fonctionnelle G est choisie *a priori*, tandis que le paramètre λ est ajusté en fonction des données d'apprentissage.

Une procédure classique consiste à sélectionner plusieurs classes d'hypothèses (ou plusieurs valeurs du paramètre de contrôle λ) et à réaliser l'apprentissage dans chacune des classes. On sélectionne alors la classe (ou la valeur de λ) qui minimise le risque mesuré sur l'échantillon de test. Il faut cependant être conscient que le risque mesuré sur l'échantillon de test est nécessairement optimiste, puisque cet échantillon est juge et partie, ayant servi à la fois à estimer les performances des classes de modèles et à en sélectionner une. Pour obtenir une estimation non biaisée du risque réel, il faut donc avoir recours à un échantillon de données n'ayant servi ni à l'apprentissage, ni au test. Ce troisième type d'échantillon est appelé *échantillon de validation*.

2.2.4 Applications de la méthode de régularisation

Le principe de régularisation a été employé pour justifier, souvent *a posteriori*, des méthodes visant à contrôler l'application du principe de minimisation du risque empirique dont la tendance au surapprentissage était connue des praticiens. Nous évoquons rapidement ici certaines de ces méthodes renvoyant le lecteur à des ouvrages spécialisés pour plus de détails (par exemple [Bis95]).

- *Pénalisation des poids dans un réseau connexionniste* (*weight decay*). Cette méthode consiste à pénaliser les réseaux connexionnistes dont les poids des connexions sont élevés. Il faut en effet des poids importants pour qu'un réseau puisse présenter une forte dynamique apte à s'adapter à n'importe quel échantillon de données. Le terme de régularisation est de la forme :

$$\lambda \sum_{\text{connexions } j} w_j^2$$

- *Règle d'arrêt avant terme* (*early stopping rule*). Cette méthode est appropriée dans le cadre de l'apprentissage par optimisation itérative du risque telle qu'elle s'effectue dans les réseaux connexionnistes, et en particulier les perceptrons multicouches. Elle consiste à stopper le processus d'optimisation avant qu'il y ait convergence vers l'optimum local. La justification invoquée est que le nombre effectif de degrés de liberté du réseau s'accroîtrait au fur et à mesure de l'apprentissage. En le stoppant avant terme, on éviterait l'obtention d'un modèle trop complexe.
- *Apprentissage avec bruit.* Son principe est de bruiter les données d'apprentissage durant l'apprentissage (éventuellement en les bruitant différemment à chaque passe). Intuitivement, il s'agit de rendre plus difficile l'obtention d'un modèle suffisamment complexe pour s'accorder aux données d'apprentissage. On espère ainsi que l'apprentissage résultera dans un modèle rendant compte des régularités profondes plutôt que des détails. [Bis95] (pp.346-349) montre que l'on peut effectivement exhiber une procédure de régularisation équivalente, ce qui justifie l'approche.

2.3 L'analyse bayésienne et le point de vue de la régularisation

On peut définir pour chaque modèle, c'est-à-dire pour chaque classe d'hypothèses $\mathcal{H}_i$, sa densité de probabilité *a priori* : $\mathbf{p}_{\mathcal{H}_i}(\cdot)$ définie sur l'ensemble des échantillons d'apprentissage possibles $\mathcal{S}$.

Copyright © 2010 Eyrolles.

Les classes d'hypothèses les plus spécifiques (souvent caractérisées par peu de paramètres ou par des paramètres à domaine de validité limité) sont associées à des densités de probabilité nulles ou quasi nulles sur une grande partie de l'espace des échantillons, mais sont en revanche bien adaptées pour des régions étroites de cet espace. Par contraste, les modèles très versatiles peuvent représenter une large région de l'espace des échantillons. Lorsque l'on normalise ces densités de probabilité, on obtient ainsi une figure telle que 21.6, dans laquelle la classe d'hypothèses $\mathcal{H}_3$ est supposée plus spécifique, et donc moins riche en possibilité de description, donc moins complexe, que les classes $\mathcal{H}_2$ (de complexité intermédiaire) ou $\mathcal{H}_1$ de plus grande complexité.

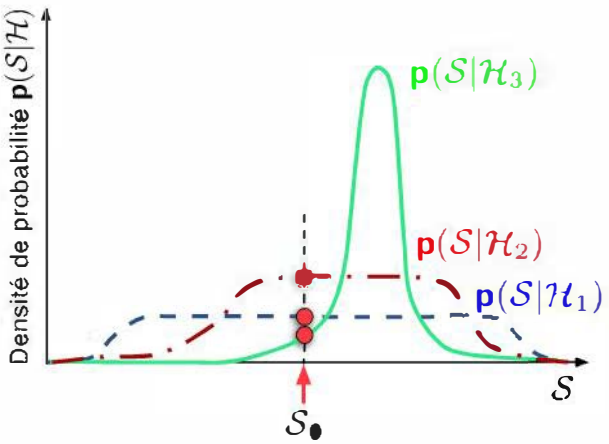


FIG. 21.6: *Sélection de modèle par l'approche bayésienne. On considère trois classes de modèles $\mathcal{H}_1$, $\mathcal{H}_2$ et $\mathcal{H}_3$. Pour chacun d'entre eux on a représenté la probabilité a posteriori des jeux de données $\mathcal{S}$ possibles. Pour un jeu de données particulier $\mathcal{S}_0$, on détermine alors sa probabilité a posteriori pour les trois modèles. Ici, c'est le modèle de complexité intermédiaire qui l'emporte.*

La probabilité des hypothèses après observation de l'échantillon d'apprentissage $\mathcal{S}$ est donnée par la formule :

$$\mathsf{p}(h_j|\mathcal{S}) = \frac{\mathsf{p}(\mathcal{S}|h_j)\,\mathsf{p}_{\mathcal{H}_i}(h_j)}{\mathsf{p}(\mathcal{S})} \tag{21.24}$$

dans laquelle on souligne l'appartenance d'une hypothèse h_j à une classes d'hypothèses $\mathcal{H}_i$.

Pour un échantillon de données particulier, disons $\mathcal{S}_0$, on peut ainsi calculer la probabilité *a posteriori* de chaque classe d'hypothèses. Dans le cas de la figure 21.6 et pour le jeu de données $\mathcal{S}_0$, c'est la classe d'hypothèses $\mathcal{H}_2$ qui fournit l'hypothèse la plus probable *a posteriori*. Sans qu'il soit facile de le démontrer, en moyenne sur les échantillons d'apprentissage $\mathcal{S}$, ce sont les classes d'hypothèses de pouvoir explicatif intermédiaire qui sont les meilleures. Celles qui sont très simples ayant par nature un domaine de validité étroit, et celles qui sont complexes étant en général dominées par des classes moins généralistes.

L'approche bayésienne conduit ainsi naturellement à prendre en compte une certaine forme de complexité des classes d'hypothèses et est donc une forme de principe d'induction avec régularisation.

Copyright © 2010 Eyrolles.

2.4 La minimisation du risque structurel (SRM)

Au terme de l'étude sur la cohérence du principe de minimisation du risque empirique, c'est-à-dire du lien entre le risque empirique minimal, le risque réel associé et le risque réel optimal, Vapnik et ses collègues ont obtenu des bornes, valables pour toute fonction cible et pour toute distribution des exemples, sous la forme générale (pour le cas du problème de classification) :

$$R_{\text{Réel}}(h_S^{\star d}) \leq R_{\text{Emp}}(h_S^{\star d}) + \Phi(\frac{m}{d_{\mathcal{H}_d}}) \tag{21.25}$$

où $d_{\mathcal{H}_d}$, la dimension de Vapnik-Chervonenkis, mesure la capacité de l'espace d'hypothèses $\mathcal{H}_d$. $h_S^{\star d}$ est l'hypothèse de risque empirique minimal dans l'espace $\mathcal{H}_d$ et Φ est une fonction de la taille de l'échantillon d'apprentissage m et de la dimension de Vapnik-Chervonenkis qui mesure un intervalle de confiance. On a une équation générale similaire pour le cas de la régression :

$$R_{\text{Réel}}(h_S^{\star d}) \leq \frac{R_{\text{Emp}}(h_S^{\star d})}{\Phi'(\frac{m}{d_{\mathcal{H}_d}})} \tag{21.26}$$

Sur cette base, Vapnik propose un nouveau principe inductif baptisé *principe de minimisation du risque structurel* (*Structural Risk Minimization : SRM*) reposant sur deux idées (voir figure 21.7).

1. La capacité d'une classe d'hypothèses $\mathcal{H}$ va être mesurée par sa dimension de Vapnik-Chervonenkis $d_{\mathcal{H}}$. Il est alors possible de définir une *structure* sur les classes d'hypothèses[2] consistant en une séquence enchâssée de classes d'hypothèses $\mathcal{H}_d$: $\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \ldots \subseteq \mathcal{H}_d \subseteq \ldots$ telle que chaque classe $\mathcal{H}_d$ est de dimension de Vapnik-Chervonenkis $d_{\mathcal{H}_d}$ finie avec : $d_{\mathcal{H}_1} \leq d_{\mathcal{H}_2} \leq \ldots \leq d_{\mathcal{H}_d} \leq \ldots$
2. Le choix, par *ERM*, de la meilleure hypothèse $h_S^{\star d}$, et donc du meilleur espace d'hypothèses, se fait en sélectionnant l'espace $\mathcal{H}_{d^\star}$ qui offre la meilleure garantie de risque (selon les équations (21.25) et (21.26), soit plus précisément les équations (22.36) et (22.38)). Sous l'hypothèse que l'intervalle de confiance donné par ces équations est serré, on peut espérer ainsi obtenir une bonne approximation du risque réel associé aux hypothèses choisies, et donc pouvoir sélectionner la meilleure d'entre elles en connaissance de cause.

Discussion

De nombreuses variantes et implémentations de l'idée générale exposée ci-dessus ont été proposées et testées (voir par exemple [BBM96], [KMNR95] pour une excellente étude comparative de plusieurs méthodes de sélection de modèles, [LZ96, Mei97, MM96, STBWA96, STBWA98, YB98]). Le principe *SRM* et l'idée essentielle de pénalisation ou de régularisation peuvent être appliqués à de nombreuses classes de modèles (*e.g.* fonctions polynomiales de degré variable, perceptrons multicouches, fonctions trigonométriques, fonctions de Fourier, etc.) ainsi qu'à des procédures d'apprentissage elles-mêmes (*e.g.* le choix des conditions initiales d'un réseau de neurones, le choix du critère d'arrêt, etc. Une bonne revue figure dans [CM98] pp.115-119). Cela met en évidence deux points importants pour la mise en pratique de l'approche *SRM*. Premièrement, le choix de la classe de modèles n'est pas spécifié par le principe *SRM* et fait partie des choix résultant de connaissances *a priori* sur le domaine ou de biais subjectifs de l'expérimentateur. Deuxièmement, si la procédure d'apprentissage elle-même peut présenter des aspects qui

[2] Plus précisément sur les classes de fonctions de perte associées. Mais pour ne pas surcharger les concepts généraux, nous n'en tenons pas compte ici. Bien sûr, dans la pratique, il faudra faire attention à ce point (voir par exemple [Vap95, Vap98]).

Copyright © 2010 Eyrolles.

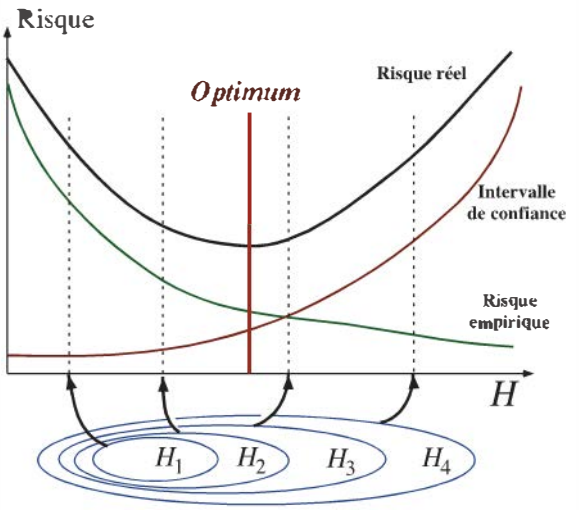


FIG. 21.7: *Le principe SRM.*

introduisent des facteurs de régularisation, il n'est plus possible de ne considérer que la seule régularisation introduite par la dimension de Vapnik-Chervonenkis. Cela complique alors considérablement l'estimation du bon facteur de régularisation, et explique les approches cherchant à déterminer celui-ci de manière empirique en fonction des conditions particulières d'apprentissage.

Par ailleurs, il existe des critères théoriques sur les structures de classes d'hypothèses ($\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \ldots \subseteq \mathcal{H}_d \subseteq \ldots$) dictant comment il faut régler la séquence $d_{\mathcal{H}_1} \leq d_{\mathcal{H}_2} \leq \ldots \leq d_{\mathcal{H}_d} \leq \ldots$ pour que la convergence vers une bonne approximation de la fonction cible f soit rapide en fonction de l'indice $d_{\mathcal{H}_d}$. Ce réglage dépend de la taille de l'échantillon d'apprentissage. L'idée principale est qu'au fur et à mesure qu'augmente la dimension $d_{\mathcal{H}_d}$ de l'espace d'hypothèse, il faut aussi augmenter la « régularité » (souvent mesurée par le degré de dérivabilité) des fonctions hypothèses. La conséquence est que si la fonction cible n'est pas très « régulière », il n'est pas possible de garantir une convergence rapide des espaces d'hypothèses $\mathcal{H}_d$ vers la fonction cible. Vapnik (dans [Vap95] pp.97-100) propose alors une idée intéressante d'approximation locale, en certains voisinages d'intérêts, de la fonction cible pour garantir une convergence rapide de la capacité d'approximation. Il s'agit là d'une idée très séduisante et qui mérite des travaux complémentaires.

2.5 La marge comme mesure de capacité et outil de régularisation

Le concept de marge est devenu ces dernières années un principe unificateur dans l'analyse de nombreuses techniques de classification binaire comme les séparateurs à vastes marges, mais aussi le boosting et les réseaux de neurones. En effet, la marge, ou niveau de confiance, est un outil central pour estimer la performance à venir d'un classificur, plutôt que l'erreur empirique. C'est en faisant appel à cette notion qu'une explication de la résistance du boosting au sur-apprentissage a été avancée [SFBL98]. Ainsi, alors même que l'erreur empirique est annulée, l'augmentation du nombre d'étapes de boosting accroît la marge entre les exemples des deux classes, ce qui resserre le lien entre risque empirique, ici devenu nul, et risque réel (voir chapitre 17).

Plus formellement, ce concept est issu d'une longue démarche. Comme nous l'avons vu, le problème est d'établir un lien entre le risque empirique, mesuré sur l'ensemble d'apprentissage (supposé i.i.d.), et le risque réel ou espérance de risque. Pour cela, il faut faire intervenir des

Copyright © 2010 Eyrolles.

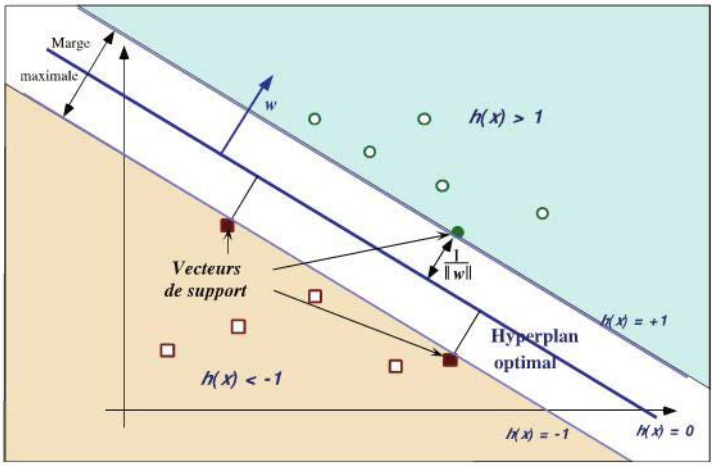


FIG. 21.8: *Séparatrice linéaire de marge maximale entre les points d'apprentissage de deux classes.*

caractéristiques de l'espace des hypothèses envisagées. La dimension de Vapnik-Chervonenkis, comme mesure du pouvoir discriminant d'un ensemble d'hypothèses, permet un tel lien. Cependant, il est assez grossier car il s'applique à des fonctions binaires et ne prend pas en compte la sensibilité des fonctions sous-jacentes. Par exemple, une fonction de décision binaire peut être obtenue par seuillage d'une fonction à valeur réelle : $T_\theta(h)$ qui prend la valeur 1 si h prend une valeur de sortie $\geq \theta$ et 0 ou -1 sinon. C'est pourquoi Vapnik en 1979 a introduit la marge comme facteur caractérisant la sensibilité des fonctions sous-jacentes, obtenant ainsi de nouvelles bornes liant les risques empiriques et réels. Cette analyse supposait néanmoins que tous les points, y compris les points test, soient du bon côté de la marge, ce qui rendait les résultats irréalistes.

Ceci a alors motivé l'introduction d'une analyse des fonctions à valeurs réelles sensibles à une granularité d'intérêt (*scale sensitive*), à savoir la *fat-shattering dimension* et les nombres de couverture. Cette nouvelle analyse plus précise conduit, dans le cas des fonctions de classification binaire, à la prise en compte de la marge dans les bornes obtenues.

Ainsi, pour des fonctions $h \in \mathcal{H}$ à valeur réelle et définies sur une boule de $\mathcal{X}$ de rayon R, il existe une constante c telle que pour toute distribution de probabilité sur $\mathcal{X}$, toute marge ρ et pour toute fonction h de marge au moins ρ sur un tirage de m exemples d'apprentissage, on a avec une probabilité supérieure à $1 - \delta$:

$$R_{\text{Réel}}(h) = \frac{c}{m}\left(\frac{R^2}{\rho^2}\log^2\left(\frac{m}{\rho}\right) + \log\left(\frac{1}{\delta}\right)\right)$$

La leçon générale à retenir est qu'il faut essayer d'avoir une erreur empirique faible tout en maintenant la marge la plus grande possible. Dans le cas des fonctions de décision linéaire, comme les séparateurs à vastes marges (chapitre 14), il faut chercher à maximiser la marge tout en ayant le moins de d'exemples d'apprentissage possibles qui ne soient pas du bon côté de la marge.

Copyright © 2010 Eyrolles.

3. Prise en compte de l'algorithme d'apprentissage dans la théorie

L'étude du principe inductif par minimisation du risque empirique conduite dans les sections précédentes mettait entièrement l'accent sur l'espace des hypothèses $\mathcal{H}$ utilisé. L'idée était de mener une étude en convergence uniforme, c'est-à-dire de borner la distance entre $R_{\text{Réel}}(h)$ et $R_{\text{Emp}}(h)$ pour toute hypothèse $h \in \mathcal{H}$ simultanément. En revanche, l'algorithme d'apprentissage n'était pas vraiment pris en compte. Il était supposé capable de trouver miraculeusement la meilleure hypothèse $\hat{h}_{\mathcal{S}} \in \mathcal{H}$ selon le critère inductif choisi en fonction de l'échantillon $\mathcal{S}$ fourni.

Un certain nombre d'analyses récentes prennent en compte *le comportement de l'algorithme d'apprentissage sur les données disponibles* pour calculer la distance entre $R_{\text{Réel}}(h)$ et $R_{\text{Emp}}(h)$, sans faire appel à une caractérisation de l'espace des hypothèses $\mathcal{H}$. Parmi ces approches figurent :

- L'analyse par compression de l'échantillon d'apprentissage.
- Le cadre de la fonction de félicité (*luckiness framework*).
- L'analyse par stabilité du risque empirique.

Par ailleurs, d'autres propriétés de l'algorithme d'apprentissage ont été considérées. Par exemple :

- La complexité calculatoire de l'apprentissage dans la première approche PAC.
- Le fait que l'algorithme d'apprentissage ne parvient pas nécessairement à identifier la meilleure hypothèse $\hat{h}_{\mathcal{S}}$ selon le critère inductif, ce qui introduit donc un terme d'erreur supplémentaire.
- La variation du taux de couverture des hypothèses dans l'apprentissage de concept et l'impact de phénomène de transition de phase sur l'apprentissage quand celui-ci est guidé par le gradient du taux de couverture.

Cette section décrit rapidement ces cadres d'analyse. La plupart d'entre eux sont l'objet de recherches actives.

3.1 Sélection de modèle à partir des données (*the luckiness framework*)

L'étude de l'induction par minimisation du risque empirique s'est concentrée sur une analyse de la convergence uniforme du risque empirique vers le risque réel pour un espace d'hypothèses $\mathcal{H}$. Les bornes de généralisation obtenues expriment la probabilité que l'hypothèse minimisant le risque empirique ait un risque réel proche de ce risque empirique, ce qui, en vertu de la convergence uniforme, permet de supposer qu'elle n'est pas loin de minimiser le risque réel. L'intérêt de cette analyse est qu'elle *s'applique à tout algorithme d'apprentissage* et à *toute distribution possible des données*. L'inconvénient est qu'elle conduit à des bornes très larges, généralement très éloignées des comportements de convergence observés expérimentalement. Au prix de devoir faire quelques hypothèses supplémentaires sur le monde et sur l'algorithme d'apprentissage, on aimerait pouvoir obtenir des bornes de généralisation plus adaptées aux situations effectivement rencontrées. Ces informations supplémentaires peuvent se décliner en :

- Informations *sur les données*. L'échantillon de données d'apprentissage fournit en effet des informations sur la distribution effective $\mathsf{p}_{\mathcal{X}\mathcal{Y}}$.
- Informations sur *l'algorithme d'apprentissage*. En effet, d'une part l'algorithme d'apprentissage peut ne pas implémenter exactement la minimisation du risque empirique, soit parce que l'optimisation n'est pas parfaite, soit parce que l'algorithme cherche le minimum d'un risque régularisé. D'autre part, l'algorithme peut chercher des régularités dans le monde plus ou moins adaptées aux données.

L'une des motivations pour chercher un nouveau cadre d'analyse a été la mise au point des Séparateurs à Vastes Marges (*Support Vector Machines*) (SVM, voir chapitres 9 et 14). Ces mé-

Copyright © 2010 Eyrolles.

thodes exhibent en effet souvent de très bonnes performances en généralisation alors même que la dimension de Vapnik-Chervonenkis de l'espace des hypothèses associé est très grande, voire infinie. Vapnik [Vap98] avait cru pouvoir justifier ces très bonnes performances par la mise en œuvre automatique d'une minimisation du risque structurel (SRM). En effet, les fonctions séparatrices des données de plus grande marge ont une dimension de Vapnik-Chervonenkis inférieure à des fonctions séparant les données avec une marge plus faible. Cependant, le SRM n'est applicable que dans la mesure où la stratification des espaces d'hypothèses est déterminée *a priori*, indépendamment des données. Or la notion de marge est entièrement dépendante d'un échantillon de données fixé. La minimisation du risque structurel ne peut donc pas être invoquée. Comment alors expliquer le bon fonctionnement des SVM ?

Plusieurs stratégies ont été proposées pour mieux rendre compte des performances effectives des algorithmes d'apprentissage. Parmi elles, la *compression* de Littlestone et Warmuth [LW86], le concept de *stabilité algorithmique* de Bousquet et Elisseeff [BE02], et le *cadre de la fonction de félicité* (*luckiness framework*), dû à Shawe-Taylor et al. [STBWA98], puis étendu au *cadre de la fonction de félicité algorithmique* par Herbrich et Williamson [HW02]. Nous décrivons succinctement le cadre de la fonction de félicité algorithmique dans la suite.

L'idée est, comme pour la minimisation du risque structurel, de stratifier l'espace des hypothèses en sous-espaces de capacité croissante, mesurée par un nombre de couverture. Dans le nouveau cadre, la stratification est cependant définie *a posteriori*, après observation de données d'apprentissage, et en tenant compte des hypothèses qui peuvent être considérées par l'algorithme d'apprentissage utilisé $\mathcal{A}$. L'espoir est que nous serons chanceux et que la fonction cible apparaîtra parmi les premiers espaces d'hypothèses de la stratification.

Définition 21.5 (Fonction de félicité algorithmique)

Une fonction de félicité L *associe un réel à un couple (algorithme d'apprentissage, échantillon de données)* : $\mathcal{A} \times \mathcal{S} \to \mathbb{R}$.

Plus grande est la valeur de la fonction de félicité, plus élevée est la probabilité que l'algorithme $\mathcal{A}$ se comporte bien sur la distribution de données dont est issu l'échantillon $\mathcal{S}$ par un tirage i.i.d..

Le principe de l'analyse est d'ordonner, sur le critère de la fonction de félicité, les hypothèses qui peuvent être apprises à partir de l'échantillon d'apprentissage. Il est alors possible de comparer le nombre de projections des hypothèses apprenables sur les donnés d'apprentissage au nombre maximum d'étiquetages possibles (voir les nombres de couverture plus haut). Comme, pour l'analyse basée sur la dimension de Vapnik-Chervonenkis, si ce nombre est inférieur au nombre maximal, il existe une forte probabilité que ce que l'algorithme $\mathcal{A}$ a appris à partir de l'échantillon d'apprentissage $\mathcal{S}$ soit, de fait, une bonne hypothèse. Les détails de la démonstration, très technique, peuvent être trouvés en particulier dans [HW02, MP03].

Si la fonction de félicité choisie est « correcte », les bornes calculées sont valides et sont beaucoup plus étroites que les bornes obtenues par l'analyse classique indépendante des données. Cette fonction de félicité (qui inclue des contraintes telles que la régularité des hypothèses produites ou bien l'ampleur des marges séparatrices pour les classifieurs linéaires) n'est cependant pas facile à déterminer en pratique. Sa définition est en fait plus ou moins équivalente à la détermination d'une bonne distribution *a priori* dans les méthodes bayésiennes, qui est notoirement un problème difficile.

Pour les bayésiens, l'idée est de définir *a priori* une distribution de probabilité sur l'espace des hypothèses $\mathcal{H}$ en favorisant les hypothèses dont on « parie » qu'elles auront une erreur faible. On peut alors obtenir des bornes d'erreur plus précises si ces hypothèses sont effectivement les meilleures sur les données, au prix d'avoir des bornes moins précises sinon.

Copyright © 2010 Eyrolles.

3.2 Analyse par la capacité à comprimer l'échantillon d'apprentissage

L'idée de base de ce type d'analyse est de regarder quelle proportion d'exemples, tirés d'un échantillon d'apprentissage $\mathcal{S}$ de taille m lui même issu d'un tirage i.i.d., aurait permis à l'algorithme d'apprentissage considéré d'obtenir la même hypothèse finale. Éventuellement, l'ordre des exemples pourra être pris en compte s'il s'agit d'un apprentissage en ligne.

Ainsi, par exemple, l'algorithme du Perceptron (voir chapitre 9) n'utilise que les exemples qui permettent d'adapter ses coefficients. Les autres exemples sont inutiles et ne changent pas le résultat de l'apprentissage. De même, les Séparateurs à Vastes Marges n'utilisent que les exemples critiques ou encore exemples support pour déterminer la séparatrice entre les deux classes d'exemples (voir figure 21.8 et le chapitre 14). Dans les deux cas, l'algorithme d'apprentissage aurait pu n'observer que $m' \leq m$ exemples d'apprentissage pour calculer son hypothèse finale.

Intuitivement, cela signifie que l'algorithme d'apprentissage avait un biais *a priori* bien « aligné » avec les données d'apprentissage, et correspondant de fait à une classe d'hypothèses restreinte. Il n'est pas étonnant alors que le risque empirique mesuré soit représentatif du risque réel.

Une analyse assez simple (voir par exemple [Her02], pp.176-182) conduit à des bornes du type :

$$R_{\text{Réel}}(h) \leq \frac{m}{m-m'} R_{\text{Emp}}(h) + \sqrt{\frac{\ln\left(\binom{m}{m'} m'!\right) + 2\ln(m) + \ln\left(\frac{1}{\delta}\right)}{2(m-m')}} \qquad (21.27)$$

On notera que cette borne est d'autant plus serrée que m' est petit. En fait, les bornes obtenues deviennent meilleures que celles dues à une analyse en convergence uniforme lorsque $m' < m/2$.

Une remarque plus essentielle est que cette borne ne prend pas en compte l'espace $\mathcal{H}$, par exemple sa taille $|\mathcal{H}|$ ou sa dimension de Vapnik-Cervonenkis $d_{\mathcal{H}}$. En un certain sens, le raisonnement est similaire à celui de l'analyse par la fonction de *félicité*, mais le raisonnement est beaucoup plus simple et ne nécessite pas la mise en œuvre d'une convergence uniforme sur l'espace des versions $V_{\mathcal{H}}(\mathcal{S})$ (l'ensemble des hypothèses de risque empirique nul) impliquant des difficultés techniques telles que l'introduction d'une régularité probable.

3.3 Les analyses par stabilité du risque empirique

À côté des analyses mettant en jeu des convergences uniformes sur $\mathcal{H}$, de celles estimant la complexité de l'espace de recherche (par exemple à l'aide du nombre de couverture) et de celles utilisant des informations *a posteriori*, comme l'analyse de la félicité ou celle de la compression, une autre approche s'appuie sur la propriété de stabilité des algorithmes d'apprentissage.

L'idée sous-jacente est toujours celle du théorème de la limite centrale et de ses dérivées puissantes appelées « inégalités de concentration » selon laquelle *une variable aléatoire dépendant, de manière « douce », de nombreuses variables indépendantes, mais pas trop de chacune d'elles, est essentiellement constante (et, très probablement, proche de son espérance).* Déclinée dans le contexte de l'apprentissage et du principe de minimisation du risque empirique, la variable aléatoire d'intérêt correspond à la différence entre le risque empirique de l'hypothèse $\hat{h}$ retournée par l'algorithme d'apprentissage et son risque réel. Il peut être montré que si l'écart entre les deux risques est à peu près constant, alors cet écart est nul.

Plus précisément, un algorithme est *uniformément β stable* si l'omission d'un seul exemple de l'échantillon d'apprentissage ne change pas la perte de l'hypothèse apprise sur n'importe quel point de $\mathcal{X} \times \mathcal{Y}$: $\ell(\hat{f}_{\mathcal{S}}(\mathbf{x}), y)$ de plus que β et cela quel que soit l'échantillon d'apprentissage $\mathcal{S}$. En notant β_m le terme β considéré comme une fonction de la taille de l'échantillon d'apprentissage,

Copyright © 2010 Eyrolles.

on dira qu'un algorithme d'apprentissage est *stable* si la valeur de β_m décroît en $1/m$. Cela signifie qu'aucun exemple ne peut avoir une influence supérieure aux autres.

L'apprentissage par plus proches voisins, qui consiste à conserver en mémoire les exemples d'apprentissage et à décider de l'étiquette de nouveaux exemples sur la base de l'étiquette de leurs plus proches voisins est typiquement un algorithme stable. Il s'agit cependant d'une condition assez forte qui exclue notamment que des exemples aberrants (*outliers*) puissent jouer une rôle supérieur aux autres exemples.

Les algorithmes stables ont la propriété que le risque empirique de l'hypothèse apprise est très proche du risque *leave-one-out* (voir chapitre 3) qui est un estimateur non biaisé du risque réel. On peut interpréter ceci comme le fait que les algorithmes stables n'explorent pas l'ensemble de toutes les fonctions possibles de $\mathcal{H}$, mais seulement un sous-espace, ce qui revient à une sorte de régularisation. Pour être stable, ces algorithmes compensent les erreurs inévitables commises sur certains points d'apprentissage par une plus grande régularité des hypothèses apprises.

En conséquence, les bornes de généralisation ne mentionnent plus des caractéristiques de l'espace $\mathcal{H}$, mais le degré de stabilité de l'algorithme considéré. Les détails de cette élégante analyse peuvent se trouver en particulier dans |BE02| et dans |Her02|.

Notons cependant que l'existence d'un algorithme stable ne signifie pas qu'il s'agit d'un bon algorithme d'apprentissage, mais seulement que le risque empirique de l'hypothèse retournée est représentatif de son risque réel. À la limite, un algorithme renvoyant la même fonction constante quel que soit l'échantillon d'apprentissage est maximalement stable ; il est aussi maximalement stupide.

L'analyse de l'influence d'un exemple d'apprentissage sur la stabilité de l'apprentissage n'est qu'un exemple du type d'analyse plus général consistant à étudier comment l'hypothèse produite par un algorithme change en fonction de perturbations des paramètres en jeu.

3.4 Sélection de modèle et complexité calculatoire

Dans tout ce qui précède sur l'analyse de l'induction, nous n'avons pris en compte que l'expressivité des espaces d'hypothèses considérés. Or, dans les travaux pionniers de Leslie Valiant (1949-...) |Val84, PV88|, à partir de 1984, celui-ci prenait également en compte la difficulté calculatoire à trouver une bonne hypothèse dans un espace d'hypothèses donné. Nous renvoyons le lecteur au chapitre 21 pour une description plus détaillée du « modèle de Valiant ». Notons que cette notion de complexité calculatoire peut avoir un impact sur la sélection de modèle comme il a été montré récemment.

C'est ainsi que par exemple |BDEL03| ont montré que des échantillons de données d'apprentissage peuvent être très difficiles à apprendre en explorant un espace d'hypothèses $\mathcal{H}_1$ (e.g. hyperrectangles) alors qu'elles sont très faciles à apprendre en explorant un autre espace $\mathcal{H}_2$ (e.g. boules). Plus précisément, alors qu'il est très coûteux de trouver une hypothèse proche de la meilleure hypothèse possible dans $\mathcal{H}_1$, cela peut être très facile dans $\mathcal{H}_2$. Dans ce cas, le choix de l'espace des hypothèses ne doit pas seulement prendre en compte les caractéristiques intrinsèques de l'espace des hypothèses, en particulier sa capacité, quelle qu'en soit la mesure, mais aussi la difficulté du problème d'optimisation dans cet espace, étant donnés les données et l'algorithme d'apprentissage.

3.5 Erreur induite par les imperfections de l'algorithme de recherche dans $\mathcal{H}$

L'analyse classique des algorithmes d'apprentissage par minimisation du risque empirique fait intervenir les facteurs suivants : la fonction optimale de Bayes $f(\mathbf{x}) = \text{Argmin}_y \mathbb{E}[\ell(\hat{y}, y)|\mathbf{x}]$, la fonction optimale $h^* \in \mathcal{H}$, la fonction de $\mathcal{H}$ minimisant le risque empirique : $\hat{h}_{\mathcal{S}}$.

Copyright © 2010 Eyrolles.

L'*erreur de biais* est due au fait que généralement $h^* \neq f$ à cause du choix de $\mathcal{H}$ qui souvent n'inclue pas f. L'*erreur d'approximation* vient de ce que la fonction $\hat{h}_{\mathcal{S}}$ minimisant le risque empirique est une variable aléatoire dépendant du tirage de l'échantillon d'apprentissage $\mathcal{S}$ et n'a pas de raison de coïncider avec h^*.

Cependant, cette analyse suppose que l'algorithme identifie $\hat{h}_{\mathcal{S}}$ miraculeusement. De fait, la recherche de $\hat{h}_{\mathcal{S}}$ implique une exploration de $\mathcal{H}$ qui peut être coûteuse. Si les contraintes de temps ne permettent pas d'identifier $\hat{h}_{\mathcal{S}}$, alors l'algorithme retourne une hypothèse $\tilde{h}_{\mathcal{S}}$ qui peut être différente de $\hat{h}_{\mathcal{S}}$. La différence $\mathbb{E}_{\mathcal{S}}[R_{\text{Réel}}(\hat{h}_{\mathcal{S}}) - R_{\text{Réel}}(\tilde{h}_{\mathcal{S}})]$, où l'espérance est prise sur choix aléatoire de l'échantillon d'apprentissage $\mathcal{S}$, est appelée *erreur d'optimisation*. On voit qu'encore une fois, les propriétés de l'algorithme d'apprentissage interviennent dans sa performance.

Une idée avancée par Bottou [1965-..] et Bousquet [1975-..], [BB08], est qu'il est intéressant de prendre en compte cette erreur d'optimisation pour distinguer deux régimes possibles d'apprentissage :

- Le régime *petite échelle* dans lequel l'apprentissage est principalement contraint par le nombre d'exemples d'apprentissage disponibles. Dans ce cas, le temps calcul n'étant pas le facteur limitant, on peut ignorer l'erreur d'optimisation.
- Le régime *grande échelle*, caractéristique par exemple de l'apprentissage en-ligne à partir de flux de données, dans lequel le facteur limitant à chaque instant est le coût calculatoire. Dans ce cas, il peut être plus judicieux de ne pas dépenser trop de ressources à diminuer l'erreur d'optimisation au détriment de la prise en compte d'exemples d'apprentissage. En effet, l'analyse de cas particuliers, tels que les algorithmes de descente de gradient, montre que l'on gagne davantage à prendre en compte l'information supplémentaire apportée par des exemples nouveaux qu'à trop essayer de réduire l'erreur d'optimisation.

Ce type d'analyse est encore préliminaire mais mérite d'être développée.

3.6 Induction guidée par gradient et phénomènes de transition de phase

Les analyses décrites précédemment prennent en compte des propriétés générales sur l'espace des hypothèses $\mathcal{H}$ ou sur l'exploration de sous-espaces de $\mathcal{H}$. Il est possible, tout en restant très général, d'aller plus loin et de prendre en compte des caractéristiques plus précises des algorithmes d'apprentissage.

Par exemple, dans le cas de l'apprentissage de concept, c'est-à-dire de la classification binaire, la grande majorité des algorithmes opèrent par essais et erreurs en étant guidés par le gradient d'une fonction du risque empirique. Plus précisément, ces algorithmes cherchent des hypothèses qui étiquettent comme positif (presque) tous les exemples d'apprentissage positifs, et étiquettent comme négatif (presque tous) les exemples d'apprentissage négatifs. En un certain sens, ces algorithmes cherchent des hypothèses qui satisfont aux contraintes imposées par les exemples d'apprentissage. On peut donc les considérer comme des algorithmes de satisfaction de contraintes. Pour chaque couple (hypothèse, échantillon $\mathcal{S}$), l'algorithme cherche si l'hypothèse est cohérente avec les exemples de $\mathcal{S}$. Et tant que l'algorithme n'a pas trouvé une hypothèse satisfaisante, il continue d'explorer $\mathcal{H}$.

Dans le cas de l'apprentissage de concept, cet apprentissage se fait généralement par modification de l'hypothèse candidate en la généralisant ou en la spécialisant (voir chapitre 4) grâce à des opérateurs spécialisés. Les opérateurs permettant d'obtenir de meilleures hypothèses que l'hypothèse candidate sont utilisés et ainsi de suite.

Malheureusement, depuis le début des années 1990, l'étude des problèmes de satisfaction de contraintes a montré que de nombreux problèmes sont sujet à une transition de phase, c'est-à-dire qu'en moyenne sur des problèmes tirés aléatoirement selon certains paramètres, les problèmes

Copyright © 2010 Eyrolles.

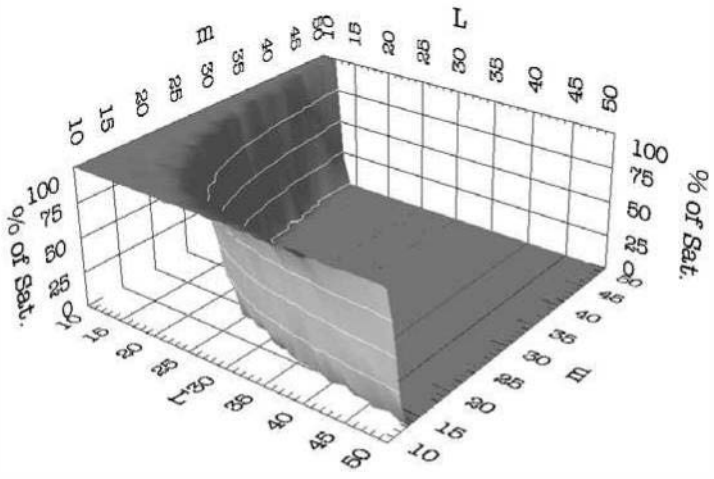


FIG. 21.9: *Taux de couverture des exemples par des hypothèses dans le cas de la programmation logique inductive, ici en fonction de deux paramètres : m, le nombre de* symboles de prédicats *dans h, et L, le nombre total de* constantes *dans l'exemple. On voit que le taux de couverture passe brutalement de 1 à 0.*

sont soit toujours satisfiables, soit quasiment jamais satisfiables. Dans le cadre de l'apprentissage de concept, des études empiriques stimulées par ces travaux, ont montré que ce phénomène de transition de phase se manifeste également pour des cadres d'apprentissage usuels, par exemple l'apprentissage d'hypothèses formulées en logique du premier ordre (chapitre 5) ou l'apprentissage de grammaires (chapitre 7). En pratique, cela signifie que dans ces apprentissages, les hypothèses considérées par les algorithmes, soit étiquettent comme positifs (presque) tous les exemples, soit étiquettent comme négatifs (presque) tous les exemples. De ce fait, un algorithme explorant l'espace des hypothèses ne peut plus être guidé par gradient puisque toutes les hypothèses considérées autour d'une hypothèse candidate ont le même risque empirique.

Ce phénomène qui explique un certain nombre d'échecs en apprentissage devrait être mieux compris par l'utilisation d'idées et de méthodes de la physique statistique qui ont porté leurs fruits dans le domaine des problèmes d'optimisation et de satisfaction de contraintes (voir [GSC10]).

4. Autres types d'analyses

4.1 Analyse de la physique statistique

L'analyse de Vapnik de l'induction par minimisation du risque empirique (ERM) à partir d'un échantillon aléatoire de m exemples conduit à des courbes d'erreur en généralisation bornées par $O(d_{\mathcal{H}}/m)$ (dans le cas de la discrimination et d'une fonction cible appartenant à l'espace d'hypothèses $\mathcal{H}$ de dimension de Vapnik-Chervonenkis $d_{\mathcal{H}}$) ou par $O(d_{\mathcal{H}}/m)$ (dans le cas de la discrimination et d'une fonction cible n'appartenant pas à $\mathcal{H}$). Rappelons que ces bornes, obtenues dans le cadre d'une analyse dans le pire cas, sont universelles : elles sont valables pour tout espace $\mathcal{H}$ d'hypothèses, pour toute distribution de données et pour toute fonction cible. Par ailleurs, il a été montré que ces bornes sont essentiellement les meilleures possibles dans le pire cas, dans le sens où, pour tout espace $\mathcal{H}$, il existe une distribution de données sur $\mathcal{X}$ pour laquelle la borne

Copyright © 2010 Eyrolles.

inférieure sur l'erreur de généralisation est égale à la borne supérieure donnée ci-dessus. Dès lors, on pourrait croire que le comportement réel en généralisation des algorithmes d'induction est décrit soit par la forme fonctionnelle $d_{\mathcal{H}}/m$ soit par $\sqrt{d_{\mathcal{H}}/m}$. Il se trouve que l'on peut observer toute une variété de comportements ne correspondant pas à ces formes fonctionnelles. Ainsi, bien souvent, des erreurs en généralisation (risques réels) faibles sont obtenues pour des échantillons d'apprentissage beaucoup plus faibles que ceux prédits par la théorie (parfois même pour des échantillons de taille $< d_{\mathcal{H}}$, c'est-à-dire pour lesquels aucune borne en généralisation n'est valide en théorie). On observe également parfois des courbes d'apprentissage présentant de brutales transitions (voir la figure 21.10) réminiscentes des phénomènes de transitions de phase en physique.

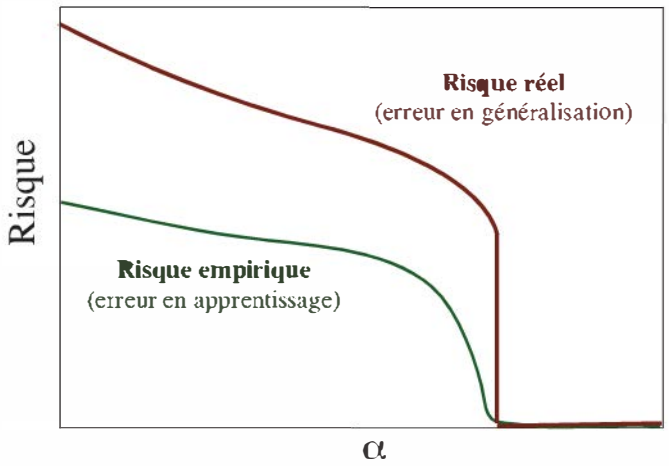


FIG. 21.10: *Un exemple de courbe d'apprentissage présentant une transition brutale vers une erreur en généralisation nulle. On étudie donc les caractéristiques de l'espérance de risque en fonction du rapport $\alpha = m/N$, m étant le nombre d'exemples dans l'échantillon d'apprentissage et N le nombre de degrés de liberté gouvernant $\mathcal{H}$ (typiquement, le nombre de connexions dans un réseau de neurones, sans qu'il soit établi que cela constitue des degrés de liberté indépendants).*

Plusieurs explications de ces phénomènes ont été proposées. Par exemple :

- L'algorithme d'apprentissage (*e.g.* un réseau connexionniste) n'accéderait effectivement qu'à un sous-espace de l'espace des hypothèses $\mathcal{H}$, dont la dimension de Vapnik-Chervonenkis serait inférieure à $d_{\mathcal{H}}$. Il faudrait donc prendre en compte la dimension de Vapnik-Chervonenkis du sous-espace réellement exploré. Il y a là d'ailleurs une direction de recherche intéressante visant à prendre en compte la stratégie d'exploration de l'algorithme.
- La distribution des données serait en général beaucoup plus favorable que la distribution la pire possible. Cela a motivé des travaux dédiés à l'étude de distributions particulières. Surtout, c'est là l'une des sources de l'excitation autour des séparateurs à vastes marges (voir chapitre 9 et plus généralement des méthodes à base de fonctions noyau.

Afin d'analyser ces phénomènes dont il semble difficile de rendre compte par l'approche de Vapnik, certains théoriciens ont eu recours à des méthodes issues de la physique statistique. L'idée essentielle est de considérer l'espace $\mathcal{H}$ des hypothèses comme un ensemble d'états possibles d'un système (physique) soumis à la contrainte d'une séquence d'apprentissage donnée. À chaque état (hypothèse) peut être associée une énergie (risque empirique). On cherche alors quelles sont les caractéristiques de cet espace d'états, et, en particulier, pour chaque état possible, la probabilité

Copyright © 2010 Eyrolles.

de s'y retrouver après une exploration stochastique guidée par l'énergie. Ainsi, au lieu d'étudier le risque réel associé à la pire hypothèse minimisant le risque empirique, comme dans l'approche en pire cas, on étudie l'espérance du risque réel dans un espace d'hypothèses sous une distribution de probabilité reflétant la performance en apprentissage de chaque hypothèse. Il s'agit donc bien d'une analyse du principe inductif ERM, mais d'une *analyse en cas moyen* sur l'ensemble de l'espace d'hypothèses en supposant donnés un échantillon d'apprentissage $\mathcal{S}$ et une densité de probabilité *a priori* sur $\mathcal{H}$.

L'étude de l'espérance de risque réel (équation (21.28))

$$\mathbf{E}_{\mathcal{H}}(R_{Réel}(h)) = \int_{h \in \mathcal{H}} R_{Réel}(h)\, p(h|\mathcal{S})\, dh \tag{21.28}$$

conduit à examiner la distribution de Gibbs sur l'espace des hypothèses et son évolution en fonction de l'échantillon d'apprentissage $\mathcal{S}$. Cette grandeur dépend de l'échantillon d'apprentissage $\mathcal{S}_m$ (ce que les physiciens associent à un « désordre gelé » pour indiquer que le système a évolué sous la contrainte fixée posée par $\mathcal{S}_m$). Il est intéressant de chercher à s'affranchir de cette dépendance en étudiant l'espérance du risque réel moyennée sur tous les échantillons d'apprentissage :

$$\mathbf{E}_{\mathcal{S}_m}[\mathbf{E}_{\mathcal{H}}(R_{Réel}(h))] = -\frac{\partial}{\partial \beta}\{\mathbf{E}_{\mathcal{S}_m} \ln[Z_m(\beta)]\} \tag{21.29}$$

Le problème est que le calcul de cette grandeur est en général très difficile. Il n'est résolu que pour des cas particuliers par l'emploi de méthodes encore mal maîtrisées. Deux idées sont essentielles pour aborder ce calcul :

1. Ce qui est important, c'est une sorte de capacité associée à chaque degré de liberté de l'espace d'hypothèses. On étudie donc les caractéristiques de l'espérance de risque en fonction du rapport $\alpha = m/N$, m étant le nombre d'exemples dans l'échantillon d'apprentissage et N le nombre de degrés de liberté gouvernant $\mathcal{H}$ (typiquement, le nombre de connexions dans un réseau de neurones, sans qu'il soit établi que cela constitue des degrés de liberté indépendants). Lorsque l'on fait tendre $m \to \infty$ en gardant α constant, on parle alors de *limite thermodynamique*. Les courbes d'apprentissage sont établies en examinant l'espérance de risque en fonction du rapport α.

2. On espère que, comme en physique des verres de spin, les propriétés macroscopiques des systèmes d'apprentissage (par exemple leur risque réel) présentent des propriétés d'*automoyennage*. Cela signifie que lorsque les contraintes (l'échantillon d'apprentissage) sont engendrées par une même distribution, les propriétés macroscopiques qui en découlent sont les mêmes et ne dépendent donc pas de la réalisation particulière d'un échantillon d'apprentissage. Cela signifie, qu'à la limite de $N \to \infty$, tous les échantillons d'apprentissage sont équivalents et l'on peut alors obtenir facilement des propriétés génériques des systèmes d'apprentissage.

Du fait de la difficulté technique des méthodes de calcul mises en jeu et de leurs domaines de validité souvent restreints quand ils ne sont pas incertains, les résultats obtenus sont parcellaires. Nous n'entrerons pas ici dans leur détails.

L'approche de la physique statistique qui cherche à étudier des propriétés typiques du principe ERM plutôt que des bornes de confiance est potentiellement très intéressante, et ce d'autant plus qu'on peut également obtenir par ce biais des informations sur la dynamique de l'apprentissage et non seulement sur ses propriétés asymptotiques. C'est pourquoi nous croyons utile de l'évoquer dans cet ouvrage.

Copyright © 2010 Eyrolles.

Cependant, cette approche qui repose sur la mise à jour de propriétés d'automoyennage dans les systèmes d'apprentissage, pose des problèmes redoutables et implique la mise en œuvre de techniques difficiles et dont les domaines de validité sont encore imprécisément connus. Cela explique sans doute le petit nombre de publications la concernant. Sans être exhaustifs, nous pouvons citer en particulier [Gar88, HKS94, HKST96, LTS89, OH91, SST92, WRB93].

4.2 Apprentissage et analyse des systèmes dynamiques

Un apprenant est un système caractérisé par un certain état qui évolue en fonction de cet état et des entrées dues à l'environnement. On peut donc le caractériser comme un système dynamique. Lorsque l'apprenant est soumis à une séquence d'entrées, il suit une trajectoire le faisant passer d'un état d'origine e_0 à un état final e_f. On peut alors chercher ce qui caractérise ces trajectoires. La physique des systèmes dynamiques nous apprend que la trajectoire suivie par un système rend extrémale une quantité que l'on appelle *action* et qui est l'intégrale le long de la trajectoire d'une quantité appelée Lagrangien (à ne pas confondre avec les multiplicateurs de Lagrange). Si l'on connaît le Lagrangien d'un système, on peut calculer sa trajectoire pour toute séquence d'apprentissage.

Ce qui est intéressant, c'est que cette approche permet de relier la notion d'information avec celle d'apprentissage. En effet, considérons maintenant un apprenant tel que, étant donné un état initial e_0, quel que soit l'ordre dans lequel est présenté un échantillon d'apprentissage, il parvienne au même état final e_f. En d'autres termes, l'apprenant est insensible à l'ordre de présentation des données. Cela correspond à un invariant sur la trajectoire qui est lié à un invariant de l'action et du Lagrangien. Cette invariance implique des relations spécifiques entre information et prise en compte de cette information par l'apprenant. Notamment, le système ne peut oublier n'importe comment l'information qui lui a été fournie. On peut ainsi établir un lien entre information, système d'apprentissage et oubli. Après tout, il est curieux que la notion d'oubli n'apparaisse qu'ici dans un livre sur l'apprentissage. Apprentissage - oubli / oubli - apprentissage, l'un est-il pourtant dissociable de l'autre ?

Il y a encore tellement de choses à apprendre sur l'apprentissage !

4.3 Discussion sur l'analyse classique : variantes et perspectives

Voici donc le terme d'un ouvrage volumineux que beaucoup de spécialistes trouveront trop court, tant il y a de choses qui n'ont pas été exposées. La science de l'aprentissage artificiel a en effet été fructueuse. Un cadre théorique s'est vigoureusement développé, solidement enraciné dans les théories statistiques de lois de convergence, la théorie bayésienne et la théorie de la complexité algorithmique. De nombreux algorithmes et techniques d'apprentissage ont été mis au point, que les praticiens de multiples domaines sont avides d'employer (génomique, fouille de données en entreprise, études de marché, etc.). La communauté des chercheurs et des praticiens de l'apprentissage artificiel est active et reconnue institutionnellement : avec des postes dans les universités et les laboratoires, des conférences et des revues spécialisées. Pourtant cette science, si vive, si féconde, répond-elle à toutes les interrogations sur l'apprentissage?

Lorsque l'on prend du recul, on peut être surpris par l'image de l'apprentissage qu'elle dessine. On s'y intéresse en effet à des agents isolés, recevant passivement des données produites de manière aléatoire par une Nature indifférente. Ces agents ne cherchent pas vraiment à comprendre le monde où ils se trouvent, mais tentent « seulement » d'être bons en moyenne (voir la figure 21.11). D'ailleurs, ils n'évoluent pas. Une seule dose de données ingurgitée d'un seul coup, et c'en est fini pour toujours. La science de l'apprentissage est une science du statique et non du dynamique ! Ce n'est pas plus une science de l'information ou de la connaissance : l'expression

Copyright © 2010 Eyrolles.

des connaissances préalables est très pauvre, se résumant essentiellement à des *a priori* sur les fonctions cible possibles et à l'algorithme utilisé ; les connaissances produites consistent le plus souvent en des procédures de décision, parfois complètement opaques comme dans les réseaux connexionnistes. Il ne s'agit évidemment pas ici de dénoncer les recherches menées en apprentissage artificiel, mais il est clair que le paradigme actuel, par ailleurs si puissant, est notoirement limité. Il y a encore des révolutions scientifiques à mener pour les esprits audacieux. Sans décrire ces révolutions à venir, il est intéressant de voir que certaines directions de recherche actuelles tendent à élargir le cadre dominant.

Sans tout bouleverser, que peut-on remettre en cause dans le cadre classique ?

- **Le critère de performance.** La plupart du temps, ce critère cherche à définir l'écart entre l'état de la Nature, la fonction cible par exemple, et son estimation par l'agent apprenant. Cet écart, qui fonde toutes les approches relevant de la théorie de l'approximation, a deux aspects. D'une part, une mesure de distance ponctuelle, par exemple une distance quadratique entre un point prévu et un point fourni par l'oracle. D'autre part, une densité de distribution sur l'espace $\mathcal{X} \times \mathcal{U}$ des points. À partir de là, l'écart prend la forme d'une espérance : l'intégrale des distances ponctuelles pondérées par la distribution.
  Il faut noter que ce critère n'est pas tourné vers l'identification ou la compréhension de la fonction cible. Il vise en effet à l'efficacité, mais pas à la précision, comme l'illustre la figure 21.1.1. Intuitivement, il serait en effet intéressant de dédier des ressources de l'apprenant (par exemple des questions à poser, ou bien des paramètres : centres de fonctions noyau, etc.) à l'aproximation dans les régions de forte dynamique, mais le critère de performance en espérance conduit à consacrer les ressources aux régions de fortes densités.



FIG. 21.11: *Le critère de performance habituel, mesurant une espérance, privilégie l'aproximation dans les régions de forte densité de données et non dans les régions de forte dynamique. De ce fait, la fonction identifiée par minimisation du risque empirique, ici une droite, peut être très différente de la fonction cible.*

D'autres critères de performance peuvent être intéressants. Par exemple, la performance d'un système d'apprentissage, l'écart précédemment défini, peut éventuellement varier en fonction d'autres paramètres, comme le taux de faux positifs admis, ou la confiance du système dans sa prédiction. Ce n'est alors plus un nombre, comme le taux d'erreur, qui caractérise un système, mais une courbe, voire une surface. On parle alors d'optimisation multicritère et la comparaison entre systèmes d'apprentissage doit faire intervenir l'ensemble des paramètres. On peut aussi vouloir prendre en compte la *complexité computationnelle* d'un apprentissage afin de traduire qu'il doit s'effectuer dans un temps raisonable, compatible avec les exigences de fonctionnement dans le monde. C'est ce que tente de capturer un versant du modèle d'apprentissage PAC (voir chapitre 2) qui pose qu'un apprentissage n'est possible que s'il est de complexité au plus polynomiale en un certain nombre de paramètres[3]. Pour le moment,

[3] Plus formellement, on dit qu'une classe de concepts $\mathcal{F}$ définie sur un espace d'exemples $\mathcal{X}$ est apprenable avec un espace d'hypothèses $\mathcal{H}$ par un apprenant $\mathcal{A}$ ssi pour tout $f \in \mathcal{F}$, toute distribution $\mathcal{D}_X$ sur $\mathcal{X}$, un taux d'erreur ε tel que $0 < \varepsilon < 1/2$ et un taux de confiance δ tel que $0 < \delta < 1/2$, et à partir d'un échantillon

Copyright © 2010 Eyrolles.

cette caractérisation formelle des apprentissages réalisables a surtout permis de montrer que certains apprentissages étaient non apprenables dans ce cadre. L'accumulation de ces résultats négatifs a lassé même les théoriciens, et ce d'autant plus qu'ils se fondent sur des bornes souvent grossières que la réalité des applications pratiques semble ignorer. Les théorèmes sur les vitesses de convergence dans l'analyse de Vapnik ont remplacé ce type d'investigations.

Cependant, les critères de performances évoqués ci-dessus privilégient le point de vue des systèmes d'apprentissages « à un coup », *batch-learning*, dans lesquels la performance n'est mesurée qu'après l'apprentissage. C'est évidemment très restrictif. La plupart des organismes naturels, les organisations sociales et les institutions, mais aussi certains systèmes artificiels, ne peuvent survivre que s'ils apprennent en permanence et que leur performance tout au long de leur existence est correcte, et pas mesurée seulement une fois à la fin. Il est donc important de définir des mesures de performances qui puissent s'appliquer tout au long de la trajectoire des états suivie par l'apprenant.

Finalement, il faudra bien un jour envisager des mesures de performances plus sophistiquées, prenant en compte à la fois l'intelligibilité des connaissances produites par l'apprenant, mais aussi la manière dont elles peuvent s'inscrire dans les connaissances antérieures, dans les connaissances de la collectivité, humaine ou non, et s'interféconder avec elles. Ce jour-là, l'apprentissage artificiel pourra renouer un dialogue fécond avec d'autres sciences de l'apprentissage, comme la psychologie ou la didactique. Il reste pour cela du chemin à parcourir.

- **Le protocole d'apprentissage**. Il règle le protocole des interactions entre l'apprenant et son environnement, celui-ci incluant éventuellement un oracle ou professeur dans le cas de l'apprentissage supervisé. Nous avons largement examiné les protocoles d'apprentissage supervisé, non supervisé et par renforcement. Ils n'épuisent cependant pas l'ensemble des possibilités et d'autres types d'apprentissages sont envisagés comme l'apprentissage incrémental ou en ligne (*on-line learning*), l'apprentissage actif ou les apprentissages collaboratifs.

- **Le type d'analyse théorique**. La majorité des études théoriques prennent comme base l'hypothèse de données tirées aléatoirement et indépendamment suivant une distribution fixe (tirage i.i.d.). C'est en effet le seul cadre dans lequel on sache établir des théorèmes de convergence uniforme sur des fonctions de distribution. Malheureusement, ou heureusement, l'environnement d'un agent obéit rarement à cette hypothèse. L'agent modifie les distributions de données par son action, la tâche d'apprentissage évolue, la Nature elle-même change. On retombe là sur le probème de la définition d'autres protocoles d'apprentissage et d'autres critères de performance. L'approche théorique de l'apprentissage va devoir regarder ailleurs que dans la théorie statistique.

Notes historiques et bibliographiques

L'historique de l'analyse de l'apprentissage par Vapnik a déjà été abordée dans les chapitres 2 et 9. Nous nous intéressons donc ici aux autres sujets de ce chapitre.

L'idée que l'induction pouvait être vue comme l'approximation d'une fonction multivariable

d'apprentissage de taille m, l'apprenant $\mathcal{A}$ produit avec une probabilité au moins $(1-\delta)$ une hypothèse $h \in \mathcal{H}$ telle que $R_{Réel} \leq \varepsilon$ (où le risque est calculé par un taux d'erreur) en un temps polynomial en $1/\varepsilon$, $1/\delta$, m et $taille(f)$.

Copyright © 2010 Eyrolles.

régulière à partir de données n'est pas nouvelle. Sa formalisation est cependant apparue progressivement et c'est vraiment Girosi et Poggio qui se sont faits les champions de ce point de vue en essayant de montrer que toutes les autres approches théoriques peuvent s'y ramener (voir [GJP95] par exemple). Ils ont notamment étudié les propriétés d'un certain nombre de critères de pénalisation ainsi que des modèles dans lesquels les variables d'entrées sont d'abord prétraitées par des fonctions de base, dont les fonctions à base radiale. Le cours de Girosi au MIT est à cet égard intéressant à consulter.

La théorie de l'estimation bayésienne est bien présentée dans [DHS01, Bis95, Bis06, CL96] avec des détails historiques dans la première référence. La difficulté de sa mise en œuvre lui fait préférer des versions simplifiées (voir les chapitres 2 et 15).

Les liens entre induction et économie d'expression d'un modèle sont très anciens comme le montre le principe du rasoir d'Occam. C'est Solomonoff (1926-2009) [Sol64] qui le premier en 1963 exposa une théorie de l'induction basée sur l'idée d'utilisation d'une probabilité *a priori* liée à la complexité de Kolmogorov. Le principe de longueur de description minimale (*MDLP*) a été introduit indépendamment par Wallace et Boulton [WB85] d'une part, et par Rissanen [Ris78] d'autre part. De nombreux travaux de nature plutôt empirique ont cherché à en tester le champ d'application. Par ailleurs, les débats théoriques actuels portent sur les liens entre le MDLP et la théorie bayésienne : celui-ci est-il premier par rapport à celle-ci ? (voir les passionnants débats sur ce sujet à NIPS-2001).

Résumé

L'analyse de l'induction supervisée nécessite d'établir le lien entre le risque empirique et le risque réel. Plusieurs étapes et directions d'analyse ont été développées au cours des années. Elles font toutes ressortir les nécessaires limites de l'espace des hypothèses considéré et, éventuellement, de l'algorithme d'apprentissage.

À la suite des premiers travaux de Vapnik sur des espaces finis de fonctions (voir chapitre 2), des analyses ont porté sur des espaces de cardinal infini et, particulièrement, sur des fonctions à valeur réelle. Ces études mettent en œuvre des théorèmes de concentration de la mesure.

Généralement, toutes les approches de l'induction mettent en exergue une « régularisation » du principe inductif en pénalisant la considération d'hypothèses à forte dynamique.

Ces analyses ont conduit au développement de nouveaux principes inductifs adaptatifs, dont le *structural risk minimisation*, et la maximisation de la marge.

Des analyses récentes cherchent à prendre en compte également la méthode d'exploration de l'espace des hypothèses conduite par l'algorithme d'apprentissage.

L'avenir est également à des études s'appliquant à des données qui ne soient plus identiquement et indépendamment distribuées, comme dans l'analyse classique.

Copyright © 2010 Eyrolles.

Sixième partie

Annexes techniques

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Chapitre 22

Annexes techniques

1. Exemples de fonctions de perte en induction

Bien que les problèmes d'apprentissage liés à la classification, la régression ou l'estimation de densité soient apparemment très différents, impliquant des espaces d'entrée et de sortie de nature diverse, ils peuvent cependant être analysés à l'intérieur du même cadre d'un problème d'optimisation du risque réel. Il suffit pour cela d'introduire des fonctions de perte adaptées à chaque cas. Cette annexe présente certaines d'entre elles.

1.1 La reconnaissance de formes ou classification

On appelle problème de *discrimination*, ou d'*apprentissage deconcept* un problème d'apprentissage de règle de classification pour lequel l'espace de sortie est binaire : $\mathcal{U} = \{0, 1\}$. Il y a donc seulement deux classes possibles : l'une vérifiant le concept à apprendre et l'autre définissant son opposé. L'espace de sortie de la machine $\mathcal{Y}$ n'a donc besoin de prendre que deux valeurs et l'espace $\mathcal{H}$ des fonctions hypothèses est alors celui des *fonctions indicatrices*, prenant leur valeur dans $\{0, 1\}$). Il est alors courant de prendre une fonction de perte qui mesure l'erreur de classification pour chaque forme présentée à l'apprenant :

$$l(u_i, h(\mathbf{x}_i)) = \begin{cases} 0 & \text{si } u_i = h(\mathbf{x}_i) \\ 1 & \text{si } u_i \neq h(\mathbf{x}_i) \end{cases} \tag{22.1}$$

Avec cette fonction de perte, le risque

$$R_{Réel}(h) = \int_{\mathcal{Z}=\mathbf{x}\times\sqcap} l(u, h(\mathbf{x}))dF(\mathbf{x}, u)$$

mesure la *probabilité de mauvaise classification* (*mis-classification*). Le problème de discrimination est donc celui de l'apprentissage d'une fonction indicatrice minimisant la probabilité d'erreur lorsque la distribution des formes $F(x, u)$ est inconnue et que seul est fourni un échantillon de données.Il est important de noter que ce type de fonction de perte n'est pas forcément celui qui

Copyright © 2010 Eyrolles.

doit être employé pour tous les problèmes de classification. Pour reprendre l'exemple du diagnostic de l'appendicite, il est beaucoup plus coûteux socialement et financièrement de passer à côté d'une appendicite que d'en diagnostiquer une à tort. Il faut donc dans ce cas définir une fonction de perte qui rende compte de cette asymétrie. De même les problèmes de classification impliquant plus que deux classes appellent d'autres fonctions de perte.

1.2 La régression

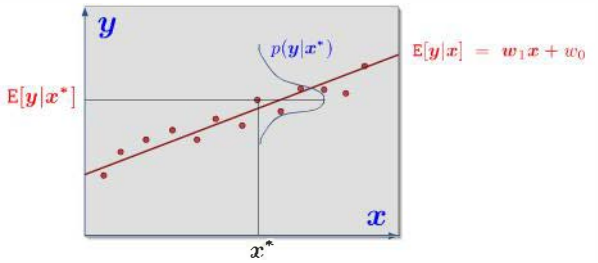
La régression consiste à estimer une fonction f à valeurs réelles, connaissant un échantillon fini de couples $(\mathrm{x}, u = f(\mathrm{x}))$ ou $(\mathrm{x}, u = f(\mathrm{x} + bruit))$ (voir figure 22.1).

La fonction f à estimer peut donc être considérée comme la somme d'une fonction déterministe et d'un signal d'erreur aléatoire de moyenne nulle (et le plus souvent considéré comme une gaussienne).

$$u = f(\mathrm{x}) + \epsilon \tag{22.2}$$

On peut aussi décrire ce phénomène en considérant que la fonction déterministe est la moyenne de la probabilité conditionnelle sur l'espace de sortie $\mathcal{U}$.

$$f(\mathrm{x}) = \int u\, p(u|\mathrm{x})\, du \tag{22.3}$$



TAB. 22.1: *La régression linéaire en supposant un bruit gaussien de moyenne nulle.*

L'espace des fonctions hypothèses $\mathcal{H}$ de l'apprenant peut ou non inclure l'espace des fonctions cibles $\mathcal{F}$. Une fonction de perte usuelle pour la régression est la fontion erreur quadratique (L_2) :

$$L(u_i, h(\mathrm{x}_i)) = (u_i - h(\mathrm{x}_i))^2 \tag{22.4}$$

L'apprentissage consiste alors à trouver la fonction $h \in \mathcal{H}$ minimisant la fonctionnelle de risque :

$$R_{Réel}(h) = \int_{\mathcal{Z}=\mathcal{X}\times\mathcal{U}} (u - h(\mathrm{x}))^2 dF(\mathrm{x}, u) \tag{22.5}$$

sur la seule base de l'échantillon d'apprentissage. Cette fonctionnelle, le *risque réel*, mesure la précision des prédictions de l'apprenant.

Remarque

Sous l'hypothèse que le signal d'erreur est une gaussienne centrée en 0, ce risque peut aussi être écrit en fonction de la capacité de l'apprenant à approximer la fonction cible $f(x)$ (et non la sortie u), comme le montre le calcul suivant :

$$\begin{aligned} R_{Réel}(h) &= \int (u - f(\mathrm{x}) + f(\mathrm{x}) - h(\mathrm{x}))^2\, p(\mathrm{x}, u)\, d\mathrm{x}du \\ &= \int (u - f(\mathrm{x}))^2\, d\mathrm{x}du + \int (h(\mathrm{x}) - f(\mathrm{x}))^2\, p(\mathrm{x})\, d\mathrm{x} \\ &\quad + 2\int (u - f(\mathrm{x}))(f(\mathrm{x}) - h(\mathrm{x}))\, p(\mathrm{x}, u)\, d\mathrm{x}du \end{aligned} \tag{22.6}$$

Copyright © 2010 Eyrolles.

Sous l'hypothèse que le bruit est de moyenne nulle, le dernier terme dans la somme ci-dessus s'écrit :

$$\begin{aligned}\int (u - f(\mathbf{x}))(f(\mathbf{x}) - h(\mathbf{x}))\ \mathbf{p}(\mathbf{x}, u)\ d\mathbf{x}du &= \int (\epsilon(f(\mathbf{x}) - h(\mathbf{x}))\ \mathbf{p}(u|\mathbf{x})\ \mathbf{p}(\mathbf{x})\ d\mathbf{x}du \\ &= \int (f(\mathbf{x}) - h(\mathbf{x}))\left[\int \epsilon \mathbf{p}(u|\mathbf{x})du\right]\ \mathbf{p}(\mathbf{x})\ d\mathbf{x} \\ &= \int (f(\mathbf{x}) - h(\mathbf{x}))E - \epsilon(\epsilon|\mathbf{x})\ \mathbf{p}(\mathbf{x})\ d\mathbf{x} \quad = \quad 0\end{aligned} \tag{22.7}$$

Le risque peut donc être réécrit comme :

$$R_{Réel}(h) = \int (u - f(\mathbf{x}))^2\ \mathbf{p}(\mathbf{x}, \mathbf{u})\ d\mathbf{x}du + \int (h(\mathbf{x}) - f(\mathbf{x}))^2\ \mathbf{p}(\mathbf{x})\ d\mathbf{x} \tag{22.8}$$

Le premier terme ne dépend pas de la fonction d'approximation h et peut être écrit en fonction de la variance sur le bruit :

$$\begin{aligned}\int (u - f(\mathbf{x}))^2 \mathbf{p}(\mathbf{x}, u)\ d\mathbf{x}du &= \int \epsilon^2 \mathbf{p}(u|\mathbf{x})\ \mathbf{p}(\mathbf{x})\ d\mathbf{x}du \\ &= \int \left[\int \epsilon^2 \mathbf{p}(u|\mathbf{x})du\right]\ \mathbf{p}(\mathbf{x})\ d\mathbf{x} \\ &= \int E_\epsilon(\epsilon^2|\mathbf{x})\ \mathbf{p}(\mathbf{x})\ d\mathbf{x}\end{aligned} \tag{22.9}$$

En substituant (22.9) dans (22.8), on obtient :

$$R_{Réel}(h) = \int E_\epsilon(\epsilon^2|\mathbf{x})\ \mathbf{p}(\mathbf{x})\ d\mathbf{x} + \int (h(\mathbf{x}) - f(\mathbf{x}))^2\ \mathbf{p}(\mathbf{x})\ d\mathbf{x} \tag{22.10}$$

Le risque pour le problème de régression (sous l'hypothèse de fonction de perte L_2 et de bruit de moyenne nulle) a donc une contribution exprimant la variance du bruit et une contribution exprimant la précision de la fonction d'approximation. Comme la variance du bruit est indépendante de la fonction d'approximation, la minimisation du second terme de (22.10) est équivalent à la minimisation du risque(voir le paragraphe 22.5). Ainsi, chercher à obtenir le risque de prédiction optimal est équivalent à la recherche de l'approximation la plus précise de la fonction cible inconnue f.

1.3 L'estimation de densité

Un autre problème inductif important consiste à estimer une densité de probabilités dans l'espace d'entrée $\mathcal{X}$ à partir d'un échantillon de données $\{x_i\}_{1 \leq i \leq m}$. Dans ce cas, il n'y a pas nécessité de considérer un espace de sortie, et la sortie $h(\mathbf{x})$ de l'apprenant représente une densité sur $\mathcal{X}$. La fonction de perte usuelle dans ce cas est la fonction :

$$l(h(\mathbf{x})) = -\ln h(\mathbf{x}) \tag{22.11}$$

donnant la fonctionnelle de risque :

$$R_{Réel}(h) = \int -\ln h(\mathbf{x})\ \mathbf{p}(\mathbf{x})d\mathbf{x} \tag{22.12}$$

Il est établi que la densité optimale h^* minimise cette fonctionnelle de risque. Par ailleurs, si la densité cible $f \notin \mathcal{H}$, alors on peut montrer que la solution h minimisant l'espérance de risque ou risque réel est caractérisable : c'est celle dont la *divergence de Kullback-Leibler* avec la vraie densité f est la plus faible (voir la définition de la divergence de Kullback-Leibler dans la section 1.3.5 du chapitre 3).

Copyright © 2010 Eyrolles.

2. Le calcul de l'intervalle de confiance de l'estimation de la probabilité d'une règle de classification

Étant donnés un taux d'erreur réel $R_{Réel}(h)$, et un échantillon de test de taille t, le nombre d'erreurs t_{err} mesuré sur un échantillon $\mathcal{T}$ distribué de manière i.i.d. (suivant la même distribution que l'échantillon d'apprentissage $\mathcal{S}$) suit une loi binomiale :

$$\begin{aligned} P(t_{err}|R_{Réel}(h),t) &= \mathcal{B}(t_{err}|R_{Réel}(h),t) \\ &\triangleq \frac{t_{err}\,!\,(t-t_{err})\,!}{t\,!} R_{Réel}(h)^{t_{err}}\,(1-R_{Réel}(h))^{(t-t_{err})} \end{aligned}$$

L'expression ci-dessus donne la probabilité que t_{err} observations dans un échantillon de test de taille t soient mal classées, étant donné que le taux d'erreur réel est $R_{Réel}(h)$. En utilisant le théorème de Bayes, nous pouvons « retourner » cette équation pour obtenir ce qui nous intéresse, c'est-à-dire une estimation du taux d'erreur réel connaissant le nombre d'erreurs mesuré sur $\mathcal{T}$:

$$P(R_{Réel}(h)|t_{err},t) = \frac{P(t_{err}|R_{Réel}(h),t)}{\int P(t_{err}|R_{Réel}(h),t)}$$

En supposant que $P(R_{Réel}(h),t)$ ne varie pas avec $R_{Réel}(h)$ et que $P(t_{err}|R_{Réel}(h),t)$ suit une distribution binomiale, nous avons une distribution bêta pour $R_{Réel}(h)$:

$$\begin{aligned} P(R_{Réel}(h)|t_{err},t) &= B_e(R_{Réel}(h))|t_{err}+1, t-t_{err}+1) \\ &\triangleq \frac{R_{Réel}(h)^{t_{err}}\,(1-R_{Réel}(h))^{(t-t_{err})}}{\int R_{Réel}(h)^{t_{err}}\,(1-R_{Réel}(h))^{(t-t_{err})} d_{R_{Réel}(h)}} \end{aligned}$$

où $B_e(x|\alpha,\beta) = [\Gamma(\alpha+\beta)/(\Gamma(\alpha)\Gamma(\beta))]\,x^{\alpha-1}\,(1-x)^{\beta-1}$. Cette densité de probabilité *a posteriori* fournit toute l'information qui peut être tirée de l'erreur mesurée $\widehat{R}_{Réel}(h)$. On utilise cependant généralement un résumé de cette information sous la forme d'intervalles de confiance autour d'une valeur estimée.

Si les échantillons d'apprentissage et de test aléatoires sont indépendants alors la précision de l'estimation ne dépend que du nombre t d'exemples de l'ensemble de test et de la valeur de $\widehat{R}_{Réel}(h)$. Une approximation suffisante dans le cas où t est assez grand (au delà de la centaine) donne l'*intervalle de confiance* de $\widehat{R}_{Réel}(h)$ à $x\%$ par la formule :

$$\left[\ \frac{t_{err}}{t} \pm \zeta(x)\sqrt{\frac{\frac{t_{err}}{t}\,(1-\frac{t_{err}}{t})}{t}}\ \right]$$

La fonction $\zeta(x)$ a en particulier les valeurs suivantes :

x	50 %	68 %	80 %	90 %	95 %	98 %	99 %
$\zeta(x)$	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Copyright © 2010 Eyrolles.

3. Estimation d'une densité de probabilité en un point.

Soit une distribution de probabilités de densité $\mathbf{p}(\mathbf{x})$ dans $\mathbb{R}^d$. La probabilité P qu'un vecteur tiré selon cette distribution se trouve à l'intérieur d'une région fermée $\mathcal{R}$ vaut :

$$P = \int_{\mathcal{R}} \mathbf{p}(\mathbf{x}) d\mathbf{x} \tag{22.13}$$

Soit $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\}$ un ensemble de m points tirés indépendamment selon $\mathbf{p}$. La probabilité que k_m d'entre eux se trouvent à l'intérieur de la région $\mathcal{R}$ est donnée par :

$$\binom{m}{k_m} P^{k_m} (1-P)^{k_m}$$

Soit k_m/m la variable aléatoire qui mesure la proportion du nombre de points qui sont dans la région $\mathcal{R}$. Cette variable aléatoire dépend de la loi binômiale donnée ci-dessus. Sa moyenne et sa variance sont connues :

$$E(k_m/m) = P \tag{22.14}$$

$$E(k_m/m - P)^2 = \frac{P(1-P)}{m}$$

Par la loi des grands nombres, nous savons que :

$$k_m/m \text{ converge vers } P \text{ quand } m \text{ augmente} \tag{22.15}$$

k_m/m est un bon estimateur de P, dans la mesure où il est non biaisé et où sa variance diminue rapidement avec l'augmentation de m.

D'autre part, en supposant que $\mathbf{p}$ est continue dans la région $\mathcal{R}$ et que cette région est assez petite autour du point $\mathbf{x}$ pour que $\mathbf{p}$ n'y varie presque pas, on a, en notant V le volume de la région $\mathcal{R}$:

$$P = \int_R \mathbf{p}(\mathbf{x}) d\mathbf{x} \approx \mathbf{p}(\mathbf{x}) \int_R d\mathbf{x} = \mathbf{p}(\mathbf{x}) V \tag{22.16}$$

Les équations précédentes (22.15 et 22.16) mènent, pour m assez grand, à :

$$\mathbf{p}(\mathbf{x}) \approx \frac{P}{V} \approx \frac{k_m/m}{V}$$

En pratique, pour assurer la convergence quand m augmente, on fait dépendre la région $\mathcal{R}$ du nombre de points m : on la note maintenant $\mathcal{R}_m$ et son volume V_m. On a donc l'estimation suivante :

$$\mathbf{p}(\mathbf{x}) \approx \frac{k_m/m}{V_m}$$

Pour que cette estimation converge vers $\mathbf{p}(\mathbf{x})$ quand m augmente, il faut :

- assurer que l'approximation de l'équation 22.16 reste valide, donc avoir :

$$\lim_{m \to \infty} V_m = 0$$

- assurer que celle de l'équation 22.15 le reste aussi, donc avoir :

$$\lim_{m \to \infty} k_m = \infty$$

- enfin, assurer que $p(\mathbf{x})$ reste à une valeur finie :

$$\lim_{m \to \infty} \frac{k_m}{m} = 0$$

Copyright © 2010 Eyrolles.

4. L'estimation des paramètres d'une distribution gaussienne.

On se place ici, pour éviter des formules fastidieuses, dans un espace monodimensionnel. Une distribution gaussienne dépend alors seulement de deux paramètres scalaires : sa moyenne μ et sa variance σ. La variable x est aussi un scalaire, c'est pourquoi elle n'est pas notée ici comme un vecteur x. Elle s'écrit :

$$\mathcal{N}(x,\mu,\sigma) = (\frac{1}{\sqrt{2\pi\sigma}})exp((-1/2)(\frac{x-\mu}{\sigma})^2$$

ou :

$$Log(\mathcal{N}(x,\mu,\sigma)) = -Log(\sigma) - Log(2\pi) - \frac{(x-\mu)^2}{2\sigma^2}$$

L'estimateur au maximum de vraisemblance suppose que tous les échantillons d'apprentissage $\mathcal{S} = \{x_1, x_2, ..., x_m\}$ sont tirés indépendamment. On cherche donc dans l'ensemble des distributions gaussiennes de paramètres σ et μ celle qui maximise le produit de ses valeurs sur les points d'apprentissage. Ceci est équivalent à trouver les paramètres pour lesquels l'expression :

$$T(\mu,\sigma,m) = \sum_{k=1}^{m} Log(\mathcal{N}(x_k,\mu,\sigma))$$

est maximale. On est donc amené à chercher les valeurs μ et σ telles que :

$$\frac{\partial}{\partial\mu}T(m,\sigma,m) = 0$$

$$\frac{\partial}{\partial\sigma}T(\mu,\sigma,m) = 0$$

d'où :

$$\sum_{k=1}^{m}\frac{x_k-\mu}{\sigma^2} = 0$$

$$-m/\sigma + \sum_{k=1}^{m}\frac{x_k-\mu}{\sigma^2} = 0$$

Finalement, la solution au problème de l'estimation au maximum de vraisemblance est donnée par les valeurs suivantes :

$$\widehat{\mu} = \frac{1}{m}\sum_{k=1}^{m} x_k$$

et :

$$\widehat{\sigma} = \frac{1}{m}\sum_{k=1}^{m}(x_k - \widehat{\mu})$$

En revenant au cas général multiclasse en dimension d, il faut estimer au maximum de vraisemblance pour chaque classe de numéro i un *vecteur moyenne* μ_i et une *matrice de covariance* Q_i ; on retrouverait, en développant des calculs analogues, les formules :

$$\widehat{\mu_i} = \frac{1}{m_i}\sum_{k=1}^{m_i} x_k$$

et :

$$\widehat{Q_i} = \frac{1}{m_i}\sum_{k=1}^{m_i}(x_k - \widehat{\mu_i})(x_k - \widehat{\mu_i})^T$$

Copyright © 2010 Eyrolles.

5. Pourquoi et comment la règle du PPV converge-t-elle ?

5.1 Pourquoi ?

Soit $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\}$ l'échantillon d'apprentissage, $\mathbf{x}$ le point dont on cherche la classe par la règle du $1 - ppv$ et $\mathbf{x}_0$ le point de X le plus proche de $\mathbf{x}$. Notons $B(\mathbf{x}, \rho)$ la sphère de rayon ρ centrée en $\mathbf{x}$. La probabilité qu'un point de $\mathcal{S}$ se trouve dans $B(\mathbf{x}, \rho)$ vaut :

$$\mathbf{P}(\rho) = \int_{B(\mathbf{x},\rho)} \mathbf{p}(\mathbf{x})d\mathbf{x}$$

La probabilité qu'aucun point de l'ensemble d'apprentissage $\mathcal{S}$ ne se trouve dans $B(\mathbf{x}, \rho)$ est égale à $(1 - \mathbf{P}(\rho))^m$ qui tend vers zéro quand m augmente. Par conséquent, $\mathbf{x}_0$ tend en probabilité vers $\mathbf{x}$, ce qui assure la convergence désirée.

5.2 Comment ?

L'erreur moyenne réalisée par la règle du $1 - ppv$ peut se calculer en remarquant qu'en attribuant à $\mathbf{x}$ la classe de $\mathbf{x}_0$, on commet l'erreur :

$$err_{ppv} = \mathbf{P}[\omega(\mathbf{x}) \neq \omega(\mathbf{x}_0)] = \sum_{i=1}^{C} \mathbf{P}[\mathbf{x} \in \omega_i, \mathbf{x}_0 \notin \omega_i] = \sum_{i=1}^{C} \mathbf{P}(\omega_i \mid \mathbf{x})[1 - \mathbf{P}(\omega_i \mid \mathbf{x}_0)]$$

Quand la taille m de l'ensemble d'apprentissage augmente, on a vu ci-dessus que $\mathbf{x}_0$ tend en probabilité vers $\mathbf{x}$, ce qui implique que $\mathbf{P}(\omega_i \mid \mathbf{x}_0)$ tend vers $\mathbf{P}(\omega_i \mid \mathbf{x})$. Par conséquent :

$$\lim_{m \to \infty} err_{ppv} = \int_{\mathbb{R}^d} \sum_{i=1}^{C} \mathbf{P}(\omega_i \mid \mathbf{x})[1 - \mathbf{P}(\omega_i \mid \mathbf{x})]\mathbf{p}(\mathbf{x})d\mathbf{x}$$

Prenons le cas à deux classes ($C = 2$). Soit $\omega_1(\mathbf{x})$ la classe que donnerait la décision bayésienne et $\omega_2(\mathbf{x})$ l'autre. L'erreur bayésienne vaut :

$$err^* = err(\omega_1) = \int_{\mathbb{R}^d} \mathbf{p}(\omega_2(\mathbf{x}) \mid \mathbf{x})\mathbf{p}(\mathbf{x})d\mathbf{x}$$

L'erreur par la décision du 1-ppv, en vertu du calcul précédent, vaut pour $C = 2$:

$$err_{ppv} = \mathbf{P}(\omega_1 \mid \mathbf{x})[1 - \mathbf{P}(\omega_1 \mid \mathbf{x})] + \mathbf{P}(\omega_2 \mid \mathbf{x})[1 - \mathbf{P}(\omega_2 \mid \mathbf{x})]$$

d'où :

$$err_{ppv} = 2\mathbf{P}(\omega_2(\mathbf{x}) \mid \mathbf{x})[1 - \mathbf{P}(\omega_2(\mathbf{x}) \mid \mathbf{x})] \leq 2\mathbf{P}(\omega_2(\mathbf{x}) \mid \mathbf{x})$$

D'où la formule dans le cas de deux classes :

$$\lim_{m \to \infty} err_{ppv} \leq 2err^*$$

Copyright © 2010 Eyrolles.

6. Pourquoi la règle de décision bayésienne est-elle optimale ?

Appelons h^* la règle de décision bayésienne, celle qui affecte au point $\mathbf{x}$ la classe :

$$\omega^*(\mathbf{x}) = \underset{\omega_i}{\text{ArgMax}}(\mathbf{P}(\omega_i \mid \mathbf{x}))$$

Soit h_0 la règle idéale de décision, affectant $\mathbf{x}$ à sa vraie classe $h_0(\mathbf{x})$ et soit h une règle de décision quelconque affectant $\mathbf{x}$ à la classe $h(\mathbf{x})$. On cherche à montrer que l'erreur moyenne $err(h^*)$ commise par h^* est inférieure ou égale à $err(h)$, commise par h, soit :

$$\int_{\mathbb{R}^d} \mathbf{P}(h^*(\mathbf{x}) \neq h_0(\mathbf{x}))\, \mathbf{p}(\mathbf{x})\, d\mathbf{x} \leq \int_{\mathbb{R}^d} \mathbf{P}(h(\mathbf{x}) \neq h_0(\mathbf{x}))\, \mathbf{p}(\mathbf{x})\, d\mathbf{x}$$

Or, pour toute règle h :

$$\mathbf{P}(h(\mathbf{x}) \neq h_0(\mathbf{x})) = 1 - \mathbf{P}(h(\mathbf{x}) \mid \mathbf{x})$$

donc :

$$err(h) = 1 - \int_{\mathbb{R}^d} \mathbf{P}(h(\mathbf{x}) \mid \mathbf{x})\, \mathbf{p}(\mathbf{x})\, d\mathbf{x}$$

De la définition de la règle h^*, on déduit immédiatement :

$$\forall \mathbf{x} \in \mathbb{R}^d, \quad \mathbf{P}(h^*(\mathbf{x}) \mid \mathbf{x}) \geq \mathbf{P}(h(\mathbf{x}) \mid \mathbf{x})$$

Par conséquent, la valeur :

$$err(h) - err(h^*) = 1 - \int_{\mathbb{R}^d} (\mathbf{P}(h(\mathbf{x}) \mid \mathbf{x}) - \mathbf{P}(h(\mathbf{x}) \mid \mathbf{x}))\, \mathbf{p}(\mathbf{x})\, d\mathbf{x}$$

est toujours positive ou nulle, ce qui établit la relation cherchée.

7. Apprentissage par estimation-maximisation

L'algorithme *estimation-maximisation* (*EM*) est une procédure générale pour apprendre la valeur de paramètres cachés de certains processus probabilistes. C'est en particulier en l'utilisant que les paramètres des HMM peuvent être appris (voir le chapitre 12). Mais cette procédure est aussi utile dans l'apprentissage des réseaux bayésiens (chapitre 11), dans la classification non supervisée (chapitre 18) et dans l'apprentissage semi-supervisée (chapitre 19). Nous allons le présenter ici sur un exemple simple, puis revenir sur son application à l'apprentissage des HMM et au mélange de distributions gaussiennes.

7.1 Un exemple

Nous disposons de deux pièces de monnaie truquées A et B, avec des probabilités respectives $\mathbf{P}_A$ et $\mathbf{P}_B$ de tomber sur *Pile* et $1 - \mathbf{P}_A$ et $1 - \mathbf{P}_B$ de tomber sur *Face*. Nous connaissons les valeurs $\mathbf{P}_A$ et $\mathbf{P}_B$. Nous demandons à un huissier de procéder en secret aux opérations suivantes :

- tirer un nombre μ au hasard entre 0 et 1 ;
- répéter N fois les manipulations suivantes :
  - choisir la pièce A avec la probabilité μ ou la pièce B avec la probabilité $(1 - \mu)$;

Copyright © 2010 Eyrolles.

- lancer la pièce choisie ;
- enregistrer le résultat du lancer : *Pile* ou *Face*.

Une fois l'affaire terminée, l'huissier nous communique la séquence des N valeurs *Pile* ou *Face* qu'il a notées. Notre problème est alors le suivant : estimer la valeur du paramètre caché μ à partir de la suite $\mathcal{O} = (O_1, \ldots O_N)$ de ces observations. Chaque observation O_i vaut donc *Pile* ou *Face*. Si nous connaissions quelle pièce a été utilisée à chaque lancer, nous pourrions estimer μ par la valeur :

$$\frac{N_A}{N} = \frac{\text{Nombre de lancers de la pièce } A}{N}$$

Mais ce n'est pas le cas : nous ne connaissons que le résultat des tirages. Nous pouvons cependant calculer μ par une technique itérative, *l'algorithme EM*[1].

7.2 Application de l'algorithme *EM* à l'exemple

Initialisation

Donner une valeur arbitraire μ_0 strictement comprise entre 0 et 1.

Étape t

- ***Estimation***

  On utilise les observations $\mathcal{O}$ pour calculer une estimation de N_A.

  On note :

  - μ_t l'estimation courante (à l'étape t) de μ ;
  - $\mathsf{P}_A(O_i)$ la probabilité de l'évènement O_i sachant que la pièce A était utilisée ;
  - $\mathsf{P}_B(O_i)$ la probabilité de l'évènement O_i sachant que la pièce B était utilisée ;
  - $E_t(A \mid \mathcal{O})$ l'estimation courante du nombre N_A de tirages de la pièce A ;

  Cette dernière valeur peut se calculer en fonction des trois précédentes par l'expression :

$$E_t(A \mid \mathcal{O}) = \sum_{i=1}^{N} \frac{\mu_t \mathsf{P}_A(O_i)}{\mu_t \mathsf{P}_A(O_i) + (1 - \mu_t)\mathsf{P}_B(O_i)}$$

  Mais les valeurs $\mathsf{P}_A(O_i)$ et $\mathsf{P}_B(O_i)$ sont inconnues. L'estimation de leurs valeurs est le cœur de l'algorithme.

- ***Maximisation***

  On calcule une nouvelle estimation de μ :

$$\mu_{t+1} = \frac{E_t(A \mid \mathcal{O})}{N}$$

Test d'arrêt

$$\mu_{t+1} \approx \mu_t$$

Pour résoudre le problème, il faut donc déterminer les valeurs $\mathsf{P}_A(O_i)$ et $\mathsf{P}_B(O_i)$. On utilise pour cela la notion de *statistique suffisante*.

7.3 Statistique suffisante

Pour notre exemple, l'ordre dans lesquels les O_i sont apparus dans l'ensemble $\mathcal{O}$ des observations n'a pas d'importance : le nombre P d'observations *Pile* dans $\mathcal{O}$ suffit. On dit qu'il s'agit

[1] E pour estimation (ou en anglais *Expectation*), M pour maximisation.

Copyright © 2010 Eyrolles.

d'une *statistique suffisante* pour estimer le paramètre caché μ. Cette observation permet de calculer explicitement $E_t(\mathcal{A} \mid \mathcal{O})$ de la façon suivante. On décompose d'abord la formule précédente sur les tirages ayant donné *Pile* et ceux ayant donné *Face*, en nombres respectifs P et F. C'est-à-dire que $\mathbf{P}_A(O_i)$ devient soit $\mathbf{P}_A(Pile)$, soit $\mathbf{P}_A(Face)$.

$$\begin{aligned} E_t(A \mid \mathcal{O}) &= \sum_{i=1}^{P} \frac{\mu_t \mathbf{P}_A(Pile)}{\mu_t \mathbf{P}_A(Pile) + (1-\mu_t)\mathbf{P}_B(Pile)} \\ &\quad + \sum_{i=1}^{F} \frac{\mu_t \mathbf{P}_A(Face)}{\mu_t \mathbf{P}_A(Face) + (1-\mu_t)\mathbf{P}_B(Face)} \end{aligned}$$

En remarquant que $\mathbf{P}_A(Pile) = \mathbf{P}_A$ et $\mathbf{P}_A(Face) = 1 - \mathbf{P}_A$ et en utilisant le fait que les tirages sont indépendants :

$$\begin{aligned} E_t(A \mid \mathcal{O}) &= \sum_{i=1}^{P} \frac{\mu_t \mathbf{P}_A}{\mu_t \mathbf{P}_A + (1-\mu_t)\mathbf{P}_B} + \sum_{i=1}^{F} \frac{\mu_t (1-\mathbf{P}_A)}{\mu_t \mathbf{P}_A + (1-\mu_t)\mathbf{P}_B} \\ &= P\frac{\mu_t \mathbf{P}_A}{\mu_t \mathbf{P}_A + (1-\mu_t)\mathbf{P}_B} + F\frac{\mu_t (1-\mathbf{P}_A)}{\mu_t \mathbf{P}_A + (1-\mu_t)\mathbf{P}_B} \end{aligned}$$

On a donc maintenant une formule pour calculer $E_t(A \mid \mathcal{O})$ en fonction de μ_t et de valeurs connues.

7.4 Plus généralement

Algorithme 47 : Algorithme EM

début
  Définir une statistique suffisante pour estimer l'ensemble Λ des paramètres cachés ;
  Initialiser Λ ;
  $t \leftarrow 1$;
  tant que $\Lambda_t \neq \Lambda_{t+1}$ **faire**
    Estimation
    Utiliser les observations pour calculer la statistique suffisante de Λ_t ;
    Maximisation
    Calculer Λ_{t+1} comme une estimation au maximum de vraisemblance de Λ à partir des résultats de l'étape t d'estimation ;
  fin tant que
  $t \leftarrow t + 1$
fin

On démontre que l'algorithme *EM* décrit dans l'algorithme 47 fait croître la vraisemblance de Λ_p vis-à-vis des données. Par conséquent, il converge vers un optimum local.

7.5 Retour sur l'exemple

Prenons $\mathbf{P}_A = 0.2$ et $\mathbf{P}_B = 0.6$. L'huissier effectue $N = 100$ tirages et fournit une série $\mathcal{O}$ d'observations comportant $P = 50$ *Pile* et $F = 50$ *Face*. Quelle est la valeur de μ que produit

Copyright © 2010 Eyrolles.

l'algorithme *EM*? La récurrence de base est la suivante, en regroupant les deux phases en une seule :

$$\mu_{t+1} = \frac{1}{P+F}\left(P\frac{\mu_t \mathbf{P}_A}{\mu_t \mathbf{P}_A + (1-\mu_t)\mathbf{P}_B} + F\frac{\mu_t(1-\mathbf{P}_A)}{\mu_t(1-\mathbf{P}_A) + (1-\mu_t)(1-\mathbf{P}_B)}\right)$$

D'où, pour deux initialisations de μ :

μ_0	0.800	0.100
μ_1	0.730	0.109
μ_2	0.659	0.118
μ_3	0.593	0.127
μ_4	0.536	0.135
μ_5	0.488	0.144
μ_{10}	0.351	0.183
μ_{20}	0.273	0.228
μ_{30}	0.256	0.243
μ_{40}	0.252	0.248
μ_{50}	0.250	0.249
μ_{60}	0.250	0.250

Ce résultat est conforme à l'intuition : il faut que l'huissier ait tiré en moyenne 4 fois plus souvent la pièce B que la pièce A pour avoir rétabli l'équilibre entre le nombre de *Pile* et celui de *Face*. Il se trouve que, pour cet exemple très simple, on peut en réalité estimer directement μ car la proportion de *Pile* est en effet une estimation de :

$$\mu\,\mathbf{P}_A + (1-\mu\,\mathbf{P}_B)$$

Dans notre application numérique :

$$1/2 = \mu\,0.2 + (1-\mu)\,0.6$$

d'où :

$$\mu = 0.25$$

7.6 L'apprentissage des paramètres des HMM

Les formules de réestimation des paramètres de modèles de Markov cachés donnés au chapitre 12, bien que beaucoup plus compliquées en apparence, ne sont pas fondamentalement différentes de celles présentées ci-dessus.

Prenons le cas de la réestimation d'une probabilité de transition, donnée par la formule[2] :

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{N}\sum_{t=1}^{|\mathbf{O}^k|-1}\xi_t^k(i,j)}{\sum_{k=1}^{N}\sum_{t=1}^{|O^k|-1}\gamma_t^k(i)}$$

avec :

- $\xi_t^k(i,j)$ la probabilité, étant donnés une phrase O^k et un HMM λ, que ce soit l'état s_i qui ait émis la lettre de rang t de $\mathbf{O}^k$ et l'état s_j qui ait émis celle de rang $t+1$;

[2] On se reportera au Chapitre 12 pour le détail des notations.

Copyright © 2010 Eyrolles.

- $\gamma_t^k(i)$la probabilité que la lettre de rang t de la phrase O^k ait été émise par l'état s_j.

$\overline{a}_{ij}$, la valeur réestimée de a_{ij}, est une composante du paramètre caché $\Lambda = (A, B, \pi)$ qui définit le HMM cherché. Elle est calculée par un comptage sur la base d'apprentissage $\mathcal{O}$; ce comptage calcule une statistique suffisante pour estimer la probabilité qu'est réellement a_{ij}. En effet, $\overline{a}_{ij}$ compte la proportion des transitions empruntées entre s_i et s_j parmi toutes celles qui quittent s_i, dans l'émission de chaque phrase O^k de $\mathcal{O}$. La normalisation assure que sur leur ensemble, les $\overline{a}_{ij}$ somment à 1.

De la sorte, l'algorithme *EM* réestime tous les paramètres cachés du HMM cherché. Comme les formules de réestimation sont conformes aux hypothèses, la convergence vers un optimum (éventuellement local)est assurée.

7.7 L'apprentissage des paramètres de distributions multigaussiennes

On dispose d'une collection $\mathcal{S} = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$ d'exemples, qui sont des vecteurs de $\mathbb{R}^d$. On fait l'hypothèse que ces vecteurs sont des tirages aléatoires d'un *mélange* de k distributions gaussiennes $\mathcal{N}_1$,...$\mathcal{N}_k$. On cherche à estimer les paramètres de chaque distribution, ainsi que la façon dont elle sont mélangées.

Le tirage d'un exemple pourrait se décrire ainsi : d'abord, choisir aléatoirement une des k distributions. Ensuite, tirer l'exemple selon cette distribution. Par conséquent, pour définir le mélange, il suffit de se donner les k valeurs qui sont les probabilités que l'une des k distributions soit tirée.

Un exemple $\mathbf{x}_i$ doit donc se décrire de manière plus complète par

$$y_i = (\mathbf{x}_i, \mathbf{z}_{i1}, ..., \mathbf{z}_{ik})$$

où :

- $\mathbf{x}_i$ est le vecteur observé ;
- pour $j = 1, k$, $\mathbf{z}_{ij}$ vaut 1 si $\mathbf{x}_i$ a été généré par $\mathcal{N}_i$. Les valeurs $\mathbf{z}_{ij}$ sont cachées à l'observateur.

Pour simplifier, nous prenons $d = 1$. Nous supposons aussi que les k distributions, de moyennes $\mu_1, ..., \mu_k$, ont la même variance σ. La méthode se généralise sans trop de difficultés à d quelconque et à des matrices de covariances différentes pour chaque gaussienne. L'algorithme *EM* s'applique comme décrit dans l'algorithme 48. Il a pour résultat le vecteur $h = (\mu_1, ..., \mu_k)$ et les estimations des valeurs $\mathbf{z}_{i1}$, $\mathbf{z}_{ik}$. Ces dernières quantités sont donc les probabilités avec lesquelles on tire les gaussiennes $\mathcal{N}_j$, pour $j = 1, k$.

8. Optimisation par descente de gradient

Introduction

Soit une fonction réelle $f(u)$, avec $u \in D \subset \mathbb{R}^d$. On cherche à trouver la valeur u^* du vecteur u telle que :

$$f(u^*) = \underset{u \in D}{\text{ArgMin}} \quad f(u)$$

Autrement dit, on cherche une valeur u^* pour laquelle la fonction f prend la valeur minimum sur son domaine. Les méthodes d'optimisation itérative consistent, d'une manière générale, à partir d'une valeur u^0, puis à construire une suite de valeurs u^n telles que :

$$f(u^{n+1}) \leq f(u^n)$$

Copyright © 2010 Eyrolles.

Algorithme 48 : Mélange de k gaussiennes

début

Initialiser aléatoirement $h = (\mu_1, ..., \mu_k)$

tant que *le processus n'a pas convergé* **faire**

Estimation

Calculer les estimations $E(\mathbf{z}_{i1})$ de $\mathbf{z}_{i1}$, ... $E(\mathbf{z}_{ik})$ de $\mathbf{z}_{ik}$en supposant que l'hypothèse courante sur h est la bonne :

Estimation

for $j = 1, k$ **do**

$$E(\mathbf{z}_{ij}) \leftarrow \frac{\mathbf{p}(x = \mathbf{x}_i \mid \mu = \mu_{,j})}{\sum_{n=1}^{k} \mathbf{p}(x = \mathbf{x}_i \mid \mu = \mu_n)}$$

Donc :

$$E(\mathbf{z}_{ij}) \leftarrow \frac{e^{-\frac{1}{2\sigma^2}(\mathbf{x}_i - \mu_j)^2}}{\sum_{n=1}^{k} e^{-\frac{1}{2\sigma^2}(\mathbf{x}_i - \mu_n)^2}}$$

end

Maximisation

Calculer une nouvelle estimation de $h = (\mu_1, .., \mu_k)$ en supposant que $\mathbf{z}_{ij}$ est égale à $E(\mathbf{z}_{ij})$

for $j = 1, k$ **do**

$$\mu_j \leftarrow \frac{\sum_{i=1}^{N} E(\mathbf{z}_{ij})\mathbf{x}_i}{\sum_{i=1}^{N} E(\mathbf{z}_{ij})}$$

end

fin tant que

fin

La construction d'une telle suite permettra de trouver u^* si F remplit des conditions que nous allons énoncer au fur et à mesure.

Définition

Notons $\frac{\partial}{\partial u}(u^n)$ la valeur du gradientde f au point u^n. Rappelons que cette valeur est un vecteur de $\mathbb{R}^d$. Les techniques de *descente de gradient* sont des méthodes d'optimisation itérative pour lesquelles on choisit

$$u^{n+1} = u^n - \alpha_n \frac{\partial}{\partial u}(u^n) \tag{22.17}$$

avec α_n réel positif. En particulier, si l'on choisit α_n de sorte que :

$$f(u^{n+1}) = \inf_{\alpha} \left(u^n - \alpha_n \frac{\partial}{\partial u}(u^n) \right) \tag{22.18}$$

la méthode porte le nom de gradient à *descente maximale*. Une visualisation intuitive de cette technique peut être proposéeainsi : un randonneur dans le brouillard cherche à redescendre le plus vite possible de la montagne où il est perdu. Il observe les alentours et choisit de prendre la direction de la plus grande pente descendante. Si la pente est faible dans cette direction, il sera obligé de recommencer l'opération à peu de distance : le relief est en effet difficile à deviner dans

Copyright © 2010 Eyrolles.

la zone où il se trouve. Si la pente est forte et la montagne assez régulière, il peut par contre raisonnablement penser que le profil du terrain ne va pas se modifier immédiatement ; il peut parcourir plus de chemin dans la direction choisie.

Le cas idéal

On dit qu'une fonction $f : D \rightarrow \mathbb{R}$ est *α-convexe*, si, par définition, il existe un réel positif α tel que :

$$\forall u \in D, \forall v \in D, \forall \delta \in \{0,1\} : f((1-\delta)u + \delta v) \leq (1-\delta)f(u) + \delta f(v) + \frac{\alpha}{2}\delta(1-\delta)\|u-v\|^2$$

Supposons f différentiable et α-convexe. En prenant pour α_n la solution de l'équation 22.18 et en notant $\langle .,. \rangle$ le produit scalaire, on a :

$$\langle \frac{\partial}{\partial u}(u^n - \alpha_n \frac{\partial}{\partial u}(u^n)), \frac{\partial}{\partial u}(u^n)) \rangle = 0$$

soit :

$$\langle \frac{\partial}{\partial u}(u^{n+1}), \frac{\partial}{\partial u}(u^n) \rangle = 0$$

Ce qui signifie que chaque pas de descente maximale se fait dans une direction orthogonale au précédent. On peut alors démontrer que :

$$\lim_{n \rightarrow +\infty} (\|u^{n+1} - u^n\|) = 0$$

La suite u_n tend donc vers une limite. De plus, cette limite est la valeur cherchée u^*, qui minimise f.

NB : la démonstration demande au passage que la fonction f soit *lipschitzienne*, c'est-à-dire que

$$\forall u \in D, \forall v \in D, \exists c \in \mathbb{R} : \quad \|f(u) - f(v)\| \leq c\,\|u - v\|$$

Par conséquent, si les propriété de f sont assez fortes, la descente de gradient amène au point cherché. On peut aussi démontrer que la convergence est encore vraie pour α_n fixé à une valeur positive bornée par une constante dépendant de α et de c.

Le cas général : quelques problèmes

La propriété d'α-convexité est très forte. Intuitivement, elle signifie que le graphe de la fonction f a une courbure supérieure en tout point à la parabole $g(v) = 1/2\alpha\|v\|^2$. La propriété d'être lipschitzienne est moins exigeante : elle implique une régularité qui n'est pas rare en pratique. Quand on cherche à minimiser une fonction, on ignore en général si elle vérifie les deux propriétés précédentes. La plupart du temps, on peut au mieux la supposer dérivable. Que donne alors la méthode de la descente de gradient ?

La figure 22.1 donne une intuition de la méthode et de ses problèmes. Elle représente le graphe d'une fonction dérivable sur le domaine $D = [0,1]$. On y voit trois cas caractéristiques :

- Si u_n se trouve dans la partie droite du graphe, le gradient $\frac{\partial}{\partial u}(u^n)$ est positif et grand. Par conséquent, en prenant

$$u^{n+1} = u^n - \alpha \frac{\partial}{\partial u}(u^n)$$

avec α réel positif, on obtiendra par exemple u^{n+1} au point indiqué. À cet endroit, le gradient y est plus petit, quoique toujours positif. Le point suivant u^{n+2}, calculé avec la même valeur de α, sera donc plus proche de u^{n+1} que u^{n+1} l'était de u_n. On assiste de la sorte à la convergence de la suite vers la valeur minimale u^* cherchée.

Copyright © 2010 Eyrolles.

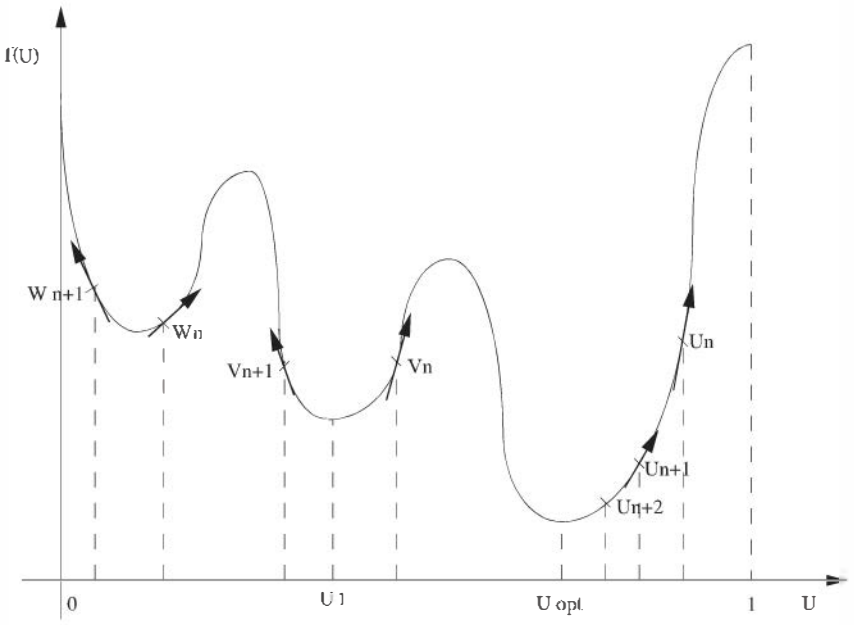


FIG. 22.1: *Une illustration de la méthode de descente de gradient.*

- Construisons maintenant une autre suite à partir du point v^n, avec une valeur supérieure pour α. Cette suite converge aussi, mais il y a deux différences avec le cas précédent :
  - La suite n'est plus strictement croissante ou décroissante, mais converge par valeurs alternativement supérieures et inférieures à la valeur finale.
  - La valeur atteinte n'est pas le minimum cherché, mais un *minimum local* u_{local}^{opt}. Cette erreur de convergence est le résultat de l'affaiblissement des propriétés de f. Dans la pratique, ce cas se rencontre très souvent.
- À partir du point w^n, en prenant α encore plus grand, par contre la suite va diverger.

En pratique

Rappelons que dans le cas général u est un vecteur. En général, son domaine D sera convexe, par exemple défini par un hyperrectangle :

$$D = \{l_1^{min} \leq x_1 \leq l_1^{max}, \ldots, l_d^{min} \leq x_d \leq l_d^{max}\}$$

De nombreuses méthodes ont été proposées pour pallier les faiblesses présentées sur l'exemple monodimensionnel de la figure 22.1. En ce qui concerne le réglage de la valeur de α, on peut par exemple utiliser la technique de la descente maximale, qui calcule au mieux cette valeur, ou en prendre une approximation (en général, faire décroître α avec n). On peut également faire intervenir la dérivée seconde de f. Il est plus difficile d'éviter les minima locaux. On peut évidemment appliquer plusieurs fois la méthode en faisant varier les paramètres de l'algorithme (réglage de α, point de départ) et choisir le minimum des minima locaux trouvés. On peut aussi introduire des perturbations aléatoires dans le calcul de la suite u_n, pour tenter de la faire sortir des zones où elle converge vers un minimum local. [Fle80] est une bonne référence pour la théorie et la pratique des méthodes d'optimisation (en particulier par descente de gradient).

Copyright © 2010 Eyrolles.

9. La rétropropagation du gradient de l'erreur

Dans cette annexe, nous montrons comment établir les formules de la section 3.2 qui dictent le changement de poids d'un réseau de neurones entraîné par la règle de la rétropropagation de l'erreur.

9.1 Notations

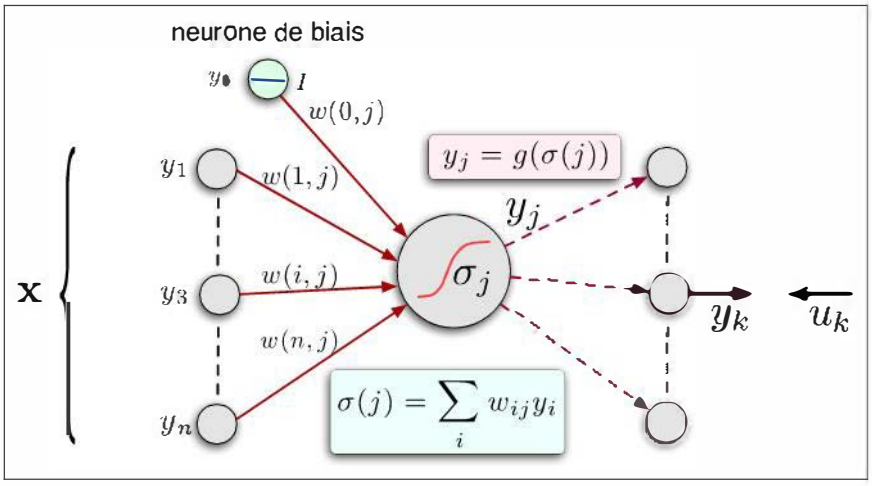


FIG. 22.2: *Dissection d'un réseau multicouche. Pour l'entrée* x, *le neurone* j *prend l'état* σ_j *et produit la sortie* y_j. *Sur la couche de sortie, l'erreur mesurée vaut* $E_{\mathbf{x}}$.

Nous notons σ_j l'état d'activation du neurone j à la présentation du vecteur d'entrée x. De la même façon, nous notons y_j la sortie du neurone j, $w(i,j)$ le poids de la connexion entre les neurones i et j. Sur la couche de sortie, pour le neurone k, on peut comparer la valeur u_k désirée et la valeur y_k produite.

Nous notons $source(j)$ l'ensemble des neurones connectés au neurone j et $dest(j)$ l'ensemble des neurones auxquels j se connecte.

9.2 Fonctionnement du système

Rappelons tout d'abord les équations de base régissant le fonctionnement du réseau. La règle de propagation dans le réseau est la suivante, pour un neurone j quelconque :

$$\sigma_j = \sum_{i \in source(j)} w(i,j) \cdot y_i \tag{22.19}$$

La fonction de sortie f permet de calculer y_j en fonction de σ_j :

$$y_j = f(\sigma_j) \tag{22.20}$$

Cette fonction f doit être continue, croissante, bornée et dérivable. On utilise souvent une *fonction sigmoïde* définie par :

$$f(u) = \frac{1}{1 + e^{-\lambda u}} \tag{22.21}$$

Copyright © 2010 Eyrolles.

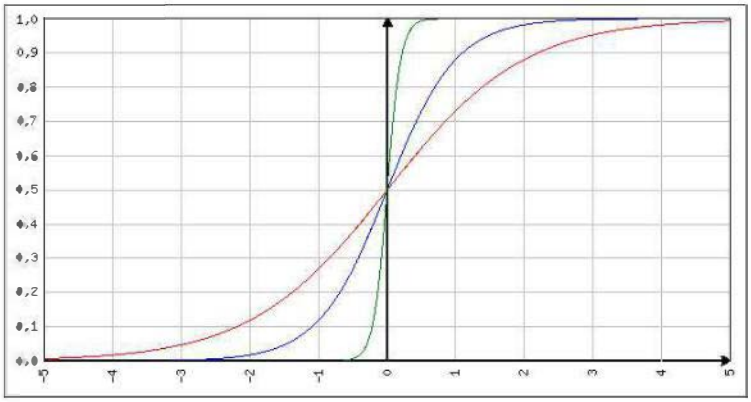


FIG. 22.3: *La fonction* sigmoïde *d'équation* $y_i = f(\sigma_i) = \frac{1}{1+e^{-\lambda\sigma_i}}$. *Cette fonction est paramétrée par sa pente à l'origine* λ. *Pour* λ *très grand, on retrouve la fonction seuil. Pour* λ *très petit, cette fonction est pratiquement linéaire dans une vaste région autour de l'origine. Les sigmoïdes de cette figure correspondent à* $\lambda = 1, 2, 10$ *pour* $\sigma_i \in [-5, 5]$.

avec λ réel positif.

Cette fonction répond à l'équation différentielle :

$$f' = \lambda f(1 - f)$$

En effet :

$$f(u) = \frac{1}{1 + e^{-\lambda u}}$$

$$f'(u) = \frac{-(-\lambda e^{-\lambda u})}{(1 + e^{-\lambda u})^2}$$

$$f'(u) = \lambda \frac{1}{1 + e^{-\lambda u}}(1 - \frac{1}{1 + e^{-\lambda u}}) = \lambda f(u)(1 - f(u)) \tag{22.22}$$

On peut remarquer que, quand λ est très grand, f se rapproche de la fonction signe ; quand λ est très petit, mais non nul, f est quasi linéaire autour de l'origine et de valeurs bornées entre 0 et 1.

On considèrera ici pour simplifier la fonction sigmoïde de paramètre λ égal à 1. Prendre λ quelconque introduirait simplement un coefficient de proportionnalité dans les calculs qui suivent.

9.3 Calcul du gradient

Nous cherchons à faire évoluer le réseau de façon à minimiser un coût $E_{\mathbf{x}}$ pour chaque vecteur d'entrée $\mathbf{x}$. On utilise ici le critère des moindres carrés :

$$E_{\mathbf{x}} = \frac{1}{2} \sum_{j \in sortie} (u_j - y_j)^2 \tag{22.23}$$

Nous cherchons le gradient de $E_{\mathbf{x}}$ par rapport aux poids du réseau, c'est-à-dire les valeurs

$$\frac{\partial E_{\mathbf{x}}}{\partial w(i,j)}$$

Copyright © 2010 Eyrolles.

Pour plus de clarté, nous pouvons omettre l'indice x dans les équations qui viennent. Nous définissons, pour chaque neurone j, la quantité :

$$\delta_j = \frac{\partial E}{\partial \sigma_j}$$

Le cas des neurones de sortie

Comme E dépend de $w(i,j)$ seulement par l'intermédiaire de σ_j, la formule des dérivations partielles chaînée permet d'écrire[3] :

$$\frac{\partial E}{\partial w(i,j)} = \frac{\partial E}{\partial \sigma_j} \cdot \frac{\partial \sigma_j}{\partial w(i,j)} = \delta_j \cdot y_i$$

Lorsque j est un neurone de sortie, avec f définie comme dans l'équation (22.21) (et λ égal à 1), on a, en remarquant que y_j ne dépend que de σ_j :

$$\delta_j = \frac{\partial E}{\partial \sigma_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial \sigma_j}$$

D'après les équations 22.20 et 22.22 :

$$\frac{\partial y_j}{\partial \sigma_j} = y_j \cdot (1 - y_j)$$

Puisque j est un neurone de sortie, d'après l'équation 22.23 :

$$\frac{\partial E}{\partial y_j} = y_j - u_j$$

Donc, pour un neurone de sortie :

$$\delta_j = y_j \cdot (1 - y_j) \cdot (y_j - u_j)$$

et finalement :

$$\frac{\partial E}{\partial w(i,j)} = y_j \cdot (1 - y_j) \cdot (y_j - u_j) \cdot y_i$$

Le cas des neurones cachés

Lorsque j est un neurone caché, on calcule δ_j par *rétropropagation*. La remarque fondamentale est que la contribution de chaque neurone formel j sur la sortie est propagée vers la sortie à travers les éléments de la couche $dest(j)$.

Comme E dépend de $w(i,j)$ seulement par l'intermédiaire de σ_j, la formule des dérivations partielles chaînée permet d'écrire :

$$\frac{\partial E}{\partial w(i,j)} = \frac{\partial E}{\partial \sigma_j} \cdot \frac{\partial \sigma_j}{\partial w(i,j)} = \delta_j \cdot y_i$$

Comme E dépend de tous les états σ_k, pour $k \in dest(j)$, et que chaque σ_k dépend (en ce qui concerne les variables qui nous intéressent) seulement de y_j (et donc seulement de σ_j), on peut aussi écrire :

$$\frac{\partial E}{\partial w(i,j)} = \sum_{k \in dest(j)} \frac{\partial E}{\partial \sigma_k} \cdot \frac{\partial \sigma_k}{\partial \sigma_j} \cdot \frac{\partial \sigma_j}{\partial w(i,j)}$$

[3] Toutes les dérivées sont notées comme partielles, par abus de langage.

Copyright © 2010 Eyrolles.

En sortant le terme non dépendant de k de la somme dans la seconde expression de $\frac{\partial E}{\partial w(i,j)}$, en égalant les deux expressions, et en utilisant la définition de δ_j et de δ_k, on obtient :

$$\delta_j = \sum_{k \in dest(j)} \delta_k \cdot \frac{\partial \sigma_k}{\partial \sigma_j}$$

Cela peut s'interpréter par le fait que la « responsabilité » δ_j de la cellule j dans l'erreur est propagée à ses successeurs δ_k avec un poids $w(j,k)$.

On peut maintenant terminer les calculs :

$$\begin{aligned}
\delta_j &= \sum_{k \in dest(j)} \delta_k \cdot \frac{\partial \sigma_k}{\partial \sigma_j} \\
&= \sum_{k \in dest(j)} \delta_k \cdot \frac{\partial \sigma_k}{\partial y_j} \cdot \frac{\partial y_j}{\partial \sigma_j} \\
&= \sum_{k \in dest(j)} \delta_k \cdot w(j,k) \cdot y_j \cdot (1 - y_j) \\
&= y_j \cdot (1 - y_j) \cdot \sum_{k \in dest(j)} \delta_k \cdot w(j,k)
\end{aligned}$$

et finalement :

$$\frac{\partial E}{\partial w(i,j)} = y_i \cdot y_j \cdot (1 - y_j) \cdot \sum_{k \in dest(j)} \delta_k \cdot w(j,k)$$

Conclusion

En résumant, on a donc les valeurs :

- Pour les neurones de sortie :

$$\frac{\partial E}{\partial w(i,j)} = y_i \cdot \delta_j = y_i \cdot y_j \cdot (1 - y_j) \cdot (y_j - u_j)$$

- Pour les neurones cachés :

$$\frac{\partial E}{\partial w(i,j)} = y_i \cdot \delta_j = y_i \cdot y_j \cdot (1 - y_j) \cdot \sum_{k \in dest(j)} \delta_k \cdot w(j,k)$$

Finalement, pour modifier $w(i,j)$, il faut lui additionner une quantité dans la direction opposée au gradient :

$$\Delta w(i,j) = -\alpha \frac{\partial E}{\partial w(i,j)} = -\alpha \cdot y_i \cdot \delta_j$$

où α, compris entre 0 et 1 est le pas de déplacement, aussi appelé le taux d'apprentissage.

10. L'analyse de l'induction de Vapnik

10.1 Cas où $|\mathcal{H}| = \infty$ et $\mathcal{F} \subseteq \mathcal{H}$

Dans le cas où le nombre d'hypothèses est fini, on comprend que l'on puisse borner la probabilité que l'on trouve parmi elles une hypothèse apparemment correcte sur l'échantillon d'apprentissage et pourtant médiocre en général. Lorsque la classe d'hypothèses contient un nombre

Copyright © 2010 Eyrolles.

infini d'éléments, il semble que l'on ne puisse pas dire grand-chose. Pourtant, intuitivement, on sent qu'il doit pouvoir exister des classes d'hypothèses de cardinal infini mais d'expressivité limitée, pour lesquelles la performance sur l'échantillon d'apprentissage peut servir d'indicateur de la performance réelle. Nous allons voir dans cette section une caractérisation possible de cette diversité qui peut servir à établir le lien entre risque empirique et risque réel.

L'idée générale est la même que pour le cas où $\mathcal{H}$ est de cardinalité finie (voir section 4.3 du chapitre 2). Il s'agit d'essayer de borner la probabilité d'avoir une hypothèse apparemment bonne, c'est-à-dire ici de risque empirique nul (hypothèse consistante) sur l'échantillon d'apprentissage, qui soit en fait de risque réel $> \varepsilon$:

$$P_{\mathcal{D}_Z}\{\mathcal{S} : \exists\ h \in \mathcal{H} : R_{\text{Emp}}(h) = 0 \quad \& \quad R_{\text{Réel}}(h) > \varepsilon\} \tag{22.24}$$

Une idée essentielle consiste à chercher une *mesure effective* de la variabilité de $\mathcal{H}$ en la mettant à l'épreuve sur un ensemble de points test tirés aléatoirement. Le risque réel sera alors évalué en utilisant un autre échantillon que l'échantillon d'apprentissage, qui sera lui aussi issu d'un tirage i.i.d. (formes tirées aléatoirement dans $\mathcal{X}$ suivant une même distribution). Nous le nommerons naturellement *échantillon de test*, noté $\mathcal{T}$. L'avantage est que l'on pourra alors *compter* les différents étiquetages de ces points que permet l'espace des fonctions hypothèses $\mathcal{H}$, et à partir de là caractériser la variabilité effective de $\mathcal{H}$.

On voudrait donc remplacer l'étude de l'expression (22.24) par celle de :

$$P_{\mathcal{D}_Z}\left\{\mathcal{S} : \exists\ h \in \mathcal{H} : R^{\mathcal{S}}_{\text{Emp}}(h) = 0 \quad \& \quad R^{\mathcal{T}}_{\text{Emp}}(h) = 0 > \varepsilon\right\} \tag{22.25}$$

où nous notons $R^{\mathcal{S}}_{\text{Emp}}(h)$ (resp. $R^{\mathcal{T}}_{\text{Emp}}(h)$) le risque empirique de l'hypothèse h mesuré sur l'échantillon $\mathcal{S}$ (resp. $\mathcal{T}$).

Une application des inégalités de Chernoff[4] avec $m > 2/\varepsilon$, permet de borner l'expression (22.24) par une probabilité relative à l'expression (22.25). Plus précisément, si on note $\mathcal{ST}$ l'échantillon constitué de la concaténation de $\mathcal{S}$ et de $\mathcal{T}$, et $R^{\mathcal{ST}}_{\text{Emp}}(h)$ le risque empirique de h mesuré sur $\mathcal{ST}$, on peut obtenir la borne :

$$\begin{aligned} &P_{\mathcal{D}_\mathcal{X}}\{\mathcal{S} : \exists\ h \in \mathcal{H} : R^{\mathcal{S}}_{\text{Emp}}(h) = 0 \quad \text{et} \quad R_{\text{Réel}}(h) > \varepsilon\} \\ &\quad \leq \quad 2P_{\mathcal{D}_\mathcal{X}}\{\mathcal{ST} : \exists h \in \mathcal{H} : R^{\mathcal{S}}_{\text{Emp}}(h) = 0 \quad \text{et} \quad R^{\mathcal{ST}}_{\text{Emp}}(h) > \frac{\varepsilon}{2}\} \end{aligned} \tag{22.26}$$

Répétons que cette nouvelle borne représente un progrès considérable dans la solution du problème initial. En effet, nous sommes maintenant ramenés à un comptage sur les étiquettes possibles de $2m$ points par les fonctions de $\mathcal{H}$, au lieu d'avoir à considérer l'ensemble infini des fonctions de $\mathcal{H}$.

Soit un échantillon i.i.d. suivant $\mathcal{D}_\mathcal{X}$ de $2m$ points, dont $l \geq \varepsilon m/2$ sont mal étiquetés par l'hypothèse h. Le nombre de cas possibles pour lesquels aucune de ces erreurs n'intervient dans les m premiers points est : C^l_m / C^l_{2m}. Or $C^l_m / C^l_{2m} \leq 1/2^l$, puisque :

$$\frac{C^l_m}{C^l_{2m}} = \prod_{i=0}^{i=l-1} \frac{m-i}{2m-i} \leq \prod_{i=0}^{i=l-1} \left(\frac{1}{2}\right) = \frac{1}{2^l}$$

La probabilité que cela arrive pour n'importe quel échantillon de taille $2m$ et pour n'importe quelle hypothèse est bornée par le nombre d'étiquetages différents de $2m$ points par des fonctions

[4] Pour les détails de la démonstration assez longue dont les développements dépassent le cadre de notre ouvrage, il est suggéré de se reporter à [AB92] pp.90-94 ou à [KV94] pp.59-61, ou encore à [AB96] pp.42-50. [DGL96] offre aussi au chapitre 12 une exposition très complète de la question.

Copyright © 2010 Eyrolles.

de $\mathcal{H}$. En d'autres termes, on évalue maintenant la richesse de l'espace d'hypothèses par le nombre de *dichotomies* différentes qu'il peut induire au maximum sur un échantillon aléatoire de $2m$ points. On appelle ce nombre la *fonction de croissance* de $\mathcal{H}$: $G_{\mathcal{H}}(2m)$.

$$P_{\mathcal{D}_{\mathcal{X}}}\{\mathcal{S} : \exists\ h \in \mathcal{H} : R^{\mathcal{S}}_{\text{Emp}}(h) = 0 \quad \text{et} \quad R_{\text{Réel}}(h) > \varepsilon\} \leq 2\ G_{\mathcal{H}}(2m)2^{-\varepsilon m/2} \tag{22.27}$$

Cette équation est très importante. Elle nous montre que la probabilité qu'il existe une mauvaise hypothèse (pour laquelle le risque empirique nul est trompeur) est bornée par un produit de deux termes, dont l'un est une exponentielle décroissante en m.

Il nous faut donc étudier le second terme : la fonction de croissance. En effet, l'inégalité ci-dessus n'est **utile que si la fonction de croissance croît à un rythme sub-exponentiel** pour laquelle la borne tend vers zéro rapidement.

10.2 Fonction de croissance et dimension de Vapnik-Chervonenkis

Définition 22.1 (Fonction de croissance)

La **fonction de croissance** $G_{\mathcal{H}}(m)$ *définit le nombre maximal de dichotomies qui peuvent être induites par l'espace des fonctions indicatrices* $\mathcal{H}$ *sur un échantillon de taille* m *sur* $\mathcal{X}$.

Il est important de noter que cette définition invoque l'échantillon le plus grand pour lequel il est possible de réaliser toutes les dichotomies (et non pas n'importe quel échantillon tiré aléatoirement).

Cette fonction est indépendante de toute distribution de probabilités sur $\mathcal{X}$ et ne dépend que de l'espace $\mathcal{H}$. Pour m points, le nombre maximal possible de dichotomies, c'est-à-dire d'étiquetages possibles, est de 2^m. Nous avons donc : $G_{\mathcal{H}}(m) \leq 2^m$.

On dit d'un ensemble de points $\{x_1, \ldots, x_m\}$ pour lequel $\mathcal{H}$ permet d'induire les 2^m dichotomies possibles qu'il est *pulvérisé* (*shattered set*). Dans ce cas, $G_{\mathcal{H}}(m) = 2^m$, c'est-à-dire que $\mathcal{H}$ permet de réaliser n'importe quelle fonction indicatrice de ces m points.

EXEMPLE **Séparatrices linéaires dans le plan**

La figure 22.4 illustre un espace d'entrée $\mathcal{X}$ de dimension 2 avec ici 4 points : x_1, x_2, x_3, x_4. Les bornes des fonctions h_1 et h_2 sont indiquées par des ellipses. D'après cette figure, les fonctions h_1 et h_2 induisent respectivement les dichotomies :

$$\mathcal{D}_1 = \{\mathcal{S}_- = \{x_3\}, \mathcal{S}_+ = \{x_1, x_2, x_4\}\} \quad \text{et} \quad \mathcal{D}_2 = \{\mathcal{S}_- = \{x_1, x_2\}, \mathcal{S}_+ = \{x_3, x_4\}\}$$

L'ensemble des dichotomies possibles de l'ensemble $\mathcal{S}$ de ces 4 points est : $|\Delta_{\mathcal{S}}| = 2^4 = 16$.

Pour la classe $\mathcal{H}$ des concepts définis par les séparatrices linéaires, il est aisé de voir que tout ensemble de trois points non colinéaires peut être pulvérisé. La figure 22.5(a) montre l'une des huit dichotomies possibles de trois points du plan. Le lecteur pourra facilement vérifier que les sept autres dichotomies sont également réalisables par une hypothèse de $\mathcal{H}$. Nous avons donc : $G_{\mathcal{H}}(3) = 2^3$.

Nous montrons qu'il est impossible de pulvériser un ensemble de quatre points quelconques par des séparatrices linéaires en considérant les deux cas génériques (l'un où tous les points sont sur l'enveloppe convexe, l'autre où seuls trois points définissent l'enveloppe convexe) (figure 22.5(b) et (c)) pour lesquels il n'existe pas de séparatrice linéaire permettant d'induire les dichotomies correspondantes.

Nous avons donc : $G_{\mathcal{H}}(4) \leq 2^4$ □

Copyright © 2010 Eyrolles.

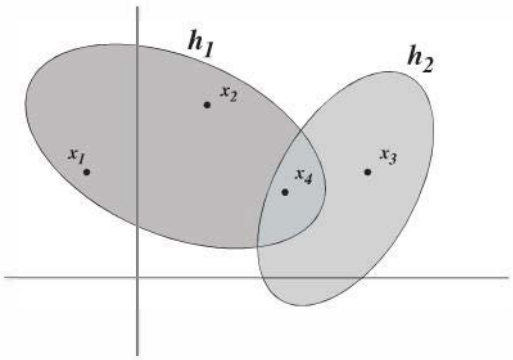


FIG. 22.4: *Un espace d'entrée $\mathcal{X}$ de dimension 2 avec les dichotomies induites par deux fonctions h_1 et h_2.*

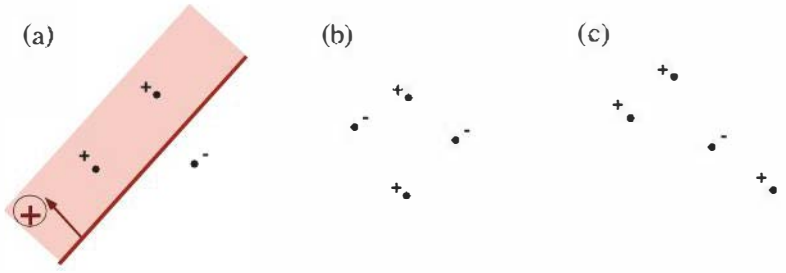


FIG. 22.5: *(a) Une dichotomie et une réalisation par une séparatrice linéaire. (b) et (c) Dichotomies irréalisables par des séparatrices linéaires.*

EXEMPLE **Rectangles parallèles aux axes dans le plan**

Il est également possible de réaliser toutes les dichotomies possibles de trois points du plan par des rectangles parallèles aux axes. Seul le cas dégénéré où le point négatif est exactement au milieu des points positifs rend impossible cette dichotomie (voir la figure 22.6 (a)) pour un exemple). Par ailleurs, il existe un échantillon de quatre points pour lesquels il est possible de réaliser tous les étiquetages possibles à l'aide de rectangles parallèles aux axes (voir la figure 22.6 (a), nous laissons au lecteur le soin de le vérifier pour les quinze autres étiquetages). Cependant, il n'est pas possible de former toutes les dichotomies de quatre points quelconques, comme l'illustre la figure 22.6 (b). L'existence d'un seul échantillon de quatre points pouvant être pulvérisé suffit à borner la fonction de croissance par le bas $G_{\mathcal{H}}(4) = 2^4$. Si nous considérons maintenant n'importe quel échantillon de cinq points, l'un de ces points est nécessairement ni en position extrême gauche, ni en position extrême droite, ni en position extrême haute, ni en position extrême basse (voir figure 22.6 (d)). Si nous étiquetons ce point non extrémal par un $-$, et les quatre autres points par $+$, alors il est impossible de trouver un rectangle permettant de couvrir ces 4 points sans couvrir le cinquième. D'où $G_{\mathcal{H}}(5) = 2^4$. □

Définition 22.1

On appelle **dimension de Vapnik-Chervonenkis** *(ou dimension entière) d'un espace de fonctions binaires $\mathcal{H}$ que l'on note $d_{VC}(\mathcal{H})$ le cardinal du plus grand ensemble de points de l'espace d'entrée $\mathcal{X}$ qu'il est possible de pulvériser. Si il n'y a pas de maximum, par convention $d_{VC}(\mathcal{H}) = \infty$.*

Copyright © 2010 Eyrolles.

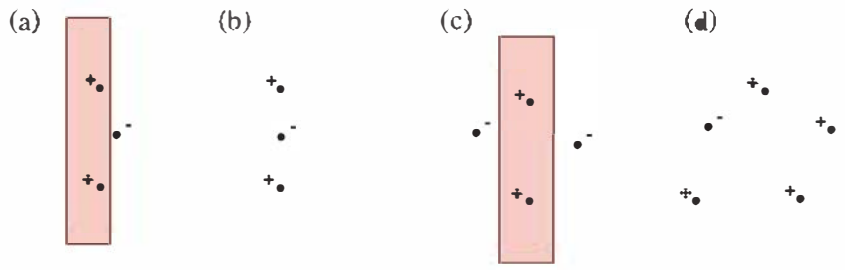


FIG. 22.6: *(a) Exemple de trois points dont la dichotomie est réalisable par au moins un rectangle. (b) Exemple de trois points ne pouvant être distingués grâce à un rectangle. Mais il s'agit du seul cas, correspondant à un cas dégénéré du cas général à trois points (a). (c) Une dichotomie de quatre points et une réalisation par un rectangle parallèle aux axes. (d) Une dichotomie irréalisable.*

En d'autres termes, la dimension de Vapnik-Chervonenkis d'un ensemble de fonctions de discriminations est le nombre maximal d'exemples d'apprentissage dont il est possible d'apprendre n'importe quel étiquetage. Cela signifie qu'il existe un ensemble de points au moins dont le cardinal est égal à $d_{VC}(\mathcal{H})$ et sur lequel la « souplesse » de $\mathcal{H}$ est totale.

La dimension de Vapnik-Chervonenkis est donc par définition liée à la fonction de croissance par la relation :

$$d_{VC}(\mathcal{H}) = \max\{m \in \mathbb{N} \ : \ G_{\mathcal{H}}(m) = 2^m\} \tag{22.28}$$

EXEMPLE **Séparatrices linéaires dans le plan**

D'après l'exemple 10.2, la dimension de Vapnik-Chervonenkis pour les fonctions séparatrices linéaires du plan est égale à 3.

Le lecteur pourra démontrer que la dimension de Vapnik-Chervonenkis des séparatrices linéaires sur $\mathbb{R}^d$ est égale à $d + 1$.

Il nous reste à voir en quoi la dimension de Vapnik-Chervonenkis est plus précisément reliée à la fonction de croissance pour nous permettre de trouver les conditions de croissance subexponentielle de cette dernière lesquelles, rappellons-le, sont nécessaires pour garantir l'utilité du principe inductif *ERM*.

10.3 Le lemme de Sauer : un lemme sauveur

Nous avons vu que la fonction de croissance $G_{\mathcal{H}}(m)$ est bornée par 2^m. Mais est-ce que toute valeur est possible en dessous de cette limite (voir figure 22.7) ? Le lemme démontré indépendamment par Sauer (1972), Shelah (1972) et Vapnik et Chervonenkis (1971) montre que ce n'est pas le cas si la dimension de Vapnik-Chervonenkis est finie.

Copyright © 2010 Eyrolles.

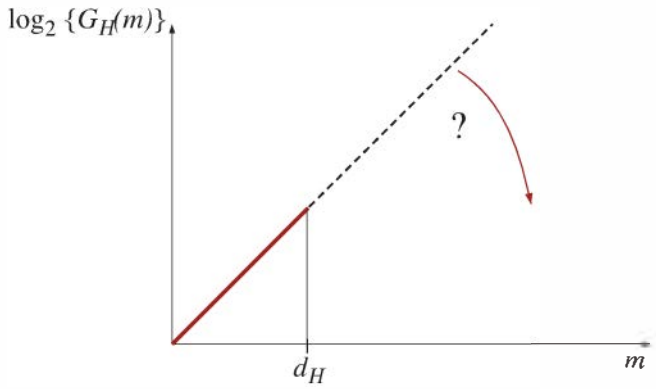


FIG. 22.7: *Le comportement de la fonction de croissance. Nous avons tracé le logarithme en base 2 de la fonction de croissance en fonction du nombre de points dans l'espace d'entrée $\mathcal{X}$. Jusqu'à $d_{\mathcal{H}}$, la dimension de Vapnik-Chervonenkis de $\mathcal{H}$, $G_{\mathcal{H}}(m) = 2^m$, d'où le tracé linéaire. Mais pour des valeurs de $m \geq d_{\mathcal{H}}$, quelle est l'enveloppe maximale des dichotomies possibles de m points dans $\mathcal{X}$ par des fonctions de $\mathcal{H}$?*

Théorème 22.1 (Lemme de Sauer)

Soient $\mathcal{H}$ un espace de fonctions hypothèses indicatrices sur l'espace d'entrée $\mathcal{X}$, $d_{\mathcal{H}}$ sa dimension de Vapnik-Chervonenkis, $d_{\mathcal{H}} = d_{VC}(\mathcal{H})$, et m points quelconques dans $\mathcal{X}$. Alors[5] :

$$G_{\mathcal{H}}(m) \quad \leq \quad \sum_{i=0}^{d_{\mathcal{H}}} C_m^i \quad \leq \quad \left(\frac{em}{d_{\mathcal{H}}}\right)^{d_{\mathcal{H}}} \quad \in \quad \mathcal{O}(m^{d_{\mathcal{H}}}) \tag{22.29}$$

Démonstration. Nous allons d'abord borner le nombre de dichotomies réalisables sur un échantillon $\mathcal{S}$ de m points quand $d_{\mathcal{H}} = d < m$ par une somme de combinaisons, puis nous montrerons que cette somme est bornée par une fonction polynomiale de m. C'est la première partie de la démonstration qui est la plus importante, et la plus intéressante.

Supposons par hypothèse que la fonction de croissance $G_{\mathcal{H}}(m)$ soit bornée par une fonction $\phi_{d_{\mathcal{H}}}(m)$. Soit $\pi_{\mathcal{H}}(\mathcal{S})$ l'ensemble des dichotomies de $\mathcal{S}$ réalisables à l'aide des fonctions de $\mathcal{H}$, et soit $\boldsymbol{x}$ un point de $\mathcal{S}$. Nous voudrions calculer $|\pi_{d_{\mathcal{H}}}(\mathcal{S})|$ à partir de $|\pi_{d_{\mathcal{H}}}(\mathcal{S} - \boldsymbol{x})|$, et ainsi obtenir une expression récursive de $|\pi_{d_{\mathcal{H}}}(\mathcal{S})|$.

La différence entre $|\pi_{d_{\mathcal{H}}}(\mathcal{S})|$ et $|\pi_{d_{\mathcal{H}}}(\mathcal{S} - \boldsymbol{x})|$ provient des ensembles de dichotomies de $\mathcal{S}$ qui ne diffèrent que par l'étiquette de $\boldsymbol{x}$, mais qui sont donc non distinguées dans $\pi_{\mathcal{H}}(\mathcal{S} - \boldsymbol{x})$. Elles correspondent à un ensemble $D_{\mathcal{S}-\boldsymbol{x}}$ de dichotomies dans $\pi_{\mathcal{H}}(\mathcal{S} - \boldsymbol{x})$. Nous avons :

$$|\pi_{d_{\mathcal{H}}}(\mathcal{S})| \quad = \quad |\pi_{d_{\mathcal{H}}}(\mathcal{S} - \boldsymbol{x})| \; + \; |D_{\mathcal{S}-\boldsymbol{x}}|$$

Il faut donc compter le nombre de ces paires de dichotomies de $D_{\mathcal{S}-\boldsymbol{x}}$. Quelle est la dimension de Vapnik-Chervonenkis de cet ensemble ?

Il est facile de voir qu'elle est nécessairement strictement inférieure à $d_{\mathcal{H}}$. En effet, si elle était égale à $d_{\mathcal{H}}$ (si toutes les dichotomies de $\pi_{\mathcal{H}}(\mathcal{S}-\boldsymbol{x})$ étaient réalisables), alors toutes les dichotomies de $\mathcal{S}$ le seraient aussi, puisque pour chaque dichotomie de $\pi_{\mathcal{H}}(\mathcal{S} - \boldsymbol{x})$ on peut étiqueter $\boldsymbol{x}$ de deux manières différentes. Et donc on aurait $d_{\mathcal{H}} = d + 1$, ce qui contredirait notre hypothèse de départ.

Copyright © 2010 Eyrolles.

Nous avons donc : $|\phi_{d_\mathcal{H}}(\mathcal{S})| = |\phi_{d_\mathcal{H}}(\mathcal{S} - x)| + |D_{\mathcal{S}-x}|$, soit encore :

$$\phi_d(m) = \phi_d(m-1) + \phi_{d-1}(m-1)$$

avec $\phi_d(0) = \phi_0(m) = 1$ (ce qui est évident).

Il est alors facile de montrer que : $\phi_d(m) = \sum_{i=0}^{d_\mathcal{H}} \mathrm{C}_m^i$.

Le deuxième temps de la démonstration est simplement technique.

Pour $0 \leq i \leq d$ et $m \geq d$: $(m/d)^d (d/m)^i \geq 1$, d'où :

$$\sum_{i=0}^{d_\mathcal{H}} \mathrm{C}_m^i \leq (m/d)^d \sum_{i=0}^{d_\mathcal{H}} \mathrm{C}_m^i (d/m)^i \leq (m/d)^d (1+d/m)^m < \left(\frac{em}{d_\mathcal{H}}\right)^{d_\mathcal{H}}$$

en utilisant des développements en série limitée classiques. □

Le comportement de la fonction de croissance $G_\mathcal{H}(m)$ est donc contrôlé par la dimension de Vapnik-Chervonenkis $d_\mathcal{H}$. De manière *a priori* surprenante, pour $m > d_\mathcal{H}$, il n'y a donc pas de fonction de croissance possible au-dessus de la borne polynomiale $(em/d_\mathcal{H})^{d_\mathcal{H}}$. Par exemple, il n'est pas possible d'avoir une fonction de croissance en $\sqrt{m}$. **La dimension de Vapnik-Chervonenkis correspond à une borne fondamentale de la capacité d'expression d'un ensemble de fonctions hypothèses.**

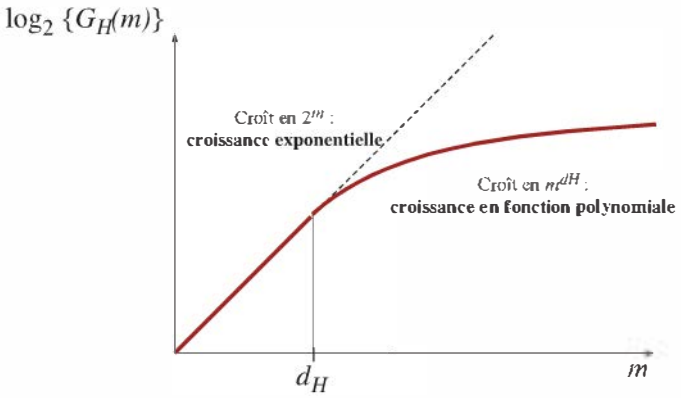


FIG. 22.8: *Le comportement de la fonction de croissance d'après le lemme de Sauer. On a tracé ici le logarithme en base 2 de la fonction de croissance.*

La figure 22.8 illustre le comportement de la fonction de croissance d'après le lemme de Sauer. Il est intéressant d'en prendre le logarithme Népérien :

$$\ln\left\{G_\mathcal{H}(m)\right\} = \ln\left(\frac{em}{d_\mathcal{H}}\right)^{d_\mathcal{H}} = d_\mathcal{H}\left(1 + \ln\frac{m}{d_\mathcal{H}}\right) \tag{22.30}$$

En reprenant l'équation 22.27, nous arrivons à l'équation fondamentale :

$$\begin{aligned} P_{\mathcal{D}_x}\{\mathcal{S} : \exists\ h \in \mathcal{H} : R^\mathcal{S}_{\mathrm{Emp}}(h) = 0 \ \ \& \ \ R_{\mathrm{Réel}}(h) > \varepsilon\} &\leq 2\ G_\mathcal{H}(2m) 2^{-\varepsilon m/2} \\ &\leq 2\left(\frac{2em}{d_\mathcal{H}}\right)^{d_\mathcal{H}} 2^{-\varepsilon m/2} \end{aligned} \tag{22.31}$$

Copyright © 2010 Eyrolles.

Cette équation montre que si la dimension de Vapnik-Chervonenkis $d_{\mathcal{H}}$ est finie, alors la convergence du risque empirique vers le risque réel est exponentiellement rapide, d'après le terme en $2^{-\varepsilon m/2}$, et ceci uniformément pour n'importe quelle fonction $h \in \mathcal{H}$.

À partir de cette équation, on peut aussi calculer la taille minimale d'un échantillon d'apprentissage pour que la probabilité dénotée par l'équation 22.31 soit inférieure à δ :

$$m = \mathcal{O}\left(\frac{1}{\varepsilon}\ln\frac{1}{\delta} + \frac{d_{\mathcal{H}}}{\varepsilon}\ln\frac{1}{\varepsilon}\right) \tag{22.32}$$

— EXEMPLE **Application numérique**

Soit $d_{\mathcal{H}} = 3$, la dimension de Vapnik-Chervonenkis de l'espace d'hypothèses (il s'agit par exemple de l'espace des séparatrices linéaires du plan, telles qu'un perceptron pourrait les réaliser). On veut une erreur d'approximation limitée à $\varepsilon = 1$ % pour un seuil de confiance de 95 % ($\delta = 5$ %). Alors, il faut avoir un échantillon d'apprentissage de taille $m \geq 2425$.

Voyons maintenant comment traduire l'équation 22.31 en une borne d'erreur pour toute hypothèse h consistante.

Théorème 22.2 (Théorème *PAC* pour les espaces infinis de fonctions indicatrices)

Soit $\mathcal{H}$ un espace de fonctions indicatrices de dimension de Vapnik-Chervonenkis $d_{\mathcal{H}}$. Pour toute distribution de probabilités $\mathcal{D}$ sur $\mathcal{X} \times \{-1, 1\}$, avec une probabilité $1 - \delta$ sur les échantillons $\mathcal{S}$ de m exemples tirés aléatoirement sur $\mathcal{Z}$, toute hypothèse $h \in \mathcal{H}$ cohérente avec l'échantillon $\mathcal{S}$ est d'erreur réelle bornée par :

$$R_{\text{Réel}}(h) \leq \frac{2}{m}\left(d_{\mathcal{H}}\ln\frac{2em}{d_{\mathcal{H}}} + \ln\frac{2}{\delta}\right) \tag{22.33}$$

pourvu que $d_{\mathcal{H}} \leq m$ et $m > 2/\varepsilon$.

Ce théorème montre que la taille de l'échantillon d'apprentissage requise pour assurer (en probabilité) une bonne généralisation est une **fonction linéaire de la dimension de Vapnik-Chervonenkis de l'espace d'hypothèses** quand on choisit une hypothèse cohérente, et ceci face à toute distribution des exemples.

10.4 L'analyse de Vapnik et Chervonenkis pour des fonctions quelconques

L'analyse précédente est limitée sur un certain nombre de points. Elle ne s'applique en effet :

- qu'à des fonctions indicatrices, donc à des applications d'apprentissage de fonctions de discrimination (apprentissage de concepts) ;
- qu'à des fonctions de perte comptant le nombre d'erreurs de classification (fonction de perte pour la discrimination) ;
- qu'au cas où l'espace d'hypothèses $\mathcal{H}$ contient l'espace de fonctions cible $\mathcal{F}$, ce qui permet d'assurer l'existence d'au moins une hypothèse cohérente avec l'échantillon d'apprentissage.

Les travaux de Vapnik et Chervonenkis, menés avec opiniâtreté sur une longue période (1968-1998 environ), ont permis de lever ces limitations et de fournir une approche générale du problème de l'induction pour des espaces infinis de fonctions hypothèses quelconques (ou presque) utilisant des fonctions de pertes quelconques (ou presque), y compris dans le cas où les espaces $\mathcal{H}$ et $\mathcal{F}$ ne coïncident pas. Cette analyse sert actuellement de cadre de référence pour toute la théorie de l'apprentissage.

Copyright © 2010 Eyrolles.

Nous en donnons un rapide aperçu dans ce qui suit, reportant le lecteur intéressé aux nombreuses publications concernant ce sujet, et en particulier à [Vap95] pour un petit livre d'un abord facile résumant l'approche, et à [Vap98] pour une description détaillée. Le lecteur motivé pourra également se reporter avec profit à [AB96] qui donne de nombreuses preuves. [DGL96] est également une référence intéressante, qui ne traite toutefois que des problèmes de classification binaire.

10.4.1 Le théorème fondamental de la théorie de l'apprentissage selon Vapnik

Rappelons que selon Vapnik, le problème central de l'induction est de chercher quelle relation existe entre une hypothèse sélectionnée selon le principe inductif *ERM*, qui minimise le risque empirique, et l'hypothèse optimisant le risque réel. Il s'agit en particulier d'étudier les conditions de pertinence du principe *ERM* correspondant aux équations (2.55) et (2.51). Vapnik et Chervonenkis [Vap82, VC91] ont montré le théorème suivant.

Théorème 22.3 (Théorème fondamental de pertinence de l'*ERM*, Vapnik)

*Pour des fonctions de perte bornées, le principe de minimisation du risque empirique (*ERM*) est pertinent si et seulement si le risque empirique* converge uniformément *vers le risque réel au sens suivant :*

$$\lim_{m\to\infty} P\left[\sup_{\hat{h}_S\in\mathcal{H}} |R_{Réel}(\hat{h}_S) - R_{Emp}(\hat{h}_S)| > \varepsilon\right] = 0, \quad \forall\,\varepsilon > 0 \tag{22.34}$$

À nouveau, insistons sur ce point : ce théorème[6] signifie quelque chose de très profond et de très important : la pertinence du principe de minimisation du risque empirique est déterminée par la *pire* des fonctions hypothèse de $\mathcal{H}$, c'est-à-dire celle dont l'écart entre le risque empirique mesuré et le risque réel est le plus grand. C'est là l'essence des convergences uniformes. Nous nous trouvons ici dans une théorie qui spécifie des garanties de généralisation face à n'importe quelle distribution des exemples tirés dans l'échantillon d'apprentissage (d'où la convergence en probabilité et le seuil de confiance δ) et face à la pire des fonctions hypothèse possibles, au sens où, choisie selon le principe inductif *ERM*, elle se révèlerait de fait la moins bonne.

Un théorème de convergence uniforme nécessite une propriété sur l'ensemble $\mathcal{H}$ des fonctions hypothèse. Pour assurer la pertinence du principe *ERM* par l'équation (22.34), il faut donc se donner une *mesure de diversité* sur $\mathcal{H}$. L'une de ces mesures de diversité ou de richesse de l'espace d'hypothèses est évidemment la *dimension de Vapnik-Chervonenkis*. C'est elle qui a permis à Vapnik et Chervonenkis d'énoncer une loi des grands nombres pour les espaces fonctionnels.

Théorème 22.4 (Conditions nécessaires et suffisantes pour la pertinence de l'*ERM*)

L'équation suivante fournit une condition nécessaire et suffisante pour la pertinence du principe ERM, *ainsi qu'une garantie de convergence rapide, indépendamment de la distribution de probabilités sur les échantillons $\mathcal{S}$.*

$$\lim_{m\to\infty} \frac{\ln G_{\mathcal{H}}(m)}{m} = 0 \tag{22.35}$$

Nous retrouvons ici la nécessité, soulignée dans l'analyse *PAC*, que la fonction de croissance croisse moins vite qu'une fonction exponentielle de la taille m de l'échantillon.

[6] Pour une preuve partielle, se reporter par exemple à [Hay99] p.93.

Copyright © 2010 Eyrolles.

10.4.2 Bornes utiles sur la capacité de généralisation

En reprenant la figure 22.9, les questions centrales dans l'analyse du principe inductif *ERM* et de sa consistance sont :

1. Quel est le risque réel associé à l'hypothèse $\hat{h}_S$ qui minimise le risque empirique, et quelle est la relation liant $R_{\text{Réel}}(\hat{h}_S)$ et $R_{\text{Emp}}(\hat{h}_S)$?
2. Quelle est la proximité entre le risque réel $R_{\text{Réel}}(\hat{h}_S)$ associé à l'hypothèse $\hat{h}_S$ sélectionnée par le principe inductif *ERM* et le risque réel optimal $R_{\text{Réel}}(h^\star)$?

Les réponses à ces questions peuvent être obtenues par le calcul de bornes de taux de convergence découlant de l'analyse théorique conduite par Vapnik et Chervonenkis. Ces bornes sont fonctions de la taille m de l'échantillon d'apprentissage, des propriétés de l'espace de fonctions hypothèse $\mathcal{H}$ et de celles des fonctions de perte $l(u_i, h(x_i))$. Nous nous limitons ici à la présentation de résultats concernant les fonctions de perte positives bornées (correspondant aux problèmes de classification). Des bornes pour d'autres fonctions de perte sont discutées notamment dans [Vap95] et [Vap98].

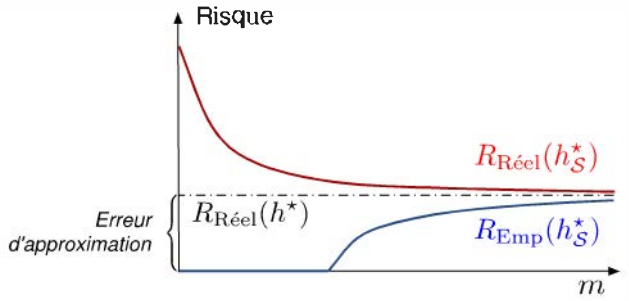


FIG. 22.9: *Reprise de la figure 2.9.*

Les fonctions de perte positives bornées. On suppose que la fonction de perte $l(u_i, h(x_i))$ est bornée par B quelle que soit la fonction hypothèse h, les exemples x_i et les réponses désirées u_i.

Alors les réponses aux questions ci-dessus sont fournies par les bornes suivantes :

1. L'inégalité suivante est vérifiée avec une probabilité $\geq 1 - \delta$ simultanément pour toutes les fonctions de $\mathcal{H}$, et en particulier la fonction $\hat{h}_S$ minimisant le risque empirique (condition de convergence uniforme) :

$$R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \frac{B\epsilon}{2}\left(1 + \sqrt{1 + \frac{4R_{\text{Emp}}(h)}{B\epsilon}}\right) \tag{22.36}$$

où :

$$\epsilon = 4\,\frac{G_{\mathcal{H}}(2m) - \ln\delta/4}{m} = 4\,\frac{d_{\mathcal{H}}[\ln\frac{2m}{d_{\mathcal{H}}} + 1] - \ln\delta/4}{m}$$

2. L'inégalité suivante est valide avec une probabilité $\geq 1 - 2\delta$ pour la fonction $\hat{h}_S$ qui minimise le risque empirique :

$$R_{\text{Réel}}(\hat{h}_S) - R_{\text{Réel}}(h^\star) \leq B\sqrt{\frac{-\ln\delta}{2m}} + \frac{B\epsilon}{2}\left(1 + \sqrt{1 + \frac{4}{\epsilon}}\right) \tag{22.37}$$

Copyright © 2010 Eyrolles.

Il faut noter que la fonction de perte classique en classification prenant ses valeurs dans $\{0,1\}$, la borne B est égale à 1 dans ce cas. Par ailleurs, dans le cas de l'analyse PAC où le risque empirique optimal est nul, on obtient pour l'équation 22.36, l'expression suivante :

$$R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + 2\,\frac{d_{\mathcal{H}}[\ln\frac{2m}{d_{\mathcal{H}}}+1] - \ln\delta/4}{m} \tag{22.38}$$

Il ressort de ces inégalités le **message important** que si la dimension de Vapnik-Chervonenkis de l'espace d'hypothèses est finie, alors l'erreur d'estimation entre $R_{\text{Réel}}(\hat{h}_S)$ et $R_{\text{Réel}}(h^*)$ converge vers zéro au rythme de $O(\sqrt{\frac{d_{\mathcal{H}}\ln m}{m}})$ dans le cas général et de $O(\frac{d_{\mathcal{H}}\ln m}{m})$ dans le cas où $\mathcal{H}$ comprend la fonction cible f^*, et ceci *pour toute distribution* sur $\mathcal{X}\times\mathcal{U}$.

La beauté de ce résultat et sa puissance proviennent de sa nature « pour toute distribution » et du fait que les propriétés de la classe $\mathcal{H}$ des fonctions hypothèses sont reflétées à travers un seul paramètre : sa dimension de Vapnik-Chervonenkis.

Il est important de souligner, que d'après [Vap95] en p.81, ces bornes ne peuvent pas être significativement améliorées. Ce sont des bornes supérieures assez serrées dans la mesure où les bornes inférieures connues sont du même ordre de grandeur (de l'ordre de $\sqrt{d_{\mathcal{H}}/m}$ au lieu de $\sqrt{(d_{\mathcal{H}}/m)\ln(d_{\mathcal{H}}/m)}$) et donc les bornes supérieures ci-dessus sont optimales à un facteur logarithmique près, et sont par conséquent d'importance pratique aussi bien que théorique.

Remarque

Afin de pouvoir appliquer les bornes de la théorie à des problèmes pratiques, il faut pouvoir estimer de manière précise la dimension de Vapnik-Chervonenkis de l'espace d'hypothèses concerné. Malheureusement, l'estimation analytique de la dimension de Vapnik-Chervonenkis n'est possible que pour des classes de fonctions simples. Il faut donc souvent avoir recours à des *estimations heuristiques de* $d_{\mathcal{H}}$, par exemple en observant le risque empirique sur des échantillons de données indépendants et de tailles différentes, comme le propose Vapnik dans [VLC94]. Le problème de l'évaluation empirique de la dimension de Vapnik-Chervonenkis est un sujet de recherche important.

Remarque

Ce chapitre a essentiellement pour but d'introduire le lecteur à un certain nombre de questions fondamentales sur l'induction. Nous avons à cet effet laissé sous silence certaines subtilités. C'est ainsi que par exemple, nous avons supposé que la tâche de classification binaire était réalisée par un apprenant implémentant des fonctions indicatrices. Lorsque, pour cette même tâche, l'apprenant implémente des fonctions à valeur réelle, alors de nouvelles analyses plus fines du principe ERM peuvent être conduites. Ce sont en particulier les analyses portant sur les *classifieurs à large marge* (*large margin classifiers*). Une réalisation concerne les machines à vastes marges décrites dans le chapitre 9.

10.5 Discussion

Finalement, le principe inductif de minimisation du risque empirique (ERM) est-il légitime ? Les analyses théoriques (PAC pour les fonctions indicatrices et de Vapnik et Chervonenkis pour les fonctions quelconques) montrent que cela dépend de la taille de l'échantillon d'apprentissage et plus précisément du rapport $m/d_{\mathcal{H}}$ qui prend en compte la dimension de Vapnik-Chervonenkis de l'espace des fonctions hypothèse $\mathcal{H}$. Si en effet nous considérons les inégalités (22.36) et (22.38), nous voyons apparaître deux cas :

- $m/d_{\mathcal{H}}$ **est grand**. Dans ce cas, ϵ est petit, et le second terme des inégalités (22.36) et (22.38) devient petit. Le risque réel est alors proche du risque empirique, et dans ce cas une valeur faible du risque empirique garantit (en probabilité) une valeur faible du risque réel.
  Le principe ERM est alors justifié.

Copyright © 2010 Eyrolles.

- $m/d_{\mathcal{H}}$ **est petit**. Dans ce cas, une faible valeur du risque empirique ne garantit rien sur la valeur du risque réel. Pour minimiser le risque réel, il faut aussi minimiser le terme de droite des inégalités (22.36) et (22.38) en prenant en compte les deux termes simultanément. Or le premier terme dépend d'une fonction spécifique dans $\mathcal{H}$, tandis que le second terme dépend de l'ensemble des fonctions $\mathcal{H}$ par l'intermédiaire de sa dimension de Vapnik-Chervonenkis. Pour minimiser le terme de droite de ces inégalités, il faut donc maintenant faire de $d_{\mathcal{H}}$ un paramètre de contrôle. Nous verrons cette idée à l'œuvre dans un nouveau principe inductif proposé par Vapnik : le *principe de minimisation du risque structurel* (*SRM : Structural Risk Minimization*), voir la section 2.4.

11. L'induction par compression d'information

11.1 La théorie de l'induction selon Solomonoff

Dans un papier visionnaire, Solomonoff en 1964 [Sol64] a proposé une formalisation du problème de l'induction. Selon lui, tout problème d'inférence inductive peut être considéré comme un problème d'extrapolation d'une séquence de symboles binaires. Soit l'espace $\mathcal{S}$ des séquences infinies de symboles binaires, et une distribution *a priori* μ sur $\mathcal{S}$, avec $\mu(\boldsymbol{x})$ dénotant la probabilité d'une séquence commençant avec $\boldsymbol{x}$. Alors, étant donnée une séquence observée $\boldsymbol{x}$, le problème inductif est de prédire le prochain symbole dans la séquence. Cela peut se faire soit par prédiction directe du prochain symbole, soit par identification d'une règle sous-jacente à la séquence permettant de prédire le prochain symbole. On peut exprimer la probabilité que la séquence $\boldsymbol{x}$ se poursuive par le symbole a sachant que la séquence initiale est $\boldsymbol{x}$ par :

$$\mu(a|\boldsymbol{x}) \;=\; \frac{\mu(\boldsymbol{x}a)}{\mu(\boldsymbol{x})} \tag{22.39}$$

La tâche centrale de l'inférence inductive est alors de trouver une approximation de μ permettant d'estimer la probabilité conditionnelle qu'un segment $\boldsymbol{x}$ soit suivi d'un segment $\boldsymbol{y}$. Ceci est dans le cas général impossible. Il faut donc trouver des moyens d'approcher μ de manière raisonnable.

11.2 La complexité de Kolmogorov

La complexité algorithmique, souvent appelée complexité de Kolmogorov du nom de l'un de ses inventeurs, cherche à mesurer la complexité intrinsèque d'une chaîne de bits.

Définition 22.2 (Complexité algorithmique)

La complexité algorithmique *d'une chaîne de bits $\boldsymbol{x}$ est définie comme la longueur (mesurée en bits) du plus court programme qui, sans données supplémentaires, permet à une machine de Turing universelle $\mathcal{U}$ d'écrire la chaîne $\boldsymbol{x}$ et de s'arrêter.*

Formellement, cela s'écrit :

$$K(\boldsymbol{x}) \;=\; \underset{l(p)}{\mathrm{Min}}[\mathcal{U}(p) \;=\; \boldsymbol{x}] \tag{22.40}$$

où $l(p)$ est la longueur, mesurée en bits, du programme p.

La complexité algorithmique est une mesure de l'incompressibilité de $\boldsymbol{x}$. Considérons par exemple une chaîne $\boldsymbol{x}$ constituée uniquement de n 1. Cette chaîne est intuitivement très simple.

Copyright © 2010 Eyrolles.

Et de fait, il est facile d'écrire un programme pour la produire. Ce programme est essentiellement une boucle qui sera exécutée n fois. Le programme est donc de longueur proportionnelle à $\log_2 n$ soit en $O(\log_2 n)$. Nous noterons cette complexité de Kolmogorov par $K(x) = \log_2 n$. Un autre exemple est celui de l'expression du nombre transcendantal π, dont la séquence binaire : 11.001001000011111101101010001... apparemment aléatoire est en fait simple : la taille du plus petit programme capable de produire cette séquence est petite et constante, indépendante du nombre de bits produits. On a donc $K(\pi) = 1$. En revanche, une séquence réellement aléatoire ne peut pas être produite par un programme plus court que la séquence elle-même. Dans ce cas, on a donc : $K(x) = |x|$.

Sans faire justice de toutes les implications et subtilités de la théorie de la complexité algorithmique (nous renvoyons pour cela le lecteur à la « somme » de Li et Vitanyi [LV97]), il suffit pour nous de savoir qu'elle est liée profondément à une mesure de probabilité universelle. En effet, on peut associer à chaque programme p, c'est-à-dire chaîne de bits, sa probabilité de production par un tirage aléatoire de bits avec probabilité 1/2. Cette probabilité est : $Pr(p) = 2^{-l(p)}$, où $l(p)$ est la longueur de la chaîne de bits correspondant à p. Cela signifie qu'un programme court est plus probable qu'un programme long. Si un programme court produit une séquence longue, celle-ci ne peut être aléatoire puisqu'elle a une description simple. On est ainsi amené à définir la probabilité universelle d'une chaîne x par :

$$P_{\mathcal{U}}(x) = \sum_{p:\mathcal{U}(p)=x} 2^{-l(p)} \tag{22.41}$$

C'est la probabilité qu'un programme p tiré aléatoirement suivant une distribution de probabilités 1/2 produise la suite x par la machine de Turing universelle $\mathcal{U}$. Cette probabilité, sans être exactement indépendante de la machine employée, en dépend relativement peu. Par ailleurs, il est clair que cette probabilité est dominée par la probabilité $2^{-K(x)}$ du plus court programme pouvant produire la séquence x. Malheureusement, la taille $K(x)$ de ce plus court programme est non calculable effectivement. En effet, le seul moyen de trouver ce plus court programme serait d'essayer tous les programmes possibles, or l'exécution de certains d'entre eux risque de ne jamais se terminer. Il semble ainsi que nous soyons ramenés au problème précédent : nous ne pouvions définir une mesure de probabilité universelle μ, et nous ne pouvons pas plus définir la complexité algorithmique d'une chaîne de bits qui aurait pu nous permettre de calculer μ. Cependant, ce lien très profond entre mesure de probabilité et complexité algorithmique a fourni le terreau sur lequel ont été développés plusieurs procédés inductifs de nature plus heuristique.

11.3 Le principe de longueur de description minimale (*MDLP*)

Le principe de longueur minimale de description, (*Minimum Description Length principle* ou *MDLP*), peut s'expliquer par une analogie avec la théorie de l'information et la transmission de message entre un émetteur et un récepteur.

Supposons qu'un agent, appelé émetteur, veuille transmettre des données à un autre agent, appelé récepteur, de la manière la plus économique, c'est-à-dire en limitant autant que possible le nombre de bits transmis sur le canal qui les relie (une ligne téléphonique par exemple). Intuitivement, une manière de faire consisterait à d'abord transmettre une description générale des données (*e.g.* « je viens de voir passer deux oiseaux style corbeau ») puis à transmettre ce qui dans les données ne correspond pas au modèle, c'est-à-dire les exceptions (*e.g.* « sauf que l'un avait le bec jaune et l'autre le bout des ailes rouges »). Il existe bien sûr un compromis entre la complexité du modèle transmis et ce qu'il faut indiquer comme exception. Si le modèle transmis est très général (em e.g. « j'ai vu des objets volants »), il faudra fournir beaucoup d'informations pour décrire exactement les données à partir de ce modèle. Inversement, si le modèle

Copyright © 2010 Eyrolles.

est très précis (*e.g.* « j'ai vu un oiseau noir, de la taille d'un corbeau environ, de plumage noir avec le bec jaune, et un autre oiseau... »), il sera coûteux à transmettre, ne factorisant pas les généralités présentes dans les données. Le meilleur compromis consiste à trouver un modèle tel que la somme de sa description, et celle des irrégularités par rapport à ce modèle, soit la plus économique possible. C'est l'essence du principe de longueur de description minimale.

Définition 22.3 (Principe de longueur minimale de description *MDLP*)

La meilleure théorie, ou hypothèse, ou le meilleur modèle, rendant compte d'un échantillon d'apprentissage minimise la somme de :

1. *la longueur, mesurée en bits, de la* description de la théorie *;*
2. *la longueur, mesurée en bits, de la* description des données *lorsqu'elles sont décrites à l'aide de la théorie.*

Formellement, cela signifie que l'hypothèse optimale $h^\star$ vérifie :

$$h^\star = \underset{h \in \mathcal{H}}{\text{ArgMin}}\ L(h) + L(\boldsymbol{x}|h) \tag{22.42}$$

où $L(h)$ mesure la longueur de description de h, et $L(\boldsymbol{x}|h)$ mesure la longueur de description des données $\boldsymbol{x}$ en utilisant l'hypothèse h pour les coder.

Avec ce principe, on retrouve l'idée essentielle des principes inductifs contrôlant la complexité des hypothèses, à savoir qu'il faut faire place à un compromis entre la complexité de l'espace d'hypothèses mis en œuvre pour rendre compte des données et la fidélité aux données elles-mêmes. Un modèle des données trop précis peut ne pas avoir de bonnes performances en généralisation, de même qu'un modèle trop général.

Il est facile de voir que le principe de longueur minimale de description est lié à la règle de Bayes. En effet, d'après cette règle :

$$Pr(h|\boldsymbol{x}) = \frac{Pr(\boldsymbol{x}|h)\, Pr(h)}{Pr(\boldsymbol{x})}$$

Soit, en prenant l'opposé du logarithme de chaque côté de l'équation :

$$-\log Pr(h|\boldsymbol{x}) = -\log Pr(\boldsymbol{x}|h) - \log Pr(h) + \log Pr(\boldsymbol{x})$$

En tenant compte du fait que le facteur $Pr(\boldsymbol{x})$ est indépendant de l'hypothèse mise en œuvre pour rendre compte des données, maximiser $Pr(h|\boldsymbol{x})$, comme le préconise le principe du maximum de vraisemblance, revient à minimiser le terme de droite de l'équation précédente, soit :

$$-\log Pr(\boldsymbol{x}|h) - \log Pr(h)$$

Idéalement, la mesure de probabilité à utiliser serait la mesure universelle μ et donc celle de la complexité algorithmique K. C'est-à-dire qu'il faudrait prendre $Pr(\boldsymbol{y})$ comme étant égal à $2^{-K(\boldsymbol{y})}$ pour une séquence arbitraire $\boldsymbol{y}$. On choisirait alors l'hypothèse h minimisant :

$$K(\boldsymbol{x}|h) + K(h)$$

À défaut de pouvoir utiliser la mesure de probabilité μ ou la mesure de complexité algorithmique K, le principe de longueur minimale de description préconise de définir un *codebook* raisonable

Copyright © 2010 Eyrolles.

permettant la description de l'univers considéré, puis de coder les hypothèses et les données à l'aide de ce codebook. La mesure de complexité, ou de longueur de description, s'effectue alors en référence au code ainsi défini, chaque élément de ce code étant associé à un coût fixé par l'utilisateur. On retrouve alors l'équation (22.42) relative au *MDLP*.

EXEMPLE Régression par des polynômes

On suppose que l'on a un échantillon de données $\mathcal{S} = \{(x_1, u_1), (x_2, u_2), \ldots, (x_m, u_m)\}$, dans lequel les formes x_i et les étiquettes u_i sont des nombres réels. On cherche à pouvoir prédire la valeur u pour une forme x donnée. Pour cela, on fait l'hypothèse que l'on peut rendre compte des données avec un polynôme. En général, plus le degré du polynôme est élevé, plus l'adéquation aux données est étroite (au sens par exemple des moindres carrés). À la limite, pour tout ensemble de m points (x_i, u_i), il est possible de trouver un polynôme de degré $m-1$ passant exactement par les m points. Mais ce polynôme n'aura en général aucun pouvoir prédictif (le lecteur peut s'en convaincre aisément en essayant de prédire les cours de la bourse de cette manière). Supposons que l'on cherche une hypothèse sous la forme d'un polynôme p_k de degré k. Pour décrire un tel polynôme, il faut $k+1$ coefficients que nous supposerons décrits avec une précision de d bits. Une hypothèse, c'est-à-dire un polynôme, sera alors décrite par : $k\,d + \mathcal{O}(\log k\,d)$ bits. (Le deuxième terme de la somme provient de considérations techniques sur le fait que le programme à fournir à la machine de Turing doit être autodélimitant.)

Il faut maintenant examiner le coût de description des données (x_i, u_i) à l'aide d'un polynôme. En général, on agit comme si le polynôme correspondait au vrai modèle sous-jacent aux données et que celles-ci étaient distribuées suivant une loi gaussienne autour de la valeur prédite par le polynôme : $u_i = p_k(x_i) + \varepsilon$ avec ε une variable centrée en $p_k(x)$ et de variance constante. Dans ce cas la probabilité d'observer la valeur u_i au lieu de $p_k(x_i)$ est de l'ordre de $e^{-(p_k(x_i)-u_i)^2}$. À l'aide de la mesure de probabilité universelle, cette grandeur est codée sur $s(p_k(x_i) - u_i)^2$ bits, où s est une constante de normalisation.

Comme l'erreur commise par l'hypothèse p_k au sens des moindres carrés est :

$$\text{erreur}(p_k) = \sum_{i=1,m} (p_k(x_i) - u_i)^2$$

on trouve, en négligeant le terme $\mathcal{O}(\log k\,d)$ que le coût d'expression des données $\mathcal{S}$ à l'aide du polynôme p_k est de :

$$k\,d + s \cdot \text{erreur}(p_k)$$

La meilleure hypothèse est, selon le principe de minimisation de longueur de description (*MDLP*), le polynome (p_k qui minimise cette expression. □

Le principe *MDLP* a été appliqué dans de nombreux autres contextes. Nous citerons par exemple le choix d'arbres de décision (voir chapitre 13) pour lequel Quinlan et Rivest [QR89] ont proposé une mesure de coût des arbres prenant en compte les nœuds de l'arbre et les branchements possibles, ainsi que les exceptions. Dans un contexte différent de celui de l'induction, [Cor96] propose d'utiliser une version du *MDLP* pour rendre compte du raisonnement par analogie, considéré comme une forme de transmission économique d'information. Si le *MDLP* a permis d'obtenir des résultats dans plusieurs applications, il n'en reste pas moins une technique largement empirique, dans laquelle on a remplacé la nécessité de fournir des probabilités *a priori*, comme dans l'approche bayésienne, par celle de concevoir un codebook avec ses coûts associés. Pour le moment on ne connaît pas de technique fondée rigoureusement pour résoudre ce problème. Par ailleurs, la recherche d'une description la plus courte est connue comme étant un problème NP-complet dans de nombreux formalismes. Il est donc nécessaire d'avoir recours à des techniques heuristiques de recherche d'hypothèses.

Copyright © 2010 Eyrolles.

11.4 Analyse : compression et pouvoir inductif

Si intuitivement il semble satisfaisant de penser qu'un modèle « simple » des données est plus susceptible qu'un modèle complexe de décrire les régularités sous-jacentes, et donc de permettre des prédictions, cela ne suffit pas à garantir le lien entre compression des informations et pouvoir inductif. Sans entrer dans les détails de ce lien ici, nous mentionnerons deux études se rapportant à cette question.

11.4.1 Un théorème de Vapnik

Dans son livre de 1995 [Vap95] en pp.102-105, Vapnik donne une preuve justifiant le principe de longueur minimale de description pour la tâche de classification. Il montre que le coefficient de compression obtenue, c'est-à-dire le rapport $r(h)$ entre la taille de description comprimée des données et la taille de leur description brute, est lié à la probabilité d'erreur de classification sur des données futures. La preuve, sans être difficile, dépasse le cadre du présent ouvrage, et nous n'en donnons que le résultat. Elle repose sur un argument de convergence uniforme appliqué au codebook utilisable pour décrire l'espace des fonctions de classification.

Théorème 22.5 (*MDLP* et probabilité d'erreur en classification, Vapnik,1995))

Si, en utilisant un codebook structuré, on trouve une hypothèse h permettant de comprimer l'expression de la chaîne de bits $u_1, u_2, \ldots, u_m$ des étiquettes des formes d'apprentissage $x_1, x_2, \ldots, x_m$ d'un facteur $R(h)$, alors, avec probabilité au moins $1-\eta$, la probabilité d'erreur de classification par l'hypothèse h est bornée par :

$$R_{Réel}(h) \;<\; 2\left(r(h)\ln 2 \;-\; \frac{\ln \eta}{m}\right)$$

Ce théorème est intéressant dans la mesure où, par contraste avec les théorèmes de pertinence de l'*ERM*, il ne fait pas intervenir directement de propriétés statistiques des données, ni de risque empirique (nombre d'erreurs de classification en apprentissage). Malheureusement, ce théorème ne dit pas comment construire un bon codebook. Nous en verrons une raison en analysant le *No-Free-Lunch theorem* (5.1).

11.4.2 Les algorithmes d'Occam en apprentissage *PAC*

Le lien entre compression d'information et généralisation a été également étudié dans le cadre de l'apprentissage de fonctions indicatrices, c'est-à-dire de concepts. Dans ce cadre, en supposant que les exemples x soient définis sur $\{0,1\}^d$ ou sur $\mathbb{R}^d$, que l'échantillon d'apprentissage $\mathcal{S}$ comporte m exemples étiquetés suivant une fonction cible f : $((x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_m, f(x_m))$, alors un algorithme d'Occam est un algorithme qui, prenant $\mathcal{S}$ en entrée, produit une hypothèse $h \in \mathcal{H}$ cohérente avec $\mathcal{S}$ et qui est succincte au sens où taille(h) est une fonction croissant suffisamment lentement en fonction de d, taille(f) et m. Plus précisément :

Copyright © 2010 Eyrolles.

Définition 22.4 (Algorithme d'Occam)

Soient deux constantes $\alpha \geq 0$ et $0 \leq \beta < 1$. On dit qu'un algorithme d'apprentissage est un algorithme d'Occam *si, à partir d'un échantillon d'apprentissage étiqueté par un concept cible f, il produit une hypothèse h vérifiant :*

1. *h est cohérente avec $\mathcal{S}$*
2. *taille$(h) \leq (d \cdot$ taille$(f))^{\alpha}\, m^{\beta}$*

Il est clair que si $m \gg d$, alors les m bits correspondants aux étiquettes de $x_1, x_2, \ldots, x_m$ sont effectivement comprimés en une chaîne d'au plus m^{β} bits. Sinon, il faut bien exprimer que l'hypothèse la plus courte peut dépendre au moins linéairement de taille(f).

Il existe alors un théorème prouvant qu'un tel algorithme, si on lui fournit un échantillon de taille

$$m \geq a\left(\frac{1}{\epsilon}\log\frac{1}{\delta} + \left(\frac{(d \cdot \text{taille}(f))^{\alpha}}{\epsilon}\right)^{\frac{1}{1-\beta}}\right)$$

où a est une constante > 0, produit une hypothèse h de probabilité d'erreur de classification en généralisation $\leq \epsilon$ avec une probabilité $1 - \delta$ où $\delta \leq 1$.

Un lien est donc établi là aussi entre compression et induction, même si la portée de ce théorème est limitée par le fait qu'il ne s'applique que pour des espaces d'hypothèses $\mathcal{H}$ de cardinal fini. Pour plus de détails, on se reportera par exemple à [KV94].

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Bibliographie

|AB92| M. Anthony and N. Biggs. *Computational Learning Theory*. Cambridge University Press, 1992.

|AB96| M. Anthony and P. Bartlett. *Neural network learning : theoretical foundations*. Cambridge University Press, 1996.

|ABC+01| J. C. Amengual, J. M. Benedí, F. Casacuberta, A. Castaño, A. Castellanos, V. M. Jiménez, D. Llorens, A. Marzal, M. Pastor, F. Prat, E. Vidal, and J. M. Vilar. The EuTrans-I speech translation system. *Machine Translation*, 15(1) :75–103, 2001.

|ABR64| M. Aizerman, E. Braverman, and L. Rozonoèr. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25 :821–837, 1964.

|ABT07| D. Aberdeen, O. Buffet, and O. Thomas. Policy-Gradient for PSRs and POMDPs. In Marina Meila and Xiaotong Shen, editors, *11th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 2, pages 3–10, San Juan, Puerto Rico, 2007. JMLR.

|ACBF02| P. Auer, N. Cesa-Bianchi, and P. Fisher. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47 :235–256, 2002.

|ACH05| S. Agarwal, C. Cortes, and R. Herbrich, editors. *Proc. of the NIPS 2005 Workshop on "Learning to rank"*. 2005.

|AD91| H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, pages 547–552, Anaheim, California, 1991. AAAI Press.

|AD94| H. Almuallim and T. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69 :279–305, 1994.

|AFvZ02| P. Adriaans, H. Fernau, and M. van Zaannen, editors. *Grammatical Inference : Algorithms and Applications, Proceedings of ICGI '02*, volume 2484 of *LNAI*, Berlin, Heidelberg, 2002. Springer-Verlag.

|Aha97| D. Aha, editor. *Lazy learning*. Kluwer, 1997.

|AHWY03| C. Aggarwal, J. Han, J. Wang, and P. Yu. A framework for clustering evolving data streams. In *Proc. of VLDB 2003*, pages 81–92, Berlin, Germany, 2003.

|Aka73| H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Proc. 2nd International Symposium on Information Theory*, pages 267–281, Akademiai Kiado, Budapest, 1973.

Copyright © 2010 Eyrolles.

|Aka74| H. Akaike. A new look at the statistical model identification. *IEEE Trans. Automat. Contrl.*, 19(6) :716–723, 1974.

|Aka81| H. Akaike. Likelihood of a model and information criteria. *Journal of Econometrics*, 1(16) :3–14, 1981.

|Aka83| H. Akaike. Information measures and model selection. In *Proc. 44th Session of the International Statistical Institute*, volume 1, pages 277–291, 1983.

|Aka85| H. Akaike. Prediction and entropy. In A. Atkinson and E. Fienberg, editors, *A Celebration of Statistics*, pages 1–24. Springer-Verlag, 1985.

|Alb75| J. Albus. A new approach to manipulator control : Cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement and Control*, 97 :220–227, 1975.

|Alb78| A. Albert. *Quelques apports nouveaux à l'analyse discriminante.* PhD thesis, Faculté des sciences, Université de Liège, 1978.

|Alb81| J. Albus. *Brains, Behavior and Robotics.* BYTE Books, McGraw-Hill, 1981.

|Alp99| E. Alpaydin. Combined 5 x 2 cv f test for comparing supervised classification learning algorithms. *Neural Computation*, 11 :1885–1892, 1999.

|AM97| N. Abe and H. Mamitsuka. Predicting protein secondary structure using stochastic tree grammars. *Machine Learning*, 29 :275–301, 1997.

|Ama68| S. Amarel. *On representation of problems of reasoning about actions*, volume 3, pages 131–171. Edingburgh University Press, 1968. Reprinted in Webber, B. and Nilsson, N. (eds.), "Readings in artificial intelligence", Tioga, 1981, pp.2-22.

|AMN94| H. Ahonen, H. Mannila, and E. Nikunen. Forming grammars for structured documents : An application of grammatical inference. In *Second International Colloquium on Grammatical Inference and Applications, ICGI'94*, volume 862 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1994.

|And82| J. Anderson. Logistic discrimination. In *Handbook of statistics, Vol.2, Classification, pattern recognition and reduction of dimensionality*, chapter 7, pages 169–191. North-Holland, 1982.

|And92| D. Andler, editor. *Introduction aux sciences cognitives.* Folio. Essais, 1992.

|Ang78| D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 29(3) :741–765, 1978.

|Ang82| D. Angluin. Inference of reversible languages. *Communications of the ACM*, 29 :741–765, 1982.

|Ang88| D. Angluin. Queries and concept learning. *Machine Learning journal*, 2 :319–342, 1988.

|Ang98| E. Angel. *La rugosité des paysages : une théorie pour la difficulté des problèmes d'optimisation combinatoire relativement aux méta-heuristiques.* PhD thesis, Laboratoire de Recherche en Informatique (LRI) de l'université Paris-Orsay, 1998.

|Ang03| D. Angluin. Queries revisited. *Theoretical Computer Science*, 313(2) :175–194, 2003.

|AP93| P. Angeline and J. Pollack. Competitive environments evolve better solutions for complex tasks. In S. Forrest, editor, *Proc. of the 5th Int. Conf. on Genetic Algorithms, ICGA-93*, pages 264–270. Morgan-Kaufmann, 1993.

Copyright © 2010 Eyrolles.

|AP94| A. Aamodt and E. Plaza. Case-based reasoning : Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1) :39–52, 1994.

|AS94| R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Very Large Data Bases (VLDB-94)*, pages 487–499, Santiage, Chile, 1994.

|ASS00| E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary : A unifying approach for margin classifiers. *Journal of Artificial Intelligence Research*, 1 :113–141, 2000.

|AU72| A. Aho and J. Ullman. *The theory of Parsing, Translation and Compiling, Vol 1 : Parsing*. Prentice-Hall, 1972.

|Aub87| J.-P. Aubin. *Analyse fonctionnelle appliquée (t.1 et t.2)*. PUF, 1987.

|AV02| P. Adriaans and M. Vervoort. The EMILE 4.1 grammar induction toolbox. In Adriaans et al. [AFvZ02], pages 293–295.

|AW01| K. Azoury and M. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning journal*, 43 :211–246, 2001.

|Axe97| R. Axelrod, editor. *The complexity of cooperation*. Princeton University Press, 1997.

|BA97| L. Brelow and D. Aha. Simplifying decision trees : a survey. *The Knowledge Engineering Review*, 12(1) :1–40, 1997.

|BA04| K. Burnham and D. Anderson. Multimodel inference. Understanding AIC and BIC in model selection. *Sociological Methods and Research*, 33(2) :261–304, 2004.

|Bak75| J. Baker. Stochastic modeling for automatic speech understanding. In R. Reddy, editor, *Speech Recognition*, pages 512–542. Academic Press, New York, 1975.

|Bau72| L. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3 :1–8, 1972.

|Bau89| E. Baum. A proposal for more powerful learning algorithms. *Neural Computation*, 1 :201–207, 1989.

|Bax00| J. Baxter. Reinforcement learning in POMDP's via direct gradient ascent. In *17th International Conference on Machine Learning (ICML'2000)*, pages 41–48. Morgan Kaufmann, 2000.

|BB92| A. Belaïd and Y. Belaïd. *Reconnaissance des formes*. InterEditions, 1992.

|BB01| P. Baldi and S. Brunak. *Bioinformatics : the machine learning approach*. MIT Press, 2001.

|BB08| L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Neural Information Processing Systems (NIPS-07)*, volume 20, pages 161–168. NIPS Foundation (http ://books.nips.cc), 2008.

|BB09| M.-F. Balcan and A. Blum. A discriminative model for semi-supervised learning. *ACM Journal*, 2009.

|BBM96| A. Barron, L. Birgé, and P. Massart. Risk bounds for model selection via penalization. *Probability Theory Related Fields*, 1996.

Copyright © 2010 Eyrolles.

|BC97| A. Brazma and K. Cerans. Noise-tolerant efficient inductive synthesis of regular expressions from good examples. *New Generation Computing*, 1997.

|BD00| H.-H. Bock and E. Diday. *Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data*. Springer Verlag, 2000.

|BDEL03| S. Ben-David, N. Eiron, and P. Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66 :496–514, 2003.

|BE02| O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2 :499–526, 2002.

|BEHW89| A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association of Computer Machinery*, 36 :929–965, 1989.

|Bel57a| R. Bellman. *Dynamic Programming*. University Press, Princeton, 1957.

|Bel57b| R. Bellman. A markov decision process. *Journal of Mathematical Mechanics*, 6 :679–684, 1957.

|Bel61| R. Bellman. *Adaptive Control Processes : A guided tour*. Princeton University Press, 1961.

|Ber00| M. Bernard. *Induction de programmes logiques*. Laboratoire EURISE, Université de St-Etienne, 2000.

|BF72| A. Biermann and J. Feldman. On the synthesis of finite-state machines from samples of their behaviour. *IEEE Transactions on Computers*, C-21 :592–597, 1972.

|BF85| D. Berry and B. Fristedt. *Bandit problems : sequential allocation of experiments*. Chapman and Hall, London, UK, 1985.

|BFOS84| L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth Inc., 1984.

|BGH89| L. Booker, D. Goldberg, and J. Holland. Classifer systems and genetic algorithms. *Artificial Intelligence journal*, 40 :235–282, 1989.

|BGV92| B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In David Haussler, editor, *Proc. of the annual Conference on Computational Learning Theory (COLT-92)*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

|BHK98| J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, Madison (WI), 1998. Morgan Kaufman.

|BHS+07| G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, and S. V. N. Vishwanathan, editors. *Predicting Structured Data*. MIT Press, 2007.

|BHW08| M.-F. Balcan, S. Hanneke, and J. Wortman. The true sample complexity of active learning. In *Computational Learning Theory (COLT-08)*, 2008.

|Bir67| B Birkhof. *Lattice Theory*. American Mathematical Society, 1967.

|Bis95| C. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

|Bis06| C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

|BK92| E. Baum and L. Kevin. Query learning can work poorly when a human oracle is used. In *Int. Joint Conf. in Neural Networks*, Beijing, China, 1992.

Copyright © 2010 Eyrolles.

|BKL00| A. Barberousse, M. Kistler, and P. Ludwig. *La philosophie des sciences au XXème siècle*. Fammarion. Champs Université, 2000.

|BKS04| P. Beame, H. Kautz, and A. Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22 :319–351, 2004.

|BL97| A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence journal*, (97) :245–271, 1997.

|BL05| Léon Bottou and Yann LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2) :137–151, 2005.

|BL07| Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*. MIT Press, 2007.

|Blu94| A. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. *SIAM Journal on Computing*, 23(5) :990–1000, 1994.

|Blu97| A. Blum. On-line algorithms in machine learning. Technical report, 1997.

|BM94| B. Bouchon-Meunier. *La logique floue*. Presses Universitaires de France (coll. « Que sais-je ? », 1994.

|BM95a| J. Boyan and A. Moore. Generalization in reinforcement learning : Safely approximating the value function. In *Advances in Neural Information Processing Systems*, chapter 7. MIT Press, 1995.

|BM95b| I. Bratko and S. Muggleton. Applications of Inductive Logic Programming. *Communications of the ACM*, 38(11) :65–70, 1995.

|BM98| A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of the 11th Annual Conf. on Computational Learning Theory, COLT'98*, pages 92–100. Morgan Kaufmann, 1998.

|BM99| L. Baird and A. Moore. *Gradient descent for general reinforcement learning*, volume 11. MIT Press, 1999.

|BMD07| S. Bayoudh, L. Miclet, and A. Delhay. Learning by analogy : a classification rule for binary and nominal data. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, volume 20, pages 678–683, 2007.

|BN99| A. Becker and P. Naïm. *Les réseaux bayésiens : modèles graphiques de la connaissance*. Eyrolles, 1999.

|BNKF98| W. Banzhaf, P. Nordin, R. Keller, and F. Francone. *Genetic Programming : An introduction*. Morgan Kaufmann, 1998.

|BPS94| A. Borgida and P. Patel-Schneider. Complete algorithm for subsomption in the classic description logic. *Artificial Intelligence Research*, 1 :278–308, 1994.

|BS97| A. Belaïd and G. Saon. Utilisation des processus markoviens en reconnaissance de l'écriture. *Revue Traitement du Signal*, 14,2 :161–177, 1997.

|BS08| O. Buffet and O. Sigaud, editors. *Processus décisionnels de Markov en intelligence artificielle (volume 2)*. Hermes-Lavoisier, 2008.

|BSR+05| Ch. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05 : Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM Press.

Copyright © 2010 Eyrolles.

|BT96| D. Bertsekas and J. Tsitsiklis. *Neuro-dynamic programming.* Athena Scientific, 1996.

|BTC96| A. Brunie-Taton and A. Cornuéjols. Classification en programmation génétique. In *Journées Francophones d'Apprentissage (JFA-96)*, pages 303–316, 1996.

|BTW00| J. Baxter, A. Tridgell, and L. Weaver. Learning to play chess using temporal-differences. *Machine Learning Journal*, 40 (3) :243–263, 2000.

|BU92| C. Brodley and P. Utgoff. Mulltivariate decision trees. Technical report, COINS, Departement of Computer Science, University of Massachussets., Amherst, USA, December 1992.

|Bun88| W. Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36 :149–176, 1988.

|BvLR| O. Bousquet, U. von Luxburg, and G. Rätsch, editors. *Advanced Lectures on Machine Learning (ML Summer School 2003)*, volume LNAI-3176. Springer-Verlag.

|CAL94| D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning Journal*, 15(2) :201–221, 1994.

|Car97a| R. Carrasco. Accurate computation of the relative entropy between stochastics regulars grammars. http ://www.dlsi.ua.es/ carrasco/rcc.html, 1997.

|Car97b| R. Caruana. Multitask learning. *Machine Learning*, 28(1) :41–75, 1997.

|CAVR98| P. Cruz-Alcázar and E. Vidal-Ruiz. Learning regular grammars to model musical style : Comparing different coding schemes. In V. Honavar and G. Slutzki, editors, *Grammatical Inference, ICGI'98*, volume 1433 of *Lecture Notes in Artificial Intelligence*, pages 211–222. Springer Verlag, 1998.

|CBFH+97| N. Cesa-Bianchi, Y. Freund, D. Haussler, R. Shapire, and M. Warmuth. How to use expert advice. *Journal of American Computing Machines*, 44(3) :427–485, 1997.

|CBL01| N. Cesa-Bianchi and G. Lugosi. Worst-case bounds for the logarithmic loss of predictors. *Machine Learning journal*, 43 :247–264, 2001.

|CBL06| N. Cesa-Bianchi and G. Lugosi. *Prediction, learning and games.* Cambridge University Press, 2006.

|CCM07| A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for computing the entropy of a stream. In *SODA-07*, 2007.

|CCM08| Alexander Clark, François Coste, and Laurent Miclet, editors. *Grammatical Inference : Algorithms and Applications, 9th International Colloquium, ICGI 2008, Saint-Malo, France, September 2008, Proceedings*, volume 5278 of *Lecture Notes in Computer Science*. Springer, 2008.

|CE05| A. Clark and R. Eyraud. Identification in the limit of substitutable context-free languages. In *Proceedings of ALT'05*, number 3734 in LNAI, pages 283–296. Springer, 2005.

|Cel89| G. Celeux. *Classification automatique des données.* Dunod, 1989.

|Cer95| R. Cerf. An asymptotic theory for genetic algorithms. In Alliot, Lutton, Ronald, Schoenauer, and Snyers, editors, *Proceedings of the European Conference on Artificial Evolution (AE95)*, pages 37–53. Springer Verlag, LNCS-1063, 1995.

|CFM05| A. Cornuéjols, C. Froidevaux, and J. Mary. Comparing and combining feature estimation methods for the analysis of microarray data. In *JOBIM-05 : Journées Ouvertes Biologie Informatique Mathématiques*, Lyon, France, 2005.

Copyright © 2010 Eyrolles.

|CGJ96| D. Cohn, Z. Ghaharamani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4 :129–145, 1996.

|CGV94| A. Castellanos, I. Galiano, and E. Vidal. Application of OSTIA to machine translation tasks. In *Grammatical Inference and Applications, ICGI'94*, number 862 in Lecture Notes in Artificial Intelligence, pages 93–105. Springer Verlag, 1994.

|CH67| T. Cover and P. Hart. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, 13(1), 1967.

|Che01| Y. Chevaleyre. *Apprentissage de règles à partir de données multi-instances.* PhD thesis, Laboratoire d'informatique de Paris 6, 2001.

|CL96| G. Caraux and Y. Lechevallier. Règles de décision de Bayes et méthodes statistiques de discrimination. *Revue d'Intelligence Artificielle*, 10(2-3) :219–283, 1996.

|Cla06a| A. Clark. Learning Deterministic Context Free Grammars : the Omphalos Competition. *Machine Learning Journal*, page To be published, 2006.

|Cla06b| A. Clark. PAC-Learning Unambiguous NTS Languages. In *Proceedings of ICGI*, september 2006.

|CM98| V. Cherkassky and F. Mulier. *Learning from data. Concepts, theory and methods.* Wiley Interscience, 1998.

|CM99| J. Chodorowski and L. Miclet. Apprentissage et évaluation de modèles de langage par des techniques de correction d'erreurs. In *Actes de la 6e conférence annuelle sur le Traitement Automatique des Langues Naturelles*, pages 253–262, 1999.

|CNBYM01| Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Proximity searching in metric spaces. *ACM Computing Surveys*, 2001.

|CO94a| R. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. *ICGI'94*, Lecture Notes in Artificial Intelligence 862 :139–152, 1994. Subseries of Lectures Notes in Computer Science.

|CO94b| R. C. Carrasco and J. Oncina, editors. *Grammatical Inference and Applications, Proceedings of ICGI '94*, number 862 in LNAI, Berlin, Heidelberg, 1994. Springer-Verlag.

|Cor96| A. Cornuéjols. Analogie, principe d'économie et complexité algorithmique. In *Journées Francophones d'Apprentissage (JFA-96)*, pages 233–247, 1996.

|Cor09| A. Cornuéjols. On-line learning : Where are we so far ? In Michael May, editor, *Blueprint on Knowledge Discovery in Ubiquitous Systems*. 2009.

|CPSK07| K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. Kurgan. *Data mining. A knowledge discovery approach.* Springer, 2007.

|CR71| S. Crespi-Reghizzi. An effective model for grammatical inference. *Proceedings of the IFIP Congress*, pages 524–529, 1971. North-Holland.

|CR74| C. Cook and A. Rosenfeld. Some experiments in grammatical inference. *NATO ASI on Computer Oriented Learning Process*, pages 157–171, 1974. Bonas, France.

|Cra85| N. Cramer. A representation for the adaptative generation of simple sequential programs. In *Proceedings of the International Conference on Genetic Algorithms and the Applications*, pages 183–187. Carnegie-Mellon University, Pittsburgh, PA, 1985.

|Cre97| D. Crevier. *A la recherche de l'intelligence artificielle.* Flammarion, Champs, 1997.

Copyright © 2010 Eyrolles.

|CS02| K. Crammer and Y. Singer. Pranking with ranking. In *NIPS 14*, 2002.

|CSM04| A. Cornuéjols, M. Sebag, and J. Mary. Classification d'images à l'aide d'un codage par motifs fréquents. In *Workshop sur la fouille d'images (RFIA-04)*, Toulouse (France), 2004.

|CST00| N. Cristianini and J. Shawe-Taylor. *Support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

|CT91| T. Cover and A. Thomas. *Elements of Information Theory*. Wiley, 1991.

|CTC00| A. Cornuéjols, A. Tiberghien, and G. Collet. A new mechanism for transfer between conceptual domains in scientific discovery and education. *Foundations of Science*, 5(2) :129–155, 2000.

|CW08| E. Candès and M. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 21(March, 2008) :21–30, 2008.

|Dar59| C. Darwin. *On the origin of species*. London : Murray, 1859.

|Das90| B. Dasarathy, editor. *Nearest neighbor (NN) norms : NN Pattern Classification Techniques*. IEEE Computer Society Press, 1990.

|DB93| L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1993.

|DB95| T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2 :263–286, 1995.

|DD97| L. De Raedt and L. Dehaspe. Clausal Discovery. *Machine Learning*, 26 :99–146, 1997.

|DDE05| P. Dupont, F. Denis, and Y. Esposito. Links between probabilistic automata and hidden markov models : probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38(9) :1349–1371, 2005. Special Issue on Grammatical Inference Techniques and Applications.

|DDG96| F. Denis, C. D'Halluin, and R. Gilleron. PAC learning with simple examples. In *13th Symposium on Theoretical Aspects of Computer Science, STACS'96*, Lecture Notes in Computer Science, pages 231–242, 1996.

|De 92| L. De Raedt. *Interactive Theory Revision : an Inductive Logic Programming Approach*. Academic Press, 1992.

|DEKM98| R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis. Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

|Del92| J-P. Delahaye. L'altruisme recomposé ? *Pour la Science*, 181, 1992.

|Dem06| J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7 :1–30, 2006.

|DF97| T. Dietterich and N. Flann. Explanation-based learning and reinforcement learning : a unified view. *Machine Learning journal*, 28 :169–210, 1997.

|DG97| F. Denis and R. Gilleron. PAC learning under helpful distributions. In *Proceedings of the congress on Algorithmic Learning Theory, ALT'97*, 1997.

|DGL96| L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*. Springer Verlag, 1996.

Copyright © 2010 Eyrolles.

|DH73| R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

|DH00| C. Drummond and R. Holte. Explicitly representing expected cost : An alternative to ROC representation. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00)*, pages 198–207, 2000.

|DHS01| R. Duda, P. Hart, and D. Stork. *Pattern classification.* Wiley, 2001.

|Die98| T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10 :1895–1923, 1998.

|Die00| T. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13 :227–303, 2000.

|DK82| P. Devijver and J. Kittler. *Pattern Recognition : a statistical approach.* Prentice-Hall, 1982.

|DKBM00| E. Diday, Y. Kodratoff, P. Brito, and M. Moulet, editors. *Induction symbolique numérique à partir de données.* Cepadues, 2000.

|DKVY06| T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proceedings of the 38th Symposium on the Interface of Statistics, Computing Science, and Applications (Interface '06)*, 2006.

|DL00| M. Dash and H. Liu. Feature selection for clustering. In *Proc. of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, volume Lecture Notes In Computer Science ; Vol. 1805, pages 110–121, 2000.

|DL03| M. Dash and H. Liu. Consistency-based search in feature selection. *Artificial Intelligence journal*, (151) :155–176, 2003.

|dlH05| C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 2005.

|dlH10| C. de la Higuera. *Grammatical Inference.* Cambridge University Press, 2010.

|dlHO06| C. de la Higuera and J. Oncina. Learning context-free languages. *Artificial Intelligence Review*, 2006.

|dlHTV+05a| C. de la Higuera, F. Thollard, E. Vidal, F. Casacuberta, and R. Carrasco. Probabilistic finite-state machines - part i. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

|dlHTV+05b| C. de la Higuera, F. Thollard, E. Vidal, F. Casacuberta, and R. Carrasco. Probabilistic finite-state machines - part ii. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

|DLLP97| T. Dietterich, R. Lathrop, and T. Lozano-Pérez. Solving the multi-instance problem with axis-parallel rectangles. *Artificial Intelligence journal*, 89 :31–71, 1997.

|DM92| B. Dolsak and S. Muggleton. The application of inductive logic programming to finite element mesh design. In S. Muggleton, editor, *Inductive Logic Programming.* Academic Press, 1992.

|DM98| P. Dupont and L. Miclet. Inférence grammaticale régulière : fondements théoriques et principaux algorithmes. Technical report, INRIA N^o 3449, 1998.

|DMS+02| G. Dreyfus, J. Martinez, M. Samuelides, M. Gordon, F. Badran, S. Thiria, and L. Hérault. *Réseaux de neurones.* Eyrolles, 2002.

Copyright © 2010 Eyrolles.

|DMS+08| G. Dreyfus, J.-M. Martinez, M. Samuelides, M. Gordon, F. Badran, and S. Thiria, editors. *Apprentissage statistique*. Eyrolles, 2008.

|DMV94| P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference? In *Grammatical inference and applications, Second International Colloquium on Grammatical Inference, ICGI'94*, volume 862 of *Lecture Notes in Artificial Intelligence*, pages 25–37. Springer-Verlag, 1994.

|dO00| A. de Oliveira, editor. *Grammatical Inference : Algorithms and Applications, Proceedings of ICGI '00*, volume 1891 of *LNAI*, Berlin, Heidelberg, 2000. Springer-Verlag.

|Dod03| Y. Dodge. *Premier pas en statistique*. Springer-Verlag, 2003.

|Dom99| P. Domingos. Metacost : A general method for making classifiers cost sensitive. In *Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 155–164, 1999.

|Don06| D. Donohoe. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4) :1289–1306, 2006.

|DP97| P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning journal*, 29 :103–130, 1997.

|Dup94| J-P. Dupuy. *Aux origines des sciences cognitives*. Éditions de la Découverte, 1994.

|Dup96| P. Dupont. *Utilisation et apprentissage de modèles de langages pour la reconnaissance de la parole continue*. PhD thesis, ENST, 1996.

|DVD96| L. Dehaspe, W. Van Laer, and L. De Raedt. Claudien : The discovery engine user's guide 3.0. Technical Report CW 239, K.U. Leuven, September 1996.

|Dze93| S. Dzeroski. Handling imperfect data in Inductive Logic Programming. In *Proceedings of Fourth Scandinavian Conference on Artificial Intelligence*, pages 111–125. IOS Press, 1993.

|EdlHJ06| R. Eyraud, C. de la Higuera, and J.-C. Janodet. Lars : a learning algorithm for rewriting systems. *Machine Learning Journal*, 2006.

|EF05| G. Elidan and N. Friedman. Learning hidden variable networks : The information bottleneck approach. *Journal of Machine Learning Research (JMLR)*, 6 :81–127, 2005.

|Efr82| B. Efron. *The Jaccknife, the Bootstrap and other resampling plans*. SIAM, 1982.

|EH81| B. Everitt and D. Hand. *Finite mixture distributions*. Chapman and Hall, 1981.

|EHJT04| Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32 :407–499, 2004.

|Elk01| C. Elkan. The foundations of cost-sensitive learning. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

|EMS97| F. Esposito, D. Malerba, and G. Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Trans. on Patterni Analysis and Machine Intelligence*, 19(5) :476–493, 1997.

|ENF07| G. Elidan, I. Nachman, and N. Friedman. "Ideal Parent" Structure Learning for Continuous Variable Networks. *Journal of Machine Learning Research (JMLR)*, 8(1799-1833), 2007.

|Eng96| P. Engel. *Philosophie et psychologie*. Folio. Essais, 1996.

Copyright © 2010 Eyrolles.

|Fer01| H. Fernau. Learning XML grammars. In P. Perner, editor, *Machine Learning and Data Mining in Pattern Recognition MLDM'01*, number 2123 in LNCS, pages 73–87. Springer-Verlag, 2001.

|FHT98| J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression : a statistical view of boosting. `http://www-stat.stanford.edu/~jhf/#reports`, 1998.

|FL94| W. Furman and B. Lindsay. Testing for the number of components in a mixture of normal distributions using moment estimators. *Computational Statistics and Data Analysis*, 17 :473–492, 1994.

|Fle80| R. Fletcher. *Practical methods of optimization.* Wiley, 1980.

|FLSS04| Y. Freund, R. Lyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4(6) :933–969, 2004.

|FN71| R. Fikes and N. Nilsson. Strips : A new approach to the application of theorem proving to problem solving. *Artificial Intelligence journal*, 2 :189–208, 1971.

|FN72| R. Fikes and N. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence journal*, 3 :251–288, 1972.

|Fog98| L. Fogel, editor. *Evolutionary computation : towards a new philosophy of machine intelligence.* IEEE Press, 1998.

|Fog99| L. Fogel. *Intelligence through simulated evolution, forty years of evolutionnary programming.* Wiley, 1999.

|FOW66| L. Fogel, A. Owens, and M. Walsh. *Artificial intelligence through simulated evolution.* Wiley, 1966.

|FPSSU96| U. Fayyad, G. Piatetsky-Shapiro, G. Smyth, and R. Uthurasamy. *Advances in knowledge discovery and data mining.* MIT Press, 1996.

|Fre99| Y. Freund. An adaptive version of the boost by majority algorithm. In *Proc. of the 12th Annual Conf. on Computational Learning Theory, COLT'99*, pages 102–113. Morgan Kaufmann, 1999.

|Fri87| J. Friedman. Exploratory projection pursuit. *J. of the American Statistical Association*, 82(347) :249–266, 1987.

|Fri97| N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Fourteenth Inter. Conf. on Machine Learning (ICML97)*, Nashville, Tenessee, 1997.

|FS97| Y. Freund and R. Shapire. A decicion-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1) :325–332, 1997.

|FS99| Y. Freund and R. Shapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5) :771–780, 1999.

|FSST97| Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning Journal*, 28 :133–168, 1997. Apparemment (d'après mon survol) très important. 12-03-02.

|FT74| J. Friedman and J. Tukey. A projection pursuit algorithm for explanatory data analysis. *IEEE Trans. on Computers*, 23(9) :881–890, 1974.

|FT98| G. Fauconnier and M. Turner. Conceptual integration networks. *Cognitive Science*, 22(2) :133–187, 1998.

Copyright © 2010 Eyrolles.

|Fu74| K. Fu. *Syntactic Methods in Pattern Recognition.* Academic Press, 1974.

|Gan93| J-G. Ganascia. Tdis : an algebraic formalization. In *International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1008–1013, 1993.

|Gar88| E. Gardner. The space of interactions in neural network models. *Journal of Physics*, A21 :257–270, 1988.

|GBD92| S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1) :1–58, 1992.

|GBMO06| Eva Gómez-Ballester, Luisa Micó, and José Oncina. Some approaches to improve tree-based nearest neighbour search algorithms. *Pattern Recognition*, 39,1 :171–179, 2006.

|GE03| I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3 :1157–1182, 2003.

|GHK01| D. Gentner, K. J. Holyoak, and B. Kokinov. *The analogical mind : Perspectives from cognitive science.* MIT Press, 2001.

|Gin38| C.W. Gini. Variabilita e mutabilita, contributo allo studio delle distribuzioni e relazioni statische. Technical report, Studi Economici-Giuridici, Universita di Cagliari, 1938.

|Gir00| G. Giraud, editor. *La théorie des jeux.* Flammarion. Champs Université, 2000.

|GJP95| F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7 :219–269, 1995.

|GLZ08| A. Goldberg, M. Li, and X. Zhu. Online manifold regularization : A new learning setting and empirical study. In *ECML-PKDD-2008*, volume 1, page 393406, Antwerp, Belgium, 2008. Springer.

|GN88| M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, 1988.

|GNOT92| D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12) :61–70, 1992.

|Gol67| E. Gold. Language identification in the limit. *Information and Control*, 10 :447–474, 1967.

|Gol78| M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37 :302–320, 1978.

|Gol96| R. Golden. *Mathematical methods for neural network analysis and design.* MIT Press, 1996.

|Gon85| T. Gonzales. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38 :293–306, 1985.

|Gre94| J. Gregor. Data-driven inductive inference of finite-state automata. *IJPRAI*, 8,1 :305–322, 1994.

|GS00| A. Giordana and L. Saitta. Phase transitions in relational learning. *Machine Learning journal*, 41 :217–251, 2000.

|GS07| S. Gelly and D. Silver. Combining online and offline knowledge in uct. In *24th international conference on Machine learning*, pages 273–280, Corvalis, Oregon, 2007. ACM.

Copyright © 2010 Eyrolles.

|GSC10| A. Giordana, L. Saitta, and A. Cornuéjols. *Phase Transitions in Machine Learning.* Cambridge University Press, 2010.

|GT07| L. Getoor and B. Taskar, editors. *An introduction to statistical relational learning.* MIT Press, 2007.

|GU00| G. Grudic and L. Ungar. Localizing policy gradient estimates to action transitions. In *17th International Conference on Machine Learning (ICML'00)*, pages 343–350. Morgan Kaufmann, 2000.

|Hal00| M. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Seventh International Conf. on Machine Learning (ICML-00)*, pages 359–366, 2000.

|Hau88| D. Haussler. Quantifying inductive bias : AI learning algorithms and Valiant learning framework. *Artificial Intelligence Journal*, 36 :177–221, 1988.

|Hau92| D. Haussler. Decision theoretic generalization of the PAC model for neural net and other learning applications. *Information and Computation*, 100 :78–150, 1992.

|Hau99| D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science dept., Univ. of California at Santa-Cruz, 1999.

|Hay99| S. Haykin. *Neural networks : A comprehensive foundation.* Prentice Hall, 1999.

|Hay08| S. Haykin. *Neural Networks and Machine Learning.* Prentice Hall, 2008.

|HB88| F. Van Harmelen and A. Bundy. Explanation-based generalization = partial evaluation. *Artificial Intelligence journal*, 36 :401–412, 1988.

|HCMF08| T. Hesterberg, N. H. Choi, L. Meier, and C. Fraley. Least angle and l1 penalized regression : A review. *Statistical Survey Volume*, 2 :61–93, 2008.

|Her02| R. Herbrich. *Learning kernel classifiers. Theory and algorithms.* MIT Press, 2002.

|Hes65| M. Hesse. Aristotle's logic of analogy. *The Philosophical Quarterly*, 15(61) :328–340, 1965.

|HGO00| R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In Alexander Smola, Peter Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in large margin classifiers*, pages 115–132. MIT Press, 200.

|Hil92| W. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In *Artificial Life II*, pages 313–324. Addison-Wesley, 1992.

|Hin89| G. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40, 1989.

|Hir90| H. Hirsh. *Incremental Version-Space merging : a general framework for concept learning.* Kluwer, 1990.

|Hir92| H. Hirsh. Polynomial-time learning with version spaces. In *National Conference on Artificial Intelligence*, pages 117–122, 1992.

|HK01| J. Han and M. Kamber. *Data Mining : Concepts and Techniques.* Morgan-Kaufmann, 2001.

|HK06| J. Han and M. Kamber. *Data Mining : Concepts and Techniques (2nd Edition.* Morgan-Kaufmann, 2006.

|HKO01a| Hyvarinen, Karhunen, and Oja. *Independent component analysis (Adaptive and learning systems for signal processing, communications and control series).* John Wiley and Sons, 2001.

Copyright © 2010 Eyrolles.

|HKO01b| A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley and Sons, 2001.

|HKP91| J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

|HKR02| J. Herlocker, J. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms". *Information Retrieval*, 5 :287–310, 2002.

|HKS94| D. Haussler, M. Kearns, and R. Shapire. Bounds on the sample complexity of bayesian learning using information theory and the VC dimension. *Machine Learning Journal*, 14 :83–113, 1994.

|HKST96| D. Haussler, M. Kearns, H.S. Seung, and N. Tishby. Rigorous learning curve bounds from statistical mechanics. *Machine Learning Journal*, 25 :195–236, 1996.

|HKTR04| J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. on Information Systems*, 22(1) :5–53, 2004.

|HMS01| D. Hand, H. Mannila, and P. Smyth. *Principles of data mining*. MIT Press, 2001.

|HO00| A. Hyvarinen and E. Oja. Independent component analysis : Algorithms and applications. *Neural Networks*, 13 :411–430, 2000.

|Hof95| D. Hofstadter. *Fluid concepts and creative analogies. Computer models of the fundamental mechanisms of thought*. Basic Books, 1995.

|Hol75| J. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975. Reprinted in 1992 by MIT Press.

|Hop82| J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, 81 :6871–6875, 1982. Owned.

|HOT06| G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18 :1527–1554, 2006.

|HRTZ04| T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5 :1391–1415, 2004.

|HS90| L. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on PAMI*, 12 :993–1001, 1990.

|HS98| V. Honavar and G. Slutski, editors. *Grammatical Inference, Proceedings of ICGI '98*, number 1433 in LNAI, Berlin, Heidelberg, 1998. Springer-Verlag.

|HTF01| T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning. Data mining, inference, and prediction*. Springer Verlag, 2001.

|HTF09| T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

|Hub95| B. Hubbard. *Ondes et ondelettes. La saga d'un outil mathématique*. Pour la Science. Diffusion Belin., 1995.

|HW02| R. Herbrich and R. Williamson. Algorithmic luckiness. *Journal of Machine Learning Research*, 2 :1–37, 2002.

|Hyv99| A. Hyvarinen. Survey on independent component analysis. *Neural Computing Surveys*, 2 :94–128, 1999.

Copyright © 2010 Eyrolles.

|ID80| R. L. Iman and J. M. Davenport. Approximations of the critical region of the friedman statistic. *Communications in Statistics*, pages 571–595, 1980.

|Jac01| Ch. Jacob. *Illustrating evolutionary computation with Mathematica*. Morgan Kaufmann, 2001.

|Jam89| M. Jambu. *Exploration informatique et statistique des données*. Dunod, 1989.

|Jan94| J. Jannink. Cracking and co-evolving randomizers. In *Advances in Genetic Programming*, pages 425–443. MIT press, 1994.

|Jay03| E. Jaynes. *Probability theory : the logic of science*. Cambridge University Press, 2003.

|JCS+03| S. Jouteau, A. Cornuéjols, M. Sebag, P. Tarroux, and J.-S. Liénard. Nouveaux résultats en classification à l'aide d'un codage par motifs fréquents. *Revue d'Intelligence Artificielle (Proc. of the EGC-03 Conf.)*, 17(1-3) :521–532, 2003.

|JCST01| T. Joachims, N. Cristianini, and J. Shawe-Taylor. Composite kernels for hypertext categorization. In *Int. Conf. on Machine Learning(ICML-06)*, pages 250–257, 2001.

|JD88| A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

|Jel76| F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64 :532–556, 1976. No. 4.

|Jel97| F. Jelinek. *Statistical Methods for Speech Recognition*, volume 112 of *Mathematics in Science and Engineering*. The MIT Press, 1997.

|JHYY09| T. Joachims, T. Hofmann, Y. Yue, and C.-N. Yu. Predicting structured objects with Support Vector Machines. *Communications of the ACM*, 52(11) :97–104, 2009.

|JJ00| T. Jebara and T. Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *16th Conference on Uncertainity In Artificial Intelligence*, pages 291–300, 2000.

|JKP94| G. John, R. Kohavi, and K. Pfleger. Irrelevant feature and the subset selection problem. In Haym Hirsh and William Cohen, editors, *Eleventh International Conf. on Machine Learning (ICML-94)*, pages 121–129, Rutgers, NB, 1994. Morgan Kauffmann.

|JL95| L. Jimenez and D. Landgrebe. High dimensional feature reduction via projection pursuit. Technical Report TR-ECE 96-5, Purdue University, School of Electrical and Computer Engineering, 1995.

|JM00| D. Jurafsky and J. Martin. *Speech and language processing*. Prentice-Hall, 2000.

|Joa99| T. Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher Burges, and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press, 1999.

|Jou02| S. Jouteau. *Reconnaissance de scènes naturelles*. Master's thesis, Paris-6, 2002.

|JP98| H. Juillé and J.B. Pollack. A stochastic search approach to grammar induction. In *Grammatical Inference*, number 1433 in Lecture Notes in Artificial Intelligence, pages 126–137. Springer-Verlag, 1998.

|JS02| N. Japkowicz and S. Stephen. The class imbalance problem : A systematic study. *Intelligent Data Analysis Journal*, 6(5), 2002.

Copyright © 2010 Eyrolles.

|JW98| R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis.* Prentice Hall, 1998.

|KA99| J. Koza and D. Andre. *Genetic programming III : Darwinian invention and problem solving.* MIT Press, 1999.

|Kal99| L. Kallel. *Convergence des algorithmes génétiques : aspects saptiaux et temporels.* PhD thesis, École Polytechnique, France, 1999.

|KCC87| S. Kedar-Cabelli and T. Mc Carty. Explanation-based generalization as resolution theorem proving. In *Proc. of the 4th Int. Worshop on Machine Learning, IWML-87*, pages 383–389. Morgan Kaufmann, 1987.

|KHM98| M. Kubat, R. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30 :195–215, 1998.

|KIKS09| H. Kashima, T. Idé, T. Kato, and M. Sugiyama. Recent advances and trends in large-scale kernel methods. *IEICE Transactions on Information and Systems*, E92-D(7) :1338–1353, 2009.

|KJ97| R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence journal*, pages 273–324, 1997.

|KL51| S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22 :79–86, 1951.

|KL02| Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *In Proceedings of the ICML*, pages 315–322, 2002.

|KLM96| L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning : A survey. *Journal of Artificial Intelligence Research*, 4 :237–285, 1996.

|KMC00| K. Kerr, M. Martin, and G. Churchill. Analysis of variance for gene expression microarray data. *J. of Comp. Biol.*, 7(6) :818–837, 2000.

|KMNR95| M. Kearns, Y. Mansour, A. Ng, and D. Ron. An experimental and theoretical comparison of model selection methods. In *Proceedings of the 8th Annual ACM Workshop on Computational Learning Theory.*, pages 21–30. Morgan Kaufmann, 1995.

|KMT93| D. Kulkarni, S. Mitter, and J. Tsitsiklis. Active learning using arbitrary binary valued queries. *Machine Learning*, 11 :23–35, 1993.

|Kol93| J. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Publishers, 1993.

|Koz92| J. Koza. *Genetic programming : on the programming of computers by mecanism of natural selection.* MIT Press, 1992.

|Koz94| J. Koza. *Genetic programming II : Automatic discovery of reusable programs.* MIT Press, 1994.

|KR92| K. Kira and L. Rendell. A practical approach to feature selection. In *Int. Conf. on Machine Learning (ICML-92)*, pages 249–256. Morgan Kaufmann, 1992.

|KS96| D. Koller and M. Sahami. Toward optimal feature selection. In *International Conference on Machine Learning (ICML'96)*, pages 284–292, Bari, Italy, 1996. Morgan Kaufmann.

|KS06| L. Kocsis and C. Szepesvari. Bandit based monte-carlo planning. In *European Conference on Machine Learning*, pages 282–293, 2006.

Copyright © 2010 Eyrolles.

|KTI03| H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Int. Conf. on Machine Learning (ICML-03)*, pages 321–328, Washington, USA, 2003.

|KV89| M. Kearns and G. Valiant. Cryptocraphic limitations on learning boolean formulae and finite automata. In *Proceedings of the 21st annual ACM symposium on Theory of computating*, pages 433–444, New York, 1989.

|KV94| M. Kearns and U. Vazirani. *An introduction to computational learning theory.* MIT Press, 1994.

|KW70| G. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41 :495–502, 1970.

|KW92| J. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. In S. Muggleton, editor, *Inductive Logic Programming.* Academic Press, 1992.

|KWJ+04| R. King, K. Whelan, F. Jones, P. Reiser, C. Bryant, S. Muggleton, D. Kell, and S. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971) :247–252, 2004.

|Lan96| P. Langley. *Elements of Machine Learning.* Morgan Kaufmann, 1996.

|LCB06| Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Svm et apprentissage des très grandes bases de données. *CAp – Conférence d'apprentissage*, 2006.

|Leb95| L. Lebart. *Statistique exploratoire multidimensionnelle.* Dunod, 1995.

|Lee96| L. Lee. Learning of context-free grammars : a survey of the literature. Technical report, Center for Research in Computing Technology, Harvard University, Cambridge, Massachussets. TR 12-96, 1996.

|Len78| D. Lenat. The ubiquity of discovery. *Artificial Intelligence journal*, 9 :257–285, 1978.

|LEPAGE03| Y. LEPAGE. *De l'analogie rendant compte de la commutation en linguistique.* Grenoble, 2003. Habilitation à diriger les recherches.

|Ler81| I-C. Lerman. *Classification et analyse ordinale des données.* Dunod, 1981.

|LFL98| T. Landauer, P. Foltz, and D. Laham. Introduction to latent semantic analysis. *Discourse Porcesses*, 25 :259–284, 1998.

|LG94| D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *International ACM-SIGIR Conf. on Research and Development in Information Retrieval*, pages 3–12, 1994. Cité dans [Roy and McCallum, ICML-01].

|Lin91| L. Lin. Programming robots using reinforcement learning and teaching. In *Proceeding of the Ninth National Conference on Artificial Intelligence (AAAI'91)*, 1991.

|Lit88| N. Littlestone. Learning quickly when irrelevant attributes abound : A new linear-threshold algorithm. *Machine Learning*, 2(4) :285–318, 1988.

|LJ09| M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3 :127–149, 2009.

|Loo06| G. Loosli. *Méthodes à noyaux pour l'apprentissage de contexte. Vers un fonctionnement autonome des méthodes à noyaux.* PhD thesis, INSA, Rouen, 2006.

|LP97| K. Lang and B. Pearlmutter. Abbadingo one competition, 1997. http ://abbadingo.cs.unm.edu/.

Copyright © 2010 Eyrolles.

|LPP98| K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. *Lecture Notes in Computer Science*, 1433 :1–12, 1998.

|LR85| T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6 :4–22, 1985.

|LRN86| J. Laird, P. Rosenbloom, and A. Newell. Chunking in Soar : the anatomy of a general learning mechanism. *Machine Learning journal*, 1 :11–46, 1986. Reprinted in Buchanan, B and Wilkins, D. (eds.), Readings in knowledge acquisition and learning, pp.518-535, Morgan Kaufmann, 1993.

|LRS83| S. Levinson, R. Rabiner, and M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *Bell system Technical Journal*, 62 :1035–1074, 1983.

|LS98| M. Liquière and J. Sallantin. Structural machine learning with galois lattice and graphs. In J. Shavlik, editor, *15th International Conference on Machine Learning (ICML-98)*, pages 305–313. Morgan Kaufmann, 1998.

|LS99| D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401 :788–791, 1999.

|LS00a| P. Langley and S. Stromsten. Learning context-free grammars with a simplicity bias. In *Proceedings of ECML 2000, 11th European Conference on Machine Learning,,* volume 1810 of *LNCS*, pages 220–228. Springer-Verlag, 2000.

|LS00b| D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS-2000*, 2000.

|LT04| S. Lallich and O. Teytaud. Évaluation et validation de l'intérêt des règles d'association. *Revue des Nouvelles Technologies de l'Information (RNTI)*, (Actes EGC-2003) :193–218, 2004.

|LTP07| S. Lallich, O. Teytaud, and E. Prudhomme. Association rules interestingness : measure and validation. In Guillet F. J. and Hamilton H., editors, *Quality Measures in Data Mining*, volume 43 of *Studies in Computational Intelligence*, pages 251–275. Springer-Verlag, 2007.

|LTS89| E. Levin, N. Tishby, and S. Solla. A statistical approach to learning and generalization in layered neural networks. In *Workshop on Computational Learning Theory (COLT'89)*, pages 245–260. Morgan Kaufmann, 1989.

|LV97| M. Li and P. Vitânyi. *Introduction to Kolmogorov complexity and its applications (2nd edition)*. Springer-Verlag, 1997.

|LV07| J. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer, 2007.

|LVA+94| S. Lucas, E. Vidal, A. Amiri, S. Hanlon, and J.C. Amengual. A comparison of syntaxic and statistical techniques for off-line ocr. *ICGI-94*, Lecture Notes in Artificial Intelligence, No 862 :168–180, 1994.

|LW86| N. Littlestone and M. Warmuth. Relating data compression and learnability. Technical report, University of California, Santa Cruz, 1986.

|LW94| N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108 :212–261, 1994.

|LZ96| G. Lugosi and K. Zeger. Concept learning using complexity regularization. *IEEE Transactions on Information Theory*, 42 :48–54, 1996.

Copyright © 2010 Eyrolles.

|M.99| Jordan M., editor. *Learning in graphical models.* MIT Press, 1999.

|MA95| A. Moore and C. Atkinson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning Journal*, 21 :199–233, 1995.

|MAA05| A. Metwally, D. Agrawal, and A. El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *ICDT-2005*, pages 398–412, 2005.

|Mag07| C. Magnan. *Apprentissage à partir de données diversement étiquetées pour l'étude du rôle de l'environnement local dans les interactions entre acides aminés.* Thèse de doctorat, Université de Provence - Aix-Marseille 1, 2007.

|Mat94| M. Mataric. Rewards functions for accelarating learning. In *Proc. of the 11th International Conference on Machine Learning, IWML-91.* Morgan Kaufmann, 1994.

|MB88a| G. McLachlan and K. Basford. *Mixture models : Inference and application to clustering.* Marcel Dekker, 1988.

|MB88b| S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In Kaufmann, editor, *Proceedings of the Fifth International Conference on Machine Learning*, pages 339–352, 1988.

|MB96| S. Matwin and F. Bergadano. Inductive Logic Programming. In *proceedings of ECAI'96,* Budapest, Hungary, 1996.

|MBB+92| O. Matan, H. Baird, J. Bromley, C. Burges, J. Denker, L. Jackel, Y. Le Cun, E. Pednault, W. Satterfield, C. Stenard, and T. Thompson. Reading handwritten digits : A zip code recognition system. *Computer,* 25(7) :59–63, 1992.

|MBD08| L. Miclet, S. Bayoudh, and A. Delhay. Analogical dissimilarity : Definition, algorithms and two experiments in machine learning. *Journal of Artificial Intelligence Research.,* 32 :793–824, 2008.

|MC68| D. Michie and R. Chambers. *An experiment in adaptive control,* volume 2, pages 137–152. Edingburgh University Press, 1968.

|MC91| S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. In *Proc. of the 9th National Conference on Artificial Intelligence, IWML-91*, 1991.

|MC95| R. Mooney and M. Califf. Induction of first-order decision lists : Results on learning the paste tense of english verbs. *Journal of Artificial Intelligence Research,* 3 :1–24, 1995.

|MD94| S. Muggleton and L. De Raedt. Inductive Logic Programming : Theory and Methods. *Journal of Logic Programming*, 19-20 :629–679, 1994.

|MdG94| L. Miclet and C. de Gentile. Inférence grammaticale à partir d'exemples et de contre-exemples : deux algorithmes optimaux : (BIG et RIG) et une version heuristique (BRIG). *Journées Acquisition, Validation, Apprentissage,* pages F1–F13, 1994. Strasbourg France.

|Mdlh96| L. Miclet and C. de la Higuera, editors. *Proceedings of ICGI '96,* number 1147 in LNAI, Berlin, Heidelberg, 1996. Springer-Verlag.

|Mei97| R. Meir. Performance bounds for nonlinear time series. In *Proceedings 10th Annual ACM Workshop on Computational Learning Theory,* pages 122–129, 1997.

Copyright © 2010 Eyrolles.

|Men03| S. Mendelson. A few notes on statistical learning theory. In Shahar Mendelson and Alexander Smola., editors, *Advanced lectures on machine learning*, volume LNAI-2600, pages 1–40. Springer, 2003.

|MF90| S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, Tokyo, 1990.

|MF98| N. Merhav and M. Feder. Universal prediction. In *Information theory. 50 years of discovery.*, pages 80–103. IEEE Press, 1998.

|Mic80| L. Miclet. Regular inference with a tail-clustering method. *IEEE Transactions on SMC*, SMC-10 :737–743, 1980.

|Mic90| L. Miclet. Grammatical inference. In H. Bunke and A. Sanfeliu, editors, *Syntactic and structural pattern recognition theory and applications*. World Scientific, 1990.

|Min61| M. Minsky. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49 :406–450, 1961.

|Min88| S. Minton. *Learning search control knowledge : An explanation-based approach.* Kluwer Academic Publishers, 1988.

|Min89| J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4 :227–243, 1989.

|Min90| S. Minton. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence journal*, 42 :363–392, 1990. Reprinted in Shavlik, J. and Dietterich, T. (eds.), Readings in Machine Learning, Morgan Kaufmann, 1990, pp.96-107.

|Mit82| T. Mitchell. Generalization as search. *Artificial Intelligence*, 18 :203–226, 1982.

|Mit96| M. Mitchell. *An introduction to genetic algorithms.* MIT Press, 1996.

|Mit97| T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

|MKKC86| T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization : a unifying view. *Machine Learning journal*, 1 :45–80, 1986.

|MKS94| S. Murphy, S. Kasif, and S. Saltzberg. A system for inducing oblique decision trees. *Journal of Artificial Intelligence Research*, 2 :1–32, 1994.

|MM96| D. Modha and E. Masrym. Minimum complexity regression estimation with weakly dependant observations. *IEEE Transactions on Information Theory*, 1996.

|MM99| R. Munos and A. Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *16th international Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 1348–1355. Morgan Kaufmann, 1999.

|MM02| G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. of the 28th VLDB Conference*, 2002.

|MO98| Luisa Micó and José Oncina. Comparison of fast nearest neighbour classifiers for handwritten character recognition. *Pattern Recognition Letters*, 19(3-4) :351–356, 1998.

|Mod93| M. Modrzejewski. Feature selection using rough sets theory. In *Proc. of the European Conference on Machine Learning (ECML-93)*, pages 213–226. Springer Verlag (LNAI 667, 1993.

|Moo96| R. Mooney. Inductive Logic Programming for Natural Language Processing. In *Proceedings of the Sixth International Workshop on Inductive Logic Programming*, pages 205–224, Stockholm, Sweden, August 1996.

Copyright © 2010 Eyrolles.

|MP43| W. McCulloch and W. Pitt. A logical calculus of ideas imminent in nervous activity. *Bulletin of Mathemetical Biophysics*, 5, 1943.

|MP69| M. Minsky and S. Papert. *Perceptrons*. MIT Press, 1969.

|MP03| S. Mendelson and P. Philips. Random subclass bounds. In Bernhard Schölkopf and Manfred Warmuth, editors, *Learning theory and kernel machines (COLT/Kernel 2003)*, pages 329–343. Springer, 2003.

|MR03| R. Meir and G. Rätsch. *An Introduction to Boosting and Leveraging*, pages 119–184. Number LNAI 2600. Springer, January 2003.

|MS| S. Mendelson and A. Smola, editors. *Advanced Lectures on Machine Learning (Machine Learning Summer School 2002)*, volume LNAI-2600. Springer-Verlag.

|MS83| R. Michalski and R. Stepp. Automated construction of classifications : conceptuel clustering versus numerical taxonomy. *IEEE Transactions on PAMI*, 5 :396–409, 1983.

|MS01| J.-F. Mari and R. Schott. *Probabilistic and Statistical Methods in Computer Science*. Kluwer, 2001.

|MTV94| H. Mannila, H. Toivonen, and A. Verkamo. Efficient algorithms for discovering association rules. In *Knowledge Discovery in Databases (KDD-94)*, pages 181–192, Menlo Park, CA, 1994. AAAI Press.

|Mug87| S. Muggleton. Duce, an oracle based approach to constructive induction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 287–292. Morgan Kaufmann, 1987.

|Mug95| S. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13(3-4) :243–286. 1995.

|MV95| L. Martin and C. Vrain. MULT_ICN : an empirical multiple predicate learner. In *Proceedings of International Workshop on Inductive Logic Programming*, 1995.

|Nea04| R. Neapolitan. *Learning bayesian networks*. Prentice Hall, 2004.

|New90| A. Newell. *Unified theories of cognition*. Harvard University Press, 1990.

|Ng98| A. Ng. On feature selection : Learning with exponentially many irrelevant features as training examples. In *Int. Conf. on Machine Learning (ICML-98)*, 1998.

|NGSA04| S. Nath, P. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. *ACM Sensys*, 2004.

|Nic93| J. Nicolas, editor. *Support de cours de l'Ecole CNRS sur l'Apprentissage Automatique*, 1993. St Raphaël, 26-27 Mars 1993. Contributions de : O. Gascuel, L. Miclet, J. Nicolas, J. Quinqueton, C. Rouveirol et C. Vrain. Disponible à l'IRISA auprès de J. Nicolas.

|Nil98| N. Nilsson. *Artificial Intelligence : a new synthesis*. Morgan-Kaufmann, 1998.

|NMWM94| C. Nevill-Manning, I. Witten, and D. Maulsby. Compression by induction of hierarchical grammars. In *Data Compression Conference (DCC '94)*, 1994.

|NN97| P. Njiwoua and E. Mephu Nguifo. Iglue : An instance-based learning system over lattice theory. In *Proceedings of the ICTAI'97*, pages 75–76, 1997.

|NS93| A. Nerode and R. Shore. *Logic for Applications*. Springer-Verlag, 1993.

|NWL+07| P. Naïm, P.-H. Wuillemin, P. Leray, O. Pourret, and A. Becker. *Réseaux Bayésiens (3ème éd.)*. Eyrolles, 2007.

Copyright © 2010 Eyrolles.

|OG92a| J. Oncina and P. García. Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis*, pages 49–61, 1992.

|OG92b| J. Oncina and P. García. Inferring regular languages in polynomial updated time. In *Pattern Recognition and Image Analysis : Selected papers from the IVth Spanish Symposium*, pages 49–61. World Scientific, 1992.

|OH91| M. Opper and D. Haussler. Calculation of the learning curve of bayes optimal classification algorithm for learning a perceptron with noise. In *Workshop on Computational Learning Theory (COLT'91)*, pages 75–87. Morgan Kaufmann, 1991.

|Orp92| P. Orponen. Neural networks and complexity theory. Technical report, Department of Computer Science, University of Helsinki., Helsinki, Finland, 1992.

|OSW86| D. Osherson, M. Stob, and S. Weinstein. *Systems that learn.* MIT Press, 1986.

|Par62| E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33(3), 1962.

|Pea78| J. Pearl. On the connection between the complexity and credibility of inferred models. *Int. J. Gen. Syst.*, 4 :255–264, 1978.

|Pea88| J. Pearl. *Probabilistic reasoning in intelligent systems.* Morgan-Kaufmann, 1988.

|Pea09| J. Pearl. *Causality. Models, reasoning and inference (2nd Ed.).* Cambridge University Press, 2009.

|PGV89| J. Pearl, D. Geiger, and T. Verma. The logic of influence diagrams. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams, Belief Nets and Decision Analysis*, pages 67–87. John Wiley and Sons, Ltd., 1989.

|PH97| R. Parekh and V. Honavar. Learning DFA from simple examples. In *Workshop on Automata Induction, Grammatical Inference, and Language Acquisition, ICML-97*, 1997.

|Pin97| S. Pinker. *How the mind works.* Penguin Books, 1997.

|Pit89| L. Pitt. Inductive Inference, DFA's and Computational Complexity. In K. P. Jantke, editor, *Workshop on analogical and inductive inference, AII89*, number 397 in Lecture Notes in Artificial Intelligence, pages 18–44. Springer-Verlag, 1989.

|PK92| M. Pazzani and D. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9(1) :56–94, 1992.

|Pla99| J. Platt. Fast training of support vector machines using sequential minimaloptimisation. In Bernhard Schölkopf, Christopher Burges, and Alexander Smola, editors, *Advance in Kernel Methods. Support Vector Learning*, pages 185–208. MIT Press, 1999.

|Plo70| G. Plotkin. A note on induction generalization. *Machine Intelligence*, 5, 1970.

|Plo71a| G. Plotkin. *Automatic methods of inductive inference.* PhD thesis, Edinburgh University, 1971.

|Plo71b| G. Plotkin. A further note on induction generalization. *Machine Intelligence*, 6 :101–124, 1971.

|PLT04| E. Prudhomme, S. Lallich, and O. Teytaud. Contrôle du risque multiple en sélection de règles d'association. In *EGC-2004*, 2004.

|PMB09| H. Paugam-Moisy and S. Bohte. Computing with spiking neuron networks. In J. Kok and T. Heskes, editors, *Handbook of Natural Computing*, pages xxx–xxx+40. Springer Verlag, 2009.

Copyright © 2010 Eyrolles.

|PN98| P. Pudil and J. Novovicová. Novel methods for subset selection with respect to problem knowledge. *IEEE Intelligent Systems*, 13(2) :66–74, 1998.

|PPK+04| G. Petasis, G. Paliouras, V. Karkaletsis, C. Halatsis, and C. Spyropoulos. E-grids : Computationally efficient grammatical inference from positive examples. *Grammars*, 7 :69–110, 2004.

|Pro97| J. Proust. *Comment l'esprit vient aux bêtes. Essai sur la représentation.* Gallimard. NRF Essais, 1997.

|PS04| Georgios Paliouras and Yasubumi Sakakibara, editors. *Grammatical Inference : Algorithms and Applications, 7th International Colloquium, ICGI 2004, Athens, Greece, October 11-13, 2004, Proceedings*, volume 3264 of *Lecture Notes in Computer Science*. Springer, 2004.

|PSF91| G. Piatetsky-Shapiro and W. Frawley. *Knowledge Discovery in Databases.* AAAI/MIT Press, 1991.

|PT95| A. Pelissier and A. Tête. *Sciences cognitives. textes fondateurs (1943-1950).* PUF, 1995.

|PV88| L. Pitt and L. Valiant. Computational limitations on learning from examples. *Communications of the ACM*, 35(4) :965–984, 1988.

|PW93| L. Pitt and M. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *JACM*, 40(1) :95–142, 1993.

|PY08| S. J. Pan and Q. Yang. A survey on transfer learning. Technical Report HKUST-CS08-08, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China, 2009 2008.

|QC95| J. Quinlan and R. Cameron-Jones. Induction of logic programs : Foil and related systems. *New Generation Computing*, 13(3-4) :287–312, 1995.

|QCSSL09| J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, editors. *Dataset shift in machine learning.* MIT Press, 2009.

|QR89| J. Quinlan and R. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80 :227–248, 1989.

|Qui90| J. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5 :239–266, 1990.

|Qui93| J. Quinlan. C4.5 *: Programs for Machine Learning.* Morgan Kaufmann, 1993.

|Rab89| L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 37, No 2, 1989.

|Rac08| L. De Raedt. *Logical and relational learning.* Springer, 2008.

|Ran00| J. Randlov. Shaping in reinforcement learning by changing the physics of the problem. In *17th International Conference on Machine Learning (ICML'2000)*, pages 767–774. Morgan Kaufmann, 2000.

|Rey00| S. Reynolds. Adaptive resolution model-free reinforcement learning : decision boundary partitioning. In *17th International Conference on Machine Learning (ICML'2000)*, pages 783–790. Morgan Kaufmann, 2000.

|RFKT08| L. De Raedt, P. Frasconi, K. Kersting, and H. Toivonen, editors. *Probabilistic inductive logic programming - Theory and applications.* LNAI-4911. Springer, 2008.

Copyright © 2010 Eyrolles.

|RHW86| D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing : Explorations in the Microstructure of Cognition*, volume 1 : Foundations. Bradford Book, MIT Press, 1986.

|Rip96| B. Ripley. *Pattern recognition and neural networks*. Cambridge University Press, 1996.

|Ris78| J. Rissanen. Modeling by the shortest data description. *Automatica*, 14 :465–471, 1978.

|RJ93| L. Rabiner and F. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.

|RLPB08| H. R. Landau, M. J. Paez, and C. Bordeianu. *A survey of computational physics. Introductory computational science*. Princeton University Press, 2008.

|RM01| N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Int. Conf. on Machine Learning (ICML-01)*, pages 441–448, 2001.

|RN06| S. Russell and P. Norvig. *Intelligence Artificielle*. Pearson Education, 2006.

|RN10| S. Russel and P. Norvig. *Artificial Intelligence : a modern approach (3rd Edition)*. Prentice-Hall, 2010.

|Rob65| J. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1) :23–41, 1965.

|Rob77| R. W. Robinson. Counting unlabeled acyclic digraphs. In C.H.C. Little, editor, *Combinatorial Mathematics*, volume 622 of *Lectures Notes in Mathematics*. Springer-Verlag, 1977.

|Rob97| S. Roberts. *An Introduction to Progol*. Oxford University, January 1997.

|Ron96| A. Ronge. Genetic programs and co-evolution. Technical report, Department of Numerical Analysis and Computer Science, Stockholm University, 1996.

|Ros62| F. Rosenblatt. *Principles of neurodynamics : Perceptrons and the theory of brain mechanisms*. Spartan. Washington, DC, 1962.

|Ros97| R. Rosenfeld. The EM algorithm. Technical report, Computer Science Department, Carnegie-Mellon University, Pittsburg, February 1997.

|RPV89| H. Rulot, N. Prieto, and E. Vidal. Learning accurate finite-state structural models of words : the ECGI algorithm. In *ICASSP'89*, volume 1, pages 643–646, 1989.

|RS89| C. Riesbeck and R. Schank. *Inside Case-based Reasoning*. Lawrence Erlbaum Associates, 1989.

|RSK03| M. Robnik-Sikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning Journal*, 53 :23, 2003.

|Rud97| G. Rudolph. *Convergence properties of evolutionary algorithms*. PhD thesis, Ph.D. thesis, Universität, Hamburg, 1997.

|RV88| H. Rulot and E. Vidal. An efficient algorithm for the inference of circuit-free automata. In G. Ferraté, T. Pavlidis, A. Sanfeliu, and H. Bunke, editors, *Advances in Structural and Syntactic Pattern Recognition*, pages 173–184. NATO ASI, Springer-Verlag, 1988.

|RW84| R. Redner and H. Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM Review*, 26 :195–239, 1984.

Copyright © 2010 Eyrolles.

|SA94| S. Schaal and C. Atkeson. Robot juggling : an implementation of memory-based learning. *Control Systems Magazine*, 18, 1994.

|Sak90| Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, 76 :223–242, 1990.

|Sak97| Yasubumi Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185(1) :15–45, 10 October 1997.

|Sak03| J. Sakarovitch. *Éléments de théorie des automates*. Vuibert, 2003.

|Sam59| A. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3 :210–229, 1959.

|Sam93| C. Sammut. The Origine of Inductive Logic Programming : a Prehistoric Tale. In *Proceedings of the 3th International Workshop on Inductive Logic Programming*, pages 127–148, Bled, Slovenia, April 1993.

|Sap90| G. Saporta. *Probabilités, analyse des données et statistiques*. Technip, 1990.

|SB97| S. Singh and D. Bertsekas. Reinforcement learning for dynamical channel allocation in cellular telephone systems. In *Advances in Neural Information Processing Systems : Proceedings of the 1996 Conference*, pages 974–980. MIT Press, 1997.

|SB98| R. Sutton and A. Barto. *Reinforcement Learning : an introduction*. MIT Press, 1998.

|SB08| O. Sigaud and O. Buffet, editors. *Processus décisionnels de Markov en intelligence artificielle (volume 1)*. Hermes-Lavoisier, 2008.

|SBE99| B. Schölkopf, C. Burges, and A. Smola (Eds). *Advances in kernel methods. Support vector learning*. MIT Press, 1999.

|SC08| I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, 2008.

|Sch78| G. Schwartz. Estimating the dimension of a model. *Annals of Statistics*, 6 :461–464, 1978.

|Sch90| R. Schapire. The strength of weak learnability. *Machine Learning journal*, 5(2) :197–227, 1990.

|Sch92| R. Schalkoff. *Pattern Recognition*. Wiley, 1992.

|Sch94| J. Schmidhuber. On learning how to learn lerning strategies. Technical Report FKI-198-94, Fakultat fur Informatik, 1994.

|Sch97| B. Schölkopf. *Support vector learning*. PhD thesis, University of Berlin, 1997. Available from http ://www.kyb.tuebingen.mpg.de/ bs.

|Seb94a| M. Sebag. A constraint-based induction algorithm. In W. Cohen and H. Hirsh, editors, *Proceedings of ICML-94*. Morgan Kaufmann, 1994.

|Seb94b| M. Sebag. Using constraints to build version spaces. In *Proceedings of the 1994 European Conference on Machine Learning*, 1994.

|Sem10| José Sempere, editor. *Grammatical Inference : Algorithms and Applications, 10th International Colloquium, September 13-16, 2010, Valencia, Spain, Proceedings*, volume to be published of *Lecture Notes in Computer Science*. Springer, 2010.

|Set09| B. Settles. Active learning literature survey. Technical Report Computer Sciences Technical Report 1648, University of Wisonsin-Madison, January 9, 2009 2009.

|SF04| Bradford Starkie and Henning Fernau. The boisdale algorithm - an induction method for a subclass of unification grammar from positive data. In Paliouras and Sakakibara |PS04|, pages 235–247.

Copyright © 2010 Eyrolles.

|SFBL98| R. Shapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin : A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5) :1651–1686, 1998.

|SG98| R. Sun and L. Giles, editors. *Sequence learning*. Springer, 1998.

|SGS01| P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search (2nd Ed.)*. MIT Press, 2001.

|Sha50| C. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41 :256–275, 1950.

|Sha83| E. Shapiro. *Algorithmic program debugging*. MIT Press, 1983.

|SHSW01| B. Schölkopf, R. Herbrich, A. Smola, and R. Williamson. A generalized representer theorem. In *COLT-2001*. Springer, 2001.

|Sim81| H. Simon. *The sciences of the artificial*. MIT Press, 1981.

|Sim85| J.-C. Simon. *La reconnaissance des formes par algorithmes*. Masson, 1985.

|Sim94| K. Sims. Evolving 3d morphology and behavior by competition. In *Proc. Artificial Life IV*, pages 28–39. MIT Press, 1994.

|SKKR01| B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. item-based collaborative filtering recommendation algorithms. In *Proceedings of the tenth International Conference on World Wide Web*, pages 202–208, 2001.

|SKM07| M. Sugiyama, M. Kraudelat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8 :985–1005, 2007.

|SKS+06| Yasubumi Sakakibara, Satoshi Kobayashi, Kengo Sato, Tetsuro Nishino, and Etsuji Tomita, editors. *Grammatical Inference : Algorithms and Applications, 8th International Colloquium, ICGI 2006, Tokyo, Japan, September 20-22, Proceedings*, volume 4201 of *Lecture Notes in Computer Science*. Springer, 2006.

|SKS07| R. Sutton, A. Koop, and D. Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th international conference on Machine learning*, pages 871–878, Corvalis, Oregon, 2007. ACM.

|SL74| H. A. Simon and G. Lea. Problem solving and rule induction : A unified view. In L. W. Gregg, editor, *Knowledge and cognition*. Erlbaum, 1974.

|SL91| S. Rasoul Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on SMC*, 21(3), 1991.

|SLD02| M. Saerens, P. Latinne, and C. Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities : A simpleprocedure. *Neural Computation*, 14(1) :21–41, 2002.

|SLJ+03| S. Singh, M. Littman, N. Jong, D. Pardoe, and P. Stone. Learning predictive state representations. In *20th International Conference on Machine Learning (ICML-2003*, pages 712–719, 2003.

|SNK+07| Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Von Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. MIT Press, 2007.

|Sol64| R. Solomonoff. A formal theory of inductive inference, part 1 and part 2. *Information and Control*, 7 :1–22, 224–254, 1964.

Copyright © 2010 Eyrolles.

|SOS92| H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *COLT'92*, pages 287–294, Pittsburgh, PA, USA, 1992. ACM Press.

|SR90| B. Smith and P. Rosenbloom. Incremental non-backtracking focussing : A polynomially-bounded generalization algorithm for version space. In *National Conference on Artificial Intelligence (AAAI'90)*, pages 848–853, 1990.

|SRSS06| S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7 :1531–1565, 2006.

|SS98a| R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Eleventh Annual Conference On Computationnal Learning Theory (COLT)*, 1998.

|SS98b| A. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22 :211–231, 1998.

|SS02| B. Schölkopf and A. Smola. *Learning with kernels. Support vector machines, regularization, optimization, and beyond.* MIT Press, 2002.

|SST92| S. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Physcal Review (A)*, 45 :6056–6091, 1992.

|SSTSW99| B. Schölkopf, J. Shawe-Taylor, A. Smola, and R. Williamson. Kernel-dependent support vector error bounds. In *Ninth International Conference on Artificial Neural Networks, 1999 (ICANN 99).*, pages 103–108, Edinburgh, UK, 1999.

|Sta96| P. Stadler. Landscape and their correlation functions. *J. of Mathematical Chemistry*, (20) :1–45, 1996.

|STBWA96| J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. A framework for structural risk minimization. *Proceedings of the 9th Annual ACM Workshop on Computational Learning Theory*, pages 68–76, 1996.

|STBWA98| J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44 :1926–1940, 1998.

|STC04| J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis.* Cambridge University Press, 2004.

|Sto77| C. J. Stone. Consistent nonparametric regression. *Ann. Statist.*, 5(4) :595–620, 1977.

|Sto04| J. Stone. *Independent component analysis. A tutorial introduction.* MIT Press, 2004.

|Str00| M. Strens. A bayesian framework for reinforcement learning. In *17th International Conference on Machine Learning (ICML'2000)*, pages 943–950. Morgan Kaufmann, 2000.

|STV04| B. Schölkopf, K. Tsuda, and J.-P. Vert, editors. *Kernels Methods in Computational Biology.* MIT Press, 2004.

|SY06| Nicolas Stroppa and François Yvon. Formal models of analogical proportions. *Technical report 2006D008, École Nationale Supérieure des Télécommunications, Paris, France*, 2006.

|Tak88| S. Takada. Grammatical inference for even linear languages based on control sets. *Information Processing Letter*, 28 :193–199, 1988.

Copyright © 2010 Eyrolles.

|Tau94| B. Tausend. Representing biases for Inductive Logic Programming. In *proceedings of ECML'94*, pages 427–430, 1994.

|TB73| B. Trakhtenbrot and Ya. Barzdin. *Finite Automata : Behavior and Synthesis.* North Holland Pub. Comp., Amsterdam, 1973.

|TD| F. Thollard and P. Dupont. Inférence grammaticale probabiliste utilisant la divergence de kullback-leibler et un principe de minimalité. In *CAP2000, Conférence d'apprentissage*, pages 259–275. Hermès.

|Tes92| G. Tesauro. Practical issues in temporal difference learning. *Machine Learning Journal*, 8 :257–278, 1992.

|Tes94| G. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6 :215–219, 1994.

|Tes95| G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3) :58–67, 1995.

|TK00| S. Tong and D. Koller. Active learning for parameter estimation in bayesian networks. In *Advances in Neural Information Processing Systems (NIPS-00)*, volume 13, pages 647–653, 2000.

|TL01| O. Teytaud and S. Lallich. Bornes uniformes en extraction de règles. In Gilles Bisson, editor, *Conférence d'Apprentissage (CAP-01)*, pages 133–148, Grenoble, 2001. PUG.

|TM95| S. Thrun and T. Mitchell. Learning one more thing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 1217–1223. Morgan Kaufmann, 1995.

|TSM85| D. Titterington, A. Smith, and U. Makov. *Statistical analysis of finite mixture distributions.* John Wiley and Sons, 1985.

|Tur92| A. Turing. Intelligent machinery. In D. Ince, editor, *Collected works of Alan Turing : Mechanical Intelligence.* Elsevier, 1992.

|Utg89| P. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4(2), 1989.

|Val84| L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11) :1134–1142, 1984.

|Vap82| V. Vapnik. *Estimation of dependences based on empirical data.* Springer-Verlag, 1982.

|Vap95| V. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, 1995.

|Vap98| V. Vapnik. *Statistical learning theory.* Wiley-InterScience, 1998.

|VC71| V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2) :264–280, 1971.

|VC91| V. Vapnik and A. Chervonenkis. The necessary and sufficient conditions for the consistency of the method of empirical risk minimization. *Pattern Recognition and Image Analysis*, 1 :284–305, 1991.

|VD02| R. Vilalta and Y. Drissi. A perspective view and a survey of meta-learning. *Artificial Intelligence Review*, 18 :77–95, 2002.

|Vid94a| E. Vidal. Language learning, understanding and translation. *Progress and Prospects of Speech Research and Technology. Proceedings of the CRIM/FORWISS Workshop*, September 1994.

Copyright © 2010 Eyrolles.

|Vid94b| E. Vidal. New formulation and improvements of the nearest-neighbour approximation and elimating search algorithm (asea). *Pattern Recognition Letters*, pages 1–7, January 1994.

|Vid97| M. Vidyasagar. *A theory of learning and generalization.* Springer-Verlag, 1997.

|Vid03| M. Vidyasagar. *Learning and Generalization. With applications to neural networks. (2nd ed.).* Springer-Verlag, 2003.

|Vit85| J. S. Vitter. Random sampling with a reservoir. *ACM Trans. on Mathematical Software,* 11(1) :37–57, 1985.

|vLBS04| U. von Luxburg, O. Bousquet, and B. Schölkopf. A compression approach to support vector model selection. *Journal of Machine Learning Research*, 5 :293–323, 2004.

|VLC94| V. Vapnik, E. Levin, and Y. Le Cun. Measuring the VC-dimension of a learning machine. *Neural Computation*, 6 :851–876, 1994.

|Vos99| M. Vose. *The simple genetic algorithm. Foundations and theory.* MIT Press, 1999.

|Wal77| R. Waldinger. Achieving several goals simultaneously. In *Machine Intelligence 8.* Ellis Horwood Ltd, 1977.

|Wat85| S. Watanabe. *Pattern recognition : human and mechanical.* Wiley, 1985.

|Wat89| C. Watkins. *Learning from delayed rewards.* PhD thesis, King's College, Cambridge, UK, 1989.

|WB85| C. Wallace and D. Boulton. An information measure for classification. *Computing Journal*, 11 :185–195, 1985.

|Web99| A. Webb. *Statistical pattern recognition.* Arnold, 1999.

|Wei99| Y. Weiss. Segmentation using eigenvectors : a unifying view. In *Proceedings IEEE International Conference on Computer Vision*, pages 975–982, 1999.

|Wer84| P. Werbos. *The Roots of Backpropagation.* Wiley, 1984.

|WF99| I. Witten and E. Frank. *Data Mining : Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 1999.

|WFRT00| A. Wilson, A. Fern, S. Ray, and P. Tadepalli. A bayesian framework for reinforcement learning. In *International Conference on Machine Learning (ICML-00)*, pages 1015–1022. ACM, 2000.

|Wid94| B. Widrow. 30 years of adaptive neural networks : Perceptron, madaline and backpropagation. *Proceedings of the IEEE*, 78(9), 1994.

|Wig60| E. Wigner. The unreasonable effectiveness of mathematics in the natural sciences. *Pure Mathematics*, 13(1), 1960.

|Wil92a| R. Wille. Concept lattices and conceptual knowledge systems. *Computers Mathematical application*, 23 :493–515, 1992.

|Wil92b| R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning Journal*, 8 :229–256, 1992.

|Win70| P. Winston. *Learning structural descriptions from examples.* PhD thesis, MIT, 1970. MIT Technical Report AI-TR-231.

|Win75| P. Winston. Learning Structural Descriptions from Examples. In *The psychology of Computer Vision.* McGraw-Hill, 1975.

Copyright © 2010 Eyrolles.

|WK91| S. Weiss and C. Kulikowski. *Computer Systems That Learn.* Morgan Kaufmann, 1991.

|Wol92a| D. Wolpert. On the connection between in-sample testing and generalization error. *Complex Systems,* 6 :47–94, 1992.

|Wol92b| D. Wolpert. Stacked generalization. *Neural Networks,* 5(2) :341–259, 1992.

|Wol95| D. Wolpert, editor. *The mathematics of generalization.* Addison Wesley, 1995.

|Wol97| D. Wolpert. No free lunch theorem for optimization. *IEEE Transactions on evolutionary computation,* 1 (1) :467–82, 1997.

|WRB93| T. Watkin, A. Rau, and M. Biehl. The statistical mechanics of learning a rule. *Review of Modern Physics,* 65 :499–556, 1993.

|WSS01| R. Williamson, A. Smola, and B. Schölkopf. Generalization bounds for regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transactions on Information Theory,* 2001.

|XJK01| E. Xing, M. Jordan, and R. Karp. Feature selection for high-dimensional genomic microarray data. In *Int. Conf. on Machine Learning (ICML-01),* 2001.

|YB98| Y. Yang and A. Barron. An asymptotic property of model selection criteria. *IEEE Transactions on Information Theory,* 1998.

|YL03| L. Yu and H. Liu. Feature selection for high dimensional data : a fast correlation based filter solution. In *12th Int. Conf. on Machine Learning (ICML-03),* pages 856–863, Washington, DC., 2003.

|YL04a| L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research,* 5 :1205–1224, 2004.

|YL04b| L. Yu and H. Liu. Redundancy based feature selection for microarray data. In *Proc. of the 10th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining,* pages 737–742, 2004.

|YLF99| M. Young-Lai and F.Tompa. Stochastic grammatical inference of text database structure. *Machine Learning,* 1999.

|YS08| J. Yaochu and B. Sendhoff. Pareto-based multiobjective machine learning : An overview and case studies. *IEEE Transactions on Systems, Man, and Cybernetics, Part C : Applications and Reviews,* 38(3) :397–415, 2008.

|YvoN99| F. Yvon. Pronouncing unknown words using multi-dimensional analogies. In *Proceeding of the European conference on Speech Application and Technology (Eurospeech),* volume 1, pages 199–202, Budapest, Hungary, 1999.

|ZD95| W. Zhang and T. Dietterich. A reinforcement learning approach to job-shop scheduling. In *International Joint Conference on Artificial Intelligence,* pages 1114–1120. Morgan Kaufmann, 1995.

|Zel95| J. Zelle. *Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers.* PhD thesis, University of Texas, August 1995.

|Zha96| W. Zhang. *Reinforcement learning for job-shop scheduling.* PhD thesis, Ph.D. thesis, Oregon State University. Technical Report CS-96-30-1, 1996.

|Zhu08| X. Zhu. Semi-supervised learning literature survey. Technical Report TR-1530, University of Wisconsin Madison, Computer Science Dept., 2008.

Copyright © 2010 Eyrolles.

|ZLG03| X. Zhu, J. Lafferty, and Z. Ghaharamani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the ICML Workshop on the Continuum from Labeled to Unlabeled Data*, pages 58–65, 2003.

|ZM93| J. Zelle and R. Mooney. Learning Semantic Grammars with Constructive Inductive Logic Programming. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 817–822, Washington, D.C., USA, August 1993.

|ZO00| T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1191–1198. Morgan Kaufmann, 2000.

|ZPD06| T. Zhang, A. Popescul, and B. Dom. Linear prediction models with graph regularization for web-page categorization. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 821–826, Philadelphia, PA, USA, 2006.

|ZR00| D. A. Zighed and R. Rakotomalala. *Graphes d'induction : apprentissage et data mining*. Hermès, 2000.

|ZRZH06| J. Zhu, S. Rosset, H. Zou, and T. Hastie. Multiclass adaboost. `http://www-stat.stanford.edu/~hastie/Papers/samme.pdf`, 2006.

|Zuc01| J.-D. Zucker. Changements de représentation, abstractions et apprentissages. Dossier d'habilitation à diriger des recherches (HDR). Technical report, 2001.

Copyright © 2010 Eyrolles.

Copyright © 2010 Eyrolles.

Index

χ^2 (critère du), 414
θ-subsomption, 168, 170, 175
θ-subsomption relative, 177

absorbtion, 180
abstraction, 208–210
accuracy, 119
action de l'agent, 535
actor-critic
    apprentissage par renforcement, 557
ADABOOST, 584
AIC
    critère de pénalisation, 704
Akaike
    critère d'information, 704
algorithme *cluster and label* pour l'apprentissage semi-supervisé, 643
algorithme AESA, 522
algorithme Apriori, 621
algorithme BRIG, 247
algorithme d'élimination des candidats, 152
algorithme d'apprentissage, 61
algorithme d'auto-apprentissage, 644
algorithme d'extension des étiquettes pour l'apprentissage semi-supervisé, 653
algorithme de Baum-Welsh, 400
algorithme de condensation, 519
algorithme de nettoyage, 519
algorithme de PLI générique, 183
algorithme de propagation des étiquettes pour l'apprentissage semi-supervisé, 652
algorithme de rétropropagation du gradient de l'erreur, 340
algorithme de Viterbi, 397
algorithme ECGI, 243
algorithme EM, 398, 610, 640
    pour l'apprentissage semi-supervisé, 642
algorithme k-RI, 243
algorithme RPNI, 247
algorithme RPNI stochastique, 259
algorithme *forward-backward*, 393
algorithme PROGOL, 189
algorithmes génétiques, 267, 269–280
    élection proportionnelle à la performance, 274
    espace génotypique, 266, 268, 270, 291
    espace phénotypique, 268
    fonction de performance, 274
    opérateurs de croisement, 271, 273
    opérateurs de mutation, 272, 273
    pression sélective, 274
    problèmes trompeurs, 280
    remplacement, 276
    sélection, 273–276
    sélection proportionnelle au rang, 275
    sélection par tournoi, 275
    systèmes de classeurs, 293
algorithmes rapides pour les k-plus proches voisins, 522
all-subsets selection, 320
alphabet auxiliaire d'une grammaire, 226
alphabet terminal d'une grammaire, 226
AM *Automated Mathematician*, 284
analogie (raisonnement par), 210–211
analyse en composantes principales communes, 99
analyse discriminante linéaire, 506
analyse discriminante non linéaire, 506
analyse en composantes indépendantes, 99, 626–630
analyse en composantes principales, 99, 626
analyse en pire cas, 63
analyse en variables latentes, 632
ANOVA, 128
apprentissage
    à partir d'échecs, 205, 206
    à partir d'explications (EBL), 201–208
    à grande échelle, 672
    actif, 16, 725
    auto-apprentissage, 638
    avec bruit, 710

Copyright © 2010 Eyrolles.

batch, 671, 725
co-apprentissage, 644–646
collaboratif, 725
dans les HMM, 398
de classement, 98
de sorties structurées, 456–458, 682–685
en ligne, 15, 60
faible, 582, 590
grande échelle, 719
hors ligne, 15
incrémental, 725
mistake-bound learning model, 676
relative loss bound model, 676
using expert advice, 677
multi-étiquette, 133
multi-instance, 685
multi-objectif, 133
non supervisé, 17
PAC, 62
de grammaires, 236–239
par évolution simulée, 263–295
par différences temporelles, 548–552
par renforcement, 531–567
par requête d'appartenance, 16
par transfert, 674–675
paresseux, 529
petite échelle, 719
semi-supervisé, 635–658
supervisé, 14, 17, 41
de concepts, 22
transductif, 639, 657, 658
apprentissage actif, 662–669
pool-based, 662
stream-based, 662
apprentissage en ligne, 675–677
apprentissage hors ligne (batch), 671
apprentissage hors-ligne (batch), 725
apprentissage par renforcement
actor-critic, 557
bayesien, 557
hors politique, 551
sur politique, 550
arbitrary binary queries, voir protocole de requêtes binaires arbitraires
arbres de décision, 105, 409–423, 528
arbres de décision obliques, 417
arbres de régression, 423–425
astuce de la représentation unique, 145, 167
attribut
arborescent, 140
binaire, 86, 140
catégoriel, 86
nominal, 86, 140
nominal hiérarchique, 87
nominal totalement ordonné, 87
numérique, 140
ordinal, 86
séquenciel nominal, 87
séquenciel numérique, 88
auto-apprentissage, 638
automates à états finis, 281

bagging, 590
bandit à deux bras, 539
bandit multi-bras, 558
bandits (problème des), 539
batch learning, 671, 725
bayésienne (approche), 46, 50
Bayes
classificateur naïf de, 491
formule de, 47, 50, 491
règle de, 45, 53
Bellman
Principe d'optimalité, 537
Bellman (équation de), 544
biais, 67
biais de recherche, 184
biais de représentation, 26
biais sémantique, 185
biais syntaxique, 185
BIC
Bayesian Information Criterion, 705
critère de pénalisation, 705
blending, 211
boosting, 582–590
bootstrap, 118, 585, 681
bruit de classe, 131
bruit de mesure, 131

cadre du regret, 675–677
capacité (d'un espace d'hypothèses), 697
cartes auto-organisatrices de Kohonen, 99, 601–603
cascading, 591–592
causalité (apprentissage de la), 378
chaînes de Markov, 107
chaînes de Markov cachées, voir HMM
chemin de régularisation, 323
classificateur naïf de Bayes, 491
classification supervisée, 41
clause, 172
but, 173
définie, 172
de Horn, 172
unitaire, 173
co-évolution, 291–293
co-apprentissage, 644–646
coûts non symétriques, 131
codes correcteurs, 577–582

Copyright © 2010 Eyrolles.

décodage par marges, 580
décodeur de Hamming, 580
coefficient de pulvérisation, 698
cognitivisme, 12–14
comité d'experts, 583
comparaison
d'algorithmes, 125–129
de méthodes d'apprentissage, 124
complexité algorithmique, voir complexité de Kolmogorov
complexité calculatoire dans l'apprentissage PAC, 724
complexité de Kolmogorov, 758–759
complexité de Rademacher, 699–700
compromis biais-variance, 68
confiance, voir règles d'association
confiance (règle d'association), 623
connexionnisme
reservoir computing, 348–351
consistance universelle, 61
corrélation de Pearson, 687
corrélation de Spearman, 690
couverture d'un motif, 619
couverture de Markov, 95–96
couverture d'une hypothèse, 142
covariate shift, 672–673
critère d'opérationnalité, 202
critère de performance, voir performance (mesure de)
critère inductif, 49–60
croisement, 273
cybernétique, 11–12

d-séparation, 361–363, 368
décision optimale, 45
data streams, 112
data mining , voir fouille de données
data warehouse, 615
deep belief networks, 346–347
déplacement covarié, 672–673
dérive de concept, 60, 674
diagnostic, 367
dilemme exploitation vs. exploration, 272, 539–541
dimension
de Vapnik-Chervonenkis, 699, 749, 751
réduction de, 89–99
dimensionnalité
fléau de, 89, 505, 608
discriminante (approche), 47–49, 493
distance, 597
distributions conjuguées, 54
données bruitées, 131
données déséquilibrées, 129–130
données en grande dimension, 131
données non indépendantes, 132–133
données temporelles, 349

early stopping rule, 710
EBL (*Explanation-Based Learning*), 201–208
échantillon
d'apprentissage, 22, 120
de test, 120
de validation, 120
effet tunnel cognitif, 211
élagage d'un arbre de décision, 418
EM (*Expectation - Maximization*), 736–740
entrepôt de données, 615
entropie (calcul sur les flux), 679
entropie croisée, 95, 496
épistasie, 280
equivalence query, voir protocole de requêtes d'équivalence
erreur
d'approximation, 67, 719
d'estimation, 67
d'optimisation, 719
de biais, 719
de première espèce, 116
de seconde espèce, 116
espace de Hilbert, 437, 438
espace de Hilbert à noyau reproduisant, 439
espace de redescription, 433
espace de représentation, 85–88
espace des versions, 150
espace génotypique, 266, 268, 270, 291
espace phénotypique, 268
estimation de densité, 59
évaluation des performances
courbe ROC, 121
par échantillon de test, 115
par la méthode du *leave-one-out*, 118
par resubstitution, 114
par validation croisée, 116
par *bootstrap*, 118
par *jackknife*, 118
évolution simulée, 267
exemple, 8
explication, 367
exploitation, voir dilemme exploitation vs. exploration, 539
exploration, voir dilemme exploitation vs. exploration, 540
extraction de connaissances des données, xv

F_mesure, 120
factorisation matricielle non-négative (NMF), 630
fat-shattering dimension, 702
fenêtres de Parzen, 512

Copyright © 2010 Eyrolles.

filtrage collaboratif, 686–692
Fisher
information de, 664
fitness function, 268, 274
fléau de la dimensionnalité, 89, 505, 608
flot de données, 112
flux de données
échantillonnage, 678
clustering, 681
min-wise sampling, 679
reservoir sampling, 678
calcul d'entropie, 679
détection de rupture, 680
résumés, 679
flux de données (apprentissage à partir de), 677–679
FOIL, 186–188
fonction cible, 20
fonction de croissance, 749
fonction de performance, 274
fonction de perte, 44
coude, 448
de substitution, 448
fonction de risque, 14
fonction de score, 374
fonction noyau, 511
apprentissage de, 481
fonction sigmoïde, 329
fonctions séparatrices, 103
fouille de données, xv, 132, 615–625
fréquence d'un motif, 619

généralisé maximalement spécifique, 146
généralité (relation de), 142
générative (approche), 47, 493
Gauss-Markov (théorème de), 305
Gini (critère de), 414
gradient stochastique, 671–672
grammaire formelle, 226
grammaire hors-contexte, 227
grammaire k-réversible, 242
grammaire régulière, 227
grande dimension (phénomènes), 90, 608
graphe
laplacien de, 474, 653
G-set, 150

Hadamard Jacques, 18
heuristiques de contrôle, 201, 206
Hilbert
espace de, 437, 438
hinge loss, 448
HMM, 107, 383–405
apprentissage des paramètres d'un, 398
ergodique, 391
gauche-droite, 391
probabilité d'observation, 390
probabilité de transition, 390
probabilités initiales, 390
Hoeffding
inégalité de, 64
Hopfield
réseaux de, 348
hors politique
apprentissage par renforcement, 551
hypothèse
cohérente, 144
complète, 142, 144
correcte, 142, 144
hypothèse nulle, 115
hypothèse optimale, 45

i.i.d. (tirage), 19
identification, 180
inductive logic programming (ILP), voir programmation logique inductive (PLI)
inférence grammaticale, 219–261
information de Fisher, 664
information mutuelle, 374, 628
instance, 85
intelligibilité, 124
inter-construction, 180
intervalle de confiance, 116
intra-construction, 180
invention de prédicat, 180
inversion de la résolution, 179–182
itemsets, 132
itemsets fréquents, 111

jackknife, 118
jeu de Go, 563–564
Johnson-Lindenstrauss
théorème de, 609

k-plus-proches-voisins (règle des), 511, 515
Karush-Kuhn-Tucker (conditions de), 445
KDD (*knowledge discovery in databases*), voir extraction de connaissances des données
kernel methods, 320, 429–486
kernel trick, 434
Kullback-Leibler (divergence de), 95, 680, 704
kurtosis, 627

lagrangien, 444
langage engendré par une grammaire, 226
langage hors-contexte, 227
langage régulier, 227
Langley Pat, 19
laplacien
d'un graphe, 475

Copyright © 2010 Eyrolles.

large-scale learning, 719
large-scale learning, 672
learning to rank, 98
leave-one-out, 118
Levi-Strauss Claude, 12
LEX, 155–156
linéaire par morceaux (modèle), 317
log-vraisemblance, 51
logique de description, 110
logique des prédicats, 110
logique des propositions, 109
loi des grands nombres, 63

macro-opérateurs, 201, 206
MAE (mean average error), 691
marge, 580
marge d'un exemple, 587
marge douce, 448
marge géométrique, 443
Markov
  condition de, 359
  couverture de, 364
  équivalence de, 364
  processus décisionnels de, 538
Markov blanket, 95–96
matrice de codage des classes, 577
matrice de confusion, 115
matrice de Gram, 441
matrice positive semi-définie, 435
maximum a posteriori (MAP), 55
maximum de vraisemblance, 50, 56
Maximum Likelihood Estimator (MLE), 56
MDLP (*Minimum Description Length Principle*), 52, 375, 497, 706–707, 759–761
mélange de gaussiennes, 741
membership query, voir protocole de requêtes d'appartenance
mémoire associative, 348
mesure de dissimilarité, 597
méta-apprentissage, 573
méta-paramètre, 573
méthodes à noyaux, 320, 429–486
méthodes ϵ-gloutonnes, 550
méthodes non paramétriques, 495, 509
méthodes semi-paramétriques, 523
  discrimination logistique, 523
  mélange de distributions, 525
méthodes spectrales, 610–611
min-wise sampling, 679
minimisation du risque empirique, 49, 50
minimisation du risque structurel (*SRM*), 712–713
mixture models, 319, 609–610
modèle
  sélection automatique, 702
modèle de mélange, 319, 609–610
modèle de Markov caché, voir HMM
modèles graphiques, 107
moindres carrés (estimation des), 302
Monte Carlo (méthode de), 547
mot engendré par une grammaire, 226
motif, 619
  couverture de, 619
  fréquence de, 619
  support de, 619
motifs fréquents, 617–622
mots dérivés selon une grammaire, 226
multiplicateurs de Lagrange, 444, 709–710
mutation, 272, 273

néguentropie, 627
neurone formel, 328
no-free-lunch theorem, 74–78, 273
nombre de couverture, 701–702
non-negative matrix factorization (NMF), 630
non-paramétriques (méthodes), 495, 509
Novikoff
  théorème de, 313
noyau
  ANOVA, 463
  arbres, 471
  astuce des, 434
  de convolution, 469
  de Fisher, 477
  diffusion, 474
  gaussien, 464
  génératif, 476–477
  graphes, 471
  reproduisant, 439
  séquences, 468
  spectre, 467
  sur des ensembles, 465
  sur des séquences, 466
  sur des textes, 466
  toutes sous-séquences, 469
  universel, 459
noyaux (méthodes à), 320, 429–486

observation, 14
Occam (algorithme d'), 762–763
Occam (rasoir d'), 706
off-policy
  apprentissage par renforcement, 551
on policy
  apprentissage par renforcement, 550
one-class SVM, 455
ontologie, 106
opérateurs de généralisation, 147–149
opérateurs de spécialisation, 148

Copyright © 2010 Eyrolles.

optimisation de performance, 17
oracle, 14

parcimonieux (codage), 632
Pearson (corrélation de), 687
performance (mesure de), 268
pertinence du *MRE*, 66
physique statistique (analyse de la), 720–723
plan d'expériences, 668
pointwise estimation, 55
politique, 531, 535
politique optimale, 542
POMDP (Partially Observable MDP), 561
pool-based active learning, 662
poursuite de projections, 629
prédiction, 15
prédiction de séquence, 281
prétraitement des données, 88
présentation
  complète, 234
  négative, 234
  positive, 234
précision, 120
pression sélective, 274
principe de longueur minimale de description, 52, 375, 497, 706–707, 759–761
principe de minimisation du risque structurel, 758
problèmes trompeurs, 280
processus décisionnels de Markov, 538
processus de Markov, 386
PRODIGY, 207
PROGOL, 189–190
programmation évolutive, 267
programmation génétique, 267, 281–290
programmation logique inductive (PLI), 161–197
proie-prédateur, 292
projection pursuit, 629
protocole d'apprentissage, 15
protocole de requêtes d'appartenance, 663
protocole de requêtes binaires arbitraires, 663
protocole de requêtes d'équivalence, 663
protocole de requêtes statistiques, 663
pseudo-dimension, 702
puissance du test, 116

Q-learning, 551
query-by-committee, 668

raisonnement à partir de cas (RàPC), 211–212
random forests, 591
ranking, 60, 98
rappel, 120
recall, 120
reconnaissance des formes, xv
redescription, 91
  espace de, 433
reformulation, 209
règle de production d'une grammaire, 226
règles d'association, 110–111, 617, 622–625, 679–680
  confiance, 680
  support, 679
règles de contrôle, 201
régression, 58–59, 301–302
  aux moindres carrés, 495
  de but, 203
régression linéaire, 301–306
régularisation
  exemple, 420
  théorie de la, 708–710
remplacement, 276
renforcement (signal de), 534, 535
représentation duale, 432
représentation par attribut-valeur, 109
representer theorem, 440
réseaux bayésiens, 95, 107, 354–380
  équivalence de Markov, 364
  apprentissage des, 369–377
  condition de Markov, 359
  couverture de Markov, 364
  d-séparation, 361–363, 368
  diagnostic, 367
  explication, 367
  fonction de score, 374
  méthode des arbres de jonction, 368
  méthode EM, 372
  méthodes de conditionnement, 368
  méthodes de Monte Carlo, 368
  raisonnement dans les, 365–368
réseaux connexionnistes, 325–352
  apprentissage, 334–345
  architecture multicouche, 330
  choix de l'architecture, 345
  exemple de fonctionnement, 332
réseaux de neurones
  profonds, 346–347
reservoir sampling, 678
reservoir computing, 348–351
resubstitution (estimation de l'erreur), 114
résumés (calcul sur les flux), 679
rétropropagation du gradient de l'erreur, 340
ridge regression, 320, 321
risque empirique, 50
risque réel, 45
RKHS, voir espace de Hilbert à noyau reproduisant
robot simulé, 288–290
ROC (courbe), 121

Copyright © 2010 Eyrolles.