N^o D'ORDRE : 8156

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PARIS XI
DISCIPLINE : INFORMATIQUE

Présentée et soutenue publiquement
par

JÉRÉMIE MARY

le 12 Décembre 2005

Étude de l'Apprentissage Actif, Application à la Conduite d'Expériences

sous la direction de

ANTOINE CORNUÉJOLS et MICHÈLE SEBAG

Jury :

STÉPHANE	CANU	(Rapporteur)
MARC	SEBBAN	(Rapporteur)
OLIVIER	BOUSQUET	
CHRISTINE	FROIDEVAUX	
MICHÈLE	SEBAG	
JEAN-DANIEL	ZUCKER	

TAO-INRIA
Tél (33)1 69 15 66 07 / Fax (33)1 69 15 65 86
Équipe TAO, LRI, UMR 8623 (CNRS - Université Paris-Sud),
Bat 490 - Université Paris-Sud 91405 ORSAY Cedex France
e-mail: Jeremie.Mary@lri.fr

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

INRIA

Résumé

Classiquement l'apprentissage s'intéresse au cas où les exemples sont indépendants identiquement distribués. En revanche, dans l'apprentissage actif, l'algorithme d'apprentissage a la possibilité d'influer sur la distribution des exemples plutôt que de « subir » la distribution naturelle, en ce point, l'approche possède des fondements communs avec le boosting.

Cette thèse s'incrit dans le formalisme PAC ; nous établirons des bornes de convergences améliorées dans différents cas d'apprentissage actif. Tout d'abord dans le cadre ou une stratification des exemples est possible, nous commencerons par estimer la variance de l'erreur d'apprentissage avant de répartir les exemples de façons optimisée. Nous présenterons des algorithmes pour lesquels la vitesse de convergence est améliorée (y compris dans le cas de nombres de couverture non polynômiaux).

Ensuite, nous utiliserons des outils venus de l'intégration numérique (les suites à faible discrépance) pour assurer une vitesse de convergence en $1/n$ dans le cas où l'on peut choisir librement les points de l'ensemble d'apprentissage (sans faire l'hypothèse de l'erreur nulle).

Les bornes obtenues seront systématiquement testées sur des données expérimentales.

Enfin, nous verrons comment transposer ces idées dans le cadre de la sélection d'attributs. Nous positionnerons ces travaux dans le domaine de la Bio-Informatique et nous montreront que l'utilisation des courbes ROC permet de définir une stratégie pour effectuer l'apprentissage. Nous complèterons cette partie sur la sélection d'attributs en essayant de faire du bagging sur des ordres issus de la sélection.

Abstract

Supervised Machine Learning usually deals with problems in which the examples are independent and equally distributed. In Active Learning, however, the learning algorithm can act upon the examples distribution, to reduce either labelling costs or example acquisition costs.

This thesis relates to PAC (Probably Approximately Correct) Learning. It establishes improved convergence boundaries in different cases of Active Learning. A first research axis concerns stratified data acquisition guided either by the class of the examples or by an auxiliary variable. In this context, an estimation of the variance of the error is used to suggest the distribution which would be optimal relatively to the different strata. The advantage of this approach is to increase convergence rate. Our results concern the definition of improved boundaries in the general framework of Donsker classes.

The second research axis, which may be linked to Experiment Design, deals with a priori choices of database instances. In this framework, inspired by numerical integration research, we proposed using low-discrepancy mathematical series. The advantage is to guaranty a $1/n$ (n being the number of examples) convergence rate without the null empirical error requirement. The resulting boundaries were experimentally analysed.

Bio-informatics gave the incentive for the third axis, which deals with attribute selection. ROC curves are used in a heuristic to actively select attributes, while an alternative approach, based upon Ensemble Learning, sorts the attributes.

Table des matières

Remerciements

Tant il est vrai que l'on ne se construit pas seul, il me paraît essentiel de commencer la conclusion de ces trois années de thèse en rendant à César ce qui lui appartient. Afin d'éviter les impairs diplomatiques, je procèderai par ordre chronologique d'apparition dans le processus ayant conduit à cette thèse :

Je m'adresse donc en premier lieu à mes parents, sans qui j'aurais presque sûrement stoppé ma carière universitaire au niveau du CP pour cause de refus d'apprentissage de cette science arbitraire qu'est la lecture... Je n'ai pas toujours été facile (pour ne pas dire jamais), merci donc à eux ainsi qu'à mon frère pour leur indéfectible soutien, sans eux je ne serais pas devant vous pour présenter ce travail.

Ensuite, merci à Éric Cancès de l'école des Ponts et Chaussées pour m'avoir poussé à m'engager sur les voies de la recherche ainsi qu'à Céline Rouveirol pour m'avoir fait confiance en m'acceptant dans son DEA malgré un profil de mathématicien.

Merci à Antoine Cornuéjols et Christine Froidevaux de m'avoir initié au monde de la recherche universitaire pendant mon stage de DEA. Double remerciement pour Antoine, puisqu'avec Michèle Sebag ils ont accepté d'être mes directeurs de thèse, me permettant de pousuivre mes travaux au sein du LRI dans des conditions de recherche idéales. En particulier, merci à eux pour leur confiance et la grande liberté dont j'ai bénéficié pendant ces années.

Merci à mes contacts chez les biologistes, principalement à l'intitut Curie : Géraldine Mercier, Nathalie Berthault et Marie Dutreix. Ils m'ont permis d'apporter ma modeste contribution à l'une des épopées du XXI^{e} siècle : le décryptage de l'information génomique.

Un merci tout particulier à Olivier Teytaud, pour m'avoir ouvert de nouvelles directions de recherche et pour son enthousiasme de tous les jours. C'est promis, je te rend bientôt le Van Der Vaart, le Vidyasagar et le Devroye...

Je me dois également d'adresser mes remerciements au LRI qui m'a accueilli ainsi qu'aux personnes qui m'ont entouré pendant ces années. Merci donc à Sarah Cohen-Boulakia (tu ne le sais pas encore, mais finalement je t'ai coiffée sur le poteau pour le babou...), Sandra Bsiri (met de l'huile dans tes roulements), Sylvain Gelly (un jour on sera PAC riches -ou pas-), Nicolas Baskiotis (non, je ne veux plus jouer à crack-attack),

Nicolas Bredeche (appel de phares), Jérôme Azé (l'homme qui ne dort jamais), Mathieu Roche (plus que deux ans à tenir), Patrick Marty (vive le Souschen) ainsi que les autres membres de l'équipe…

Un très grand merci à Marc SEBBAN et Stéphane CANU d'avoir accepté d'être mes rapporteurs, ainsi qu'à Christine FROIDEVAUX et Olivier BOUSQUET d'avoir accepté d'être membres du jury.

Un mot également pour Céline, ma chère et tendre, qui m'a soutenu durant la phase terminale.

Enfin je conclurai en remerciant tous ceux que j'ai oublié, en leur adjoignant mes excuses pour cet oubli…

Introduction

Cela fait maintenant trois ans que lorsque je dois parler en société du sujet de ma thèse, j'élude la question en disant que je travaille sur l'Intelligence Artificielle... Les spécialistes savent bien que ce terme est trop générique pour être porteur d'information, mais il possède l'avantage d'éveiller un écho dans l'esprit du grand public. On peut sans doute à ce sujet remercier Spielberg et son navet AI. A ceux qui ont essayé d'en savoir davantage, j'ai avoué que le terme correct serait plutôt *Machine Learning*, que l'on peut joliement traduire comme Antoine Cornuéjols « Apprentissage Artificiel ». Même si la nuance ne saute pas aux yeux du profane, elle est de taille ! En effet, au risque d'en décevoir certains, non, mes petits robots ne sont pas intelligents, pas plus que je ne discute avec mon ordinateur lors de la pause café.

Bref, les machines ne sont pas plus intelligentes dans nos laboratoires que chez le contribuable moyen. Alors pourquoi parler d'apprentissage ? Ce terme regroupe en fait tous les moyens par lesquels il est possible d'apprendre à une machine à remplir une tâche donnée. Si l'objectif peut-être atteint par une suite d'opérations simples, on dispose généralement d'un algorithme efficace adapté à cette tâche. Il suffit alors de l'implémenter dans le langage de son choix. Ainsi, il est relativement simple d'apprendre à une machine à faire de la comptabilité ou à tenir à jour un stock.

Cependant, les choses ne se passent pas toujours aussi bien : même un bon joueur d'échecs, n'est pas capable d'expliquer pas à pas comment bien jouer. En effet, le joueur combine son instinct du jeu et ses capacités de déduction dans la tâche. Et il en va de même dans de nombreux autres domaines, tel le médical ou l'analyse de données où parfois l'on ne dispose même pas d'un expert capable d'effectuer correctement l'analyse. Il arrive aussi que ce soit l'ampleur de la tâche qui pose problème : on connait un algorithme atteignant l'objectif mais son temps d'exécution est trop élevé. Par exemple, il est possible de résoudre le problème du joueur d'échec en créant l'arbre de tous les coups possibles jusqu'à toutes les fins de parties possibles (comme un humain le fait naturellement en jouant au *tic-tac-toe)* mais il n'existe pas d'ordinateur capable de le calculer avant que le soleil n'ait brûlé tout son hydrogène...

C'est à ce moment que l'apprentissage artificiel prend tout son sens. Il s'agit de mettre au point des techniques par lesquelles un ordinateur va pouvoir apprendre une tache « par l'exemple ». Ainsi, si en analysant des milliers de parties, la machine se rend compte que dans une situation donnée, tel mouvement conduit à 90% de victoires en fin de partie, elle jugera que c'est un mouvement favorable. Bien sûr, il est impossible d'avoir en mémoire toutes les parties possibles, la machine devra donc faire des regroupements de situations qui seront guidés le plus souvent par des experts humains.

Malheureusement, il n'est souvent pas possible de parvenir à réaliser une tâche de manière parfaite. Cependant, il existe plusieurs exemples où la machine, par l'utilisation de ces techniques est maintenant supérieure à l'homme. On peut citer le « puissance quatre », le backgammon (jeu comportant une part d'aléatoire), la gestion de systèmes de stocks répondant à une demande aléatoire. De même, d'autres applications, hier encore irréalistes font de grands progrès et sont en passe d'être transférés dans notre vie quotidienne : reconnaissance de la parole, dialogue homme-machine, analyse de textes, recherche de nouveaux médicaments, analyse de l'activité cérébrale, conduite d'automobiles...

Plus précisément, cette thèse s'intéresse à l'apprentissage actif. Habituellement en apprentissage, les exemples à partir desquels l'on travaille sont indépendants identiquement distribués. Cela signifie que l'on prend les données dans le monde réel sans se poser de questions et sans influer sur la collecte. C'est exactement le contraire de l'attitude d'un être humain apprenant une nouvelle tâche. En effet, celui-ci focalise son attention sur des points précis. Par exemple une personne apprenant le tennis commencera par répéter plusieurs fois le même mouvement en posant des questions à un instructeur pour savoir ce qui va et ne va pas. En outre, il n'approfondira son apprentissage qu'après avoir maîtrisé les mouvements de base. Notre but est d'étudier la possibilité et les avantages de doter les machines de tels comportements. Plus formellement, cela se traduit pour l'algorithme d'apprentissage par la possibilité d'influer sur la distribution des exemples lors de la phase d'apprentissage.

Ces méthodes, naturelles pour les humains qui focalisent leur attention sur les points qu'ils ne comprennent pas, sont nouvelles en apprentissage artificiel. Ceci n'est pas sans rappeler le *boosting* (bien qu'il ne s'agisse pas à proprement parler d'une technique d'apprentissage actif) qui essaye de donner plus ou moins d'importance à certains exemples. On ne peut s'empêcher de penser que puisque l'apprentissage actif permet d'influer sur les exemples d'apprentissages, il possède certainement des liens théoriques avec le *boosting*, liens qui restent à explorer.

Même avec ces quelques précisions, le domaine reste extrêmement vaste, et je n'en n'ai bien sûr pendant ma thèse exploré qu'une partie. Ce manuscrit commence par une vue d'ensemble des travaux présentés. La première partie présente l'état de l'art, les parties suivantes détaillent les contributions de mon travail de thèse relatives à l'apprentissage actif dans un formalisme PAC et à la sélection d'attributs. Enfin, une annexe présente succintement quelques travaux réalisés parallèlement à la ligne directice.

Vue d'ensemble

Les fondements des travaux de cette thèse trouvent leur source dans mon stage de DEA sous la direction d'Antoine Cornuéjols et de Christine Froidevaux au sein du groupe Bio-informatique du LRI. Ce stage était motivé par la demande de biologistes de l'institut Curie d'Orsay, Géraldine Mercier, Nathalie Berthault et Marie Dutreix, mettant au point des nouvelles techniques d'analyse du génome (puces à ADN). Celles-ci, après avoir développé des techniques d'acquisition de données biologiques sur des levures exposées à des radiations, elles se sont heurtées à un problème plus difficile que ce à quoi elles s'attendaient : l'analyse des résultats. C'est alors que nous sommes intervenus. Nous avons étés parmi les premiers en France à utiliser des techniques issues de l'intelligence artificielle dans l'analyse du génome. En particulier, nous nous sommes intéressés aux méthodes de sélection d'attributs plutôt qu'à des techniques de classification. Ces travaux ont donnés lieu à plusieurs publications dont une dans *Nucléic Acid Research* [Mercier et al.; 2004] (revue internationale, facteur d'impact de 7.24 en 2004).

Ce contact avec la biologie nous a poussé à définir un cadre de travail pour l'intelligence artificielle dont les contraintes sont issues du monde réel :

- Les communautés extérieures font appels à nous essentiellement lorsqu'elles rencontrent des problèmes non-linéaires (qui ne peuvent se traiter comme s'ils étaient linéaires) ou quand les données dont elles disposent ne sont clairement pas indépendantes identiquement distribuées (et quand «faire comme si» conduit à de mauvais résultats empiriques). Ce phénomène est dû au fait que les statisticiens ont déjà résolu dans les grandes lignes la classe des problèmes linéaires i.i.d. Selon moi, l'avenir de l'apprentissage passe par la maîtrise de ces nouveaux problèmes. Nous nous y sommes attaqués dans cette thèse, nous verrons une partie des vastes problèmes que cela pose et des débuts de solutions.
- Les données collectées à l'heure actuelle contiennent souvent un nombre énorme d'attributs *(i.e.* de nombreux descripteurs). En effet, on se dit qu'il est plus facile de traiter le problème en ayant trop d'information plutôt que pas assez. Malheureusement l'abondance d'informations est parfois source de difficultés.
- L'acquisition de données coûte cher... Pour cette raison, l'expert est souvent limité dans leur acquisition ou leur étiquetage. Par contre les liens avec la communauté de l'apprentissage s'appronfondissant, les experts sont favorables à une concertation avec les apprentistes avant et/ou pendant la démarche d'acquisition. Ceci nous a conduit à considérer la problématique des plans d'expériences [Montgomery; 2001] *i.e.* la définition *a priori* de l'ensemble des données expérimentales considérées, ou

encore l'apprentissage actif [Tong and Koller; 2000], *i.e.* le choix des prochaines expérimentations compte tenu des exemples déjà disponibles et des hypothèses abstraites. Ainsi, on ne se contente plus d'attendre que toutes les expériences soient finies pour analyser les données, mais on rentre dans une véritable boucle d'apprentissage où les résultats d'analyse guident les nouvelles expérimentations.

Ces trois points illustrent une tendance générale, consistant à développer une approche spécifique en fonction des applications considérées. Cette voie est celle que montre le *no-free lunch theorem* [Wolpert and Macready; 1995] selon lequel il est impossible de trouver un algorithme d'apprentissage meilleur que les autres sur tous les types de données. Par conséquent, l'étude des algorithmes, des types de données et l'adaptation sont plus que jamais de mise dans notre domaine. Les premiers mouvements dans ce sens sont le développement de cartes de compétences des algorithmes et l'étude des transitions de phase en apprentissage. Pour cette raison, le domaine repose sur l'étude fine des types d'algorithmes, des types de données et de leur adéquation réciproque.

Cette thèse commence par une description rapide du formalisme PAC (Probablement Approximativement Correct [Vidyasagar; 2002][Van Der Vaart and Wellner; 1996][Devroye et al.; 1997]) et par les principaux résultats de convergence dans ce domaine. En particulier nous verrons qu'aujourd'hui d'autres outils viennent en complément de la VC-dimension dans le calcul de bornes. En effet, grâce aux classes de Glivenko-Cantelli et de Donsker, il existe maintenant des bornes en $1/\sqrt{n}$ alors que les nombres de couvertures sont non-polynomiaux. Je présenterai également des méthodes issues du calcul numérique des intégrales. En effet, les techniques dites de « Quasi-Monte-Carlo » utilisent des suites dites à « faible discrépance » pour déterminer les points à utiliser dans l'approximation du calcul d'une intégrale. Nous verrons que ces curiosités analytiques peuvent aussi être utilisé en apprentissage pour améliorer la vitesse de convergence.

Une seconde partie aborde l'apprentissage actif sous un angle inspiré de l'apprentissage PAC. Nous verrons comment la stratification dans un apprentissage actif en deux passes permet d'améliorer les vitesses de convergence et les bornes en généralisation (par rapport aux résultat de Vapnik). L'idée est d'attaquer les problèmes d'apprentissage en deux temps : dans un premier temps, on évalue la variance en fonction de la classe ou d'une variable auxilaire. Dans un second temps on répartit les points utilisés pour l'apprentissage de façon optimale par rapport à cette variance. Nous démontrerons que cette manière de procéder fournit un avantage par rapport à une équirépartition des données dans le cas où le problème ne présente pas une variance uniforme pour l'erreur.

Nous nous placerons ensuite dans un cadre plus proche des points d'expériences, où l'on peut choisir librement les points à utiliser pour effectuer l'apprentissage. Nous montrerons l'apport de techniques issues du calcul des intégrales à la Monte-Carlo en montrant qu'il est est utile de considérer des suites déterministes (en particulier à faible discrépance). Ici encore, il s'agit d'un cas d'apprentissage actif puisque l'utilisation de cette méthode requiert de choisir les points sur lesquels l'apprentissage est effectué. L'approche présentée est fondée sur l'inégalité de Koksma-Hlawaka et les propriétées des suites à faible discrépance. L'intérêt est de permettre une de convergence plus rapide des

algorithmes d'apprentissage.

La troisième partie, traite de sélection d'attributs. Nous verrons l'intérêt d'utiliser des techniques de sélection d'attribut adaptées au problème considéré. Ceci nous conduira vers une idée nouvelle transportant les idées de base de l'apprentissage actif dans le domaine de la sélections d'attributs. Celle-ci consiste à minimiser le nombre de mesures d'attributs effectués en attaquant le problème sous un angle « actif ». La méthode que nous présenterons est guidée par l'intuition qu'il est souvent possible de détecter avec peu de moyens *(i.e.* d'exemples) que certains attributs ne sont pas utiles. Ceci permet d'éliminer rapidement ces attributs pour se concentrer sur les autres. Notre approche est basée sur l'étude de la courbe ROC des algorithmes de sélection d'attributs.

Nous poursuivrons l'étude de la sélection d'attributs par un essai d'attaque de problèmes non-linéaires en conjonction avec ROGER (un algorithme évolutionnaire de sélection d'attribut se basant sur l'AUC pour sa *fitness*). L'idée est ici d'étudier des techniques d'ensemble de classement d'attributs, dont le premier représentant historique est WINNOW. Nous verrons que le *bagging* (prise de décision en consultant plusieurs experts les plus indépendants possible) peut apporter des résultats encourageants dans le domaine non-linéaire.

Enfin je terminerai par une annexe présentant succinctement des travaux n'ayant pas trouvé leur place dans cette thèse. En particulier en présentant des outils développés pendant ces trois ans (dont un simulateur de robots, WoB ayant servi à la thèse de Nicolas Godzik et à des enseignement à Polytechnique). On y trouvera également des résultats sur de nouvelles directions de recherches autour de la programmation dynamique (utilisation de Taylor Lagrange pour définir des pseudo-métriques sur des variables aléatoires) et l'utilisation dans le cadre de la classification d'images de *Frequent items sets* comme base recodage .

Notations

Pour faciliter la lecture des parties techniques, voici un résumé des différentes notations utilisées. J'essayerai dans la suite, quand cela est possible d'utiliser toujours les mêmes lettres pour désigner les mêmes objets (ainsi P sera sera toujours une probabilité et n le nombre d'exemples dont on dispose pour effectuer l'apprentissage).

On note $\mathbb{R}$ le corps des réels, $\mathbb{N}$ celui des entiers naturels et $\mathbb{Z}$ celui des entiers relatifs.

Pour $a, b \in \mathbb{N}$, $a \leq b$, on note $[\![a, b]\!]$ l'ensemble $\{a, a+1, a+2, \ldots, b\}$.

Pour deux ensembles A et B, on note :

- Si $A \subset B$, alors $\mathbb{1}_A$ dénote la fonction indicatrice de A, définie par $\mathbb{1}_A(x) = 1$ ssi $x \in A$ et 0 sinon,
- $Card\ A$ désigne le nombre d'éléments de A quand celui-ci est fini (aussi appelé Cardinal),
- A^B désigne l'ensemble des applications de B dans A.

Pour une matrice ou un vecteur M :

- $M_{i,j}$ désigne le coefficient à la i^e ligne et j^e colonne. Dans le cas d'un vecteur on M_i désigne la i^e composante.
- M^t désigne la transposée de M. $(M^t)_{i,j} = M_{j,i}$
- $< a, b >$ désigne le produit scalaire de a et b. Par défaut il s'agit du produit scalaire usuel de $\mathbb{R}^n$. A un produit scalaire, on associe la norme : $||a||^2 =< a, a >$

Pour une fonction f de $X \to \mathbb{R}$:

- La norme L_p est la quantité $L_p(f) = ||f||_p = \sqrt[p]{\int_X |f(x)|^p dx}$,
- $L_\infty(f) = ||f||_\infty = \sup_{x \in X} |f(x)|$

Pour une variable aléatoire Z :

- On note $E(Z)$ (ou parfois $\bar{Z}$) l'espérance de la variable aléatoire Z (c'est-à-dire son intégrale). Il sera souvent utile de préciser la loi de probabilité P intervenant dans le calcul de l'espérance. On note alors $E_P(Z)$ (où $E_Z(Z)$ quand on utilise la loi de Z),
- Var$(Z) = E\left((Z - E(Z))^2\right)$ désigne la variance Z. Classiquement l'écart-type $\sigma(Z)$ est défini comme la racine de la variance,
- La norme L_∞ de Z est le $\sup$ de l'ensemble des $x \in X$ tels que $Pr(|Z| > x) > 0$

Pour une suite de variables aléatoires :

- Un ensemble de variables aléatoires est dit **i.i.d.** si elles sont indépendantes identiquement distribuées,
- Habituellement on indicera la suite de variables aléatoires par $\mathbb{N}$, obtenant ainsi un vecteur $(Z_1, Z_2, \dots,)$. Cependant, il arrivera que nous ayons besoin d'indicer des variables aléatoires par des fonctions $f \in F$ dans ce cas l'indiçage sera noté $(Z)_f$.

Pour une loi de probabilité P définie sur un espace X :

- Soit une fonction f définie sur X à valeurs dans $\mathbb{R}$, $||f||_{P,q} = \sqrt[q]{E_P(|f|^q)}$,
- Pour $f, g : X \to \mathbb{R}$, on note d_P ou $L_1(P)$ la métrique sur $\mathbb{R}^X$ définie comme :

$$d_p(f,g) = L_1(P)(f,g) = \int_{x \in X} |f(x) - g(x)| \, dP$$

- $\mathbb{P}_n$ désigne la loi empirique estimée de P à partir de n réalisations.

Pour une famille de fonctions F de $X \to \mathbb{R}$:

- Une **enveloppe** de F est une application qui à $x \in X$ associe une valeur $\geq \sup_{f \in F} |f(x)|$,
- Si P est une mesure de probabilités $||P||_F = \sup_{f \in F} E_P(|f|)$

La probabilité de réalisation d'un évènement sera notée $P($évènement$)$ (ou $Pr($évènement$)$ pour éviter toute ambiguïté).

On note δ_a la mesure de **Dirac** en a. Par construction $\forall f \in \mathbb{R}^X, \ \int_X f \ \delta_a = f(a)$

En général n désignera le nombre d'exemples $z_1, \dots, z_n$ dont on dispose pour effectuer l'apprentissage. Sauf mention explicite, ils seront i.i.d. et distribués selon une loi P. Par abus de notation, on utilisera P pour désigner la loi de l'échantillon et la loi d'un seul élément (en fait la loi de l'échantillon est la loi produit de n copies indépendantes de P).

Une suite de variables aléatoires $Z_1, \dots, Z_n$ converge (quand $n \to \infty$) vers Z :

- Presque sûrement si : avec probabilité 1, $Z_n \to Z$,
- En probabilités si : $\forall \varepsilon > 0, P(|Z_n - Z| > \varepsilon) \to 0$,
- Dans L_p pour $\infty > p > 1$ si : $\sqrt[p]{E(|Z_n - Z|^p)} \to 0$
- En moyenne si : $E_{Z_i}(Z_i) \to E_Z(Z)$
- En loi (ou faiblement) si pour toute fonction continue bornée f,

$$E_f(Z_i) \to E_Z(f(Z))$$

L'espace L_p désigne l'espace des fonctions de norme L_p bornée. L'espace l_p désigne l'espace des suites (x_i) pour lesquelle $\sum |x_i|^p$ converge.

Dans le cas de la classification, le plus souvent possible F désignera notre espace d'hypothèses et L_F l'espace des fonctions de coût associé. De même, on prend comme convention de désigner $\hat{f}$ les approximations tirées de l'utilisation d'une loi empirique et par des f^* les résultats optimaux dans F. On réservera la lettre g pour désigner la fonction idéale (qui n'est pas nécessairement dans F).

Première partie

MATHÉMATIQUES POUR L'APPRENTISSAGE

Chapitre 1

Formalisme Mathématique pour l'Apprentissage

1.1 Apprentissage PAC

Ce chapitre introduit les définitions traditionnelles utilisées en apprentissage, sur lesquelles se fondent les chapitres « Stratification » et « Suites déterministes ».

1.1.1 Généralités

En apprentissage supervisé, le problème posé informellement peut se résumer comme suit :

- On dispose d'une base d'exemples de la forme (Attributs,Étiquette). Le but est d'être capable de prévoir l'étiquette (ou *label*) en fonction de la valeur des attributs (*features)*. On parle d'apprentissage supervisé dans la mesure où la supervision par un expert est nécessaire pour définir le *label* y_i associé à une description x_i.
- Les attributs (*features*) contiennent des informations décrivant l'exemple sur lequel on va travailler. On peut penser par exemple aux valeurs d'expressions de gènes dans le cas de l'étude de biopuces ou à des caractéristiques telles que « jour », « mois », « humidité de l'air » . . . dans le cas des prévisions météo.
- Les étiquettes contiennent le résultat à prévoir. Par exemple, malade ou sain dans le cas des puces à ADN ou le taux de pluviosité pour la météo.

On dispose donc d'un certain nombre d'exemples nous montrant quelle étiquette doit être associée à une série de valeurs d'attributs. L'opération d'apprentissage est la détermination d'une règle (hypothèse, où fonction de X dans Y) permettant de retrouver les étiquettes en fonction des attributs. Une fois choisie cette règle, il reste à déterminer sa qualité pour prédire les étiquettes de nouvelles données (il s'agit de **tester la capacité de généralisation** de notre règle). L'hypothèse de base faite en apprentissage est qu'il existe

un lien entre la manière d'attribuer les étiquettes pour les premières données et pour les nouvelles ; sinon le concept ciblé par l'apprentissage change : il serait donc extrêmement difficile -voire impossible- d'avoir des garanties de résultat.

Une solution simple est d'apprendre par cœur les données initiales. L'hypothèse apprise est alors très bonne sur le jeu de données d'apprentissage, mais répond au hasard sur de nouvelles données. Plus généralement, les stratégies conduisant à d'excellentes performances sur l'ensemble d'apprentissage, et à une performance bien moindre sur le reste de l'espace des instances sont dites sur-apprenantes (*overfitting*). Comment éviter le sur-apprentissage constitue le problème majeur de l'apprentissage (et il se pose avec d'autant plus acuïté que l'on dispose de peu d'exemples).

L'approche PAC (**Probablement Approximativement Correcte**) peut se voir comme une proposition de formalisme universel pour les problèmes d'apprentissage supervisé. Comme nous l'avons dit une difficulté provient du fait que l'on peut être arbitrairement bon sur le jeu de données d'apprentissage et mauvais sur un nouveau. Le but du PAC-learning est d'essayer de **borner la différence de comportement entre le jeu de données d'apprentissage et de nouvelles données.** La borne est définie en probabilité, elle dépendra du nombre d'exemples d'apprentissage.

Plus formellement, sous un angle de statisticien, on peut voir les exemples comme des points d'un espace Z. On peut supposer que ces exemples sont donnés avec une loi de probabilité P et on considère en général que $z_1, \ldots, z_n$ sont indépendants et identiquement distribués. Dans le cas de l'apprentissage supervisé, on a $z = (x, y)$ et l'on vise à être capable de prédire correctement y à partir de x (dans ce cas on appelle les composantes de x les **attributs** et y la **classe** ou encore étiquette ou *label*).

Notons F l'espace des hypothèses, *i.e.* l'espace de recherche de l'apprentissage. Pour toute hypothèse $f \in F$, et pour $z = (x, y)$ on définit $L_f(z)$, application de $Z \to \mathbb{R}$, comme le coût engendré par le fait de prédire $f(x)$ au lieu de y (on a fait l'hypothèse f alors que l'on dispose de l'exemple z). On notera $L_F = \{z \to L_f(z) | f \in F\}$ la famille des fonctions de coût. Dans le cas de la classification, typiquement le coût est nul si $y = f(x)$.

Formellement, le but de l'apprentissage est alors de trouver $f \in F$ tel que L_f ait une espérance la plus faible possible. Cette espérance appelée **coût en généralisation** ou **erreur en généralisation** et est souvent notée par abus de notation L_f (en omettant l'opérateur d'espérance). Un estimateur empirique de l'erreur en généralisation est appelé **coût empirique** ou **risque empirique** est construit sur un échantillon $(z_1, \ldots, z_n) \in Z^n$ des données. En notant $(\hat{L}_f)$ l'erreur empirique on a la formule : $\hat{L}_f = \dfrac{1}{n} \sum_{i=1}^{n} L_f(z_i)$

Par construction, l'erreur empirique $\hat{L}_f$ repose sur l'échantillon des données fait selon la loi empirique notée $\mathbb{P}_n$ estimant la distribution P des exemples. On notera $\hat{L}_f = E_{\mathbb{P}_n} = \hat{E}(L_f)$. Sauf mention contraire, $\hat{E}$ et $\hat{P}$ dénoteront l'espérance et la probabilité par rapport à la loi $\mathbb{P}_n$.

Dans ce formalisme, un **algorithme d'apprentissage** devient une fonction programmable (sur une machine de Turing dotée d'un ruban aléatoire par exemple) de l'ensemble des sous-ensembles finis de Z dans F. Un tel algorithme n'est pas nécessairement déterministe.

La première idée qui vient à l'esprit pour l'apprentissage est la **minimisation du risque empirique**. C'est à dire que l'on cherche dans F un élément minimisant $\hat{L}_f$ nous le verrons plus en détail dans la section 1.2.3. [Devroye et al.; 1997] [Vidyasagar; 2002]

Un paradigme d'apprentissage A pour $Y = \{0, 1\}$ est dit **PAC** (statistiquement probablement approximativement correct) si :

$$\forall \varepsilon > 0, \lim_{n \to \infty} \sup_{P \in \mathcal{P}'} P(|\hat{L}_n - L^*| > \varepsilon) = 0$$

où $\hat{L}_n$ désigne l'espérance empirique de L basée sur n exemples et L^* l'erreur optimale pour $f \in F$.

1.2 Première approche de la VC dim

D'après ce qui précède, l'apprentissage correspond à la sélection d'une hypothèse f dans une famille F de fonctions. L'approche, historiquement la première de l'apprentissage [Vapnik and Chervonenkis; 1971] cherche une hypothèse f minimisant l'erreur empirique. Nous y reviendrons dans la section 1.2.3. Plaçons nous dans le cas de l'apprentissage binaire $Y = \{0, 1\}$ et notons g le « vrai concept ». La base d'apprentissage étant formée d'échantillons $(x_i, g(x_i))$. L'objectif est d'identifier dans l'espace de recherche F la meilleure approximation du concept g, sur la base des échantillons d'apprentissage. On distingue classiquement le cas où g appartient à l'espace de recherche F du cas contraire.

1.2.1 Coefficient de pulvérisation

Définition 1.1 *Étant donné F, une famille de fonctions de $X \to Y = \{0, 1\}$, le* ***n^{ème} coefficient de pulvérisation*** *de F, est :*

$$S(F, n) = \sup_{(x_1, \ldots, x_n) \in X^n} Card\left\{(f(x_1), \ldots, f(x_n)) \,|\, f \in F\right\}$$

Intuitivement, les coefficients de pulvérisation de F mesurent la diversité de l'espace de fonctions F. Dans le cadre de l'apprentissage, F est l'espace d'hypothèses (choisi par nos soins) et les x_i constituent la base d'exemples. Si l'on retient l'hypothèse $f \in F$ pour classer nos exemples, la classe estimée de x_i est $f(x_i)$. On utilise les exemples pour contraindre nos classifieurs. Si les choses se passent raisonnablement bien, plus on dispose d'exemples, plus on sera guidé dans le choix de f.

$S(F,n)$ est donc en fait le nombre de manières dont on peut répartir les classes $f(x_1),\ldots,f(x_n)$, sachant que l'on dispose de n exemples et que f est dans F. Plus ce nombre sera petit plus il sera facile de choisir le «bon» f en ayant peu d'exemples.

On peut remarquer que f étant à valeurs dans $\{0,1\}$, on a $S(F,n) \leq 2^n$ puisque si l'on est complètement libre pour choisir les classes des x_i, on aura 2^n combinaisons possibles (cela revient à compter en binaire avec n bits). Remarquons que si $S(F,n) = 2^n$, c'est qu'il existe une base d'exemples de taille n telle que celle-ci ne nous guide absolument pas dans le choix de f. On commence ici à deviner le lien entre l'apprentissage et les coefficients de pulvérisation.

On peut également définir des coefficients de pulvérisation pour des fonctions à valeurs dans $Y = \mathbb{R}$. On parlera alors de P-pulvérisation et de γ-pulvérisation.

1.2.2 VC-dimension, Pollard-dimension

Définition 1.2 *Étant donné Fune famille de fonctions de $X \to Y = \{0,1\}$, la **VC-dimension** de F est, s'il existe, le plus grand entier n, tel que $S(F,n) = 2^n$. Sinon, (c'est à dire que $\forall n\ S(F,n) = 2^n$) on dit que la VC-dimension est infinie.*

Intuitivement, si la VC-dimension est infinie, on aura beau ajouter des exemples, l'espace d'hypothèses sera toujours tellement riche qu'il sera toujours possible de trouver des hypothèses compatibles avec la base d'apprentissage et telles qu'elles classent différement tout nouvel exemple. En résumé, toute généralisation est impossible. Inversement, une VC-dim finie nous garantit qu'avec assez d'exemples (ce « assez » dépendant de la VC-dim), on identifiera une meilleure hypothèse.

La base d'apprentissage peut être vue comme une contrainte : intuitivement, si les coefficients de pulvérisation n'augmentent pas exponentiellement avec le nombre d'exemples, tout nouvel exemple réduit potentiellement l'ensemble des hypothèses f compatibles avec les exemples (***i.e*** telle que $\forall i \in [\![1,n]\!],\ f(x_i) = y_i$).

Il est capital pour pouvoir garantir la convergence de l'apprentissage que les coefficients de pulvérisation soient bornés de manière sub-exponentielle. Heureusement, il existe un lemme, souvent appelé lemme de Sauer qui borne les coefficients de pulvérisation en fonction de la VC-dim. En effet, si la VC-dim de F est majorée par V alors

$$S(F,n) \leq \sum_{i=0}^{V} C_n^i < \left(\frac{en}{V}\right)^V \quad ;\forall n \geq V \geq 1$$

En fait selon [Vidyasagar; 2002, p76], elle aurait été démontrée indépendamment par [Sauer; 1972], [Vapnik and Chervonenkis; 1971](preuve incomplète) et [Shelah; 1971]. (La première majoration par une somme de C_n^p apparaît en fait dans un lemme conduisant à la preuve).

En remplaçant dans la définition précédente $S(F, n)$ par le $n^{\text{ème}}$ coefficient de P-pulvérisation, on obtient la Pollard-dimension qui est le pendant de la VC-dimension pour $Y = \mathbb{R}$

Définition 1.3 (Pollard-dimension) *[Vidyasagar; 2002, p74]. Soit $(X, \mathcal{S})$ un espace mesurable et soit $F \in [0,1]^X$ une famille de fonctions mesurables. Un ensemble $S = \{x_1, \ldots, x_n\} \subseteq X$ est **P-pulvérisé** par F s'il existe un vecteur $c \in [0,1]^n$ tel que, pour tout vecteur binaire $e \in \{0,1\}^n$, il existe une fonction correspondante $f_e \in F$ telle que :*

$$f_e(x_i) \geq c_i \text{ si } e_i = 1, \text{ et } f_e(x_i) < c_i \text{ si } e_i = 0$$

*La **P-dimension** de F, notée $P - dim(F)$ est le plus grand entier n tel qu'il existe un ensemble de cardinal n qui est P-pulvérisé par F.*

Un exemple classique : pulvérisation des points du plan par des droites

Prenons pour l'espace X des instances le plan réel et pour famille de fonctions F est l'ensemble de droites affines. Chaque droite $f \in F$ sépare le plan en deux parties ; un point x est classé comme positif ou négatif selon la valeur de $f(x)$.

La figure 1.1 montre que les droites affines pulvérisent les ensembles de trois points (non-alignés) mais pas les ensembles de quatre points. C'est donc que la VC-dim des droites affines du plan est trois. De même on montre que celle des rectangles est quatre.

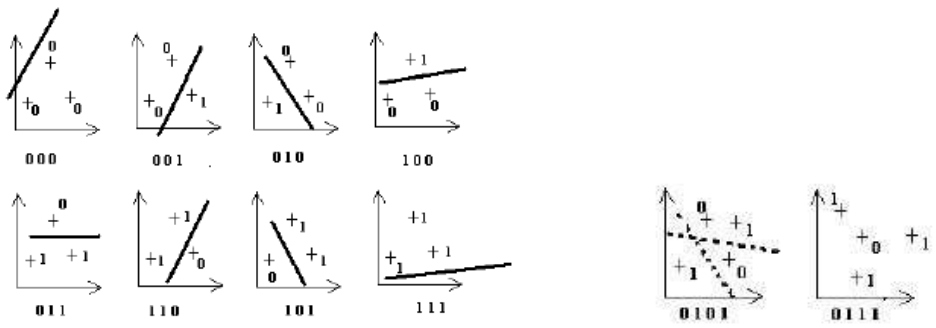


FIG. 1.1 : *(à gauche) Pour trois points non-dégénérés les droites affines pulvérisent tous les indiçages - (à droite) Par contre pour quatre points, il existe des indiçages ne pouvant pas être réalisés par un classifieur basé sur une droite affine.*

1.2.3 Minimisation du risque empirique

Un moyen de choisir une fonction f dans F est de minimiser le risque empirique $\hat{L}$ (l'estimateur du risque). Par exemple, si la fonction de coût est celle associée à la norme L_1 on pourra minimiser :

$$\hat{L}_n(f) = \frac{1}{n} \sum_{i=1}^{n} |f(x_i) - y_i|$$

Notons $\hat{f}_n$ une hypothèse minimisant l'erreur empirique sur les n premiers exemples d'apprentissage.

$$\hat{f}_n \in \operatorname{argmin}_{f \in F} \hat{L}_n(f)$$

Il est important de comprendre que même $\hat{f}_n$ fonction optimale sur la base d'exemples, cela ne donne pas de garanties quant à son comportement d'optimalité sur de nouveaux exemples. C'est tout le problème du sur-apprentissage : l'hypothèse peut-être sur-adaptée à nos données. Cette sur-adaptation se révèle néfaste dans deux cas de figure :

- Si les données sont bruitées (on dit alors que f *fitte* le bruit),
- Si le concept cible g n'appartient pas à l'espace d'hypothèses F considéré.

Le cas limite est celui de l'apprentissage «par cœur», ne donant aucune garantie de généralisation.

L'objectif devient ainsi que l'erreur en généralisation de $\hat{f}_n$ soit proche de l'erreur optimale sur F $L^* = \inf_{f \in F} L(f)$.

Dans le cas où le concept cible appartient à F, et où les données ne sont pas bruitées, $L^* = 0$ (désigné dans la littérature sous de nom de « cas de l'erreur nulle »). Alors pour F de taille finie et si $\hat{f}$ désigne la limite de $\hat{f}_n$, on a alors pour une famille de fonctions de taille finie F on a par Kolmogorov ([Li and Vitanyi; 1993]) :

$$P(L(\hat{f}) > \varepsilon) \leq Card(F) e^{-n\varepsilon}$$

et

$$E(L(\hat{f}_n)) \leq \frac{1 + \log(Card(F))}{n}$$

où E désigne l'opérateur espérance et $\hat{f}$ la limite de $\hat{f}_n$ quand n tend vers ∞.

On en déduit que dans le cas de l'erreur nulle, il suffit que n soit grand devant $\log(Card(F))$. L'utilisation de la VC-dim permet de s'affranchir de l'hypothèse d'existence d'une erreur minimale nulle.

Inégalité de Vapnik-Chervonenkis

Soit μ une mesure de probabilité de (X, Y) sur $\mathbb{R}^d \times \{0, 1\}$. Soit μ_n une mesure empirique basée sur n réalisations de μ (ces réalisations constituant la base d'apprentissage).

Pour un ensemble mesurable $A \subset \mathbb{R}^d \times \{0, 1\}$, $\mu(A) = P((X, Y) \in A)$ et $\mu_n = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{(X_i, Y_i) \in A\}}$.

En reprenant les notations précédentes pour $L, \hat{L}_n, f$ et F on a :

$$\sup_{f \in F} |\hat{L}_n(f) - L(f)| = \sup_{A \in \mathcal{A}} |\mu_n(A) - \mu(A)|$$

où $\mathcal{A}$ est la collection de tous les ensembles de la forme :

$$\{\{x|f(x)=1\}\times\{0\}\}\cup\{\{x|f(x)=0\}\times\{1\}\} \text{ avec } f\in F$$

Théorème 1.1 (Inégalité de Vapnik-Chervonenkis) *Pour toute mesure μ et pour $\mathcal{A}$, $\forall\varepsilon>0$ si $S(\mathcal{A},n)$ désigne le $n^{ème}$ coefficient de pulvérisation de $\mathcal{A}$:*

$$P(\sup_{A\in\mathcal{A}}|\mu_n(A)-\mu(A)|>\varepsilon)\leq 8S(\mathcal{A},n)e^{-n\varepsilon^2/32}$$

La preuve est dans [Devroye; 1982].

Remarque 1 :
Ce résultat est une amélioration de la borne initiale donnée par [Vapnik; 1995].

♦

Remarque 2 :
Compte tenu du lemme de Sauer, (voir 1.2.2) on peut borner les coefficients de pulvérisation et donc la déviation de l'erreur en fonction de la VC-dim.

♦

Ce résultat montre qu'un algorithme minimisant l'erreur empirique converge vers une hypothèse d'erreur optimale avec une vitesse **exponentielle**. Ce résultat est de très bon augure pour l'apprentissage puisqu'il donne des garanties sur le pire cas possible. En réalité, le coefficient $S(\mathcal{A},n)$ vient quand même un peu ternir ce résultat, mais on peut utiliser le lemme de Sauer pour le borner. Il faut tout de même signaler que les praticiens savent bien que cette borne est assez lâche, justement car elle correspond au pire cas possible. Évidemment, dans le cas où la VC-dim est infinie, ce théorème ne nous fournit plus de garanties sur la qualité de l'hypothèse apprise.

1.3 Approfondissements

Afin d'entrer davantage dans les détails de la théorie de Vapnik-Chervonenkis, il est nécessaire d'introduire quelques définitions, qui ne sont habituellement utilisées que dans les preuves.

1.3.1 Classes de Glivenko-Cantelli

Définition 1.4 *étant donnée L_F une famille d'applications de $(X,Y)\to\mathbb{R}$, et $\mathcal{P}$ une famille de distributions sur (X,Y), On dit que L_F est faiblement (resp. fortement) $\mathcal{P}$-**Glivenko-Cantelli** [Van Der Vaart and Wellner; 1996] pour la convergence faible (resp.*

pour la convergence presque sûre) si et seulement si quelle que soit la distribution $P \in \mathcal{P}$, on a :

$$\sup_{L_f \in L_F} |\frac{1}{n} \sum_{i=1}^{n} L_f(z_i) - E_P(L_f)| \xrightarrow{n\to\infty} 0$$

ce qui est équivalent à :

$$\sup_{L_f \in L_F} |E_{\mathbb{P}_\kappa}(L_f) - E_P(L_f)| \xrightarrow{n\to\infty} 0$$

au sens de la convergence faible (resp. presque sûre). Et où $z_i = (x_i, y_i)$ $(z_1, \ldots, z_n) \in (X, Y)^n$ sont n réalisations indépendantes d'une variable aléatoire de loi P.

*De plus L_F est dite **universellement Glivenko-Cantelli** si L_F est $\mathcal{P}^*$-Glivenko-Cantelli où $\mathcal{P}^*$ est l'ensemble de toutes les distributions P définies sur (X, Y).*

*Enfin, L_F est dite **uniformément Glivenko-Cantelli** si en plus d'être universellement Glivenko-Cantelli, la vitesse de convergence de la mesure empirique est uniformément majorée en P :*

$$\forall \varepsilon > 0, \exists N,\ \forall P^* \in \mathcal{P},\ n > N \Rightarrow Pr(||\mathbb{P}_n - P||_{L_F} > \varepsilon) < \varepsilon$$

où $\mathbb{P}_n$ est la distribution empirique estimant P à partir de n réalisations i.i.d. et $||P||_{L_F} = \sup_{L_f \in L_F} E_P(|L_f|)$

Notons que la loi des grands nombres de la théorie classique des probabilités nous dit que pour tout P et L_f,

$$\forall n \geq N\ Pr(\sup_{m \geq n} |\mathbb{P}_m(L_f) - P(L_f)| > \varepsilon) \xrightarrow{N\to\infty} 0$$

Pour qu'une classe soit Glivenko-Cantelli uniforme, il faut de plus que le temps de convergence soit borné pour tout P et tout L_f.

1.3.2 Classes de Donsker

Nous souhaitons enfin disposer de certaines hypothèses de régularité sur les familles de mesures concernées.

Définition 1.5 (Famille tendue) *On dit qu'une famille de probabilités (ou de mesures) $\mathbb{G}$ est tendue si :*

$$\forall \varepsilon > 0\ , \exists K_\epsilon \textit{ compact de } \mathbb{R}^d \textit{ tel que } \sup_{P \in \mathbb{G}} P(\mathbb{R}^d \setminus K_\varepsilon) \leq \varepsilon$$

Dans le cas d'une famille tendue $\mathbb{G}$, il est ainsi possible de choisir *a priori* un compact $K\varepsilon$ et de ramener l'étude de convergence à l'étude des propriétés des mesures considérées dans K_ε.

L'intuition est de contrôler la répartition des «masses» d'une famille de mesures. En effet, pour les mesures P de $\mathbb{G}$, il est possible de choisir uniformément un compact de $\mathbb{R}^d$ (donc un fermé borné) tel que la mesure de ce qui n'est pas dans ce compact soit inférieur à ε. On dira qu'il s'agit d'une condition imposant que pour toutes les mesures de $\mathbb{G}$, l'essentiel de la masse se situe dans un fermé borné (et donc ne part pas à l'infini).

Définition 1.6 (Classes de Donsker) *étant donné une famille L_F de fonctions de (X, Y) dans $\mathbb{R}$, on dit que L_F est $\mathcal{P}$-Donsker (ou que c'est une $\mathcal{P}$-classe de Donsker) [Van Der Vaart and Wellner; 1996, p81] ssi :*

1. $\forall P \in \mathcal{P}, \sup_{L_f \in L_F} |L_f(x) - E_P(L_f)| < \infty$
2. *avec $\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^{n} \delta_i$ et $\mathbb{G}_n = \sqrt{n}(\mathbb{P}_n - P)$, pour tout $P \in \mathcal{P}$, $\mathbb{G}_n$ converge faiblement dans $l_\infty(L_F)$ vers un certain $\mathbb{G}$ tendu mesurable (attention $\mathbb{G}$ est bien une famille car P n'est pas fixe).*

Cette convergence correspond à la convergence faible pour les classes de Glivenko-Cantelli avec en plus une notion de vitesse. Il s'agit de la même vitesse que dans les classes de VC, c'est à dire que l'on demande une précision en $\varepsilon \sim 1/\sqrt{n}$. La différence avec les classes de Glivenko-Cantelli est que l'on a pas de lien entre les classes presque sûres et faibles. La notion de classe de Donsker est plus générale, plus précise mais asymptotique.

Si $\mathcal{P}$ est l'ensemble de toutes les distributions sur X alors une $\mathcal{P}$-classe de Donsker est appelée **classe universellement de Donsker.** L'intérêt de ces classes est du même ordre que les classes de VC. La principale différence tient au fait que l'on a un résultat asymptotique sur l'erreur d'apprentissage au lieu d'une borne.

Remarques :
- Une $\mathcal{P}$-classe de Donsker est $\mathcal{P}$-Glivenko-Cantelli,
- Une classe de VC-dimension finie est une classe de Donsker universelle et est une classe de Glivenko-Cantelli uniformément en la distribution mais l'inverse n'est pas vrai.

1.3.3 Liens entre F et L_F

L'idée générale des statistiques procède en deux temps : 1) on essaye de montrer que la famille de fonction n'est pas trop compliquée ; 2) on montre que les variations du sup *sur l'espace de fonctions n'est pas trop grand. Une difficulté pour les applications de l'apprentissage réside dans le fait que l'on connait généralement l'allure de F mais pas celle de L_F (puisque L_F fait intervenir le concept cible qui est inconnu). Cette section récapitule des résultats liant les deux espaces, renvoyant le lecteur à [Vidyasagar; 2002, p146-151] pour les preuves et davantage de détails (le théorème 1.2 et 1.6 donnés ici, correspondent en fait à des morceaux de la preuve du th5.12).*

Théorème 1.2 (Dimensions, pulvérisation et fonctions de coût) *Soit F une famille de fonctions sur X à valeurs dans $\{0,1\}$, et L fonction de coût définie sur $X \times \{0,1\}$ telle que $L(a,b) = 1$ si $a \neq b$ et 0 sinon. Alors :*

$$S_0(L_F, n) \leq S(F, n)$$
$$S_1(L_F, n) \leq S(F, n)$$
$$S(L_f, n) \leq 2S(F, n)$$

où $S_i(L_F, n)$ est le nombre de pulvérisations de L_F restreint à $X \times \{i\}$ pour $i \in \{0,1\}$ et $S(F,n)$ est le $n^{ème}$ coefficient de pulvérisation de F.

De plus si F a pour VC-dimension V, alors L_F a pour VC-dimension V.

Théorème 1.3 (Donsker et fonctions de coût) *Si F, famille de fonctions, est Donsker, alors L_F l'est aussi pour toute fonction de coût $L(a,b)$ lipschitzienne.*

Théorème 1.4 (Glivenko-Cantelli et fonctions de coût) *Soit F une famille de fonctions définies sur X à valeurs dans $Y = [0,1]$ et $\mathcal{P}$ une famille de distributions sur $X \times Y$. Soit $\mathcal{P}_M$ la famille des distributions marginales sur X correspondant à $\mathcal{P}$. Supposons que F est $\mathcal{P}_M$-Glivenko-Cantelli et que $L(y,.)$ est uniformément équicontinue, c'est-à-dire que :*

$$\forall \varepsilon, \exists \delta, \forall y, \forall (u_1, u_2) \in Y^2, |u_1 - u_2| \leq \delta \Rightarrow |L(y, u_1) - L(y, u_2)| \leq \varepsilon$$

ce qui est en particulier le cas pour $L(y_1, y_2) = |y_1 - y_2|^s$ avec $s \geq 1$.
Alors L_F est Glivenko-Cantelli

Théorème 1.5 (Glivenko-Cantelli et fonctions de coût, classification) *Sous les mêmes hypothèses que le théorême précédent. Supposons que $|L(y,y')| = |y - y'|$.*
Alors L_F est $\mathcal{P}$-Glivenko-Cantelli si et seulement si F est $\mathcal{P}_\mathcal{M}$-Glivenko-Cantelli.

Théorème 1.6 (Nombres de couverture et fonctions de coût) *Soit F une famille de fonctions à valeurs dans $Y = [0,1]$. Supposons que la fonction de coût vérifie pour un couple $\varepsilon, \varepsilon'$ de réels positifs :*

$$\forall y_1, y_2, y_3 | y_2 - y_3| \leq \varepsilon' \Rightarrow |L(y_1, y_2) - L(y_1, y_3)| \leq \varepsilon$$

alors $N(\varepsilon, L_F) \leq N(\varepsilon', F)$.

La définition des nombres de couverture est donnée un peu plus loin section 1.4.4.

1.3.4 Bornes sur la vitesse de convergence

Voila la forme généralisée du théorème déjà vu lors de la présentation de la minimisation du risque empirique (section 1.2.3), il est tiré de [Vidyasagar; 2002, p196] :

Théorème 1.7 (Vitesse de convergence dans les familles de P-dimension finie) *Soit L_F un espace de fonctions sur $X \times Y$ à valeurs dans $[0,1]$, tel que L_F soit de P-dimension finie V. Supposons de plus que $0 < \varepsilon < \frac{e}{2\log_2(e)} \backsimeq 0.94$. Alors :*

$$P(\Delta(\varepsilon, n, L_f)) \leq 8\left(\frac{16e}{\varepsilon}\ln(\frac{16e}{\varepsilon})\right)^V \exp(-n\frac{\varepsilon^2}{32})$$

avec $\Delta(\varepsilon, n, L_f))$ l'évènement $\exists L_f \in L_F, |L_f - \hat{L}_f| > \varepsilon$.
En outre, L_F est universellement Glivenko-Cantelli et universellement Donsker.

Théorème 1.8 (Vitesse de convergence dans les familles VC : cas des familles binaires) *Soit L_F un espace de fonctions définies sur $X \times Y$ à valeurs dans $\{0,1\}$, et supposons que $L_F \subset \{0,1\}^X$ soit de VC-dimension V. Alors V est fini si et seulement si L_F est universellement Glivenko-Cantelli, ou de manière équivalente si et seulement si L_F est universellement Donsker. Alors $P(\Delta(\varepsilon, n, L_F))$ est borné par :*

$$4\left(\frac{2en}{V}\right)^V \exp\left(-\frac{n\varepsilon^2}{8}\right) \tag{1.1}$$

$$8S(F,n)\exp\left(-\frac{n\varepsilon^2}{32}\right) \tag{1.2}$$

$$4\exp(4\varepsilon + 4\varepsilon^2)S(F,n^2)\exp(-2n\varepsilon^2) \tag{1.3}$$

$$16\left(\sqrt{n}\varepsilon\right)^{4096V}\exp(-2n\varepsilon^2) \quad \textit{avec} \quad n\varepsilon^2 > 64 \tag{1.4}$$

$$4\exp(4\varepsilon + \varepsilon^2 - 2n\varepsilon^2)\cdot n^{2V} \quad \textit{avec} \quad V \geq 2 \textit{ et } n \geq 4V \tag{1.5}$$

$$4\exp(4\varepsilon + \varepsilon^2 - 2n\varepsilon^2)\cdot (n+1)^2 \quad \textit{avec} \quad V = 1 \tag{1.6}$$

$$8n^V \exp\left(\frac{-1}{32}n\varepsilon^2\right) \quad \textit{avec} \quad V \geq 3 \textit{ et } n \geq 2V \tag{1.7}$$

$$8\left(n+1\right)^V \exp\left(\frac{-1}{32}n\varepsilon^2\right) \tag{1.8}$$

$$4\exp(4\varepsilon + \varepsilon^2 - 2n\varepsilon^2)\cdot \exp\left(\mathcal{H}(\frac{2V}{n})\right) \quad \textit{avec} \quad V \geq 2 \textit{ et } n > 4V \tag{1.9}$$

$$8\exp\left(n\cdot\mathcal{H}(\frac{V}{n})\right)\exp\left(\frac{-1}{32}n\varepsilon^2\right) \quad \textit{avec} \quad V \geq 3 \textit{ et } n > 2V \tag{1.10}$$

avec $\Delta(\varepsilon, n, L_f))$ l'évènement $\exists L_f \in L_F, |L_f - \hat{L}_f| > \varepsilon$.
et $\mathcal{H}(x) = -x\log(x) - (1-x)\log(1-x)$ (fonction d'entropie, prolongée par continuité en 0 et 1)

Dans la littérature, les résultats les plus utilisés sont 1.5 et 1.7 [Devroye et al.; 1997, p197]. Ils dépendent du nombre d'exemples (1.9 et 1.10 sont leurs raffinements mais moins connus et plus compliqués détaillés dans [Devroye et al.; 1997]). En pratique, les bornes 1.2 et 1.3 (tirées de [Devroye; 1982]) sont souvent les meilleures. 1.1 vient de [Vidyasagar; 2002, p207]. Signalons le cas de 1.4 ([Alexander; 1984] présente aussi dans [Devroye et al.; 1997, p207]) qui ne devient effective (*i.e* inférieure à 1) que pour $n >$

2^{6144}, limitant quelque peu l'intérêt pratique de cette borne... Cependant, il s'avère qu'il s'agit de la meilleure borne asymptotiquement. Elle permet de monter que :

$$E\left(\sup_{L_f \in L_F} |L_f - \hat{L}_f|\right) \leq \frac{8 + \sqrt{2048V \ln(4096V)}}{\sqrt{n}}$$

Ce théorème peut être adapté au cas de la minimisation du risque empirique et au cas des familles de fonction à valeurs réelles (bornées ou non).

1.4 Outils analytiques supplémentaires

1.4.1 Processus et Vecteurs Gaussiens

Dans toute la suite, t' désigne la transposée de t.

Définition 1.7 (Processus) *Soit (Ω, P) un espace de probabilité. On appelle processus aléatoire à valeurs dans $\mathcal{E}$ un terme $X = (\Omega, (X_n)_{n\geq 0}, P)$ où pour tout n, X_n est une variable aléatoire à valeurs dans $\mathcal{E}$.*

Un processus peut se voir comme une suite infinie de réalisations de variables aléatoires indicées par n (n'ayant pas nécessairement la même loi mais vivant dans le même espace). Par défaut, l'espace des valeurs $\mathcal{E}$ est $\mathbb{R}$ et on parlera de processus réel.

Définition 1.8 (Processus borné) *Soit $1 \leq p < \infty$. On dit que le processus réel X_n est borné dans L^p, si $\sup_n E(|X_n|^p) < \infty$.*

Un processus borné est parfois appelé fini ce qui peut porter à confusion car le terme « fini » ne se rapporte pas au fait que la suite comporte un nombre fini de termes. On utilisera aussi parfois le terme « presque sûrement borné dans L_p » la borne portant sur l'espérance de $|X^p|$. Par défaut quand on ne précisera pas p, on utilise L_∞.

Remarque :
Dans les définitions ci-dessus, le processus est à indices entiers. Plus généralement, il est possible d'indicer un processus par une famille de fonctions (non-nécessairement dénombrable, voir Def.1.9),

♦

Définition 1.9 (Processus indicé par F) *Soit F un ensemble quelconque (dans la suite de cette thèse, il s'agira en géréral d'un ensemble de fonctions) et (Ω, P) un espace de probabilités. On appelle processus aléatoire à valeurs dans $\mathcal{E}$ un terme $X = (\Omega, (X_f)_{f\in F}, P)$ où pour tout f, X_f est une variable aléatoire à valeurs dans $\mathcal{E}$.*

Définition 1.10 (Fonction caractéristique d'une variable aléatoire) *Soit X une variable aléatoire à valeurs dans $\mathbb{R}^d$ et de loi μ. On appelle fonction caractéristique (ou transformée de Fourier de μ) la fonction définie pour $t \in \mathbb{R}^d$ par*

$$\hat{\mu}(t) = \varphi_X(t) = E(e^{2\pi i<t,X>}) = \int_{\mathbb{R}^d} e^{2\pi i<t,x>}\, d\mu(x)$$

Si la v.a. X a une densité f, alors $\varphi_X(t) = \int f(x)e^{i<t,x>}\, dx$. Dans ce cas, la fonction caractéristique coïncide avec la transformée de Fourier de f. Notons que l'application $\mu \to \widehat{\mu}$ est injective ce qui permet d'identifier les variables aléatoires à partir de leur fonction caractéristique.

Définition 1.11 *On dit qu'une variable aléatoire réelle X est gaussienne si sa fonction caractéristique vérifie*

$$\varphi_X(t) = E[e^{2i\pi<t,X>}] = e^{2i\pi tm - 2\pi^2\sigma^2 t^2}$$

pour $m, \sigma \in \mathbb{R}$. On dit alors que X suit une loi normale $\mathcal{N}(m, \sigma^2)$.

Définition 1.12 *Un vecteur $X = (X_1, \ldots, X_n)$ de variables aléatoires est gaussien, si toute combinaison linéaire à coefficients réels des $X_1, \ldots, X_n$ est une variable aléatoire gaussienne réelle.*

Théorème 1.9 *Un vecteur aléatoire réel $X = (X_1, X_2, \ldots X_n)$ est gaussien si et seulement si sa fonction caractéristique est donnée par*

$$\varphi_X(t) = E[e^{2i\pi t'X}] = e^{2i\pi t' m_X - 2\pi^2 t'\Sigma_X t}, \qquad t \in \mathbb{R}^n$$

Où $m_X \in \mathbb{R}^n$ et $\Sigma_X \in \mathbb{R}^{n\times n}$ dénotent respectivement la moyenne et la matrice de covariance de X, On notera $X \sim \mathcal{N}(m_X, \Sigma_X)$.

Définition 1.13 (Processus Gaussien) *Un processus réel $(X)_t$ est dit gaussien si tout système fini extrait de $(X)_t$ est un vecteur aléatoire gaussien réel.*

1.4.2 Classes de Donsker et processus gaussiens [Adler; 1990]

Théorème 1.10 (Comportement asymptotique de $\hat{L} - L$) *Soit F une classe de fonctions $\mathcal{P}-$Donsker alors :*

$$\sqrt{n}\left(\hat{L}(f) - L(f)\right)_f \to PG$$

où PG est un processus gaussien borné indicé par F .

Ce théorème produit un résultat intéressant lorsqu'il est adjoint à l'inégalité de Borell :

Théorème 1.11 (Inégalité de Borell) *Celle-ci peut être trouvée dans [Van Der Vaart and Wellner; 1996, p438] mais la formulation utilisée ici vient de [Ledoux; 1988] Soit $(Y_t)_{t\in T}$ processus gaussien presque surement borné avec $\sigma(Y)^2$ la plus grande variance d'un Y_t pour $t \in T$, alors :*

$$\lim_{s\to\infty} \frac{\log P(\sup_t Y_t > s)}{s^2} = -1/(2\sigma(Y)^2)$$

et pour tout $\varepsilon > 0$, pour s assez grand, on a :

$$P(\sup_t Y_t > s) \leq \exp(\varepsilon s^2 - \frac{1}{2}s^2/\sigma(Y)^2)$$

Ce théorème montre que asymptotiquement $\sigma(Y)$ caractérise la largeur du *supremum* de Y_t. Ce résultat peut sembler contre intuitif puisque tout se passe comme si le processus gaussien n'avait qu'une seule valeur de t. Notons que la combinaison effectuée ici porte sur deux résultats asymptotiques ce qui n'est pas de très bon augure pour la pratique .

1.4.3 ε-nets (ε-squelettes)

Définition 1.14 (ε-nets à la Matoušek) *Soit X un ensemble, soit μ une mesure définie sur X. Soit F une collection de sous ensembles μ-mesurables pour X et $0 < \varepsilon < 1$.*
$A \subset X$ est un ε-net pour (X, F) si pour tout $S \in F$, $\mu(S) \geq \varepsilon$ implique que $A \cap S \neq \emptyset$

Intuitivement, cela ne revient pas à couvrir F mais simplement à ce que les éléments de F ayant une mesure plus grande que ε aient une intersection non nulle avec le réseau. Les ε-nets constituent un outil très apprécié en géométrie, il existe une application à l'apprentissage dans le cadre de la VC-théorie :

Théorème 1.12 (VC-dim et ε-nets à la Matoušek) *Soit (X, F, μ) un système μ-mesurable tel que la dimension de VC de F est bornée par d (avec $d > 2$). Soit $k \geq 2$ un paramètre entier, alors il existe un $(1/k)$-net de taille $O(dk \log k)$ pour (X, F, μ)*

Ce résultat prouvé par Matoušek dans [Matousek; 2002] utilise des résultats de Mulmuley dans le cas de d'espace de dimension finie. L'utilité d'un tel théorème est de permettre de prouver des résultats de convergence en utilisant un ε-net.

Classiquement en apprentissage, la définition la plus utilisée pour les ε-net est la suivante [Devroye et al.; 1997] :

Définition 1.15 (ε-nets) *Soit X un ensemble normé dont nous noterons $||.||_X$ la norme sur X, et soit F un sous ensemble de X et ε>0. On dira que $A \subset X$ est un ε-net de F si tous les éléments de A sont de norme finie et que $F \subset \bigcup_{a\in A} B(a, \varepsilon)$ (ou $B(x, \varepsilon)$ désigne la boule de centre x et de rayon ε pour la norme $||.||_X$).*

Cette définition est celle que nous utiliserons dans la suite de ce document. Elle est plus restrictive que la précédente. Avec celle ci, si on dispose d'un recouvrement de F par des boules de rayon ε, alors l'ensemble constitué des centroïdes de ces boules est un ε-net pour F dans X. Essentiellement on se sert des ε-nets pour remplacer un ensemble trop difficile à traiter par un autre ensemble plus simple en étant sur que tous les éléments de l'ensemble de départ ne sont pas à une distance supérieure à ε du nouvel ensemble.

1.4.4 Nombre de couverture et *Bracketing* nombre de couverture

Dans un ensemble normé X, la taille minimale d'un ε-net de F est appelée ε-**nombre de couverture** de F noté $N(\varepsilon, F)$. Si pour un ε il est impossible de trouver un réseau fini on dira que le nombre de couverture correspondant à ε est infini.F est dit totalement borné si pout tout $\varepsilon > 0$, $N(\varepsilon, F) < \infty$.

On appelle ε-entropie de F le logarithme de $N(\varepsilon, F)$.

Bien qu'il n'apparaisse pas dans la notation, il faut bien préciser X. En effet, les centres des boules des ε-nets sont dans X et non dans F, ce qui peut parfois changer les résultats.

Étant données deux fonctions b et h de X dans $\mathbb{R}$, on appelle **segment** $[b, h]$, l'ensemble des applications f de X dans $\mathbb{R}$ telles que $||b||_X \leq ||f||_X \leq ||h||_X$.

Ce segment est dit ε-**segment** si $||b - h||_X \leq \varepsilon$ et que b et h ont des normes finies.

Le ***Bracketing nombre de couverture*** [Van Der Vaart and Wellner; 1996, p83] noté $N_{[]}(\varepsilon, F)$ est le nombre minimal d'ε-segments nécessaires pour recouvrir F. b et h ne sont pas nécessairement des fonctions de F mais sont supposées être de norme finie. Comme précédemment on appelle $\varepsilon_{[]}$-entropie le logarithme de $N_{[]}(\varepsilon, F)$.

1.5 Bornes classiques

Cette section rappelle quelques inégalités classiques portant sur la déviation de la moyenne d'une variable aléatoire par rapport à son espérance.

1.5.1 Espaces préhilbertiens

Célèbre inégalité, ancienne mais néanmoins très utile :

Théorème 1.13 (Inégalité de Cauchy-Schwartz) *Dans un espace préhilbertien complexe E (muni de son produit scalaire $< ., . >$)*

$$\forall (x, y) \in E^2 \mid < x, y > \mid \leq \|x\|.\|y\|$$

L'égalité arrive si et seulement si x et y sont liés ($\exists \alpha \in \mathbb{R}$ tel que $x = \alpha y$).

Dans les espaces L_p (et l_p), l'inégalité de Cauchy-Schwartz est un cas particulier de l'inégalité de Hölder (avec $p = q = 2$) :

Théorème 1.14 (Inégalité de Hölder pour les suites) *Soient x, y deux vecteurs correspondant à une suite respectivement dans l_p et l_q alors :*

$$\textit{Si } \frac{1}{p} + \frac{1}{q} = 1,\ p, q > 0 \textit{ alors } | < x, y > | \leq || x ||_p || y ||_q$$

Cette inégalité est particulièrement utilisée dans le cas des produits scalaires définis par des intégrales dans les espaces L_p. Si une fonction f est dans $L_p(X)$, alors la norme L_p de f est notée $|| f ||_p$. Soit$(X, \mathfrak{B}, \mu)$ un espace mesuré. Si f est dans $L^p(X)$ et g est dans $L^q(X)$ (avec $1/p + 1/q = 1$), alors l'inégalité de Hölder devient :

$$\|fg\|_1 = \int_X |fg| \mathrm{d}\mu \leq \left(\int_X |f|^p \mathrm{d}\mu \right)^{\frac{1}{p}} \left(\int_X |g|^q \mathrm{d}\mu \right)^{\frac{1}{q}} = \|f\|_p \, \|g\|_q$$

1.5.2 Inégalités à la Bienaymé-Tchebychev

Théorème 1.15 *Soit $r > 0$ un nombre réel, et X une variable aléatoire réelle définie sur un espace probabilisé . On suppose que $E(|X|^r)$ est fini (on dira que X admet un moment d'ordre r ou encore que X est r-intégrable). Alors on a :*

$$P(|X| \geq t) \leq \frac{E(|X|^r)}{t^r}$$

Pour $r = 1$ (resp. $r = 2$), ce résultat est connu sous le nom d'inégalité de Markov (reps. d'inégalité de Bienaymé-Tchebychev). Il s'agit des premières inégalités pouvant servir à évaluer des déviations par rapport à la moyenne. Malheureusement elles ne présentent pour ainsi dire pas d'intérêt pratique : les bornes obtenues sont extrêmement lâches à cause des faibles hypothèses faites sur X. En effet, on suppose simplement l'existence d'un moment d'ordre r.

1.5.3 Le théorème de la limite centrée (TCL)

La plus célèbre des inégalités de déviation est donnée par le théorème suivant qui éclaircit la loi des grands nombres dans le cas de variables i.i.d. possédant un moment d'ordre 2 :

Théorème 1.16 (TCL) *Soit $X_1, \ldots, X_n$ une suite de variables aléatoires indépendantes identiquement distribuées d'espérance m et de variance σ^2 bornée (existence d'un moment d'ordre deux). Alors :*

$$\frac{1}{\sigma\sqrt{n}} \sum_{i=1}^{n} (X_i - m) \overset{n\to\infty}{\rightsquigarrow} \mathcal{N}(0,1)$$

ou $\rightsquigarrow$ signifie «converge en loi vers»

La signification intuitive de ce théorème est que les sommes de variables aléatoires i.i.d. possédant un moment d'ordre deux ont toujours tendance à se comporter de manière gaussienne. Ce résultat étant vrai quelque soit la forme de la distribution, il est très exploité. De plus, en pratique, le théorème possède une certaine forme de résistance qui permet de l'utiliser même dans le cas de variables ne possédant pas la même moyenne ou écart type. En résumé, les sommes de v.a. indépendantes ont tendance à donner des gaussiennes ce qui permet d'assimiler une somme de variables inconnues (voire inobservables) en une unique gaussienne dont on se ramène à évaluer les paramètres. On trouve une exploitation originale de cette idée dans l'analyse en composantes indépendantes (ICA, [Hyvärinen et al.; 2001]).

Dans le cas vectoriel, on obtient :

Théorème 1.17 (TCL pour les vecteurs) *Soit X_n une suite de v.a. à valeurs dans $\mathbb{R}^d$, indépendantes, de même loi, de carré intégrable, ayant pour moyenne m, et pour matrice de covariance Σ à coefficients finis. On pose $S_n = X_1 + \ldots, X_n$. Alors $W_n = \frac{1}{\sqrt{n}}(S_n - nm)$ converge en loi vers $\mathcal{N}_d(0, \Sigma)$.*

1.5.4 Théorèmes *central limit* fonctionnels

Essentiellement, on considère un échantillon aléatoire simple $x_1, \ldots, x_n$. On dispose d'un espace de fonctions F, qui dans un premier temps est supposé de cardinal d fini. On considère les moyennes de $(f_1, f_2, f_3, \ldots, f_d)$ ($f_i \in F$) sur cet échantillon : $(M_1, M_2, \ldots, M_d)$. On a donc $M_i = (f_i(x_1) + f_i(x_2) + \ldots + f_i(x_n))/n$ pour $i \in [\![1, d]\!]$.

Soit E_i la limite de M_i quand n tend vers l'infini. On note $(D_1, \ldots, D_d)$ le vecteur des écarts à la moyenne défini par $D_i = M_i - E_i$.

La loi forte des grands nombres nous dit que $D = (D_1, \ldots, D_d)$ tend presque sûrement vers $(0, 0, \ldots, 0)$, sous des hypothèses très légères. Le théorème central limite multi-varié nous fournit la limite de $G = \sqrt{n} \cdot (D_1, \ldots, D_d)$. L'extension de ces résultats au cas où F est un espace fonctionnel de cardinal infini, est ce que l'on appelle un théorème *central limit* fonctionnel.

Le cas des classes de Glivenko-Cantelli

Théorème 1.18 (Convergence pour les classes de Glivenko-Cantelli) *Soit $x_1, \ldots, x_n$, n réalisations indépendantes d'une variable aléatoire X possédant un moment d'ordre 1. Dans le cas où F est Glivenko-Cantelli,*

$$\forall \varepsilon > 0,\ P\left(\text{limsup}_{n\to\infty} \sup_{f\in F} \left|\sum_{i=1}^{n} \frac{f(x_i)}{n} - E(f(X))\right| > \varepsilon\right) = 0$$

Ainsi sous l'hypothèse où F est Glivenko-Cantelli, le vecteur précédent D (le vecteur des écarts à la moyenne) tend presque surement vers le vecteur nul.

Notons que ce vecteur peut être infini (il est indexé par un ensemble de fonctions). C'est d'ailleurs tout l'intérêt des classes de Glivenko-Cantelli qui désignent des ensembles de fonctions F infinis pour lesquelles on peut étendre la loi forte des grands nombres (si $Card(F)$ était fini, Kolmogorov s'appliquerait directement).

Notons cependant que l'on ne peut pas ici parler de théorème *central limit* fonctionnel puisqu'il s'agit en fait d'une convergence presque sûre, dont on ne peut pas borner la vitesse de convergence. On ne dispose pas pour cette famille de fonctions de borne sur la vitesse de convergence. En fait, par définition, le cas des classes de Donsker est celui ou il existe un théorème *central limit* fonctionnel.

Le cas des classes de Donsker

En revanche, de telles bornes sont disponibles sous l'hypothèse plus forte où F est une classe de Donsker. Dans ce cas, le vecteur G des écarts à la moyenne multiplié par $\sqrt{n}$, tend vers un processus gaussien. Nous proposerons deux énoncés de théorème limite fonctionnel également connus sous le nom de théorèmes de Donsker [Van Der Vaart and Wellner; 1996, p171-172(Th2.8.3-2.8.4)].

Théorème 1.19 (Entropie universelle) *Si la classe F est mesurable, dont on possède une enveloppe G supposée L_2 :*

$$\int_0^\infty \sup_Q \sqrt{\log N(\varepsilon ||G||_{Q,2}, F, L_2(Q))} d\varepsilon < \infty \text{ avec } ||f||_{Q,2} = \sqrt{E_Q(|f|^2)}$$

*Avec Q parcourant toutes les mesures de probabilités discrètes finies et où $N(\varepsilon, F, L_2(Q))$ désigne le ε-nombre de couverture de F pour la norme $L^2(Q)$. Si de plus les classes $\mathcal{F}_\delta = \{f - g | f, g \in \mathcal{F}, ||f - g||_{P,2} < \delta\}$ sont mesurables pour $\delta \in \mathbb{R}^{*+}$ alors pour tout $f \in \mathcal{F}$, la suite des processus empiriques $Z_n = \frac{1}{\sqrt{n}}(f(X_i) - E(f))$ converge faiblement dans l_∞ vers un borélien tendu. De plus le $\sup$ de la différence est d'ordre $O(1\sqrt{n})$ (c'est-à-dire que F **est Donsker pour** P).*

Si de plus les processus étaient gaussiens alors le processus limite l'est aussi.

Théorème 1.20 (Bracketing entropie plus entropie) *Si F enveloppe de $\mathcal{F}$ possède un moment d'ordre deux fini, et que :*

$$\int_0^\infty \sqrt{\log N_{[]}(\varepsilon, \mathcal{F}, L_2(P))} d\varepsilon < \infty$$

ou $N_{[]}(u, \mathcal{F}, L_2(P))$ désigne le ε-braketing nombre de couverture de F pour la norme L^2, alors $\mathcal{F}$ est P-Donsker.

Ces théorèmes un peu rébarbatifs (dont le second est aussi connu sous le nom de théorème d'Ossiander) signifient que pour les classes de Donsker le processus limite G (différences aux moyennes fois $\sqrt{n}$) tend vers un processus gaussien dont on connaît la covariance ; ceci correspond intuitivement à la généralisation d'une loi normale au cas infini.

Intuitivement, si l'entropie (logarithme du nombre de couverture) est en $O(1/\varepsilon^a)$ avec $a < 2$ alors la convergence de l'erreur empirique vers l'erreur généralisation est en $1/\sqrt{n}$, donc du **même ordre de grandeur que le cas de la VC-dimension finie**[1]. De plus contrairement à la VC-dimension qui impose que les nombres de couverture soient polynomiaux, ce résultat permet de traiter des **nombres de couverture exponentiels** (tant que le $\log$ est polynomial de degré inférieur à 2).

1.5.5 Chernoff

Théorème 1.21 (Inégalités de Chernoff) *[Vidyasagar; 2002, p22] Soient $X_1, \ldots, X_n$ une suite de variables aléatoires indépendantes identiquement distribuées valant 1 avec une probabilité p et 0 sinon. Notons $\bar{X}$ la moyenne des X_i. Alors pour tout ε positif :*

$$P(\bar{X} \geq p + \varepsilon) \leq e^{-2n\varepsilon^2} \tag{1.11}$$

$$P(\bar{X} \leq p - \varepsilon) \leq e^{-2n\varepsilon^2} \tag{1.12}$$

$$P(|\bar{X} - p| \geq \varepsilon) \leq 2e^{-2n\varepsilon^2} \tag{1.13}$$

$$P(\bar{X} \geq (1+\gamma)p) \leq e^{-\gamma^2 np/3} \quad \textit{avec} \quad \gamma \in [0,1] \tag{1.14}$$

$$P(\bar{X} \geq (1-\gamma)p) \leq e^{-\gamma^2 np/3} \quad \textit{avec} \quad \gamma \in [0,1] \tag{1.15}$$

$$P(\bar{X} \leq r) \leq e^{-\frac{n(p-r)^2}{2p(1-p)}} \quad \textit{avec} \quad p \leq 0.5 \textit{ et } r \leq p \tag{1.16}$$

Les inégalités 1.11, 1.12, 1.13 sont appelées **bornes additives** de Chernoff. 1.14 et 1.15 sont les **bornes multiplicatives**. L'inégalité 1.16 connue sous le nom de Chernoff-Okamoto qui est plus fine que celle de Chernoff surtout pour les petites valeurs de p, elle figure [Vidyasagar; 2002, p24].

Quand p est petit, les bornes multiplicatives sont meilleures car ε évolue en $1/n$ au lieu de $1/\sqrt{n}$.

[1] Excepté dans le cas de l'erreur nulle où la convergence est en $1/n$

On pourra préférer une autre formulation de ce théorème :

Théorème 1.22 (Chernoff non-i.i.d. pour des variables bornées) *[Levchenko; 2005] Soit $X_1, \ldots, X_n$ des variables aléatoires indépendantes discrètes telles que pour tout i*
- $E(X_i) = 0$
- $|X_i| \leq 1$

Alors en posant $X = \sum_{i=1}^{n} X_i$ et $\sigma^2 = Var(X)$ on a :

$$\forall \lambda \in [0; 2] \; P(|X| \geq \lambda\sigma) \leq 2e^{-\lambda^2/4}$$

Cette inégalité s'applique à des variables non identiquement distribuées, ce qui est fort pratique. La contrainte supplémentaire sur λ n'est pas gênante car l'inégalité ne nous intéresse en général que pour des petites valeurs de λ.

Ce théorème peut être étendu au cas de variables continues. En fait il est surtout important de retenir que les bornes de Chernoff permettent d'évaluer la probabilité de déviation à la moyenne dans le cas de variables binaires de manière bien plus fine que Bienaymé-Tchebychev, grâce à des hypothèses plus fortes (les x_i sont bornées et non plus simplement possédant un moment d'ordre r).

1.6 Bornes à la Talagrand

A la suite de la VC-théorie qui a mis en évidence l'intérêt de bornes exponentielles non-asymptotiques pour le $\sup$ de processus empirique, une bande de mathématiciens ont tenté de généraliser les bornes exponentielles classiques (Hoeffding, Bernstein, Bennett, Chernoff). Ils ont alors développé un domaine en plein essort : les inégalités de concentration. Je vais présenter rapidement les grandes idées de cette mouvance mais je ne suis pas extrêmement familier de ces techniques.

L'intuition du phénomène de concentration de la mesure dans les espaces produits est la suivante : une variable aléatoire Z fonction de nombreuses variables aléatoires indépendantes $X = (X1, \ldots, Xn)$ (c'est-à-dire $Z = f(X)$) qui ne dépend pas trop de chacun de ces arguments ne peut pas s'écarter beaucoup de sa moyenne.Les inégalités de concentration ont pris de plus en plus d'importance ces dernières années ; on peut les voir comme des extensions des inégalités exponentielles classiques

Bref, une inégalité de concentration dans les meilleurs cas est une version de l'inégalité de Bernstein pour les sommes de variables bornées. Ce qui produit des inégalité du type :

$$P(Z - E[Z] > t) \leq \exp(-\frac{t^2}{2(v + t)}),$$

où v est un majorant raisonnable de la variance de Z. On vise à traiter des fonctions compliquées de variables indépendantes (ou faiblement dépendantes). L'une des grandes

forces de ces inégalités de concentration par rapport aux probabilités classiques, provient de l'existence de résultats non-asymptotiques et qu'elles permettent de dire des choses non-triviales sur des événements de très faible probabilité.

La dérivation d'inégalités de concentration a suscité de nombreux travaux ces vingt-cinq dernières années. Les principales avancées sont dûes à Talagrand ([Talagrand; 1995], [Talagrand; 1996]). Ses méthodes, proviennent de la géométrie et de l'analyse fonctionnelle. Au dire des initiés, les résultats sont simples mais nécessitent une excellente connaissance de la géométrie des espaces de Banach (ce qui n'est pas mon cas).

Une manière plus facile d'aborder ces résultats consiste à utiliser les techniques de martingales (un processus réel adapté indicé par n est une martingale si pour tout n, X_n est intégrable et avec probabilité 1 : $\mathbb{E}(X_{n+1}|\mathcal{F}_n) = X_n$ - en remplaçant $=$ par $\geq$ ou $\leq$ on obtient respectivement une sous-martingale et une sur-martingale). Les variables Z qui nous intéressent peuvent toujours se représenter comme des sommes d'accroissements de martingales (c'est le plongement de Doob), si ces incréments sont contrôlés, alors les fluctuations de Z sont liées au comportement de la « variation quadratique prévisible » de la martingale. Cette approche en théorie équivalente et plus facile à comprendre reste cependant difficile à exploiter. Les inégalités les plus puissantes (celles qui portent sur la sur-martingale exponentielle et qui sont connues depuis plus de vingt ans) n'ont été redécouvertes que très récemment [Dembo and Zeitouni; 1998].

Il existe deux approches alternatives aux méthodes déployées par Talagrand. Les méthodes de transport proposées par [Marton; 1996] et la « méthode entropique » proposée par [Ledoux; 2001]. Cette dernière trouve son origine dans la construction d'inégalités de Sobolev logarithmiques. Cette méthode s'est révélée très prometteuse : elle a permis de construire des preuves simples de résultats très difficiles de Talagrand sur les suprema de processus empiriques (voir les travaux de M. Ledoux, P. Massart, E. Rio et O. Bousquet). Ces travaux ont abouti à une version optimale de l'inégalité de Bennett pour les suprema de processus empiriques [Bousquet; 2002].

1.7 Apprendre à partir de plusieurs experts

Dans cette section uniquement, afin d'alléger les notations, on se place dans le cadre de l'apprentissage supervisé à deux classes $\{-1, 1\}$.

Intuitivement pour faire de la classification, il semble intéressant d'essayer plusieurs approches et d'être d'autant plus confiant dans les résultats que les différentes techniques donnent des résultats identiques. En développant cette idée, il parait séduisant d'essayer de faire voter des systèmes d'appentissage différents (désignés sous le nom d'experts). Dans les premières formulations dues à Warmuth, Lugosi, Cesabianchi, l'apprentissage dit actif procède en itérant deux étapes :

- Le diagnostique sur l'exemple courant, se fondant sur les poids courants des hypothèses en cours (des experts),
- La mise à jour des poids des experts en fonction de leur performance sur le cas

courant

Le critère d'évaluation essentiel est le rejet, défini par la perte de performance de la combinaison pondérée des experts, par rapport au meilleur expert. Cette approche par système de votes conduira plus tard à l'apprentissage dit d'ensemble [Brown et al.; 2005].

Le problème quand on dispose de plusieurs experts, c'est que l'on ne dispose pas forcément de mesure de qualité sur chacun d'eux. Si on dispose de n experts, on se retrouve avec n prédictions pour la classe qui ne sont pas nécessairement concordantes. L'idée de base est de faire voter les experts et de donner comme réponse la classe majoritaire. Cependant si tous les experts sont mauvais sauf un, celui-ci n'aura jamais voix au chapitre. Pour éviter cela, on attribue une confiance (aussi appelée poids, noté w_i pour le $i^{\text{ème}}$ expert) qui est régulièrement réactualisée pour chacun des experts. Le but de tels algorithmes est d'avoir une performance au moins égale à celle du meilleur de nos experts.

Ce type d'apprentissage peut-être considéré comme précurseur de l'apprentissage actif car on se place ici dans le cadre d'une utilisation en ligne de nos experts. Remarquons tout de même qu'ici, l'algorithme n'a pas accès aux données, il doit se contenter de la classification donnée par les experts. En particulier, il ne peut pas influencer la distribution des exemples, donc ce n'est pas un algorithme «actif». Dans ce contexte, il n'est déjà pas facile d'avoir un taux d'erreur égal au meilleur des experts. On commence par réaliser la phase d'apprentissage des différents experts, puis on leur présente de nouvelles données à classer. Selon la qualité des vote, les poids des experts sont rectifiés à la volée.

1.7.1 Première version

Dans un premier temps on se placera dans le cadre de la logique booléenne. C'est à dire que les attributs sont booléens et le concept cible est une formule logique formée à partir de ces attributs. Généralement dans ce cadre on considérera que le concept cible est une disjonction ; en effet on dispose déjà d'experts ayant donné leur avis et on souhaite choisir à notre tour, il n'est donc pas forcément pertinent de considérer des conjonction d'experts. Une base de n exemples est la donnée de n interprétations et de leur valeur de vérité.

Théorème 1.23 (Majorité pondérée [Littlestone and Warmuth; 1989]) *Le nombre d'erreurs M faites par l'algorithme de la majorité pondérée (Algo. 1) n'est jamais plus de $2.41(m + \log(n))$, où m est le nombre d'erreurs faites par le meilleur expert.*

L'algorithme de la majorité pondérée aléatoire

Cet algorithme peut être amélioré en modifiant :
– Le mécanisme de mise à jour des poids : au lieu de multiplier par $1/2$ le poids des experts ayant commis une erreur, on le multiplie par β (qui sera à choisir par

Algorithme 1 Algorithme de la majorité pondérée

1: Initialiser tous les w_i à 1
2: **Boucler**
3: A l'arrivée d'un nouvel exemple, noter $\{f_1, \ldots, f_n\}$ les prédictions des experts sur cette donnée. Renvoyer la prédiction pour laquelle la somme des poids des experts est la plus grande. C'est-à-dire renvoyer 1 si

$$\sum_{i \backslash f_i = 1} w_i \leq \sum_{i \backslash f_i = -1} w_i$$

et -1 sinon.
4: Regarder la bonne réponse et diviser par 2 le poids de tous les experts ayant fait une erreur.
5: **Fin Boucle**

l'utilisateur).

– L'interprétation du vote pondéré en termes de probabilistes incorporant la capacité de s'abstenir au lieu d'un mécanisme de majorité.

Algorithme 2 Algorithme de la majorité pondérée aléatoire

1: Initialiser tous les w_i à 1
2: **Boucler**
3: A l'arrivée d'un nouvel exemple, noter $\{x_1, \ldots, x_n\}$ les prédictions des experts sur cette donnée. Renvoyer la prédiction x_i avec probabilité $w_i / \sum_i w_i$.
3: Regarder la bonne réponse et multiplier par β le poids de tous les experts ayant fait une erreur.
4: **Fin Boucle**

Théorème 1.24 (majorité pondérée aléatoire [Littlestone and Warmuth; 1989]) *Sur n'importe quelle séquence de test, le nombre d'erreurs M fait par l'algorithme aléatoire de la majorité pondérée (Alg. 2) est tel que :*

$$E(M) \leq \frac{m \ln(1/\beta) + \ln(n)}{1 - \beta}$$

où m est le nombre d'erreurs faites par le meilleur des experts.

Intuitivement, l'avantage de la majorité pondérée aléatoire est clair dans les situations proches de l'*ex-aequo* ; en effet, si l'erreur domine d'une très courte majorité, le diagnostic est erroné avec une probabilité 1 dans le cas non-aléatoire. Alors que dans le cas aléatoire, il est erroné avec seulement une probabilité proche de 50%. En cas d'erreur, le poids total sera ensuite environ divisé par $1/4$ (qui est une grande valeur pour une prédiction correcte), et on peut montrer que la probabilité de faire une prédiction correcte est fortement liée à la décroissance du poids total (le poids décroît pour les erreurs).

De plus en ajustant β, on peut faire tendre le coefficient devant m vers 1. Au prix cependant de l'élévation d'une constante additive.

1.7.2 Winnow

Restons pour le moment dans le cadre de la logique propositionnelle (avec le concept cible étant une formule logique des attributs). Les résultats proviennent de [Blum; 1998] bien que les auteurs initiaux soient Littlestone, Long et Warmuth en 92.

Algorithme 3 Winnow

Initialiser les poids $w_1, \ldots, w_n$ à 1
Boucler
  Étant donné un exemple $x = (x_1, \ldots, x_n)$ où x_i désigne la valeur du $i^{\text{ème}}$ attribut (0 ou 1) pour l'exemple x répondre 1 si

$$w_1 x_1 + \ldots + w_n x_n \geq n$$

  et -1 sinon.
  Si l'algorithme fait une erreur **Alors**
    Si l'algorithme a prédi -1 au lieu de 1 **Alors**
      alors pour chaque x_i valant 1, doubler w_i
    FinSi
    Si l'algorithme a prédi 1 au lieu de -1 **Alors**
      alors pour chaque x_i valant 1, diviser w_i par deux.
    FinSi
  FinSi
Fin Boucle

Théorème 1.25 (Erreurs de Winnow) *Dans le cas où le concept cible est une disjonction de r variables alors l'algorithme Winnow (Algo. 3) commet au plus $2 + 3r(1 + \log n)$ erreurs.*

On peut de plus faire la même amélioration que précédemment (*randomization* de winnow) en voyant les poids comme des probabilités de choix. Dans le cas où le concept cible n'est pas une disjonction, le comportement de winnow est connu : pour une disjonction c soit :

- Notons m_c le nombre d'erreurs faites par cette disjonction sur les exemples ;
- A_c le nombre d'erreurs sur les attributs, défini comme suit. Pour chaque exemple positif qui ne satisfait aucune variable de c ajouter 1 à A_c. Pour chaque exemple négatif satisfaisant k variables de c, ajouter k à A_c.

Théorème 1.26 (Erreurs de Winnow randomizé) *étant donné r, il est possible de trouver un algorithme de winnow aléatoire tel que pour n'importe quelle séquence d'exemples*

et n'importe quelle disjonction de r variables, l'espérance du nombre d'erreurs faites est :

$$A_c + (2 + o(1))\sqrt{A_c r \ln n/r}$$

puisque $A_c/(r\ln(n/r)) \to 0$

1.7.3 *Bagging*

Une grande part des difficultés dans ce cadre provient de la présence d'experts potentiellement mauvais. C'est pour cette raison qu'un vote simple (non-pondéré) ne présente aucune garantie de performance.

Toutefois dans le cas où la qualité des experts peut être encadrée, des résultats plus puissants sont obtenus, c'est notament le cas du *bagging* défini par [Breiman; 1996]. Il s'agit également de vote d'experts mais le *bagging* s'inscrit dans un cadre plus large que l'apprentissage actif.

Tout repose sur le fait qu'il est souvent raisonnablement aisé d'obtenir des classifieurs faibles (*weak learners*), *i.e.* des hypothèses dont l'erreur est inférieure à 50% mais dont le taux d'erreur est tout de même jugé insatisfaisant. Le *bagging* correspond à l'exploitation de classeurs faibles **indépendants**.

En effet, comme chaque classeur prédit correctement la classe de plus de 50% des exemples, si les erreurs sont uniformément réparties alors en **moyenne pour chaque donnée à classer il y aura plus de 50% des classifieurs donnant une réponse correcte.**

Fondamentalement, ce sont les inégalités de Chernoff (voir 1.5.5) qui permettent de borner le taux d'erreur produit par le bagging en fonction du nombre de *weak learners* et de leurs performances individuelles. Il est très important de comprendre qu'ici tout repose sur l'indépendance entre les votants. En effet si tous commettent leurs erreurs dans les mêmes zones il n'y a rien à attendre du *bagging* [Esposito and Saitta; 2004].

Cependant, on notera que dans la pratique cette hypothèse d'indépendance est souvent violée ne serait-ce que parceque les classeurs appris dépendent du même jeu de données d'apprentissage. Pour remédier à cette limitation, plusieurs méthodes sont utilisées :

- Le sous échantillonnage des données (comme en *cross-validation*).
- Le bruitage aléatoire des données

La condition essentielle est celle de l'instabilité de l'algorithme d'apprentissage, garantissant que des hypothèses suffisamment diversifiées sont obtenues à partir de faibles modifications de l'ensemble d'apprentissage. Les différents classifieurs obtenus pourront donc se comporter comme étant plus ou moins indépendants. (Pour davantage de détails, on se réfèrera à [Grandvalet; 2004]).

1.7.4 *Boosting*

Le *boosting* est comme le *bagging* un méta-algorithme visant à améliorer les résultats d'apprentissage d'un algorithme produisant des *weak-learner* de façon suffisament instable [Schapire; 1990].

Cette fois, on suppose que l'algorithme d'apprentissage est capable de prendre en compte une pondération des exemples, la stratégie est globalement la suivante : On exécute K fois l'algorithme d'apprentissage sur différentes distributions de probabilités P_i des exemples de l'échantillon d'apprentissage. On n'obtient pas de nouveaux exemples à chaque itération, on se contente en fait de perturber la distribution en modifiant le poids des exemples de la base d'apprentissage pour favoriser telle ou telle région. En fait, à chaque itération du *boosting*, on augmente le poids des exemples ayant été mal classés.

Les bonnes propriétés pratiques du *boosting* sont restés pendant longtemps très mystérieuses. En effet, le *boosting* a une erreur empirique sur l'échantillon d'apprentissage qui décroît exponentiellement et parallèllement semble ne pas rencontrer de problèmes de sur-apprentissage (sauf dans le cas où la base d'apprentissage est bruitée où les *outliers* sont surpondérés). Aujourd'hui il existe de nombreux théorèmes relatifs aux marges ou à la convergence du *boosting* et des travaux tendent à montrer que le *boosting* possède des liens avec la maximisation du niveau de confiance dans la prédiction de la classe d'un exemple (marge) [Rudin et al.; 2004] [Rudin et al.; 2005].

Le *boosting* présente des liens avec l'apprentisage actif, puisqu'il s'agit dans les deux cas d'influer sur la distribution des exemples. Je pense cependant qu'il s'agit d'un cadre un peu plus restrictif car la base de données utilisée est fournie par la distribution naturelle des exemples, ce n'est que par la suite qu'elle est influencée par la pondération choisie.

1.8 Suites à Faible Discrépance

Dans cette section nous allons sortir du cadre de l'apprentissage pour introduire des notions provenant du calcul numérique des intégrales. Nous y définirons en particulier ce qu'est une suite à faible discrépance et quel est sont intérêt. Ces notions seront ensuite reprises lors de la présentation des contributions de cette thèse dans la partie 3. Cette notion de discrépance est aussi utilisée dans les algorithmes évolutionnaires depuis peu par [Kimura and Matsumura; 2005].

Les suites à faible discrépance sont initialement apparues dans le domaine du calcul intégral. En effet, l'intégrale d'une fonction f sur un domaine D étant la moyenne de f sur D, une astucieuse technique d'évaluation est de tirer aléatoirement (de manière i.i.d. et uniforme) une suite de points $x_1, \ldots, x_n$ dans D et d'évaluer $\int_D f$ par $\frac{1}{n}\sum_{i=1}^{n} f(x_i)$. En effet, les $f(x_i)$ sont en fait des variables aléatoires (puisque les x_i sont aléatoires i.i.d.) dont la moyenne est $\int_D f$. On évalue ensuite l'erreur commise dans l'approximation de l'intégrale en utilisant les théorèmes d'écart à la moyenne d'une variable aléatoire

(théorème central limite par exemple). Cette technique est connue sous le nom de calcul d'intégrale à la Monte-Carlo en référence au célèbre casino.

L'idée des suites à faible discrépance est de remplacer le générateur de nombres aléatoires. En effet dans les techniques de calcul de type Monte-Carlo (pour l'évaluation d'intégrales par exemple), il peut être intéressant de remplacer le tirage aléatoire par une suite de point plus « ordonnée» [Millet], [Niederreiter; 1992]. Par « ordonnée », on entend essentiellement plus uniformément répartie. En effet, il est bien connu qu'une suite aléatoire uniformément répartie présente souvent des sortes de points d'accumulation (figure 1.2). Attention tout de même, cette intuition peut se révèler trompeuse car nous verrons que des points placés sur une « grille » uniforme ne présentent pas la plus faible discrépance possible (à cause du placement des bords de l'objet nous intéressant qui peuvent mal tomber par rapport à la grille).

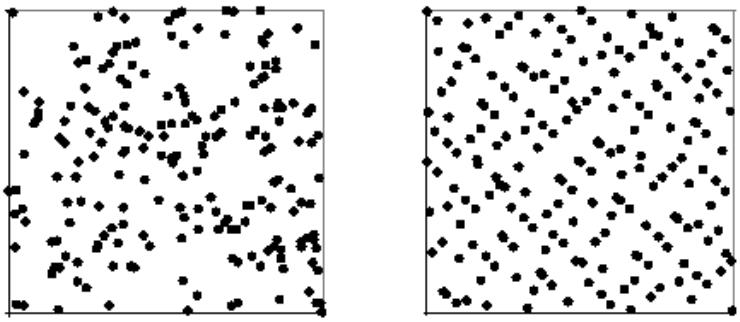


FIG. 1.2 : *Exemple en dimension deux de points tirés uniformément (à gauche) et selon un générateur de suites à faible discrépance (à droite). Il est amusant de constater qu'un humain n'étant pas habitué à voir des points tirés uniformément tend à dire que le tirage uniforme est à droite...*

1.8.1 Suites équiréparties

Définition 1.16 *Une suite $x_1, \ldots, x_n$ est équirépartie dans $[0,1]^d$ si pour tout $a = (a_1, \ldots, a_d) \in [0,1]^d$,*

$$\lim_n \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}_{\{x_k \in [0,a]\}} = \prod_{i=1}^{d} a_i$$

la discrépance de la suite (x_i) de $[0,1]^d$ est définie par :

$$D_n^*(X) = \sup_{a \in [0,1]^d} \left| \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}_{\{x_k \in [0;a]\}} - \prod_{i=1}^{d} a_i \right|$$

Le but est de mesurer si la suite X_i est répartie proportionnellement à l'aire des hyperrectangles de $[0,1]^d$. Le terme $\prod_{i=1}^{n} a_i$ correspond à un volume. La discrépance d'une suite équirépartie converge vers 0. Ce qui signifie qu'une suite est équirépartie si la fraction de ses points contenus dans un hyperrectange R converge vers le volume de R. Une faible

discrépance correspond en fait à une répartition des points uniformément dans **tous** les hyper-rectangles de $[0,1]^d$.

1.8.2 Variation au sens de Hardy et Krause

Cette section définit d'après [Owen; 2004] les variations associées à une fonction f, quantifiant intuitivement sa régularité. Le lecteur peut dans un premier temps sauter cette section et passer directement à la section 1.8.3, il suffira de retenir que l'on définit $V(f), V_{[a,b]}(f)$ et $V_{HK}(f)$.

Soit $x \in \mathbb{R}^d$ et x^j sa j^{eme} composante. Pour $u \subseteq [\![1;d]\!]$, on notera x^u l'ensemble des x^j avec $j \in u$. et x^{-u} l'ensemble des x^j avec $j \notin u$.

On définit l'opérateur « : » (qui ne sera utilisé que dans cette section) de la façon suivante, $x^u : y^v$, soit le point $z \in [x^{u\cup v}, y^{u\cup v}]$ tel que $z^i = x^i$ pour $i \in u$ et $z^i = y^i$ pour $i \in v$ (cet opérateur est bien défini dans tous les cas).

Soit f une fonction sur $\mathbb{R}^d$ à valeurs dans $\mathbb{R}$.

On notera :

$$\Delta(f;a,b) = \sum_{v\subseteq[\![1;d]\!]} (-1)^{|v|} f(a^v : b^{-v})$$

et si $u \subseteq [\![1;d]\!]$

$$\Delta_u(f;a,b) = \sum_{v\subseteq u} (-1)^{|v|} f(a^v : b^{-v})$$

On appelle Y *ladder* (échelle) en dimension 1 sur $[a,b]$ ($-\infty < a < b < \infty$), un ensemble contenant a et un ensemble fini de valeurs de $]a,b[$. Chaque élément possède un successeur naturel (noté $y+$) dans le ladder (par convention, le successeur du plus grand élément est b). On notera $\mathbb{Y}$ l'ensemble de tous les *ladders* sur $[a,b]$. On définit des *ladders* en dimension quelconque par produit cartésien de *ladders* (le successeur étant défini en prenant le successeur dans chacune des dimensions).

La variation de f sur Y est définie comme (parfois on omettra d'écrire le Y comme dans la variation au sens de Vitali ou de Hardy and Krause) :

$$V(f) = V_Y(f) = \sum_{y\in Y} |\Delta(f;y;y+)|$$

La variation de f au sens de Vitali sur hyper-rectangles $[a,b]$ est :

$$V_{[a,b]}(f) = \sup_{Y\in\mathbb{Y}} \sum_{y\in Y} V_Y(f)$$

La variation de f au sens de Hardy et Krause sur hyper-rectangles $[a, b]$ est :

$$V_{HK}(f) = V_{HK}(j; a, b) = \sum_{u \subsetneq [\![1;d]\!]} V_{[a^{-u}, b^{-u}]} f(x^{-u} : b^u)$$

Une fonction f est dite bornée au sens de Hardy et Krause (noté BVHK) sur hyper-rectangles $[a, b]$ si et seulement si $V_{HK}(f) < \infty$. (ce qui implique en particulier que $V_{[a^{-u}, b^{-u}]}(f(x^{-u} : z^u)) < \infty$ pour tout $0 < |u| < d$ et tout $z \in [a^u, b^u]$.)

Si f et g sont BHKV (définies sur un hyper-rectangles $[a, b]$ en dimension d) alors $f + g$, $f - g$ et fg sont BHKV. De même, si $|f| > C > 0$ $1/f$ BHKV.

f est BHKV sur $[a, b]$ ssi $f = f_1 - f_2$ avec $\Delta_u(f_i; x, y) \geq 0$ pour i=1,2, pour tout $x \leq y$ et $u \subseteq [\![1; d]\!]$.

Remarque 1 :
Quand l'intégrande f est BVHK, l'approximation de son intégrale par des techniques de Quasi-Monte-Carlo est meilleur asymptotiquement que celle réalisée par un échantillonnage de Monte-Carlo.

♦

Remarque 2 :
Notons qu'il est difficile de savoir si une fonction est BHKV, en effet des outils existent pour la dimension 1 mais ne s'étendent pas à d quelconque. De même être capable de le faire en dimension 2 ne suffit pas pour généraliser à d quelconque. Ainsi si $f(x)$ et $g(x)$ sont des fonctions linéaires de l'hypercube en dimension d alors $\min(f(x), g(x))$ BHKV pour $d = 2$ et non BHKV pour $d > 2$).

♦

Remarque 3 :
Il existe des théorèmes permettant de borner V_{HK} à partir des dérivées de f si f est dérivable sur $[a, b]$ (avec éventuellement des discontinuités parallèles aux axes en nombre fini). Notons que ici encore, les parallèles aux axes jouent un rôle particulier comme pour la définition de la discrépance.

♦

1.8.3 Koksma-Hlawka

Le théorème de Koksma-Hlawka relie l'approximation de l'intégrale de f et la moyenne de f le long d'une suite (X_n) par la discrépance de X.

Théorème 1.27 *(Voir [Niederreiter; 1992]) Si f est une application de $[0,1]^d$ dans $\mathbb{R}$, de variation totale au sens de Hardy et Krause majorée par $V_{HK}(f)$, et si x_i est une suite à discrépance faible dans $[0,1]^d$, alors pour une certaine fonction c :*

$$\left|\frac{1}{n}\sum_{i=1}^{n} f(x_i) - \int f\right| \leq V_{HK}(f)c(d)\log(n)^d/n$$

mais aussi

$$\left|\frac{1}{n}\sum_{i=1}^{n} f(x_i) - \int f\right| \leq V_{HK}(f)D_n^*(x_1,\ldots,x_n)$$

Cette inégalité, réputée très bonne dans le pire cas possible, justifie l'intérêt de disposer de suites à la discrépance la plus faible possible lors du calcul d'intégrales. Dans le cas de l'apprentissage, nous nous intéressons à la discrépance de l'échantillon d'apprentissage avec comme objectif d'encadrer la déviation de l'erreur empirique.

1.8.4 Le cas des suites uniformément réparties

Dans le cas de variables $U_1,\ldots,U_n$ aléatoires uniformément réparties de loi uniforme, on notera $D_n^*(n) = D_n^*(U_1,\ldots,U_n)$. Cette notation particulière est réservée à une suite de variables uniformes.

Définition 1.17 (D_∞) *Soit $U_1,\ldots,U_n$ une suite de variables aléatoires **uniformément** réparties de loi uniforme sur $[0;1]^d$ et $F_1,\ldots,F_n$ la suite des fonctions de répartition empiriques associées (la fonction $t \to \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{u_k\in[0;t]}$). On note*

$$D_\infty(n) = D_\infty^*(U_1,\ldots,U_n) = \sup_{t\in[0;1]^d}\left\{\left|F_n(t) - \prod_{i=1}^{d} t_i\right|\right\}$$

Théorème 1.28 (Discrépance d'une va uniforme) *[Kiefer; 1961] Soit $U_1,\ldots,U_n$ une suite de variables aléatoires uniformément réparties de loi uniforme sur $[0;1]^d$, alors :*

$$\text{limsup}_{n\to\infty}\sqrt{\frac{2n}{\ln(\ln n)}}D_n^*(n) = 1 \text{ presque sûrement}$$

Définition 1.18 *On dit qu'une suite $X = X_1,\ldots,X_n$ à valeurs dans $[0;1]^d$ est à discrépance faible si sa discrépance $D_n^*(X)$ est asymptotiquement meilleure que la discrépance $D_\infty^*(n)$ d'une suite aléatoire $U_1,\ldots,U_n$ de loi uniforme sur $[0;1]^d$.*

On peut prouver que la discrépance $D_n^*(X)$ d'une suite quelconque vérifie :

$$\text{limsup}_{n\to\infty}\frac{nD_n^*(X)}{\ln(n)^{d/2}} \geq C_d$$

Où C_d est une constante ne dépendant que de la dimension d. Les meilleures discrépances connues sont asymptotiquement en $O\left(\frac{\ln(n)^d}{n}\right)$ [Niederreiter; 1992].

1.8.5 Exemples

Cette section présente quelques exemples de construction de suites déterministes à faible discrépance. Le lecteur se réfèrera à [Niederreiter; 1992] pour plus de détails.

Grille uniforme

La première idée venant à l'esprit pour avoir des points les plus uniformément répartis possible est d'essayer d'utiliser une grille régulière. Cependant il est possible de monter que la discrépance d'une telle grille est de l'ordre de $\frac{1}{\sqrt{n}}$, ce qui en fait un mauvais résultat.

Pour en comprendre l'origine, il faut se rendre compte que pour avoir une faible discrépance, il faut échantillonner le plus uniformément possible tous les rectangles parallèles aux axes. Du coup, pour une grille uniforme comme on le voit sur la figure 1.3, il y a des rectangles dont l'échantillonnage n'est pas bon (problèmes selon le placement des bords par rapport à la grille).

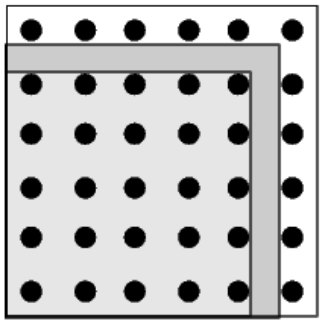


FIG. 1.3 : *Points placés selon une grille et visualisation de deux rectangles parallèles au axes n'étant pas échantillonnés uniformément.*

Suites de Van Der Corput

Soit p un entier strictement supérieur à 1. Pour tout nombre entier positif n, on note $a_0, \ldots, a_r$ les coefficients de la décomposition $p-$addique de n, c'est à dire les nombres entiers tels que :

$$n = a_0 + a_1 p + \ldots + a_r p^r,\ a_r > 0, 0 \leq a_i < p \text{ pour } 0 \leq i \leq r$$

La suite de Van Der Corput en base p est donnée par :

$$\phi_p(n) = \frac{a_0}{p} + \frac{a_1}{p^2} + \ldots + \frac{a_r}{p^{r+1}}.$$

Ainsi quand n s'écrit en base p : $n = a_r a_{r-1} a_1$, la décomposition $p-$adique de $\phi_p(n)$ est $a_0 a_1 a_r$. La discrépance de la suite $(\phi_p(n))_{n\in\mathbb{N}}$ est :

$$D_n^*(\phi) = O\left(\frac{\ln(n)}{n}\right)$$

Suites de Halton

C'est la version en dimension d des suites de Van Der Corput. Soit $p_1 \ldots, p_d$ des entiers strictement supérieurs à 1 et premiers entre eux. La suite de Halton est

$$x_n^d = (\phi_{p_1}(n), \ldots, \phi_{p_d}(n))$$

et sa discrépance est :

$$D_n^*(x^d) \leq \frac{1}{n}\prod_{i=1}^{d}\frac{p_i \ln(p_i\, n)}{\ln(p_i)} = O\left(\frac{(\ln(n))^d}{n}\right)$$

Suites de Faure

Soit $d > 1$ la dimension dans laquelle on travaille. Soit $p \geq d$ un entier premier et Δ_p l'ensemble des nombres x pour lesquels il existe un entier $K \geq 0$ tel que :

$$x = \sum_{i=0}^{K}\frac{a_i}{p^{i+1}} \text{ avec } \forall i \in [\![1, K]\!],\ a_i \in \mathbb{Z}$$

et soit $T_p : \Delta_p \to \Delta_p$ l'application définie par

$$T_p\left(\sum_{i=0}^{K}\frac{a_i}{p^{i+1}}\right) = \sum_{i=0}^{K}\frac{b_i}{p^{i+1}} \text{ avec } b_i = \sum_{j=i}^{K}\frac{j!}{i!(j-i)!}a_j \text{mod}(p)$$

Si ϕ_p est définie comme dans 1.8.5, la suite de Faure de dimension d est alors :

$$x_n = \left(\phi_p(n-1), T_p(\phi_p(n-1)), \ldots, T_p^{d-1}(\phi_p(n-1))\right)$$

La discrépance de cette suite est :

$$D_n^*(x^d) = O\left(\frac{(\ln(n))^d}{n}\right)$$

ou $d = 2$ de $d \geq 3$ qui donnent plus précisément l'ordre de grandeur.

1.8.6 Analogie avec les noyaux auto-reproduisants

Notons que l'on peut aussi voir apparaître un terme semblable à la discrépance dans le cadre des noyaux auto-reproduisants. En effet :

Définition 1.19 (Reproducing kernel) *Soit $(X, < ., . >$ un espace de Hilbert de fonctions réelles sur $[0,1]^d$. Une fonction $K : [0,1]^d \times [0,1]^d \to \mathbb{R}$ est appelée Reproducing kernel (noyau auto-reproduisant) sur X si :*

$$\forall x, \forall y \in [0,1]^d : K(x,y) = K(y,x)$$

$$\forall x \in [0,1]^d : K(.,x) \in X$$

$$\forall x \in [0,1]^d, \forall f \in X : f(x) =< f, K(.,x) >$$

Pour $P \subset [0,1]^d$ on définit :

$$\xi_p(x) = \int_{[0,1]^d} K(x,y)\mathrm{d}y - \frac{1}{n}\sum_{p \in P} K(x,p)$$

On montre alors :

$$\begin{aligned} |\int_{[0,1]^d} f(x)\mathrm{d}x - \frac{1}{n}\sum_{p\in P} f(p)| &= | < f, \xi_p > | \\ &\leq ||\xi_p|| \cdot ||f|| \end{aligned}$$

On peut interpréter $||\xi_p||$ comme la discrépance de P. $||f||$ étant une variante de la variation de Hardy et Krause $V_{HK}(f)$ utilisée dans l'inégalité de Koksma-Hlawka. Notons que cette inégalité devient une égalité dans le pire cas.

1.9 Réduction de la Variance

La dénomination «réduction de variance» existe chez les mathématiciens dans le cadre de calcul d'intégrales à la Monte-Carlo [Fishman; 1996]. En effet ces techniques de calcul d'intégrales ont une vitesse de convergence en $\frac{\sigma}{\sqrt{n}}$ (ou σ est la l'écart-type de la variable aléatoire X dont on veut calculer l'espérance). L'idée est de remplacer X par Y tel que $E(X) = E(Y)$ mais avec $\mathrm{Var}(Y) \ll \mathrm{Var}(X)$

Importance Sampling

Également appelé échantillonnage biaisé, il s'agit d'une technique pour réduire la variance des méthodes à la Monte-Carlo. Un point essentiel pour obtenir de petites erreurs sur le résultat final (pour un nombre fixé d'échantillons) est d'avoir une bonne stratégie

d'échantillonnage dans toutes les dimensions du problème (Monte-Carlo n'est vraiment utile que lorsque la dimension du problème est élevée). Si le volume à échantillonner est grand mais caractérisé par une faible densité à beaucoup d'endroits, on peut réaliser *l'importance sampling* en approximant la distribution par une fonction $f(X)$ et en générant aléatoirement X selon $f(X)$. Chaque point obtenu est ensuite pondéré.

Le calcul à la Monte-Carlo peut-être vu comme une intégration numérique, dans laquelle *l'importance sampling* joue le rôle d'un changement de variables (en voyant $f(x)$ comme une densité). Plus formellement, le but est de calculer une intégrale de la forme :

$$I = \int_{\Omega} g(X) dX$$

où $X = (X_1, \ldots, X_m)$ et $\Omega \in \mathbb{R}^m$. On va plutôt l'évaluer en utilisant la formule :

$$I = \int_{\Omega} \frac{g(X)}{f(X)} f(X) dX$$

Où $f(X)$ est souvent appelée « fonction d'importance » et, est une densité de probabilités sur Ω et telle que $g(X) \neq 0 \Rightarrow f(X) \neq 0$ (Les ensembles de mesure nulle pour f le sont aussi pour g).

Après avoir tiré de manière i.i.d. n points s_1, s_2, ..., s_n, $s_i \in \Omega$, selon la distribution donnée par $f(X)$, On estime l'intégrale par I :

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^{n} \frac{g(s_i)}{f(s_i)}$$

La variance estimée est :

$$\hat{\sigma}^2(\hat{I}_n) = \frac{1}{n \cdot (n-1)} \sum_{i=1}^{n} \left(\frac{g(s_i)}{f(s_i)} - \hat{I}_n \right)^2$$

Il est possible de montrer que tous ces estimateurs convergent sans biais vers les quantitées espérées. De plus le minimum de variance intervient pour :

$$f(X) = \frac{|g(X)|}{\int_{\Omega} |g(X)| \, dX}$$

Un problème ennuyeux réside dans le fait que cet optimum est lié à la valeur de I (pour les fonctions positives, il s'agit même exatement de I au dénominateur). Or c'est justement cette valeur que l'on cherche à estimer.

En pratique, on se contentera de choisir $f(X)$ proche de $g(X)$ afin de minimiser la variance. Quand il y a plusieurs dimensions, on procède généralement indépendamment pour chaque dimension. Il est (surtout dans les dimensions élevées) plus intéressant de

choisir avec soin la fonction d'importance, plutôt que d'augmenter la taille de l'échantillon. Pour davantage de détails on se référera à [Press et al.; 1995].

Notons que le prix à payer pour obtenir une meilleure valeur moyenne (pour un nombre fixé d'exemples) est la détérioration de l'estimation des corrélations. En effet, les variables calculées ne reproduisent pas le modèle de la même façon qu'un Monte-Carlo classique.

On trouve parfois aussi l'expression *importance sampling* hors de la réduction de variance. Il s'agit simplement de désigner la pondération adéquate des exemples.

Stratification

L'idée est cette fois de découper une intégrale en zones et d'affecter davantage de tirages aux zones possédant une grande variance. Plus formellement, on cherche à estimer l'espérance de $g(X)$ où X variable aléatoire dans $\mathbb{R}^d$ de densité f :

$$I = E(g(X)) = \int_{\mathbb{R}^d} g(x)f(x)\ dx$$

Soit $(D_i, i = 1, \ldots, m)$ une partition de $\mathbb{R}^d$

$$I = \sum_{i=1}^{m} E(g(X)|X \in D_i)P(X \in D_i)$$

La portion de I relative à D_i est désignée par :

$$I_i = \frac{E(\mathbb{1}_{\{X \in D_i\}} g(X))}{P(X \in D_i)}$$

Supposons connu $p_i = P(X \in D_i)$.

Définissions X^i comme une variable aléatoire suivant la loi de X restreinte à D_i. On simule la loi de X^i par rejet (ou une autre méthode) puis on estime I_i en utilisant un échantillon de taille n_i par :

$$\hat{I}_i = \frac{1}{n_i}(X_1^i + \ldots + X_{n_i}^i)$$

Puis I par :

$$\hat{I} = \sum_{i=1}^{m} P(X \in D_i)\hat{I}_i$$

La méthode converge si tous les n_i tendent vers l'infini. Le problème est maintenant de choisir l'affectation des n_i avec comme contrainte $\sum_{i=1} mn_i = n$ (répartition des n mesures parmis les m régions). On souhaite minimiser :

$$\text{Var}(\hat{I}) = \sum_{i=1}^{m} p_i^2 \text{Var}(\hat{I}_i) = \sum_{i=1}^{m} p_i^2 \frac{\sigma_i}{n_i} \text{ avec } p_i = P(X \in D_i)$$

Par la technique des multiplicateurs de Lagrange, on obtient :

$$n_i = n\frac{p_i\sigma_i}{\sum_{i=1}^{m} p_i\sigma_i}$$

On observe une **réduction de la variance, si les régions D_i ont été bien choisies** (*i.e.* si les régions D_i correspondent à des variances différentes). Le gain correspond a celui de la minimisation et peut donc être évalué. Il est essentiel que les régions aient des variances différentes pour voir apparaitre l'intérêt de ne pas échantillonner uniformément.

Notons l'existence d'autres techniques similaires pour la réduction de la variance dans le calcul des intégrales à la Monte-Carlo :

- échantillonnage corrélé ;
- Contrôle des variations ;
- Variables antithétiques ;
- Réduction géométrique de la variance (également utilisée dans les chaînes de Markov par [Munos; 2005]).

1.10 Apprendre à Partir de Données Partiellement étiquetées

Dans certains cas, il est facile d'obtenir des données mais leur étiquetage est coûteux. Ainsi, en reconnaissance de la parole, il est facile d'enregistrer des sons mais l'étiquetage doit se faire par des experts humains qui doivent séparer les phonèmes, ce qui est long et fastidieux. Le but de l'apprentissage actif dans ce cas est de réussir à apprendre un classifieur performant ne demandant pas l'étiquetage de tous les exemples. Plus précisément on considère au départ que toutes les étiquettes sont inconnues et on pénalise l'algorithme chaque fois qu'il demande l'étiquetage d'un exemple. On espère qu'en faisant des demandes « intelligemment », l'algorithme aura besoin de significativement moins d'exemples pour apprendre.

Ainsi, considérons d'un espaces d'instances X réduit à $\mathbb{R}$ où les classifieurs sont simplement des seuils. L'espace d'hypothèses est $H = h_w$, avec $h_w(x) = \mathbb{1}_{x>w}$. Selon la théorie de Vapnik-Chervonenkis, si le concept cible est dans H alors pour trouver un classifieur avec un taux d'erreur au plus ε il suffit de tirer et d'étiqueter $m = O(1/e)$ exemples et de trouver un classifieur qui soit consistant avec eux. Cependant, sur un cas aussi simple, on peut faire beaucoup mieux : si on commence par tirer m exemples sans leur label, par dichotomie on pourra facilement trouver une estimation de la valeur de w de la transition. Dans ce cas on utilise seulement $\log m$ exemples.

Malheureusement les choses deviennent dramatiquement plus difficiles dès que l'on s'éloigne de cet exemple jouet. Ainsi dans [Freund et al.; 1997], les auteurs s'attaquent au problème du *query by committee*, qui est un algorithme pour filtrer les requêtes contenant de l'information parmi un flux aléatoire d'entrées.

Leur algorithme recoit un flux de données non étiquetées et prend immédiatement la décision de demander ou non l'étiquetage. Ils démontrent que si les données sont tirées uniformément sur la sphère unité en dimension d et que si les étiquettes sont séparables par un hyperplan passant par l'origine alors il est possible d'atteindre l'erreur optimale en voyant $O(d/e)$ points et en demandant $O(d\log(1/e))$ étiquetages. Ceci correspond à un gain exponentiel par rapport aux résultats classiques d'apprentissage d'un séparateur linéaire. L'algorithme proposé possède cependant une complexité exponentielle.

Ces deux exemples sont des encouragements à essayer d'adapter les algorithmes d'apprentissage en leur permettant d'être actif. Par actif, on entend essentiellement influencer la base d'apprentissage (c'est ce que nous considèreront dans la partie suivante). En effet, la possibilité de choisir les points à étiqueter nous permet d'influer sur la base d'apprentissage en fonction des résultats déjà obtenus. Cela ressemble un peu à de l'apprentissage semi-supervisé sauf que l'on étiquette de plus en plus d'exemples.

Dans [Dasgupta et al.; 2005], des progrès ont été réalisés dans la compréhension de ce cas d'apprentissage actif en utilisant une variante du perceptron pour obtenir le même genre de résultats. On trouvera également dans [Dasgupta; 2005] plusieurs bornes inférieures appliquées à l'apprentissage actif. Par exemple dans le cas d'hyperplans ne passant pas nécessairement par l'origine, il est montré que la complexité passe obligatoirement au minimum en $\theta(d/e)$.

Des travaux ont également étés menés dans le cas où l'on est autorisé pendant l'apprentissage à demander des points (au lieu de simplement choisir dans un *pool* de points non-étiquetée) [Angluin; 2004]. Ce qui peut mener à de nouveaux problèmes puisque les instances demandées par l'algorithme sont parfois plus difficiles à étiqueter pour l'expert (voir [Blum et al.; 1995]) que des instances « normales ».

De plus, les données non encore étiquettées peuvent être mieux utilisées que pour estimer la distribution des exemples [Bilenko et al.; 2004]. Ces approches montrent la potentialité de l'apprentissage actif, *i.e* le contrôle en ligne de la distribution des données d'apprentissage. Toutefois, on peut s'interroger sur la validité des résultats obtenus par *cross-validation* sur des bases de données dont on a choisi l'étiquettage. En effet, la distribution des exemples n'a dans ce cas pas été influencée, mais la distribution des exemples étiquettées l'est.

1.11 Conclusion

Jusqu'à la fin des années 1990, la plupart des travaux théoriques sur l'estimation du risque en apprentissage comportaient pour les praticiens deux inconvénients. Soit ils cherchaient à produire des résultats universels, ne dépendant que de F et pas de la distribution des données, auquel cas ils fournissaient des bornes pessimistes.

Soit ils prenaient en compte la distribution des exemples et devenaient alors difficiles à utiliser car sur des données réelles, on ne connait généralement pas la distribution. Au-

jourd'hui on sait traiter de manière assez systématique les déviations de la variance (en particulier grâce aux inégalités de concentration). L'étude des bornes à également pris de l'importance avec l'essort des SVM ou cette fois on optimise une borne plutôt que le risque empirique. Cette démarche est parfois vivement contestée.

Le propos de la seconde partie de cette thèse est de voir comment ces outils analytiques peuvent-être combinés pour attaquer le problème de l'apprentissage actif. En particulier nous verrons quelles stratégies nous pouvons utiliser pour réduire la variance lors de l'apprentissage (plus exactement pour optimiser la borne sur la variance).

Nous verrons également l'apport de l'utilisation des suites à faible discrépance dans certains contextes de l'apprentissage (en effet puisque l'on est parfois libre de choisir nous même les points à utiliser pour l'apprentissage, pourquoi ne pas prendre en compte la discrépance de notre suite de points, comme lors du calcul d'une intégrale).

Deuxième partie

APPRENTISSAGE ACTIF DANS LE FORMALISME PAC

Cette partie est matheuse et nécessite d'avoir intégré la partie sur les Mathématiques pour l'Apprentissage. Le chapitre 2 requiert des notions sur les classes de Donsker, tandis que le chapitre 3 se base sur les suites à faible discrépance.

L'apprentissage supervisé dispose classiquement d'une base d'apprentissage $\mathcal{E} = \{(x_i, y_i)\}$ où (x_i, y_i) sont généralement i.i.d. échantillonnés selon la distribution jointe $P(X, Y)$. Le but est de trouver un procédé permettant de déduire Y de X. Dans la plupart des approches, on suppose que les données d'apprentissage (les exemples) sont indépendants et identiquement distribués (i.i.d.). L'apprentissage actif est le cas où l'algorithme d'apprentissage peut influencer la distribution de la base d'apprentissage. Les données ne sont donc plus i.i.d. Plusieurs types d'apprentissage actifs sont possible selon l'influence dont nous disposons sur la récolte des exemples. Voici quelques possibilités (non exhaustives) :

1. La référence, l'apprentissage non actif, *i.e.* les exemples sont des réalisations de (X, Y) ;
2. L'algorithme choisit x et un Oracle fournit y, réalisation de Y suivant la loi $P(Y|X = x)$;
3. L'algorithme fournit E, et un Oracle donne X, Y distribué selon $P(X, Y|X \in E)$. Ici on ne contrôle plus directement la valeur de x, on se contente de demander un exemple qui soit dans une certaine région de l'espace.
4. Étant donné $h(.)$, l'algorithme fournit k (Étiquette prédite) et un Oracle donne (X, Y) distribué selon $P(X, Y|h(X) = k)$. Ici on ne contrôle même plus directement la région de l'espace pour X, on peut simplement imposer qu'une certaine contrainte soit satisfaite (traduite par $h(X) = k$).

Une des difficultés majeures est de réussir un compromis entre exploration et exploitation. De même doit-on se concentrer en demandant davantage d'exemples dans des zones où l'apprentissage se passe mal ou augmenter la précision d'autres régions de l'espace ?

Dans cette partie, nous allons considérer un apprentissage en deux passes : on commence par obtenir des informations sur la distribution des données puis on influe ensuite sur la distribution pour améliorer l'apprentissage. La première phase d'observation est importante, en effet il n'est sans doute pas pertinent de chercher à augmenter la qualité de l'apprentissage dans une région où la densité des exemples est faible.

Dans cette partie par abus de notation $x = \text{argmin } f$ est utilisé même lorsque l'unicité n'est pas garantie ; il s'agit de $x \in \text{argmin } f$. Soit P une loi de probabilités sur un espace X. Soit F une famille de fonctions de X dans $\mathbb{R}$.

Sauf mention explicite du contraire, on définit $L(f)$ pour f dans F par $L(f) = E(f(x) - y)^2$. On définit $\hat{P}$ une loi empirique, associée à un échantillon aléatoire simple, indépendamment identiquement distribué sauf mention explicite du contraire (en particulier, on s'intéressera fréquemment à une combinaison d'échantillons), suivant P, de n

exemples. On note $\hat{E}$ l'opérateur espérance associé. Sauf précision spécifique, on définit $\hat{L}(f)$, l'erreur empirique d'une hypothèse f pour f dans F par $\hat{L}(f) = \hat{E}(f - g)^2$ où g désigne le concept cible à apprendre.

Chapitre 2

Apprentissage Actif et Stratification

Classiquement l'apprentissage statistique s'intéresse à la minimisation du risque empirique ou plus précisément d'une borne de l'erreur en généralisation dans l'espoir de minimiser l'erreur en généralisation elle-même, Le bien fondé de cette stratégie repose sur la nature i.i.d. de l'échantillon d'apprentissage. Même si en fait la méthodologie PAC (minimiser $E(L) - E(\hat{L})$) reste valable dans les cas non i.i.d. Cependant dans ce cas de figure, l'estimation de $E(L)$ par une simple moyenne n'est pas judicieux. Il faut pouvoir construire un estimateur différemment, ce qui soulève de nombreux problèmes.

Dans la pratique, l'apprenant peut souvent influer sur la distribution des exemples. C'est particulièrement souhaitable quand le coût de l'étiquetage est élevé par rapport à la collecte des données (par exemple, une collecte automatique, et un étiquetage réalisé par un expert humain). On peut également souhaiter influencer la distribution des exemples lorsque l'on souhaite biaiser l'ensemble d'apprentissage pour étudier plus précisément un sous-espace du domaine d'étude (par exemple pour isoler l'impact du tabac ou de la surcharge pondérale dans une étude médicale). Cette possibilité d'influer sur les données va à l'encontre de l'hypothèse i.i.d. des exemples d'apprentissage, ce qui n'est généralement pas pris en compte dans le processus d'apprentissage.

La plupart des travaux en apprentissage actif s'intéressent de fait à l'apprentissage « en ligne » ou incrémental, quoique les problématiques de l'apprentissage incrémental et actif soient différentes. Dans ce travail, nous allons nous intéresser à des procédures d'apprentissage en deux étapes. Après avoir initialement observé un petit échantillon des données tirées selon la distribution des données, le système d'apprentissage pourra demander davantage de données en imposant une contrainte sur la distribution des nouveaux exemples. Les résultats peuvent naturellement être généralisés au cas de p étapes Concrètement, les résultats présentés dans ce chapitre, concernent :

- Des algorithmes calculant une distribution des exemples optimisée entre des classes ou des sous-ensembles du domaine d'apprentissage.
- Des vitesses de convergences optimisées par ces algorithmes ;
- Les bornes d'erreur en généralisation correspondantes ;

Enfin sont présentés, des algorithmes mettant en œuvre ces stratégies, ainsi que la validation expérimentale sur des données artificielles et réelles.

2.1 Introduction et notations

Le contexte est toujours celui de l'appentissage supervisé avec les notations du chapitre 1. On dispose donc d'une base d'apprentissage $\mathcal{E} = \{x_{,}y_i\}$ à partir de laquelle on travaille. On peut parler d'apprentissage actif dans le cas où ces exemples ne sont pas indépendants et identiquement distribués et que l'on peut les influencer. L'influence dont on dispose sur ces exemples peut prendre plusieurs formes :

- Historiquement les premiers travaux dans se situant dans ce cadre sont ceux du *learning by queries*, [Vidyasagar; 2002].
- Une seconde approche vient du coté de l'apprentissage en-ligne (voir les travaux de Lugosi, déterminant à chaque pas l'exemple le plus informatif compte tenu de l'hypothèse courante). La part active est dans ce cas complètement embarquée dans l'algorithme d'apprentissage. Dans cette lignée, on peut aussi citer [Bryant et al.; 2001] et le *self-directed learning* de [Goldman and Sloan; 1994].
- Un cas limite est celui ou l'ensemble d'apprentissage est construit de manière complètement disjointe de l'algorithme. On s'approche cette fois des plans d'expériences utilisés en statistiques [Cervellera and Muselli; 2004] nous nous pencherons dessus lors du chapitre 3.
- Enfin sont également actifs les apprentissages guidés par un professeur sans collusion (voir l'HDR de F.Denis). On touche ici au paradigme « d'enseignabilité » (*Teachability*). On suppose que le professeur n'est pas « beaucoup » plus intelligent que l'élève et on s'intéresse à la transmission d'un concept sans utiliser de codage de la représentation interne (l'apprenant et l'enseignant n'ayant pas nécessairement la même). Il est toutefois difficile de juger si l'on a pas violé ce principe dans les stratégies mises en œuvre ici.

Ce chapitre explore une approche différente inspirée des notions de stratification [Brewer; 1992] dans le but d'améliorer la vitesse de convergence de l'apprentissage (*i.e* le nombre d'exemples nécessaires pour atteindre un certain niveau de performances). L'idée est la suivante : dans les instituts de sondages, on peut préférer qu'une certaine catégorie soit sur-représentée, si pour des raisons statistiques, cela conduit à des estimateurs plus précis [Tille; 2001]. Ce principe n'est pas seulement utilisé dans les sondages : le Monte-Carlo biaisé est fondé sur la même idée. Dans le cadre de l'apprentisage, les connaissances *a priori* peuvent guider la sélection des données d'apprentissage (*e.g.* sur-représentation d'une classe réputée difficile à apprendre, ou d'importance critique).

Aussi, en apprentissage par renforcement les expérimentateurs préfèrent souvent privilégier certaines régions de l'espace sachant qu'ils doivent généralement faire face à des données présentant de grandes dimensions et ne peuvent donc pas être exhaustifs).

L'usage de stratégies de stratification suppose que les exemples ne soient pas tous d'importance égale. L'usage de poids attachés aux exemples et réflétant leur importance est possible dans le cadre de nombreux algorithmes d'apprentissage, tels que les réseaux de neurones, les SVM, et même les arbres de décision [Buntine; 1989].

Le protocole d'apprentissage actif considéré dans la suite de ce chapitre est le suivant :
- L'algorithme reçoit dans un premier temps une base d'apprentissage i.i.d.
- L'algorithme d'apprentissage est utilisé sur ce premier jeu de données ; une hypothèse est apprise.
- L'algorithme fournit une stratification, c'est à dire qu'il fournit un classifieur capable de ranger les exemples en K catégories et donne K entiers $n_1, \ldots, n_K$.
- Une nouvelle distribution des données est considérée, où la masse de la catégorie K_i est proportionnelle à n_i.

Plus précisément, dans ce contexte on considère les cas suivants :

1. L'apprentissage passif, dans lequel les exemples sont tirés selon la loi jointe $P(X, Y)$ du domaine.
2. Un premier protocole d'apprentissage actif ou l'algorithme choisit x et l'oracle fournit y selon $P(Y|X = x)$. C'est en particulier le cas de figure de l'approximation de fonctions déterministes ([Cervellera and Muselli; 2004]).
3. Un second protocole où un partitionnement par h en K catégories est défini sur X. L'algorithme d'échantillonnage repose ensuite sur une distribution uniforme des exemples, conditionné par chacune des catégories. Formellement, l'on fournit une fonction discrète h à valeurs dans $[\![1, K]\!]$ puis l'algorithme fournit un vecteur $(n_1, \ldots, n_K)$ d'entiers. Ensuite pour chaque $i \in [\![1, K]\!]$ l'oracle fournit n_i exemples (X, Y) distribués selon $P(X, Y|h(X, Y) = i)$.
4. Enfin, dans une troisième version, l'algorithme fournit à la fois h et $(n_1, \ldots, n_K)$ puis l'oracle fournit pour chaque $i \in [\![1, K]\!]$ un ensemble de n_i exemples (X, Y) distribué selon $P(X, Y|h(X, Y) = i)$.

Dans de chapitre, nous allons considérer un apprentissage actif en deux passes. Dans un premier temps, l'algorithme évalue la distribution des exemples (X, Y) sur le premier jeu d'apprentissage. Puis, dans un second temps l'apprenant requiert un jeu de données plus grand, fournissant à l'oracle des exigences sur la strate des exemples(dans le cas des jeux de données naturellement stratifiées). Dans le cas de la stratification automatique, l'apprenant fournit en plus du nombre de points souhaité par strate, la manière de stratifier les données. Le but est de battre les performances d'un algorithme passif disposant du même volume de données.

Notations

On suppose $Y = \mathbb{R}$.

Soit P une loi de probabilité sur un espace X. Soit F une famille de fonctions de X à valeurs dans $\mathbb{R}$.

Pour $f \in F$, et $z = (x, y)$ avec $x \in X$ et $y \in \mathbb{R}$ on définit $c_f(z) = (f(x) - y)^2$ la fonction de coût.

Par définition, l'erreur en génréralisation de f est donnée par l'espérance du coût de f, $L(f) = E(c_f)$

On notera $\hat{P}$ la loi empirique associée à la distribution P appliquée indépendamment et identiquement à n exemples. De la même façon, $\hat{E}$ est l'espérance associée à $\hat{P}$ et $\hat{L}(f) = \hat{E}(f(x) - y)^2$.

$\sigma_i(f)^2$ est la variance de $c_f(z)$ dans la strate i.

Quand $Z = (Z_t)_{t \in T}$ est un processus gaussien, $\sigma(Z)^2$ désigne la plus grande variance de Z_t pour $t \in T$.

2.2 Données naturellement stratifiées

Considérons tout d'abord le cas des données naturellement stratifiées, cela signifie que l'algorithme peut demander des exemples choisis dans une strate donnée, et que les probabilités *a priori* des différentes strates sont connues. *i.e* lorsque l'on dispose d'une partition naturelle de l'espace des instances de X en K catégories dont les masses $p_1, \ldots, p_k$ sont connues.

Le cas le plus simple de données naturellement stratifiées est celui de la classification, où les catégories coïncident avec les classes. Il est alors possible d'échantillonner indépendamment les classes, par exemple de contrôler le nombre d'individus sains ou malades dans la base d'apprentissage, indépendamment de leur fréquence effective dans la population, mais en fonction par exemple des coûts d'erreur de classification [Domingos; 1999].

Un autre cas de figure est celui où la stratification peut être fondée sur une variable auxiliaire. Par exemple, il peut être utile de sur-représenter la catégorie des fumeurs dans une étude sur risques cardio-vasculaires si l'on juge qu'une bonne connaissance de ce type de personnes est important pour la qualité des prédictions.

2.2.1 Classification

L'algorithme A d'apprentissage stratifié actif comprend les étapes suivantes.

1. Soit $\mathcal{E} = \{(x_i, y_i) | x_i \in X,\ y_i \in Y = [\![1, k]\!],\ i \in [\![1, n]\!]\}$ une base d'apprentissage i.i.d.
2. L'apprentissage d'une hypothèse h_{init} conduit à estimer la variance du coût d'erreur

dans chacune des classes notée $\text{Var}_i(c_f)$.

3. Un poid p_i est associé à la classe i.
4. Une base stratifiée $\mathcal{E}_{strat}$ est ensuite constituée où $P(y = i) = p_i$.
5. Une nouvelle hypothèse h_{strat} est apprise à partir de cette seconde base.

Nous allons démontrer que pour une allocation pertinente des exemples (une allocation désigne le nombre d'exemples par strate), il est possible d'améliorer les vitesses de convergence et l'erreur en généralisation.

Le schéma de la preuve est le suivant :

- Lemme 2.1 établit quelle est la variance associée à une allocation donnée des exemples aux différentes strates ;
- Une forme spécialisée de l'apprentissage PAC (Définition 2.1) montre que nous pouvons nous restreindre à considérer des hypothèses « assez bonnes ».
- Le lemme 2.2 utilise l'inégalité de Chernoff pour tirer parti de la variance optimisée grâce à la stratification ;
- Le lemme 2.3 établit grâce à Chernoff que l'algorithme A améliore la vitesse de convergence par rapport à une base d'apprentissage i.i.d.
- Le théorème 2.1 conclut ; il considère l'optimisation d'un risque empirique sous des contraintes additionnelles techniques. Une variante 2.2 présente le résultat sous une forme plus naturelle et plus simple à exploiter.

Soit f un classifieur. Soit $L(f)$ le coût de f. On a :

$$L(f) = \sum_{k,l} P(class = k \wedge f = l) L(k, l)$$

où $L(k, l)$ est le coût de prédire l alors que la vraie classe est k.

$$L(f) = \sum_{k} P(k) \int L(k, f(x)) d_k(x)$$

où $P(k)$ est la probabilité *a priori* de la classe k et $d_k(x)$ sa densité.

Si $n(k)$ dénote le nombre d'exemples dans la classe k, l'estimateur empirique de $L(f)$ noté $\hat{L}(f)$ est donné par :

$$\hat{L}(f) = \sum_{k} P(k) \frac{1}{n(k)} \sum_{i=1}^{n(k)} L(k, f(x_{k,i}))$$

où $x_{k,i}$ est le i^e exemple de classe k.

Lemme 2.1 (Variance de la fonction de coût pour un classifieur f donné) *Cette variance est :*

$$\textit{Var}\ \hat{L}(f) = \sum_k P(k)^2 \frac{1}{n(k)} \textit{Var}\ L(k, f(.))$$

Le maximum est atteint lorsque les termes (relatifs aux différentes classes) sont égaux[1] Ainsi, la variance du coût d'erreur atteint son minimum lorsque $n(k)$ est proportionnel à $P(k)\sqrt{\text{Var } L(k, f(.))}$.

Pour $L(k, l) = \mathbb{1}_{k \neq l}$ (fonction de coût qui est égale à la probabilité d'avoir une mauvaise classification), cette allocation est proportionnelle à $P(k)\sqrt{p_k(1 - p_k)}$ avec p_k la probabilité d'avoir une classification correcte.

Ceci montre que la solution optimale est d'allouer davantage d'exemples aux classes possédant soit une forte variance soit une probabilité d'apparition plus élevée. La formule générale prouve qu'il peut-être utile de demander plus d'exemples pour les classes qui fournissent les coûts les **plus variables**. Cela peut être mis en correspondance avec l'intuition qui conseille de se focaliser davantage sur les classes **les plus difficiles**.

Définition 2.1 (Forme affaiblie de l'apprentissage PAC) *Les fonctions qui sont empiriquement bonnes sont bonnes en généralisation et possèdent des variances locales proches de celles des fonctions optimales.*

Soient $\alpha_i \in \mathbb{R}^+$ pour $i \in [\![1, K]\!]$ tels que $\sum_i \alpha_i = 1$.

Soit $\hat{f}_n$ le classifieur sélectionné par l'algorithme d'apprentissage après l'observation pour tout i de $\lfloor \alpha_i n \rfloor$ exemples dans la classe i. $\hat{f}_n$ est une variable aléatoire.

Soit F_ε l'ensemble des classifieurs proches de l'optimum avec une tolérance ε :

$$F_\varepsilon = \{f | L(f) \leq L^* + \varepsilon\}$$

où L^* désigne l'erreur en généralisation minimale sur F.

Nous faisons l'hypothèse que l'algorithme sur l'espace F que nous allons utiliser dans la phase d'apprentissage satisfait la propriété suivante. Celle-ci est moins forte mais inspirée et très liée à l'apprentissage PAC.

$$P(\exists f; \hat{L}(f) \leq \inf \hat{L} + \varepsilon'/2 \wedge f \notin F_{\varepsilon'}) \leq \delta(\varepsilon', n)$$

ou $\delta(\varepsilon', n)$ converge vers 0 quand $n \to \infty$ pour tout $\varepsilon' > 0$.

Ceci signifie que si f est empiriquement presque optimale alors f est, avec une forte probabilité, presque optimale en généralisation. C'est une conséquence directe des résultats de convergence uniforme basée sur la VC dim.

[1] Ce résultat, classique en optimisation, peut être démontré par décomposition Lagrangienne voir [Tille; 2001].

Remarque :
Un point important : quand on suppose que la fonction optimale g est dans F, c'est pour que à la fois dans le cadre de la régression L_2 et dans le cadre de la classification multi-classes avec matrice de coût (telle que $l \mapsto L(k,l)$ ne possède qu'un seul minimum pour tout k), on ait, lorsque la propriété PAC précédente est vérifiée, avec probabilité au moins $1-\delta(\varepsilon', n)$:

$$\exists f, L(f) \leq \inf L + \varepsilon' \wedge \exists i; |\sigma_i(f) - \sigma_i(f^*)| \geq w(\varepsilon')$$

pour un $w(.)$ tel que $\lim_{\varepsilon' \to 0} w(\varepsilon') = 0$.

Dans le cas de la régression, ceci sera montré en détails dans le point 2 du théorème 2.4 (avec $w(.)$ étant principalement un opérateur racine carrée). Dans le cas de la classification,

$$|\sigma_i(f) - \sigma_i(f^*)| = O(P(f \neq f^*)) = O(L_f - L_{f^*})$$

d'où le résultat avec $w(.)$ linéaire (dépendant de la matrice de coût). L'importance du fait qu'un petit L_f conduise à un petit σ a été introduite dans d'autres contextes ; se référer à la condition de bruit de Massart et Tsibakov (voir [Boucheron et al.; 2005] pour un panorama).

♦

Lemme 2.2 (Utilisation de la variance)

Le but est d'utiliser la variance pour construire des bornes sup sur $|\hat{L}(f) - L(f)|$, pour un $f \in F_{\varepsilon'}$ fixé. On suppose sans perte de généralité que $0 \leq L(.) \leq 1$. Soit $f \in F_{\varepsilon'}$, alors :

$$\hat{L}(f) = \sum_k \frac{1}{n(k)} \sum_i^{n(k)} P(k) L(k, f(x_{k,i}))$$

et

$$L(f) = \sum_k E(P(k) L(k, f(.)))$$

On note $n_0 = \min_i n_i$ et

$$Y_{k,i} = n_0 \left(\frac{P(k)}{n(k)} L(k, f(x_{k,i})) - \frac{P(k)}{n(k)} E_{X_{(k)}}(L(k, f(X_{(k)}))) \right)$$

avec $x_{k,i}$ le i^e exemple de classe k et $X_{(k)}$ la variable aléatoire associée à la classe k.

On a donc :

$$Y = \sum_{k,i \in [\![1; n(k)]\!]} Y_{k,i} = n_0 \left(\hat{L}(f) - L(f) \right)$$

Les X_i sont des variables aléatoires indépendantes et centrées bornées par 1 (Car $|L(.)| \leq 1$; $0 \leq P(k) \leq 1$ et $n_0 = \min_i n_i$). En utilisant l'inégalité de Chernoff en posant :

$$\Delta \geq \sup_{f \in F_{\varepsilon'}} \text{Var}(\hat{L}(f))$$

$$0 \leq \varepsilon \leq \frac{2\sigma^2(Y)}{n_0} = 2n_0 \text{Var}(\hat{L}(f))$$

$$\delta'(\varepsilon, \Delta) = 2 \exp\left(\frac{-\varepsilon^2}{4\Delta}\right)$$

On obtient :

$$P(|\hat{L} - L| > \varepsilon) = P(n_0 \times |\hat{L} - L| > n_0\varepsilon) = P(|Y| > n_0\varepsilon)$$

$$\leq 2 \exp\left(\frac{-n_0^2\varepsilon^2}{4\sigma^2(Y)}\right) = 2 \exp\left(\frac{-\varepsilon^2}{4\text{Var}(\hat{L}(f))}\right) \leq \delta'(\varepsilon, \Delta)$$

Lemme 2.3 (Apprentissage dans $F^{\varepsilon}_{\varepsilon'}$)

Soit $\hat{L}'$ une fonction de perte empirique sur F (et L_{max} une borne sur le coût d'erreur en classification). Définissons la distance suivante sur les classifieurs :

$$||f - f'|| = L_{max} \cdot P(f \neq f')$$

On suppose que la fonction optimale f^* est dans $\{f; \hat{L}'(f) \leq \varepsilon'/2\}$,

On considère

$$F^{\varepsilon}_{\varepsilon'} = \{f \in F; \hat{L}'(f) \leq \inf_F \hat{L}' + \varepsilon'/2\} \cap S_\varepsilon$$

ou S_ε est un ε-net de F pour la norme précédente et inclus dans F (ce dernier point n'est pas habituellement requis dans la définition d'un ε-net). On définit $\hat{f} = \arg\min_{f \in F^{\varepsilon}_{\varepsilon'}} \hat{L}(f)$. Et on suppose que :

$$\forall f \in F^{\varepsilon}_{\varepsilon'} \ \ \Delta \geq \text{Var}(\hat{L}(f))$$

alors :

$$L(\hat{f}) \leq L(f^*) + 3\varepsilon$$

avec confiance au moins

$$1 - |F^{\varepsilon}_{\varepsilon'}|\delta'(\varepsilon, \Delta)$$

Démonstration :
Pour toute fonction f sur le réseau (dans l'ε-*net*), avec probabilité au moins $1 - \delta'(\varepsilon, \Delta)$ on a

$$|\hat{L}(f) - L(f)| \leq \varepsilon$$

Avec probabilité au moins $1 - |F^{\varepsilon}_{\varepsilon'}|\delta'(\varepsilon, \Delta)$, toutes les fonctions f sur le réseau vérifient :

$$|\hat{L}(f) - L(f)| \leq \varepsilon$$

En supposant $f^* = \text{argmin}\, L$ et $\hat{f} = \text{argmin}\, \hat{L}$. Pour tout f dans F on note f' un des membres de l'ε-net qui est à distance au plus ε de f (pour la distance considérée par l'ε-net). On obtient donc :

$$L(\hat{f}) \leq \hat{L}(\hat{f}) + \varepsilon \leq \hat{L}(f^{*'}) + \varepsilon \leq L(f^{*'} + 2\varepsilon) \leq L(f^*) + 3\varepsilon$$

■

A partir de ces quatre éléments, nous allons pouvoir étudier une série d'algorithmes basés sur l'algorithme A. Celui-ci dépend d'un niveau de risque $\delta_1 + \delta_2 + \delta_3$. Cet algorithme utilise la classe pour faire une partition *a priori*. L'algorithme B montre qu'il existe une version plus simple, évitant une optimisation sur les ε-nets tout en conservant presque les mêmes vitesses de convergence. L'algorithme C est une adaptation naturelle à des partitions définies par une variable auxiliaire ; celui-ci est particulièrement intéressant dans le cadre de la régression. Enfin, l'algorithme D montre qu'un partitionnement automatique est possible. Il requiert des partitions qui n'aient pas de trop grands coefficients de pulvérisation.

Algorithme A :

1. Tirer $k = o(n)$ exemples ;
   Soit $\hat{L}'$ le coût empirique associé ; soit ε' la précision telle que pour tous les f optimaux avec une précision $\varepsilon'/2$ pour $\hat{L}'$ sont aussi optimaux pour L avec précision ε' au niveau de confiance $1 - \delta(\varepsilon', k) \geq 1 - \delta_1$.
2. Phase d'apprentissage : Trouver $h = \text{argmin}\, \hat{L}'$;
3. Tirer $k' = o(n)$ exemples (i.i.d.) pour estimer $\sigma_i(h)$ qui est la racine de la variance du coût de h dans la classe i (avec $h = \text{argmin}\, \hat{L}'$) ;
4. Calculer, grâce aux estimations de $\sigma_i(h)$, et à partir de la remarque de la définition 2.1, un vecteur $(\sigma'_i)_{i \in [\![1,K]\!]}$ tel que

   $$\forall f; \hat{L}'(f) \leq \hat{L}'(h) + \varepsilon'/2 \Rightarrow \forall i; \sigma_i(f) \leq \sigma'(i)$$

   au niveau de confiance $1 - \delta_2$.
5. Déduire de cette borne une formule $\Delta'(n_1, \ldots, n_K)$ telle que pour un coût empirique $\hat{L}$ associé à n_1 exemples dans la classe 1, n_2 exemples dans la classe 2,...,n_K exemples dans la classe K :

   $$P(\exists f; L(f) \leq \inf_F \hat{L} + \varepsilon'/2 \wedge \text{Var}\, \hat{L}(f) > \Delta'(n_1, \ldots, n_K)) \leq \delta_2$$

6. Avec $n_1, \ldots, n_k$ optimisant $\Delta'(n_1, \ldots, n_k)$ sous la contrainte $\sum_i n_i = n$, tirer n exemples avec n_i exemples dans la classe i ;

7. Soit ε tel que $|F^{\varepsilon}_{\varepsilon'}|\delta'(\varepsilon, \Delta') \leq \delta_3$,
8. Apprendre $\hat{f} \in \text{argmin}_{F^{\varepsilon}_{\varepsilon'}} \hat{L}$.

Théorème 2.1 (Borne non asympotique pour l'apprentissage) *Soit l'algorithme A. On a :*

$$P(L(\hat{f}) \geq \inf_f L(f) + 3\varepsilon) \leq |F^{\varepsilon}_{\varepsilon'}|\delta'(\varepsilon, \Delta'(n_1, \ldots, n_K)) + \delta(\varepsilon', k) + \delta_2$$

Remarque 1 :
Si la racine de l'erreur est équi-distribuée entre les classes, (*i.e* $\sigma(k)$ est constant), on retombe sur les mêmes résultats que dans ce cas classique de l'apprentissage i.i.d.

♦

Remarque 2 :
On pourrait regrouper les k points du premier jeu d'exemples et les k' du second au prix de l'utilisation supplémentaire de bornes de déviation uniforme. Nous ne détaillerons pas ce point qui n'apporte pas grand chose si ce n'est une complication des écritures.

♦

Démonstration :
D'après la définition 2.1, $\hat{f}$ est dans $F_{\varepsilon'}$ avec un niveau de confiance $1 - \delta(\varepsilon', k)$.
D'après le lemme 2.3, $\hat{f}$ est optimal avec une tolérance 3ε avec un niveau de risque

$$|F^{\varepsilon}_{\varepsilon'}|\delta'(\varepsilon, \Delta) + \delta(\varepsilon', k) \text{ avec } \Delta = \sup_{f; \hat{L}'(f) \leq \hat{L}'(h) + \varepsilon'/2} \text{Var } \hat{L}(f)$$

On en déduit que l'on converge vers l'optimum avec une précision 3ε au niveau de risque

$$|F^{\varepsilon}_{\varepsilon'}|\delta'(\varepsilon, \Delta') + \delta(\varepsilon', k) + \delta_2$$

■

Pour simplifier l'utilisation, on peut considérer l'algorithme B qui optimise sur F^{ε}_{∞} au lieu de $F^{\varepsilon}_{\varepsilon'}$. Il dépend également d'un niveau de risque $\delta_1 + \delta_2 + \delta_3$:

Algorithme B : La seule différence provient du 6^e point :
6. Soit ε tel que $|F^{\varepsilon}_{\infty}|\delta'(\varepsilon, \Delta') + \delta'(\varepsilon', \frac{1}{4}) \leq \delta_3$; Trouver $\hat{f} \in \text{argmin } \hat{L}$ parmi F^{ε}_{∞}.

Théorème 2.2 (Apprentissage sans optimisation sur les ε-nets)
Avec probabilité $1-\delta_1-\delta_2-\delta_3$, l'algorithme B est optimal avec une tolérance 3ε, i.e.

$$L(\hat{f}) \leq \inf_f L(f) + 3\varepsilon$$

Remarque :
La seule différence avec le théorème 2.1 est que $\hat{L}$ est optimisé sur F^{ε}_{∞} et que le risque d'avoir $\hat{f} \notin F^{\epsilon}_{\epsilon'}$ est intégré à δ_3.

♦

2.2.2 Stratification à l'aide d'une variable auxiliaire

On se place dans le cas où les données présentent une variable notée v qui peut-être qualifiée d'auxiliaire et qui est cencée d'après un expert permettre un partitionnement en plusieurs zones d'intérêt du problème. On peut penser par exemple à une variable « sexe » dans une étude sur des risques cardio-vasculaires[2].

On suppose donc dans cette partie que l'on a des données de la forme (x, v, y). Le but est de prédire y grâce à (x, v) (ou seulement grâce à x).

On définit la variance $\sigma_j(f)$ dans la classe j par : $\mathrm{Var}_{(x,v,y)} c_f(x, y)$ conditionnellement à $v = j$. La variable à apprendre y peut être nominale (classification) ou réelle (régression).

On considère l'algorithme C qui dépend d'un niveau de risque $\delta_1 + \delta_2 + \delta_3$. Ici, on utilisera des ε-nets avec une norme qui garantit $||f - g|| \leq \varepsilon \Rightarrow \sup |c_f - c_g| \leq \varepsilon$ (par exemple la norme $||.||_\infty$ dans la L_p régression).

Algorithme C :

1. Tirer $k = o(n)$ exemples ;
   Soit $\hat{L}'$ le coût empirique associé ; Soit ε' la précision telle que pour tout f optimal avec une tolérance $\varepsilon'/2$ pour $\hat{L}'$, f est aussi optimal pour L avec une précision ε' au niveau de confiance $1 - \delta(\varepsilon', k) \geq 1 - \delta_1$.
2. Tirer $k' = o(n)$ exemples (i.i.d.) pour estimer $\sigma_i(h)$ la racine de la variance du coût de h dans la classe i (avec $h = \mathrm{argmin}\, \hat{L}'$) ;
3. Calculer, grâce à l'aide des estimations de $\sigma_i(h)$, et avec la remarque de la définition 2.1, un vecteur $(\sigma'_i)_{i \in [\![1,K]\!]}$ tel que

$$\forall f; \hat{L}(f) \leq \hat{L}(h) + \varepsilon'/2 \Rightarrow \forall i; \sigma_i(f) \leq \sigma'(i)$$

[2]Notons que l'on n'est pas obligé de se limiter à une variable catégorielle, on aurait aussi pu utiliser une variable « poids » dont on sait qu'elle change les réactions de l'organisme.

avec un niveau de confiance $1 - \delta_2$.

4. Déduire de cette borne une formule $\Delta'(n_1, \ldots, n_K)$ telle que pour une fonction de perte empirique $\hat{L}$ associée à n_1 exemples dans la classe 1, n_2 exemples dans la classe 2,. . .,n_K exemples dans la classe K :

$$P(\exists f; L(f) \leq \inf_F \hat{L} + \varepsilon'/2 \wedge \text{Var } \hat{L}(f) > \Delta'(n_1, \ldots, n_K)) \leq \delta_2$$

5. Tirer n exemples, en respectant la répartition en classes optimisant Δ' ;
6. Soit ε tel que $|F^{\varepsilon}_{\varepsilon'}|\delta'(\varepsilon, \Delta') \leq \delta_3$,
7. Apprendre $\hat{f} \in \text{argmin}_{F^{\varepsilon}_{\varepsilon'}} \hat{L}$.

Théorème 2.3 (Borne non-asympotique pour l'apprentissage avec variable auxiliaire)
La probabilité $P(L(\hat{f}) \geq L^* + 3\varepsilon)$ *est bornée par* $|F_{\varepsilon'}{}^{\varepsilon}|\delta'(\varepsilon, \Delta') + \delta(\varepsilon', k) + \delta_2$.
avec $L^* = \inf_{f \in F} L(f)$

La preuve est essentiellement la même que celle du théorème 2.1.

Remarque 1 :
De même que précédemment, on peut faire une variante pour l'optimisation dans F^{ε}_{∞} au lieu de $F^{\varepsilon}_{\varepsilon'}$ (se référer à la différence entre les algorithmes A et B).

♦

Remarque 2 :
Si on utilise la norme L_∞, on obtient un résultat similaire pour la minimisation du risque empirique sans avoir besoin d'optimiser dans un ε-net.

♦

2.3 Stratification automatique

Les données ne présentent pas toujours une stratification naturelle détectée par les experts du domaine ; dans ce cas on peut tenter une stratification automatique ce qui correspond exactement à la problématique du *clustering* ; voir [Devroye et al.; 1997, chap 21] pour un panorama sur le partitionnement dépendant des données.

Cette section étend l'approche précédente au cas du *clustering*, et étudie spécifiquement leur convergence asymptotique, par opposition aux résultats non-asymptotiques présentés précedemment.

Comme dans la section précédente, pour simplifier les écritures, on notera $c_f(z)$ la variable aléatoire $(f(x) - y)^2$, z désignant la variable aléatoire (x, y). Remarquons que $Ec_f = L(f)$.

2.3.1 Algorithmes et théorèmes non-asymptotiques

La démarche suivie peut être résumée de la façon suivante :

- Une première partition est définie sur une première base d'apprentissage i.i.d.,
- La variance de la fonction de partitionnement est évaluée sur chaque sous ensemble, en utilisant un échantillon i.i.d. de ce sous ensemble,
- À partir des variances estimées, une stratification est définie,
- Une seconde base d'apprentissage, stratifiée, est considérée ;
- On apprend $\hat{f}$ sur cette nouvelle base.

Comme précédemment, l'algorithme présenté comporte un niveau de risque $\delta_1 + \delta_2 + \delta_3$. Soit (Π_n) une famille de partitions, indexée par le nombre n d'exemples. (on peut aussi considérer le cas où $|\Pi_n| = 1$).

On demande aux ε-nets d'utiliser une norme $||.||$ telle que

$$||f - g|| \leq \varepsilon \Rightarrow ((L(f) - L(g) \leq \varepsilon) \wedge (\hat{L}(f) - \hat{L}(g) \leq \varepsilon))$$

ce qui est vérifié par $||.||_\infty$ et par $||.||_1$ pour la régression L_∞ ou L_1.

Soit L une fonction de perte à valeurs dans $[0; 1]$.

On note $\sigma_{\pi,i}(f)$ la variance de la fonction de perte associée à la fonction de prédiction f dans le i^e ensemble de la partition Π (il s'agit d'un abus de notation car les sous-ensembles d'une partition ne présentent pas de numérotation naturelle, en fait n'importe quelle numérotation convient).

Algorithme D :

1. Tirer $k_1(\varepsilon, \delta_1)$ exemples.
2. Choisir $h \in F$ en minimisant le risque empirique.
3. Tirer $k_2(\varepsilon, \delta_2, n)$ exemples.
4. Évaluer $\hat{E}(c_h)^2 - (\hat{E}c_h)^2$ sur ces données pour la loi restreinte à χ, pour tout $\chi \in \pi$, et pour tout $\pi \in \Pi_n$.
5. Choisir la meilleure partition $\pi \in \Pi_n$ du point de vue de la variance globale de la fonction de perte associée à h.
6. Choisir les α_i tels que $\alpha_i n$ points seront tirés dans le i^e ensemble de la partition.
7. Tirer les n points
8. Calibrer $\hat{f}$ sur ces points par minimisation du risque empirique.

Théorème 2.4 (Borne non asymptotique avec stratification automatique)
On suppose $\mathcal{H}_1$ et $\mathcal{H}_2$:

- *$\mathcal{H}_1$: $k_1(\varepsilon, \delta)$ est tel que si au moins $k_1(\varepsilon, \delta)$ exemples (i.i.d.) sont tirés, alors $|\hat{L} - L| \leq \frac{\varepsilon}{2}$ avec probabilité au moins $1 - \delta$;*
- *$\mathcal{H}_2$: Pour tout $f \in F$, $k_2(\varepsilon, \delta_2, n)$ est tel que si au moins $k_2(\varepsilon, \delta_2, n)$ exemples (i.i.d.) sont tirés, alors $(E\chi c_f^2)/(E\chi)$ et $\frac{E\chi c_f}{E\chi}$ sont estimés avec précision ε, pour tout $\chi \in \Pi_n$, avec probabilité $1 - \delta_2$.*

On suppose de plus que les fonctions de pertes prennent leurs valeurs dans $[0, 1]$. On note $N(\eta)$ le nombre de couverture de F pour η.

Avec $\delta'(\varepsilon, \Delta) = 2 \exp\left(\frac{-\varepsilon^2}{4\Delta}\right)$, et tant que $0 \leq \varepsilon \leq 2 \min_i(n_i)Var(\hat{L}(f))$ alors la probabilité $P(L(\hat{f}) \geq \inf_f L(f) + 3\varepsilon')$ à la fin de l'algorithme D est majorée par :

$$\delta_1 + \delta_2 + N(\varepsilon')\delta'(\varepsilon', \text{Var } f^* + 4\varepsilon) + N(\varepsilon)\delta'(\varepsilon, \frac{1}{4})$$

ainsi que par (si l'on ne dispose pas de Varf^*) :

$$\delta_1 + \delta_2 + N(\varepsilon')\delta'(\varepsilon', \hat{\text{Var}}(h) + 2\varepsilon + 2C(n)) + N(\varepsilon)\delta'(\varepsilon, \frac{1}{4})$$

avec

$$C(n) = \sqrt{\varepsilon}\frac{4}{n}\sum_i \frac{P(\chi_i)^2}{\alpha_i}$$

Remarque 1 :
Le plus grand terme est $N(\varepsilon')\delta'(\varepsilon', \text{Var } f^* + 4\varepsilon)$, ce qui montre l'influence de Var f^* et le gain apporté par l'apprentissage actif. En effet, $\delta'(\varepsilon, \Delta) = 2 \exp\left(-\frac{\varepsilon^2}{4\text{Var}f^* + 4\varepsilon}\right)$ alors que traditionnellement la vitesse de convergence non-asymptotique est au mieux en $C \cdot S(F, n) \exp(-\frac{n\varepsilon^2}{32})$.

♦

Remarque 2 :
Les bornes indépendantes de la partition demandées à l'étape 4 de l'algorithme D peuvent être trouvées en utilisant la théorie de VC quand $|\Pi_n|$ est infini [Devroye et al.; 1997, chap 21].

♦

Démonstration :
On note $f^* = \text{argmin}_F L$. Soit χ_i le i^e sous ensemble de la partition définie par Π et $x_{(i)}$ une variable aléatoire suivant la loi des exemples restreinte à χ_i.

Étape 1 : Estimer la variance de la fonction de perte pour h avec une précision 2ε

En utilisant $\mathcal{H}_2$, on peut dire avec probabilité au moins $1-\delta_2$, que toutes les variances des fonctions de perte associées à h restreintes à $\chi \in \pi$ pour $\pi \in \Pi_n$ sont estimées avec une erreur au plus 2ε.

Étape 2 : Toutes les fonctions de perte empiriques de fonctions optimales avec une précision 2ε ont la même variance que $L(f^*)$ avec une précision $e = \sqrt{\varepsilon}\frac{4}{n}\sum_i \frac{P(\chi_i)^2}{\alpha_i}$

Toutes les fonctions de perte empiriques de fonctions f optimale pour L avec une précision ε ont la même variance que $L(f^*)$ avec une précision $e = \sqrt{\varepsilon}\frac{4}{n}\sum_i \frac{P(\chi_i)^2}{\alpha_i}$ car :

$$|\text{Var } \hat{L}(f) - \text{Var } \hat{L}(f^*)| \leq \frac{1}{n}\sum_i \frac{P(\chi_i)^2}{\alpha_i}|\text{Var } ((f(x_{(i)}) - y_{(i)})^2) - \text{Var } ((f^*(x_{(i)}) - y_{(i)})^2)| \tag{2.1}$$

$$|\text{Var } (f(x_{(i)}) - y_{(i)})^2) - \text{Var } (f^*(x_{(i)}) - y_{(i)})^2)|$$

$$\begin{aligned}
&= |E(c_f(z)^2) - (E(c_f(z)))^2 - E(c_{f^*}(z)^2) + (E(c_{f^*}(z)))^2)| && (2.2)\\
&= |E\left((c_f(z_{(i)}) + c_{f^*}(z_{(i)}))(c_f(z_{(i)}) - c_{f^*}(z_{(i)}))\right) \\
&\quad + \left((E(c_{f^*}(z_{(i)})) - E(c_f(z_{(i)})))(E(c_{f^*}(z_{(i)})) + E(c_f(z_{(i)})))\right) && (2.3)\\
&\leq |E((c_f(x_{(i)}, y_{(i)}) + c_{f^*}(x_{(i)}, y_{(i)}))(c_f(x_{(i)}, y_{(i)}) - c_{f^*}(x_{(i)}, y_{(i)})))| \\
&\quad + 2\varepsilon \sup(L) && (2.4)
\end{aligned}$$

Considérons le premier terme $|E((c_f(x,y) + c_{f^*}(x,y))(c_f(x,y) - c_{f^*}(x,y)))|$. Nous travaillerons indépendamment pour chaque valeur de i ; pour faciliter la lecture l'indice $_{(i)}$ sera donc omis. En utilisant l'inégalité de Cauchy-Schwartz on obtient :

$$|E((c_f(x,y) + c_{f^*}(x,y))(c_f(x,y) - c_{f^*}(x,y)))| \leq$$
$$(E((c_f(x,y) + c_{f^*}(x,y))^2))^{\frac{1}{2}} \cdot (E((c_f(x,y) - c_{f^*}(x,y))^2))^{\frac{1}{2}}$$

Puisque $c_f(x,y)$ est dans $[0,1]$, on peut borner le premier terme par 2 donc :

$$|E((c_f(x,y) + c_{f^*}(x,y))(c_f(x,y) - c_{f^*}(x,y)))| \leq 2 \cdot (E((c_f(x,y) - c_{f^*}(x,y))^2))^{\frac{1}{2}}$$

A ce niveau remarquons que $L_2^2(f-g) = L_2^2(f-f^*) + L_2^2(f^*-g)$, ce qui nous conduit à : $E(y-f(x))^2 - E(y-f^*(x))^2 = E(f-f^*)$ or comme f est optimal pour L avec une précision ε :

$$L_2^2(f-f^*) < \varepsilon \tag{2.5}$$

Revenant à l'inégalité principale il vient :

$$\begin{aligned} 2L_2(c_f - c_{f^*}) &= 2L_2((f-y)^2 - (f^*-y)) \\ &= 2\sqrt{E((f-f^*)^2)(f+f^*-2y)^2} \\ &\leq 2 \cdot 2E\sqrt{E(f-f^*)^2} \\ &\leq 4L_2(f-f^*) \\ &\leq 4\sqrt{\varepsilon} \text{ en utilisant l'inégalité (2.5)} \end{aligned} \tag{2.6}$$

En reportant ce résultat dans l'inégalité 2.1, on conclut la seconde étape.

Étape 3 : Borne de Chernoff : soit $f \in F_{\varepsilon'}$. Alors une déviation $\varepsilon \leq |\hat{L}(f) - L(f)|$ arrive avec une probabilité majorée par $\delta'(\varepsilon, \Delta)$.

en conséquence du lemme 2.2 avec

$$\delta'(\varepsilon, \Delta) = 2\exp\left(\frac{-\varepsilon^2}{4\Delta}\right)$$

et

$$\Delta \geq \sup_{F_{\varepsilon'}} \text{Var}(\hat{L}(.))$$

Étape 4 : Uniformité : une déviation $\varepsilon' \leq \sup_{F_\varepsilon} |\hat{L} - L|$ arrive avec probabilité au plus $N(\varepsilon')\delta'(\varepsilon', \Delta)$.

Soit $F_{\varepsilon'}$ un ε-net de F inclus dans F. Si $\Delta \geq \sup_{F_{\varepsilon'}} \text{Var}(\hat{L}(.))$,

- $P(\sup_{F_\varepsilon \cap F} |\hat{L} - L| > \varepsilon')$ est majoré par $N(\varepsilon')\delta'(\varepsilon', \Delta)$;
- $\sup_{f \in F_\varepsilon} |L(f) - L(f')| \leq \varepsilon'$ avec f' une des fonctions de l'ε'-net a distance au plus ε' de f.
- $\sup_{f \in F_\varepsilon} |\hat{L}(f) - \hat{L}(f')| \leq \varepsilon'$

On en déduit que la minimisation du risque empirique (sans la restriction à un ε-net) est optimale avec une précision $3\varepsilon'$ avec probabilité au plus $N(\varepsilon')\delta'(\varepsilon', \Delta)$, à condition que $\hat{f}$ soit optimale avec une précision ε et que Δ soit une borne sup de la variance des fonctions de perte associées aux fonctions optimales avec précision ε.

Étape 5 : Conclusion

Dans cette étape, on notera Var$f$ la variance de la fonction de perte empirique associée à une fonction f.

En reprenant les Étapes 1 et 2, il vient :

$$|\hat{\text{Var}}(h) - \text{Var}(h)| \leq 2\varepsilon, \text{ et } |\text{Var}(f^*) - \text{Var}(h)| \leq e$$

pour tout f tel que :

$$L(f) - L(f^*) \leq \varepsilon \text{ , } |\text{Var}(f) - \text{Var}(h)| \leq 2e \text{ et } |\text{Var}(f) - \text{Var}(f^*)| \leq e$$

Donc $\Delta = \hat{\text{Var}}(h) + 2\varepsilon + 2e$, est une borne sup sur toutes les variances des fonctions de perte associées aux fonctions optimales avec précision ε.

Donc avec probabilité au moins $1 - \delta_1 - \delta_2 - N(\varepsilon')\delta'(\varepsilon', \text{Var } f^*)$, $\hat{f}$ est soit non-optimale avec précision ε ou optimale avec précision $3\varepsilon'$.

Avec probabilité au moins $1 - N(\varepsilon)\delta'(\varepsilon, \frac{1}{4})$, $\hat{f}$ est optimale avec une précision ε. Donc avec probabilité au moins $1 - \delta_1 - \delta_2 - N(\varepsilon')\delta'(\varepsilon', \text{Var } f^*) - N(\varepsilon)\delta'(\varepsilon, \frac{1}{4})$, on a le fait que f est optimale avec précision $3\varepsilon'$.

De plus, avec probabilité au moins $1-\delta_1-\delta_2-N(\varepsilon')\delta'(\varepsilon', \hat{\text{Var}}(h)+6\varepsilon)-N(\varepsilon)\delta'(\varepsilon, \frac{1}{4})$, f est optimale avec précision $3\varepsilon'$

■

Remarque :
La précision est plus où moins divisée par le ratio entre l'ancien et le nouvel écart type. **La borne est meilleure que celle de l'apprentissage passif si la moyenne des variances locales est plus faible que la variance globale**.

♦

2.3.2 Version asymptotique

Cette section traite de résultats asymptotiques présentant les deux avantages suivants :
- Ils considèrent la minimisation du risque empirique sur F, au lieu d'une minimisation sur un ε-net[3].
- Ils fournissent des estimations asymptotiques plus serrées que le supremum des déviations. On montre que leurs estimations sont **équivalentes** au véritable $\sup_f |\hat{L}_f - L_f|$.

Éléments pour la version asymptotique

Présentons tout d'abord l'intuition de la preuve qui sera formalisée par la suite.

[3]Cependant, comme signalé, la preuve précédente pourrait être améliorée pour faire la même chose

Soit $X_1, \ldots, X_k$ une partition mesurable de X. Notons E_i l'opérateur espérance pour la loi P restreinte à X_i. On définit $\hat{P}_i$ une loi empirique associée à un échantillon i.i.d. de la loi P restreinte à X_i contenant n_i exemples et $\hat{E}_i$ est l'opérateur d'espérance associé. Le j^e exemple de la strate X_i est noté $(x_{i,j}, y_{i,j})$. Par commodité, on notera parfois $y = g(x)$ pour montrer la dépendance de y par rapport à x. Dans ce cas, g est la fonction à approximer. On suppose que $n_i = \alpha_i n$ avec $\alpha_i > 0$ et $\sum_i \alpha_i = 1$. On note $L_i(f)$ et $\hat{L}_i(f)$ respectivement $E_i(f-g)^2$ et $\hat{E}_i(f-g)^2$. Remarquons que $L_i(f)$ est une variable aléatoire car g, la fonction objectif, n'est pas déterministe.

Sous certaines hypothèses, en particulier si F est $\mathcal{P}$-Donsker - ce qui est notamment le cas si la VC dimension est finie -, on a :

$$\sqrt{n}(\hat{L}_i(f) - L_i(f))_f \rightarrow \sqrt{1/\alpha_i} PG_i$$

au sens de la convergence faible, avec PG_i un processus gaussien presque sûrement borné. Notons σ_i le vecteur (indexé par F) des variances de PG_i.

En sommant de manière pondérée par les $P(X_i)$, on obtient :

$$\sqrt{n}(\sum_i P(X_i)(\hat{L}_i(f) - L_i(f)))_f \rightarrow \sum_i \sqrt{1/\alpha_i} P(X_i) PG_i$$

Le vecteur indexé par F des variances σ^2 de $PG = \sum_i \sqrt{1/\alpha_i} P(X_i) PG_i$ est alors :

$$\sigma^2(f) = \sum_i \frac{1}{\alpha_i} P(X_i)^2 \sigma_i(f)^2$$

Sous certaines conditions (que nous détaillerons ensuite), les fonctions sont proches de f^*. Reste alors à minimiser

$$\sum_i \frac{1}{\alpha_i} P(X_i)^2 \hat{\sigma}_i(f)^2$$

vu comme une fonction des α_i avec la contrainte $\sum_i \alpha_i = 1$ avec f dans le voisinage de f^*.

Ceci conduit à :

$$\alpha_i \simeq \frac{\sigma_i(f^*) P_i}{\sum_j P(X_j) \sigma_j(f^*)}$$

qui dépend de f. Nous verrons que $\hat{f}$ peut être utilisée au lieu de f sans grosses difficultés.

Il vient $\sigma^2 \simeq (\sum_i P(X_i)\sigma_i(f^*))^2$, et à la limite d'un grand nombre de strates, on obtient :

$$\sigma^2 \simeq \left(E_x \sqrt{(\mathrm{Var}_y(f(x) - y)^2)} \right)^2$$

où l'opérateur d'espérance porte sur x (de loi P) et l'opérateur de variance porte su y, conditionnellement à x. La méthode n'utilisant qu'une strate conduit à $\sigma^2 \simeq \mathrm{Var}_{x,y}((f(x) - y)^2)$. En résumé, cette méthode **supprime la variance inter-strates, conservant seulement la part intra-strates**.

Algorithme et théorème asymptotique

Nous allons traiter le cas où nous ne voulons pas qu'il subsiste une optimisation sur un ε-net. Nous montrerons que des résultats de convergence peuvent également être établis. Les couples (x, y) sont tirés selon la loi P ou des restrictions de cette loi. Soit l'algorithme suivant (dépendant de n, p et k) :

Algorithme E :

1. Tirer $k = o(n)$ exemples en suivant la loi P ;
2. Apprendre $f_1 \in F$ sur les k exemples ;
3. Séparer en p strates en optimisant $\sum_i (P(X_i)^2/\alpha_i)\hat{\text{Var}}_i\, c_f$ avec $\hat{\text{Var}}_i(c_f)$ la variance estimée par $\hat{E}(\chi_i \times c_f)/P(X_i)$ où χ_i est la fonction caractéristique de X_i, sous la contrainte $\alpha_i \geq \alpha_0(p)$ (utilisée pour imposer un minima d'exemples par strates).
4. Tirer n_i exemples par strate avec la loi P restreinte à X_i ;
5. Apprendre f_2 par minimisation du risque empirique sur les $n = \sum n_i$ exemples.

Tous les tirages de l'étape 1 sont indépendants. Conditionnellement à l'étape 1, tous les tirages de l'étape 5 sont indépendants.

Théorème 2.5 *On s'intéresse à l'algorithme E avec les hypothèses suivantes :*

1. $\text{argmin}\, L \in F$ *;*
2. *La partition est choisie de façon à ce que tous les X_i sont dans une famille $\mathcal{P}_p$ dépendant de p et telle que les produits $\chi_p \times l$ et $\chi_p \times l^2$ pour l une fonction de perte associée à une fonction dans F soient Glivenko-Cantelli pour la convergence presque sûre.*
3. *La partition est choisie pour garantir que $P(X_i) \geq k(p)$ ou $k(.)$ est une fonction positive décroissante.*
4. *$(f(x) - y)^2$ est majoré par 1 (e.g., toutes les fonctions sont à valeurs dans $[0, 1]$).*
5. *Les fonctions de perte associées à F forment une classe de Glivenko-Cantelli pour la convergence presque sûre.*
6. *$\alpha_i \geq \alpha_0(p) \geq 0$;*
7. *On dispose d'une suite de partitions $(X^k{}_1, \ldots, X^k{}_k) \in \mathcal{P}_k$ (la k^e partition est de taille k) et telle que $\lim_k \sum_{i=1}^{k} P(X^k{}_i)\sqrt{\text{Var}_i(c_f^*)} = E_x\sqrt{\text{Var}_y c_f^*}$.*

Alors

$$\text{limsup}_{s\to\infty} \frac{\log(\frac{1}{2}P(L(f_2) > L^* + s/\sqrt{n}))}{s^2} \leq \frac{-1}{\sigma'^2}$$

et pour tout $\varepsilon > 0$, pour s assez grand :

$$P(\sup_{f\in F} |\hat{L}(f) - L(f)| > s/\sqrt{n}) \leq 2\exp(\varepsilon s^2 - \frac{s^2}{2\sigma'^2})$$

Ce qui peut aussi s'écrire pour tout $(2\sigma^2)^{-1} > \varepsilon > 0$ *avec le changement de variables* $s = \sqrt{\dfrac{-\log(\delta/2)}{1/2\sigma'^2 - \varepsilon}}$ *si* δ *est assez petit et* n *suffisamment grand :*

$$P(\sup_{f\in F} |\hat{L}(f) - L(f)| > \sqrt{\frac{-\log(\delta/2)/(1/(2\sigma'^2) - \varepsilon)}{n}}) \leq \delta$$

avec

$$\lim_{p\to\infty} \lim_{k\to\infty} \sigma'^2 = E_x Var_y (f^*(x) - y)^2$$

Où l'opérateur d'espérance porte sur x *(de loi* P*) et l'opérateur de variance sur* y*, conditionnellement à* x*.* f^* *est optimal au sens des moindres carrés (nous avons supposé que* f^* *est dans* F*).*

Remarque 1 :
Les hypothèses sont assez lâches. En effet les conditions de Glivenko-Cantelli et de Donsker sont beaucoup moins restrictives que des hypothèses à la VC. Le résultat est par contre « très » asymptotique (limite pour p, k et n). En particulier à cause de l'inégalité de Borell, nous n'étudions en fait que de « grandes » déviations.

♦

Remarque 2 :
Un second point important est que la variance du processus gaussien est réduite grâce à la stratification, au moins pour les fonctions proches de la fonction optimale. On peut supposer qu'il est possible d'obtenir des résultats « moins » asymptotiques. La méthode en une strate conduit à un résultat similaire avec :

$$\lim \sigma'^2 \to \text{Var}_{x,y}(f^*(x) - y)^2$$

La différence entre les deux termes ressemble à la différence entre un écart type et un écart type intra classes.

♦

Démonstration :
On note $z = (x, y)$, $c_f(z) = (f(x) - y)^2$. Remarquons que $E(c_f) = L(f)$

Étape 1 : f_1 **et** f_2 **sont dans** F_ε **si** k **et** n **sont assez grands.**

Avec probabilité 1,

$$L(f_1) \leq L^* + \varepsilon(k)$$

avec $\varepsilon(k) \to 0$, et $L^* = \inf_{f\in F} L(f)$, car F est une classe de Glivenko-Cantelli pour la convergence presque sûre.

Si n est suffisamment grand alors,

$$L(f_2) \leq L^* + \varepsilon(k)$$

avec $\varepsilon(k) \to 0$, et $L^* = \inf_{f \in F} L(f)$, car F est une classe de Glivenko-Cantelli pour la convergence presque sûre.

Notons F_ε l'ensemble de fonctions $f \in F$ telles que $L(f) \leq L^* + \varepsilon$.
Alors $f_1 \in F_\varepsilon$ et $f_2 \in F_\varepsilon$.

Étape 2 : Évaluation de la variance dans F_ε

On veut évaluer

$$\begin{aligned} \text{Var } \hat{L}(f) &= \text{Var } (\sum_i P(X_i)\hat{L}_i(f)) \\ &= \sum_i P(X_i)^2 \text{Var } \hat{L}_i(f) \\ &= \sum_i P(X_i)^2 \frac{1}{(\alpha_i n)^2}(\alpha_i n)\text{Var } ((f(x)-y)^2) \end{aligned}$$

pour la loi P restreinte à X_i

$$\text{Var } \hat{L}(f) = \sum P(X_i)^2 \frac{1}{n_i^2} \sum_j \text{Var } ((f(x_{(i)}) - y_{(i)})^2)$$

avec $(x_{(i)}, y_{(i)})$ variable aléatoire suivant la loi de (x, y) restreinte à X_i

$$\text{Var } \hat{L}(f) = \sum \frac{P(X_i)^2}{n\alpha_i} \text{Var } (c_f(z_{(i)}))$$

Notons $K = \sum \frac{P(X_i)^2}{\alpha_i} \text{Var}_i \; c_{f^*(z)}$.

Lemme 2.4 *Si $f \in F_\varepsilon$ alors pour la loi restreinte à X_i,*

$$|\textit{Var } \hat{L}(f) - \textit{Var } \hat{L}(f^*)| \leq e = \sqrt{\varepsilon}\frac{4}{n} \sum_i \frac{P(X_i)^2}{\alpha_i}$$

Démonstration :
Voir l'Étape 2 du théorème 2.4

■

Alors

$$\forall f \in F_\varepsilon, \; n\text{Var } \hat{L}(f) \leq K \pm 4\sqrt{\varepsilon} \sum \frac{P(X_i)^2}{\alpha_i}$$

On obtient également :

$$\lim_{k\to\infty}\lim_{n\to\infty}\sup_{f\in F_{\varepsilon(k)}}|n\text{Var }\hat{L}(f)-K|=0$$

Étape 3 : La limite du processus empirique $\sqrt{n}(L-\hat{L})_{f\in F}$ et sa restriction à F_ε

$$\sqrt{n}(\hat{L}-L)_f\to_n PG \text{ et } \sup_{f\in F_{\varepsilon(k)}}\text{Var } PG_f=K+o_k(1)$$

où $o_k(1)\to 0$ quand $k\to\infty$. Nous verrons plus tard l'intérêt de la constante K dans l'utilisation de l'inégalité de Borell.

Étape 4 : Précision de l'évaluation de α_i

Notre but final est de montrer que l'on peut estimer précisément K. Nous allons commencer par nous intéresser au lemme suivant :

Lemme 2.5 *Le supremum sur $(X_1,\ldots,X_p)\in\mathcal{P}$ et $f\in F$ et $(\alpha_1,\ldots,\alpha_p)$ de*

$$|\sum_i(P(X_i)^2/\alpha_i)\hat{\textit{Var}}_i\ c_f-\sum_i(P(X_i)^2/\alpha_i)\textit{Var}_i\ c_f|$$

converge vers 0 presque sûrement quand $k\to\infty$,
où $\hat{\textit{Var}}_i(w)$ est un estimateur de la variance $\hat{E}_i((w-\hat{E}_iw)^2)$ et $\hat{E}_i$ est une espérance empirique estimée grâce à $\hat{E}(\chi_i\times w)/P(X_i)$ où χ_i est la fonction caractéristique de X_i.

Démonstration :
On doit s'interroger sur la convergence de $\hat{\text{Var}}_i\ c_f$ vers $\text{Var}_i\ c_f$.

Or f appartient à une famille F qui est supposée être Glivenko-Cantelli pour la convergence presque sûre. Nous avons vu dans la section 1.5.4 que dans ce cas on avait la convergence presque sûre vers 0 des écarts de la moyenne empirique à l'espérance de $f(X_i)$.

Les coefficients de la fonctionnelle sont tous des combinaisons linéaires d'éléments inclus dans une classe que nous avons supposé être une classe de Glivenko-Cantelli pour la convergence presque sûre. On a donc la convergence vers 0 du PG indicé par F vers $(0,\ldots,0)$. Donc en particulier du *supremum*.

■

Étape 5 : application de l'inégalité de Borell

En appliquant l'inégalité de Borell, avec $\sigma'=\sup_{f\in F_\varepsilon}\sigma_f$, on obtient :

$$\lim_{s\to\infty}\frac{\log(\frac{1}{2}P(\sup_{f\in F_\varepsilon}|\hat{L}(f)-L(f)|>s/\sqrt{n}))}{s^2}=-1/(\sigma'^2)$$

Que nous pouvons réécrire en :

$$\text{limsup}_{s\to\infty}\frac{\log(\frac{1}{2}P(L(f_2) > L^* + s/\sqrt{n}))}{s^2} \leq -1/(\sigma'^2)$$

et pour tout ε positif, pour s assez grand, on a :

$$P(\sup_{f\in F}|\hat{L}(f) - L(f)| > s/\sqrt{n}) \leq 2\exp(\varepsilon s^2 - \frac{1}{2}s^2/\sigma'^2)$$

En effectuant le changement de variable $s = \sqrt{\frac{-\log(\delta/2)}{1/2'^2 - \varepsilon}}$, pour tout $(2\sigma^2)^{-1} > \varepsilon > 0$, si δ est assez petit, si n est assez grand,

$$P(\sup_{f\in F}|\hat{L}(f) - L(f)| > \sqrt{\frac{-\log(\delta/2)/(1/(2\sigma'^2) - \varepsilon)}{n}}) \leq \delta$$

Étape 6 : La limite quand $p \to \infty$

Pour $p \to \infty$,

$$\lim_{p\to\infty}\lim_{k\to\infty}\sigma'^2 \to (E_x\sqrt{\text{Var}_y((f^*(x) - y)^2)})^2$$

Où l'opérateur d'espérance porte sur x (de loi P) et l'opérateur de variance sur y, conditionnellement à x. f^* est optimal au sens des moindres carrés (nous avons supposé que f^* est dans F). La méthode à une strate conduit à un résultat identique avec :

$$\lim\sigma'^2 \to \text{Var}_{x,y}((f^*(x) - y)^2)$$

■

2.4 Expérimentations

Le but de cette section est de voir si les améliorations portant sur les bornes de vitesse de convergence se répercutent dans la pratique. Le principe est de comparer un apprentissage effectué avec la distribution naturelle à un apprentissage fait avec la distribution active.

2.4.1 Jeux de données artificielles

Données stratifiées *a priori*

Le protocole utilisé est le suivant :

- Les données sont constituées de quatre classes équiprobables ;
- La sortie de la i^e classe est une gaussienne de moyenne dépendant de la classe et de variance proportionnelle à i.
- On détermine la proportion d'exemples dans chaque classe de deux façons différentes :
  1. équirépartition ($1/4$ dans chaque classe) ;
  2. répartition optimale vue précédemment.

- Tirer n exemples selon la répartition choisie ;
- Apprendre par minimisation empirique de l'erreur L_2 ;
- Analyser analytiquement la différence entre l'erreur en généralisation et la meilleure erreur en généralisation possible.

Les résultats sont une moyenne des différences entre l'erreur en généralisation obtenue et l'erreur analytique sur $10\ 000$ essais indépendants.

Les données présentent donc une stratification naturelle qui est leur classe. De plus les classes ne présentent pas toutes la même variance (condition nécessaire pour que cette approche soit potentiellement plus intéressante qu'un apprentissage passif). Le coté actif de l'apprentissage consiste ici à choisir le nombre d'exemples que l'on souhaite dans chaque classe. On espère que la répartie optimisée en fonction de l'estimation de la variance dans chacune des classes va conduire à de meilleurs résultats pour l'erreur en généralisation.

En fait, le théorème correspondant (2.1) établit de manière non-asymptotique une meilleure borne sur la probabilité d'avoir un écart grand entre $L(\hat{f})$ (le cout de la fonction choisie) et $L(f^*)$ (le coût optimal).

nombre d'exemples	% des cas ou la stratification est meilleure que l'équirépartition	écart type de l'erreur en généralisation pour le cas équiprobable	écart type de l'erreur en généralisation pour la stratification
20	0.54390	0.41019	0.36924
25	0.53615	0.41340	0.38329
30	0.53947	0.41769	0.38488
35	0.53787	0.42213	0.39431
40	0.53754	0.42527	0.39546
45	0.53593	0.42826	0.40077
50	0.53553	0.43241	0.40188

Interprétation : L'erreur en généralisation est améliorée par la stratification dans une faible proportion des cas (0.54), l'écart se réduisant avec le nombre d'exemples. Mais comme annoncé par le théorème 2.1, la variance de l'erreur en généralisation est diminuée par la stratification. Notons également que si l'on augmente les différences de variances entre de chacune des classes, l'écart entre l'apprentissage actif et passif s'accroît (et est nul si les variances sont partout égales).

Stratification automatique

On utilise un coefficient p valant arbitrairement de 1 à 5 pour paramétrer la fonction objectifs. La tâche objectif est une classification pour $x \in [0, 1]$. La classe de x est 1 avec une probabilité $\frac{1}{2} + p(x - \frac{1}{2})$ (tronquée entre 0 et 1 si nécessaire) et 0 sinon.

La méthode naïve utilise $n + k$ exemples, la stratification est basée sur $k = 200$ points (utilisés à la fois pour apprendre une première fonction et pour évaluer la variance) et $n = 5000$.

La stratification est faite en partitionnant automatiquement le domaine en 3 sous-espaces connexes. L'apprentissage est réalisé par la technique des moindres carrés.

Dans plus de la moitié des expériences, la stratification est meilleure ou équivalente. Les cas ou celle ci se fait battre devenant de moins en moins nombreux avec l'augmentation de p. On voit donc ici l'intérêt d'avoir le maximum d'informations *a priori* sur nos données. En effet, dans certains cas, l'apprentissage ne présente pas d'intérêt.

p	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
% stratification meilleure que méthode naïve	44	50.5	52	44	46.5
% *ex aequo*	11	12.5	23.5	29.5	30
% stratification pire que méthode naïve	45	37	24.5	26.5	23.5

paramètre p	% des cas ou l'erreur médiane dans l'apprentissage stratifié est meilleure que l'apprentissage non-stratifié
$p = 1$	51.25 %
$p = 2$	63 %
$p = 3$	53.5 %
$p = 4$	66.5 %
$p = 5$	76.75 %

Avec 3 strates et $p = 1$, l'apprentissage stratifié automatiquement est meilleur dans 44% des cas, équivalent dans 11% et pire dans 45%. Avec $p = 5$, la stratification est meilleure dans 46.5%, équivalente dans 30% et pire dans 23.5%. Avec $p = 5$, dans 76.5% des cas l'apprentissage sans stratification est pire que le résultat médian de l'apprentissage stratifié.

Interprétation : Le résultat est un peu décevant pour les petites valeurs de p. L'algorithme est plus complexe que la méthode naïve et ne produit pas de meilleurs résultats.

Son principal avantage est sa stabilité : la proportion de très mauvais cas est plus faible. De manière prévisible, les résultats s'améliorent quand p grandit.

2.4.2 Expérience avec des données réelles.

Les données proviennent d'analyse du trafic routier. Nous utiliserons le protocole suivant :

- Estimer la variance de la fonction de perte par strate ; il y a sept strates qui sont des « types » de jour (lundi, mardi, ...),
- Choisir le nombre de points par strate comme expliqué précédemment (nombre de points par type de jour proportionnel au produit écart type $\times$ probabilité).
- Apprendre.

L'objectif est de prédire le taux d'occupation sur une route. Les résultats sont moyennés en choisissant aléatoirement plusieurs 5-*cross validations* : pour chacun des cinq apprentissages des validations croisées, l'ensemble d'apprentissage est si nécessaire sous-échantillonné pour que l'apprentissage soit toujours effectué selon la distribution de la stratification optimale (*i.e* on sélectionne seulement certains exemples d'apprentissage pour coller à la distribution).

L'apprentissage est réalisé comme suit :

- les catégories sont définies par l'utilisateur (ces catégories sont définies par un expert et sont des informations sur l'appartenance du jour à un type de comportement : we prolongé, jour férié, vacances ...).
- La prédiction dans la catégorie k est la moyenne de l'ensemble d'apprentissage restreint à la catégorie k.

Les résultats, moyennés sur plusieurs essais sont les suivants :

Allocation	erreur moyenne	écart type
naïve	466.6	66.8
optimale	462.5	72.7

Les résultats sont meilleurs en moyenne, mais la variance a augmenté. Remarquons que la somme erreur+écart type est a peu près la même dans les deux cas. Ce n'est pas exactement ce qui est prédit par la théorie, il est possible que la démarche basée sur des *cross-validations* biaise les résultats.

2.5 Conclusion

Nous avons traité dans ce chapitre de l'apprentissage supervisé optimisé par la stratification. Les différents cas considérés sont :

– Stratification avec des données présentant naturellement des strates (dont la classification). Dans cette partie, nous avons montré dans le cas non-asymptotique des vitesses de convergence améliorées et de meilleures bornes sur l'erreur en généralisation pour des algorithmes minimisant le risque empirique sur des ε-nets.
– La stratification avec une construction automatique des strates. Nous avons montré à la fois une amélioration non-asymptotique dans les bornes de l'erreur en généralisation ainsi qu'une vitesse de convergence améliorée pour les algorithmes minimisant l'erreur empirique sur des ε-nets et asymptotiquement, une convergence améliorée pour des hauts niveaux de risque.

De plus nous observons une légère amélioration des résultats en pratique, ce qui n'est pas souvent le cas quand on travaille avec des bornes…

Notons également que notre théorème asymptotique (2.5) utilise des hypothèses très faibles ; l'algorithme est standard, la stratification facilement implémentable et s'applique à des classes de Glivenko-Cantelli et de Donsker, qui sont des familles de fonctions que l'on rencontre bien plus souvent dans la vraie vie que des familles ayant une VC-dim finie. De plus leur nombre de couverture peut-être exponentiel. Cependant il souffre de deux défauts :

– Les résultats sont « très » asymptotiques ;
– Les résultats non-asymptotiques reposent sur des optimisations sur des ε-nets.

Chapitre 3

Apprendre grâce à des suites déterministes

Ce chapitre s'intéresse à un autre aspect de l'apprentissage actif, à savoir la détermination de la distribution utilisée pour la construction de la base d'apprentissage. Cette problématique est voisine de la notion de plan d'expériences et nous discuterons la différence. Concrètement, ce chapitre présente de nouvelles bornes sur l'erreur en généralisation et des vitesses de convergence, pour les cas :

- D'une distribution fondée sur la minimisation de la discrépance ;
- De points issus d'un ε-net, et modifiés dans le but d'accroître la résistance au bruit de l'apprentissage.

3.1 Introduction et notations

Ce chapitre s'intéresse à l'approximation de fonctions déterministes en présence ou non de bruit.

Dans la suite, P est une loi de probabilité sur un espace X et F est une famille de fonctions de $X \to \mathbb{R}$.

Comme dans le chapitre précédent, $\hat{P}_n$ désigne la loi empirique associée à la distribution P appliquée à n exemples supposés indépendant identiquement distribués (sauf mention explicite du contraire). $\hat{E}_n$ est l'espérance associée à $\hat{P}_n$. Considérant la fonction de coût quadratique, si g désigne le concept cible à apprendre, l'erreur empirique d'une hypothèse f est notée $\hat{L} = \hat{E}(f-g)^2$.

Pour faciliter la lecture, rappelons les définitions relatives à l'inégalité de Koksma-Hlawaka (présentée section 1.8.3), pour les fonctions BHKV (section 1.8.2). Cette inéga-

lité fournit essentiellement une borne sur l'erreur de l'approximation d'une intégrale par des techniques de Quasi-Monte-Carlo[1]. Cette majoration de l'erreur fait appel à la notion de discrépance. Celle-ci mesure le biais d'une suite aléatoire.

Définition 3.1 (Star discrépance) *La star-discrépance de la suite (x_i) de $[0,1]^d$ est définie par :*

$$D_n^*(X) = \sup_{a\in[0;1]^d} \left| \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}_{\{x_k\in[0;a]\}} - \prod_{i=1}^{d} a_i \right|$$

Théorème 3.1 (Koksma-Hlawka) *(Voir [Niederreiter; 1992]) Si f est une application de $[0,1]^d$ dans $\mathbb{R}$, de variation totale au sens de Hardy et Krause majorée par $V_{HK}(f)$, et si x_i est une suite à discrépance faible dans $[0,1]^d$, alors :*

$$\left| \frac{1}{n}\sum_{i=1}^{n} f(x_i) - \int f \right| \leq V_{HK}(f) D_n^*(x_1,\ldots,x_n)$$

Notons que cette inégalité a déjà été utilisée dans le cadre de l'apprentissage actif par [Cervellera and Muselli; 2004]. Nous discuterons de la portée des résultats présentés par rapport à leurs travaux.

3.2 Apprentissage de fonctions

L'objet de cette section est **d'apprendre des fonctions** en utilisant **des suites à faible discrépance**.

3.2.1 Approche théorique

Soit $X = [0,1]^d$.

Théorème 3.2 (Corollaire de l'inégalité de Koksma-Hlawka) *Soit g une fonction de X dans $\mathbb{R}$, et F un espace de fonctions tel que $V_{HK}((f-g)^p) \leq M$ pour tout f dans F, où V_{HK} désigne la variation au sens de Hardy et Krause et p est un entier positif. Soit $\hat{L}(f)$, pour $f \in F$, la moyenne de $|f(x_i) - g(x_i)|^p$ pour i de 1 à n.*

Pour tout f dans F, si on note $D_n^*(x_1,\ldots,x_n)$ la star-discrépance de la suite x_i on a :

$$|\hat{L}(f) - L(f)| \leq M D_n^*(x_1,\ldots,x_n)$$

et pour $\hat{f} = \operatorname{argmin} \hat{L}(f)$,

$$L(\hat{f}) \leq \inf_f L(f) + 2M D_n^*(x_1,\ldots,x_n) \tag{3.1}$$

[1]Dans un Quasi-Monte-Carlo la distribution uniforme de Monte-Carlo utilisée pour échantillonner est remplacée par une suite déterministe

En supposant que x_i ***est une suite à faible discrépance,*** *la différence entre* $\hat{L}(f)$ *et* $L(f)$*, est majorée par* $Mc(d)\log(n)^d/n$ *uniformément en* $f \in F$ *avec* $c(d)$ *désignant une « constante » ne dépendant que de la dimension* d *de* X
Si de plus $\hat{f} = \text{argmin}\, \hat{L}(f)$*, on a :*

$$L(\hat{f}) \leq \inf_{f \in F} L(f) + 2Mc(d)\log(n)^d/n \tag{3.2}$$

Les hypothèses de ce théorème permettent comme dans le cas classique de l'apprentissage statistique de borner la différence entre l'erreur en généralisation de l'hypothèse apprise et la meilleure erreur en généralisation possible. La différence est double :

- D'une part la régularité de l'espace d'hypothèse, classiquement mesurée en termes de VC-dim intervient ici en terme de variation de Hardy et Krause,
- D'autre part, et surtout, un autre terme apparaît qui mesure la régularité de la base d'apprentissage, plus précisément, sa discrépance.

Démonstration :
Par définition de $\hat{f}$, pour tout $f \in F$

$$\hat{L}(\hat{f}) \leq \hat{L}(f)$$

et par l'inégalité de Koksma-Hlawka :

$$\hat{L}(f) \leq L(f) + MD_n^*(x_1, \ldots, x_n)$$

Ainsi

$$\begin{aligned} \forall f\ L(\hat{f}) &\leq \hat{L}(\hat{f}) \\ &\leq \hat{L}(f) + MD_n^*(x_1, \ldots, x_n) \\ &\leq L(f) + 2MD_n^*(x_1, \ldots, x_n) \end{aligned}$$

■

Remarque 1 :
Il existe des outils permettant de calculer la star-discrépance d'une suite, voir par exemple [Thiedmard; 2001], ce qui rend l'inégalité directement exploitable.

♦

Remarque 2 :
Ce résultat peut aussi s'appliquer à des suites non déterministes comme suit :

- Récupérer des données aléatoires ;
- Calculer leur star-discrépance ;

– Estimer une borne sur M et sur la variation totale de $f - g$. On pourra par exemple prendre le double de M si g a une variation totale bornée par M et que l'on utilise F, ensemble de fonctions à variation totale bornée par M pour approcher g.
– Utiliser les points précédents pour effectuer l'appentissage et utiliser leur discrépance pour obtenir une borne sur l'erreur.

♦

Remarque 3 :
Notons que [Niederreiter; 1992] montre que la discrépance d'une suite aléatoire simple uniforme est plus mauvaise (en $1/\sqrt{n}$) que celle d'une suite à faible discrépance (en $O(\log(n)^d/n)$). Ce résultat suggère ainsi que le fait de remplacer le tirage uniforme des données d'apprentissage par une suite à faible discrépance améliore la borne.

♦

Théorème 3.3 (Minimisation du risque structurel avec variation totale finie) *Soit g une fonction de $X \to \mathbb{R}$ et F une famille de fonctions de $X \to \mathbb{R}$. On suppose que :*

$$\forall f \in F,\ V_{HK}(|f - g|) \text{ est finie.}$$

On note $L(f) = E(|f - g|)$ et $\hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} |f(x_i) - g(x_i)|$.

Considérons la fonction de coût L, avec $L^* = \inf_{f \in F} L(f)$ et $f^* \in \operatorname{argmin} L$ (ou une fonction arbitrairement proche du minimum de L si celui-ci n'est pas atteint).

Supposons que $\hat{f}$ optimise $\hat{L}(f) + c(d)V_{HK}(f)\log(n)^d/n$ avec une précision ε. Les points utilisés pour effectuer l'apprentissage sont choisis selon une suite à faible discrépance.

Alors :

$$L(\hat{f}) \leq L^* + \varepsilon + (V_{HK}(f^*) + V_{HK}(g) + V_{HK}(f^* - g))\log(n)^d c(d)/n$$

Par rapport au théorème précédent, le théorème 3.3 s'affranchit de la restriction $V_{HK}(|f - g|^p) < M$. En effet dans la pratique, cette borne n'est souvent pas facile à estimer (à moins de fortes hypothèses sur l'espace de fonction et sur la fonction objectif). Par contre il semble raisonnable de supposer qu'elle est finie. Il reste bien sûr des variations dans la majoration finale mais celles-ci ne portent plus que sur f^* et g et non sur tous les éléments de F.

Démonstration :
Soit $R(n, h) = V_{HK}(|h|)\log(n)^d c(d)/n$.

Nous avons les trois inégalités suivantes :

$$\begin{aligned} L(\hat{f}) &\leq \hat{L}(\hat{f}) + R(n, \hat{f} - g) && (3.3)\\ \hat{L}(\hat{f}) &\leq \hat{L}(f^*) - R(n, \hat{f}) + R(n, f^*) + \varepsilon && (3.4)\\ \hat{L}(f^*) &\leq L(f^*) + R(n, f^* - g) && (3.5) \end{aligned}$$

Par simple sommation de 3.3, 3.4 et 3.5, on obtient le résultat voulu.

■

Théorème 3.4 (Borne sur l'erreur en généralisation) *Supposons que $\hat{f}$ converge simplement vers g quand $n \to \infty$, où $\hat{f}$ est la fonction fournie par un algorithme après apprentissage sur un échantillon de taille n de star-discrépance D_n. Alors :*

$$E|\hat{f} - g| \leq \hat{E}|\hat{f} - g| + \Big(V_{HK}(\hat{f}) + V_{HK}(g)\Big) D_n$$

Et à la condition que tous les $\{\hat{f} - g\}$ soient dans un espace de fonctions borné pour la norme de Hölder de degré supérieur à d (non nécessairement un entier), alors :

$$E|\hat{f} - g| \leq \underbrace{\hat{E}|\hat{f} - g|}_{\textit{erreur empirique}} + \Big(2V_{HK}(\hat{f}) + o(1))\Big) D_n$$

Ce théorème borne l'erreur en généralisation en se fondant uniquement :

1. Sur la variation de la fonction cible,
2. La discrépance de la suite utilisée pour l'appentissage.

Le terme d'erreur empirique quant à lui dépend de l'algorithme et ne peut donc pas être optimisé ici.

Démonstration :
La première inégalité est une conséquence directe de la borne de Koksma-Hlawka.
La seconde s'obtient comme suit :

$$V(g) \leq V_{HK}(\hat{f}) + V_{HK}(\hat{f} - g)$$

Le résultat est établit en bornant $V_{HK}(\hat{f} - g)$ par un $o(1)$, ce qui est l'objet de l'étape 1.

Étape 1 : réduction du problème aux dérivées de $\hat{f} - g$

$V(\hat{f} - g)$ a une variation totale linéaire en la somme des normes $||.||_\infty$ des d premières dérivées de $\hat{f} - g$. On peut donc ne s'intéresser qu'aux bornes de ces dérivées. Montrons à l'étape 2 que toutes ces dérivées tendent vers 0.

Étape 2 : utilisation des propriétés de la norme de Hölder pour borner les dérivées de $\hat{f}-g$ Soit d' tel que $\{\hat{f}-g\}$ soit dans un espace de fonctions borné pour la norme de Hölder de degré d' ; notons M la borne.

Alors pour tout $0 \leq k \leq d'$, avec $||f||_{a,b}$ la norme de Hölder de degré $a+b$ ($b \in [0,1]$), il est connu dans la théorie des espaces de Hölder que pour tout ε [Zuily and Queffélec; 1995, p277] :

$$||h||_{k,0} \leq \varepsilon ||h||_{\lfloor d' \rfloor, d'-\lfloor d' \rfloor} + K(\varepsilon)||h||_\infty$$

pour une fonction K et avec $h = \hat{f} - g$

$$||\hat{f} - g||_{k,0} \leq 2\varepsilon M + K(\varepsilon)||\hat{f} - g||_\infty \tag{3.6}$$

En choisissant ε suffisamment petit. La convergence de $||\hat{f} - g||_\infty$ est suffisante pour assurer la convergence des dérivées. Nous allons donc montrer cette convergence à l'étape 3.

Étape 3 : borne sur $||\hat{f} - g||_\infty$
$||\hat{f} - g||_1$ est minoré par $\alpha||\hat{f} - g||_\infty$ où α ne dépend que du coefficient de Lipschitz de $\hat{f} - g$. Ce coefficient de Lipschitz est borné grâce à l'inégalité 3.6, qui nous montre que la dérivée première est finie car majorée par $2\varepsilon M + K(\varepsilon)||\hat{f} - g||_\infty$ ($\hat{f}$ et g étant bornées).

■

Remarquons que le théorème 3.3 fournit une vitesse de convergence tandis que le second 3.4 donne une majoration de l'erreur en généralisation. En termes statistiques, l'erreur en généralisation est dite asymptotiquement consistante si $F \cup \{g\}$ a une norme de Hölder bornée pour un degré $> d$

3.2.2 Comparaison des résultats

Les résultats ci-dessus vont être discutés par rapport aux résultats non-asymptotiques de la théorie de VC et aux résultats empiriques venant de la théorie des processus. Les résultats sont regroupés dans le tableau 3.1 :

Comparaison à VC

En VC dimension finie si $\hat{L}(f^*) = 0$, alors $\hat{L}(\hat{f})$, avec f correctement estimée, décroît en $1/n$ avec un niveau de confiance $1-\delta$ (cf [Vapnik; 1995]). Les différences avec nos résultats sont les suivantes :

– Nous n'avons pas besoin d'être en VC-dim finie. Cette hypothèse est remplacée par une variation totale bornée. Notre hypothèse relativement simple à satisfaire mais pas toujours évidente : des discontinuités non parallèles aux axes peuvent engendrer

	Suites à faible discrépance	Monte-Carlo avec VC-dimension finie	Monte-Carlo avec classes de Donsker
Norme de Hölder bornée de degré α	$\alpha > d$, $O(\log(n)^d/n)$	-	$\alpha > d/2$,$O(1/\sqrt{n})$
VC-dim finie et $L^* = 0$	- -	$O(\log(n)/\sqrt{n})$ $O(\log(n)/n)$	$O(1/\sqrt{n})$ $O(1/\sqrt{n})$
Variation totale bornée	$O(\log(n)^d/n)$	-	-
Minimisation du risque structurel	pour une variation totale finie, $O(\log(n)^d/n)$	Consistance universelle et $O(\log(n)/\sqrt{n})$ dans une famille de VC-dimension ∞ d'approximation de fonctions ([Devroye et al.; 1997] chap 18)	- -

TAB. 3.1 : *Résumé des vitesses de convergences sous différentes hypothèses.*

une variation totale infinie. Il existe de nombreuses classes de fonction de VC-dim finie qui ont une variation au sens de Hardy et Krause infinie... D'un autre coté, les espaces de Hölder[2] bornés ont des nombres de couverture exponentiels et donc une VC-dim infinie alors que leur variation au sens de Hardy et Krause est facile à traiter (et qu'elle est bornée).

- Notre borne d'erreur est seulement asymptotique et dépend de l'hypothèse ; cependant, comme nous le verrons l'hypothèse ne possède qu'un impact assez faible et nous avons une **borne déterministe** (pas de niveau de confiance).
- Nous avons un terme supplémentaire en $\log(n)^d$;
- **Nous n'avons pas besoin de supposer $\hat{L}(f^*) = 0$ comme dans la théorie de VC pour obtenir une vitesse de convergence en $1/n$ au lieu de $1/\sqrt{n}$.**

Comparaison à la minimisation du risque empirique

Dans la minimisation du risque structurel ([Vapnik; 1995, p90],[Devroye et al.; 1997, chap18]), considérant une suite emboîtée de familles de VC-dim finie, alors $L(f)$ décroît en $\sqrt{\log(n)/n}$ avec un niveau de confiance $1 - \delta$. A notre connaissance, il n'existe pas de résultat montrant la disparition de la racine sous l'hypothèse du déterminisme des fonctions. Les différences avec nos résultats sont :

- Nous avons une convergence en $\log(n)^d/n$ au lieu de $\sqrt{\log(n)/n}$;
- Notre borne est déterministe ;
- Nous n'avons pas de niveau de confiance (borne déterministe) ;
- En minimisant le risque structurel, on peut avoir la consistance sur une famille de fonctions qui sont des approximateurs universels sur des domaines continus. Ce n'est pas le cas avec une famille à variation totale bornée.

[2]En pratique, les espaces de Hölder sont intéressants pour des applications concernant l'imagerie ou la vidéo, permettant ainsi d'évaluer les erreurs de première et de seconde espèce.

Comparaison aux résultats sur les classes de Donsker

En utilisant les résultats de [Van Der Vaart and Wellner; 1996], pour des exemples aléatoires et sous l'hypothèse que la famille de fonction est bornée pour la norme de Hölder, on obtient une convergence asymptotique si le logarithme du nombre de couverture est plus petit que quadratique (hypothèse beaucoup moins forte qu'une VC-dim finie). Le logarithme du nombre de couverture a un exposant d/α où α est le degré de la norme de Hölder utilisée. La contrainte est donc satisfaite si la norme de Hölder est bornée pour un degré $> d/2$ (ce qui est une contrainte moins forte que la notre puisque nous voulons $> d$). Cependant la convergence n'est pas très rapide : $O(1/\sqrt{n})$.

Discussion

Le facteur logarithmique disparait dans des inégalités de VC pour de très grandes valeurs de n (bornes d'Alexander). Il reste cependant présent dans tous les cas réels. A notre connaissance, il n'existe pas de résultats issus des classes de Donsker pour des familles à variation totale bornée.

L'inégalité de Koksma-Hlawka a déjà été utilisée en apprentisage actif par [Cervellera and Muselli; 2004]. Cet article prouve la convergence de $\hat{L}$ vers L^* si pour tout $f \in F$, $f - g$ a une variation bornée uniformément au sens de Hardy et Krause par une quantité M comme dans le théorème 3.2 présenté ici.

Une des contributions de notre travail porte sur le fait qu'il suffit d'avoir g et tout $f \in F$ de variation bornée au sens de Hardy et Krause. En particulier, la borne n'a pas besoin d'être commune à toutes les fonctions. C'est un point utile car s'il est vraisemblable de supposer que la variation est bornée, nous n'avons souvent aucune information *a priori* sur cette variation. De plus, notre borne sur l'erreur en généralisation peut aussi être utilisée pour des séquences non déterministes.

En résumé, le principal mérite de cette approche concerne l'amélioration de la borne, en $1/n$ alors que toutes les bornes utilisables en pratique (cf tableau 3.1) sont en $1/\sqrt{n}$ si l'on ne peut supposer que l'on est dans le cas de l'erreur nulle.

3.3 Validation expérimentale

On pourrait penser qu'il est possible d'obtenir des résultats similaires à ceux des faibles discrépances en se contentant d'utiliser des grilles (ou d'autres procédés d'échantillonnage simples). Il est cependant possible de montrer que la discrépance d'une grille est beaucoup plus grande que pour une suite de Monte-Carlo. De même, des suites de Monte-Carlo ont une discrépance très supérieure à des suites à faible discrépance. Ainsi, pour des suites bien choisies, la loi du logarithme itérée passe d'une convergence en $1/\sqrt{n}$ à $O(\log(n)^d/n)$.

Graphiquement les suites à faible discrépance semblent un peu bizarres (voir la figure 3.1). Peut-être à cause de leur spécialisation sur les rectangles parallèles aux axes. Elles sont cependant efficaces et leur génération peut-être faite assez aisément ([Burkardt; 2003]).

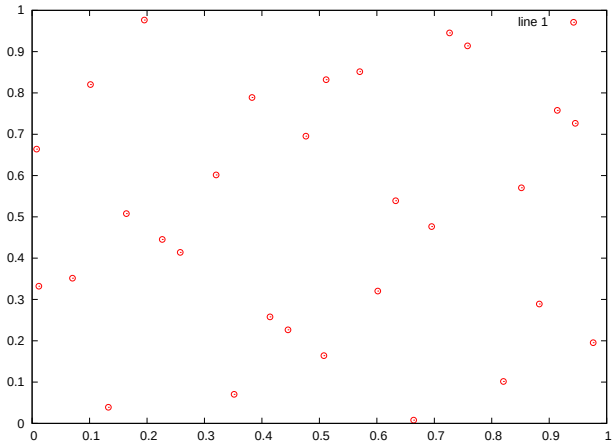


FIG. 3.1 : *Cette figure représente* 30 *points extraits d'une suite à faible discrépance dans* $\mathbb{R}^2$. *Seuls* 30 *points sont affichés mais la star-discrepance est calculée en utilisant* $100\ 000$ *points et est dans l'intervalle* $[0.0007; 0.001]$.*Avec la séquence « aléatoire » de Gnu Octave, dans les mêmes conditions, la discrépance est dans* $[0.0120; 0.0130]$.

Nous allons comparer expérimentalement sur un problème jouet, un algorithme utilisant comme base d'apprentissage un échantillonnage à la Monte-Carlo avec le même algorithme mais utilisant cette fois un échantillonnage à faible discrépance.

On se place en dimension $d = 7$(puis $d = 14$). Les points de l'apprentissage sont dans $[0, 1]^d$. La fonction cible d'un point $(x_1, \ldots, x_d)$ est la somme des d dimensions : $\sum_{i=1} dx_i$.

L'apprentissage est effectué par un RBF avec $\sigma = 0.1$ (SVM avec les options par défaut). Sur la figure 3.2, on repère en abscisse le nombre de points utilisés pour l'apprentissage et en ordonnée l'erreur en généralisation. La moyenne et l'écart type ont été évalués sur 30 exécutions indépendantes.

3.4 Quasi-Monte-Carlo biaisé pour l'apprentissage L_∞

Cette section s'intéresse à l'approximation au sens de la norme L_∞ d'une fonction g dans un espace de fonctions F tel que :

1. g n'appartienne pas nécessairement à F,
2. F ne contient pas nécessairement de « bonne » approximation de g.

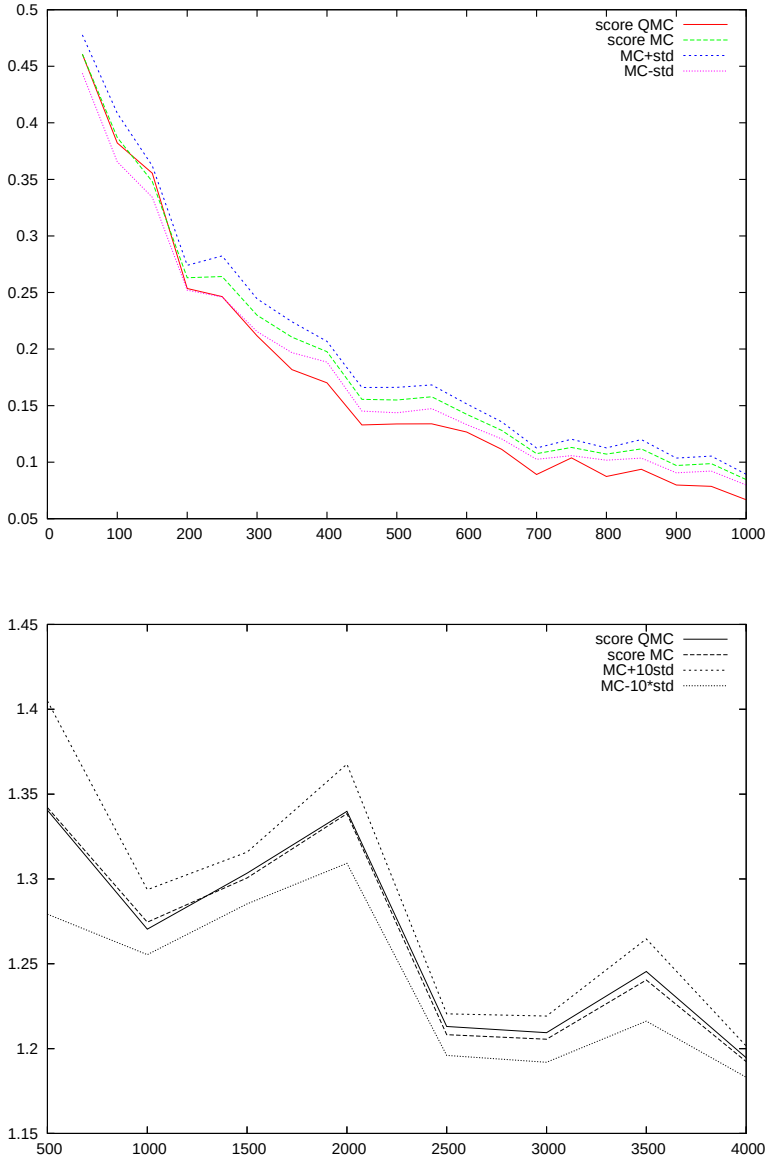


FIG. 3.2 : *Résultats expérimentaux. Fonction déterministe. La suite de Niederreiter à faible discrépance à été fournie par le soft [Burkardt; 2003]. L'ordonnée représente le taux d'erreur. QMC représente l'erreur en généralisation pour la suite à faible discrépance et MC l'erreur pour un échantillonnage uniforme (autour de MC on évalue des intervalles de confiances de* 10 *fois l'écart-type). La dimension est de* 7 *sur le premier fichier et de* 14 *sur le second. La suite est très efficace en dimension* 7 *mais présente des résultats similaires à un échantillonnage aléatoire en en dimension* 14 *(Ceci est dû en partie à la dépendance en la dimension de la vitesse de convergence qui est en* $\log(n)^d/n$*) ; de plus, les suites à faible discrépance sont réputées difficiles à trouver si* d *est élevé).*

3.4.1 Le cas linéaire

Soit F l'ensemble des fonctions linéaires envoyant la boule unité B (de la norme $||.||$) de $\mathbb{R}^{dim}$ dans $\mathbb{R}$.
Soit $x_1, \ldots, x_n$ un échantillon de points de B et $y_i = f(x_i)$, où f est une fonction déterministe.

Notons M la matrice définie par les x_i (M_{ij} correspond à la $j^{\text{ème}}$ coordonnée de x_i)
Notons $f^* \in \underset{f \in F}{\text{argmin}} \parallel f - g \parallel_\infty$.
Posons :
- $\hat{L}(f) = \sup_i |f(x_i) - y_i|$;
- $L(f) = ||f - g||_\infty - ||f^* - g||_\infty$;
- $\hat{L}^* = \sup_i |f^*(x_i) - g(x_i)| = \hat{L}(f^*)$.

Supposons l'existence d'un algorithme d'apprentissage déterminant le choix de $\hat{f}_n \in F$ comme approximation de g grâce à n exemples. On s'intéresse à la convergence de $L(\hat{f}_n)$.

Considérons $h_n = \hat{f}_n - f^*$ et remarquons que h_n est aussi dans F (ensemble des fonctions linéaires).

Pour tout x de B, notons x' le point de l'ensemble d'apprentissage le plus proche de x. ($x' \in \text{argmin}_{x' \in \{x_1, \ldots, x_n\}} ||x - x'||$). En utilisant la linéarité de h_n :

$$h_n(x) = h_n(x') + h_n(x - x')$$

$$||h_n(x) - h_n(x')||_2 \leq \xi \sup_{u \in B} |h_n(u)| \tag{3.7}$$

où $\xi = \sup ||x' - x||$; ξ dépend de l'exemple (voir la notion de dispersion dans [Niederreiter; 1992]). De plus par Cauchy Schwartz,

$$\sup_{u \in B} |h_n(u)| \leq \sqrt{n}(\sup_{x \in B} \inf_{\lambda | x = \sum \lambda_i x_i} ||\lambda||_2) \times \sup_i |h_n(x_i)| \tag{3.8}$$

En combinant 3.7 et 3.8, on a :

$$\sup_{u \in B} |h_n(x) - h_n(x')| \leq \xi\sqrt{n}(\sup_{x \in B} \inf_{\lambda | x = \sum \lambda_i x_i} ||\lambda||_2) \times \sup_i |h_n(x_i)|$$

et

$$\sup_{u \in B} |h_n(x)| \leq \sup_i |h_n(x_i)| + \xi\sqrt{n}(\sup_{x \in B} \inf_{\lambda | x = \sum \lambda_i x_i} ||\lambda||_2) \times \sup_i |h_n(x_i)|$$

Nous pouvons également borner $\sup_i |h_n(x_i)|$ par $2||f^* - g||_\infty$.

D'où :

Proposition 3.1

$$L(\hat{f}) \leq (1+K)\sup_i |h_n(x_i)| \leq (1+K)(\hat{L}(\hat{f}_n) + ||f^* - g||_\infty) \leq 2(1+K)||f^* - g||_\infty$$

avec $K = \xi\sqrt{n}\sup_{x\in B}\inf_{\lambda|x=\sum_i \lambda_i x_i} ||\lambda||_2$ *et* $\xi = \sup_x \inf_{x'\in\{x_1,\ldots,x_n\}} ||x - x'||$.

Remarque :
On s'attend à ce que $L(\hat{f}_n)$ étant la différence $||\hat{f}_n - g|| - ||f^* - g||$, converge vers 0. Sinon cette borne n'a pas d'intérêt. En fait, la borne n'est intéressante que si :

- L'algorithme d'apprentissage utilisé conduit à une borne d'erreur empirique (*i.e* $|h_n|$ est petit sur les x_i). En effet, notre borne contient un terme $L(\hat{f}) \leq (1+K)\sup_i |h(x_i)|$. C'est une hypothèse raisonnable puisque $\hat{f}$ est choisi en regardant les x_i.
- Il existe une bonne approximation f^* du concept cible g. C'est-à-dire que L'on suppose que $||f^* - g||_\infty$ est petit (hypothèse que nous ferons dans les expériences) à cause du terme $L(\hat{f}) \leq 2(1+K)||f^* - g||_\infty$.

♦

3.4.2 Apprendre dans l'espace des attributs

Soit F l'espace des fonctions définies sur un noyau K symétrique et les points d'apprentissage $x_1, \ldots, x_n$:

$$F = \{f : x \mapsto f(x) = \sum_i \lambda_i K(x, x_i)\}$$

Notons $h(x) = \hat{f}(x) - f^*(x)$. Le but est de majorer $|h|$ en fonction du maximum empirique de h ($\max_i |h(x_i)|$).

Posons : $h(x) = \sum_i \lambda_i K(x, x_i)$.

Pour tout x, on note x' le plus proche voisin de x dans l'échantillon d'apprentissage ($x' \in \text{argmin}_{x_1,\ldots,x_n} ||x, x'||_2$). Soit $\mu(x)$ le vecteur de $\mathbb{R}^n$ tel que pour tout x, $K(x, x_i) - K(x', x_i) = \sum_j \mu_j(x)K(x_j, x_i)$ (on prend x' tel que $\mu(x)$ ait la petite norme possible). $\mu(x)$ existe bien car $K(x, xi) - K(x', xi)$, indexé par i, est un vecteur à n composantes ; et que la matrice $K(xj, xi)$ est de rang plein de même taille. Intuitivement, $\mu(x)$ correspond à une mesure de la variation de $(K(., x_i))_{i=1\ldots n}$ entre x et x' exprimée dans la base canonique de la matrice du noyau ; il s'agit d'une estimation de l'erreur faite en

prenant x' au lieu de x. On a :

$$\begin{aligned} h(x) - h(x') &= \sum_i \lambda_i (\sum_j \mu_j(x) K(x_j, x_i)) \\ &= \sum_{i,j} K_{j,i} \lambda_i \mu_j(x) \\ &= \lambda^t K \mu(x) \end{aligned}$$

et donc avec Cauchy-Schwartz,

$$|h(x) - h(x')| \leq \sup_x ||\mu(x)||_2 ||(h(x_1), h(x_2), \ldots, h(x_n))||_2$$

d'où :

$$\sup |h| \leq (1 + \underbrace{\sup_x ||\mu(x)||_2}_{L(x_1,\ldots,x_n)} \sqrt{n}) \sup_i |h(x_i)| \tag{3.9}$$

$$\sup |h| \leq (1 + L(x_1, \ldots, x_n)\sqrt{n})(2||f^* - g||_\infty)$$

Proposition 3.2 *Avec les notations ci-dessus, l'erreur en généralisation de $\hat{f}$ est bornée par :*

$$L(\hat{f}) \leq 2(1 + L(x_1, \ldots, x_n)\sqrt{n})||f^* - g||_\infty$$

L'intérêt essentiel de ce résultat est de conduire à **optimiser la répartition des** x_i en utilisant une borne.

3.5 Validation expérimentale

L'intérêt du critère $L(x_1, \ldots, x_n)$ qualifiant le qualité d'une base d'apprentissage et défini comme dans l'équation 3.9, est étudié par comparaison à la mesure de qualité uniforme de la base d'apprentissage, donnée par :

$$M(x_1, \ldots, x_n) = \sup_{x \in B} \left[\inf_{i=1\ldots n} ||x - x_i||_2 \right]$$

Formellement, un problème est représenté par deux points :
- $L(\hat{f}), L(x_1, \ldots, x_n)$,
- $L(\hat{f}), M(x_1, \ldots, x_n)$,

Les problèmes considérés sont définis en dimension 1 en :
- Choisissant aléatoirement une fonction linéaire f,
- tirant $x_1, \ldots, x_n$ uniformément dans $[0, 1]$,
- On calcule les étiquettes par $y_i = f(x_i)$, en ajoutant un bruit gaussien centré d'écart type σ,

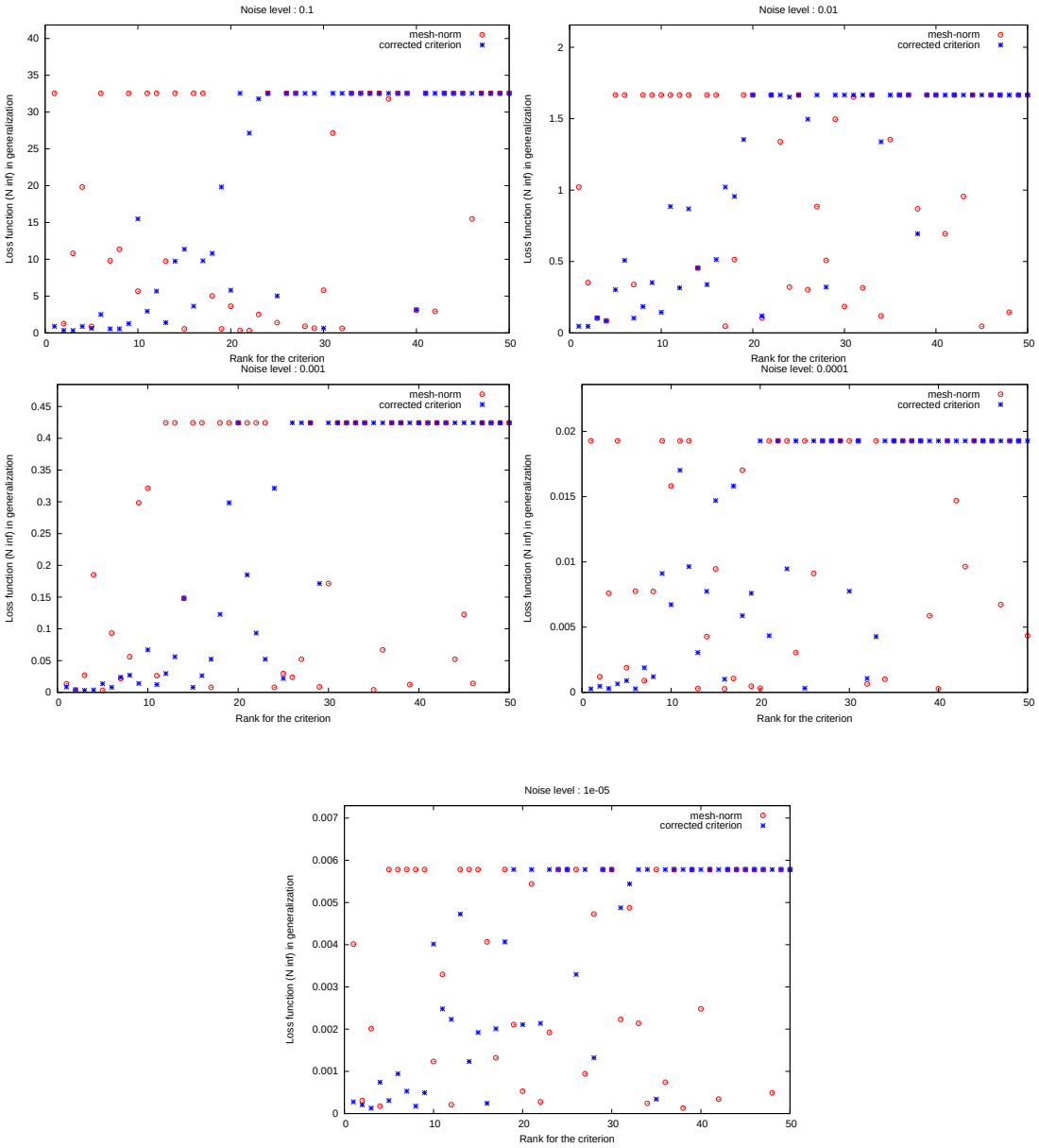


FIG. 3.3 : *Utilisation d'une approche quasi-aléatoire dans l'espace des attributs pour l'apprentissage ; le premier graphe représente les performances en généralisation obtenues par un apprentissage à partir de* 10 *exemples avec un bruit de* 10^{-1}*, en fonction de leur ordre pour un critère de type grille (*$\sup_x \inf_i ||(x, x_i)||_2$*). La même chose est ensuite réalisée en fonction de notre critère. On répète ensuite les mêmes graphes avec des taux de bruit différents (de* 10^{-2} *à* 10^{-5}*). Notre critère est toujours le meilleur puisque l'ordre qu'il nous donne est plus proche des erreurs réelles en généralisation. remarque : un cutoff a été placé pour les valeurs supérieures à la médiane pour l'erreur en généralisation afin de ne pas écraser les graphes (ce qui explique le fait que plusieurs erreurs en généralisation élevées apparaissent comme identiques sur le graphe).*

Pour chaque problème :

1. On effectue l'apprentissage en utilisant la norme L_∞ et un RBF.
2. On évalue les capacités en généralisation de l'hypothèse apprise.

On répète ceci pour 50 problèmes, puis on fait un graphe en portant en abscisse le rang selon le critère considéré (L ou M) et en ordonnée la capacité en généralisation.

Les graphiques ainsi obtenus pour cinq niveaux de bruits sont donnés dans la figure 3.3 ; des graphiques supplémentaires peuvent être otenus et montrent les mêmes résultats si l'on dispose seulement de 5 exemples pour apprendre.

Au niveau de bruit 0.1, pour le critère corrigé, les 9 meilleurs ensembles de points ont tous une erreur en généralisation inférieure à 3 alors que pour le critère naïf, le taux d'erreur pour les 9 meilleurs points peut-être supérieur à 10. Notre critère est le meilleur dans tous les cas et sa supériorité s'affirme de plus en plus quand le bruit est petit. Des résultats similaires sont obtenus avec seulement 5 points pour l'apprentissage (figure non donnée car trop semblable à la précédente).

3.6 Conclusion

Ce chapitre aborde un point complémentaire à l'analyse classique faite en apprentissage statistique, et s'intéresse à la distribution des points de la base d'apprentissage. Formellement, la distribution classique, *i.e* uniforme, est comparée à l'usage de suites de points déterministes optimisant la dispersion collective de la suite dans l'espace des instances.

Cette approche a pour avantage principal une vitesse de convergence en $1/n$ (à comparer au $1/\sqrt{n}$ standard en apprentissage), sous des hypothèses modérées, portant *in fine* sur la variation bornée des fonctions intéressantes.

L'intérêt opérationnel de l'approche est étudié sur des problèmes jouets ; il est démontré de façon convaincante en faible dimension, mais se heurte à la malédiction de la dimension.

Dans le cas élémentaire des fonctions linéaires, un critère de discrépance simplifié est défini, dont l'intérêt est démontré aussi bien théoriquement (pour borner l'erreur en généralisation) qu'opérationnellement (pour prédire l'erreur en généralisation).

Notons que le fait de pouvoir prédire *a priori* la qualité de l'apprentissage effectué constitue un des grands défis de l'apprentissage, qui a été au cœur du projet européen METAL et qui est repris par le défi d'Isabelle Guyon.

Il est également possible d'étendre ces résultats au cas non déterministe, c'est-à-dire au cas ou f est de la forme $f(x, w)$ avec w une variable aléatoire. La norme $L\infty$ n'ayant

plus de sens ici, on la remplace par le supremum presque sûr $||q||_{\infty,ps} = \sup\{l; P(|q| > l) > 0\}$.

Proposition 3.3 (extension non-déterministe) *Sous ces conditions, approximer $f(x,w)$ par $\{(x,w) \mapsto g(x); g \in F\}$ conduit aux bornes adaptées suivantes :*

$$L(\hat{f}) \leq (1+K) \sup_i |\hat{f}(x_i) - f^*(x_i)|_{ps} \textit{ (dans le cas linéaire)}$$

$$L(\hat{f}) \leq (1+L\sqrt{n}) \sup_i |\hat{f}(x_i) - f^*(x_i)|_{ps} \textit{ (dans le cas de l'espace d'attributs)}$$

L'intérêt principal se trouve alors dans une forme particulière d'apprentissage actif :

- L'algorithme choisit un ensemble fini de x_i ;
- On récupère plusieurs $f(x_i, w)$ pour chaque i ;
- Alors, si f^* (optimal dans F) est optimal dans l'espace de toutes les fonctions réelles sur le domaine, alors pour tout i, $\hat{f}(x_i)$ est très proche de $f^*(x_i)$;
- La borne précédente est alors petite et assure une bonne généralisation si K et L sont petits.

Troisième partie

SÉLECTION D’ATTRIBUTS

Chapitre 4

Bioinformatique

La troisième partie de cette thèse est consacrée à la sélection d'attibuts. La motivation applicative de ces travaux est la Bio-Informatique. Le chapitre 4 introduit et discute ainsi les enjeux de la sélection d'attributs en Bio-Informatique.

Ce chapitre présente les applications grandeur réelle situées dans le domaine de la Bio-Informatique qui ont motivé cette thèse. Les spécialistes de l'apprentissage y verront quelles sont les difficultés et les exigences des biologistes, tandis que les autres pourront le considérer comme un moyen de se fixer les idées sur le type de problèmes adressés par l'apprentissage.

4.1 Présentation du problème

Avec les nouvelles possibilités de séquençage rapide de segments de génomes ou de mesures d'expression de gènes dans les puces ADN, la biologie s'est trouvée confrontée à deux nouveaux problèmes :

- La **masse de données** collectée, interdit de traiter « à la main » les résultats des expériences. Ainsi, dans l'impossibilité d'utiliser leur expertise sur les données, les biologistes ont commencé à développer leurs propres méthodes d'apprentissage puis se sont tournés vers la littérature de l'apprentissage et des collaborations fructueuses biologistes/apprentistes ont pu être initiées.
- La **malédiction de la dimension** (*Curse of dimensionality*), problème bien connu en apprentissage et qui constitue un de ses principaux défis. Celle-ci ne se pose que lorsque l'on dispose d'un nombre élevé de descripteurs. Dans ce cadre en effet, surgissent de grandes difficultées liées au mauvais comportement des distances L_2 en grande dimension, à la difficulté de prendre en compte les effets croisés, croissance de la dimension des familles d'hypothèses (VC dimension, nombres de couvertures ...). Ce fléau de la dimensionalité est au cœur de la Bio-Informatique ; les descripteurs correspondent souvent à des gènes, au nombre de quelques milliers, parmi lesquels le but est d'itentifier les responsables de tels ou tels effets.

Clairement, les enjeux de la Bio-Informatique pour la santé et pour les sciences du vivant sont considérables : la question devient ainsi de faire parler ces données génomiques. Ces bases de données sont souvent constituées (1) de descriptions d'objets : par exemple des séquences de nucléotides, décrits par de nombreux attributs (des milliers dans le cas des puces ADN), et (2) d'une étiquette associée, par exemple un degré de décalage dans des phénomènes de traduction non conventionnels (observés chez les virus). L'hypothèse de travail classique en apprentissage est qu'un petit nombre de descripteurs permettent de prédire l'étiquette, et on tente de découvrir les régularités pertinentes en mettant en oeuvre des techniques d'apprentissage automatique. Un problème essentiel est que le nombre d'exemples (description, étiquette) est le plus souvent très restreint eu égard au nombre important de descripteurs, dont la plupart sont supposés non pertinents (la couleur des yeux n'est vraisemblablement pas pertinente pour l'évolution d'un cancer de la prostate). De ce fait, il existe de très nombreuses régularités vérifiées sur la base d'apprentissage et n'ayant aucune portée générale. Une démarche classique d'apprentissage requerrait soit de disposer de connaissances *a priori*, soit des bases d'exemples beaucoup plus importantes.

Le travail présenté dans ce chapitre s'appuie sur la combinaison des techniques informatiques et des intuitions biologiques pour analyser les données issues des puces à ADN. Il s'inscrit dans le cadre d'un projet pluri-disciplinaire de collaboration entre le LRI et l'institut Curie.

Ce chapitre introduit tout d'abord le problème posé par les biopuces, avant de discuter l'apport de l'apprentissage, plus spécifiquement des techniques de sélection d'attributs pour la compréhension de phénomènes biologiques.

Le cœur du travail présenté consiste à tirer de l'information inclue dans un grand nombre d'attributs et un petit nombre d'exemples. Les hypothèses de travail faites seront discutées et les algorithmes étudiés expérimentalement sur des données réelles (des résultats sur des données artificielles existent également mais ne seront pas présentés ici).

4.1.1 Les puces à ADN

Les puces à ADN (microarrays) sont une technique maintenant courante utilisée pour l'analyse de l'expression de gènes. Elles correspondent à une sorte de photographie de l'activité des gènes dans une cellule et permettent donc de découvrir comment s'expriment les gènes dans telle ou telle situation. Plusieurs techniques existent : nylon, lames de verre, et Affymetrix sont les principales. Les deux premières diffèrent essentiellement par le nombre de gènes que l'on peut mesurer simultanément ainsi que par la précision des mesures (l'avantage va aux lames de verre qui sont plus récentes). Par contre la technique Affymetrix est assez différente. Les données que nous avons essayé d'analyser sont issues de lames de verre. Voici donc un aperçu de la technique (fig.4.1). Le but n'est pas de mesurer une valeur absolue de l'expression d'un gène mais de comparer son niveau d'expression dans deux cellules ; une des cellules est celle que l'on souhaite observer et la seconde sert de référence. On commence par déposer sur une lame de verre des sé-

quences d'ADN qui potentiellement, codent pour une protéine (Open Reading Frame) ; ces séquences proviennent de banques de gènes. Ensuite, on extrait l'ARN d'une cellule dont on souhaite mesurer l'expression ainsi celui d'une cellule de référence[1]. Cet ARN est transformé en ADN par transcription inverse en lui adjoignant un marqueur fluorescent (la fluorescence n'est pas la même pour les cellules témoins et les cellules mesurées). On place ensuite l'ADN obtenu au contact de la lame de verre. Les ADN vont s'hybrider avec les ORFs leur correspondant ; il ne reste plus qu'à laver les lames et à détecter avec un laser les niveaux de fluorescence observés. Pour chaque ORF déposée sur la lame initiale on dispose donc de deux valeurs qui correspondent théoriquement à l'intensité de l'expression d'un gène dans la cellule témoin et dans la cellule à mesurer. Il pourrait sembler superflu de refaire la mesure pour les cellules témoins à chaque lame de verre, cependant la pratique révèle que c'est indispensable car l'hybridation en particulier ne se passe pas toujours de la même façon, sans que l'on ne comprenne pourquoi. Aussi au lieu d'utiliser les intensités de l'expression des cellules, on préfère utiliser les ratios des intensités (on divise par l'intensité du témoin). Ceci permet en particulier de comparer des lames entre elles.

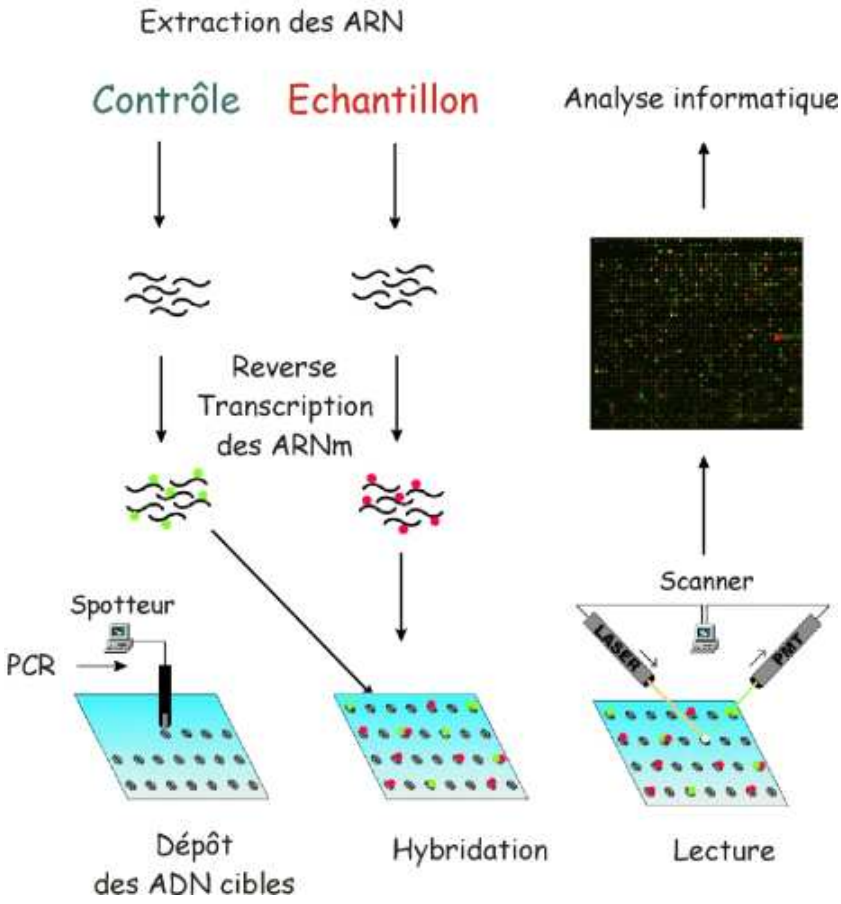


FIG. 4.1 : *Schéma récapitulatif du principe des puces à ADN. Source : www.bio.espci.fr/ puces/principe.html*

[1] L'ARN car reflète quels sont les gènes en cours de transcription, l'idéal serait de savoir quelles sont les protéines présentes dans la cellule mais la technique n'est pas encore au point.

Cette technique présente toutefois deux problèmes majeurs. Le premier est l'analyse des résultats. En effet, les lames présentent du bruit provenant à la fois de la technique de mesure et de la variabilité biologique des cellules. En effet, les cellules témoins n'ont pas nécessairement exactement la même expression. De plus, un génome simple comme celui d'une levure compte environ 6400 gènes (le nombre exact de gènes mesurés varie suivant les laboratoires). Idéalement, un processus d'apprentissage à partir de telles lames devrait donc compter quelques dizaines de milliers de lames pour permettre une bonne généralisation et résistance au bruit. Ce point nous amène naturellement à la seconde difficulté, qui est plus terre à terre : une seule lame coûte environ 1000 €, ce qui conduit à se contenter d'un nombre de lames extrêmement réduit (une vingtaine). L'apprentissage ne sera donc réalisable qu'en utilisant des informations supplémentaires issues de la biologie, ou connaissance du domaine.

4.1.2 Les souhaits des biologistes

Le problème étudié, au cœur des travaux de l'équipe de Marie Dutreix, concerne la détection des radiations. A terme il s'agira de construire des détecteurs à placer sur des sites potentiellement exposés aux radiations. La première de leurs hypothèses est que l'effet des radiations est visible dans l'expression du génome des levures. Ce point est essentiel car les niveaux d'irradiation des cultures sont tellement peu élevés, qu'il n'y a pas de conséquence visible évidente sur le métabolisme des levures. Toutefois la dose d'irradiation choisie est juste en dessous du seuil détectable et les biologistes ont de fortes présomptions sur la possibilité d'un effet à long terme. Ils estiment en revanche qu'à cause de cette faible dose d'irradiation, l'effet ne concernera qu'un petit nombre de gènes.

4.1.3 Description et remarques sur l'expérimentation

De manière répétée, des cultures de *S. Cerevisae* sont mises en croissance en étant soumises ou non à de faibles radiations . Elles sont prélevées au bout d'un nombre fixe de divisions et leurs ARN sont extraits et examinés sur des biopuces. A noter que deux souches différentes de levures ont été utilisées. Nous disposons de 12 cultures de levures qui n'ont été soumises à aucun stress particulier et de 6 cultures faiblement irradiées. Cette répatition s'explique par le fait qu'il est plus facile expérimentalement de cultiver des levures sans stress que des levures irradiées toutes de la même façon.

Par ailleurs, on dispose d'un jeu supplémentaire de 6 lames correspondant à un ensemble test.

Ce protocole appelle un certain nombre de remarques :

Signification des mesures : Les données ont déjà été renormalisées par des statisticiens afin de permettre la comparaison de deux lames entre elles. Nous ne nous sommes pas attardés sur cette technique mais elle constitue cependant un préliminaire indispensable à toute analyse, et certains points sont encore en discussion sur ce thème.

La valeur qui nous a été transmise est la suivante :

$$\frac{\textit{niveau d'expression dans la cellule}}{\textit{niveau d'expression pour le témoin}}$$

Afin toutefois de ne pas favoriser artificiellement les gènes sur-exprimés par rapport aux réprimés, nous allons travailler dans l'espace logarithmique. On travaillera avec le logarithme en base deux des données. Finalement, le codage adopté est :

- -1 : Le gène mesuré s'exprime deux fois moins que pour le témoin ;
- 0 : Le gène mesuré s'exprime de la même façon que pour le témoin ;
- 1 : Le gène mesuré s'exprime deux fois plus que pour le témoin.

La technique de mesure : On espère en fait que les 12 levures "saines" vont avoir des profils d'expression identiques. En fait, idéalement, on souhaiterait même que tous les gènes aient une expression de 0. En effet, cela reviendrait à dire que dans ces cultures le niveau d'expression de chaque cellule est le même que pour le témoin. Ce n'est bien sûr pas le cas ... La mesure est donc entachée d'erreurs. Leur répartition semble ne pas se faire uniformément sur tous les gènes (certains sont plus touchés que d'autres, ce qui pourrait provenir d'une certaine variabilité biologique).

Redondances : Pour des raisons technique, certains gènes ont été mesurés deux fois.

On peut regarder les profils d'expression de ces gènes pour évaluer l'erreur induite par la seule mesure (fig.4.2)

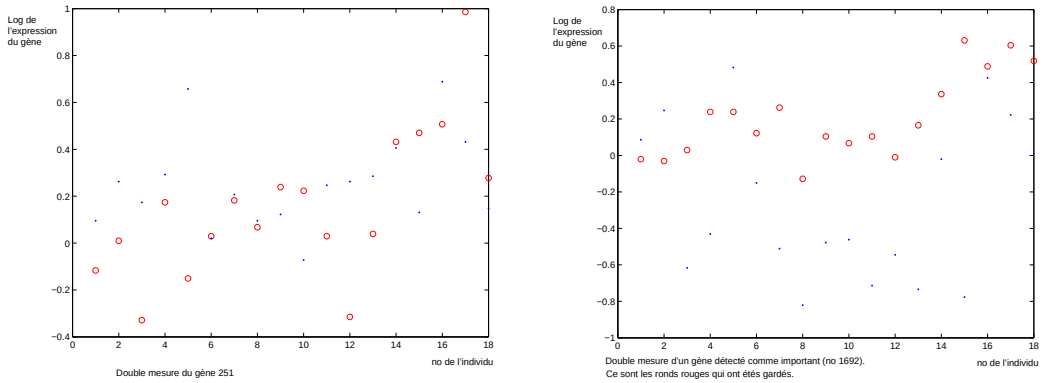


FIG. 4.2 : *Le premier graphique est l'expression du gène 251 et le second du gène 1692. La première mesure est représentée avec des points bleus et la seconde avec des cercles rouges.*

En observant les graphes obtenus pour la répétition de la mesure d'un gène (4.2), on peut commencer à s'inquiéter sur la reproductibilité des mesures. En effet, on peut comprendre que l'expression d'un gène chez des levures saines ne soit pas toujours la même (variabilité biologique) ; en revanche, il est plus ennuyeux que de telles différences existent entre la première et la seconde mesure (points rouges et points bleus). L'estimation de la variabilité de l'expression d'un gène en utilisant ces doubles mesures conduit à une valeur de 0.78, ce qui est énorme en regard des valeurs prises par les expressions. Il faut tout de même tempérer cette première impression : les gènes qui ont été mesurés deux fois correspondent à de petites séquences d'ARN, pour lesquelles on suppose que l'hybridation n'est pas très bonne.

Si ce point se trouvait vérifié, ceci signifierait que ces mesures sont beaucoup moins fiables que la moyenne.

4.1.4 Fiabilité des mesures

Pendant l'expérimentation, si l'un des marqueurs fluorescents (Cy3 ou Cy5) ne s'accroche pas bien ou que l'hybridation des gènes portant ce marqueur se fait mal, la mesure peut se retrouver complètement décalée dans le sens d'une sur-expression ou d'une sous-expression. Pendant l'expérimentation il est possible d'attribuer une certaine fiabilité aux mesures. C'est ce qui a été fait, malheureusement il n'y a que trois niveaux possibles :

- 0 : les deux marqueurs sont incorrects ;
- 1 : un des deux marqueurs est correct ;
- 2 : les deux marqueurs sont corrects.

A cause du probable fort décalage des valeurs dès qu'un marqueur est incorrect, plutôt que d'introduire une pondération liée à la fiabilité, nous avons décidé de supprimer les valeurs manquantes. Il faudra donc que les algorithmes utilisés soient capables de gérer ces valeurs manquantes. Pour se faire une idée de leur importance, dans le jeu de données initial, il y a $97,37\%$ des valeurs qui ont une fiabilité de 2. On peut se faire une idée de leur répartition sur la figure 4.3 ; en particulier, on constate que la lame $n^o 2$ porte beaucoups d'erreurs, ce qui peut jeter un soupçon sur la lame entière (la supprimer fait passer le pourcentage de mesures avec fiabilité 2 à $98,26\%$).

Ce travail de détection de fiabilité ainsi que celui de renormalisation des données (à cause du décalage dans les courbes de réponse de Cy3 et de Cy5) a été réalisé par J. Peyre and A. Antoniadis par la méthode qu'ils ont mise au point : LOWESS.

4.1.5 Indépendance ou non ?

Une question se pose : peut-on considérer les gènes comme indépendants ou non ? Une réponse est immédiate : non. En effet, il suffit de savoir qu'une cellule ne peut pas produire plus d'une certaine quantité d'ARN en un temps donné. Ceci introduit donc une contrainte sur la somme des expressions et donc interdit l'indépendance. Un second élément est l'existence de réseaux de régulation : certains gènes sont régulièrement co-activés ou en activent d'autres. Cependant, ces réseaux, bien qu'importants, ne sont pas encore bien connus (même pour les levures) et sont si complexes que le bruit présent dans nos mesures ne permet probablement pas de les prendre en compte. L'indépendance est donc fausse en pratique. Cependant les biologistes considèrent qu'il n'est pas absurde de la supposer puis, dans un second temps de se pencher sur les gènes considérés comme les meilleurs afin de déterminer quels sont les gènes dont ils influencent l'expression.

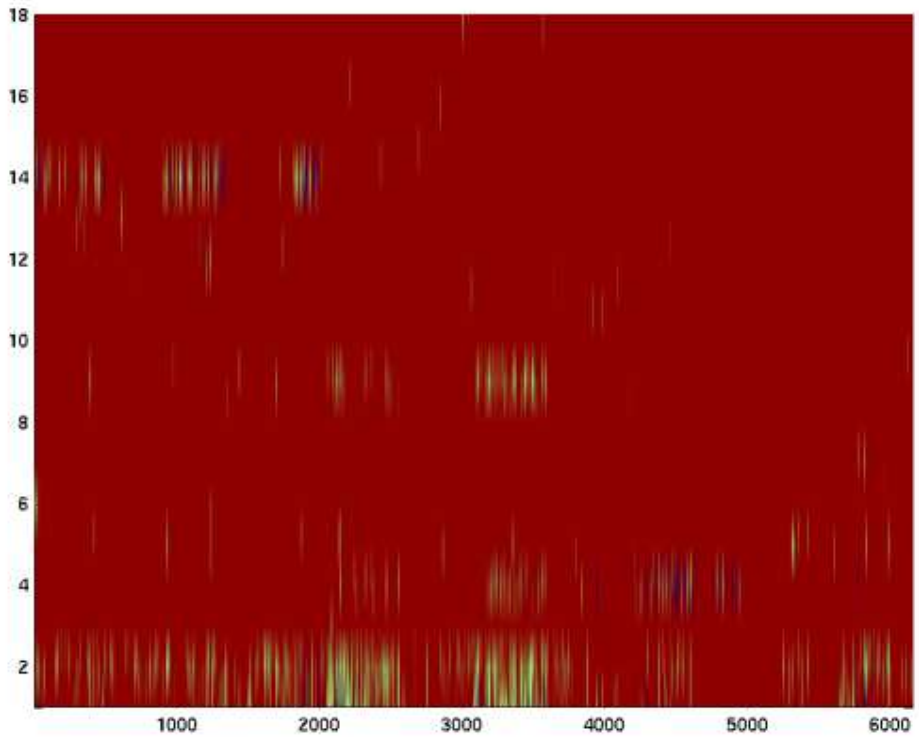


FIG. 4.3 : *Visualisation de la fiabilité des gènes. Gènes en abscisse, les lames en ordonnée. La couleur rouge correspond à une fiabilité de 2, le vert à 1 et le bleu à 0. Les erreurs sont nombreuses et ne sont certainement pas uniformément réparties.*

4.1.6 Récapitulons le problème

Nous sommes confrontés à un problème relativement inhabituel ; ses principales caractéristiques sont :

- les données sont bruitées. Il existe deux origines à ce bruit : le premier vient de la variabilité biologique ; il correspond à une certaine différence normale entre les levures. Le second est une erreur sur les mesures. En effet, la technique est imprécise en particulier à cause des différences pouvant survenir au cours de l'hybridation.
- La technique de mesure peut produire des valeurs aberrantes (lorsque l'hybridation ne s'est pas faite de la même manière pour chacun des marqueurs).
- Le nombre de lames (exemples) est très faible par rapport au nombre de gènes (attributs) : 18 mesures alors qu'il y a 6157 gènes.
- Le concept cible est supposé être particulièrement simple. Il s'agit, d'après les estimations des biologistes d'une conjonction d'une cinquantaine d'attributs. Son repérage devrait en outre être facilité par le fait que pour les individus sains, les gènes ne sont pas supposés être activés (valeur autour de 0).

4.2 Étude de l'existant

Nous allons regarder brièvement différentes techniques d'apprentissage, ou plus précisément de sélection d'attributs pertinents telles qu'elles ont été développées par des informaticiens dans des contextes complètement étrangers à la biologie.

4.2.1 « forward » ou « backward » ?

Deux grands types d'approches sont possibles pour la sélection d'attributs :

- Les techniques "forward" où l'on démarre sans attribut et on ajoute au fur et à mesure les attributs qui seront estimés comme étant les plus pertinents ;
- Les méthodes "backward" procèdent exactement à l'inverse : on démarre avec tous les attributs et on essaye d'éliminer ceux qui ne servent pas pour classer.

L'exemple classique d'algorithme "forward" est ID3 [R.; 1986] : en effet, on choisit d'ajouter dans les attributs utiles au classement celui qui maximise une fonction de Gain (calculée à partir d'une entropie). Une des grandes difficultés lors de la mise en œuvre est de savoir quand s'arrêter dans l'ajout d'attributs. Ce point devient particulièrement critique pour nos données car si l'on prend comme notation la fonction de gain de ID3, plusieurs attributs la maximisent et font chuter l'entropie du système à 0. Ceci nous empêche de poursuivre ID3 puisque toutes les données sont classées. De plus il n'est pas évident que la classification obtenue ait un sens car le nombre d'attributs étant très élevé en regard du nombre d'instances, on a peut-être eu un malheureux coup de chance.

La méthode “backward” semble plus adaptée à notre problème. Une illustration de cette technique est l’utilisation des “Markov Blankets” (Couvertures de Markov, voir section 4.2.4).

4.2.2 Le clustering

Le *clustering* est une technique d’apprentissage non-supervisé qui est employée massivement par les biologistes. Leur but est de modifier l’agencement des données afin de voir l’émergence de zones dans l’expression génétique. Les techniques de clustering auxquelles ils font appels tentent de regrouper les lames puis les gènes (indépendamment des lames) en les plaçant dans un arbre (ex : clustering hiérarchique). Les données sont ensuite affichées en respectant l’arbre obtenu (fig. 4.5). L’analyse est ensuite réalisée de manière visuelle. Nous allons voir que le principe de base est très simple (il existe cependant de nombreux raffinements). Une implémentation est disponible librement [Eisen; 98-99].

Pour effectuer leurs ré-arrangements, les biologistes effectuent à la fois un clustering sur les gènes et sur les lames. Les représentations des valeurs des attributs (les spots) sont ensuite réorganisés en respectant les arbres donnés par le clustering. La méthode de clustering la plus répandue est le clustering hiérarchique. Par exemple, pour les lames, on commence par regarder lesquelles sont les plus semblables (distance au choix ou mesure de similarité), ces deux lames sont associées dans le même cluster, et l’algorithme continue à essayer d’associer des lames aux clusters jusqu’à ce qu’il ne reste plus de lame seule. Notons qu’il s’agit d’une méthode d’apprentissage non-supervisée : on pourra donc contrôler dans une certaine mesure nos résultats en vérifiant que les clusters obtenus pour les lames correspondent bien à des regroupements liés aux classes I et S.

Les principales variations de cet algorithme portent sur la manière de mesurer la similarité ainsi que sur le type de regroupements qui sont réalisés. On trouvera les principales variations sur les figures 4.4,4.5. La mesure de similarité la plus courante est le coefficient de corrélation de Pearson qui suppose une corrélation linéaire entre les variables (N est le nombre de mesures pour X et Y) :

$$Similarite(X,Y) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i-\bar{x}}{\phi_x}\right)\left(\frac{y_i-\bar{y}}{\phi_y}\right) \text{ avec } \phi_x = \sqrt{\sum_{i=1}^{N}\frac{(x_i-\bar{x})^2}{N}}$$

Cette méthode est loin d’être exempte de défauts. En particulier les arbres obtenus par ces techniques n’ont aucune signification biologique. De plus, la validation des résultats est presque impossible et les tentatives d’estimation de la fiabilité des clusters obtenus, concluent à une faible confiance [Tibshirani et al.]. Mais encore plus gênant, appliquer cet algorithme directement sur les données conduit à de très mauvais résultats car trop d’importance est donnée aux gènes non pertinents dans le calcul de la distance. Pour éviter ceci, les biologistes décident arbitrairement de supprimer tous les gènes qui ne sont pas exprimés au moins une fois au-dessus d’un certain seuil. Ce point est délicat car modifie totalement les arbres obtenus par clustering et revient à une sorte de filtration sommaire

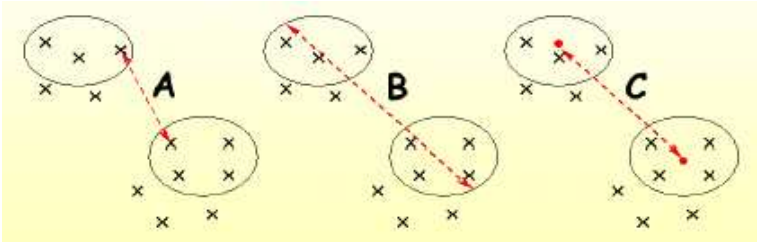


FIG. 4.4 : *Différentes techniques pour calculer la mesure de similarité d'un groupe de vecteurs à un autre ; A) lien simple B) lien complet C) lien moyen.*

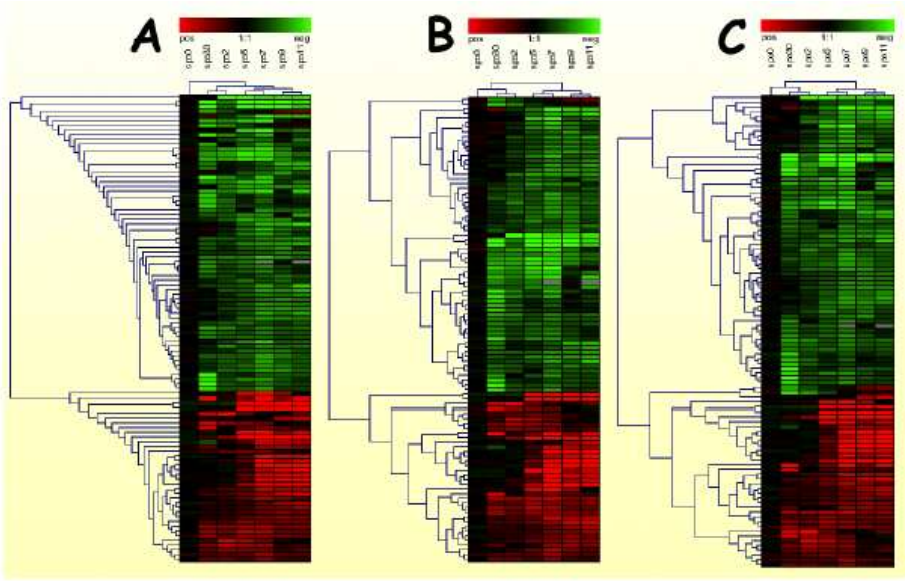


FIG. 4.5 : *Exemple de Clustering en utilisant les différentes mesures de similarités précédentes (fig 4.4) pour des données biologiques. Graphique tiré de [Marc; 2001] et généré avec Treeview.*

des données. De plus, ils ne se limitent pas à ce *cut-off* mais se permettent de remettre certains de leurs gènes favoris dans le cluster (s'ils avaient été éliminés).

4.2.3 Significance Analysis for Microarrays (SAM)

Il s'agit d'un outil statistique développé par Stanford pour l'analyse de résultats provenant de puce à ADN ([Tusher et al.; 2001]). Il se présente sous la forme d'une macro Excel. Clairement, cette technique suppose que les gènes ne sont pas corrélés entre eux et que leur expression dépend de s'ils sont irradiés ou non. On suppose de plus, que les écarts entre les différents niveaux d'expression d'un même gène à l'intérieur d'une même classe sont dues au "bruit" (biologique ou expérimental). Chaque gène i se voit attribué une note $d(i)$ (pour "relative difference") qui est :

$$d(i) = \frac{\bar{x}_I(i) - \bar{x}_S(i)}{s(i) + s_0}$$

où $\bar{x}_I(i)$ et $\bar{x}_S(i)$ sont les moyennes des niveaux d'expression pour le gène i dans les échantillons Irradiés (classe I) et Sains (classe S). $s(i)$ est l'écart-type pour le gène i.

$$s(i) = \sqrt{a\left\{\sum_m \left(x_m(i) - \bar{x}_I(i)\right)^2 + \sum_n \left(x_n(i) - \bar{x}_S(i)\right)^2\right\}} \tag{4.1}$$

avec $\sum_m$ et $\sum_n$ les sommes pour les expressions dans les classes I et S. $a = (1/n_1 + 1/n_2)/(n_1 + n_2 - 2)$; (n_1 et n_2 sont le nombre de mesures dans l'état irradié et sain). Le terme s_0 est ajouté dans l'expression pour minimiser la variation de $d(i)$. Il permet de ne pas favoriser des gènes qui auraient (par chance) une petite variance et du même coup une valeur de $d(i)$ grande. Le type de profils recherchés par cette formule est clairement deux paliers, les plus nets possibles avec la variance la plus faible possible pour chacun des paliers. Effectivement, il est raisonnable de s'intéresser à de tels gènes si on en trouve.

Se pose alors le problème du choix du seuil pour la valeur de d. C'est ici que réside le principal intérêt de SAM ; en effet, pour chaque gène, il calcule $d_E(i)$ qui correspond à la valeur attendue pour le gène i. Cette valeur est calculée en mélangeant les étiquettes des classes de toutes les façons possibles équilibrées pour chaque classe : toutes les permutations contiennent le même nombre de S et de I et il y a toujours la moitié des S et des I initiaux qui conservent leur classe ; ce qui impose d'avoir autant de S que de I au départ. Ensuite, on fait la moyenne de $d(i)$ pour chacune de ces permutations afin d'obtenir $d_E(i)$. On sélectionne ensuite les gènes qui s'écartent de la droite $d_E(i) = d(i)$. SAM évalue également un taux de fausses découvertes (False Discovery Rate : FDR) qui est le nombre moyen de gènes qui s'écartent de la droite dans les permutations précédentes. Cette méthode, en particulier à cause de l'estimation du FDR, est en train de devenir une référence dans le domaine de l'analyse de microarrays. Cependant il est possible d'émettre certaines critiques : tout d'abord, on présuppose la répartition en paliers (mais nous verrons ensuite

qu'elle est raisonnable). Ensuite, SAM ne permet pas de traiter plusieurs classes simultanément. Enfin, concernant la résistance au bruit, SAM est conçu pour tenir compte des erreurs de mesure mais résiste très mal aux valeurs aberrantes.

4.2.4 Utilisation des "Markov Blankets"

Le principe des couvertures de Markov est d'essayer d'identifier les attributs qui sont corrélés les uns aux autres ; et en particulier de repérer ceux qui ont l'air d'être corrélés à la classe mais qui sont en réalité corrélés à des gènes corrélés à la classe. Ce type d'approche a été utilisé dans des problèmes biologiques dans [Xing et al.; 2001]. Voyons comment procéder :

Soit G un sous-ensemble de l'ensemble de tous les attributs F. Soit f_G la projection de f (réalisation de F) sur les variables de G. C désigne l'étiquette (la Classe). Le but de l'utilisation des couvertures de Markov est de trouver un ensemble G (petit) pour lequel la différence entre les distributions $P(C|F = f)$ et $P(C|G = f_G)$ est le plus faible possible (idéalement nul). La mesure de la différence se fait en utilisant la mesure de l'entropie conditionnelle :

$$\Delta_G = \sum_f P(f) D(P(C|F = f) \| P(C|G = f_G))$$

où $D(P\|Q) = \sum_x P(x) \log(P(x)/Q(x))$ (distance de Kullback-Leibler)

Définition : Pour un ensemble d'attributs G et d'étiquettes C, l'ensemble $M_i \subseteq G(F_i \notin M_i)$ est une "Markov Blanket" (Couverture de Markov) de F_i tel que $(F_i \in G)$ si

$$(F_i \perp G - M_i - \{F_i\}, C)|M_i$$

$A \perp B$ signifie que $Covariance(A, B) = 0$ et $|$ est le "sachant" en probabilité. Il a été prouvé par Koller et Sahami que si M_i est une couverture de Markov pour F_i alors pour tout sous-ensemble E de F

$$\Delta_E = \Delta_{E-\{F_i\}}$$

ce qui est bien sûr très intéressant pour trouver G.

Intuitivement, dire que M_i est une couverture de Markov pour F_i, signifie que, pour une instance, si l'on connaît les valeurs pour les attributs de M_i, connaître la valeur de F_i n'est pas important pour déterminer la classe de l'instance. Mais il y a encore un peu plus d'information : on est sûr que le fait d'enlever plus tard un autre attribut $F_j \in M_i$ de G ne va pas rendre F_i pertinent de nouveau.

Cette méthode pourrait se révéler utile sur nos données. Cependant il ne faut pas oublier plusieurs points :

- Lors de la mise en oeuvre on ne travaillera pas avec les densités de probabilité exactes mais avec un algorithme qui les approche.
- La mise en oeuvre nécessite une discrétisation du problème. Il faudra être vigilant pour ne pas introduire d'erreurs supplémentaires.

- La méthode n'est pas réputée pour résister au bruit. C'est même plutôt l'inverse qui risque de se produire. En effet, si deux gènes ont systématiquement exactement la même valeur, ils sont des couvertures de Markov l'un pour l'autre (au moins dans l'approximation) donc l'un des deux risque de se faire éliminer. Ce n'est pas judicieux dans la mesure où pour la prédiction, les données seront bruitées et qu'il vaudra sans doute mieux utiliser plusieurs gènes pour prédire, plutôt qu'un seul.
- La mise en oeuvre des couvertures de Markov est coûteuse en temps de calcul : dans le cas idéal il faudrait examiner tous les sous-ensembles de F. Ceci est impraticable car il y a $2^{\#F}$ sous-ensembles. En pratique on se limite aux sous-ensembles de longueur l fixée, mais même ainsi, la complexité reste élevée.

4.2.5 RELIEF

RELIEF [Kononenko; 1994] est un algorithme de la classe « filter », c'est-à-dire, qu'il essaye, indépendamment de tout algorithme d'apprentissage, de déterminer quels sont les attributs les plus pertinents pour classer. Remarquons que RELIEF ne fournit pas de classification mais cherche à trouver les attributs les plus utiles pour établir une classification. Notons qu'il s'agit d'une méthode nécessitant que les données soient étiquetées (ici le label est l'appartenance à la classe « sain » ou « irradié »). RELIEF prend donc en entrée les données étiquetées et retourne un vecteur de « Poids ». Celui-ci a pour longueur le nombre d'attributs et contient une note qui essaye de refléter la pertinence de l'attribut pour classer. L'introduction du poids utilise une idée voisine de la mesure de l'entropie pour ID3. Il est constitué pour l'attribut A de l'estimation de la quantité suivante :

$Poids(A) = P($Valeur différente de A pour l'instance la plus proche de la classe opposée à A)
$-P($Valeur différente de A pour l'instance la plus proche de la même classe que A).

Initialement RELIEF est conçu pour prendre en compte deux valeurs pour chacun des attributs : {-1,1}. Il est facile de l'adapter pour traiter des attributs numériques. Cependant il faut garder à l'esprit qu'il répartira toujours les valeurs en deux types de comportements : les valeurs autour de 0 correspondent pour lui à des valeurs dont la classe n'est pas nette. Dans notre cas cette propriété est très intéressante, puisqu'un gène est grossièrement soit activé (valeurs positives) soit réprimé (valeurs négatives). Les valeurs autour de 0 concernant les gènes pour lesquels on ne voit pas de différence importante avec le témoin.

Plusieurs améliorations sont possibles : traitement de plus de deux classes, des valeurs manquantes, et plus intéressant, augmentation de la résistance au bruit. Pour cela, on utilise k voisins au lieu d'un seul.L'algorithme d'évaluation du $Poids$ est donc :

Pour tous les Attributs A, Poids[A] :=0
Pour i :=1 **jusqu'au** nombre_d'instances **faire**

Début
R :=instance no i
trouver les k plus proches voisins de la même classe que R : $H_1 \ldots H_k$
trouver les k plus proches voisins de classe différente de celle de R : $M_1 \ldots M_k$
Pour tous les attributs A
Poids[A] :=Poids[A]-diff(A,R,$H_1 \ldots H_k$)+diff(A,R,$M_1 \ldots M_k$)
Fin
Pour tous les attributs A, Poids[A] :=Poids[A]/nb_d'instances

Avec diff(A,R,$H_1 \ldots H_k$) qui est le calcul de la différence entre la valeur de A pour R et la moyenne des valeurs de A pour $H_1 \ldots H_k$.

Si les valeurs des attributs sont comprises pour chaque lame dans $[-1, 1]$; alors pour k=1, l'algorithme précédent donne bien une estimation raisonnable de la différence des probabilités voulues (il s'agit de l'estimateur du maximum de vraisemblance). Cet algorithme peut bien sûr s'étendre à des attributs dont les valeurs ne sont pas dans $[-1, 1]$, dans ce cas l'algorithme ne fournit plus exactement une estimation de la probabilité, mais une valeur qui lui est proportionnelle (tous les poids sont multipliés par le sup des valeurs des attributs dans l'expérience).

Ce point est en fait lié à une autre difficulté : supposons que A_1 et A_2 soient deux attributs présents dans le jeu de données et que C_1 et C_2 soient les deux classes possibles pour les instances. Pour simplifier on suppose qu'il n'y a pas de bruit. Imaginons que :

- Pour A_1
  - -1 signifie que A_1 est réprimé
  - 0 signifie que A_1 est exprimé de manière normale
  - 1 signifie que A_1 est sur-exprimé

- Pour A_2
  - -2 signifie que A_2 est réprimé
  - 0 signifie que A_2 est exprimé de manière normale
  - 2 signifie que A_2 est sur-exprimé

On voit tout de suite le problème : A_2 sera très favorisé par RELIEF car ses valeurs sont plus grandes. Une solution serait dans ce cas de renormaliser attribut par attribut afin que chacun prenne ses valeurs dans $[-1, 1]$. Ici il s'agit effectivement d'une solution. Mais dans le cas réel, les données sont bruitées ; du coup, un attribut qui devrait idéalement toujours prendre la valeur 0 (ce qui est le cas pour la majorité des gènes dans nos mesures), avec le bruit, va être artificiellement étiré et on lui donnera une importance qu'il n'a pas. Si on a de la chance, le bruit ne sera pas corrélé à la classe et on pourra quand même s'en sortir. Cependant, la détermination des plus proches voisins (point crucial) va être fortement perturbée. Dans nos données nous avons choisi de ne pas effectuer cette correction d'échelle ; ceci signifie que l'on favorise les gènes qui ont des niveaux d'expression élevés. D'après les biologistes, un gène activé prendra typiquement une valeur autour de 1

(pour le $\log_2$ du ratio), mais il existe certains gènes pouvant s'exprimer davantage (jusqu'à la valeur 3 ou 4). Ce choix de favoriser de tels gènes est motivé par le fait qu'à cause de la présence du bruit, ceux-ci ont plus de chances d'être détectés dans nos expériences. Ce choix a été fait d'après les désirs des biologistes car étant donné la présence de bruit, on préfère avoir des différences nettes (de plus lors de la technique de mesure, la fiabilité est plus grande pour les valeurs élevées).

Nous avons légèrement modifié RELIEF pour pouvoir choisir une valeur différente de k selon chacune des classes considérées lors de l'apprentissage. Ceci, car lors d'expérimentations sur des données artificielles, nous nous sommes rendu compte qu'il était souvent intéressant d'avoir un k assez élevé par rapport au nombre total d'individus dans une classe. Nous pouvons donc traiter différemment plusieurs classes, ce qui est intéressant surtout dans le cas ou celles ci présentent un nombre d'exemples déséquilibré.

4.2.6 Le test de Wilcoxon

On peut poursuivre dans cette voie, qui est de regarder quelles sont les répartitions des valeurs des expressions des gènes. Le test de Wilcoxon est un test statistique idéal pour cette approche :

- Il est adapté aux petits échantillons
- Il n'y a pas de supposition sur la répartition de la valeur des gènes

Mais ce test a l'inconvénient d'être sensible aux valeurs aberrantes. Le principe est pour chaque gène G de trier les niveaux d'expression de G dans toutes les expériences. On regarde ensuite la somme des rangs pour la classe 1 et pour la classe 2. Intuitivement, si pour les deux classes, les valeurs des attributs sont tirées selon la même distribution, les deux valeurs vont se situer dans une fourchette dépendant du nombre d'individus de classe 1 et de classe 2. Wilcoxon a formalisé ceci et construit des tables de valeurs.

On constate qu'au niveau 5% l'expression de 1500 gènes est considérée comme significativement différente selon la présence ou non de l'irradiation (588 au niveau 1%). (fig. 4.6). Puisqu'il y a initialement 6157 gènes dans notre jeu de données, en théorie, au niveau 1%, le nombre de gènes ayant un comportement semblable suit une loi binômiale $\mathcal{B}(\frac{1}{100}, 6157)$. C'est-à-dire que en moyenne on a 6157 gènes avec ce type de répartition, l'écart type étant de 60,9543. On constate que l'on est loin des 588 observations réalisées.

4.2.7 Retour sur RELIEF

Un des inconvénients que nous avons évoqués pour RELIEF est de choisir un seuil pour sélectionner les attributs. De plus, nous ne savons pas quelle confiance accorder à la sélection effectuée. Pour tenter de se faire une idée, il nous est apparu important de comparer les notes obenues par RELIEF à celles obtenues pour des données n'ayant pas de sens particulier. Comme nous ne disposons pas d'un modèle fiable pour les données,

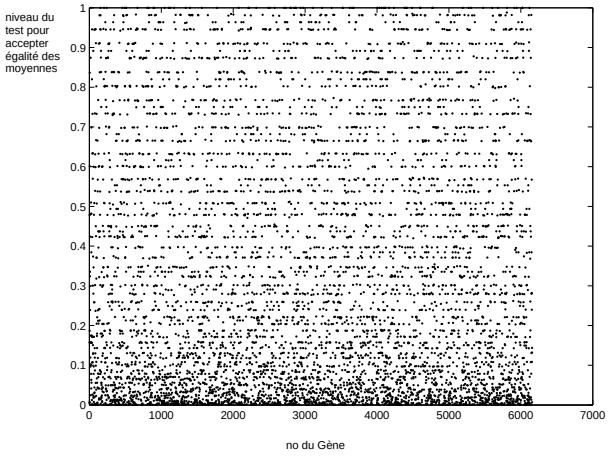


FIG. 4.6 : *Visualisation, pour chacun des gènes, de la probabilité d'observer une répartition des mesures identiques à celle mesurée sous l'hypothèse que l'expression des gènes est indépendante de la classe (hypothèse nulle). La présence de "lignes" sur lesquelles s'alignent les valeurs est due à la faible taille de l'échantillon. On constate que de nombreux gènes ont, sous l'hypothèse nulle, une probabilité faible d'obtenir le profil qu'ils ont dans l'expérimentation.*

nous nous sommes inspirés de SAM : nous avons conservé le même jeu de données mais nous avons réparti au hasard les classes. Nous avons tout de même conservé la répartition 12 sains et 6 irradiés. On observe ensuite le nombre d'attributs qui dépassent un seuil fixé. En répétant cette expérience pour divers tirages de classes, on peut se faire une idée du nombre moyen d'attributs qui obtiennent pour RELIEF un poids supérieur à un seuil. On peut même répéter cette expérience pour différentes valeurs des paramètres k pour RELIEF.

L'analyse de la figure 4.7 confirme les résultats donnés par Wilcoxon. En effet, les courbes sont décalées : l'écart atteint environ 500 gènes entre la courbe moyenne et la courbe correspondante. C'est intéressant car ceci laisse supposer qu'il y a plus de 50 gènes impliqués dans la réponse à l'irradiation. La même expérience réalisée avec des données artificielles permet de se rendre compte que pour obtenir un tel écart, il faut avoir au moins 500 gènes corrélés à la classe.

Ces courbes sont intéressantes par elles-mêmes mais constituent également un guide dans le choix du seuil pour RELIEF. En effet, grâce à la courbe moyenne, on peut se faire une idée du taux de faux positifs dans la sélection effectuée par un seuil donné. De plus on peut se rendre compte d'un problème désormais classique en apprentissage : on doit choisir entre la précision et le rappel.

Si l'on favorise la précision (seuil de sélection élevé), on aura une liste de gènes courte avec peu de gènes non pertinents mais on manquera des gènes. Par contre avec un rappel élevé, c'est l'inverse : on a une liste plus longue contenant presque tous les gènes pertinents mais avec des erreurs. Les deux chapitres suivants traitent de l'utilisation de l'optimisation de ce compromis pour la sélection d'attributs (à travers l'étude des courbes ROC pour la sélection).

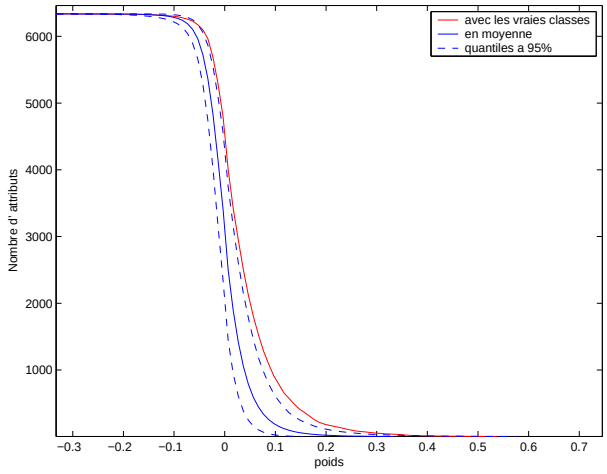


FIG. 4.7 : *Nombre de gènes dont la notation pour RELIEF dépasse le seuil fixé en abscisse. Les courbes moyennes correspondent à des notations obtenues en permutant les classes* 10000 *fois et en regardant combien de fois le seuil est dépassé en moyenne. On représente également les intervalles de confiance à* 95% *pour chaque point de la courbe.*

4.2.8 Optimisation de k_{IR} et k_{NI}

Afin de déterminer les meilleures valeurs pour les paramètres k_{IR} et k_{NI} (nombre de voisins à utiliser dans chacune des deux classes) de RELIEF, il nous faut un critère. Faire de la *cross-validation* parraît peu raisonnable, d'une part car il faudrait exclure au moins un exemple (pour faire un *jack-knife*) de la base d'apprentissage qui n'en comporte pas beaucoup, et d'autre part parcequ'il faudrait ajouter un algorithme de classification à la sélection d'attributs. La solution que nous avons utilisée pour contourner le problème est d'essayer de maximiser la différence de comportement avec le hasard. C'est-à-dire que l'on réutilise la courbe des poids moyens (obtenue en permuttant les *labels*) pour calculer l'aire de la différence avec la courbe obtenue avec les vraies étiquettes.

4.3 Conclusion

J'ai passé volontairement assez rapidement sur ces travaux ; ils sont ici davantage pour présenter les difficultés de ce type de problèmes et pour introduire le chapitre suivant.

En particulier nous avons aussi fait des essais de classification sur de nouvelles lames fournies en aveugle par les biologistes. Les résultats ont montrés qu'il était utile de faire une sélection d'attributs. Signalons toutefois que les lames n'ayant pas reçu exactement les mêmes traitements que celles de l'apprentissage, il est prudent de pas conclure trop rapidement. Par contre, de manière plus intéressante, les meilleurs gènes (ceux pour lesquel le ratio « courbe avec les vraies classes » sur « courbe moyenne » atteignait son maximum) ont fait l'objet d'une étude plus poussée par l'équipe de l'institut Curie. Leur idée a été de regarder parmi les gènes quels étaient ceux dont la fonction était déjà connue. Il

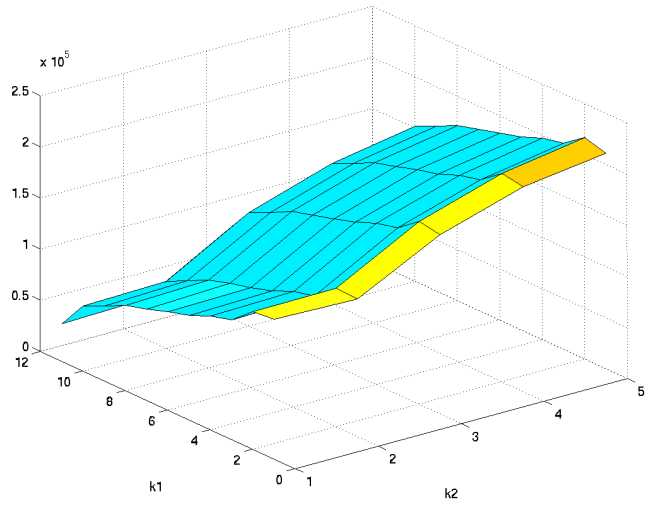


FIG. 4.8 : *Visualisation de la différence d'aire entre la courbe moyenne et la courbe avec les vraies classes en fonction du choix de k pour chacune des deux classes.*

en est ressorti le tableau suivant :

fonction of 91 genes induits/171	**nb d'ORFs**	**% dans cette liste**	**% total ORFS**	**sur-rep**
unknown	38	41,8	50,4	0,8
oxidative stress response	4	4,4	0,3	14,3
oxidative phosphorylation	9	9,9	0,3	30,5
transport	4	4,4	2,2	2,0
gluconeogenesis	1	1,1	0,1	16,9
protein processing & synthesis	3	3,3	2,0	1,6
ATP synthesis	7	7,7	0,4	20,6
glucose repression	1	1,1	0,2	4,8
respiration	2	2,2	0,1	22,0
fonction de 80 genes réprimés/171	**nb d'ORFs**	**% dans cette liste**	**% total ORFS**	**sur-rep**
unknown	45	56,3	50,4	1,1
stress response (putative)	1	1,3	0,2	7,0
glycerol metabolism	2	2,5	0,1	30,8
protein processing & synthesis	3	3,8	2,0	1,9
secretion	2	2,5	2,0	1,3
transport	4	5,0	2,2	2,3
glycolysis	2	2,5	1,0	2,0

Ces gènes ont d'abord été regroupés selon les voies métaboliques dans lesquelles ils sont impliqués. La fréquence de ces voies dans la liste des gènes induits ou réprimés par l'exposition aux faibles doses de radiations a été comparée à celle de ces voies métaboliques sur l'ensemble de la lame. Plus précisément, la deuxième colonne de la partie supérieure (respectivement inférieure) du tableau indique le nombre d'ORFs induites (resp. réprimées) parmi les 171 (il s'agit de l'intersection des meilleurs gènes pour deux méthodes différentes dont RELIEF, pour davantage de détails voir [Mercier et al.; 2004])

sélectionnées, qui participent à l'activité mentionnée en première colonne (*e.g.* 9 gènes sur les 171 sont induits et correspondent à la fonction d'oxidative phosphorylation). La troisième colonne donne le rapport entre le nombre d'ORFs induites (resp. réprimées) sélectionnées participant à cette activité et le nombre d'ORFs induites (resp. réprimées) sélectionnées, soit 91 (resp. 80) (*e.g.* 9/91 = 9.9% des 91 gènes induits correspondent à l'oxidative phosphorylation) ; la quatrième colonne indique le pourcentage des ORFs ayant cette activité (on considère alors l'ensemble de tous les gènes de la levure) (e.g. 0.3% de tous les gènes induits correspondent à l'oxidative phosphorylation). La dernière colonne donne alors le taux de sur-représentativité de l'activité considérée dans la population des 171 meilleurs gènes sélectionnés (e.g. la fonction d'oxidative phosphorylation est surreprésentée dans les 91 gènes induits sélectionnés par rapport à sa représentation moyenne dans les gènes induits parmi 6157 : soit 9.9% / 0.3% = 30.5).

Sur le tableau, on voit ainsi que 20 gènes participant au stress oxydatif, à la phosphorylation oxydative et à la synthèse d'ATP sont induits, ce qui représente un excédent de 30 fois par rapport à la population totale.

Sans entrer dans le détail de l'interprétation biologique, de ces groupes fonctionnels, il est cependant possible d'en conclure que la méthode de sélection de gènes utilisée dans cette étude est validée par le fait que les fonctions mises en évidence par l'analyse des gènes sélectionnés sont connues pour intervenir dans l'élimination de certains des produits cellulaires des rayonnements ionisants (radicaux libres). (Des informations supplémentaires sur ces fonctions sont publiées dans [Mercier et al.; 2004]).

Quelques mots de conclusion sur ce sujet. Tout d'abord, le problème par les puces à ADN, malgré sa complexité, laisse la place à des analyses simples, telles que la classification en utilisant un seul gène, ou la recherche d'un concept conjonctif. La difficulté réside plutôt dans l'impossibilité, dans l'état actuel de nos connaissances, de faire la différence entre un concept conjonctif bruité par l'expérimentation et un concept disjonctif (surtout avec aussi peu d'exemples).

Ensuite, sur le jeu de données de Marie Dutreix, il semble que contrairement à l'intuition des biologistes, le nombre de gènes influencés par la présence de l'irradiation soit assez important (probablement autour de 400 alors qu'on en attendait une cinquantaine). Cette impression est confirmée en essayant de faire des intersections de techniques différentes de sélection et en évaluant la part de sur-représentation de la partie commune.

Enfin, il semble plus que jamais important de travailler de concert avec les biologistes afin de bénéficier de leur savoir lors de la phase de validation. En effet, c'est de loin le point le plus délicat soulevé par ce problème. Ce point milite en faveur de l'utilisation de techniques de sélection d'attribut plutôt que de classification. En effet les biologistes acceptent facilement de travailler sur une liste de gènes pour construire une théorie bilogique collant bien au problème, mais sont souvent incapables pour des raisons financières de nous fournir suffisament de données pour qu'une validation croisée soit significative.

Chapitre 5

Sélection Active

Classiquement en apprentissage, on demande beaucoup plus d'exemples que d'attributs afin d'éviter des problèmes tels que le sur-apprentissage. Dans ce cas idéal où l'on dispose facilement de beaucoup d'exemples, les méthodes de sélections d'attributs étaient principalement utilisées pour réduire le temps d'exécution des algorithmes ou réduire la taille de la base de données. Toutefois, avec l'apparition de la Bio-Informatique, en particulier dans l'analyse des puces à ADN, la sélection d'attributs est revenue en force. En effet, le biologiste trouve bien souvent plus intéressante la liste de gènes impliqués dans un processus que la méthode utilisée pour classer. De plus des coûts financiers imposent de limiter le nombre de mesures effectuées, interdisant de se trouver dans les « normes » d'utilisation des algorithmes courants. Remarquons que cette pénurie de données se retrouve aussi dans d'autres domaines comme en text-mining ou l'indexation d'images où cette fois, l'on dispose de nombreuses données mais dont l'étiquetage doit être fait par un expert humain, avec tous les inconvénients que cela comporte (coûts, risque d'erreur...)

Je présenterai ici une nouvelle méthodologie pour la sélection active, l'idée est d'utiliser des connaissances *a priori* sur les algorithmes que l'on va utiliser (*via* leur courbe ROC) pour sélectionner les attributs les plus pertinents en économisant le nombre de mesures effectuées. On se met ainsi, par exemple à la place du biologiste qui possède une somme fixe d'argent et qui désire l'allouer de la meilleure manière.

5.1 Le problème

5.1.1 Description

On suppose que l'on dispose de N_0 attributs (par exemple des gènes) potentiellement pertinents. On se doute qu'il existe un lien entre une partie de ces attributs et la classe d'un exemple. En reprenant l'exemple des gènes, cela signifie que l'on fait l'hypothèse que l'observation de l'activation du génome permet de déterminer si l'individu observé

est sain ou malade (dans ce cas la classe est « sain » ou « malade » et un exemple est un individu dont on connait l'expression du génome et sa classe).

Pour identifier les attributs utiles pour prédire la classe, il est nécessaire d'observer un certain nombre d'exemples. Malheureusement, les exemples ne sont que rarement gratuits. Dans le cas de notre biologiste, celui-ci devra débourser une somme fixe (pour simplifier, disons 1 unité) chaque fois qu'il souhaite mesurer le niveau d'activation d'un gène chez une personne. Supposons pour commencer, qu'il s'agisse d'un biologiste fortuné. Dans ce cas, pas de problème, il va commander la mesure complète de X individus. Le coût sera donc $X \cdot N_0$. Comme les algorithmes d'apprentissage préfèrent avoir davantage d'exemples que d'attributs, il aura intérêt à choisir $X \gg N_0$. Mauvaise nouvelle, dans la génomique, N_0 est au minimum de l'odre de 5000 et peut aller jusqu'à 30000. Le coût d'une telle opération serait donc entre 25000000 et $900000000\ldots$

Mais tout cela ne tiens pas compte d'une chose : de nombreux attributs, peuvent être éliminés rapidement. En effet si on étudie une maladie du foie, il y a peu de chance pour que les gènes impliqués dans la couleur des cheveux aient une quelconque importance. Du coup si l'on arrive à repérer suffisament tôt ces gènes, on pourra cesser de demander leur mesure et donc économiser sur le coût total de l'expérience. Reste à faire cela de la manière la plus astucieuse possible.

5.1.2 Courbes ROC et Apprentissage

On dispose d'un algorithme permettant d'itentifier les attributs pertinents pour la classification. En pratique, celui-ci se contente en fait d'établir un classement des attributs (éventuellement en leur attribuant une note). On se trouve donc obligé après son passage de déterminer un seuil séparant les attributs que l'on choisit de conserver et ceux que l'on rejette.

En fonction de cette sélection, on distingue quatre types d'attributs (voir figure 5.1.2).
- VP : Les Vrais Positifs. Ce sont les attributs qui sont pertinents et qui ont été détectés comme tels par l'algorithme,
- FP : Les Faux Positifs. Ce sont des attributs non-pertinents qui ont été considérés comme pertinents par l'algorithme. En statistiques, on parlerait d'erreur de seconde espèce,
- VN : Les Vrais Négatifs. Ce sont des attributs non-pertinents qui ont été détectés comme tels,
- FN : Les Faux Négatifs. Ce sont des attributs ayant étés rejetés par l'algorithme et qui sont effectivement non-pertinents.

Notons immédiatement que $VP + FP + VN + FN = N_0$ (puisque ces catégories partitionnent sans recouvrement l'ensemble des attributs).

Pour un algorithme idéal on a $FP = FN = 0$. Puisque le monde n'est pas parfait, on essaye simplement d'avoir FP (resp .FN) petit par rapport au nombre total d'attributs

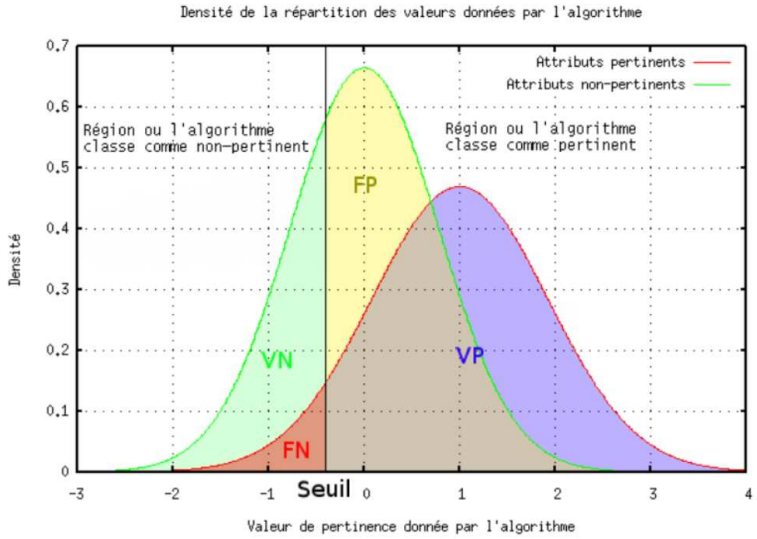


FIG. 5.1 : *Visualisation du positionnement de VP, FP, VN, FN en fonction du seuil de tolérance. Pour une séparation parfaite, on voudrait $FN = FP = 0$. Plus le seuil est faible -sélection faible- plus on réduit FN (mais FP grandit) et plus le seuil est élevé -sélection forte- plus on réduit FP (mais FN grandit).*

pertinents $VP + FN$ (resp. non-pertinents $VN + FP$). Classiquement, on définit deux notions :

- La précision p, qui mesure la probabilité qu'un attribut donné comme pertinent par l'algorithme le soit effectivement. $p = \dfrac{VP}{VP + FP}$
- Le Rappel r, qui mesure la proportion d'attributs positifs détectés par l'algorithme. $r = \dfrac{VP}{VP + FN}$.

Un algorithme idéal présenterait une précision et un rappel de 1. Dans la pratique, ces deux notions sont antagonistes. En effet, il est facile d'avoir une précision élevée, il suffit pour cela de ne déclarer pertinents que des attributs pour lesquels il ne subsiste que peu de doutes. L'inconvénient est alors que tous les autres attributs sont déclarés non pertinents (on peut faire l'analogie avec un étudiant qui ne répondrait qu'aux questions dont il est sûr de la réponse, quitte a ne pas répondre aux autres, ou encore avec la notion de système correct en logique). Il est encore plus simple d'avoir un rappel de 1 : il suffit de déclarer que tous les attributs sont pertinents. En effet tous les attributs pertinents seront forcément dans la liste, on se doute bien que la précision de cette technique n'est pas fabuleuse... On peut cette fois faire l'analogie avec un étudiant qui répondrait systématiquement à toutes les questions quitte à se tromper.

Il semble donc intéressant de disposer d'une courbe donnant pour un algorithme donné, la précision en fonction du rappel (cette courbe étant obtenue en faisant bouger le seuil de sélection de notre algorithme). Cependant, si le rappel varie toujours entre 0 et 1 en fonction du choix de notre seuil, il n'en va pas de même pour la précision qui elle

varie entre la proportion initiale de pertinents dans N_0 et les performances maximales de l'algorithme. On peut même imaginer que la précision ne soit pas une fonction croissante en la valeur du seuil. Pour remédier à cet inconvénient, on remplace la précision par le taux de faux négatifs $\tau = \frac{FP}{VP+FN}$. On obtient alors ce que l'on appelle la courbe ROC de l'algorithme (Receiver Operating Characteristic). Celle-ci peut être envisagée comme une « courbe de caractéristiques d'efficacité ».

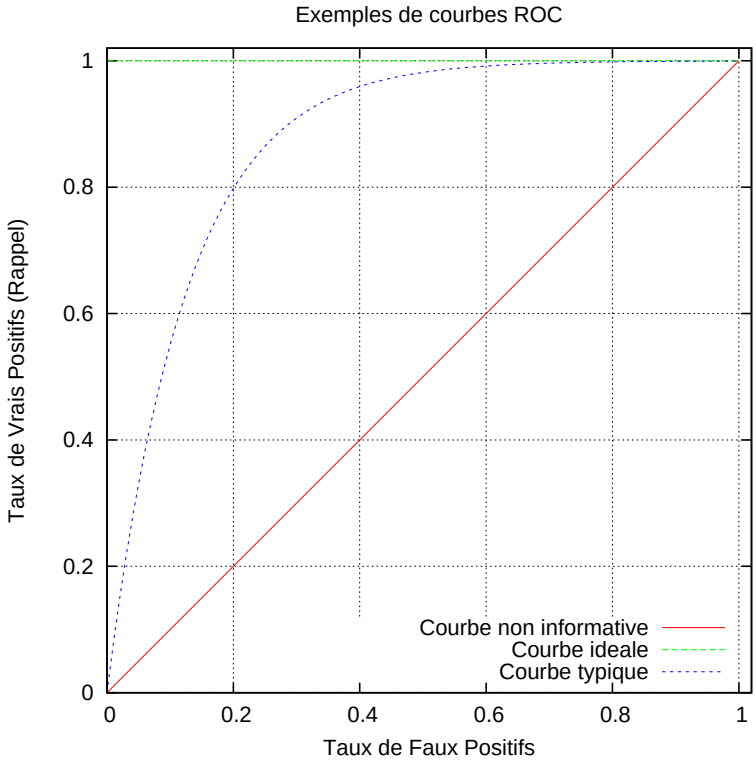


FIG. 5.2 : *Exemples de coubes ROC. Remarquez l'ordre de grandeur de l'aire sous chacune des trois courbes.*

Voyons tout de suite, quelques cas particuliers de courbes (figure 5.1.2) :

- Le courbe ROC idéale, est le point $(0, 1)$. Il s'agit d'un cas limite ou l'algorithme détecte parfaitement tous les attributs pertinents sans jamais se tromper.
- Pour un algorithme non informatif, on se situe sur la diagonale qui passe par le point 0. En effet, si le seuil ne permet pas de discriminer les pertinents des non-pertinents, la proportion d'attributs détectés comme positifs est la même parmi les attributs pertinents et non-pertinents.
- Trouver une courbe sous-diagonale pour un algorithme est louche. En effet cela signifie qu'il sépare effectivement les attributs pertinents des non pertinents mais en se trompant : il déclare positif préférentiellement des attributs non pertients et négatif des attributs pertinents. Souvent il s'agit d'un bête *bug* sur la sortie des résultats de l'algorithme.

La courbe ROC usuelle est une fonction croissante concave partant de $(0, 0)$ pour arriver en $(0, 1)$. Elle est d'autant meilleure que la courbe se situe proche du point $(0, 1)$

et loin de la diagonale. En théorie, le caractère croissant et concave n'est pas obligatoire mais au vu de la méthode de détection de notre algorithme (classement des attributs et choix d'un seuil), il ne peut pas en être autrement.

Suite à ces quelques remarques, il vient naturellement l'idée de résumer la qualité de l'agorithme par l'aire se situant sous la courbe ROC. C'est ce que l'on appelle le critère AUC (Area under the Curve).

- Une AUC de 1 est le résultat idéal ;
- Une AUC de $1/2$ représente le cas non informatif (la diagonale) ;
- Une AUC inférieure à $1/2$ est louche...

5.1.3 Forme de la courbe ROC

Plus l'algorithme est performant plus sa coube ROC est proche de la courbe idéale. En fait il n'est pas correct de parler de « la » courbe ROC d'un algorithme car celle-ci dépend bien sûr des données considérées.

- Généralement, plus on fournit d'exemples d'apprentissage à l'algorithme, plus celui-ci se comporte bien.
- D'un petit nombre d'attributs entraine de meilleures performances.
- Mécaniquement, la valeur de Δ (nombre d'attributs réellement pertinents) a elle aussi un rôle dans l'allure de la courbe ROC. En effet, même si on travaille avec des taux pour éviter les problèmes les plus flagrants, il n'en demeure pas moins que Δ a une influence...
- De l'adaptation de l'algorithme au données traitées. Ainsi la courbe ROC d'un algorithme de sélection ne considérant pas les effets croisés sera très mauvaise par exemple sur un concept disjonctif.
- Parfois on peut même observer une part stochastique dans la prise de décision. Dans ce cas, la courbe ROC devient en fait une espérance.

A partir de ces constatations, on peut-construire une série de pseudo-courbes ROC artificielles (correspondant à un algorithme de sélection imaginaire). On peut également estimer les performances d'un algorithme réel. Pour cela nous avons besoin de données pour lesquelles les attributs vraiment pertinents sont connus (soit provenant d'un problème résolu, soit d'un problème artificiel). En effet, il n'est pas possible dans le cas de la sélection d'estimer la courbe ROC (pour la sélection) par validation croisée (dans ce cas la courbe ROC estimée est celle de la classification).

Pour générer des courbes ROC évoluant selon des paramètres nous avons utilisés deux méthodes. La première est complètement artificielle : on décide analytiquement de la forme à donner (voir la figure 5.3 pour des courbes générées à partir d'exponentielles). Une seconde méthode consiste a prendre un algorithme réel et à le faire tourner sur des données dont on connaît par avance quels sont les attributs pertinents (en les générant artificiellement par exemple). C'est ce qui a été fait sur la figure 5.4.

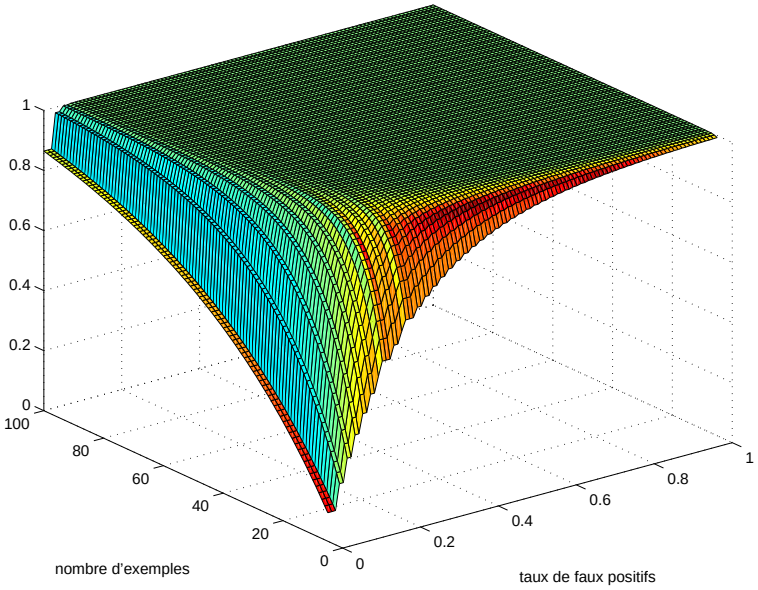


FIG. 5.3 : *Evolution de courbes ROC artificielles en fonction du nombre d'exemples.*

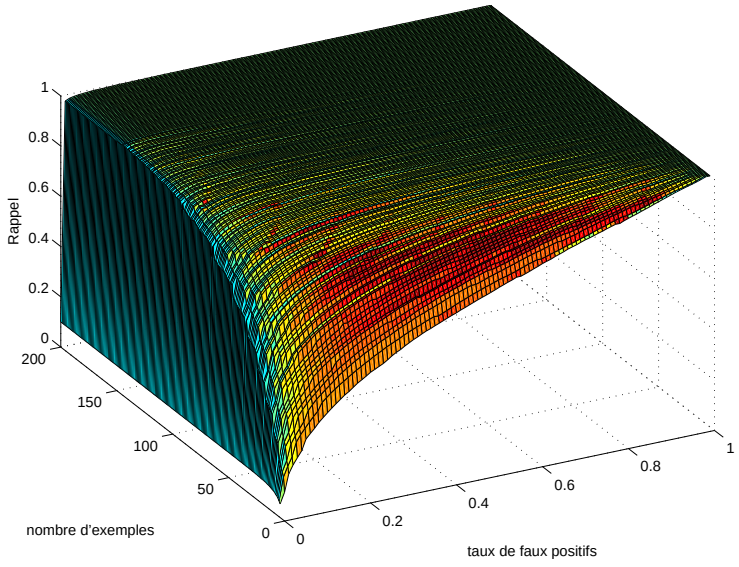


FIG. 5.4 : *Evolution des courbes ROC de RELIEF avec deux voisins dans chaque classe sur des jeux de données artificiels en fonction du nombre d'exemples (*100 *attributs,* 10 *pertinents, bruit* 0.5*).*

5.2 Séquence Active d'apprentissage

5.2.1 Algorithme

On suppose que l'on dispose d'un algorithme A (ou éventuellement d'une série d'algorithmes) capables de sélectionner un sous ensemble d'attributs à partir d'un ensemble d'exemples. Cet algorithme est caractérisé par ses courbes ROC qui lient τ, le taux de faux positifs au rappel r. De plus on suppose que plus l'on fournit d'exemples à A plus sa courbe ROC se rapproche de la courbe idéale.

Pour obtenir ces exemples, on suppose que l'on doit payer une quantité proportionnelle au nombre d'attributs mesurés. Notons C_{max} le nombre maximal de mesures d'attributs que l'on peut effectuer. Au départ nous disposons de N_0 attributs. Dans une utilisation classique de A, on demandera $\lfloor N_0/C_{max} \rfloor$ exemples puis on choisira le compromis voulu entre r et τ sur la courbe ROC correspondante.

Pour des raisons pratiques, on supposera que k_0 exemples contenant N_0 attributs sont offerts. Cela ne change pas le problème d'optimisation mais permet de tenir compte d'algorithmes qui nécessitent un nombre minimal d'exemples pour pouvoir être utilisés (tel RELIEF).

Au commencement, on dispose donc de N_0 attributs dont seulement Δ sont utiles (le but étant de les identifier le mieux possible). On dispose déjà de k_0 exemples (de taille N_0). Notre stratégie active, consiste à éliminer progressivement les attributs dont on pense qu'ils ne sont pas pertinents afin de se focaliser sur les attributs pour lesquels il est plus difficile de trancher. Le i^e pas de notre stratégie consiste à conserver les $\sum_{j=0}^{i-1} k_j$ exemples dont on dispose déjà et a en demander k_i nouveaux. Ces nouveaux exemples ne contiennent d'information que pour les attributs qui ont été considérés comme pertinents au pas précédent ($i-1$).

Tout ceci peut se résumer ainsi :

$$\begin{array}{ccccccc} k_0 \text{ exemples} & & & k_0+k_1 \text{ exemples} & & \sum_{i=0}^{3} k_i \text{ exemples} & \\ N_0 \text{ attributs} & \overset{A}{\longrightarrow} & & N_1 \text{ attributs} & \overset{A}{\longrightarrow} & N_2 \text{ attributs} & \dots \\ & \text{choix de } k_1, r_1, \tau_1 & & \text{(choisis par } A\text{)} & \text{choix de } k_2, r_2, \tau_2 & \text{(choisis par } A \text{ dans les } N_1 \text{ précédents)} & \end{array}$$

Il faut cependant faire attention dans ce processus à bien distinguer la différence entre le rappel et taux de faux positifs lors de chaque application de A avec le rappel et taux de faux positif du processus complet (c'est à dire comme si l'on était passé en une seule étape de N_0 attributs à N_i). On notera VP_i, VN_i, FP_i, FN_i les valeurs locales $\underline{VP_i}, \underline{VN_i}, \underline{FP_i}, \underline{FN_i}$ les valeurs pour le processus complet à l'étape i.

On a les liens suivants entre ces valeurs :

$\underline{VP_i} = VP_i$ car les attributs considérés comme pertinents sont uniquemement sélectionnés lors de la dernière étape.

$\underline{FP_i} = FP_i$ pour la même raison.

$\underline{VN_i} = \sum_{j=0}^{i} VN_j$ Les attibuts considérés non-pertinents sont tous les attributs rejetés lors d'une étape précédente.

$\underline{FN_i} = \sum_{j=0}^{i} FN_j$ pour la même raison.

De plus on peut écrire :

$$\begin{aligned} \underline{VP_0} &= \Delta \\ \underline{VN_0} &= 0 \\ \underline{FP_0} &= N_0 - \Delta \\ \underline{FN_0} &= 0 \end{aligned}$$

Car à ce stade, tous les attribus sont «retenus».

Soit $\mathcal{H}_s$ l'hypothèse :

$$\begin{aligned} \underline{VP_s} &= \Delta \cdot \prod_{i=1}^{s} r_i \\ \underline{VN_s} &= \sum_{i=1}^{s} (1-\tau_i)(\prod_{j=1}^{i-1} \tau_j)(N_0 - \Delta) \\ \underline{FP_s} &= (\prod_{i=1}^{s} \tau_i)(N_0 - \Delta) \\ \underline{FN_s} &= \sum_{i=1}^{s} (\prod_{j=1}^{i} (1-r_j))\Delta \end{aligned}$$

Nous allons prouver par récurrence que pour toute étape $s > 0$, $\mathcal{H}_s$ est vraie.

Pour $s = 1$, en utilisant la définition du rappel et du taux de faux positifs, on a :

$$\begin{aligned} \tau_1 &= \frac{FP_1}{FP_1 + VN_1} \\ r_1 &= \frac{VP_1}{VP_1 + FN_1} \end{aligned}$$

En utilisant cette fois les définitions de Δ et N_0 on obtient :

$$\begin{cases} (1-\tau_1)FP_1 + \tau_1 VN_1 = 0 \\ (1-r_1)VP_1 - r_1 FN_1 = 0 \\ VP_1 + VN_1 + FP_1 + FN_1 = N_0 \\ VP_1 + FN_1 = \Delta \end{cases}$$

Ce qui nous conduit à :

$$\begin{aligned} VP_1 &= \Delta \cdot r_1 \\ VN_1 &= (1-\tau_1)(N_0 - \Delta) \\ FP_1 &= \tau_1(N_0 - \Delta) \\ FN_1 &= (1-r_1)\Delta \end{aligned}$$

De plus $\underline{VN_1} = VN_1$ et $\underline{FN_1} = FN_1$ donc l'hypothèse est valable pour $s = 1$ (cas de base ou d'initialisation de la récurrence). Notons que $N_i = VP_i + FP_i$.

Supposons maintenant que $\mathcal{H}_s$ est vraie pour un rang s. Nous allons montrer qu'elle l'est aussi au rang $s+1$.

Au pas $s+1$ on a

$$\begin{cases} (1-\tau_{s+1})FP_{s+1} + fp_{s+1} VN_{s+1} = 0 \\ (1-r_{s+1})VP_{s+1} - r_{s+1} FN_1 = 0 \\ VP_{s+1} + VN_{s+1} + FP_{s+1} + FN_{s+1} = N_1 = VP_s + FP_s \\ VP_1 + FN_1 = VP_s \end{cases}$$

Ce système admet pour solution :

$$\begin{aligned} VP_{s+1} &= r_{s+1} VP_s \\ VN_{s+1} &= (1-\tau_{s+1})FP_s \\ FP_{s+1} &= \tau_{s+1} FP_s \\ FN_{s+1} &= (1-r_{s+1})VP_s \end{aligned}$$

Avec l'hypothèse de récurrence $\mathcal{H}_s$, on obtient en utilisant le lien entre les valeurs locales et globales :

$$\begin{aligned} \underline{VP_{s+1}} &= (\Delta \cdot \prod_{i=1}^{s} r_i) \cdot r_{s+1} \\ \underline{VN_{s+1}} &= (\sum_{i=1}^{s}(1-\tau_i)(\prod_{j=1}^{i-1}\tau_j)(N_0 - \Delta)) + (1-\tau_{s+1})(\prod_{i=1}^{s}\tau_i)(N_0 - \Delta) \\ \underline{FP_{s+1}} &= (\prod_{i=1}^{s}\tau_i)(N_0 - \Delta) \cdot \tau_{s+1} \\ \underline{FN_{s+1}} &= \left(\sum_{i=1}^{s}(\prod_{j=1}^{i}(1-r_j))\Delta\right) + (1-r_{s+1})\Delta \end{aligned}$$

Donc par récurrence, $\mathcal{H}_s$ est vraie pour tout entier $s > 1$.

Nous avons donc une expression pour le taux de faux positifs, le rappel et le coût pour toute étape $s > 1$:

$$\begin{aligned}
\underline{\tau}_s &= \frac{\underline{VP_s}}{\underline{FP_s} + \underline{VN_s}} = \frac{1}{1 + \sum_{i=1}^{s} \frac{(1-\tau_i)}{\prod_{j=i}^{s} \tau_j}} \\
\text{Rappel}_s &= \frac{\underline{VP_s}}{\underline{VP_s} + \underline{FN_s}} = \frac{1}{1 + \sum_{i=1}^{s} \frac{\prod_{j=1}^{i}(1-r_j)}{\prod_{j=1}^{s} r_i}} \\
\text{Coût}_s &= N_0.k_0 + \sum_{i=0}^{s-1}(k_i \Delta(\prod_{j=1}^{i} r_i) + (N_0 - \Delta)(\prod_{j=1}^{i} \tau_j))
\end{aligned}$$

Petit point désagréable : la dépendance en Δ (nombre d'attributs réellement pertinents) ne disparaît pas de l'expression du coût final. Par contre, les autres expressions sont relativement simples, quoique difficiles à optimiser. Ces formules sont absolument génériques et ne font pas d'hypothèses sur A. Dans la pratique, on devra choisir les différentes valeurs de k_i (nombre d'exemples demandés à l'étape i) et de τ_i (ou de r_i) ; le lien entre τ_i et r_i étant donné par la courbe ROC de A correspondant au nombre d'exemples et d'attributs à l'étape i.

5.2.2 Optimisation

Connaissant les courbes ROC d'un algorithme de sélection en fonction du nombre d'exemples et d'attributs, pour un coût maximum fixé, il est possible d'essayer d'optimiser les choix du nombre d'exemples que l'on demande à chaque étape pour obtenir la meilleure courbe ROC possible. Notons toutefois que les courbes ROC présentées ci-après sont optimisées point par point. C'est à dire que l'on se fixe un taux de faux positif visé et on optimise le rappel sous la contrainte du Coût borné. On peut tout de même savoir quelle sera l'allure de la courbe ROC sur laquelle on effectue le choix, mais la sélection finale des attributs n'est disponible que pour une seule valeur du taux de faux positifs. Cela provient du fait que tous les attributs ne sont pas classés les uns par rapport aux autres (en particulier les attributs éliminés dans les premières étapes). *A contrario,* dans une utilisation classique en une seule étape, il est possible de déplacer le long de la courbe ROC. En effet, l'algorithme classant tous les attributs, cela correspond à déplacer le seuil.

Il est important de comprendre qu'il n'est pas possible de pousser plus loin l'optimisation si l'on ne fait pas une hypothèse sur le lien entre τ et r. Dans la suite, ce lien est donné par des courbes ROCs dépendant des paramètres Δ, nombre d'exemples disponibles, nombre d'attributs à l'étape en cours. Les courbes sont obtenues sur des données artificielles générées de la même manière que les données de test.

Une fois déterminée la courbe ROC de l'algorithme en fonction des paramètres précédents (comme pour les figures 5.4 et 5.3 où est représentée l'évolution de la courbe ROC en sélection en fonction du nombre d'exemples fournis). On optimise les valeurs des k_i et des τ_i. Ici on s'est limité à une sélection en trois étapes et l'optimisation a été effectuée par la force brute. Pour avoir davantage d'étapes, il faudrait par exemple avoir recours à des techniques d'optimisation par algorithmes génétiques par exemples.

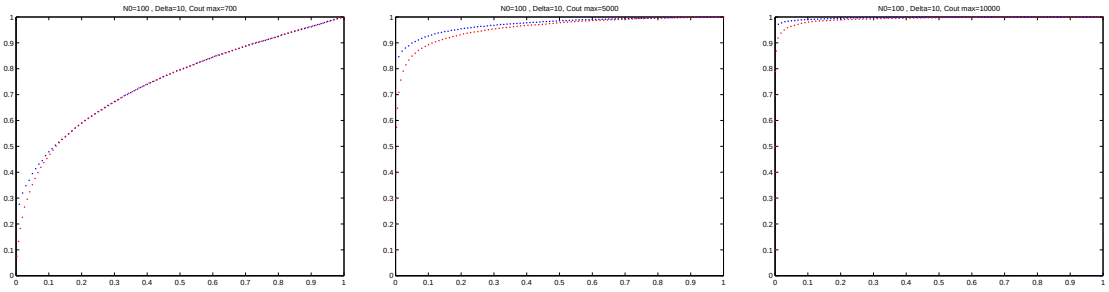


FIG. 5.5 : *Optimisation pour les courbes ROC de RELIEF. En abscisse : taux de faux positifs. En ordonnée : Rappel. En rouge : courbe ROC obtenue brutalement (tous les exemples en une seule étape). En bleu : Courbe ROC optimisée pour une demande des exemples en trois étapes. Le coût maximum autorisé est successivement de* $700, 5000$ *et* 10000*.*

On peut tirer quelques enseignements de ces expérimentations :

- Si nos ressources sont trop faibles, peu de gains sont à attendre... En effet, si le coût maximal autorisé est petit, on ne tire pas de bénéfice a éliminer progressivement des attributs car l'on n'est pas suffisament sûr que les premiers attributs éliminés ne sont pas pertinents.
- Si nous ressources sont très élevées, encore une fois les gains sont faibles... Intuitivement cela provient du fait que si l'on peut payer pour effectuer beaucoup de mesures, autant être le plus sûr possible de notre classification et ne négliger aucun attribut.
- Entre les deux, il existe une plage où l'apprentissage actif tire son épingle du jeu. Ceci est d'autant plus vrai si tous les attributs n'ont pas tous le même niveau de pertinence et que la présence de bruit dans les données est importante.
- Plus la courbe ROC est applatie au voisinage de 1, plus les gains possibles sont importants. C'est logique puisque nous éliminons progressivement des attributs, on a donc tendance à utiliser l'extrémité de la courbe.

On en conclut qu'en général il existe un intervalle de coût où il est intéressant d'avoir une approche active. Cet intervalle peut se déterminer par l'observation des courbes ROC, et il serait intéressant d'essayer de l'évaluer en fonction de la rapidité de l'évolution des courbes ROC vers une courbe idéale (en fonction du nombre d'exemples). De même il faudrait essayer d'évaluer le gain (augmentation d'AUC par exemple) en fonction de la valeur tangente en 1 de la courbe.

A noter qu'en plus de se focaliser sur une élimination des attributs les moins pertinents, on pourrait conjuguer à cela la sélection immédiate des attributs semblant les plus pertinents (ce qui correspond à travailler avec le début de la courbe ROC plutôt qu'avec

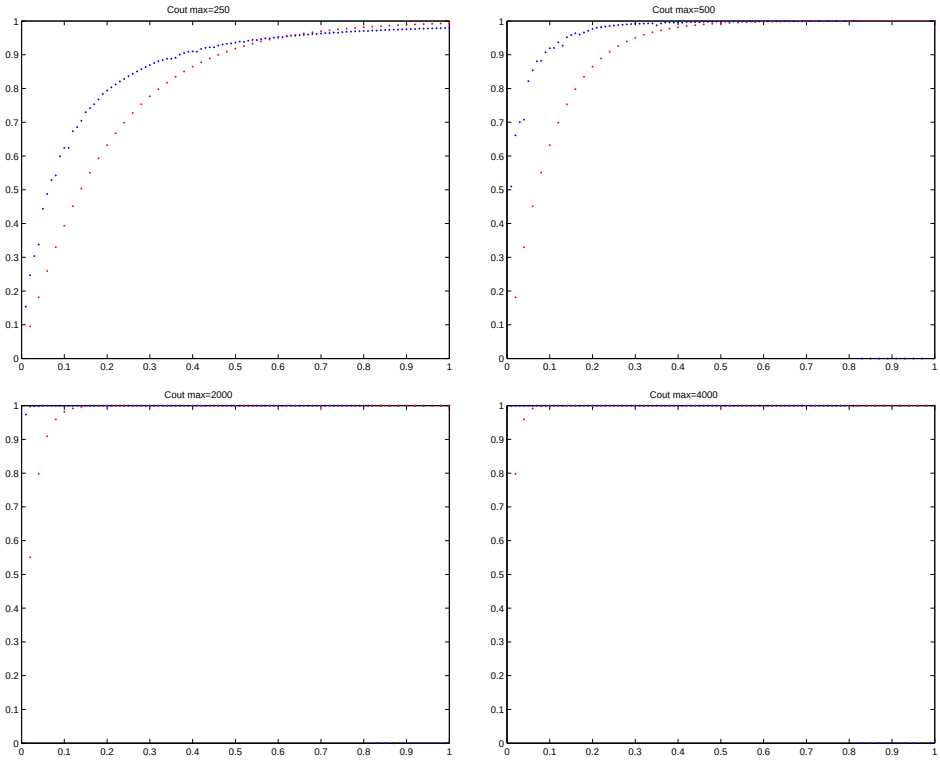


FIG. 5.6 : *Optimisation pour les courbes ROC artificielles. en abscisse : taux de faux positifs. en ordonnée : Rappel. En rouge : courbe ROC obtenue brutalement (tous les exemples en une seule étape). En bleu : Courbe ROC optimisée pour une demande des exemples en trois étapes. Le coût maximum autorisé est successivement de* $250, 500$ *et* 4000.

la fin). Le principe reste le même sauf que VP échange son rôle avec VN et FP avec FN. On retrouve une différence semblable à celle entre les approches *forward*, *backward* et *top-down*.

Dans tous les cas si on est capable d'effectuer l'optimisation, notre intérêt est de le faire en tenant compte des deux extémités de la courbe. Sinon, on a intérêt a se concentrer sur l'extrémité de la courbe ROC qui se comporte le mieux (proche tangente horizontale en 1 pour l'élimination des non-pertinents, et proche d'une tangente verticale en 0 pour la sélection immédiate). Dans tous les cas si ces deux valeurs sont mauvaises, une approche active est à procrire : les incertitudes sur le comportement de l'algorithme sur les données réelles (qui ne seront forcément pas les mêmes que celles ayant servi a estimer les courbes ROC) risquent de réduire à néant les gains (voire même de diminuer les performances).

Chapitre 6

Ensemble Feature Ranking

6.1 État de l'art

Dans la suite nous traiterons uniquement de concepts binaires et utiliserons les notations suivantes :

- L'ensemble d'apprentissage $\mathcal{E}$ contient n exemples ;
- $\mathcal{E} = \big\{(x_i, y_i),\ x_i \in \mathbb{R}^d,\ y_i \in \{-1, 1\},\ i = 1 \dots n\big\}$.
- Le i-ème exemple est décrit par les valeurs de ses d attributs ;
- L'étiquette (ou label) y_i de chaque exemple indique si l'exemple appartient à la classe du concept cible (exemple positif) ou non (exemple négatif)

6.1.1 Classement univarié d'attributs

Dans les approches univariées, on attribue à chaque attribut un score indépendamment des autres attributs. C'est la manière la plus simple de procéder. Elle est prise en défaut sur de nombreux points, en particulier elle ne peut pas détecter la redondance des attributs ou des concepts disjonctifs.

Le score est calculé à partir d'un test statistique, quantifiant la qualité d'un attribut pour la séparation des exemples positifs et négatifs. Par exemple le test de Mann-Whitney, réputé pour être capable de classer les attributs selon un degré de pertinence. On associe au k-ième attribut le score $MW(k)$ défini selon [Pepe et al.; 2003] par :

$$MW(k) = Max\ \{\ Pr(x_{i,k} > x_{j,k} \mid y_i > y_j),\ Pr(x_{i,k} < x_{j,k} \mid y_i > y_j)\ \}$$

Ou $P(x_{i,k} > x_{j,k} \mid y_i > y_j)$ est la fraction des paires d'exemples (positif,negatif) telles que l'attribut k classe l'exemple positif plus haut que le négatif. Ce critère est équivalent au test de la somme des rangs de Wilcoxon, qui est lui même équivalent au critère AUC (aire sous la courbe ROC [Yan et al.; 2003]).

6.1.2 Classement univarié d'attributs associé à une orthogonalisation de Gram Schmidt

Dans [Hervé et al.; 2003], il est proposé une extension de l'approche univariée, basée sur un procédé itératif. Le score associé à chaque attribut est proportionnel à son cosinus avec le concept cible :

$$score(k) = \frac{\sum_{i=1}^{n} x_{i,k}.y_i}{\sqrt{\sum_{i=1}^{n} x_{i,k}^2}}$$

En deux étapes, le processus consiste à :

1. Déterminer l'attribut courant maximisant le score précédent ;
2. Projeter tous les attributs restants et le concept cible sur l'hyperplan perpendiculaire à l'attribut k.

Le critère d'arrêt est basé sur une étude analytique d'une variable aléatoire définie comme le cosinus du concept cible avec un attribut aléatoire uniforme.

Cette approche résout les problèmes liés à la présence de redondance dans les données mais continue à souffrir de la myopie caractéristique des approches gloutonnes (sans backtrack).

6.1.3 Approches basées sur l'apprentissage

Plutôt que de s'intéresser un à un aux attributs, il est possible de travailler conjointement avec un algorithme d'apprentissage (souvent appelées approches embarquées). Ce qui peut permettre de tenir compte des relations entre les attributs [John et al.; 1994].

Quand on apprend une hypothèse linéaire ($h(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i \ [+b]$), on associe un score à chaque attribut k qui est la valeur absolue de w_k. Plus le score est élevé, plus l'attribut est pertinent, **en combinaison avec les autres attributs**.

Une approche itérative en deux pas appellée SVM-Recursive Feature Elimination est proposée par [Guyon et al.; 2002]. Dans chaque pas on procède ainsi :

1. On apprend un SVM linéaire ;
2. Les attributs sont classés par poids décroissants et les plus mauvais sont éliminés.

Une autre approche, basée sur la régression linéaire [Bi et al.; 2003] utilise une approche aléatoire pour essayer d'accroitre la robustesse. Plus précisément, un ensemble d'hypothèses linéaires sont calculées à partir de sous ensembles indépendant des données d'apprentissage. Le score de k-ème attribut est la moyenne des poids de cet attribut parmi toutes les hypothèses apprises grâce à ces sous-ensembles. Cependant, cette approche n'est possible que si l'on dispose de suffisament de données ; en effet les sous ensembles doivent être beaucoup plus petits que l'ensemble de tous les exemples afin

d'assurer la diversité des hypothèses linéaires (ce qui exclut certain champs d'application comme l'analyse des puces à ADN).

Un autre travail moins connu, a été fait dans [Guerra-Salcedo and Whitley; 1999] sur l'apprentissage d'un ensemble d'hypothèses par un algorithme génétique lors de runs indépendants. L'algorithme recherche un sous ensemble d'attributs (individu génétique) qui est évalué par le résultat d'un algorithme de classification (k plus proches voisins dans l'article) utilisant ces attributs.

6.2 *Ensemble Feature Ranking*

L'idée est ici d'adapter ROGER (voir 10) à la sélection d'attributs. En fait plutôt que de faire de la sélection, il est souvent pertinent de commencer par établir un ordre sur les attrituts (*Feature Ranking*). Comme ROGER est un algorithme évolutionnaire, plusieurs ordres différents sur les attributs vont être découverts, nous allons essayer d'en déduire un plus satisfaisant par un système de votes (comme dans le *bagging*).

6.2.1 Notations

En s'inspirant des techniques d'*ensemble learning* (combinaisons de différents modèles, [Breiman; 2003]) et des algorithmes aléatoires [Esposito and Saitta; 2003], nous voulons combiner des ordres indépendants sur les attributs en un ordre que nous espérons plus robuste. Nous noterons O_t un ordre sur les attributs (il s'agit simplement d'une permutation de $\{1 \dots d\}$). Sans perte de généralité, nous supposerons que les attributs sont donnés par pertinence décroissante (i.e. l'attribut i est moins pertinent que l'attribut j ssi $i < j$).

Supposons $O_1, \dots O_T$, T ordres d'attributs indépendants et identiquement distribués. Pour chaque paire (i, j) d'attributs, notons $N_{i,j}$ le nombre d'ordres O_t qui classent l'attribut i avant l'attribut j. On pose également $Y_{i,j} = Vrai$ si et seulement si $N_{i,j} > \frac{T}{2}$. Nous allons commencer par voir que l'on peut construire un ordre à partir des variables $Y_{i,j}$, puis nous étudierons sa qualité.

6.2.2 Consistent Ensemble Feature Ranking

Pour obtenir un ordre à partir des $Y_{i,j}$, il faut que ceux ci définissent une relation transitive. C'est à dire que $Y_{i,k}$ est vrai si $Y_{i,j}$ et $Y_{j,k}$ sont vrais ; si ceci est vrai pour tout i, j, k ont dira alors que les ordres $O_1, \dots, O_T$ sont **consistants**.

On travaille à partir de l'ordre idéal des attributs (Ceux-ci étant ordonnés par ordre de pertinence, l'ordre que nous souhaitons retrouver est $1, 2, \dots, T$). Les perturbations de

cet ordre seront modélisées par des variables booléennes $X_{i,j}$. On définit $X_{ij}(O_t)$ comme étant vrai si et seulement si O_t échange les positions relatives des attributs i and j (i.e. $(O_t(i) < O_t(j)) \neq (i < j)$).

Comme dans [Rosset; 2004], pour l'étude théorique, nous supposerons que les variables $X_{i,j}$ sont des variables indépendantes suivant une loi de Bernouilli de même paramètre p. Cette supposition est bien sûr réductrice. La supposition d'indépendance est, comme souvent en statistiques, fausse. Cependant elle permet d'obtenir des résultats analytiques sur la qualité de l'ordre obtenu par EFR. En effet, en étant plus rigoureux dans la modélisation des permutations, on se heurte rapidement à d'importantes difficultés mathématiques.

Proposition 6.1 *Soit p le taux d'erreur de l'ordre O_t, i.e. la probabilité pour O_t d'échanger les positions respectives d'un couple (i, j) d'attributs ($p = Pr((O_t(i) < O_t(j)) \neq (i < j))$).*
Si $p = \frac{1}{2} - \varepsilon,\ \varepsilon > 0$, alors $O_1, \ldots, O_T$ sont presque surement consistants quand T tend vers l'infini.

Démonstration :
Soit i, j, k trois attributs distincts tels que Y_{ij} et Y_{jk} soient vrais.
Y_{ij} compare la loi de Bernoulli $\frac{1}{T}\mathcal{B}(T, p)$ (moyenne de T réalisations de la variable booléenne X_{ij} suivant une Bernouilli de paramètre p), au seuil de la majorité $\frac{1}{2}$.
D'où : $Pr(i < j|Y_{ij}) = Pr(\frac{1}{T}\mathcal{B}(T, p) < \frac{1}{2})$. Posons $\eta = Pr(\frac{1}{T}\mathcal{B}(T, p) < \frac{1}{2})$.
Puisque nous avons supposé que les X_{ij} étaient indenpendants identiquement distribués, $Pr(i < k|Y_{ij} \wedge Y_{jk}) = \eta^2$.
Ce qui nous donne $Pr(\neg Y_{ik}|i < k) = Pr(\frac{1}{T}\mathcal{B}(T, p) > \frac{1}{2}) = 1 - \eta$.
Finalement :

$$\begin{array}{rcl} Pr(\neg Y_{ik}|Y_{ij} \wedge Y_{jk}) & \leq & Pr(i > k|Y_{ij} \wedge Y_{jk})) + Pr(\neg Y_{ik}|i < k) \times Pr(i < k|Y_{ij} \wedge Y_{jk}) \\ & = & (1 - \eta^2) + (1 - \eta) \times \eta^2 = 1 - \eta^3 \end{array}$$

En utilisant l'inégalité classique $Pr(|\frac{1}{T}\mathcal{B}(T, p) - p| \geq \varepsilon) \leq \frac{1}{4T\varepsilon^2}$, on obtient la convergence de η vers 1 quand T tend vers l'infini, ce qui prouve la convergence presque sûre.

■

6.2.3 Convergence

Si l'on suppose la cohérence des ordres d'attributs O_t, le classement final O^* se définit naturellement en comptant pour chaque attribut i le nombre d'attributs j qui sont classés avant i pour plus de la moitié des O_t ($O^*(i) = Card\{Y_{ji}$ true, $j = 1..d\}$).

La convergence du vote est étudiée par rapport à la probabilité de mal classer un attribut i d'au plus τ rangs ($P(|O^*(i) - i|) \geq \tau$). Ici encore, pour pouvoir traiter les résultats analytiquement, nous supposerons que cette probabilité d'erreur ne dépend pas

du véritable rang de l'attribut i. Encore une fois, cette supposition est fausse en pratique puisque la probabilité de se tromper légèrement sur le rang des meilleurs ou des moins bons attributs est biaisée par rapport aux autres attributs. Cependant, cette étude se focalise davantage sur la probabilité de se tromper de beaucoup sur l'ordre des attributs (*e.g* rejeter un attribut faisant partie du top 10 alors que l'on conserve 50% des attributs).

Proposition 6.2 *Soit p^* la probabilité pour le vote d'échanger deux attributs, $p^* = P(Y_{i,j} \neq (i < j))$ et soit $\tau = (d-1)(p^* + \varepsilon)$, $\varepsilon > 0$. Alors :*

$$Pr(|O^*(i) - i| \geq \tau) \leq \frac{1}{4(d-1)\varepsilon^2}$$

Démonstration :
L'attribut i est mal classé d'au plus τ indices, s'il existe au moins τ attributs j dans les $d-1$ attributs restants tels que $Y_{i,j} \neq (i < j)$.
De plus $Pr(|O^*(i) - i| \geq \tau) < P(\frac{1}{d-1}\mathcal{B}(d-1, p^*) \geq \frac{\tau}{d-1})$.
En utilisant la même inégalité que dans la proposition précédente, on obtient :

$$P(\frac{\mathcal{B}(d-1,p^*)}{d-1} \geq \frac{\tau}{d-1}) \leq P(|\frac{\mathcal{B}(d-1,p^*)}{d-1} - p^*| \geq \varepsilon) \leq \frac{1}{4(d-1)\varepsilon^2}$$

ce qui prouve le résultat.

■

L'origine de la bonne convergence des votes de classements d'attributs provient du fait que la probabilité p^* d'erreur du classement final diminue avec le nombre de votants T, en amplifiant **exponentiellement** l'avantage d'utiliser un expert par rapport à un classement aléatoire. De plus la distribution des erreurs de classement est centrée sur $p^* \times (d-1)$ (ou d est le nombre total d'attributs).

6.3 Validation Statistique du modèle

Commençons par remarquer que les performances d'un système de sélection d'attributs est souvent calculée par l'intermédiaire d'un algorithme d'apprentissage placé en sortie de la sélection. En fait il est difficile d'évaluer la qualité d'un système de sélection d'attributs.

Pour contourner cet obstacle, nous mettons en place un modèle statistique permettant l'évaluation directe de classement des attributs. Ce modèle s'inspire de l'analyse de complexité statistique développée dans la communauté de la satisfaction de contraintes [Hogg et al.; 1996]. Ces notions ont étées rapportées en Machine Learning par Giordana and Saitta [Giordana and Saitta; 2000].

6.3.1 Principe

Dans le paradigme de l'analyse statistique, l'espace du problème est défini par un ensemble de paramètres d'ordre (par exemple, contrainte de densité et de précision [Hogg et al.; 1996]). La performance d'un algorithme donné est vue comme une variable aléatoire, observée dans l'espace du problème. A chaque point l'espace du problème (valeurs des paramètres d'ordre), on associe le comportement moyen de l'algorithme sur toutes les instances du problème avec les mêmes valeurs des paramètres d'ordre.

Ce paradigme a prouvé son utilité dans l'étude du passage à l'échelle de célèbres algorithmes d'apprentissage et dans la détection de zones «d'échec» inattendues où les performances chuttent soudainement au niveau d'une décision aléatoire.

6.3.2 Paramètres d'ordre

Pour la sélection d'attributs, nous définissons sept paramètres d'ordre :
- Le nombre d'exemples n ;
- Le nombre total d'attributs d ;
- Le nombre d'attributs pertinents r. Un attribut est appellé pertinent si et seulement s'il intervient dans la définition du concept cible.
- Le type de concept l, linéaire ($l = 1$) ou non linéaire ($l = 2$), avec :

$$l = 1 : \quad y(\mathbf{x}) = 1 \quad \text{ssi } (\sum_{i=1}^{r} x_i > s) \tag{6.1}$$

$$l = 2 : \quad y(\mathbf{x}) = 1 \quad \text{ssi } (\sum_{i=1}^{r} (x_i - .5)^2 < s) \tag{6.2}$$

- La redondance (k booléen) des attributs pertinents. En pratique, la redondance est obtenue en remplacant r attributs non pertinents par une combinaison linéaire choisie aléatoirement des r attributs pertinents.
- Le taux de bruit e dans les étiquettes des classes (flip du label avec une probabilté e).
- Le taux de bruit σ dans les valeurs des attributs : chaque valeur est perturbée par l'ajout d'un bruit gaussien tiré selon une $\mathcal{N}(0, \sigma)$.

6.3.3 Génération de problèmes artificiels

Pour chaque point $(n, d, r, l, k, e, \sigma)$ dans l'espace du problème nous générons des instances indépendantes du problème d'apprentissage.

Les d attributs des n exemples sont tirés uniformément dans $[0; 1]$. L'étiquette de chaque exemple est calculée selon l'équation (6.1) pour $l = 1$ (resp (6.2) pour $l = 2$).

Le seuil s auquel se réfère le concept cible est mis à $r/2$ dans l'équation (6.1) (resp à $r/12$ pour (6.2)), afin de garantir une répartition équilibrée des exemples positifs et négatifs. On ne tiendra pas compte ici des difficultées supplémentaires liées aux distributions biaisées des exemples.

Dans le cas où on s'intéresse à la redondance ($k = 1$), on remplace r attributs non pertinents par une combinaison linéaire choisie aléatoirement des r attributs pertinents. Enfin on insère le bruit à la fois dans les étiquettes et sur les attributs. Les labels calculés sont « flippés » avec une probabilité e et on ajoute un bruit gaussien de variance σ^2.

Ce générateur de problèmes diffère de celui de [Guyon and Elisseeff; 2003] sous plusieurs aspects. Ainsi [Guyon and Elisseeff; 2003] ne s'intéresse qu'aux concepts cibles linéaires (définis comme étant une combinaison linéaire des attributs pertinents), ce qui fait que le concept cible ne dépend pas de la même manière de tous les attributs. Dans notre modèle tous les attributs utiles ont la même pertinence. De plus, nous traitons une portions des problèmes non-linéaires.

6.3.4 Format des résultats

Le classement des attributs est fait en se basant sur des courbes ROC. A chaque index $i \in \{1 \dots d\}$, on associe la portion des attributs réellement pertinents (respectivement la portion des attributs non-pertinents ou faussement perninents) qui ont un rang plus haut que i, noté $TR(i)$ (resp $FR(i)$). On appellera $\{(FR(i), TR(i)), i = 1, \dots, d\}$ la courbe ROC-FS associée à $\mathcal{O}$.

La courbe ROC-FS permet de visualiser le compromis effectué par l'algorithme entre les deux objectifs : donner de hauts rangs (resp de bas rangs) aux attributs pertinents (resp non-pertinents). La courbe ROC-FS associée à un classement parfait (classant tous les attributs pertinents avant les non pertinents), atteint l'optimum global $(0, 1)$ (pas d'attribut non pertinent de sélectionné, $FR = 0$ tandis que tous les attributs pertinents ont étés conservés $TR = 1$).

En regardant les courbes ROC-FS, on peut facilement évaluer si un algorithme de classement d'attributs est plus efficace qu'un autre. De plus la courbe donne une image précise des performance de l'algorithme. En effet, le début de la courbe montre la pertinence des attributs les mieux classés (suggérant éventuellement une approche itérative comme dans [Hervé et al.; 2003]) ; tandis que la fin montre à quel point les attributs les moins bien classés sont inutiles (suggérant une approche par élimination successives comme dans [Guyon et al.; 2002]). On peut même imaginer à partir d'une cartographie des courbes ROC-FS en fonction des paramètres d'ordre d'un problème essayer d'optimiser une série de choix de seuils.

En bref, nous définissons trois indicateurs de performance pour un algorithme de classification d'attributs.

– Le premier, noté p_b mesure la probabilité pour l'attribut le mieux classé d'être pertinent, reflétant ainsi les qualités de l'algorithme dans une utilisation forward.

– Le second noté p_w est le plus mauvais rang d'un attribut pertinent divisé par d, reflétant ainsi le comportement de l'algorithme dans le cadre d'une utilisation backward.
– Enfin, nous regarderons l'aire sous la courbe ROC-FS (AUC), qui peut-être vue comme un indicateur global de la performance de l'algorithme (la valeur maximale 1 est obtenue pour un classement parfait).

6.4 Analyse expérimentale

Nous nous intéresserons ici à la validation expérimentale de l'algorithme présenté dans la section 6.2. Les résultats sont comparés à l'état de l'art ([Hervé et al.; 2003]) en utilisant le critère cosinus. Les algorithmes sont comparés en utilisant à la fois les courbes ROC-FS et les mesures de performance précédemment définies dans la section 6.3.4.

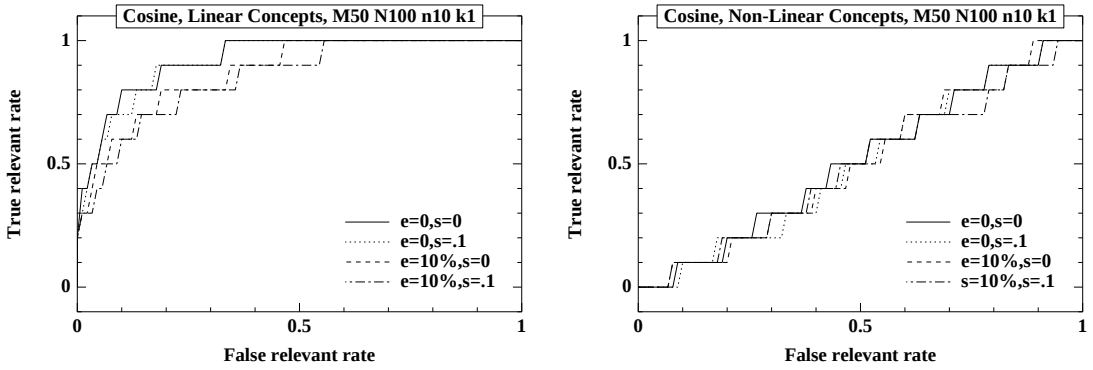
6.4.1 Protocole expérimental

En suivant les principes énoncés dans la section 6.3, nous proposons une étude expérimentale. Le nombre d d'attributs est successivement choisi à $100, 200$ et 500. Le nombre r d'attributs pertinents est $d/20$, $d/10$ et $d/5$. Le nombre n d'exemples est pris à $d/2$, d et $2d$. On regarde à la fois des concepts linéaires et non-linéaires avec et sans attributs redondants. Enfin, le bruit e est réglé à $0, 5$ et 10% et la variance du bruit gaussien est de $0, 0.05$ et 0.10.

Au total on travaille avec 972 points $(d, r, m, l, k, e, \sigma)$ dans l'espace du problème. pour chaque point, 20 jeux de données sont générés indépendamment. Pour chaque jeu de données, 15 runs indépendants de ROGER sont exécutés pour construire un ensemble feature ranking $\mathcal{O}$ (EFR). Les indicateurs associés p_b, p_w et AUC sont calculés et leurs médianes sur les jeux de données ayant les mêmes paramètres d'ordre sont reportées.

Les résultats de référence sont obtenus de manière similaire pour le critère du cosinus ([Hervé et al.; 2003]) : pour chaque point de l'espace du problème, 30 jeux de données sont générés, le classement des attributs basé sur le cosinus est calculé à partir des indicateurs p_b, p_w et AUC puis on calcule la médiane.

Les algorithmes sont codés en C++, exécutés sur des Pentium-IV. ROGER est paramétré pour utiliser un (20+200)-ES avec mutation auto-adaptative et cross over uniforme de 0.6. L'initialisation est faite uniformément dans $[0; 1]$ et un seuil d'au plus 50000 évaluations de fitness est fixé.



(a) concepts linéaires (b) concepts Non-linéaires

FIG. 6.1 : *Critère du cosinus : Mediane des courbes ROC-FS sur 30 training datasets sur des concepts linéaires et Non-Linéaires, avec d = 100, $n = d/2$, $r = d/10$. Pas de redondance.*

n	d	r	e	σ	p_b	p_w	AUC	p_b	p_w	AUC
50	100	10	0	0	0.87	.33	0.920	0.97	.10	0.97
50	100	10	0	0.1	0.9	.33	0.916	0.9	.10	0.97
50	100	10	10%	0	0.87	.47	0.87	0.77	.16	0.95
50	100	10	10%	0.1	0.8	.56	0.848	0.83	.17	0.95
100	100	10	0	0	1	.18	0.97	1	.10	0.99
100	100	10	0	0.1	1	.22	0.966	1	.10	0.99
100	100	10	10%	0	0.93	.29	0.944	0.97	.10	0.98
100	100	10	10%	0.1	0.93	.36	0.934	0.97	.10	0.97

Pas de redondance Redondance

Concepts linéaires

n	d	r	e	σ	p_b	p_w	AUC	p_b	p_w	AUC
50	100	10	0	0	0.03	.93	0.49	0.17	.42	0.74
50	100	10	0	0.1	0.03	.94	0.49	0.1	.45	0.74
50	100	10	10%	0	0.1	.93	0.49	0.1	.44	0.73
50	100	10	10%	0.1	0.03	.93	0.51	0.1	.44	0.75
100	100	10	0	0	0	.91	0.53	0.23	.42	0.76
100	100	10	0	0.1	0.03	.90	0.52	0.17	.41	0.75
100	100	10	10%	0	0.17	.92	0.52	0.27	.47	0.76
100	100	10	10%	0.1	0.1	.92	0.52	0.3	.46	0.74

Pas de redondance Redondance

Concepts non-linéaires

TAB. 6.1 : *Critère du cosinus : Probabilité p_b de classer premier un attribut pertinent, Mediane du rang relatif p_w de l'attribut pertinent le plus mal classé. Aire sous la courbe ROC-FS.*

6.4.2 Résultats de référence

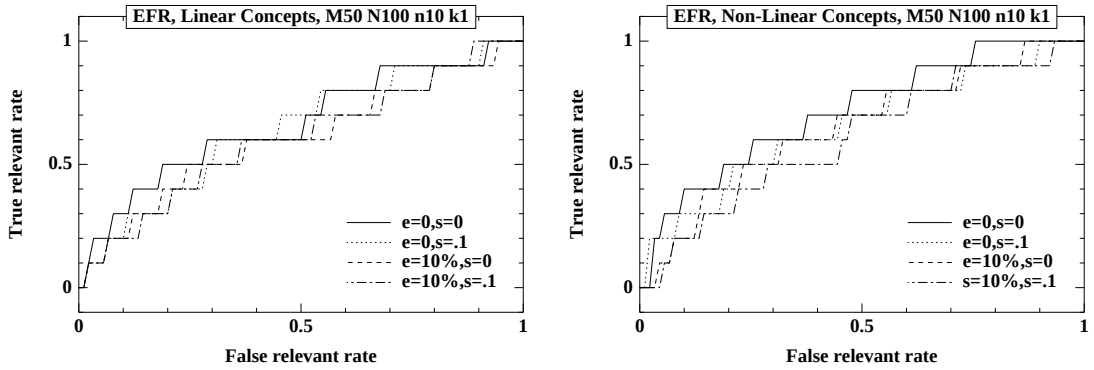
Les performances du critère du cosinus sur des concepts linéaires et non-linéaires sont illustrées sur la figure 6.1, ou le nombre d d'attributs est 100, le nombre n d'exemples 50 et le nombre d'attributs pertinents r 10. Les indicateurs de performance sont résumés dans le tableau 6.1. Des résultats similaires sont obtenus pour de plus grandes valeurs de d.

D'excellents résultats sont obtenus pour des concepts linéaires. Avec deux fois plus d'attributs que d'exemples, la probabilité p_b de classer premier un attribut pertinent est d'environ 90%. On observe une lente dégradation de p_b avec l'adjonction de bruits, la sensibilité au bruit sur les labels étant plus marquée que sur les attributs. Les attributs pertinents sont dans les top p_w attributs avec p_w variant grosso-modo de $1/3$ à $1/2$.

De plus les performances s'améliorent avec l'augmentation du nombre d'exemples : p_b atteint 100% et p_w passe entre $1/5$ et $1/3$ pour $n = d$.

Par contre, sur des concepts non linéaires, le critère du cosinus ne fait pas mieux que le hasard. Ceci peut s'observer sur les courbes ROC-FS qui sont proches de la diagonale, et la situation ne s'améliore pas en doublant le nombre d'exemples. Notons que les performances sont illusoirement meilleures en introduisant de la redondance car l'extraction des attrituts pertinents se fait parmi $2r$ attributs.

6.4.3 Classement évolutionnaire d'attributs



(a) Concept linéaire (b) Concept non-linéaire

FIG. 6.2 : *EFR performance : Médianne des courbes ROC-FS sur 20 training sets sur des concepts linéaires et non-linéaires, avec d = 100, $n = d/2$, $r = d/10$, Pas d'attributs redondants.*

L'évaluation se fait selon le même protocole. L'EFR est clairement moins performante que le critère du cosinus dans le cas linéaire. Avec deux fois plus d'attributs que d'exemples, la probabilité p_b oscille entre 0.3 et 0.5 alors qu'elle était entre 0.8 et 0.9 pour les résultats de référence. Quand le nombre d'exemples augmente, p_b augmente aussi mais n'atteint que 0.55 à 0.85 (contre 0.93 à 1).

n	d	r	e	σ	p_b	p_w	AUC	p_b	p_w	AUC
50	100	10	0	0	0.5	.92	0.67	0.65	.29	0.86
50	100	10	0	0.1	0.5	.80	0.63	0.75	.30	0.85
50	100	10	10%	0	0.35	.94	0.61	0.45	.31	0.85
50	100	10	10%	0.1	0.35	.89	0.62	0.60	.40	0.82
100	100	10	0	0	0.85	.79	0.79	0.90	.23	0.92
100	100	10	0	0.1	0.50	.74	0.77	0.95	.21	0.92
100	100	10	10%	0	0.55	.77	0.72	0.65	.27	0.89
100	100	10	10%	0.1	0.65	.82	0.75	0.45	.28	0.88

Pas de redondance Redondance

Concepts linéaires

n	d	r	e	σ	p_b	p_w	AUC	p_b	p_w	AUC
50	100	10	0	0	0.20	.75	0.71	0.50	.26	0.88
50	100	10	0	0.1	0.45	.82	0.68	0.25	.33	0.84
50	100	10	10%	0	0.25	.81	0.68	0.30	.32	0.83
50	100	10	10%	0.1	0.25	.88	0.61	0.35	.28	0.83
100	100	10	0	0	0.55	.63	0.81	0.80	.20	0.92
100	100	10	0	0.1	0.60	.72	0.78	0.50	.22	0.90
100	100	10	10%	0	0.65	.78	0.77	0.50	.20	0.91
100	100	10	10%	0.1	0.40	.72	0.75	0.40	.26	0.87

Pas de redondance Redondance

Concepts non-Linéaires

TAB. 6.2 : *Ensemble Feature Ranking avec ROGER : Probabilité p_b de classer premier un attribut pertinent, Mediane du rang relatif p_w de l'attribut pertinent le plus mal classé, Aire sous la courbe ROC-FS.*

Par contre, dans le cas non-linéaire, l'EFR devient plus performante que le cosinus. p_b est autour de 0.3 (contre 0.03 à 0.1 dans les résultats de référence). De plus en doublant la valeur de n, p_b s'élève à 0.55

Au niveau de la complexité des calculs, le critère du cosinus est linéaire par rapport au nombre d'exemples et en $d\log(d)$ par rapport au nombre d'attributs. Le temps d'exécution dans l'expérience est négligeable. De même, pour l'EFR, la complexité est linéaire en le nombre d'exemples. Par contre, par rapport à d le nombre d'attributs, l'évaluation de la complexité est difficile car la taille de l'espace de recherche de ROGER augmente ($[0;1]^{2d}$). Expérimentalement, pour $n = 50, d = 100$ le coût total est de 6 minutes, 12 minutes pour $d = 100$ et 23 minutes pour $d = 500$.

Les résultats ne sont donc pas encore compétitifs dans le cas linéaire, cependant les résultats sont prometteurs dans le cadre de problèmes non-linéaires. Il faut maintenant utiliser des données provenant du monde réel pour évaluer l'intérêt pratique de l'EFR. De plus l'utilisation de classements obtenus selon des runs indépendant de ROGER, permet de penser qu'il est possible de s'intéresser à des problèmes plus complexes d'apprentissage (tels que des concepts disjonctifs par exemple).

That’s All Folks

En conclusion, je voudrais revenir sur les leçons que j’ai tirées des travaux présentés dans cette thèse.

Le sujet central est celui des capacités actives de l’apprentissage, c’est à dire la capacité de choisir les exemples les plus utiles et/ou les descripteurs les plus pertinents pour l’apprentissage.

L’importance d’un tel choix est claire dans le cadre de l’apprentissage humain ; on pourrait même dire que l’essence de la pédagogie est de savoir choisir l’ordre dans lequel les exemples et les exercices vont être communiqués à l’apprenant, ainsi que la bonne description de ces exemples.

Notons que la capacité de choisir les exemples et/ou les descripteurs nécessaires pour apprendre peut être vue comme « meta », comparée à la capacité d’apprendre.
Spécifiquement, le fait de caractériser un exemple comme « difficile » ou « utile » est un objectif d’un autre niveau, que celui de lui associer une classe donnée. Plus précisément, la sélection des exemples d’apprentissage repose ainsi sur un diagnostic préliminaire, et sur la confiance en ce diagnostic.

Dans le cadre de l’apprentissage artificiel, la sélection des descripteurs comme la sélection des exemples d’apprentissage constituent des problèmes de pointe : de la résolution de ces problèmes dépend le passage à l’échelle de l’apprentissage, et sa faisabilité en terme d’effort de calcul ou d’effort de l’expert.

Une des premières motivations de l’apprentissage actif, i.e. de la sélection des exemples d’apprentissage, concerne en effet la réduction du coût d’étiquetage des exemples, et l’effort demandé à l’expert. Dans ce contexte, la réduction de l’effort d’étiquetage a pour contre-partie classique une réduction des performances d’apprentissage.

Le travail présenté ne fait pas exception : si de beaux résultats théoriques ont pu être obtenus dans le cadre de l’apprentissage actif, les validations expérimentales en ont honnêtement montré quelques limites, soulignant s’il en était besoin la différence entre le fait de minimiser une borne sur l’erreur en généralisation, et le fait de minimiser l’erreur en généralisation elle-même.

Au contraire, et comme on pouvait s’y attendre, les résultats obtenus dans le cadre de la sélection d’attributs ont montré un bien meilleur accord entre les avancées théoriques

et la validation expérimentale. Ceci s'explique par le fait que la réduction du nombre d'attributs influence directement la capacité de l'espace d'hypothèses et facilite ainsi l'apprentissage. Par opposition, la dépendance entre la facilité de l'apprentissage et la distribution des exemples est beaucoup plus difficile à étudier.

En résumé, les leçons essentielles tirées du travail accompli concernent :
- L'importance essentielle de la distribution des exemples. Par ailleurs des motivations applicatives majeures, allant de l'identification de concept dynamique à la robotique, remettent en cause l'hypothèse classique de l'apprentissage artificiel, concernant la distribution i.i.d. des exemples.
- La difficulté de l'étude théorique du rôle de la distribution des exemples pour l'apprentissage.

Dans la continuité de ce travail, mes perspectives de recherche à court et moyen terme concernent la jonction entre l'apprentissage actif et le boosting [Freund and Schapire; 1996][Rudin et al.; 2005]. Ces deux approches partagent en effet le fait d'agir dans l'espace des distributions des exemples. Le boosting en particulier, a été explicitement formulé comme un système dynamique, couplant l'espace des distributions sur l'espace des instances, et l'espace des hypothèses [Rudin et al.; 2004].

Les différences essentielles entre apprentissage actif et boosting peuvent être analysées de la façon suivante :
- Le boosting est un phénomène pourvu de mémoire ; une suite d'hypothèses est construite et utilisée, même s'il s'avère que cette suite pourrait être fortement élaguée sans réduire les performances [Dietterich; 2000][Derbeko et al.].
- L'apprentissage actif cherche à sur-échantillonner les régions dans lesquelles la variance de l'erreur est maximale ; par opposition, le boosting cherche à sur-échantillonner les régions dans lesquelles l'erreur est maximale.

Du côté du boosting, nous avons donc un critère instable d'échantillonnage (l'erreur) et l'instabilité du critère est compensée par la mémoire du processus ; du côté de l'apprentissage actif, le critère est plus stable (la variance de l'erreur), par contre le processus ne remet pas en cause l'hypothèse issue de la distribution fondée sur ce critère ; les éventuels biais initiaux ne sont jamais corrigés.

Il vient ainsi naturellement de combiner ces deux approches, en considérant une suite de distributions, obtenues par combinaison du critère d'erreur et de variance de l'erreur.

Idéalement, une telle approche devrait permettre de construire un système dynamique plus stable que le boosting, e.g. évitant les divergences dues aux exemples bruités [Ratsch; 2001], mais plus instable que l'apprentissage actif, et donc plus à même d'exploiter les capacités d'ensemble de la suite des hypothèses construites.

La clé du problème, l'ajustement du critère de mélange entre l'erreur et la variance de l'erreur, conduit à proposer une nouvelle formulation implicite pour l'apprentissage,

fondée sur la recherche d'une distribution idéale des exemples. Notons qu'une telle formulation rejoint directement le cadre de l'apprentissage par renforcement.

De façon plus générale en apprentissage PAC, je pense qu'il faudrait maintenant essayer de borner $|L(\text{argmin}\,\hat{L}) - \hat{L}(\text{argmin}\,\hat{L})|$ plutôt que $\sup_f |L - \hat{L}|$. Ainsi en ne bornant plus uniformément sur F tout entier, il est sans doute possible d'affaiblir encore les hypothèses de régularité et resserrer les bornes (qui sont encore trop lâches à mon goût). C'est dans ce sens que je fais partie du comité de programme d'un challenge PASCAL avec G. Gavin, S. Gelly, Y. Guermeur, S. Lallich, M. Sebag, et O. Teytaud.

Je terminerai cette conclusion en remerciant une fois encore tout ceux qui m'ont permis d'arriver ici. Faire une thèse c'était trop bien ! ☺

Enfin, pour ceux qui sont arrivés jusque là et à qui il reste un peu de courage, nous allons poursuivre avec quelques présentations sommaires d'autres travaux effectués pendant ma thèse mais n'entrant pas dans la ligne directrice principale ou purement techniques (développement de logiciels).

Quatrième partie

ANNEXES

Chapitre 7

Projections aléatoires

En apprentissage supervisé, dans le cas ou le concept cible vit dans un sous-espace étant une combinaison linéaire des attributs, SVD est une très bonne façon de retrouver le sous espace en question. On rentre ici dans le domaine de la *feature construction*. Il est amusant de remarquer que dans ce cas une simple projection aléatoire des données (*via* une matrice générée aléatoirement) dans un sous espace plus petit, conduit également a de bons résultats ([Achlioptas; 2001][Fradkin and Madigan; 2003]). Ce résultat est principalement utilisé en fouille de données pour éviter la malédiction de la dimension avec un coût computationnel faible (à la place du SVD utilisé par [Deerwester et al.; 1990] en fouille de textes). C'est assez étonnant, mais opérationnel…

Si une seule projection aléatoire peut déjà s'avérer satisfaisante, alors pourquoi ne pas en utiliser plusieurs (choisies indépendamment) et utiliser le principe du *Bagging* pour essayer d'augmenter les performances. En effet les projections étant indépendantes, on peut espérer augmenter le degré d'indépendance de chacun des apprenants et donc tirer avantage du *bagging*.

En pratique, sur des données réelles n'ayant aucune raison d'être facilement séparable par un hyperplan ou par la méthode des moindres carrés, on obtient des résultats intéressants. Il faut cependant faire attention aux éventuels problèmes de conditionnement qui peuvent apparaître (en particulier si l'algorithme d'apprentissage commence par renormaliser les données). Au final, on obtient des résultats meilleurs que par une SVD mais pour un coup computationnel (beaucoup) plus élevé.

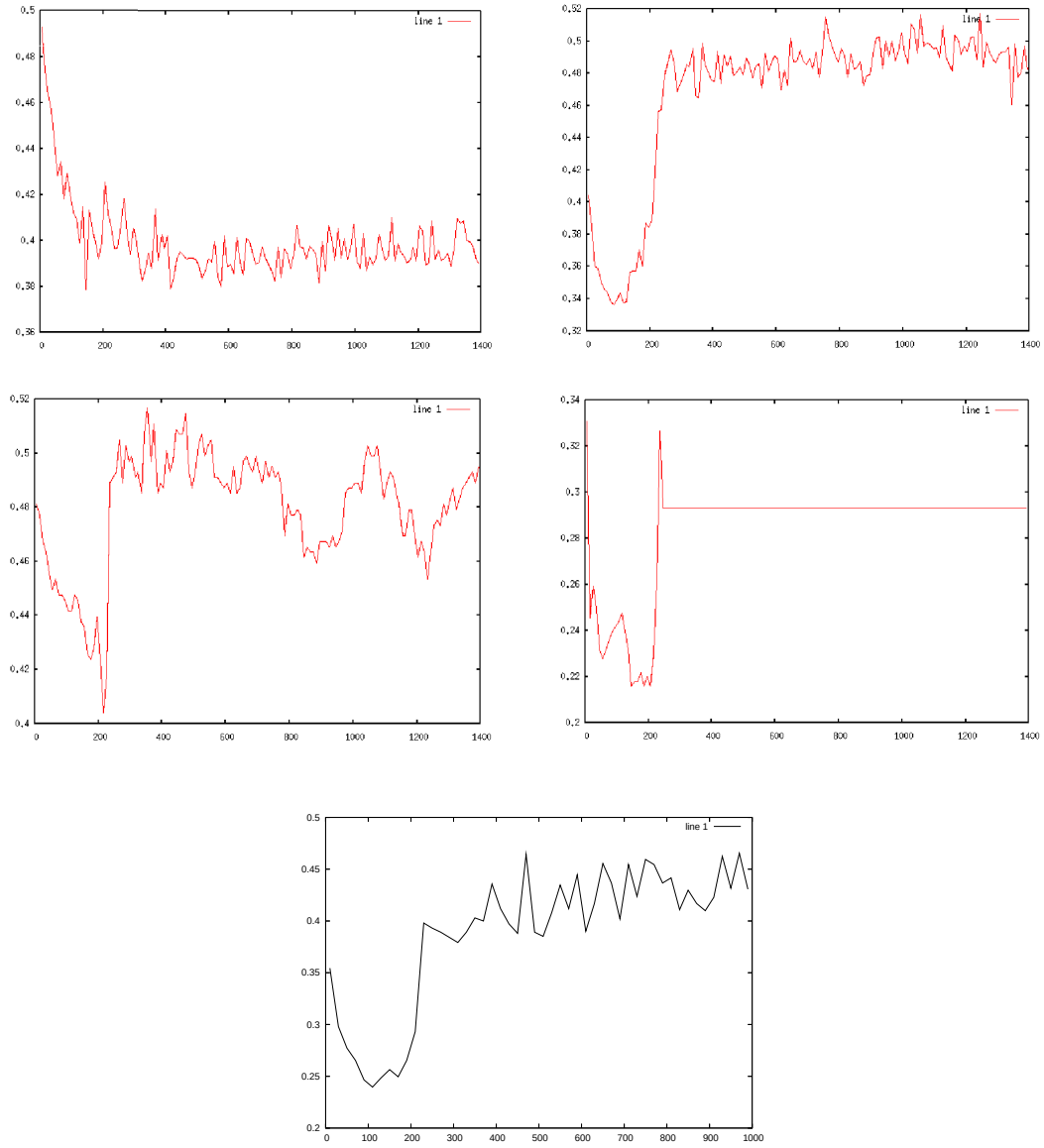


FIG. 7.1 : *Résultats sur l'erreur en généralisation sur les données du challenge TREC (données textuelles, les textes sont représentés comme des vecteurs de mots, présents ou non). En abscisse on donne la dimension de l'espace dans lequel on a projeté. De gauche à droite et de haut en bas, erreur obtenue par une projection aléatoire + SVM, erreur obtenue par une projection aléatoire + moindres carrés, SVD+SVM, SVD+Moindres carrés et enfin bagging de projections aléatoires (pour les moindres carrés). On s'aperçoit que les données semblent plus facile à traiter dans une dimension plus petite pour les moindres carrés (nous avons démontré analytiquement ce comportement). Nos taux d'erreurs les plus faibles (quand on projette en dimension* ~ 150*) sont de la même grandeur que les meilleures approches publiées sur ces données. L'apport du bagging est net (il creuse le trou) et permet de gagner en stabilité.*

Chapitre 8

Taylor-based pseudo-metrics

Ce travail s'inscrit dans le cadre de l'optimisation dynamique stochastique. Le principe est d'optimiser $E(C(x, A))$, ou E est l'opérateur espérance, C une fonction de coût et A une variable aléatoire. x représente le contrôle sur lequel nous pouvons agir pour essayer de minimiser le Coût. On dira que x est une stratégie. Par comodité dans la suite nous noterons $EC(x, A)$ au lieu de $E(C(x, A))$.

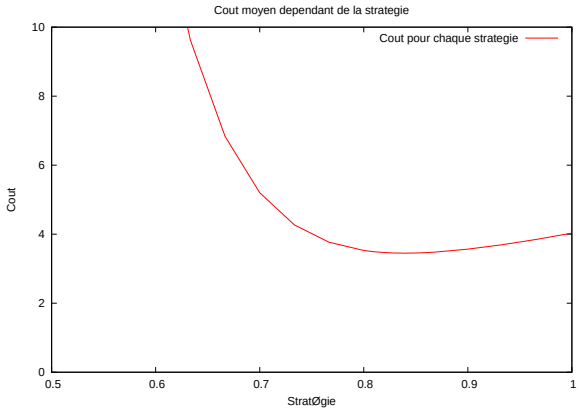


FIG. 8.1 : *Allure typique d'une fonction de Coût en dimension* 1. *Se tromper en étant un peu trop grand dans le choix de la stratégie est moins grave que de se tromper en étant trop petit.*

On suppose de plus que l'on dispose déjà d'un dispositif d'optimisation très efficace sur une certaine classe de variables aléatoires. Tout le problème est qu'il y a des variables aléatoires pour lesquelles l'optimiseur ne fonctionne pas. L'idée est alors de choisir parmi les variables pour lesquelles l'optimisation fonctionne, une variable qui est la plus « proche » possible de celle qui nous intéresse. Cette stratégie (guidée par ce qui se fait en entreprise, en particulier chez EDF) est représentée sur la figure 8.2. Ceci est dû à l'existence de solveurs relativement rapides (le temps se compte vite en semaines en optimisation stochastique) dans le cas ou le problème présente une certaine convexité (naturelle dans le cadre de la gestion de stocks).

On voit donc qu'il est important de savoir mesurer une distance entre deux variables

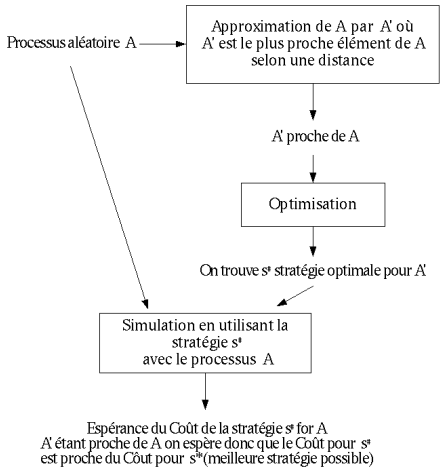


FIG. 8.2 : *Méthodologie classiquement utilisée pour résoudre le problème de l'optimisation de* $E(C(s,A))$ *quand on a déjà un optimiseur pour une certaine classes de variables aléatoires.*

aléatoires. Classiquement on utilise la distance de Kantorovitch-Rubinstein [Pflug; 2001] (qui dans les conditions de lipschitz sont plus performantes que des statistiques de plus haut niveau [Keefer; 1994]). Cependant ces distance ne tiennent jamais compte de la fonction de coût. En effet, dans notre problème, il ne s'agit pas seulement de donner une distance absolue entre deux variables aléatoires mais bien d'essayer de faire en sorte que la stratégie optimale pour A' soit la plus performante possible pour A. Et pour cela, il semble intéressant de tenir compte de la forme du problème. C'est ce que nous avons fait en proposant une série de pseudo-métriques basées sur des formules de Taylor.

On considère :

$$\begin{aligned}\delta_s(A,A') &= |EC(s,A') - EC(s,A)| \\ \delta_s(A,A') &= \left| E_A\left(\nabla_A C(s,A)(\pi(A)-A) + \varepsilon \parallel \frac{\partial^2 C}{\partial A^2} \parallel^{\infty}_{S,[A,\pi(A)]} (\pi(A)-A)^2/2 \right) \right|\end{aligned}$$

avec $\varepsilon \in [-1,1]$.

On suppose maintenant que $d(s,s^*) \leq \eta$ and $d(s_0,s^*) \leq \eta$. Alors

$$\begin{aligned}\delta_s(A,A') &= \Big| E_A\Big(\nabla_A C(s_0,A)(\pi(A)-A) \\ &\quad + \varepsilon \parallel \frac{\partial^2 C}{\partial A^2} \parallel^{\infty}_{S,[A,\pi(A)]} (\pi(A)-A)^2/2 \\ &\quad + 2\varepsilon'\eta \parallel \frac{\partial C}{\partial S} \parallel^{\infty}_{S,[A,A']} \Big) \Big|\end{aligned}$$

avec $\varepsilon \in [-1,1]$ et $\varepsilon' \in [0,1]$. Donc

$$\begin{aligned}\delta_s(A,A') &\leq distance(\pi,A,s_0) \\ &\quad + \left| E_A\left(\varepsilon \parallel \frac{\partial^2 C}{\partial A^2} \parallel^{\infty}_{S,[A,\pi(A)]} (A-\pi(A))^2/2 + 2\varepsilon'\eta \parallel \frac{\partial C}{\partial S} \parallel^{\infty}_{S,[A,A']} \right) \right|\end{aligned}$$

avec

$$\begin{aligned} distance(\pi, A, s_0) &= |deviation(\pi, A, s_0)| \\ deviation(\pi, A, s_0) &= E_A \nabla_A C(s_0, A)(\pi(A) - A) \end{aligned}$$

Cette «distance» (qui n'en est pas une au sens formel du terme), contient des termes négatifs et peut ainsi être proche de 0 alors que certains termes sont grands. On peut donc préférer une version qui semble plus sûre : $distance'(\pi, A, s_0) = E_A|\nabla_A C(s_0, A)(\pi(A) - A)|$. Nous verrons cependant qu'en pratique cette précaution ne s'avère pas nécessaire.

On peut pousser l'idée et faire un développement de Taylor à l'ordre 2 :

$$\begin{aligned} \delta_s(A, A') \leq\ & distance^{(2)}(\pi, A, s_0) \\ & + \left| E_A \left(\varepsilon \parallel \frac{\partial^3 C}{\partial A^3} \parallel^{\infty}_{S,[A,\pi(A)]} (A - \pi(A))^3/6 + 2\varepsilon'\eta \parallel \frac{\partial C}{\partial S} \parallel^{\infty}_{S,[A,A']} \right) \right| \end{aligned}$$

Avec $distance^{(2)}(\pi, A, s_0) = |deviation^{(2)}(\pi, A, s_0)|$ avec

$$\begin{aligned} deviation^{(2)}(\pi, A, s_0) &= E_A \nabla_A C(s_0, A)(\pi(A) - A) \\ &\quad + \frac{1}{2}(\pi(A) - A)^t H_A C(s_0, A)(\pi(A) - A) \end{aligned}$$

où $H_A C(s_0, A)$ désigne la Hessienne of $C(s_0, .)$.

Comme précédemment on définit :

$$\begin{aligned} distance'^{(2)}(\pi, A, s_0) &= E_A(|\nabla_A C(s_0, A)(\pi(A) - A)| \\ &\quad + |\frac{1}{2}(\pi(A) - A)^t H_A C(s_0, A)(\pi(A) - A)|) \end{aligned}$$

Dans nos expériences nous avons poussé jusqu'à l'ordre 3. Comme la dérivée troisième est réputée instable, nous l'avons remplacée par sa valeur absolue, qui donne l'ordre d'erreur sur la dérivée seconde.

De plus on peut raffiner notre modèle pour tenir compte d'un bruit additif dont on connaît la variance. Plus précisément, on définit $\pi'(a) = a + b$, ou b est un bruit additif (π' devient donc une fonction aléatoire). On optimise alors $EC(s, \pi'(A))$ au lieu de $EC(s, A)$.

$$\begin{aligned} EC(s, \pi'(A)) - EC(s, \pi(A)) &= EC(s, \pi'(A)) - EC(s, A) \\ &\quad + EC(s, A) - EC(s, \pi(A)) \\ &= deviation^{(2)}(\pi', A, s_0) - deviation(\pi, A, s_0) \\ &\quad + error\ term \end{aligned}$$

Si b est invariant par rotation le terme d'ordre 1 disparait et il reste

$$\frac{1}{2} E\ Tr(H, A)K - deviation^{(2)}(\pi, A, s_0)$$

ou K (choisi par l'utilisateur) est $L_2(b)^2$ et $Tr(H, A)$ la trace de la Hessienne $H_A C(s_0, A)$. La distance proposée est donc :

$$distance^{(2)}_K(\pi, A, s_0) = |deviation^{(2)}(\pi, A, s_0) - \frac{1}{2} E\ Tr(H, A)K|$$

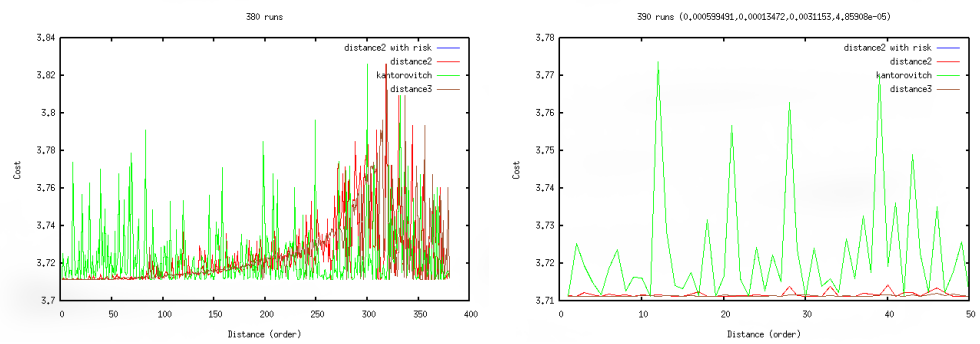


FIG. 8.3 : *Résultats en dimension* 1 *sans bruit. En abscisse on repère le rang de* A' *selon la distance considérée par la courbe. En ordonnée le Coût en simulation. Pour une «bonne» distance, on souhaite que le Coût soit petit quand la distance est petite. Le graphique de droite est un zoom sur les plus petites distances entre* A *et* $\pi(A) = A'$*. On voit que les résultats de Kantorovitch-Rubinstein sont bien moins bons que ceux de nos distances. Nous avons obtenus des résultats similaires en dimension* 2*.*

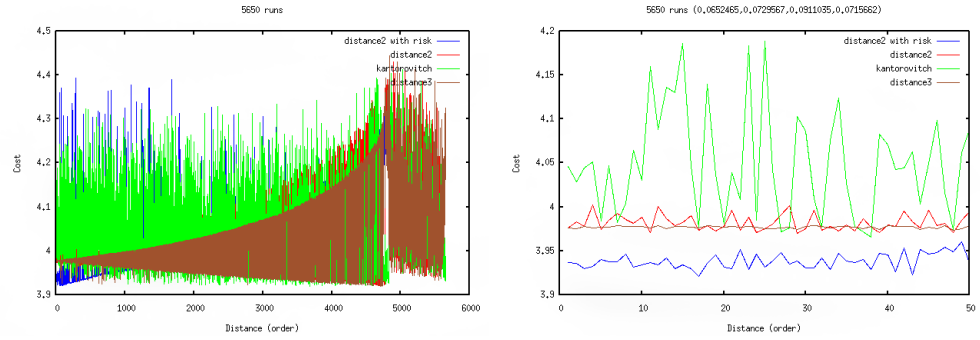


FIG. 8.4 : *Résultats en dimension* 1 *en présence d'un bruit additif. En abscisse on repère le rang de* A' *selon la distance considérée par la courbe. En ordonnée le Coût en simulation. Pour une «bonne» distance, on souhaite que le Coût soit petit quand la distance est petite. Le graphique de droite est un zoom sur les plus petites distances entre* A *et* $\pi(A) = A'$*. On voit que l'adaptation au bruit de la mesure représente un gain important.*

Chapitre 9

FISICA

La méthode FISICA est issue d'un projet sur la reconnaissance de scènes naturelles en vision. L'hypothèse de base est que les régularités présentes dans le monde sont traduites par des dépendances statistiques complexes, et des redondances. Un objectif des systèmes de vision est d'extraire ces dépendances statistiques de telle manière que les images puissent être expliquées en termes d'une collection d'événements indépendants. Cette hypothèse de décomposabilité est évidemment très forte, quoiqu'en disent par exemple des chercheurs comme Bruno Olshausen ou David Field, mais elle permet des traitements simples et elle semble compatible avec nos connaissances sur le cortex visuel.

Par ailleurs, les régularités présentes dans les images naturelles se trahissent par des corrélations statistiques particulières, avec, spécialement, des corrélations non linéaires (*e.g.* des corrélations de trois régions ou plus). L'idée est donc de chercher un codage des images en terme de combinaisons linéaires de fonctions de base représentant ces corrélations de haut degré).

De plus, les études de l'aire V1 du cortex visuel, qui font apparaître qu'une scène visuelle est codée par un petit nombre de neurones à l'intérieur d'une collection qui en comprend plus d'un milliard, suggèrent que le codage utilisé est économe ou clairsemé (*sparse-coding*) dans un système de fonctions de base sur-complet (*over-complete*) c'est-à-dire dont la dimension est supérieure à celle des formes codées.

Le problème est alors de trouver une base de telles fonctions qui soient adaptées au codage des images de scènes naturelles et telles qu'elles soient indépendantes et permettent un codage clairsemé des images attendues.

Le problème de recherche d'un codage clairsemé

On peut décrire le problème de recherche du codage linéaire d'une image $I(x, y)$ comme celui d'une base de fonctions $\phi_i(x, y)$ satisfaisant :

$$I(x,y) \;=\; \sum_i a_i \phi_i(x,y) \tag{9.1}$$

Le but est de trouver une base qui soit à la fois une base **complète** (permettant la description de l'espace d'entrée) et **clairsemée** (permettant la représentation des images avec peu de coefficients $a_i \neq 0$). En d'autres termes, la distribution de probabilité sur chacun des coefficients devrait être très piquée autour de 0, avec des queues épaisses. Une telle distribution a une faible entropie et permet également de réduire les dépendances statistiques entre les fonctions de base.

Ce problème peut être formulé comme un problème d'optimisation avec la fonction de coût :

$$E(a,\phi) \;=\; \sum_{x,y} \big[I(x,y) \;-\; \sum_i a_i \phi_i(x,y)\big]^2 \;+\; \beta \sum_i S(\frac{a_i}{\sigma_i}) \tag{9.2}$$

où $\sigma_i^2 = \langle a_i^2 \rangle$. Le premier terme mesure l'adéquation du code avec l'image, suivant une mesure d'écart quadratique, tandis que le second terme tend à favoriser les codes clairsemés, en fonction de la fonction de coût S.

Deux démarches existent. Soit utiliser un code linéaire choisi *a priori*, comme les transformées de Fourier ou les représentations en ondelettes[1]. 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) [Hyvarinen; 1999, Hyvarinen and Oja; 2000, Hyvarinen et al.; 2001] et la factorisation non-négative en matrices (NMF) [Lee and Seung; 1999]. Si l'analyse ACP est trop limitée pour s'appliquer au domaine de la vision, en revanche des résultats intéressants ont été obtenus par ICA et par NMF.

9.0.4 Les bases de la méthode FISICA

Si l'approche par l'analyse en composantes indépendantes d'un codage clairsemé est impossible, une approche directe est-elle envisageable ?

En faisant l'hypothèse que les données résultent d'une somme de formes latentes, et que ces formes pour être intéressantes doivent figurer suffisamment souvent dans les données, le rapprochement avec la technique de recherche de motifs fréquents s'impose. Les données étant décrites par un ensemble de descripteurs (attributs-valeur), on cherche les conjonctions d'attributs-valeur présentes dans un certain pourcentage des données. Grâce à certaines contraintes, on peut guider la recherche de tels motifs de manière à

[1] Elles ont à la fois d'intéressantes propriétés mathématiques et une certaine plausibilité biologique, qui en font une représentation de choix dans les travaux sur le système visuel.

favoriser la découverte d'un ensemble de primitives (les motifs) permettant un codage clairsemé des données.

À ces contraintes peuvent s'ajouter des critères supplémentaires liées au domaine d'application permettant de sélectionner les motifs fréquents les plus intéressants. Par exemple, dans le domaine de l'analyse d'images, on pourra favoriser la recherche de fonctions de base ou motifs correspondant à des régions connexes, ou à des lignes (voir plus bas, les résultats expérimentaux).

Les fonctions de base cherchées sont des conjonctions d'attributs-valeur, on parlera aussi d'atomes. Dans le cas de l'analyse d'images, il s'agira de collections de pixels, chacun de ceux-ci étant associé à un niveau de gris donné. Une fonction de base est ainsi une fonction booléenne prenant la valeur `vrai` si la collection d'atomes (pixel = valeur) correspondante est satisfaite dans l'image étudiée. On dira que le support d'une fonction de base est de $\varepsilon\%$ si cette fonction prend la valeur `vrai` dans $\varepsilon\%$ des images testées. Réciproquement, on appellera code d'une forme d'entrée (ici une image), l'ensemble des fonctions de base vérifiant cette forme.

Dans ce nouveau cadre, les propriétés désirées du système de codage se traduisent comme suit :

1. **Représentativité**. Chaque fonction de base a un support supérieur à $\varepsilon\%$. Elle est donc suffisamment représentée dans la base d'exemples pour être utile.
2. **Parcimonie**. Peu de fonctions de base sont vérifiées par un exemple (e.g. image). On vérifie ainsi l'une des propriétés du codage clairsemé.
3. **Suffisance**. Tout exemple rend vrai un nombre minimal de fonctions de base.
4. **Orthogonalité**. Pour chaque paire de fonctions de base, l'intersection des exemples qui rendent `vrai` l'une et l'autre est réduite. Les exemples sont donc décrits par des codes différents.

Nous avons adapté la méthode de recherche de motifs fréquents dans une base de données pour chercher un codage tendant à vérifier les propriétés ci-dessus.

La recherche de motifs fréquents de taux de couverture faible dans des données décrites par de nombreux attributs ne peut s'effectuer sans précautions. C'est pourquoi nous présentons rapidement l'algorithme développé à cet effet.

Approche randomisée

Nous avons développé une méthode de construction incrémentale de motifs fréquents par ajouts successifs d'atomes (ici, de pixels d'un certain niveau de gris) en les sélectionnant à chaque pas afin que le motif en construction satisfasse aux critères désirés. L'exploration des motifs fréquents est donc maintenant stochastique, guidée mais non exhaustive. Des essais successifs peuvent ainsi produire des bases de motifs différentes.

Algorithme 4 Recherche itérative et stochastique de motifs fréquents.

Paramètres : taux de couverture $\varepsilon\%$. Nombre de motifs cherchés = N.
Nombre de motifs trouvés = $n \leftarrow 0$.
TantQue $n \leq N$ **Faire**
  Choix dans un exemple x_i encore peu couvert, d'un premier atome a_0 présent dans au moins ε% des exemples.
  motif $\leftarrow a_0$
  TantQue Taux de couverture de motif > ε% **Faire**
    Tirer au hasard un atome a de x_i couvrant au moins ε% des exemples et peu utilisé dans les motifs existants et satisfaisant des contraintes additionnelles sémantiques (voir section 9.0.5).
    Si motif $+a$ couvre au moins ε% des exemples **Alors**
      motif $\leftarrow$ motif $+a$
    FinSi
  Fin TantQue
Fin TantQue

Exploitation pour l'apprentissage supervisé

Une fois N fonctions de base trouvées sur un ensemble d'apprentissage, chaque exemple est recodé, devenant un vecteur de N booléens prenant la valeur `vrai` ou `faux` selon que la fonction de base correspondante couvre l'exemple ou non. [2].

Dans le nouvel espace d'exemples ainsi construit, il est possible d'utiliser n'importe quelle méthode d'apprentissage supervisé. Dans les expériences rapportées ici, nous avons utilisé une méthode de classification par plus proche voisin. Les exemples d'apprentissage utilisés pour la recherche de fonctions de base sont également employés comme exemples étiquetés servant à la classification des exemples testés.

9.0.5 Application à la reconnaissance d'images

La méthode développée a été testée sur des tâches de classification d'images de scènes naturelles et de chiffres manuscrits. Elle implique deux phases : d'abord une étape de détermination d'une base de fonctions de base permettant de redécrire les données, ensuite l'emploi du système de codage ainsi obtenu pour classer de nouvelles formes.

Dans le cas de la reconnaissance de scènes naturelles, le problème consiste à apprendre à reconnaître des images de scènes naturelles classées en 12 catégories (voir la figure 9.1). Ces images proviennent de la base COREL (`http://www.corel.com/gallery_line/`). Les images sont redécrites par 128 $\times$ 128 pixels en 128 niveaux de gris. Pour ces expériences, la base utilisée comportait 1082 images réparties également entre les 12

[2]Nous avons aussi utilisé une formule d'appariement plus souple utilisant une fonction sigmoïde à valeur dans $[0, 1]$ qui tient compte du nombre d'atomes de la fonction de base qui couvrent l'exemple, et donnant un recodage dans $[0, 1]^N$ au lieu de $\{0, 1\}^N$

FIG. 9.1 : *Échantillon d'images utilisées dans cette étude. Noms des classes : avions (1), plats (2), Utah (3), minéraux (4), chiens (5), poissons (6), verres (7), papillons (8), porcelaines (9), figurines (10), voitures (11), fleurs (12). (Cette figure est reprise de [Denquive and Tarroux; 2003]).*

classes. Nous considérerons dans la suite que chaque pixel est un attribut pouvant prendre une valeur parmi 128. La dimension de l'espace d'entrée est donc dans ce cas de 32768 = 128×128.

Pour l'application étudiée, nous avons fixé à 1000 le nombre de fonctions de base recherchées. Plusieurs bases ont été obtenues en faisant varier les paramètres suivants :

- **Taux de couverture** : 1%, 2%, 5% et 10%
- **Critère sémantique additionnel**. Nous avons introduit des contraintes supplémentaires sur la construction des fonctions de base afin de tester des équivalences possibles avec d'autres types de codages classiques en traitement d'images. Quatre conditions ont été testées :

  1. Aucune contrainte.
  2. Les fonctions de base doivent correspondre à des régions connexes sur l'image : un nouveau pixel n'est ajouté à la fonction de base courante que s'il est contigu à un pixel déjà sélectionné.
  3. Les fonctions de base doivent correspondre à des lignes de l'image (régions de dimension 1). L'idée ici est de voir si l'on peut forcer le système de codage à retenir des contours dans l'image.
  4. Les fonctions de base doivent correspondre à des lignes raisonnables de l'image, c'est-à-dire plus contraintes dans les changements de directions possibles. Cette contrainte a été imposée lorsqu'il s'est avéré que la précédente produisait des «vermiceaux» remplissant des régions et non pas des lignes.

Environ la moitié des images de la base initiale de 1082 images, soit 500, ont été utilisées pour le calcul des fonctions de base. (Note : notre algorithme calcule une base de 1000 motifs en quelques minutes sur un PC équipé d'un Pentium II à 266 Mhz et 384

Mo de RAM). Les figures 9.2 et 9.3 illustrent le type de fonctions de base obtenues pour certaines conditions expérimentales.

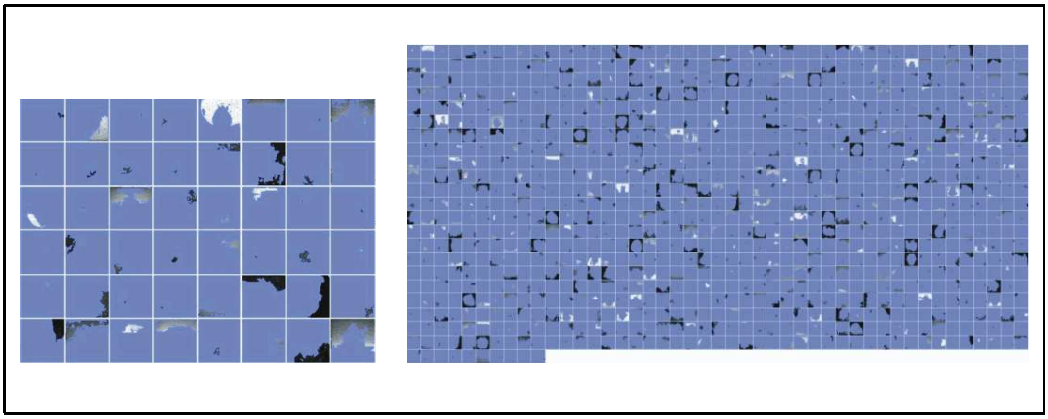


FIG. 9.2 : *(À gauche) Le détail de quelques unes fonctions de base obtenues sur des images 64 × 64 en 16 niveaux de gris avec un taux de couverture de 1%, et en cherchant des régions connexes de l'image. Dans les images accessibles sur le site internet, le fond bleu correspond aux zones qui ne font pas partie des fonctions de base, tandis que les fonctions de base sont figurées par des pixels de niveaux de gris variés. Ici, le gris moyen correspond au fond. (À droite) Figure une base de 1000 fonctions de base.*

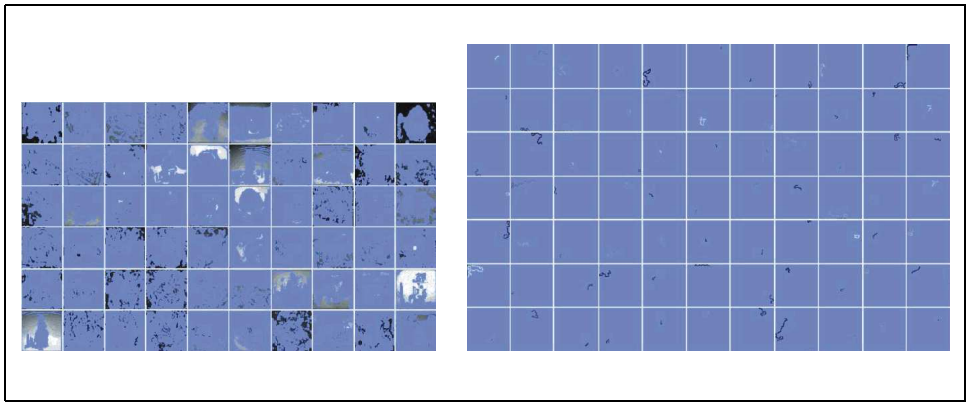


FIG. 9.3 : *(À gauche) Des exemples de fonctions de base obtenues avec un taux de couverture de 1%, et sans contrainte. (À droite) Des exemples de fonctions de base obtenues avec un taux de couverture de 1%, et sous contrainte de linéarité raisonnable. L'examen des motifs trouvés montre qu'ils ne correspondent pas à des contours des images de la base d'exemples.*

L'histogramme présenté dans la figure 9.4 (à gauche) permet de contrôler l'orthogonalité des fonctions de base obtenues. Ces fonctions de base sont orthogonales lorsqu'elles sont rarement vérifiées par les mêmes images. La figure montre que les différentes bases obtenues pour des conditions différentes peuvent effectivement être considérées comme orthogonales. Inversement, l'histogramme de la figure 9.4 (à droite) indique le nombre de fonctions de base qui sont vérifiées par les images. On constate que ce nombre varie autour d'une dizaine, ce qui traduit bien que le codage obtenu est clairsemé.

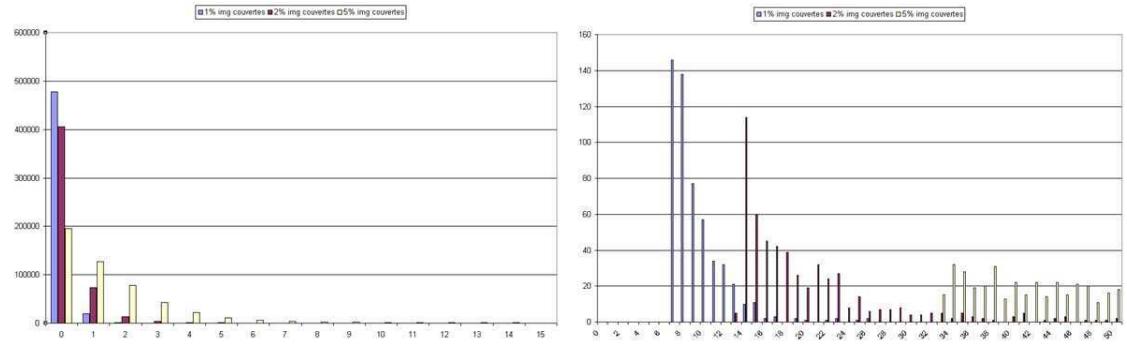


FIG. 9.4 : *À gauche : Histogramme représentant le nombre y de couples de fonctions (en ordonnée) ayant x images en commun (en abscisse). Ces fonctions ont été calculées à partir d'images de taille 64x64 en 16 niveaux de gris et sous la contrainte de connexité. Les résultats sont présentés pour des fonctions de base de taux de couverture 1%, 2% et 5%. Les fonctions de base obtenues pour $\varepsilon = 1\%$ sont les plus orthogonales entre elles. À droite : Histogramme représentant le nombre y d'images (en ordonnée) activant x motifs (fonctions de base) en abscisse. Ces fonctions ont été calculées à partir d'images de taille 64x64 en 16 niveaux de gris et sous la contrainte de connexité. Les résultats sont présentés pour des fonctions de base de taux de couverture 1%, 2% et 5%. Plus le taux de couverture est élevé, plus chaque exemple est couvert en moyenne par un nombre élevé de fonctions. On peut ainsi régler la parcimonie de la représentation et donc son caractère clairsemé.*

Les résultats en classification

Les performances en classification ont été calculées sur les 582 images non utilisées pour déterminer la base des fonctions de base. Toutes les images sont recodées en utilisant les fonctions de base obtenues, et s'expriment donc sous la forme d'un vecteur de 1000 valeurs booléennes (en fait, dans certaines expériences, cette valeur booléenne était remplacée par une mesure plus continue d'appariement de l'image avec une fonction de base). Les 500 images employées pour la détermination des fonctions de base sont également utilisées comme base d'exemples étiquetées. Les images à classer sont alors étiquetées en utilisant une méthode de plus proche voisin. Dans les expériences rapportées ici, la distance utilisée est la distance L_1.

La table 9.0.5 fournit les résultats obtenus avec une base de 1000 fonctions de base de taux de couverture de 5% soumis à la contrainte de connexité. Quoique les différents nombres puissent varier sensiblement, on observe en général que les résultats obtenus dans une grande variété de conditions sont assez similaires en moyenne. Ils sont très sensiblement supérieurs à ceux rapportés dans [Denquive and Tarroux; 2003] utilisant un réseau de neurones à bases radiales, ou à ceux que nous avons obtenus avec des Séparateurs à Vastes Marges (SVM) sur un codage à base d'ondelettes de Gabor[3].

[3] Un travail non publié réalisé avec Olivier Bousquet au printemps 2001. Autant de classifieurs «un-contre-tous» que de classes. Les descripteurs consistaient en 20 filtres de Gabor (4 orientations $\times$ 5 fréquences).

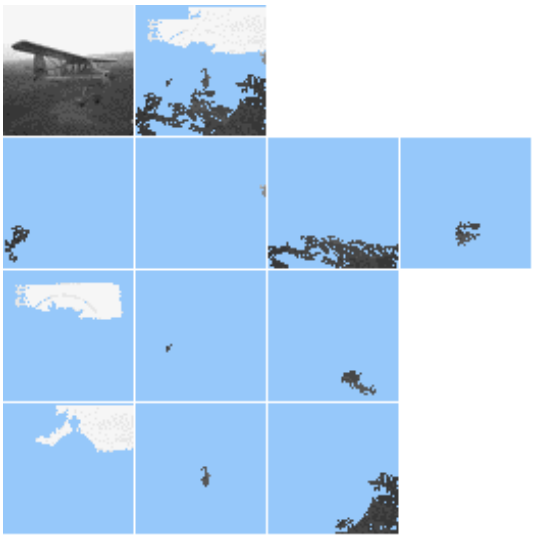


FIG. 9.5 : *Exemple de recodage d'un avion dans la base des FIS (taux de couverture 5%, connexité imposée).Il est clair que le codage obtenu ici ne permet pas de reconstruire l'image d'origine et ne vérifie donc pas la propriété d'approximation des méthodes classiques de codage*

	Av	Pl	Ut	Mi	Ch	Po	Ve	Pa	Por	Fi	Vo	Fl
Avi	**67%**	2%	-	-	2%	2%	10%	10%	4%	2%	-	-
Pla	-	**21%**	-	2%	7%	19%	10%	12%	5%	-	19%	5%
Uta	17%	-	**33%**	-	7%	-	-	3%	10%	10%	13%	7%
Min	-	-	-	**100%**	-	-	-	-	-	-	-	-
Chi	26%	5%	7%	-	**14%**	9%	12%	9%	5%	2%	12%	-
Poi	5%	13%	3%	8%	-	**13%**	18%	21%	-	3%	10%	8%
Ver	2%	2%	-	-	10%	7%	**43%**	-	21%	5%	7%	2%
Pap	6%	6%	-	-	2%	14%	14%	**35%**	6%	-	12%	4%
Por	2%	2%	-	-	-	2%	-	12%	**70%**	10%	-	2%
Fig	-	-	-	-	-	-	6%	-	24%	**70%**	-	-
Voi	21%	6%	-	-	4%	4%	8%	4%	4%	29%	**19%**	-
Fle	2%	9%	-	-	-	9%	21%	14%	-	-	16%	**28%**

TAB. 9.1 : *Matrice de confusion obtenue avec des fonctions de base de taux de couverture $\epsilon = 5\%$ sous contrainte de connexité, en utilisant une formulle d'appariemment continu entre les images et les fonctions de base.*

Chapitre 10

ROGER

ROGER est un algorithme évolutionnaire d'apprentissage présenté initialement dans [Sebag et al.; 2003] et [Sebag et al.; 2004]. Le principe est d'utiliser des stratégies évolutionnaires ($(\mu + \lambda)$-ES) pour déterminer les hypothèses maximisant l'aire sous la courbe ROC (AUC criterion, [Bradley; 1997, Ling et al.; 2003]). Notons au passage que l'AUC est équivalente aux statistiques du test de Wilcoxon [Yan et al.; 2003].

Un des atouts majeurs de ROGER est d'autoriser la construction de certaines hypothèses non linéaires. En fait pour une hypothèse h, on mesure la distance L_1 a un point c dans $\mathbb{R}^d$. Un individu de la population évoluée est de la forme $Z = (w_1, \ldots, w_d, c_1, \ldots, c_d)$ et est associé à l'hypothèse h_Z définie comme suit :

$$h_Z : \mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d \mapsto \mathbb{R}, E(h_Z(\mathbf{x})) = \sum_{i=1}^{d} w_i \times |x_i - c_i|$$

Ainsi, ROGER explore un espace de dimension $\mathbb{R}^{2d}$. La taille d'un individu croit linéairement avec le nombre d'attributs mais l'algorithme conserve la possibilité de détecter des non linéarités dans les données (grâce aux c_i et aux valeurs absolues). La fonction de fitness $\mathcal{F}$ que ROGER essaye de maximiser est $\mathcal{F}(\mathcal{Z}) = P(h_Z(\mathbf{x}_i) > h_Z(\mathbf{x}_j) | y_i > y_j)$, qui est la statistique de Wilcoxon associée à h_Z.

Un outil existe en ligne de commande mais la documentation est inexistante, ce qui rend son utilisation un peu délicate. J'ai commencé a développer une version de ROGER basée sur EO mais cette implémentation n'a pas atteint le niveau de *release* faute de temps. Celle-ci m'a servi dans des travaux d'évolution par îles dans l'apprentissage de préférences d'experts (pour essayer de regrouper des experts). Il s'agit d'une collaboration avec l'équipe d'Antonio Bahamonde en Espagne. Les résultats ne sont pas encore assez matures pour être publiés.

Chapitre 11

WoB

«World in a Bottle» (WoB) est un simulateur de robots écrit en C++ utilisant OpenGL pour l'affichage. Conçu pour simuler des Khépéras, sa conception lui permet de simuler facilement d'autre types de robots (de formes différentes possédant des capteurs différents). Écrit en collaboration avec un autre membre de l'équipe TAO, il est utilisé en particulier par le laboratoire de l'École Polytechnique pour les modeX de robotique. Disponible sous licence GPL, il a été téléchargé à ce jour 654 fois par 152 machines différentes. Il représente un total de 29 784 lignes de code (sans compter les `makefiles` auto-générés et autres fichiers de configurations).

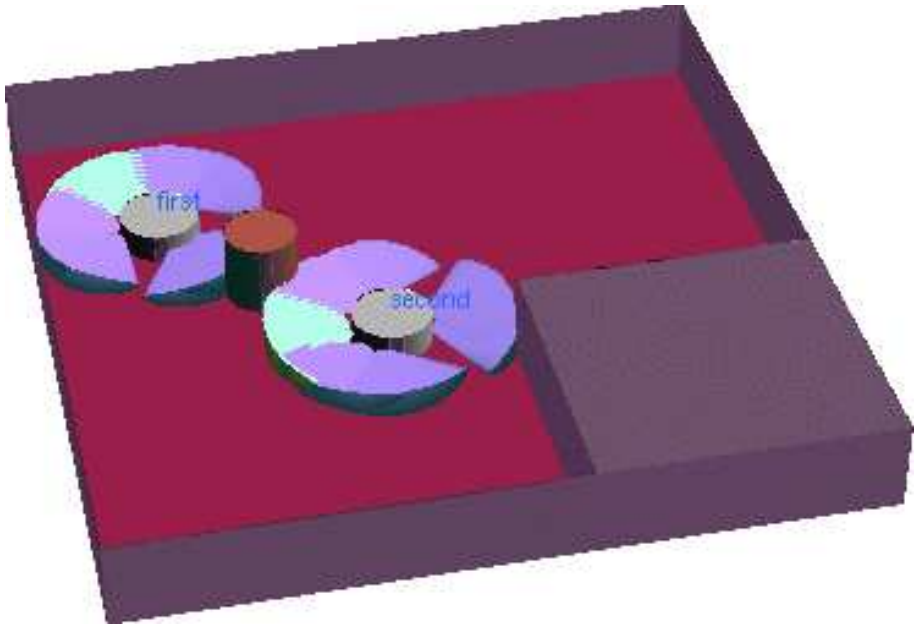


FIG. 11.1 : *Vue de la fenêtre graphique du simulateur, les zones colorées autour des robots représentent les cônes de détection des capteurs.*

En quelques mots, voila les principales caractéristiques de ce simulateur.

Avantages :

- Affichage 3D en temps réel de la simulation et des valeurs des capteurs. Cet affichage peut être désactivé pour accroître la vitesse de simulation. OpenGL n'est utilisé que pour cet affichage, ce qui permet d'utiliser le simulateur sur une machine dépourvue d'OpenGL (utile sur les Clusters par exemples).
- **L'environnement des robots est particulièrement facile a paramétrer**, et peut-être stocké dans des fichiers facilement éditables par un humain.
- Il n'y a **pas de limitation au nombre de robots** présents simultanément dans la simulation (mis à part la puissance de calcul de la machine).
- Les robots peuvent être a tout moment contrôlés par un être humain au clavier.
- Simulation des capteurs IR en mode passif et actif. De plus il est possible de modifier les capacités de ces capteurs.
- Simulation des caméras 1D et 2D (nécessite OpenGL pour la caméra 2D).
- Les obstacles peuvent être fixés au sol ou mobiles.
- **L'écriture d'un contrôleur de robots est extrêmement simplifiée** et ne nécessite pas de connaissance sur la structure du simulateur (se référer au tutoriaux).
- Il est possible d'écrire un **contrôleur dans n'importe quel langage** car il existe une **version «serveur»** de WoB. Cette version permet de recevoir des ordres via un *socket* et utilise le même protocole de communication que les robots khépéras réels. Il faut juste être capable de lire/écrire sur un *socket*. Ce point permet également d'utiliser un réseau pour distribuer les calculs.
- Le contrôleur que vous avez écrit et testé en simulation peut être transposé à un **Khépéras réel simplement en pressant une touche**. Attention, dans le monde réel certains capteurs ne fonctionnent plus (*i.e.* il est impossible de demander sa localisation absolue ou de différencier un choc avec un élément de l'environnement ou un autre robot).
- Le simulateur a été pensé pour **s'interfacer avec EO** (une librairie pour les algorithmes génétiques)
- Permet de réaliser des vidéos des expériences simulées pour réaliser des démonstrations sur n'importe quelle machine.
- Facilement extensible à d'autres types de robots (la forme peut déjà être quelconque et les capteurs montés modifiés sans grosse difficulté).

Limitations :

- Pour le moment, tous les robots simulés n'ont que deux roues. Cette limitation n'est pas très difficile à supprimer mais n'est pas à la portée d'un novice sur le simulateur.
- La gestion du gripper du Khepera n'est pas très évoluée : le robot attrape et relâche les objets instantanément, ne rate jamais le grip quand l'objet est bien placé et ne voit pas sa forme modifiée par la présence de l'objet grippé (dans les collisions). Ce point est difficile a améliorer.
- Dans le monde réel, le simulateur ne peut contrôler qu'un seul robot à la fois (via le port série).

Chapitre 12

BioRelief

BioRelief est la mise en oeuvre des techniques de sélections d'attributs inspirées de RELIEF dans le cas de l'analyse de Micro-Arrays. Développé en C++ sous licence GPL, il est constitué de deux parties :

- Un moteur en ligne de commande. Fonctionnant à la fois sous Windows et Linux, respectant les standards de diffusion GNU - 1321 lignes de code - .
- Une interface pour l'utilisateur. Basée sur la librairie WxWindows [lots of peoples], ce qui assure un passage sans douleur sur de nombreuses plateformes (dont Linux et Windows) - 2053 lignes de codes plus les ressources-.

12.1 Utilisation en ligne de commande

Tout d'abord voici les différentes options actuellement implémentées :

```
Help for BIO-RELIEF - command line version

-h or --help : Display this message and exit
-c FILE.conf : Set the configuration FILE to use
-d FILE.data : choose the data FILE to read
-o FILE      : Create FILE and use it to store results
-p NUMBER    : Do a comparaison to randomness using NUMBER permutations
-b           : Use only balanced permutations (no effect is -p is not set )
-nointerval  : Do not calc quantiles around the randomness'curve (saves memory)
-inf NUMBER  : Set the lower bound of reject intervall (default is 0.025)
-sup NUMBER  : Set the upper bound of reject intervall (default is 0.025, so
default
               confidence intervall is 95%)
```

Le principe est de fournir deux fichiers à BioRelief le premier, le fichier de configuration, contient la liste de noms des classes à discriminer ainsi que le nombre de voisin à utiliser pour cette classe. Dans une version ultérieure il sera possible de demander à ce que ce nombre de voisins s'auto-optimise. Dans l'interface, par défaut on propose d'utiliser la moitié des éléments de la classe.

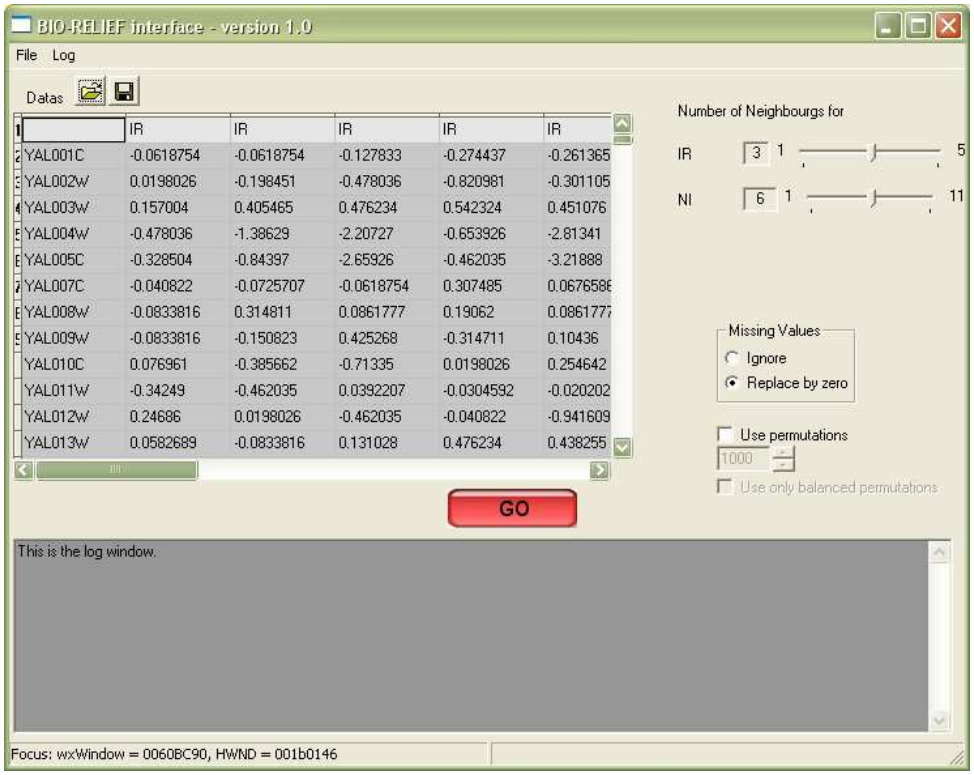


FIG. 12.1 : *Vue de l'Interface de BioRelief*

Le format est très simple :

```
name_of_class1 2
name_of_class2 2
name_of_class_xx number_of_neighbours_to_use_in_this_class
```

Le second fichier contient les données proprement dites. Sur la première ligne se trouve la classe des échantillons. La première colonne contient le nom des gènes étudiés. Le reste du fichier est constitué des données proprement dites (les données sont des nombres en flottant ; quand la valeur est manquante logiciel attend un `nan`).

Exemple de fichier de données :

```
class1 class1 class2 class2 class1 class2
gene1 0 0 0 0 1 2
gene2 -1 1 0 -1 1 0
gene3  0 1 0 1 -1 1
gene4 1 1 0 0 0 0
gene5 0 0 1 1 0 1
gene6 -1 -1 1 1 -1 0
gene7 1 1 -1 -1 0 1
gene8 nan -1 1 0 -1 0
gene9 0 1 -1 -1 0 0
gene10 0 0 0 0 0 1
gene11 0 1 0 0 -1 1
```

En sortie on obtient la liste des gènes avec leur pertinence estimée.

```
relief -d example

Configuration file not given ; assuming its name is example.conf
gene1 -0.25
gene2 0
gene3 -0.416667
gene4 0.416667
gene5 1
gene6 1.33333
gene7 0.25
gene8 0.25
gene9 0.333333
gene10 0
gene11 -0.416667
```

Si l'on souhaite estimer les écarts au hasard, il faut utiliser l'option :
`-p nb_de_permutations`.
Le résultat comportera alors cinq colonnes :

- La première contient le seuil minimal de pertinence exigé.
- La seconde le nombre de gènes dépassant ce seuil avec les vraies classes.
- La troisième, le nombre de gènes dépassant ce seuil en moyenne en permutant les classes.
- Les deux dernières contiennent respectivement les quantiles inférieurs et supérieurs. Par défaut 95% des permutations sont comprises entre les deux bornes.

12.2 Utilisation de l'interface

C'est encore plus facile (voir la figure 12). Cette fois le fichier d'entrée est au format `csv` lisible par Excel. La première ligne contient la classe de chaque colonne. Les différents réglages se font au niveau de la page principale et des valeurs par défaut sont proposées pour le nombre de voisins à utiliser dans chaque classe. A noter que **l'interface s'adapte automatiquement selon le nombre de classes détectées dans le fichier de données**.

Bibliographie

D. Achlioptas. Database-friendly random projections. In Symposium on Principles of Databases systems (PODS), pages 274–281, 2001.

R.J. Adler. An introduction to continuity, extrema, and related topics for general gaussian processes. In IMS Lecture Notes-Monograph Series, pages vii + 160, 1990.

K. Alexander. Probability inequalities for empirical processes and a law of the iterated logarithm. In Annals of Probability, volume 4, pages 1041–1067, 1984.

Dana Angluin. Queries revisited. Theor. Comput. Sci., 313(2) :175–194, 2004. ISSN 0304-3975. doi : http://dx.doi.org/10.1016/j.tcs.2003.11.004.

Jinbo Bi, Kristin Bennett, Mark Embrechts, Curt Breneman, and Minghu Song. Dimensionality reduction via sparse support vector machines. J. Mach. Learn. Res., 3 :1229–1243, 2003. ISSN 1533-7928.

Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In ICML '04 : Proceedings of the twenty-first international conference on Machine learning, page 11, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5. doi : http://doi.acm.org/10.1145/1015330.1015360.

A. Blum, P. Chalasani, S. Goldman, and D. Slonim. Learning with unreliable boundary queries, 1995. URL `sherry.ifi.unizh.ch/article/blum97learning.html`.

Avrim Blum. On-line algorithms in machine learning. In Developments from a June 1996 seminar on Online algorithms, pages 306–325, London, UK, 1998. Springer-Verlag. ISBN 3-540-64917-4.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification : some recent advances. ESAIM Probability & Statistics, 2005. to appear.

O. Bousquet. A bennett concentration inequality and its application to suprema of empirical processes. In C. R. Math. Acad. Sci. Paris, volume 334(6), pages 495–500, 2002.

A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. In Pattern Recognition, 1997.

L. Breiman. Arcing classifiers. In Annals of Statistics, pages 26(3) :801–845, 2003.

Leo Breiman. Bagging predictors. Machine Learning, 24(2) :123–140, 1996. URL `citeseer.ist.psu.edu/breiman96bagging.html`.

K. Brewer. Stratified designs. In Wiley, New York, pages 1–8, 1992.

G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods : a survey and categorisation, 2005. URL `citeseer.ist.psu.edu/article/brown04diversity.html`.

C.H. Bryant, S.H. Muggleton, S.G. Oliver, and D.B. Kell. C.h. bryant, s.h. muggleton, s.g. oliver and d.b. kell. In Elect. Transactions in Artificial Intelligence, 5-B1, no012, November, pages 1–36, 2001.

W.L. Buntine. Decision tree induction systems : A bayesian analysis. pages 190–197, 1989.

J. Burkardt. niederreiter2.cc, niederreiter2.h. 2003. URL `http://www.csit.fsu.edu/\~burkardt/cpp\_src/niederreiter2/niederreiter2.html`.

C. Cervellera and M. Muselli. A deterministic learning approach based on discrepancy. In IEEE transactions on Neural Networks 15, pages 533–544, 2004.

S. Dasgupta, A. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In Eighteenth Annual Conference on Learning Theory (COLT), 2005.

Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, Advances in Neural Information Processing Systems 17, pages 337–344. MIT Press, Cambridge, MA, 2005.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. Journal of the American Society of Information Science, 41(6) :391–407, 1990.

A. Dembo and O. Zeitouni. Large Deviations Techniques and Applications, 2nd ed. 1998.

N. Denquive and P. Tarroux. Codages fréquentiels et catégorisation de scènes visuelles. Technical report, CNRS-LIMSI, 2003.

Philip Derbeko, Ran El-Yaniv, and Ron Meir. Variance optimized bagging. pages 60–71.

L. Devroye. Bounds for the uniform deviation of empirical measures,. In Journal of Multivariate Analysis, volume 12, pages 72–79, 1982.

L. Devroye, L. Györfi, and G. Lugosi. A probabilistic Theory of Pattern Recognition. 1997.

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees : Bagging, boosting, and randomization. Machine Learning, 40(2) :139–157, 2000. URL `citeseer.ist.psu.edu/dietterich98experimental.html`.

Pedro Domingos. Metacost : a general method for making classifiers cost-sensitive. In KDD '99 : Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 155–164, New York, NY, USA, 1999. ACM Press. ISBN 1-58113-143-7. doi : http://doi.acm.org/10.1145/312129.312220.

Michael Eisen. Cluster and Treeview manual. Stanford University, rana.lbl.gov/manuals/ClusterTreeView.pdf, 98-99.

R. Esposito and L. Saitta. Monte carlo theory as an explanation of bagging and boosting. In IJCAI'03, pages 499–504, 2003.

Roberto Esposito and Lorenza Saitta. A monte carlo analysis of ensemble classification. In ICML '04 : Proceedings of the twenty-first international conference on Machine learning, page 34, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5. doi : http://doi.acm.org/10.1145/1015330.1015386.

George S. Fishman. Monte Carlo, Concepts, Algorithms and Applications. 1996.

Dmitriy Fradkin and David Madigan. Experiments with random projections for machine learning. In SIGKDD2003, 2003. URL `citeseer.ist.psu.edu/576622.html`.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In International Conference on Machine Learning, pages 148–156, 1996. URL `citeseer.ist.psu.edu/freund96experiments.html`.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. Machine Learning, 28(2-3) :133–168, 1997. URL `citeseer.ist.psu.edu/freund97selective.html`.

A. Giordana and L. Saitta. Phase transitions in relational learning. In Machine Learning, pages 41 :217–251, 2000.

Sally A. Goldman and Robert H. Sloan. The power of self-directed learning. Mach. Learn., 14 (3) :271–294, 1994. ISSN 0885-6125. doi : http://dx.doi.org/10.1023/A:1022605628675.

Yves Grandvalet. Bagging equalizes influence. Mach. Learn., 55(3) :251–270, 2004. ISSN 0885-6125. doi : http://dx.doi.org/10.1023/B:MACH.0000027783.34431.42.

C. Guerra-Salcedo and D. Whitley. Genetic approach to feature selection for ensemble creation. In Morgan Kaufmann, editor, GECCO-99 : Proceedings of the Genetic and Evolutionary Computation Conference, pages 236–243, 1999.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. J. Mach. Learn. Res., 3 :1157–1182, 2003. ISSN 1533-7928.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. Machine Learning, 46(1-3) :389–422, 2002. URL `citeseer.ist.psu.edu/guyon02gene.html`.

Stoppiglia Hervé, Gérard Dreyfus, Rémi Dubois, and Yacine Oussar. Ranking a random feature for variable and feature selection. J. Mach. Learn. Res., 3 :1399–1414, 2003. ISSN 1533-7928.

T. Hogg, B.A. Huberman, and C.P. Williams (Eds). Artificial intelligence : Special issue on frontiers in problem solving : Phase transitions and complexity. volume 81(1-2), 1996.

A. Hyvarinen. Survey on independent component analysis. Neural Computing Surveys, 2 :94–128, 1999.

A. Hyvarinen and E. Oja. Independent component analysis : Algorithms and applications. Neural Networks, 13 :411–430, 2000.

A. Hyvarinen, J. Karhunen, and E. Oja. Independent Component Analysis. John Wiley and Sons, 2001.

A. Hyvärinen, J. Karhunen, and E. Oja. Independent Component Analysis. 2001.

George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In International Conference on Machine Learning, pages 121–129, 1994. Journal version in AIJ, available at http ://citeseer.nj.nec.com/13663.html.

Donald L. Keefer. Certainty equivalents for three-point discrete-distribution approximations. Manage. Sci., 40(6) :760–773, 1994. ISSN 0025-1909.

J. Kiefer. On large deviations of the empiric d.f. of vector chance variables and a law of the iterated logarithm. In Pacific J. Math., volume 11, pages 649–660, 1961.

Shuhei Kimura and Koki Matsumura. Genetic algorithms using low-discrepancy sequences. In Proceedings of the 2005 conference on Genetic and evolutionary computation Washington DC, USA, 2005.

Igor Kononenko. Estimating attributes : Analysis and extensions of RELIEF. In European Conference on Machine Learning, pages 171–182, citeseer.nj.nec.com/kononenko94estimating.html, 1994.

M. Ledoux. The concentration of measure phenomenon. In AMS, 2001.

Michel Ledoux. Inégalités isopérimétriques et calcul stochastique. In Séminaire de probabilités de Strasbourg, 22, pages 249–259, 1988.

D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 401 :788–791, 1999.

K. Levchenko. Notes from a lecture of van vu at university of california, san diego. In http ://www.cs.ucsd.edu/ klevchen/techniques/index.html, 2005.

Ming Li and Paul M. B. Vitanyi. An Introduction to Kolmogorov Complexity and Its Applications. Springer-Verlag, Berlin, 1993. URL `citeseer.ist.psu.edu/li97introduction.html`.

C.X. Ling, J. Hunag, and H. Zhang. AUC : a better measure than accuracy in comparing learning algorithms. 2003.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. In IEEE Symposium on Foundations of Computer Science, pages 256–261, 1989. URL `citeseer.ist.psu.edu/littlestone92weighted.html`.

lots of peoples. Wxwindows librairy. URL `http://www.wxwindows.org/`.

Philippe Marc. Analyse bio-informatique des réseaux de régulation transcriptionnels de la levure Saccharomyces cerevisae grâce aux puces à ADN. PhD thesis, ENS Ulm, 2001.

K. Marton. Bounding d-distance by informational divergence : a method to prove measure concentration. In Annals of Probability, volume 24, pages 857–866, 1996.

Jiri Matousek. Lectures on Discrete Geometry. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002. ISBN 0387953744.

G. Mercier, N. Berthault, J. Mary, J. Peyre, A. Antoniadis, J.-P. Comet, Ch. Froidevaux A. Cornuéjols, and M. Dutreix. Biological detection of low radiation by combining results of two microarray analysis methods. page 32(1) :e12, 2004.

Annie Millet. Méthodes de monte-carlo.

Douglas Montgomery. Design and analysis of experiments 5th edition. 2001.

Rémi Munos. Geometric variance reduction in markov chains. application to value function and gradient estimation. In AAAI, 2005.

H. Niederreiter. Random number generation and quasi-monte-carlo methodes. In Society of Industrial and Applied Mathematics, 1992.

A.B. Owen. Multidimensional variation for quasi-monte carlo. 2004.

M. S. Pepe, G. Longton, G. L. Anderson, and M. Schummer. Selecting differentially expressed genes from microarray experiments. In Biometrics, pages 59 :133–142, 2003.

G.C. Pflug. Scenario tree generation for multi-period financial optimization by optimal discretization. In Mathematical Programming, pages Series B 89 :251–271, 2001.

W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. Numerical Recipes in Fortran (The Art of Scientific Computing) - Second Edition. Cambridge University Press, 1995.

Quinlan J. R. Induction of decision trees. In Machine learning, 1, pages 81–106, 1986.

G. Ratsch. Sparse ensemble learning. PhD thesis, University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany, 2001.

S. Rosset. Model selection via the AUC. In Carla Brodley, editor, 21^{st} International Conference on Machine Learning ICML'04. Morgan Kaufmann, 2004.

Cynthia Rudin, Robert E. Schapire, and Ingrid Daubechies. Boosting based on a smooth margin. In Springer, editor, COLT (Computational Learning Theory), 2004.

Cynthia Rudin, Corinna Cortes, Mehryar Mohri, and Robert E. Schapire. Margin-based ranking meets boosting in the middle. In 18th Annual Conference on Computational Learning Theory, 2005.

Norbert Sauer. On the density of families of sets. J. Comb. Theory, Ser. A, 13(1) :145–147, 1972.

Robert E. Schapire. The strength of weak learnability. In Machine Learning, volume 5 :2, pages 197–227, 1990.

M. Sebag, J. Azé, and N. Lucas. Impact studies and sensitivity analysis in medical data mining with roc-based genetic learning. In Proceedings of IEEE International Conference on Data Mining, ICDM03 IEEE-ICDM03, pages 637–640, 2003.

M. Sebag, J. Azé, and N. Lucas. ROC-based evolutionary learning : Application to medical data mining. In LNCS 2936 Springer Verlag, editor, Artificial Evolution VI, pages 384–396, 2004.

S. Shelah. Stability, the f.c.p, and superstability : model tehoretic properties of formulas in first-order theory. In Annal of Math. Logic, volume 3, pages 271–362, 1971.

M. Talagrand. Concentration of measure and isoperimetric inequalities in product space, 1995. URL `citeseer.ist.psu.edu/talagrand95concentration.html`.

M. Talagrand. A new look at independence. In Annals of Probability, volume 24, pages 1–34, 1996.

Thiedmard. An algorithm to compute bounds for the star discrepancy. In Journal of Complexity 17, 4, pages 850–880, 2001.

Robert Tibshirani, Guenther Walther, David Botstein, and Patrick Brown. Cluster validation by prediction strength. citeseer.nj.nec.com/tibshirani01cluster.html.

Y. Tille. Théorie des sondages, echantillonnage et estimation en population finie. 2001.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. In Pat Langley, editor, Proceedings of ICML-00, 17th International Conference on Machine Learning, pages 999–1006, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US. URL `citeseer.ist.psu.edu/article/tong01support.html`.

Virginia Goss Tusher, Robert Tibshirani, and Gilbert Chu. Significance analysis of microarrays applied to the ionizing radiation response. PNAS, 98(9) :5116–5124, 2001. Stanford University.

A. Van Der Vaart and J.A. Wellner. Weak Convergence and Empirical Processes. 1996.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In Theory of Prob. Its Applic, volume 16, pages 264–280, 1971.

V. N. Vapnik. The Nature of Statistical Learning. Springer-Verlag, 1995.

M. Vidyasagar. A theory of learning and generalization, second edition. 2002.

David H. Wolpert and William G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe, NM, 1995. URL `citeseer.ist.psu.edu/wolpert95no.html`.

Eric P. Xing, Michael I. Jordan, and Richard M. Karp. Feature selection for high-dimensional genomic microarray data. In Proc. 18th International Conf. on Machine Learning, pages 601–608, citeseer.nj.nec.com/xing01feature.html, 2001. Morgan Kaufmann, San Francisco, CA.

L. Yan, R. H. Dodier, M. Mozer, and R. H. Wolniewicz. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In Morgan Kaufmann, editor, ICML'03, the 20^{th} International Conference on Machine Learning, pages 848–855, 2003.

C. Zuily and H. Queffélec. Eléments d'analyse pour l'agrégation. 1995.