Université Paris XIII – Paris-Nord
U.M.R. 7539 — Laboratoire Analyse, Géométrie et Applications

Thèse présentée pour obtenir le grade de
Docteur de l'Université Paris XIII

Spécialité : Mathématiques Appliquées

par

Gilles Blanchard

Méthodes de mélange et d'agrégation d'estimateurs en reconnaissance de formes. Application aux arbres de décision.

Soutenue le 5 Janvier 2001 devant le Jury composé de :

MM.	Olivier CATONI	
	Donald GEMAN	rapporteur
	Pascal MASSART	
	Francesco RUSSO	
	Alain TROUVÉ	directeur de thèse

Rapporteur : Gábor LUGOSI

Remerciements

Alain Trouvé et Olivier Catoni ont encadré ce travail depuis son commencement. C'est peu dire qu'il n'aurait pas vu le jour sans leur présence, leur soutien, leurs idées : la première moitié de ma thèse est entièrement fondée sur les travaux d'Olivier ; de nombreux développements importants m'ont été suggérés par Alain. Mais je tiens ici à les remercier en particulier pour la gentillesse, la diponibilité et la patience dont ils ont fait preuve envers moi : ce que j'ai appris pendant ces années de thèse, et qui dépasse largement le cadre de la seule connaissance scientifique, c'est à eux que je le dois.

Je tiens à témoigner de ma plus grande gratitude à Yali Amit pour son invitation au département de statistiques de l'Université de Chicago, où j'ai passé quinze mois en 1998 et 1999. Cette expérience m'a apporté beaucoup, tant pour mes travaux (la deuxième partie de cette thèse en témoigne) que d'un point de vue personnel. Je considère cette invitation comme une chance exceptionnelle qui m'a été offerte.

Je veux remercier ici Gábor Lugosi et Donald Geman d'avoir accepté d'être rapporteurs de ma thèse ; je tiens de plus à remercier Donald Geman pour de nombreuses discussions avec Yali Amit et moi-même, qui ont amené une plus grande rigueur dans notre travail.

Merci à Francesco Russo et Pascal Massart d'avoir bien voulu participer au jury de cette thèse ; que Pascal trouve également ici l'expression de ma reconnaissance pour la bienveillance et le soutien qu'il m'a régulièrement témoignés.

La liste des personnes qui m'ont apporté leur indipensable amitié et fait partager leurs connaissances sur des sujets variés durant ce long parcours est trop étendue pour que je puisse l'énumérer sans être sûr d'oublier quelqu'un. Je tiens par-dessus tout à remercier Yannick Baraud, Ken Wilder, Massimo Mascaro et Jean-Philippe Vert. J'espère que tous les autres savent qu'ils ont également une place ici.

Je remercie mes parents et ma sœur Ariane pour leur soutien ; enfin Anne pour ses efforts de relecture et surtout sa patience à mes côtés durant cette dernière année : son sourire a effacé bien des périodes de découragement.

Foreword

The present dissertation is a collection of somewhat heterogeneous material (published, submitted or intended for publication), more precisely:

- Chapter 2 contains the text of the published paper [14] (see also an abriged published version [13]) ;
- Chapter 3 contains the text of submitted material ;
- Chapter 4 is an importantly revised version of the technical report [1], a collaboration with Pr. Yali Amit ;
- Chapter 5 has been written specifically as a chapter of the present work.

To present these different pieces inside a common context, I decided to write an extensive chapter of introduction (in french). It is intended as a concise presentation of the general background in which the present work takes its place. I also hope it will help making clear the connexion between the different parts.

Table des matières

Chapitre 1

Introduction

1.1 Classification et apprentissage

1.1.1 Présentation élémentaire du problème

Un problème de classification peut être décrit abstraitement de la façon suivante: on présente à quelqu'un un objet X sur lequel il peut faire les mesures de son choix. À partir de ces mesures, son but est de classer cet objet parmi un nombre fini de catégories numérotées de 0 à c. On suppose que l'objet X appartient *a priori* à l'une de ces classes, mais que celle-ci est inconnue à l'observateur: on peut imaginer qu'un arbitre omniscient tire au hasard dans un sac un objet X, le présente à l'observateur et note si celui-ci commet ou non une erreur en le rangeant dans une catégorie. Le but de l'observateur est donc d'avoir, en moyenne, l'erreur la plus faible possible.

Si l'observateur ne sait rien de la nature des objets et des classes qu'on va lui présenter, il ne pourra jamais faire mieux que de deviner la classe au hasard. On suppose donc que l'arbitre lui a donné avant l'épreuve un certain nombre d'objets également tirés au hasard et déjà étiquetés, qu'on appelle un *échantillon d'apprentissage* (ou *training sample* en anglais). Notre observateur doit donc tirer parti de cette information pour avoir un jugement plus précis sur l'objet de classe inconnue qui lui sera présenté par la suite: en ce sens, on lui a fait «apprendre» comment résoudre le problème de classification[1].

1. Ce cadre, dans lequel l'échantillon d'apprentissage est étiqueté, est appelé aussi *apprentissage supervisé*; mais on peut envisager l'*apprentissage non supervisé*, dans lequel l'échantillon d'apprentissage n'est pas étiqueté et où c'est alors à l'observateur lui-même de créer une taxino-

Le but est donc de tirer parti au mieux des informations apportées par l'échantillon d'apprentissage pour, ensuite, commettre le moins possible d'erreurs sur la classification d'objets inconnus. De plus, on veut pouvoir automatiser cet apprentissage sous forme d'un algorithme, afin de faire accomplir cette tâche par un ordinateur.

Ce cadre général peut s'appliquer à nombre de situations très différentes, comme par exemple:

- la reconnaissance et l'identification de caractères manuscrits;
- l'aide au diagnostic médical;
- la reconnaissance de parole ...

Évidemment, il est illusoire de penser qu'un algorithme développé dans le cadre abstrait donné ci-dessus, pourra résoudre chacun de ces problèmes concrets directement. La façon dont les objets sont représentés dans l'ordinateur, autrement dit la manière dont l'objet est observé, est dépendante du problème pratique considéré et décidée par un opérateur humain. D'une façon générale, on dispose d'outils spécifiques qui permettent de faire sur l'objet X des mesures (ce qui correspond à une étape d'«extraction de caractéristiques», ou *feature extraction* en anglais). Dans ce travail, il ne sera pas question de cette étape, qui est indispensable pour toute application à un problème réel; mais il va de soi que l'extraction de caractéristiques pertinentes est souvent cruciale pour le succès pratique des algorithmes: ainsi, dans l'analyse de signaux sonores, l'analyse de Fourier et la décomposition en ondelettes sont des outils indispensables qui donnent une représentation beaucoup plus pertinente des signaux que, par exemple, la simple observation de l'amplitude au cours du temps[2]. Citons également le travail de Amit et Geman [2, 4] sur la reconnaissance de caractères manuscrits, dont le succès est dû autant à l'algorithme de classification utilisé (les arbres de décision multiples, sur lesquels nous reviendrons au chapitre 4) qu'au choix original des caractéristiques géométriques extraites des images.

Ici, nous supposerons donc l'étape d'extraction de caractéristiques déjà accomplie en amont, et nous chercherons comment faire le meilleur apprentissage possible avec les caractéristiques dont on dispose.

---

mie. Il ne sera pas évoqué dans ce travail.

2. voir par exemple [40] pour des travaux récents sur le sujet et sur la classification active de signaux.

1.1.2 Cadre mathématique

Présentation

En toute généralité, le problème de la classification se formalise de la façon suivante: on considère un couple de variables aléatoires $Z = (X, Y)$ où $X \in \mathcal{X}$ est l'objet observé, $Y \in \mathcal{Y} = \{0, \ldots, c\}$ représente sa classe, et (X, Y) suit la loi P. L'échantillon d'apprentissage est un tirage i.i.d. de N variables $Z_1^N = Z_1, \ldots, Z_N$ tirés selon la loi P.

Le résultat de la procédure d'apprentissage sera une fonction $\widehat{f} : \mathcal{X} \to \{0, \ldots c\}$, dépendant de Z_1^N. À défaut d'un meilleur équivalent français de l'anglais *classifier*, nous appellerons une telle fonction un *classifieur*.

Nous appellerons erreur d'apprentissage (en anglais *training error*) l'erreur faite par $\widehat{f}$ sur l'échantillon d'apprentissage, soit $\sum_{i=1}^{N} \mathbf{1}\{\widehat{f}(X_i) \neq Y_i\}$, et erreur de généralisation l'erreur moyenne de $\widehat{f}$ sur un nouvel objet tiré indépendemment, soit $L_P(\widehat{f}) = E[\mathbf{1}\{\widehat{f}(X) \neq Y\}]$ (le P en indice explicite dans la notation L_P, qui pourra par la suite être omis s'il n'y a pas d'ambiguïté, sert à rappeler que la perte dépend de la probabilité inconnue P). Lorsque nous voudrons valider la procédure d'apprentissage de manière expérimentale, cette quantité pourra être estimée empiriquement sur *échantillon de test*, c'est-à-dire un ensemble d'objets tirés indépendamment et non utilisés dans l'apprentissage.

Classification et régression

Si l'on considère que la classe Y est entièrement déterminée par l'objet X sous la forme $Y = f^*(X)$, la fonction $\widehat{f}$ est un *estimateur* de la «vraie» fonction de classification f^* sous-jacente. On peut élargir ce cadre au cas où la classe Y ne dépend de X que partiellement, et dépend aussi d'un facteur aléatoire indépendant: il est alors naturel de prendre en considération la loi conditionnelle de Y sachant X, déterminée par les distributions conditionnelles $s_P(x, i) = P(Y = i|X = x), i = 0, \ldots, c$.

Si celles-ci étaient connues de façon exacte, on pourrait construire le classifieur optimal f_P (appelé «classifieur de Bayes»), celui qui minimise l'erreur de généralisation $L(f)$: il associe à tout x la classe Y de plus grande probabilité conditionnellement à $X = x$:

$$f_P(x) = \operatorname*{Argmax}_i P(Y = i|X = x).$$

Comme dans le cas précédent, on peut alors considérer $\widehat{f}$ comme un estimateur de f_P. On peut aussi envisager une façon détournée de résoudre ce problème: construire d'abord un estimateur $\widehat{s}$ de $s_P(x, y)$, puis en déduire un classifieur par la même règle que ci-dessus appliquée à $\widehat{s}$. L'estimateur $\widehat{s}$ peut alors être replacé dans des cadres statistiques plus classiques, celui de l'estimation de distribution, ou bien celui de la régression (en effet, en définissant le vecteur aléatoire $\widetilde{Y}$ par $\widetilde{Y}_i = \mathbf{1}\{Y = i\}$, on a $s_P(x, \cdot) = E[\widetilde{Y}|X = x]$).

On peut objecter qu'il semble inutile, voire dommageable, de tenter de résoudre un problème apparemment plus complexe pour ensuite revenir au problème de départ. Mais cette démarche offre la possibilité de se servir d'outils adaptés à la régression ou à l'estimation de densité. D'autre part, une estimation de s_P peut être souhaitable en pratique. Nous avons jusqu'ici considéré comme seule mesure de performance l'erreur de généralisation $L(f)$: cela revient à traiter toutes les classes de façon équivalente. Dans certains cas, on peut préférer qu'une erreur de classification pour certaines classes soit plus pénalisante que pour d'autres: ainsi, pour un diagnostic médical, il semble prioritaire d'éviter de classer comme bien portante une personne malade, l'erreur inverse étant moins grave *a priori*. Si, par exemple, on considère une fonction de perte pondérée selon la classe, de la forme $\widetilde{L}(f) = E[w(Y)\mathbf{1}\{f(X) \neq Y\}]$, alors le classifieur de perte minimale est donné par $f_{P,w}(x) = \operatorname{Argmax}_i w(i)P(Y = i|X = x)$: ainsi, avoir à sa disposition un estimateur de $P(Y|X)$ permettra de s'adapter de façon plus souple à différentes formes de pondération possibles.

1.1.3 Validation de l'apprentissage et fonctions de perte

Différentes mesures de performance

Dans la formulation du problème de classification, nous avons vu que le but premier était de réduire $L_P(\widehat{f}) = E[\mathbf{1}\{\widehat{f}(X) \neq Y\}]$, l'erreur moyenne du classifieur construit à partir des exemples d'apprentissage, lorsqu'on l'applique à un nouvel exemple inconnu. Nous appellerons *fonction de perte* un tel critère quantitatif servant à mesurer la performance de l'estimateur, c'est-à-dire, par définition, une fonction L_P de l'ensemble des fonctions d'intérêt dans $\mathbb{R}$ qui admet un minimum strict $L_P^* = L_P(f_P)$ atteint de manière unique pour la fonction-cible à estimer f_P.

Il est usuel qu'une fonction de perte s'écrive de cette manière, c'est-à-dire sous la forme de l'espérance par rapport à la vraie distribution (inconnue en pratique) d'une fonction $\alpha(\widehat{f}, Z)$ appelée *contraste*, qui ne dépend pas de la vraie distribution P (par exemple, ci-dessus $\alpha(\widehat{f}, Z) = \mathbf{1}\{\widehat{f}(X) \neq Y\}$).

Pour un problème donné, on cherche donc à obtenir un contrôle de la perte $L_P(\widehat{f})$ associée à un estimateur $\widehat{f}$. Cependant, au lieu d'avoir une majoration directement sur la perte, on peut souhaiter contrôler plutôt le défaut de performance de $\widehat{f}$ par rapport à l'optimum f_P. Nous définissons ainsi une autre quantité, le *risque*[3] de l'estimateur $\widehat{f}$, comme la quantité $R_P(\widehat{f}) = L_P(\widehat{f}) - L_P^* = L_P(\widehat{f}) - L_P(f_P)$. Bien que la distinction entre risque et perte puisse apparaître peu pertinente à première vue puisque ces quantités diffèrent seulement de la constante L_P^*, il faut souligner que L_P^* est inconnue *a priori*. De fait, les contrôles obtenus en termes de perte ou de risque pourront être de nature différente.

Au paragraphe précédent, nous avons vu d'une part qu'il pouvait y avoir d'autres fonctions de perte (associés à d'autres contrastes) envisageables pour mesurer la performance d'un classifieur, d'autre part qu'on pouvait résoudre indirectement le problème de classification en estimant les lois conditionnelles $s_P(x, i) = P(Y = i|X = x)$. En statistiques, la performance des estimateurs de lois ou de densités (conditionnelles ou non) est également mesurée à l'aide de fonctions de perte et de risques classiques.

Parmi eux, nous nous intéresserons de façon privilégiée[4] au risque de Kullback-Leibler (également appelé entropie relative, ou redondance), qui mesure un écart entre deux distributions P et Q, et qui est défini par

$$K(P, Q) = \begin{cases} \displaystyle\int \log \frac{dP}{dQ} dP & \text{si } P \ll Q\ ; \\ +\infty & \text{sinon.} \end{cases}$$

De fait, si nous construisons un estimateur $\widehat{s}$ de la fonction $s_P(x, i) = P(Y = i|X)$, le risque que nous voudrons minimiser sera $D(s_P, \widehat{s}) = E[K(s_P(X, \cdot), \widehat{s}(X, \cdot))]$, où l'espérance est prise sur la variable X, et le risque de Kullback-Leibler est mesuré à X fixé entre les deux vecteurs $s_P(X, \cdot)$ et $\widehat{s}(X, \cdot)$ en tant que distributions sur $\{0, \ldots, c\}$. Comme cet ensemble est fini, on peut écrire ce risque plus explicitement

3. La terminologie *contraste, fonction de perte, risque* varie suivant les auteurs. Nous nous tiendrons aux définitions données dans ce paragraphe.

4. Cf. infra 1.2.2

sous la forme

$$\begin{aligned} D(s_P, \widehat{s}) &= E\left[\sum_{i=0}^{c} s_P(X,i) \log \frac{s_P(X,i)}{\widehat{s}(X,i)}\right] \\ &= E\left[-\log \widehat{s}(X,Y)\right] - E\left[-\log s_P(X,Y)\right] \\ &= E[\beta(\widehat{s}, Z)] - E[\beta(s_P, Z)], \end{aligned}$$

en définissant le contraste $\beta(s,Z) = -\log s(X,Y)$. Ceci justifie donc l'appellation de risque donné a la quantité $D(s,\widehat{s})$: il est associé à la fonction de perte $H(\widehat{s}) = E[-\log(\widehat{s}, Z)]$.

Inégalités entre pertes et entre risques

En résumé, la fonction de perte naturelle pour mesurer les performances d'un classifieur $\widehat{f}$ est l'erreur de généralisation $L(\widehat{f})$ associée au contraste $\alpha(\widehat{f}, Z) = \mathbf{1}\{\widehat{f}(X) \neq Y\}$; d'autre part, si nous construisons indirectement un classifieur en estimant d'abord les probabilités conditionnelles $s_P(x,i)$, nous serons amenés à employer un risque qu'on retrouve habituellement dans ce cadre, le risque de Kullback-Leibler associé au contraste $\beta(s,Z) = -\log s(X,Y)$.

Toutefois, comme le but final est la classification, il est important, en dernière analyse, de savoir contrôler l'erreur de généralisation. En particulier, si nous savons contrôler le risque de Kullback-Leibler pour une procédure d'estimation donnée, que pouvons-nous en déduire sur le risque du classifieur associé? La proposition suivante résume un certain nombre d'inégalités élémentaires permettant de se ramener à un contrôle de l'erreur de généralisation.

Proposition 1. *Soit $Z = (X,Y)$ une variable aléatoire sur $\mathcal{X} \times \mathcal{Y}$ suivant la loi P. On note $s_P(x,i) = P(Y = i|X = x)$ et $f_P(x) = \mathrm{Argmax}_i\, s(x,i)$.*

Soit $\widehat{s} : \mathcal{X} \times \{0, \ldots, c\} \to [0,1]$ tel que $\forall x \in \mathcal{X}, \sum_i s(x,i) = 1$ et $\widehat{f} : \mathcal{X} \to \{0, \ldots, c\}$. On pose

$$\left\{\begin{array}{rcl} L(\widehat{f}) &=& E[\mathbf{1}\{\widehat{f}(X) \neq Y\}] \\ L^* &=& \min_{\widehat{f}} L(\widehat{f}) = L(f_P) \\ R_P(\widehat{f}) &=& L(\widehat{f}) - L^* \end{array}\right. \quad \textit{et} \quad \left\{\begin{array}{rcl} H(\widehat{s}) &=& E[-\log(\widehat{s}(X,Y))] \\ H^* &=& \min_{\widehat{s}} H(\widehat{s}) = H(s_P) \\ D(s_P, \widehat{s}) &=& H(\widehat{s}) - H^* \end{array}\right.$$

On a alors les inégalités suivantes:

Inégalité entre pertes minimum:

$$\frac{H^* - \log(2)}{c} \leq L^* \leq H^*; \tag{1.1}$$

Inégalité entre risques:

pour $\widehat{f} = \mathrm{Argmax}_i\, \widehat{s}(X,i)$, *on a*

$$R_P(\widehat{f}) \leq \sqrt{2D(s_P, \widehat{s})}; \tag{1.2}$$

Inégalité entre pertes:

on peut construire à partir de $\widehat{s}$ *un classifieur randomisé* $\widetilde{f}$ *tel que*

$$L(\widetilde{f}) \leq H(\widehat{s}). \tag{1.3}$$

Preuve. Il suffit en fait de raisonner à X fixé, et de prouver les inégalités pour une distribution donnée de la variable Y, puis de prendre l'espérance par rapport à X; pour l'inégalité (1.2) l'inégalité de Jensen permet de sortir la racine carrée hors de l'espérance. Notons que pour la même raison, ces inégalités seront également valables en moyenne sur le tirage des exemples d'apprentissage Z_1^N.

Soit donc une distribution de la variable $Y \in \{0, \ldots, c\}$, définie par $p_i = P(Y = i)$; supposons de plus que $\max_i p_i = p_0$.

La première partie de (1.1) se déduit du lemme de Fano, voir par exemple [28],p. 39. On a d'autre part

$$L^* = 1 - p_0 \leq -\log p_0 \leq -\sum_i p_i \log p_0 \leq -\sum_i p_i \log p_i = H^*$$

ce qui achève de prouver (1.1).

Pour l'inégalité suivante, on a

$$R_P(\widehat{f}) = L(\widehat{f}) - L^* \leq \sum_{i=0}^{c} |p_i - \widehat{s}(i)| \leq \sqrt{2D(s_P, \widehat{s})}$$

(ces deux inégalités, assez élémentaires, peuvent être trouvées par exemple dans [32] pour la première et [28],p. 300 pour la seconde.)

Pour la dernière inégalité, au lieu de construire le classifieur $\widehat{f}$ à partir de $\widehat{s}$ en prenant la classe de probabilité maximum comme auparavant, on construit un classifieur $\widetilde{f}$ randomisé en tirant au hasard et indépendamment une classe selon la loi donnée par $\widehat{s}$. On a alors

$$L(\widetilde{f}) = P(\widetilde{f} \neq Y) = \sum_{i=0}^{c} p_i(1 - \widehat{s}(i)) \leq -\sum_{i=0}^{c} p_i \log \widehat{s}(i) = H(\widehat{s}).$$

□

Les inégalités (1.2) et (1.3) nous permettent donc d'obtenir, pour les classifieurs $\widehat{f}$ et $\widetilde{f}$ construits à partir de l'estimateur $\widehat{s}$, des majorations sur l'erreur de généralisation à partir du risque de Kullback de $\widehat{s}$ ou de la fonction de perte correspondante.

Dans la suite, la notation $\widehat{f}$ désignera indifféremment un estimateur de classification ou de distribution conditionnelle, sauf mention explicite.

1.1.4 Modèles et complexité des modèles

Exemple sur les arbres de décision

D'une façon générale, pour construire un estimateur $\widehat{f}$, on fixe *a priori* un certain ensemble $\mathcal{F} \subset \{0, \dots, c\}^{\mathcal{X}}$ (resp. $\mathcal{S} \subset \{s : \mathcal{X} \times \mathcal{Y} \to [0,1] \mid \forall x \in \mathcal{X} \ \sum_{i=0}^{c} s(x,i) = 1\}$ pour l'estimation de distributions conditionnelles) dans lequel on va sélectionner en appliquant une procédure statistique un élément $\widehat{f}$. Un tel ensemble est appelé un *modèle*.

Pour donner un exemple, nous allons sans plus attendre définir la classe de modèles qui constituera l'application privilégiée de nos résultats: les arbres de décision. Ceux-ci ont été introduits et popularisés dans les années 80, notamment par Breiman, Ohlsen et Stone sous l'acronyme CART (Classification And Regression Trees, [21]). Un arbre de décision A est défini par un triplet $A = (T, K, \Theta)$, avec:

- T une structure d'arbre binaire complet. Un arbre binaire est défini comme un sous-ensemble stable par préfixe des chaînes de bits de longueur finie $\emptyset \cup \{0,1\}^*$. Chaque chaîne représente un nœud de l'arbre binaire: si t est la chaîne représentant un nœud, les chaînes $t0$ et $t1$ représentent ses fils droit et gauche. Un binaire arbre *complet* a de surcroît la propriété suivante: tout nœud non terminal possède exactement deux nœuds-fils; ce qui se traduit ainsi: pour toute chaîne non vide $t \in T$, la chaîne obtenue en inversant le dernier bit de t est aussi dans T. Enfin, on distingue l'ensemble des nœuds intérieurs, notés $\operatorname{int}(T)$, et l'ensemble des nœuds terminaux, noté ∂T;

- $K : \operatorname{int}(T) \times \mathcal{X} \to \{0,1\}$; pour $t \in \operatorname{int}(T)$ on appelle $K(t, \cdot)$ la question attachée au nœud t;

- $\Theta : \partial T \to F$, où $F = \{0, \dots, c\}$ pour les arbres de classification, et $F = \mathcal{S}_c$ pour les arbres de régression, où $\mathcal{S}_c$ est le simplexe de dimension c, $\mathcal{S}_c = \{\theta \in$

$[0,1]^{c+1} | \sum_i \theta_i = 1\}$ (c'est-à-dire l'ensemble des répartitions de probabilité sur $\mathcal{Y} = \{0, \ldots, c\}$).

Un arbre de décision fonctionne comme suit: pour un élément $X \in \mathcal{X}$ donné, on commence par poser la question attachée à la racine $K(\emptyset, X)$; si la réponse est 0, on fait descendre X au nœud droit de la racine, au nœud gauche sinon, et on réitère l'opération en posant la question attachée au nœud où X se trouve et en le faisant progresser dans l'arbre selon la réponse, jusqu'à atteindre un nœud terminal t. On définit alors $A(X) = \Theta(t)$, qui est soit une classe, soit une distribution de probabilité sur les classes, selon le cas choisi (construction d'un classifieur ou estimation de probabilité conditionnelle).

Dans ce cadre, on a une famille de modèles naturels paramétrée par $m = (T, K)$, donnés par $\mathcal{A}_m = \{A = (T, K, \Theta) \,|\, \Theta : \partial T \to F\}$. Cela correspond à fixer la structure d'arbre T et les questions K et à considérer toutes les façons possibles d'étiqueter les feuilles.

La décomposition du risque en fonction du modèle

Lorsqu'on a fixé un modèle $\mathcal{F}$, on construit l'estimateur $\widehat{f} \in \mathcal{F}$ à partir des exemples d'apprentissage en utilisant une certaine procédure statistique. Si l'on mesure la performance de l'estimateur à l'aide d'une fonction de perte L_P, qui atteint son minimum sur la fonction-cible f_P, on peut décomposer le risque $R_P(\widehat{f}) = L_P(\widehat{f}) - L_P(f_P)$ en la somme de deux termes distincts:

- le plus petit risque possible au sein du modèle, c'est-à-dire $\inf_{f \in \mathcal{F}} R_P(f)$, supposé atteint pour une certaine fonction $\bar{f}_P$, la «projection» de f_P sur le modèle $\mathcal{F}$;

- le «défaut de performance» $\Delta_P(\bar{f}_P, \widehat{f}) = R_P(\widehat{f}) - R_P(\bar{f}_P) = L_P(\widehat{f}) - L_P(\bar{f}_P)$.

Cette décomposition est parfois appelée «décomposition biais-variance» par référence au cadre statistique de régression classique où R est le risque L^2, $\mathcal{F}$ est un espace affine et $\widehat{f}$ est une projection orthogonale sur $\mathcal{F}$ des observations. Dans ce cas $\bar{f}_P$ est la valeur moyenne (relativement au tirage de Z_1^N) de l'estimateur $\widehat{f}$: la moyenne du deuxième terme relativement au tirage de Z_1^N s'identifie alors à la variance de l'estimateur $\widehat{f}$.

Le premier terme mesure une «distance» entre la fonction-cible et le modèle choisi, distance qui ne peut être réduite à moins de changer de modèle; le second mesure le défaut de performance entre l'estimateur $\widehat{f}$ et la meilleure fonction possible $\bar{f}_P$ dans le modèle.

Le premier terme est *indépendant* du tirage de l'échantillon d'apprentissage Z_1^N, et de la procédure d'estimation choisie au sein du modèle, alors que ce n'est pas le cas du second.

Il faut noter que le deuxième terme Δ_P ne peut pas, en général, s'exprimer comme un risque au sens ou nous l'avons défini. Toutefois, dans le cas du risque de Kullback-Leibler D, et pour un modèle d'arbre de décision $\mathcal{A}_m$, on peut montrer facilement que, pour $\bar{f}_m = \text{Argmin}_{f_m \in \mathcal{A}_m} D(f_P, f_m)$,

$$\Delta_P(\bar{f}_m, \widehat{f}) = D(f_P, \widehat{f}) - D(f_P, \bar{f}_m) = D(\bar{f}_m, \widehat{f}). \tag{1.4}$$

Estimation et complexité des modèles

En général, comme dans le cas des arbres de décision exposé plus haut, le statisticien dispose d'un grand nombre de modèles parmi lesquels il doit choisir. Un modèle peut être plus ou moins riche: par exemple, s'il s'agit d'un espace vectoriel, plus sa dimension est grande et plus le modèle est riche. Dans le cas des arbres de décision, il semble intuitivement clair que plus la structure d'arbre T a de feuilles, plus le modèle est riche (un arbre plus grand peut en effet distinguer plus finement entre les objets X qu'un arbre ayant moins de feuilles). En contrepartie, plus un modèle est riche et plus il y aura de risques d'erreur d'estimation: dans le cas des arbres de décision pour l'estimation de loi conditionnelle, le nombre de paramètres à estimer varie comme $|\partial T|^c$.

Ainsi, si l'on s'autorise à augmenter indéfiniment la taille de l'arbre de classification, il sera toujours possible de trouver un arbre dont chaque feuille contient au plus un exemple d'apprentissage: cet arbre aura une erreur d'apprentissage nulle, mais l'estimation de la distribution de Y en chaque feuille sera très mauvaise, ce qui donnera lieu le plus souvent à une très mauvaise capacité de généralisation: c'est le phénomène dit d'*overfitting*.

Si l'on reprend la décomposition du paragraphe précédent, on constate donc qu'en augmentant la richesse d'un modèle, on peut espérer diminuer le terme de «biais» $R_P(\bar{f}_P)$, mais on fera sans soute augmenter en même temps le deuxième terme de «variance» $\Delta_P(\bar{f}_P, \widehat{f}) = L_P(\widehat{f}) - L_P(\bar{f}_P)$.

Comment formaliser cette notion de «richesse» (ou de complexité) d'un modèle? On peut s'intéresser au cas particulier suivant: supposons que $f_P \in \mathcal{F}$, ce qui implique que le terme de biais est nul et que le risque est égal au terme de variance.

Pour un estimateur donné $\widehat{f}_N$, cherchons le pire cas possible au sens du risque moyen, c'est-à-dire le maximum de $E_N[R_P(\widehat{f}_N)]$, sur l'ensemble des distributions P telles que $f_P \in \mathcal{F}$.

Il est souhaitable d'avoir une borne inférieure uniforme sur tous les estimateurs possibles, et on est donc amené à définir *risque minimax* $R^N_{\mathcal{F}}$ d'estimation comme suit:

$$R^N_{\mathcal{F}} = \min_{\widehat{f}_N} \max_{P: f_P \in \mathcal{F}} E_N[R_P(\widehat{f}_N)]$$

où E_N désigne l'espérance par rapport au tirage de l'échantillon d'apprentissage Z_1^N, et où $\widehat{f}_N$ désigne un *estimateur*, c'est-à-dire un élément de $\mathcal{F}$ dépendant de Z_1^N. Par définition du risque minimax, pour tout estimateur $\widehat{f}_N$ construit à partir d'un échantillon de taille N, il existe donc P tel que $f_P \in \mathcal{F}$ et $E_N[R_P(\widehat{f}_N)] \geq R^N_{\mathcal{F}}$.

A fortiori lorsque f_P peut ne pas appartenir au modèle, le terme de variance Δ_P sera, pour certaines distributions P, en moyenne sur le tirage de Z_1^N, nécessairement supérieur à $R^N_{\mathcal{F}}$. En particulier, dans le cas des arbres de décision, on a la minoration suivante sur le risque minimax de Kullback-Leibler:

Proposition 2. *Soit $\mathcal{A}_m$ un modèle d'arbre fixé, tel que pour toute feuille $s \in \partial T$, il existe un objet $x \in \mathcal{X}$ qui parvient à la feuille s lorsqu'on le fait descendre dans l'arbre. Soit D^N_m le risque minimax de Kullback-Leibler pour ce modèle, avec un échantillon de taille N. On a alors*

$$\forall N \geq |\partial T| \qquad D^N_m \geq C \frac{|\partial T|}{N}, \tag{1.5}$$

où C est une constante.

On voit que la vitesse de convergence (asymptotique) est plus lente lorsque le modèle $\mathcal{A}_m$ est plus riche, tout en restant de l'ordre de grandeur de $1/N$, qui est la «vitesse paramétrique»[5]. De plus, si l'on définit le modèle $\mathcal{M} = \bigcup \mathcal{A}_m$, un corollaire immédiat de la proposition 2 est que $D^N_{\mathcal{M}}$ ne tend pas vers 0 quand $N \to \infty$: il n'existe pas d'estimateur dont le risque par rapport à tous les éléments de $\mathcal{M}$ tende uniformément vers 0. Ceci explique pourquoi il est plus pertinent de considérer séparément les modèles $\mathcal{A}_m$ plutôt que $\mathcal{M}$.

Preuve de la proposition 2. Cette preuve s'inspire d'une minoration montrée par

5. Il est possible d'obtenir des estimations de vitesses minimax pour des ensembles de fonctions plus généraux, comme des boules dans un espace de Sobolev, où l'on trouve des vitesses de convergence asymptotique d'ordre plus lent que $1/N$, dites «non paramétriques».

Lugosi [47] pour la perte donnée par l'erreur de classification. On se restreint au cas où il y a seulement 2 classes (cela ne change en rien la démonstration).

On pose $L = |\partial T|$; soit $N \geq L$ fixé par la suite. Notons $s_1, \dots, s_L$ les feuilles de l'arbre. Pour $x \in \mathcal{X}$, on note $m(x)$ la feuille à laquelle parvient l'objet x lorsqu'on le fait descendre dans l'arbre. Par hypothèse, il existe $x_1, \dots, x_L$ tels que $m(x_i) = s_i, i = 1, \dots, L$. On définit la distribution P_X sur $\mathcal{X}$ par

$$\begin{cases} P_X(X = x_i) = 1/N & \text{pour } i < L \\ P_X(X = x_L) = 1 - (L-1)/N. \end{cases}$$

Soit $\mathcal{B}$ l'ensemble des mots binaires de $L-1$ bits: pour $B \in \mathcal{B}$, on définit la fonction f_B par $f_B(s_i, 0) = B_i, f_B(s_i, 1) = 1 - B_i$ pour $i < L$ et $f_B(s_L, 0) = 1, f_B(s_L, 1) = 0$. On définit P_B, distribution du couple (X, Y), par $P_B(X = x, Y = y) = P_X(X = x) f_B(m(x), y)$. Si D désigne le risque de Kullback-Leibler, on a, pour tout estimateur $\widehat{f}_N$ de la distribution conditionnelle de Y sachant X:

$$\begin{aligned} \max_{P: f_P \in \mathcal{A}_{T,K}} E_N[D(f_P, \widehat{f}_N)] &\geq \max_{P_B: B \in \mathcal{B}} E_N[D(f_B, \widehat{f}_N)] \\ &\geq E_{\mathcal{B}} E_N E\left[\log\left(\frac{P_B(Y|X)}{\widehat{f}_N(X,Y)}\right)\right] = E_Q\left[\log\left(\frac{P_B(Y|X)}{\widehat{f}_N(X,Y)}\right)\right] \end{aligned}$$

À la deuxième ligne, on a majoré le maximum sur $\mathcal{B}$ par une espérance uniforme sur $\mathcal{B}$, notée $E_{\mathcal{B}}$. On a noté Q la probabilité jointe ainsi obtenue sur (B, Z_1^N, X, Y). Le minimum du membre de droite est atteint pour $\widehat{f}_N = Q(Y|X, Z_1^N)$ (c'est une propriété élémentaire de l'entropie conditionnelle, voir [28]). On voit facilement que

$$\log\left(\frac{P_B(Y|X)}{Q(Y|X, Z_1^N)}\right) = \begin{cases} \log 2 & \text{si } X \neq X_1, \dots, X \neq X_N, X \neq x_L \\ 0 & \text{sinon.} \end{cases}$$

On a donc

$$\begin{aligned} D^N_{\mathcal{A}_{T,K}} &\geq \sum_{i=1}^{L-1} P_X(X = x_i)(1 - P(X = x_i))^N \log(2) \\ &= \frac{K-1}{N} \log(2) \left(1 - \frac{1}{N}\right)^N \geq \frac{(K-1)\log(2)}{Ne}\left(1 - \frac{1}{N}\right) \end{aligned}$$

□

La sélection adaptative de modèles

Le choix du modèle dans lequel va être faite l'estimation s'avère donc crucial. On dispose en général d'une famille de modèles $\mathcal{F}_m, m \in \mathcal{M}$ et, pour chaque modèle, d'un estimateur $\widehat{f}_m$, dont le risque, idéalement, est proche du risque minimax du modèle lorsque $f_P \in \mathcal{F}$. Pour chaque modèle $m \in \mathcal{M}$, le risque minimax R_m^N tend vers 0 quand $N \to \infty$, mais ce n'est pas le cas de $R_{\mathcal{F}_\infty}^N$ où $\mathcal{F}_\infty = \bigcup_{m \in \mathcal{M}} \mathcal{F}_m$: en somme, $\mathcal{F}_\infty$ est trop riche pour constituer en soi un modèle intéressant.

Il s'agit donc de choisir un bon modèle dans la famille, mais pour cela il faudrait connaître à l'avance le modèle réalisant le meilleur «compromis biais/variance»:

$$m^* = \operatorname*{Argmin}_{m \in \mathcal{M}} [R_P(\bar{f}_m) + \Delta_P(\bar{f}_m, \widehat{f}_m)],$$

où $\bar{f}_m = \operatorname{Argmin}_{f_m \in \mathcal{F}_m} R_P(f_m)$.

En l'absence d'un tel «oracle», on peut suggérer de trouver une méthode pour choisir un modèle $\widehat{m}$ à l'aide des données. Une telle méthode de *sélection adaptative* aura une bonne performance si son risque n'est pas considérablement plus grand que celui de l'estimateur-oracle $\widehat{f}_{m^*}$. On cherche donc à obtenir des bornes sur le risque de l'estimateur obtenu $\widehat{f}$ qui aient la forme générale

$$E_N[R_P(\widehat{f})] \leq \inf_{m \in \mathcal{M}} \left(\inf_{f_m \in \mathcal{F}_m} E_N[R_P(f_m)] + \Delta(m, N) \right) \tag{1.6}$$

où $\Delta(m, N)$ est, idéalement, de l'ordre de grandeur du risque minimax R_m^N. C'est à cette forme de majoration «idéale» que nous nous référerons pour comparer les différentes méthodes de construction d'estimateurs présentés dans la partie suivante.

1.2 De la sélection de modèle au mélange de modèles

L'objet de la première partie de ce travail (chapitres 2 et 3) sera l'étude théorique et pratique d'un estimateur introduit par Catoni. Cet estimateur n'utilise pas de méthode de sélection de modèle, mais un principe de «mélange» des modèles. Avant de définir cet estimateur, nous commençons dans ce paragraphe par présenter un aperçu des méthodes de sélection de modèle qui nous serviront ensuite de point de comparaison.

1.2.1 Un survol des méthodes de sélection de modèle

La minimisation empirique du risque

Nous n'avons pas jusqu'ici précisé de méthode statistique pour choisir un élément $\widehat{f} \in \mathcal{F}$ à partir des données Z_1^N. Lorsque la fonction de perte L peut s'exprimer sous la forme $L_P(\widehat{f}) = E_P[\alpha(\widehat{f}, Z)]$ où α est un contraste, il est naturel de penser à estimer $L_P(\widehat{f})$ en remplaçant l'espérance par rapport à la probabilité inconnue P par la moyenne empirique, et de choisir dans le modèle $\mathcal{F}$ la fonction qui minimise ce risque empirique:

$$\widehat{f} = \operatorname*{Argmin}_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \alpha(f, Z_i). \tag{1.7}$$

En pratique, il peut être difficile d'un point de vue algorithmique de trouver le minimum effectif; on peut alors supposer que $\widehat{f}$ réalise ce minimum à ε près.

Notons que pour le cas où l'on a un contraste de type $\alpha(f, Z) = ||Y - f(X)||^2$, cette procédure d'estimation, appelée minimisation empirique du risque, est exactement celle des *moindres carrés*; pour le contraste $\beta(s, Z) = -\log s(X, Y)$ qui correspond au risque de Kullback-Leibler, c'est le *maximum de vraisemblance*. Pour le contraste $\alpha(f, Z) = \mathbf{1}\{f(X) \neq Y\}$, cela revient à choisir dans notre modèle un classifieur $\widehat{f}$ qui fait le moins d'erreurs sur l'échantillon d'apprentissage.

C'est cette méthode spécifique d'estimation qui est prise en considération dans les méthodes de sélection de modèle que nous allons exposer.

Contrôle uniforme des déviations: la théorie de Vapnik et Chervonenkis

La théorie de Vapnik et Chervonenkis [55, 56, 57], dont les bases remontent aux années 70, consiste à essayer de contrôler la déviation entre la perte empirique $E_{\widehat{P}}[\alpha(\widehat{f}, Z)]$ (où $\widehat{P}$ est la distribution empirique) et la perte réelle $E_P[\alpha(\widehat{f}, Z)]$. Comme $\widehat{f}$ peut *a priori* être n'importe quel élément du modèle $\mathcal{F}$, on souhaite obtenir un contrôle uniforme, c'est-à-dire qu'il faut de fait borner la quantité

$$V_{\mathcal{F}} = \sup_{f \in \mathcal{F}} \left| E_{\widehat{P}}[\alpha(f, Z)] - E_P[\alpha(f, Z)] \right|. \tag{1.8}$$

Il semble intuitivement clair que la richesse du modèle $\mathcal{F}$ joue un rôle dans cette quantité puisque plus le modèle est riche, plus il y a de déviations à contrôler. Vapnik et Chervonenkis ont réussi à formaliser cette idée en introduisant une mesure particulière de la richesse d'un modèle, dite *dimension VC* du modèle.

Pour plus de simplicité, nous nous restreindrons ici au cas où le risque est mesuré par l'erreur de généralisation et où le modèle $\mathcal{F}$ est un ensemble de classifieurs, bien que la théorie de Vapnik et Chervonenkis s'applique à des cadres plus généraux. La présentation des résultats ci-dessous s'inspire d'une formulation plus récente [47] que les travaux originaux de Vapnik et Chervonenkis, dans laquelle le contrôle de la quantité $V_{\mathcal{F}}$ se fait en deux étapes:

1. Majoration de l'espérance $E_N[V_{\mathcal{F}}]$: c'est ici qu'intervient la complexité du modèle $\mathcal{F}$;

2. Contrôle des déviations de $V_{\mathcal{F}}$ par rapport à l'espérance en utilisant des inégalités de concentration telles que l'inégalité de Hoeffding ou l'inégalité de McDiarmid pour les fonctionnelles à différences bornées.

Définition 1. *Soit $\Gamma \subset \mathcal{P}(\mathcal{X})$ et x_1^N un ensemble de N points de $\mathcal{X}$. On appelle* coefficient d'éclatement *et notera par la suite Δ^Γ la quantité définie par*

$$\Delta^\Gamma(x_1^N) = \mathrm{card}\left\{S \cap x_1^N | S \in \Gamma\right\}.$$

Théorème 1 (Vapnik, Chervonenkis). *Soit $\mathcal{F}$ un ensemble de fonctions de $\mathcal{X}$ dans $\mathcal{Y} = \{0, \dots, c\}$, on définit l'ensemble*

$$\Gamma = \{\ \{x \in \mathcal{X} | f(x) = y\} | f \in \mathcal{F}, y \in \mathcal{Y}\}$$

On a alors, avec probabilité au moins $1 - \eta$ sur le tirage de l'ensemble d'apprentissage Z_1^N, la majoration suivante pour tout $f \in \mathcal{F}$:

$$E_P[\mathbf{1}\{f(X) \neq Y\}] \leq E_{\widehat{P}}[\mathbf{1}\{f(X) \neq Y\}] + 2\sqrt{\frac{\log\left(2\Delta^\Gamma(X_1^N)\right)}{N}} + \sqrt{\frac{-\log(\eta)}{2N}} \tag{1.9}$$

De plus, le coefficient d'éclatement Δ^Γ jouit de la propriété suivante:

Proposition 3 (Lemme de Sauer). *On note*

$$G^\Gamma(N) = \log \max_{x_1^N} \Delta^\Gamma(x_1^N);$$

la fonction $G^\Gamma(N)$ est ou bien identiquement égale à $N\log 2$ ou bien majorée pour tout $N \geq h$ suivant

$$G^\Gamma(N) \leq h\log(N+1)$$

où $h+1$ est le premier entier N violant l'égalité $G^\Gamma(N) = N\log 2$. h est appelé dimension VC *(éventuellement infinie) de Γ.*

Ainsi, le terme $\log(2\Delta^\Gamma(X_1^N))$ dans le membre droit de l'inégalité (1.9) peut être majoré par $h\log(N+1)+\log 2$, où h est la dimension VC du modèle.

La majoration (1.9) permet donc d'avoir une «perte maximum garantie» avec probabilité arbitrairement grande, lorsqu'on connait la dimension de VC de l'ensemble Γ. Cette majoration n'est pas asymptotique, mais vraie pour toute valeur de N fixée à l'avance.

Si $h<\infty$, cette borne garantit également la consistance de la méthode de la minimisation de risque empirique, c'est-à-dire que $\sup_{f\in\mathcal{F}}|E_{\widehat{P}}[\alpha(f,Z)]-E_P[\alpha(f,Z)]|\to 0$ en probabilité quand $N\to\infty$.

En particulier, la majoration (1.9) est valable avec probabilité $1-\eta$ pour la fonction $\widehat{f}$ qui minimise la perte empirique.

A partir de ces majorations, Vapnik propose l'algorithme général suivant: supposons qu'on ait à choisir parmi un nombre fini de modèles de classifieurs imbriqués

$$\mathcal{F}_1\subset\mathcal{F}_2\subset\ldots\subset\mathcal{F}_M$$

dont les dimensions VC sont finies et croissantes

$$h_1\le h_2\le\ldots\le h_M<\infty$$

alors l'ensemble des majorations du type de (1.9) obtenues en appliquant le théorème 1 à chacun des ensembles $\mathcal{F}_k$ sont valables simultanément pour tous les $k\le M$ avec une probabilité $1-\eta M$, ceci résultant directement du théorème 1. Pour avoir une perte minimum garantie, Vapnik propose de minimiser en k et en f le membre de droite de (1.9) (perte empirique + terme de pénalité). Par cette méthode appelée *Minimisation Structurelle du Risque*, l'un des modèles est donc sélectionné de façon automatique.

En notant L l'erreur de généralisation, $\widetilde{L}_e$ l'erreur empirique pénalisée définie par

$$\widetilde{L}_e(f)=E_{\widehat{P}}[\mathbf{1}\{f(X)\neq Y\}]+2\sqrt{\frac{\log(2\Delta^\Gamma(X_1^N))}{N}},$$

et $\widehat{f}$ l'estimateur obtenu en minimisant $\widetilde{L}_e$, on a alors, $\forall k\le M$, $\forall g_k\in\mathcal{F}_k$,

$$\begin{aligned}E_N[L(\widehat{f})-L(g_k)] &= E_N[L(\widehat{f})-\widetilde{L}_e(\widehat{f})]+E_N[\widetilde{L}_e(\widehat{f})-\widetilde{L}_e(g_k)]+E_N[\widetilde{L}_e(g_k)-L(g_k)]\\ &\le 2\sqrt{\frac{h_k\log(N+1)+\log 2}{N}}+\sqrt{\frac{\log(Me)}{2N}}\end{aligned}$$

(pour majorer le premier terme, on a intégré par rapport à η l'inégalité de type (1.9), uniforme en k et donc valide en particulier dans le modèle sélectionné; le second terme est négatif par construction de $\widehat{f}$; dans le troisième les termes de perte s'annulent par passage à l'espérance et seule la pénalité relative à $\mathcal{F}_k$ reste).

Ceci mène immédiatement à l'inégalité pour les risques

$$E_N[R(\widehat{f})] \leq \inf_{k \leq M} \left(\inf_{g_k \in \mathcal{F}_k} E_N[R(g_k)] + 2\sqrt{\frac{h_k \log(N+1) + \log 2}{N}} \right) + \sqrt{\frac{\log(Me)}{2N}} \tag{1.10}$$

qui est à rapprocher de la forme générale (1.6) recherchée dans le cadre de la sélection de modèle.

La sélection par estimation pénalisée

Comme nous l'avons vu au paragraphe précédent, la méthode proposée par Vapnik revient à faire de la *minimisation pénalisée de la perte empirique*, c'est-à-dire qu'un modèle $m \in \mathcal{M}$ est choisi en utilisant un critère qui comporte un premier terme de perte empirique de l'estimateur $\widehat{f}_m \in \mathcal{F}_m$, et un second terme de *pénalité* $\mathrm{pen}(m)$ dépendant de la richesse du modèle m. Cette méthode de sélection de modèle se retrouve dans de nombreux domaines des statistiques et a été le sujet de beaucoup de recherches récentes. Birgé et Massart ont notamment proposé un cadre général d'étude de l'estimation pénalisée ([10, 11, 6]), que nous allons présenter brièvement. Nous supposons comme auparavant que la perte est mesurée par $L_P(\widehat{f}) = E_P[\alpha(\widehat{f}, Z)]$, et nous noterons la fonction de perte empirique $L_e(\widehat{f}) = E_{\widehat{P}}[\alpha(\widehat{f}, Z)]$. Le classifieur $\widehat{f}$ obtenu par estimation pénalisée minimise[6] en (m, f_m) la quantité $L_e(f_m) + \mathrm{pen}(m)$, avec $f_m \in \mathcal{F}_m$; le résultat de cette procédure sélectionne effectivement un certain modèle $\widehat{m} \in \mathcal{M}$ et un classifieur $\widehat{f} \in \mathcal{F}_{\widehat{m}}$.

Le but est de contrôler le risque $R_P(\widehat{f}) = L_P(\widehat{f}) - L_P(f_P)$, où f_P est la fonction-cible. De l'inégalité définissant $\widehat{f}$,

$$\forall m \in \mathcal{M},\ \forall f_m \in \mathcal{F}_m, \qquad L_e(\widehat{f}) \leq L_e(f_m) + \mathrm{pen}(m) - \mathrm{pen}(\widehat{m})$$

on déduit facilement

$$\forall m \in \mathcal{M},\ \forall f_m \in \mathcal{F}_m,$$
$$R_P(\widehat{f}) \leq R_P(f_m) + \overline{L}_e(f_m) - \overline{L}_e(\widehat{f}) + \mathrm{pen}(m) - \mathrm{pen}(\widehat{m}) \tag{1.11}$$

6. Cette minimisation se fait éventuellement à ε_N près en pratique, mais cela ne change pas la teneur des résultats.

où $\overline{L}_e(\cdot)$ désigne le processus empirique recentré

$$\overline{L}_e(f) = E_{\widehat{P}}[\alpha(f,Z)] - E_P[\alpha(f,Z)].$$

On a vu que la méthode de Vapnik et Chervonenkis consistait à contrôler, pour tout $m \in \mathcal{M}$, la quantité $\overline{L}_e(f_m)$ uniformément sur $f_m \in \mathcal{F}_m$.

Le principe de Birgé et Massart, en partant de l'équation (1.11), est le contrôle, pour $m \in \mathcal{M}, f_m \in \mathcal{F}_m$, fixés, de la quantité $\overline{L}_e(f_m) - \overline{L}_e(f_{m'})$, uniformément pour $m' \in \mathcal{M}, f_{m'} \in \mathcal{F}_{m'}$.

Les étapes de ce contrôle sont, de façon très schématique:

1. Définir un processus convenablement renormalisé

$$V_{m'}(f_m) = \sup_{f_{m'} \in \mathcal{F}_{m'}} \left[\frac{|\overline{L}_e(f_{m'}) - \overline{L}_e(f_m)|}{w_{m'}(f_{m'})} \right],$$

   où la pondération $\omega_{m'}(f_{m'})$ dépend d'une certaine mesure de distance entre la fonction-cible f_P et $f_m, f_{m'}$;

2. Majorer $E_N[V_{m'}(f_m)]$: c'est principalement ici qu'intervient la «complexité» du modèle m', mesurée en utilisant des techniques variées suivant les cas;

3. Contrôler les déviations de $V_{m'}(f_m)$ par rapport à son espérance $E_N[V_{m'}(f_M)]$ en utilisant des inégalités fines de concentration exponentielles démontrées notamment par Ledoux et Talagrand.

En choisissant des probabilités de déviation $\varepsilon_{m'},\ m' \in \mathcal{M}$ sommables, on peut ainsi obtenir un contrôle de déviation uniforme sur m'; enfin, par intégration, on obtient une majoration pour le risque moyen $E_N[R_P(\widehat{f})]$. Birgé et Massart montrent que l'on peut appliquer ce schéma général dans des cadres variés [10, 11].

Massart [48] a appliqué avec succès ces principes au cadre de la classification en utilisant les techniques de Vapnik et Chervonenkis pour le point (2) ci-dessus. De plus, il montre avec Lugosi et Boucheron [16] que la «dimension de VC empirique» définie par

$$H_m(X_1^N) = \log \Delta^{\Gamma}(X_1^N)$$

satisfait également une inégalité de concentration et peut être utilisée en pratique à la place de son espérance dans la fonction de pénalité, qui est donc alors définie de façon aléatoire. Cependant, le résultat obtenu en classification n'est valable que pour la fonction de perte[7] $G_P(f) = E_P[(P(Y|X) - \mathbf{1}\{f(X) = Y\})^2]$, où f est un classifieur. Nous citons ici le théorème obtenu.

Théorème 2 (Massart). *Soit $\mathcal{F}_m, m \in \mathcal{M}$ une famille de modèles de classifieurs et Z_1^N un échantillon d'apprentissage tiré de façon i.i.d. selon une loi P sur $\mathcal{X} \times \mathcal{Y}$.*

On pose

$$\Gamma_m = \{\ \{X \in \mathcal{X} | f(X) = Y\} | f \in \mathcal{F}_m, Y \in \mathcal{Y}\}$$

et, pour un ensemble x_1^N de N points de $\mathcal{X}$,

$$H_m(x_1^N) = \log \Delta^{\Gamma_m}(x_1^N).$$

On suppose fixée une famille de poids positifs $(\omega_m), m \in \mathcal{M}$ tels que

$$\sum_{m \in \mathcal{M}} e^{-\omega_m} \leq \sum.$$

Il existe alors deux constantes K_1 et K_2 telles que, en définissant la fonction de pénalité (aléatoire)

$$\mathrm{pen}(m) = K_1 \frac{H_m(X_1^N)}{N} + K_2 \frac{\omega_m}{N},$$

l'estimateur $\widehat{f}$ obtenu par minimisation pénalisée de l'erreur empirique satisfait

$$\begin{aligned} &E_N E[(s_P(X,Y) - \mathbf{1}\{\widehat{f}(X) = Y\})^2] \\ &\leq C \left[\inf_{m \in \mathcal{M}} \left(\inf_{f \in \mathcal{F}_m} E[(s_P(X,Y) - \mathbf{1}\{f(X) = Y\})^2] + K_1 \frac{E_N[H_m(X_1^N)]}{N} + K_2 \frac{\omega_m}{N} \right) + C' \frac{1 + \Sigma}{N} \right], \end{aligned} \tag{1.12}$$

où $s_P(x,i) = P[Y = i | X = x]$ et C, C' sont des constantes.

Discussion

Faisons une comparaison succincte des deux méthodes présentées dans ce paragraphe. Elles sont toutes deux fondées sur un contrôle en probabilité des déviations de certains processus empiriques et mènent à une méthode de minimisation pénalisée de la perte empirique. En intégrant ces inégalités, on peut dans les deux cas obtenir une borne sur l'espérance d'un risque $E_N[R(\widehat{f})]$.

7. Cette fonction de perte coïncide avec l'erreur de classification dans le cas où $P(Y|X) \in \{0; 1\}$. Cf. discussion infra.

La différence importante est l'ordre de grandeur de la pénalité: dans le cas de Vapnik et Chervonenkis, elle est de l'ordre de $\sqrt{h_m \log(N)/N}$ (où h_m est la dimension de VC du modèle $\mathcal{F}_m$); dans le théorème de Massart, elle est de l'ordre de $h_m \log(N)/N$. Or on souhaite que la pénalité soit le plus faible possible, ce qui donne la préférence à cette dernière majoration.

Cependant, cet avantage sur la pénalité est contrebalancé par la présence de la constante C (plus grande que 1!) dans l'équation (1.12). En particulier, si la probabilité conditionnelle $s_P(x, i)$ n'est pas toujours 0 ou 1, alors, pour la fonction particulière de perte G_P prise en compte dans le théorème 2, $\inf_{f\in\mathcal{F}} G_P(f) > c > 0$ pour tout ensemble de classifieurs $\mathcal{F}$, et l'inégalité (1.12) n'assure donc pas que $\liminf_{N\to\infty}[G_P(\widehat{f}_N) - \inf_{f\in\mathcal{F}} G_P(f)] \leq 0$.

L'inégalité de Massart est le plus intéressante lorsque $P(Y|X) \in \{0; 1\}$: G_P coïncide alors avec l'erreur de classification. Si on suppose de plus que $\inf_{f\in\bigcup \mathcal{F}_m} G_P(f) = 0$, cette inégalité fournit un bon compromis biais/variance.

De fait, de récents résultats (par exemple [47, 26]) reprenant l'approche de Vapnik et Chervonenkis ont montré que l'on peut obtenir une borne sur la fonction de perte du type de l'inégalité (1.9), avec une pénalisation d'ordre $h_m \log(N)/N$ seulement, mais au prix d'une constante plus grande que 1 en facteur devant le terme d'erreur empirique.

Pour résumer de façon lapidaire, si nous comparons les majorations obtenues à l'inégalité «idéale» (1.6), on constate qu'on doit soit introduire une constante $C > 1$ en facteur devant le terme de «biais» (méthode de Birgé et Massart, ou bien développements de la méthode de Vapnik), soit accepter que le terme de pénalité (ou de «variance») soit d'ordre (peut-être) plus grand que le risque minimax[8].

Enfin, un point technique intéressant à souligner est que Massart se sert dans sa preuve d'un critère «inter-modèle» (puisqu'il étudie les variations de $\overline{L}_e(f_m) - \overline{L}_e(f_{m'})$) alors que les majorations de Vapnik sont toujours obtenues modèle par modèle. En ce sens on peut espérer que la technique utilisée par Massart puisse dans le futur donner lieu à des résultats plus fins.

8. Le risque minimax en classification dans un modèle $\mathcal{F}$ de dimension VC h est d'ordre h/N si l'on suppose qu'il existe dans le modèle une fonction de classification sans erreur, et d'ordre $\sqrt{h/N}$ sinon. On pourra consulter [47] pour des résultats fins sur la transition entre ces deux régimes.

1.2.2 Le mélange de modèles

La méthode d'estimation de classifieur qui fait l'objet de la première partie de ce travail est une alternative à la sélection de modèle: il s'agit du «mélange» de modèles. Cette méthode a été introduite et étudiée par Catoni [24]; d'autres auteurs [5, 7] en ont indépendamment proposé le principe.

Les idées initiales de l'estimation par mélange sont issues de méthodes pour la compression de données. Nous commençons donc par faire un détour en les introduisant dans ce cadre, ce qui permettra de motiver la façon dont est construit l'estimateur de mélange.

Une digression sur la compression de données

La compression de données considère le problème suivant: soit W une variable aléatoire, appelée *variable source*, à valeur dans un ensemble fini $\mathcal{W}$, tirée selon une loi P. On appelle *fonction de codage* une fonction $g : \mathcal{W} \to \mathcal{B}^*$, où $\mathcal{B}$ est un alphabet fini à b éléments, et $\mathcal{B}^*$ est l'ensemble des mots formés avec l'alphabet $\mathcal{B}$, telle que pour tous éléments distincts c, c' de $g(\mathcal{W})$, c ne soit pas un préfixe de c'. Ceci assure que si un opérateur transmet la chaîne concaténée $g(W_1)\ldots g(W_k)$, le récepteur de la chaîne pourra, connaissant la fonction g, reconstituer sans ambiguïté $W_1, \ldots, W_k$.

Le but de la compression de données est de construire une fonction de codage qui minimise, la longueur moyenne de la chaîne à transmettre, soit $E_P[|g(W)|]$, où $|g(W)|$ désigne la longueur du mot $g(W)$. Le théorème fondamental de Shannon, cité par exemple dans [28], est le suivant:

Théorème 3 (Shannon). *Soit W une v.a. à valeurs dans $\mathcal{W}$, de loi P, et g une fonction de codage dans un alphabet à b éléments. Alors*

$$E_P[|g(W)|] \geq E_P[-\log_b(P(W))] = H_b(W).$$

La quantité $H_b(X)$ est appelée entropie en base b de P. De plus, le théorème de Kraft[9] assure qu'il existe une fonction de codage g_1 telle que $|g_1(W)| = \lceil -\log_b(W) \rceil$, et donc assurant $E_P[|g_1(W)|] \leq H_b(W) + 1$.

Notons que si l'on considère le codage «par blocs de k», il existe de même une fonction de codage $g_k : \mathcal{W}^k \to \mathcal{B}^*$ telle que $E_k[|g_k(W_1^k)|] \leq kH_b(W) + 1$,

9. Le théorème de Kraft assure que, pour une collection de nombres entiers $(l_w), w \in \mathcal{W}$, on a $\sum_w b^{-l_w} \leq 1$ si et seulement s'il existe une fonction de codage g de $\mathcal{W}$ dans $\mathcal{B}^*$ telle que $|g(w)| = l_w$.

où $W_1^k = W_1, \ldots, W_k$ sont tirés de façon i.i.d. selon P. La longueur moyenne de codage par symbole source est alors plus petite que $H_b(W) + 1/k$.

Par extension, plutôt que de chercher à construire directement une fonction de codage g, on voudra trouver une fonction $\zeta : \mathcal{W} \to \mathbb{R}_+$ satisfaisant l'inégalité de Kraft $\sum_{w\in\mathcal{W}} b^{-\zeta(w)} \leq 1$, qui associe à tout symbole $w \in \mathcal{W}$ une «longueur de codage» (on interprète alors les longueurs de codage non entières comme des longueurs moyennes par symbole source en utilisant le codage par blocs de k, asymptotiquement lorsque $k \to \infty$).

Ainsi, par abus on dira que la borne inférieure de Shannon est atteinte en employant la longueur de codage ζ_P donnée par $\zeta_P(w) = -\log_b P(w)$. D'autre part, on peut supposer que pour les fonctions de codage ζ considérées, l'inégalité de Kraft est en fait une égalité: si ce n'est pas le cas, on peut en multipliant ζ par une constante positive inférieure à 1, obtenir une longueur de codage uniformément plus courte. De ce fait, $P_\zeta(w) = b^{-\zeta(w)}$ est alors une distribution de probabilité sur $\mathcal{W}$.

Ce cadre étant posé, soulignons les points suivants:

- Il faut connaître parfaitement la loi P pour obtenir le taux de compression optimal donné par $\zeta_P(w) = -\log_b(P(w))$;

- Si la loi P est inconnue, et qu'on l'estime au moyen d'une distribution $\widehat{P}$, le taux de compression obtenu pour $\zeta_{\widehat{P}}(w) = \log_b(\widehat{P}(w))$ sera $E_P[-\log_b(\widehat{P}(W))]$.

Il y a donc une équivalence formelle entre l'estimation de la densité P et la compression optimale de données de loi-source P; la mesure naturelle de performance pour la compression de données est la longueur moyenne de codage $\ell(\widehat{P})$, et la différence avec la borne inférieure de Shannon, $\ell(\widehat{P}) - \ell(P)$ est exactement le risque de Kullback en base b, $K_b(P, \widehat{P})$. Pour plus de simplicité, nous omettrons toutefois par la suite de prendre en compte que les logarithmes sont dans une certaine base b, et nous ne considérerons que des logarithmes naturels.

Compression «on-line» et mélange de sources

La compression dite «on-line» considère un cas particulier pour lequel $\mathcal{W} = \mathcal{Y}^N$, $\mathcal{Y} = \{0, \ldots, c\}$ et $W = Y_1^N$. La loi P selon laquelle est tiré le N-uplet Y_1^N est quelconque et en particulier, les Y_i ne sont pas supposés i.i.d. Le principe très simple du codage par mélange est le suivant: supposons que nous ayons un ensemble dénombrable $\mathcal{Q}$ de lois de probabilité (appelées «sources») sur $\mathcal{Y}^{\mathbb{N}}$, et une loi *a priori* π sur $\mathcal{Q}$. On définit la «probabilité-mélange»

$$Q_\pi(w) = \sum_{Q\in\mathcal{Q}} \pi(Q)Q(w);$$

alors, si nous utilisons Q_π pour coder $W = Y_1^N$, il est immédiat que la longueur de codage par symbole satisfait, *pour toute suite* Y_1^N

$$\frac{1}{N} - \log(Q_\pi(Y_1^N)) \leq \inf_{Q\in\mathcal{Q}} \left(\frac{1}{N} - \log(Q(Y_1^N)) + \frac{-\log(\pi(Q))}{N}\right). \tag{1.13}$$

Naturellement, en passant à l'espérance par rapport à la distribution P de tirage de Y_1^N, on obtient la même inégalité pour la longueur moyenne de codage.

Ce simple constat a déjà des conséquences théoriques importantes, par exemple (voir [53]), on peut choisir pour $\mathcal{Q}$ l'ensemble des probabilités calculables, et $\pi(Q) = \exp(-L_Q)$, où L_Q est la longueur du programme générant Q (en effet, comme un programme ne peut être le préfixe d'un autre, la famille (L_Q) vérifie l'inégalité de Kraft).

En pratique, il est fréquent d'être amené à considérer l'ensemble des probabilités Markoviennes d'ordre k, noté $\mathcal{Q}_k$. Cependant, $\mathcal{Q}_k$ n'est pas dénombrable. Il est néanmoins possible de construire pour tout k, une probabilité $\widehat{Q}_k$ ([53, 45]) telle que

$$\forall N\in\mathbb{N},\ \forall Y_1^N \in \mathcal{Y}^N,\ -\log(\widehat{Q}_k(Y_1^N)) \leq \inf_{Q\in\mathcal{Q}_k} \big(-\log(Q(Y_1^N)) + a_k\log(N)\big)\,, \tag{1.14}$$

où a_k est une constante.

On peut alors appliquer le principe du mélange de sources à l'ensemble $\{\mathcal{Q}_k | k \in \mathbb{N}\}$: en choisissant une probabilité π sur $\mathbb{N}$, et en définissant $\widehat{Q}_\pi = \sum_{i\in\mathbb{N}} \pi(i)\widehat{Q}_k$, on obtient donc la propriété suivante: pour toute chaîne Y_1^N,

$$-\frac{1}{N}\log(\widehat{Q}_\pi(Y_1^N)) \leq \inf_{k\in\mathbb{N}} \left(\inf_{Q\in\mathcal{Q}_k}\left(-\frac{1}{N}\log(Q(Y_1^N))\right) + \frac{a_k\log(N)}{N} + \frac{-\log(\pi(k))}{N}\right). \tag{1.15}$$

On appelle $\widehat{Q}_\pi$ une «probabilité universelle de codage» dans la classe des probabilités Markoviennes, au sens où la longueur de codage par symbole obtenue en utilisant cette probabilité est asymptotiquement aussi bonne que la longueur obtenue en utilisant n'importe quelle source Markovienne.

De plus, on peut choisir, à un terme supplémentaire d'ordre inférieur (en N) près, la valeur $a_k = c^k/2$, qui correspond asymptotiquement à la redondance minimax[10]. Enfin pour le modèle $\mathcal{M}_k$, Ryabko [53] montre que la redondance de $\widehat{Q}_\pi$ par rapport à une source stationnaire quelconque tend également vers 0 quand $N \to \infty$, presque sûrement et en moyenne.

De la compression de données à l'estimation

Nous voyons apparaître la similarité entre l'équation (1.15) obtenue pour le mélange de sources et l'équation-type (1.6) pour les méthodes de sélection adaptative; mais ici, (1.15) a été obtenue par mélange, et non par sélection, de modèles.

De plus, les distributions $\widehat{Q}_k$ vérifiant (1.14) sont en pratique construites par estimation statistique des distributions $P(Y_t|Y_{t-k}^{t-1})$ successivement pour $t = 1, \ldots, N$, à partir de l'échantillon déjà reçu Y_1^{t-1}: ces estimations sont utilisées pour prédire la probabilité de l'élément suivant de la chaîne et donc le coder (voir par exemple [50, 59]). La chaîne $Y_{t-k}^{t-1} = \mathrm{cont}(Y_t)$ est appelée le *contexte*[11] à l'ordre k de Y_t. En posant $X = \mathrm{cont}(Y)$, on voit ici apparaître une analogie supplémentaire avec le problème posé en classification et en régression, qui était l'estimation de $P(Y|X)$.

D'un point de vue formel, on construit donc pour tout $i \in \mathbb{N}$ un estimateur $\widehat{Q}_k^i(Y|\mathrm{cont}(Y))$ dépendant de l'échantillon Y_1^i, puis on définit la probabilité $\widehat{Q}_k$ comme

$$\widehat{Q}_k(Y_1^N) = \prod_{i=1}^{N} \widehat{Q}_k^{i-1}(Y_i|\mathrm{cont}(Y_i)) \tag{1.16}$$

dont on montre qu'elle vérifie (1.14) lorsque les estimateurs $\widehat{Q}_k^i$ sont bien choisis.

On peut penser à renverser la construction et, réciproquement, à partir d'une probabilité «universelle» $\widehat{Q}$ (par exemple, la probabilité $\widehat{Q}_\pi$ obtenue par mélange), construire une suite d'estimateurs naturels associés:

$$\widehat{Q}^i(Y|\mathrm{cont}(Y)) = \widehat{Q}\left(Y_{i+1}|Y_1^i\right) \tag{1.17}$$

10. Dans un sens similaire à celui proposé plus haut au paragraphe 1.1.4. Des résultats variés sur les redondances minimax sont exposés dans [35, 51, 8]. Il est en particulier possible de montrer que la redondance minimax dans un modèle non dénombrable peut également être obtenue par mélange en utilisant la loi *a priori* la moins favorable, voir [29, 30]

11. On suppose en général, pour éviter les difficultés dues aux effets de bord, qu'il existe un contexte initial Y_{1-k}^0.

Ceci suggère donc un principe général de construction d'estimateurs adaptatifs qui pourrait être résumé informellement de la manière suivante:

$$\begin{gathered}
\text{Famille d'estimateurs (à partir d'un échantillon de taille } i\text{)} \\
\widehat{Q}^i_k(Y|\mathrm{cont}(Y)),\ k \in \mathbb{N} \\
\downarrow \\
\text{Famille de probabilités de codage} \\
\widehat{Q}_k\left(Y_1^N\right) = \prod_{i=0}^{N-1} \widehat{Q}^i_k\left(Y_{i+1}|\mathrm{cont}(Y_{i+1})\right),\ k \in \mathbb{N} \\
\downarrow \\
\text{Probabilité universelle obtenue par mélange} \\
\widehat{Q}_\pi(Y_1^N) = \sum_{k\in\mathbb{N}} \pi(k)\widehat{Q}_k(Y_1^N) \\
\downarrow \\
\text{Estimateur adaptatif par rapport à la famille de modèles de départ} \\
\widehat{Q}^{N-1}_\pi(Y_N|\mathrm{cont}(Y_N)\,) = \widehat{Q}_\pi(Y_N|Y_1^{N-1})
\end{gathered}$$

On peut donner une interprétation Bayésienne de cette construction: l'estimateur final est un estimateur Bayésien prédictif formé à partir des distributions $\widehat{Q}_k,\ k \in \mathbb{N}$ et de la distribution *a priori* π.

Il y a néanmoins une différence de but importante entre la compression de données et l'estimation statistique: la compression «on-line» de données vise à coder une chaîne Y_1^N dans son intégralité; comme nous l'avons vu, la mesure naturelle de performance est donc la longueur de codage $-\log \widehat{Q}(Y_1^N)$. Dans le cadre de l'estimation statistique, l'échantillon Y_1^N est connu et on cherche à minimiser un risque sur l'estimation de la distribution d'*une* nouvelle donnée $\widehat{Q}(Y_{N+1}|Y_1^N)$. Ainsi, en construisant l'estimateur $\widehat{Q}^N_\pi(Y|\mathrm{cont}(Y))$ selon le principe proposé ci-dessus, le statisticien cherchera à minimiser la perte de Kullback (cf. 1.1.3)

$$H^N\left(\widehat{Q}^N_\pi\right) = E_P\left[-\log\left(\widehat{Q}^N_\pi\left(Y_{N+1}|\mathrm{cont}(Y_{N+1})\,\right)\right)\right]$$

alors que la longueur de codage moyenne par symbole pour la probabilité universelle est donnée par

$$\frac{1}{N}E_P\left[-\log(\widehat{Q}_\pi(Y_1^N))\right] = \frac{1}{N}\sum_{i=0}^{N-1} E_P\left[-\log\left(\widehat{Q}^i_\pi(Y_{i+1}|\mathrm{cont}(Y_{i+1})\,)\right)\right] = \frac{1}{N}\sum_{i=0}^{N-1} H^i\left(\widehat{Q}^i_\pi\right)$$

et correspond donc au risque *cumulé* moyen du statisticien.

Ainsi, on ne peut pas déduire directement de l'inégalité (1.15) une inégalité de type (1.6). C'est pour remédier à ce problème que l'on doit modifier quelque peu la construction présentée ci-dessus en introduisant l'estimateur de *mélange progressif*.

L'estimateur de mélange progressif

Le détour par la compression de données nous a donc permis de montrer de manière informelle comment on peut espérer passer d'une probabilité «universelle» obtenue par mélange à une estimation adaptative de distribution. Le chapitre 2 du présent travail définit et étudie dans un cadre général de régression l'estimateur *mélange progressif* introduit par Catoni [24], qui utilise les idées présentées au paragraphe précédent en surmontant la difficulté que nous avons évoquée. En voici le principe général.

Considérons dans un premier temps un cas simplifié où tous les modèles $\mathcal{F}_m, m \in \mathcal{M}$ sont réduits à des singletons:

$$\forall m \in \mathcal{M},\ \mathcal{F}_m = \{Q_m(\cdot|\cdot)\}$$

où Q_m est une distribution conditionnelle de Y sachant X. On suppose fixée une probabilité *a priori* π sur $\mathcal{M}$ (qui est est dénombrable par hypothèse).

On définit la probabilité de mélange $Q_{\text{mix.}}$ par

$$\forall n \in \mathbb{N},\ Q_{\text{mix.}}(Y_1^n|X_1^n) = \sum_{m\in\mathcal{M}} \pi(m) \prod_{i=1}^{n} Q_m(Y_i|X_i),$$

et l'estimateur de mélange progressif $\widehat{Q}_\pi^N(Y|X)$ par

$$\begin{aligned}\widehat{Q}_\pi^N(Y_{N+1}|X_{N+1}) &= \frac{1}{N}\sum_{M=1}^{N} Q_{\text{mix.}}(Y_{N+1}|X_{N+1}, Z_M^N) \\ &= \frac{1}{N}\sum_{M=1}^{N} \frac{\sum_{m\in\mathcal{M}} \pi(m) \prod_{i=M}^{N+1} Q_m(Y_i|X_i)}{\sum_{m'\in\mathcal{M}} \pi(m') \prod_{j=M}^{N} Q_{m'}(Y_j|X_j)}.\end{aligned} \tag{1.18}$$

Ainsi, en comparant avec le paragraphe précédent, on constate que $Q_{\text{mix.}}$ joue le rôle de la probabilité universelle Q_π. La distribution $\widehat{Q}_\pi^N$ n'est pas définie comme le suggérait initialement l'équation (1.17), mais par une somme de Cesaró du deuxième membre de (1.17) où l'on fait varier la taille de l'échantillon Z_M^N.

Cet estimateur satisfait alors la

Proposition 4. *Pour un échantillon* $Z_1^{N+1} = ((X_i, Y_i)_{i=1,\ldots,N+1})$ *tiré selon une loi de probabilité* P_{N+1} échangeable*, l'estimateur de mélange progressif* $\widehat{Q}_\pi^N$ *défini ci-dessus vérifie l'inégalité*

$$E_{N+1}\left[-\log \widehat{Q}_\pi^N(Y_{N+1}|X_{N+1})\right] \leq \inf_{m\in\mathcal{M}}\left(E\left[-\log Q_m(Y|X)\right] + \frac{-\log(\pi(m))}{N}\right) \tag{1.19}$$

Commençons par souligner une différence importante avec l'inégalité (1.13), dont la proposition ci-dessus est l'analogue dans le cadre statistique: l'inégalité pour le codage était valable pour *toute* chaîne Y_1^N, mais l'inégalité (1.19) n'est vraie qu'en espérance prise par rapport à P_{N+1}; de plus, l'hypothèse d'échangeabilité faite sur P_{N+1} joue un rôle essentiel dans la démonstration.

Dans le cas «réaliste» où les modèles $\mathcal{F}_m$ ne sont pas des singletons, on suppose qu'on dispose d'estimateurs $\widehat{Q}_m^N$ qui satisfont

$$E_{N+1}\left[-\log(\widehat{Q}_m^N(Y_{N+1}|X_{N+1}))\right] \leq \inf_{Q_m\in\mathcal{F}_m}\left(E\left[-\log(Q_m(Y|X))\right]\right) + \Delta(m,N),$$

où, idéalement, $\Delta(m,N)$ est proche du risque minimax R_m^N. Par exemple, nous construirons de tels estimateurs dans le cadre des arbres de régression dans la section suivante.

Cependant, on ne peut pas remplacer directement les distributions «fixes» dans la proposition 4 par des estimateurs dépendant des mêmes données Z_1^N. Nous recourrons donc à une méthode classique en divisant l'échantillon Z_1^N en deux, $E = Z_1^{N-L}$ sera utilisé pour l'estimation au sein de chaque modèle m, donnant lieu à une probabilité estimée $\widehat{Q}_m^E(Y|X)$, et $T = Z_{N-L+1}^N$ sera utilisé pour le mélange inter-modèle, construit exactement comme ci-dessus en considérant les distributions $\widehat{Q}_m^E$ comme fixées. L'estimateur de mélange progressif $\widehat{Q}_\pi^N$ ainsi construit vérifie alors l'inégalité

$$\begin{aligned} &E_{N+1}\left[-\log(\widehat{Q}_\pi^N(Y_{N+1}|X_{N+1}))\right] \\ &\leq \inf_{m\in\mathcal{M}}\left(\inf_{Q_m\in\mathcal{F}_m}\left(E\left[-\log(Q_m(Y|X))\right]\right) + \Delta(m,N-L) + \frac{-\log(\pi(m))}{L}\right) \end{aligned} \tag{1.20}$$

Discussion

Pour conclure sur ce point, comparons brièvement l'estimateur de mélange progressif et les méthodes de sélection adaptative de modèle présentées auparavant en soulignant ses points forts et ses inconvénients.

D'une part, la majoration (1.20) est plus proche de l'inégalité «idéale» (1.6): le terme de biais est exact (il n'y a pas de constante $C > 1$ en facteur) et le terme de «variance» reste de l'ordre (en N) du risque minimax (en choisissant $L = \gamma N$, $0 < \gamma < 1$, dans (1.20), on conserve le même ordre en N). Catoni montre par ailleurs dans [24] sur un modèle-jouet contenant seulement deux distributions de Bernoulli de paramètres $\frac{1}{2} \pm \frac{\alpha}{\sqrt{N}}$, qu'on ne peut pas espérer, en toute généralité, obtenir une majoration telle que (1.20) par une méthode de sélection de modèle.

Le mélange de modèles présente d'autre part un avantage supplémentaire: il ne requiert pas de calculer explicitement les vitesses de convergence au sein des modèles considérés. Dans les méthodes de sélection de modèle présentées en 1.2.1, le terme de pénalité $\mathrm{pen}(m)$ fait intervenir la complexité du modèle m sous forme de la dimension VC h_m ou de son estimée empirique H_m (Dans des cas plus généraux d'estimation pénalisée, Birgé et Massart [10, 11] définissent également la pénalité à partir d'autres mesures de la complexité du modèle telles que l'entropie métrique). Or, en pratique, il est souvent difficile de calculer h_m ou même H_m. Avec l'estimateur de mélange progressif, le mélange de modèles remplace en quelque sorte une pénalisation explicite par une pénalisation «automatique»: les estimateurs $\widehat{Q}_m^E$ moins bons se verront affecter un poids (exponentiellement) plus faible dans le mélange du fait de leur moindre performance sur l'échantillon $T = Z_{N-L+1}^N$, qui sert ainsi d'échantillon «de sélection» [12].

La méthode de mélange permet donc de tirer le meilleur parti des estimateurs disponibles, que ceux-ci atteignent ou non la vitesse optimale au sein de leur modèle.

Il y a un revers à cette médaille: il faut en principe calculer une somme sur tous le modèles $m \in \mathcal{M}$, ce qui pose problème lorsqu'ils sont en très grand nombre: nous proposerons au paragraphe 1.2.3 une méthode approchée de calcul par algorithme de Monte-Carlo.

Soulignons également, d'une part, que (1.20) n'est valable que pour un risque spécifique, le risque de Kullback (Catoni montre dans [23] qu'on peut déduire de cette majoration des bornes correspondant à d'autres risques, en particulier l'erreur de classification, mais on perd en général le terme de biais exact); d'autre part, que cette majoration ne provient pas d'inégalités de déviation, et qu'on ne peut pas *a priori* en déduire de telles inégalités, qui étaient disponibles dans les méthodes de

12. Il est toutefois souhaitable de tenir compte de la complexité des modèles dans le choix de la probabilité *a priori* π, mais cela n'est pas aussi déterminant que dans la sélection de modèles.

sélection de modèle.

Enfin, on peut trouver discutable la division en deux de l'échantillon Z_1^N: en pratique, on est tenté d'utiliser l'échantillon en entier pour les deux étapes de l'estimation[13].

1.2.3 L'estimateur de mélange progressif appliqué aux arbres de décision

On peut donc à présent revenir au modèle particulier des arbres de décision présenté au 1.1.4 pour y appliquer la construction du paragraphe précédent. Nous présentons ici un aperçu des résultats qui seront établis en détail au chapitres 2 pour la partie théorique, et 3 pour l'implémentation pratique.

Estimateurs pour les modèles d'arbres et résultats obtenus

Rappelons qu'un arbre de décision A est défini comme un triplet (T, K, Θ) où T est une structure d'arbre binaire complet, K est une famille de «questions» attachées aux nœuds intérieurs de l'arbre, et Θ est une famille de distributions sur $\mathcal{Y} = \{0, \ldots, c\}$ attachées aux feuilles terminales de l'arbre. En adoptant la notation $m = (T, K)$ pour simplifier, on définit la famille de modèles $\mathcal{A}_m = \{A = (T, K, \Theta) \,|\, (T, K) = m; \Theta : \partial T \to \mathcal{S}_c\}$, où $\mathcal{S}_c$ est le simplexe de dimension c. Pour appliquer la construction du paragraphe précédent, la famille de modèles $\mathcal{M}$ doit être dénombrable: nous supposerons donc que les questions attachées aux nœuds intérieurs sont prises dans une «bibliothèque» $\mathcal{Q} \subset \mathcal{X}^{\{0,1\}}$ fixée et au plus dénombrable.

Il faut d'abord définir un estimateur $\widehat{Q}_m^N$ associé à chaque modèle $\mathcal{A}_m$: celui-ci sera donné par un estimateur de Laplace en chaque nœud terminal. Pour $x \in \mathcal{X}$, notons $m(x)$ la feuille à laquelle parvient l'objet x lorsqu'on le fait descendre le long des branches de l'arbre étiqueté m. Pour toute feuille $s \in \partial T$, notons n_s^i le nombre d'exemples $Z = (X, Y)$ de l'échantillon Z_1^N tels que $m(X) = s$ et $Y = i$. On définit alors

$$\widehat{Q}_m^N(Y = i | m(X) = s) = \frac{n_s^i + 1}{\sum_i n_s^i + c + 1}. \tag{1.21}$$

13. Récemment, Catoni a proposé une autre méthode de mélange, l'estimateur «de Gibbs» [23], qui permet sous certaines hypothèses d'éviter la division en deux de l'échantillon en procédant à un mélange sur un ensemble non dénombrable de modèles.

Nous montrerons au chapitre 2 que cet estimateur vérifie, sous l'hypothèse que la probabilité de tirage des exemples P_{N+1} est échangeable:

$$E_{N+1}\left[-\log(\widehat{Q}_m^N(Y_{N+1}|X_{N+1})\right] \leq \inf_{Q_m\in\mathcal{A}_m} \left(E\left[-\log(Q_m(Y|X))\right]\right) + \frac{c|\partial T|}{N+1}.$$

Notons, en vertu de cette majoration et de l'inégalité (1.5), que l'estimateur $\widehat{Q}_m^N$ a une vitesse de convergence d'ordre optimal en $|\partial T|$ et en N.

Il reste enfin à choisir la distribution *a priori* π. Parmi les multiples choix possibles, nous en proposons un particulier permettant d'obtenir une borne entièrement explicite: la structure d'arbre T est tirée selon la distribution de l'arbre généalogique d'un processus de branchement binaire de paramètre $\rho \leq 1/2$ (un individu donne naissance à deux fils avec la probabilité ρ, ou meurt sans descendants avec la probabilité $1-\rho$). La bibliothèque $\mathcal{Q}$ est supposée finie et on tire les questions attachées aux nœuds intérieurs de T uniformément dans $\mathcal{Q}$. On obtient alors

$$\pi(m) = (1-\rho)\left(\frac{\rho(1-\rho)}{|\mathcal{Q}|}\right)^{|\partial T|}.$$

Il ne reste qu'à expliciter la borne donnée par 1.20 dans ce cas, à savoir

$$\begin{aligned}&E_{N+1}\left[-\log(\widehat{Q}_\pi^N(Y_{N+1}|X_{N+1})\,)\right]\\ &\leq \inf_{m\in\mathcal{M}}\left(\inf_{Q_m\in\mathcal{A}_m}\left(E\left[-\log(Q_m(Y|X))\right]\right) + \frac{c|\partial T|}{N-L} + \frac{\log|Q| - |\partial T|\log(\rho(1-\rho))}{L}\right).\end{aligned} \tag{1.22}$$

En pratique cependant, la bibliothèque $\mathcal{Q}$ des questions disponibles n'est en général pas dénombrable. Un exemple très classique et souvent rencontré en pratique est le suivant: $\mathcal{X} = \mathbb{R}^p$, et les questions sont de la forme $\mathbf{1}\{x^i > t\}, i \in \{1,\dots,p\}, t \in \mathbb{R}$. Nous montrerons au chapitre 2 que l'on peut généraliser la construction de l'estimateur de mélange progressif à ce cas (et plus généralement au cas où $\mathcal{Q}$ est de dimension VC h finie) en définissant une loi «*a priori*»[14] $\widetilde{\pi}$ qui dépend des données Z_1^N. L'idée est de regrouper les questions qui sont statistiquement équivalentes vis-à-vis de l'échantillon, et de choisir une loi uniforme sur ces regroupement de questions (qui sont alors en nombre fini). Dans ce cas, nous établirons que l'on peut obtenir une borne analogue à (1.22) avec un terme supplémentaire d'ordre $\log(N)/N$, dont nous montrerons qu'il est d'ordre optimal au sens minimax.

14. C' est un abus de continuer à l'appeler loi *a priori* puisqu'elle dépend alors des données...

L'implémentation pratique par méthode de Monte-Carlo

Comme nous l'avons évoqué dans la section précédente, un des principaux problèmes lié d'une façon générale aux estimateurs de mélange est leur calcul effectif, qui implique en principe une somme portant sur tous les modèles considérés. Pour simplifier, nous nous restreindrons ici au calcul de l'estimateur bayésien

$$Q^N_\pi(Y_{N+1}|X_{N+1}) = Q_{\text{mix.}}(Y_{N+1}|X_{N+1}, Z_1^N) = \frac{Q_{\text{mix.}}(Y_1^{N+1}|X_1^{N+1})}{Q_{\text{mix.}}(Y_1^N|X_1^N)}.$$

avec

$$Q_{\text{mix.}}(Y_1^N|X_1^N) = \sum_{m\in\mathcal{M}} \pi(m) \prod_{i=1}^N \widehat{Q}^E_m(Y_i|X_i).$$

Rappelons que l'estimateur de mélange progressif $\widehat{Q}^N_\pi$ est défini (voir (1.18)) comme la moyenne de Cesaró des $Q^i_\pi(Y_{N+1}|X_{N+1})$ lorsque i varie de 1 à N, et d'autre part que l'estimation au sein de chacun des modèles $m \in \mathcal{M}$ se fait en principe sur un échantillon E séparé.

Un moyen largement utilisé pour calculer de façon approchée des sommes de cette forme apparaissant dans un cadre bayésien est de simuler la loi *a posteriori* sur l'ensemble $\mathcal{M}$ par méthode de Monte-Carlo. Ainsi, lorsque les modèles sont des arbres de décision, on peut proposer de construire une chaîne de Markov à valeurs dans $\mathcal{M}$ et admettant pour loi stationnaire la loi *a posteriori*. Rappelons qu'un modèle $m \in \mathcal{M}$ est alors défini comme un couple (T, K) formé d'une structure d'arbre et d'un ensemble de questions attachées aux nœuds intérieurs de l'arbre. Une façon naturelle de construire une telle chaîne de Markov est de considérer uniquement des transitions de $m = (T, K)$ vers $m' = (T', K')$ où T' est obtenu à partir de T en supprimant ou en ajoutant deux feuilles-sœurs, et K' à partir de K en ajoutant où en supprimant la question attachée au nœud-père correspondant. Cette construction est détaillée à la fin du chapitre 2 et des méthodes similaires ont par ailleurs été proposées par d'autres auteurs [25, 31]. Cependant, nos expériences ont montré que la vitesse de convergence d'une telle méthode est très mauvaise. En effet, on constate que la chaîne fait croître l'arbre rapidement et a ensuite tendance à rester bloquée près de minima locaux: par suite seule une petite partie de l'espace des arbres est visitée. Il est donc fortement souhaitable de pouvoir corriger ce défaut.

Amélioration de l'algorithme de Monte-Carlo

Lorsque que l'on considère des arbres de décision, il y a un cas particulier où $Q_{\text{mix.}}(Y_1^N|X_1^N)$ peut en fait se calculer de façon exacte et très efficace grâce à l'algorithme dit de «pondération récursive des arbres» (*recursive tree weighting*) introduit par Willems et al. [59] dans le cadre de la compression de données.

Fixons un entier d et considérons $\mathcal{T}_d$ l'arbre maximal de profondeur d. On fixe arbitrairement l'ensemble des questions $\mathcal{K}_d$ qui sont attachées au nœuds intérieurs de l'arbre et on considère l'ensemble de modèles $\mathcal{M}_d$ composé tous les arbres de décision qui sont des sous-arbres de $(\mathcal{T}_d, \mathcal{K}_d)$. Sur cet ensemble, on considère, comme au paragraphe précédent, la loi *a priori* π_1 sur les structures d'arbres binaires donnée par la distribution de l'arbre de généalogie d'un processus de branchement binaire de paramètre ρ, et stoppée à la profondeur d (ce qui correspond à dire que tout individu dela d-ième génération meurt sans descendants). Il n'y a pas de loi a priori sur les questions, puisque celles-ci sont fixées à l'avance. La distribution $Q_{\text{mix.}}$ s'écrit alors

$$\begin{aligned}
Q_{\text{mix.}}(Y_1^N|X_1^N) &= \sum_{(T,K)=m\in\mathcal{M}_d} \pi_1(T) \prod_{s\in\partial T}\prod_{i=0}^{c} \left(\widehat{Q}_m(Y=i|m(X)=s)\right)^{n_s^i} \\
&= \sum_{(T,K)=m\in\mathcal{M}_d} \pi_1(T) \prod_{s\in\partial T} \xi(s)
\end{aligned}$$

où les compteurs n_s^i sont définis comme au paragraphe précédent et on a posé $\xi(s) = \prod_{i=0}^{c}\left(\widehat{Q}_T(Y=i|m(X)=s)\right)^{n_s^i}$. Du fait de la structure particulière de la somme et de la loi *a priori*, cette expression peut être factorisée de façon à être calculée récursivement de la manière suivante: on définit récursivement

$$\forall s\in T \quad \sigma(s) = \begin{cases} \xi(s) & \text{si } s \text{ est à la profondeur } d, \\ (1-\rho)\xi(s) + \rho\sigma(s0)\sigma(s1) & \text{sinon.} \end{cases}$$

On rappelle qu'un nœud s de l'arbre est identifié à un mot binaire donnant sa position dans l'arbre: $s0$ et $s1$ désignent donc les deux nœuds-fils de s. On peut alors montrer facilement que $\sigma(\emptyset) = Q_{\text{mix.}}(Y_1^N|X_1^N)$. Ceci signifie que l'on peut calculer cette somme de façon exacte en $\mathcal{O}(2^d)$ opérations, ce qu'il faut comparer au cardinal de $\mathcal{M}_d$, de l'ordre de 2^{2^d}.

Cet algorithme exact va nous permettre de construire une méthode de Monte-Carlo plus efficace dans le cas général où l'ensemble $\mathcal{K}_d$ des questions n'est pas fixé.

Pour cela, on fait l'hypothèse que la loi *priori* π peut s'écrire comme un produit

$$\pi(m) = \pi(T, K) = \pi_1(T)\pi_2(K|T),$$

où π_1 est la loi déjà définie ci-dessus, et $\pi_2(K|T)$ est la marginale, sur les nœuds intérieurs de T, d'une distribution π_2 fixée à l'avance sur l'ensemble $\mathcal{K}_d$ des questions attachées aux nœuds intérieurs de $\mathcal{T}_d$, l'arbre maximal de profondeur d.

On peut alors décomposer la loi *a posteriori* sur les structures d'arbre étiquetées comme produit de la loi *a posteriori* sur l'ensemble $\mathcal{K}_d$ (les questions) par la loi *a posteriori* sur la structure d'arbre T connaissant $\mathcal{K}_d$. La somme par rapport à cette dernière loi *a posteriori* conditionnellement à $\mathcal{K}_d$ peut être calculée exactement comme nous venons de le voir. Quant à la loi *a posteriori* $\overline{\pi}$ sur $\mathcal{K}_d$, on propose de la simuler par une méthode de Monte-Carlo par un algorithme semblable à l'échantillonneur de Gibbs:

1. Initialiser en tirant au hasard une configuration de questions $\mathcal{K}_d$ attachées aux nœuds intérieurs de $\mathcal{T}_d$ et poser $S = 0$.

2. Choisir au hasard un nœud intérieur s de $\mathcal{T}_d$ selon une distribution fixée à l'avance.

3. Tirer une nouvelle question f_s pour le nœud s selon la loi π_2. Appelons $\mathcal{K}'_d$ le nouvel ensemble de questions.

4. Accepter la transition vers $\mathcal{K}'_d$ avec la probabilité
$$1 \wedge \frac{\overline{\pi}(\mathcal{K}'_d)}{\overline{\pi}(\mathcal{K}_d)}. \tag{1.23}$$

5. Calculer exactement la somme sur $T \in \mathcal{T}_d$ lorsque l'ensemble $\mathcal{K}_d$ est fixé et incrémenter S de la valeur obtenue. Puis retourner à l'étape (2).

Si l'on stoppe l'algorithme au bout de k itérations, S/k donne donc une valeur approchée de la quantité voulue.

Au chapitre 3, nous donnerons les détails de cette construction, en montrant notamment que la loi *a posteriori* $\overline{\pi}$ sur les questions peut également être calculée à constante près en utilisant l'algorithme de pondération récursive. Le temps de calcul correspondant s'avère raisonnable (de l'ordre de la minute pour générer plusieurs milliers d'itérations de l'algorithme pour des arbres de profondeur 8). Nous exposerons également une méthode pour calculer des quantités d'intérêt telles que la profondeur moyenne des arbres visités, puis nous donnerons le résultat d'expériences sur des bases de données classiques en étudiant diverses variantes de l'algorithme de base.

1.3 L'agrégation d'estimateurs

1.3.1 Motivation initiale: observations expérimentales

L'un des résultats expérimentaux les plus intéressants et surprenants observés à partir de l'algorithme décrit ci-dessus est obtenu avec la variante suivante: on introduit un paramètre de «température inverse» $0 \leq \beta \leq 1$ qui va modifier la loi *a posteriori* $\overline{\pi}$ sur les questions: on remplace $\overline{\pi}$ par $\widetilde{\pi} \propto \overline{\pi}^{\beta}$. D'un point de vue algorithmique, cela revient juste à changer l'étape (4) de l'algorithme de Monte-Carlo décrit ci-dessus en remplaçant l'équation (1.23), donnant la probabilité d'acceptation de la transition, par

$$1 \wedge \left(\frac{\overline{\pi}(\mathcal{K}_d')}{\overline{\pi}(\mathcal{K}_d)} \right)^{\beta}.$$

L'idée d'introduire un tel paramètre a été initialement suggérée par l'article [23], dans lequel O.Catoni propose d'autres méthodes de mélanges que l'estimateur de mélange progressif: il montre que sous certaines hypothèses, en introduisant un tel paramètre de température sur une loi *a posteriori*, on peut arriver à montrer des inégalités-oracle en risque de Kullback pour l'estimateur de mélange ainsi obtenu qu'il appelle «estimateur de Gibbs» (l'avantage de cette méthode par rapport à l'estimateur de mélange progressif est qu'elle s'affranchit notamment de l'hypothèse de dénombrabilité des modèles qui nécessitait de couper l'échantillon en deux, comme nous l'avons montré au paragraphe 1.2.2).

Nous n'avons pas suivi avec précision ce cadre théorique qui nous a seulement inspiré pour réaliser des expériences supplémentaires. Le résultat assez surprenant obtenu est que dans tous les exemples que nous avons testé, il s'est avéré avantageux (en terme d'erreur de généralisation mesurée sur un échantillon de test) de choisir un tel $\beta < 1$. Dans nos expériences, les valeurs de β donnant la meilleure erreur de généralisation sont de l'ordre de 0.1.

Ce résultat est surprenant car, intuitivement, élever la loi *a posteriori* à une puissance strictement plus petite que 1 revient à la rendre plus «uniforme». Or le rôle de la loi *a posteriori* revient, en première analyse, à mettre plus de poids sur les modèles plus performants (au sens où ils ont une vraisemblance plus grande). Si la différence entre les poids *a posteriori* des modèles les plus performants et des autres est moindre, il semble que les modèles vont être moins bien discriminés. Pourtant, en pratique on observe une augmentation des performances de l'estimateur obtenu par cette méthode.

La première explication qui peut venir à l'esprit est que lorsque $\beta = 1$, ce qui revient à l'algorithme «standard», le paysage de la loi *a posteriori* $\overline{\pi}$ présente des barrières d'énergie trop grandes pour que l'algorithme de calcul approché par méthode de Monte-Carlo ait une convergence satisfaisante. En augmentant la «température», ces barrières sont plus basses et la convergence de la chaîne serait améliorée, ce qui compenserait la perte théorique de performance du le mélange exact. Pour tester cette hypothèse, nous avons procédé à l'expérience suivante: dans l'esprit d'un algorithme de recuit simulé, on lance une chaîne de Monte-Carlo à haute température puis on diminue la température progressivement jusqu'à arriver à $\beta = 1$. Le dernier arbre obtenu est gardé, et on réitère l'expérience plusieurs fois pour obtenir plusieurs arbres, dont on fait alors la moyenne. Si l'explication proposée était correcte, cette méthode ne pourrait qu'améliorer encore les résultats: or on observe que l'on retrouve les résultats obtenus pour $\beta = 1$.

Pour étudier ce fait, nous n'avons pas utilisé directement le cadre proposé par Catoni pour les «estimateurs de Gibbs»: en effet, d'une part celui-ci nécessiterait que l'on fasse également intervenir le paramètre de température pour la loi *a posteriori* sur les structures d'arbres et sur l'estimation de paramètres, or nous avons constaté que ceci mène en pratique à une détérioration de la performance. D'autre part, l'étude théorique menée par O.Catoni donne lieu à des inégalités-oracle de type (1.6), qui comparent la performance du mélange à celle de chacun des éléments du modèle. Or le résultat des expériences montre que le classifieur obtenu par moyenne a en fait une meilleure performance que n'importe lequel des classifieurs individuels. Il y a donc un effet supplémentaire dû au fait qu'une combinaison convexe de classifieurs peut faire mieux qu'un classifieur seul.

C'est pourquoi nous avons choisi d'explorer une piste d'explication plus élémentaire suggérée initialement par Y.Amit et D.Geman [4] dans leurs travaux sur les arbres de classification pour la reconnaissance de caractères manuscrits.

Le rôle de la corrélation conditionnelle dans l'agrégation d'estimateurs

Y.Amit et D.Geman proposent dans [4] un algorithme «randomisé» (c'est-à-dire partiellement aléatoire) pour construire indépendemment plusieurs arbres de décision dont ils font ensuite la moyenne, et montrent que la classifieur composite ainsi construit obtient des résultats bien supérieurs à ceux du meilleur d'entre les classifieurs individuels. Pour expliquer cette amélioration, ils utilisent la notion de *covariance conditionnelle* relativement à la classe.

L'idée, très simple, repose sur une analyse biais/variance du classifieur composite $H = \frac{1}{K}\sum_{i=1}^{K} h_i$, où les classifieurs h_i sont des fonctions qui à tout $x \in \mathcal{X}$ associent une répartition de probabilité sur l'ensemble des classes $\mathcal{Y}$. La classe prédite par le classifieur est donnée par la classe à qui est attribuée la plus grande probabilité. Si nous testons H sur un nouvel exemple X dont la classe est $Y = k$, par l'inégalité de Chebychev on obtient une majoration de l'erreur de classification de H sur des éléments de classe k. Posons $M_k = E_k[H(X)]$, où E_k désigne l'espérance sur le tirage d'un nouvel exemple de classe k. Définissons θ_k, la marge moyenne sur la classe k, comme la différence entre la k-ième composante de M_k (i.e. la probabilité moyenne attribuée la vraie classe) et la plus grande de ses composantes restantes. Alors il est aisé de montrer que

$$P_k[\text{Erreur}(H)] \leq \frac{1}{\theta_k^2} \sum_{0 \leq i \leq c} Var_k[H(X)_i]. \tag{1.24}$$

De plus, on a

$$Var_k[H(X)_p] = \frac{1}{K^2}\left(\sum_{i=1}^{K} Var_k[h_i(X)_p] + \sum_{\substack{1 \leq i,j \leq K \\ i \neq j}} Covar_k[h_i(X)_p, h_j(X)_p]\right);$$

lorsque le nombre de classifieurs K construits devient grand, le deuxième terme de cette dernière équation devient prépondérant. Y.Amit et D.Geman suggèrent donc qu'il faut s'efforcer en pratique de construire des classifieurs successifs ayant une faible covariance conditionnelle.

Il existe en fait de nombreux autres algorithmes proposés par différents auteurs pour construire des classifieurs multiples qui sont aussi agrégés. Citons notamment le «Bagging» [17] et le «Boosting» [54], qui jouissent d'une grande popularité. Dans tous les cas, une amélioration des performances est rapportée par les auteurs.

Dans le chapitre 4, qui expose un travail en collaboration avec Y.Amit, nous reprenons le raisonnement ci-dessus dans un cadre général incluant de nombreuses méthodes de construction de classifieurs que nous appelons génériquement «classifieurs randomisés multiples» («*multiple randomized classifiers*»). Ce travail est en grande partie expérimental: nous comparons différents protocoles de construction sur un exemple artificiel ainsi que sur un exemple de reconnaissance de symboles mathématiques générés par LaTeX. Nous montrons que les meilleures performances obtenues vont toujours de pair avec une faible covariance conditionnelle, et qu'il y a en général un compromis à trouver entre la marge moyenne (θ_k ci-dessus) que l'on souhaite élevée et la covariance moyenne que l'on souhaite faible.

En général, on constate qu'il est avantageux d'inclure dans le mélange des classifieurs dont la marge (et donc la performance en termes d'erreur de généralisation) est plus faible mais qui ont une covariance faible avec les autres classifieurs construits. Nous suggérons enfin une méthode itérative de construction d'estimateurs basée seulement sur les critères de variance et de marges empiriques: les performances obtenues sont du même ordre que celles des meilleures autres méthodes «classiques».

1.3.2 Bornes en performance pour l'agrégation d'estimateurs

A la suite des résultats que nous avons obtenus avec Y.Amit de façon en partie heuristique, nous nous intéressons dans le dernier chapitre au contrôle de l'erreur de généralisation de classifieurs obtenus par agrégation à partir de quantités purement empiriques. Nous développons trois points de vue donnant lieu à trois types de bornes.

Pour simplifier, nous nous plaçons dans le cas où il y a seulement deux classes, -1 et 1; on dispose d'un ensemble de classifieurs $\mathcal{H}$ dont les éléments sont des fonctions $h : \mathcal{X} \to [-1; 1]$ (la classe prédite étant identifiée au signe du classifieur). Pour $\alpha \in \mathcal{M}_1(\mathcal{H})$, l'ensemble des distributions de probabilité sur $\mathcal{H}$, on note H_α le classifieur obtenu par la combinaison convexe α des éléments de $\mathcal{H}$. Toutes les bornes que nous montrons s'appuient sur l'étude des *marges* observées sur l'ensemble d'apprentissage, la marge d'un classifieur composite H_α sur un exemple $Z = (X, Y)$ étant définie par

$$M_\alpha(Z) = H_\alpha(X)Y.$$

La première borne que nous montrons est inspirée des travaux de Shapire et al. [54] et de Breiman [18], dont elle peut être considérée comme une généralisation.

Nous notons $F_\alpha(t)$ la fonction de répartition de la variable M_α et $\widehat{F}_\alpha^N$ sa version empirique à partir d'un échantillon de taille N. Alors, sous l'hypothèse que $\mathcal{H}$ est fini de cardinal L, nous montrons que, sauf sur un ensemble d'échantillons d'apprentissage de probabilité inférieure $\delta > 0$, pour tous $\theta \in [-1;1]$ et $\Delta \in [0,2]$, on a l'inégalité

$$F_\alpha(\theta - \Delta) \leq \widehat{F}_\alpha^N(\theta) + \mathcal{O}\left\{ \sqrt{\widehat{F}_\alpha^N(\theta)} \left(\frac{\log L \log N + \log(1/\delta)}{\Delta^2 N} \right)^{1/2} + \left(\frac{\log L \log N + \log(1/\delta)}{\Delta^2 N} \right) \right\}. \tag{1.25}$$

L'intérêt de cette inégalité est qu'elle fait apparaître une transition de régime pour les petites valeurs de la fonction de répartition empirique: on retrouve pour $\widehat{F}_\alpha^N(\theta) = 0$ l'inégalité montrée par Breiman, et pour $\widehat{F}_\alpha^N(\theta) = \mathcal{O}(1)$ celle de Shapire et al. L'interpolation obtenue entre ces deux types de régime donne ainsi une borne plus précise. Nous donnons ensuite une généralisation de cette borne au cas où $\mathcal{H}$ n'est plus supposé fini, mais de dimension de Vapnik-Chervonenkis d finie, avec l'hypothèse supplémentaire que les éléments de $\mathcal{H}$ sont à valeurs dans $\{-1;1\}$ (au lieu de $[-1;1]$). La démonstration de cette extension suit en tous points celle de Shapire et al. dans [54].

Le deuxième point de vue que nous étudions est le contrôle de l'erreur de généralisation en utilisant seulement les statistiques des premiers et deuxièmes moments de la distribution des marges, par une inégalité telle que (1.24), comme cela a été suggéré au chapitre 4. Il est à noter que comme ces bornes sont basées sur la simple inégalité de Chebychev, elles donneront, asymptotiquement quand la taille de l'échantillon devient grand, un contrôle moins précis que les bornes de type (1.25) s'appuyant sur la fonction de répartition des marges, qui contient plus d'informations. Cependant, on peut espérer que les bornes obtenues en utilisant seulement les deux premiers moments de cette distribution seront plus robustes, en particulier pour les petits ensembles d'apprentissage, et, partant, que les méthodes de construction de classifieurs utilisant les moments empiriques correspondants (telle la «minimisation séquentielle de la variance» proposée au chapitre 4) seront également plus robustes.

Dans la section correspondante du chapitre, nous établirons à partir d'outils très classiques des inégalité de contrôle pour l'estimation des premiers et seconds moments, uniforme sur l'ensemble des combinaisons convexes de classifieurs, ce qui permet d'avoir un contrôle correspondant de la borne obtenue par l'inégalité de

Chebychev, uniformément sur tous les classifieurs composites. Nous proposons de contrôler les déviations de façon absolue (inégalité de type Hoeffding) ou relative à la variance (de type Bernstein). En effet, il est suggéré dans le chapitre 4 que la réduction de la variance de H_α donne aussi lieu à une estimation plus précise des moments. Nous discuterons dans quelle mesure cela peut être valable uniformément sur l'ensemble des classifieurs composites.

Enfin, le troisième paragraphe du chapitre 5 propose encore un autre point de vue possible, fondé sur le principe de «marge minimax». Plusieurs auteurs, partant des inégalités fondées sur la répartition empirique des marges (1.25), ont proposé de rechercher le classifieur composite H_α dont la marge minimum empirique observée sur les exemples d'apprentissage soit la plus haute possible. Dans [54], il est suggéré que l'algorithme de Boosting réalise cette optimisation de façon approchée. Dans [18], Breiman construit des algorithmes spécialement dédiés à cette tâche.

Nous commençons par rappeler les liens, déjà soulignés par les auteurs cités ci-dessus, entre le principe de marge minimax et les problèmes d'optimisation apparaissant dans le cadre des Support Vector Machines [26], et d'autre part, son équivalence avec la recherche de stratégies d'équilibre en théorie des jeux. Ceci nous amènera à proposer un contrôle de l'erreur de généralisation du classifieur obtenu à partir du nombre de «vecteurs de support», lorsque ceux-ci sont correctement définis, par analogie aux Support Vector Machines. Nous montrons ensuite que la stratégie de Blackwell, un algorithme classique de théorie des jeux dont la propriété importante est d'être asymptotiquement «sans regret», peut être utilisée pour construire un estimateur composite qui réalise la marge minimax de façon approchée.

Chapitre 2

The “progressive mixture” estimator for regression trees

RÉSUMÉ – Nous donnons une version, adaptée à un cadre de régression, de l'estimateur dit de “mélange progressif” introduit par O.Catoni [6]. De façon analogue à Catoni, nous donnons une borne à horizon fini pour la perte de Kullback-Leibler de l'estimateur dans ce cadre. Nous explicitons la forme de cette borne dans le cas d'arbres de régression, présentons une variante dans un cadre étendu, et proposons une méthode de calcul approché par algorithme de Metropolis.

ABSTRACT – We present a version of O.Catoni's “progressive mixture estimator” [6] suited for a general regression framework. Following basically Catoni's steps, we derive strong non-asymptotic upper bounds for the Kullback-Leibler risk in this framework. We give a more explicit form for this bound when the models considered are regression trees, present a modified version of the estimator in an extended framework and propose an approximate computation using a Metropolis algorithm.

2.1 Introduction

The so-called “progressive estimator” was introduced by Catoni [22](1997) using ideas inspired by the recent work of Willems et al. [59, 60] on universal coding. The principle of a progressive estimator has also been also proposed independently by Barron and Yang [5, 7]. One of the main attracting features of this estimator is that we can easily derive strong non-asymptotic bounds on its mean Kullback-Leibler risk in an extremely general framework.

Our goal in this paper is to present a version of this estimator first in a general regression estimation framework, then more precisely for classification and regression trees (CARTs). As the computation of this estimator involves performing sums of a Bayesian type on a very large (and possibly non-finite) set of models, we are naturally led in practice to compute these sums approximately using a Monte Carlo algorithm. A very similar method (a Bayesian search among CARTs using a stochastic procedure) has been proposed by Chipman et al. [25] (1997). It is interesting to note that if we start from a different theoretical point of view, which allows us to derive non asymptotic bounds on the Kullback-Leibler risk, we are led to a very similar algorithm (see discussion at the end of the paper). On the other hand, the work of Chipman et al. suggested to us the idea of a data-dependent Bayesian prior on the set of models which we develop in section 5.

Finally we also want to point out the relation of this algorithm of random walk in the model tree space with the method of tree selection by local entropy minimization developed by Amit, Geman, and Wilder [2, 4]. This work has been a large source of inspiration and in some regards, the local entropy minimization can be seen as the deterministic (or "zero-temperature") analogous of the Monte Carlo method presented here. This will also be discussed in some more detail in section 7.

The structure of the paper is as follows. In section 2, we present a version of the progressive mixture estimator for regression estimation in a very general framework. Section 3 focuses on classification and regression trees and gives a precise and explicit form of the estimator in this case. Section 4 briefly deals with the case of a "fixed-design" tree where an exact computation of the estimator can be performed thanks to the tree weighting recursive algorithm developed by Tjalkens, Willems and Shtarkov in [59] (other authors have also used this technique for classification trees in a different framework, see [42]). Section 5 presents an extension of the framework to a possibly uncountable family of model trees, using a data-dependent prior, and we show for a precise and non trivial example that the estimator thus obtained still achieves the minimax rate of convergence. In section 6, we explain how to construct a Monte Carlo chain to obtain an approximate computation of the estimator for general tree models. Section 7 concludes the paper with a discussion of the results.

2.2 Catoni's progressive estimator in a regression framework

In this section we present a version of Catoni's estimator adapted for regression estimation purposes.

Let us specify the framework and notations. Let $(X_i, Y_i)_{i=1}^N = (Z_i)_{i=1}^N = Z_1^N$ be a sequence of random variables, taking their values in some measurable space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ with probability distribution P_N. The only hypothesis we will make on P_N is that it is exchangeable, i.e. that this distribution is invariant under any permutation of the variables Z_i. Our goal in a regression framework is, given a sample $z_2, \dots, z_N$, to get an estimation of the conditional probability of a new observation Y_1 given X_1 (The fact that this "new" observation is actually given the index 1 is purely for later notational convenience and should not be misleading). We thus want to estimate $P_N(Y_1|X_1, Z_2^N)$. This can include any case where the order of the observations (including the test example Z_1) is not of importance, because we can then re-draw them in a random order. This case is of course of but minor practical relevance since in such a predictive framework, the test example is usually given separately and cannot be drawn at random (in which case the hypothesis of exchangeability has to hold fully). However, it covers for example the customary procedure used for classifier testing, when one has a fixed database of examples, that is split at random between a set used to train the classifier and a set used to validate its accuracy.

We will assume that there exists a regular version of this conditional probability, and, more generally throughout this paper, we will assume that all of the conditional probabilities that we are dealing with have a regular version, allowing us to make use of all standard integration properties. For that purpose we will assume that $\mathcal{Z}$ is a Borel space.

To begin with, we will split the sample data set z_2^N into two subsamples $E = z_{L+1}^N$ called "estimation set" and $T = z_1^L$ called "test set" for some fixed integer $L > 0$ (note that we consider the new observation z_1 to be part of the test set). The reason for this will become clearer in the sequel and the issue of the choice of L will be discussed later on. Note that the ordering of the variables is purely arbitrary since the probability distribution P according to which they have been drawn is exchangeable.

We then assume that we have at our disposal a countable family of estimators

$(Q_m)_{m\in\mathcal{M}}$, that is, a family of conditional probabilities depending on the observations of the estimation set $Q_m(Y \in \cdot\,|X, E)$, also denoted by $Q_m^E(Y \in \cdot\,|X)$ for a more compact writing (In a more general way in this paper we will often use a superscript E for probability distributions depending on the estimation set, thus writing equally $P^E(\cdot|\cdot)$ for $P(\cdot|\cdot, E)$). We will speak of $m \in \mathcal{M}$ as a "model".

Our purpose is to find an estimator in this family which will minimize (approximately) the Kullback-Leibler (K-L) risk with regard to $P_N(Y_1|X_1, Z_2^N)$, i.e. find a $m \in \mathcal{M}$ achieving

$$\inf_{m\in\mathcal{M}} E_{P_N} H\left(P_N\left(Y_1 \in \cdot\,|X_1, Z_2^N\right), Q_m^E\left(Y_1 \in \cdot\,|X_1\right)\right),$$

where H is the Kullback-Leibler divergence: for ρ, ν two probability distributions on the space $\mathcal{Y}$:

$$H(\rho,\nu) = \begin{cases} \int \log \frac{d\rho}{d\nu} d\rho & \text{if } \rho \ll \nu \\ +\infty & \text{otherwise.} \end{cases}$$

To get a solution to our estimation problem we will actually not perform a model selection as first announced, but rather build a composite estimator for which we will be able to explicitly upper bound the Kullback-Leibler risk.

Suppose that we have chosen some probability distribution π on the model set $\mathcal{M}$. If we keep the estimation set E "frozen", we can consider the estimator probabilities $Q_m^E(Y|X)$ as simple product (conditional) probabilities on the test set T. We then define the mixture conditional probability on the test set $Q_{\text{mix.}}^E$ obtained by summing these product probabilities Q_m^E weighted by the prior π:

$$Q_{\text{mix.}}^E\left(dY_1^L|x_1^L\right) \triangleq \sum_{m\in\mathcal{M}} \pi(m) \prod_{i=1}^{L} Q_m^E\left(dY_i|x_i\right).$$

We eventually build Catoni's progressive mixture estimator $Q_\pi(Y_1 \in \cdot\,|x_1, z_2^N)$ also denoted by $Q_\pi^E(Y_1 \in \cdot\,|x_1, z_2^L)$ in the following way:

$$Q_\pi(dY_1|x_1, z_2^N) = Q_\pi^E\left(dY_1|x_1, z_2^L\right) \triangleq \frac{1}{L}\sum_{M=1}^{L} Q_{\text{mix.}}^E\left(dY_1|x_1, z_2^M\right). \tag{2.1}$$

In order to make the last formula somewhat clearer, let us consider for a moment the simple case where the space $\mathcal{Y}$ is discrete (This is actually the case that will be considered in the following sections, for classification problems). In that case the definition (2.1) yields the following expansion (everywhere the denominator does not vanish):

$$Q_\pi(Y_1|x_1, z_2^N) = \frac{1}{L} \sum_{M=1}^{L} \frac{\displaystyle\sum_{m\in\mathcal{M}} \pi(m) \prod_{i=1}^{M} Q_m^E\left(Y_i|x_i\right)}{\displaystyle\sum_{m'\in\mathcal{M}} \pi(m') \prod_{j=2}^{M} Q_{m'}^E\left(Y_j|x_j\right)}. \tag{2.2}$$

Let us point out that each individual term of the sum over M is nothing but a (predictive) Bayesian estimator based on the probabilities $Q_m^E(dY_1|X_1)$ (when the estimation sample E is kept frozen), using the prior π. Thus Q_π^E is a Cesàro mean of Bayesian estimators with a sample of growing size z_2^M. The subsample E is used for estimation within each model $m \in \mathcal{M}$ ("estimation set") whereas the other subsample z_2^L is then used to "choose" between the models in a Bayesian way ("test set").

We now state a theorem upper-bounding the risk for this estimator.

Theorem 1. *With the previous assumptions we get the inequality:*

$$\begin{aligned} E_{P_N} H &\left(P_N\left(Y_1 \in \cdot\,|X_1, Z_2^N\right), Q_\pi\left(Y_1 \in \cdot\,|X_1, Z_2^N\right)\right) \\ &\leq \inf_{m\in\mathcal{M}} \left\{ E_{P_N} H\left(P_N\left(Y_1 \in \cdot\,|X_1, Z_2^N\right), Q_m\left(Y_1 \in \cdot\,|X_1, Z_{L+1}^N\right)\right) \right. \\ &\quad \left. + \frac{1}{L}\log\frac{1}{\pi(m)} \right\}. \end{aligned} \tag{2.3}$$

Proof. There is actually little to be changed in comparison to [22] so we will follow quite closely Catoni's steps.

Assume that the estimators $Q_m^E(Y_1 \in \cdot\,|x_1)$ have densities $q_m^E(y_1|x_1)$ with respect to some dominating conditional measure μ. We similarly will denote by lower-case letters the densities w.r.t. μ of the other probability distributions. In case there is no "obvious" choice for μ we can build the following dominating measure depending on the estimation set E:

$$\mu^E\left(Y_1 \in \cdot\,|x_1\right) = \sum_{m\in\mathcal{M}} 2^{-\sigma(m)} Q_m^E\left(Y_1 \in \cdot\,|x_1\right),$$

where σ is some injective mapping of $\mathcal{M}$ into $\mathbb{Z}^+$.

Take an $m \in \mathcal{M}$; if $E_{P_N} H\left(P_N\left(Y_1 \in \cdot\, |X_1, Z_2^N\right), Q_m^E\left(Y_1 \in \cdot\, |X_1\right)\right) = +\infty$, the inequality is true for this m; else we consider the difference (whose second term is now finite)

$$\begin{aligned} E_{P_N} H\left(P_N\left(Y_1 \in \cdot\, |X_1, Z_2^N\right), Q_\pi^E\left(Y_1 \in \cdot\, |X_1, Z_2^L\right)\right) \\ - E_{P_N} H\left(P_N\left(Y_1 \in \cdot\, |X_1, Z_2^N\right), Q_m^E\left(Y_1 \in \cdot\, |X_1\right)\right) \\ = -E_{P_N} \log \frac{q_\pi^E\left(Y_1|X_1, Z_2^L\right)}{q_m^E\left(Y_1|X_1\right)}. \end{aligned}$$

Because $-\log$ is a convex function we have

$$-E_{P_N} \log \frac{q_\pi^E\left(Y_1|X_1, Z_2^L\right)}{q_m^E\left(Y_1|X_1\right)} \leq -\frac{1}{L} \sum_{M=1}^{L} E_{P_N} \log \frac{q_{\text{mix.}}^E\left(Y_1|X_1, Z_2^M\right)}{q_m^E\left(Y_1|X_1\right)}. \tag{2.4}$$

Now using the hypothesis that P_N is an exchangeable probability distribution, we can swap the role played by X_1 and X_{M+1} in the Mth term of the sum (formally replacing X_{L+1}, Y_{L+1} by X_1, Y_1 whenever necessary).

We thus get to:

$$-E_{P_N} \log \frac{q_\pi^E\left(Y_1|X_1, Z_2^L\right)}{q_m^E\left(Y_1|X_1\right)} \leq -\frac{1}{L} E_{P_N} \sum_{M=1}^{L} \log \frac{q_{\text{mix.}}^E\left(Y_{M+1}|X_{M+1}; Z_2^M\right)}{q_m^E\left(Y_{M+1}|X_{M+1}\right)},$$

where by definition

$$q_{\text{mix.}}^E\left(y_{M+1}|x_{M+1}, z_2^M\right) = \frac{q_{\text{mix.}}^E(y_2^{M+1}|x_2^{M+1})}{q_{\text{mix.}}^E(y_2^M|x_2^M)}.$$

Let us point out that we have no more conditioning with respect to x_{M+1} in the last denominator because of the relation coming straightforward from the definition of $Q_{\text{mix.}}^E$:

$$\int_{\mathcal{Y}} q_{\text{mix.}}^E(y_2^{M+1}|x_2^{M+1})\, d\mu(y_{M+1}) = q_{\text{mix.}}^E(y_2^M|x_2^M). \tag{2.5}$$

Thus the sum in (2.4) reduces to the first component of its last term since the other terms cancel each other successively, leading to:

$$
\begin{aligned}
-E_{P_N} \log \frac{q_\pi^E\left(Y_1|X_1,Z_2^L\right)}{q_m^E\left(Y_1|X_1\right)} &\leq -\frac{1}{L} E_{P_N} \log \frac{q_{\text{mix.}}^E\left(Y_1^L|X_1^L\right)}{q_m^E\left(Y_1^L|X_1^L\right)} \\
&\leq -\frac{1}{L} E_{P_N} \log \frac{\displaystyle\sum_{m'\in\mathcal{M}} \pi(m') q_{m'}^E\left(Y_1^L|X_1^L\right)}{q_m^E\left(Y_1^L|X_1^L\right)} \\
&\leq -\frac{\log \pi(m)}{L}.
\end{aligned}
$$

□

It has been pointed out that an "aesthetic" flaw of the progressive mixture estimator is that it is not a symmetric function of the data (different orderings give different estimators), because of the arbitrary division of the sample in two sets and because of the "progressive" sum in the definition of the estimator, which is a Cesàro average of Bayesian estimators with a growing sample. The reason for this apparently surprising progressive sum is to be able to bound the Kullback-Leibler distance between probability distributions on the only variable Z_1 by the K-L distance between full multivariate distributions on the whole test set (for which the idea of a mixture distribution comes from the compression and information theory literature), as appears in the proof of the theorem. However, it is possible to define theoretically a fully symmetric estimator by averaging $Q_\pi(dY1|x_1, z_2^N)$ over all the $(N-1)!$ possible orderings of the sample z_2^N, that will satisfy the same majorations as Q_π (as can be seen using once more the convexity of the K-L distance $H(\rho, \mu)$ with respect to μ).

2.3 Application to a CART model

In this section we deal with a classification problem, that is, the variable Y will take its values in $\mathcal{Y} = \{0, 1, \dots, c\}$ (thus there are $(c+1)$ distinct classes and c is the number of *free parameters* for a distribution on $\mathcal{Y}$), and we want to "recognize" the observation X taking its values in a certain space $\mathcal{X}$, deciding to which class it belongs, having already observed a "learning set" z_2^N. In order to do that we want to make use of a classification tree using the "20-questions game" principle: suppose we have at our disposal a countable set of "questions" $\mathcal{Q}$ we can ask on the observation X, to which we can only answer by 0 or 1 (for example we can think of X as a finite or infinite binary sequence, the questions being of the form "what is the n-th bit of X?"). Having asked a first question, we are then allowed to choose a second one depending on the first answer, and so on. At each step we can decide whether to ask a new question or to stop and guess X's class (or, alternatively, to estimate the probability of X belonging to a certain class, if we prefer a regression framework). If the rule is that we can only ask at most d questions, all the possible sequences of questions and answers can be represented as a complete binary tree of depth at most d, whose internal nodes are labeled with questions (CART tree model).

Our set of models $\mathcal{M}$ will therefore be a set of couples (T, F_T), where T is a tree and $F_T = (f_s)_{s\in \mathrm{int}(T)}$ the set of questions indexed by the internal nodes of T. A question $f \in \mathcal{Q}$ is nothing but the indicator of a certain measurable set $A_f \subset \mathcal{X}$. We will think of a binary tree T as a subset of $\Gamma = \{\emptyset\} \cup \{0,1\}^*$ (the set of all finite, possibly empty strings formed of zeroes and ones) such that if $s \in T$, every ancestor (that is, every prefix string) t of s also belongs to T. By definition, $\emptyset$ is the root of the tree, and if s is a node, its two sons are $s0$ and $s1$. In our case we will only consider complete binary trees (such that every $s \in T$ either is a terminal node or has two sons). If m is such a model and x is an observation, we denote by $m(x)$ the terminal node associated to x by m (obtained by asking questions on x and letting him follow the tree branches until we reach a terminal node of m). Also, we will denote by ∂m the set of terminal nodes (or leaves) of the model tree m.

Given $m = (T, F_T)$ we now have to define our set of estimators $Q_m(Y_1|X_1, E)$. For the remainder of the paper, when dealing with tree models we will take for Q_m a conditional Laplace estimator at every terminal node: given a terminal node $s \in T$ we define the following counters on the estimation set $E = z_{L+1}^N$:

$$n_s^i \triangleq \sum_{k=L+1}^{N} \mathbf{1}\{m(x_k) = s, y_k = i\} \text{ for } i = 0, \ldots, c,$$

where $\mathbf{1}_A$ denotes the indicator function of the set A. Then, if $m(X_1) = s$, we take for $Q_m^E(Y_1|X_1)$ a multivariate distribution defined by

$$Q_m^E(Y = i|m(X) = s) \triangleq \frac{n_s^i + 1}{\sum_i n_s^i + c + 1}.$$

Let us then denote by $M_{m,\theta^m}(Y|X)$ the general conditional multivariate distribution based on the labeled tree m, where $\theta^m = (\theta_s)_{s\in\partial m}$ is a set of real vectors of parameters all belonging to the c-dimensional simplex $\mathcal{S}_c = \{\theta \in [0,1]^{(c+1)} | \sum_i \theta_i = 1\}$, and indexed by the terminal nodes of m, defined by

$$M_{m,\theta^m}(Y = i|m(X) = s) = (\theta_s)_i$$

(where $(\theta_s)_i$ denotes the i-th coordinate of the vector parameter θ_s).

Once again following very closely [22], we give the following upper bound for the risk of the Laplace estimator:

Theorem 2. *For any exchangeable distribution P_N on the variables $Z_1^N \in (\{0,\ldots,c\} \times \mathcal{X})^N$, for a given model m, the conditional Laplace estimator Q_m satisfies*

$$\begin{aligned} &E_{P_N} H\left(P_N\left(Y_1|X_1, Z_2^N\right), Q_m\left(Y_1|X_1, Z_{L+1}^N\right)\right) \\ &\qquad\qquad \leq \inf_{\theta^m} E_{P_N} H\left(P_N\left(Y_1|X_1, Z_2^N\right), M_{m,\theta^m}\left(Y_1|X_1\right)\right) + \frac{c\,|\partial m|}{N-L+1}, \end{aligned}$$

where $|\partial m|$ is the number of terminal nodes of the tree model m.

Proof. To shorten the equations below we will write in the sequel equally X_{N+1}, Y_{N+1} for X_1, Y_1.

For a given terminal node $s \in \partial m$, let us first introduce the modified counters a_s^i, $i = 0,\ldots,c$:

$$a_s^i = \sum_{k=L+1}^{N+1} \mathbf{1}\{m(X_k) = s, Y_k = i\} = n_s^i + \mathbf{1}\{m(X_1) = s, Y_1 = i\},$$

and denote by $\Sigma_s = \sum_i a_s^i$ the total number of examples at node s.

With these new counters we get

$$Q_m^E\left(Y_1 = i|m(X_1) = s; Z_{L+1}^N\right) = \frac{a_s^i}{\Sigma_s + c},$$

thus

$$\begin{aligned} &E_{P_N} - \log Q_m\left(Y_1|X_1, E\right) \\ &\quad = -\frac{1}{N-L+1} E_{P_N} \sum_{j=L+1}^{N+1} \log Q_m\left(Y_j|X_j; Z_k, k \neq j, L+1 \leq k \leq N+1\right) \\ &\qquad\qquad\qquad\qquad = -\frac{1}{N-L+1} E_{P_N} \sum_{s\in\partial m} \sum_{i=0}^{c} a_s^i \log \frac{a_s^i}{\Sigma_s + c}, \end{aligned}$$

where the first inequality follows from the exchangeability of the distribution P_N. In the second equality we put $0 \log 0 = 0$ if necessary. Now for a fixed $s \in \partial m$, if at least one example reached this node, that is, if $\Sigma_s > 0$, we have

$$\begin{aligned}
\sum_{i=0}^{c} a_s^i \log \frac{a_s^i}{\Sigma_s + c} &= \sum_{i=0}^{c} a_s^i \log \frac{a_s^i}{\Sigma_s} - \Sigma_s \log\left(1 + \frac{c}{\Sigma_s}\right) \\
&= - \inf_{\theta \in \mathcal{S}_c} \left[- \sum_{i=0}^{c} a_s^i \log \theta_i \right] - \Sigma_s \log\left(1 + \frac{c}{\Sigma_s}\right) \\
&\geq - \inf_{\theta \in \mathcal{S}_c} \left[- \sum_{i=0}^{c} a_s^i \log \theta_i \right] - c.
\end{aligned}$$

Now this last inequality obviously remains true if $a_s^i = 0$ for all i and thus

$$\begin{aligned}
&E_{P_N} - \log Q_m \left(Y_1 | X_1, E\right) \\
&\qquad \leq \frac{1}{N-L+1} E_{P_N} \left(\inf_{\theta^m} - \log M_{m,\theta^m}^{\otimes (N-L+1)} \left(Y_{L+1}^{N+1} | X_{L+1}^{N+1}\right) + c\,|\partial m| \right) \\
&\qquad \leq \inf_{\theta^m} E_{P_N} - \log M_{m,\theta^m} \left(Y_1 | X_1\right) + \frac{c\,|\partial m|}{N-L+1}.
\end{aligned}$$

□

Now suppose we have chosen some *a priori* measure π on the tree models $\mathcal{M}$ ($\mathcal{M}$ is countable since $\mathcal{Q}$ is countable); we thus get as a straightforward consequence of theorems 1 and 2:

Corollary 1. *With the previous choice of the estimators Q_m for labeled tree models, and a prior π on $\mathcal{M}$ we get for the estimator Q_π defined in section 2 the inequality*

$$\begin{aligned}
&E_{P_N} H \left(P_N \left(Y_1 | X_1, Z_2^N\right), Q_\pi \left(Y_1 | X_1, Z_2^N\right)\right) \\
&\qquad \leq \inf_{m,\theta^m} \Big\{ E_{P_N} H \left(P_N \left(Y_1 | X_1, Z_2^N\right), M_{m,\theta^m} \left(Y_1 | X_1\right)\right) \\
&\qquad\qquad + \frac{\log \pi(m)^{-1}}{L} + \frac{c\,|\partial m|}{N-L+1} \Big\}.
\end{aligned} \tag{2.6}$$

For example, let us take for prior π the distribution of the genealogy tree of a branching process of parameter $\rho \leq 1/2$ (an individual gets two sons with probability ρ, or dies without descendents with probability $1-\rho$) — this prior will be used several times in the next sections. This prior is explicitly given by

$$\pi(m) = \frac{(\rho(1-\rho))^{|\partial m|}}{\rho},$$

so that the first penalization term in equation (2.6) is

$$\frac{\log \pi(m)^{-1}}{L} = -\frac{|\partial m|(\log \rho(1-\rho))}{L} + \frac{\log \rho}{L},$$

thus the natural choice for L (the size of the test sample) in this case is of the form kN, with $k \in [0,1]$ such that the two penalization terms of equation (2.6) are balanced.

Remark: Although the K-L risk is interesting in itself and has been customarily used for classification tree construction (through "local entropy minimization", see e.g. [2, 4]), one may be legitimately interested by the true classification error of the estimator. A thorough discussion on a precise control of this error is beyond the scope of this paper; we simply recall some basic inequalities allowing to control the classification error through the K-L risk.

Let us forget for a moment the conditioning with respect to X and let P be a probability distribution on $\mathcal{Y} = \{0, \ldots, c\}$. If P is the "true" distribution, the best classification rule is to predict the class of highest probability. Let $P_* = \max_i P(i)$, then the lowest attainable average classification error is $L_* = 1 - P_*$. Now suppose we have some estimate Q of P and predict the class $\alpha = \operatorname{argmax}_i Q(i)$; the average classification error for this rule is $L(Q) = 1 - P(\alpha)$. Then the following elementary inequalities hold:

$$L(Q) - L_* \leq \sum_{i=0}^{c} |P(i) - Q(i)| \leq \sqrt{2H(P,Q)}.$$

The first inequality can be found in [32] and the second one for example in [28], p.300.

Now if me make this reasoning conditional to some X, taking the expectation over X of the above quantities we get (somewhat informally)

$$EL(Q(Y|X)) - EL_*(X) \leq E\left(\sqrt{2H(P,Q|X)}\right) \leq \sqrt{2}\sqrt{E(H(P,Q|X))}.$$

Using the bounds derived for the K-L distance, we can thus obtain a coarse upper bound for the difference between the classification error obtained with the estimator $Q(Y|X)$ and the optimal average error $EL_*(X)$.

2.4 Exact computation for a fixed-design tree

In this section we will assume that there is actually no choice in the question to be asked at each internal node of the model tree. In this case we will speak of a "fixed-design" tree model; the set of models is then the set of all complete subtrees of the maximal complete binary tree of depth d denoted by $\mathcal{T}$. The most classical example for that is a "context tree" model where X is a binary string of length d and all nodes at depth r are labeled with the question "what is the r-th bit of X?". Recall that since we are dealing with a discrete space $\mathcal{Y}$, the developed formula for the progressive mixture estimator is given by equation 2.2.

We will take for prior π the distribution of the genealogy tree of a branching process of parameter ρ stopped at depth d, meaning that every node of the tree will have two sons with probability ρ or will be a terminal node with probability $1-\rho$, except for the nodes at depth d which are always terminal. In this framework we can use an efficient algorithm found by Willems, Shtarkov and Tjalkens [59] for universal coding using weighted context trees to compute a sum of the form

$$\sum_{m \in \mathcal{M}} \pi(m) \prod_{i=1}^{M} Q_m^E\left(y_i | x_i\right). \tag{2.7}$$

Namely, let us define for every node (internal or terminal) s of $\mathcal{T}$ the counters n_s^i, b_s^i for $i = 0, \dots, c$ containing the number of examples x_k of class i whose "path" passes through node s (i.e. for which s is an ancestor of or is equal to $\mathcal{T}(x_k)$), respectively for the estimation set $E = z_{L+1}^N$ and the truncated test set z_1^M:

$$\left\{ \begin{array}{l} n_s^i = \displaystyle\sum_{j=L+1}^{N} \mathbf{1}\{s \preceq \mathcal{T}(x_j), y_j = i\} \\ b_s^i = \displaystyle\sum_{j=1}^{M} \mathbf{1}\{s \preceq \mathcal{T}(x_j), y_j = i\} \end{array} \right. ,$$

where $\preceq$ means "to be an ancestor of or to be equal to".

Let us now define the local (Laplace) estimator at node s (estimated using E, and applied to the set z_1^M):

$$L(s) = \prod_{i=0}^{c} \left(\frac{n_s^i + 1}{\sum_i n_s^i + c + 1} \right)^{b_s^i}$$

(note that any other local estimator could actually be taken here instead of the Laplace estimator; this is only for the sake of simplicity and coherence with the previous section).

By backward induction on the depth of the nodes $s \in \mathcal{T}$ we build the quantities

$$\sigma(s) = \begin{cases} L(s) & \text{if s is a terminal node of } \mathcal{T} \\ (1-\rho)L(s) + \rho\,\sigma(s0)\,\sigma(s1) & \text{if s is an internal node of } \mathcal{T} \end{cases} \tag{2.8}$$

(we recall that $s0$ and $s1$ are the two sons of the node s).

It has been proved in [59] that with this construction

$$\sigma(\emptyset) = \sum_{m \in \mathcal{M}} \pi(m) \prod_{i=1}^{M} Q_m^E\,(y_i|x_i)\,.$$

To compute all the terms of the Cesàro sum defining the progressive estimator, we have to perform $2L$ such sums, but having performed one sum the next one is obtained only by adding an observation (x_i, y_i); so it is sufficient to update the calculations for the counters and $\sigma(s)$ along the branch it follows, giving rise to d computations. However, to determine completely the distribution Q_π we have to consider every 2^d possibilities for $\mathcal{T}(x_1)$ (that is, it can be any terminal node of $\mathcal{T}$). In conclusion the complexity of the algorithm is of order $d2^dN$; this is to be compared to the number of models considered which is greater than $2^{2^{d-1}}$.

It has to be noted that the form of the prior π is crucial in order this algorithm to be used (namely, the principle is that it allows a nice factorization of equation (2.7)). However, we can make a straightforward extension to a more general set of priors by allowing the branching parameter ρ to depend on the node s considered in the tree, in which case one has just to replace the constant ρ by the the local branching parameter ρ_s in equation (2.8). This set of priors is large enough to allow a wide range of flexibility, for example, one can choose to give less *a priori* weight to "shallow" trees which we expect not to be so relevant, by taking ρ_s close to 1 for those nodes s that are close to the root.

2.5 Data-dependent scaling for continuous-valued observations

In this section we will consider the case when X takes its values in $\mathbb{R}^p$, where each coordinate corresponds to a "feature" of the observation. Denoting by x^k the k-th coordinate of x, we will restrict ourselves to questions of the form "$x^k < a$?" depending on k and a. We can thus represent all possible questions by couples $(k, a) \in \{1, \dots, p\} \times \mathbb{R}$. In principle it would be possible to discretize the values taken by a over $\mathbb{R}$, for instance by restricting $a \in \mathbb{Q}$, in order to get a countable set $\mathcal{Q}$ of questions. If we know nothing about the range of values taken by x however, it seems clear that choosing some *a priori* distribution on $\mathbb{Q}$ would give quite bad results in practice unless we have a huge number of examples.

Hence we propose to modify slightly the construction of Catoni's estimator by letting the prior depend on the data. A model is now a triplet $m = (T, K_T, A_T)$ where T is a tree structure, and K_T,A_T are families of integers and real numbers, respectively, both indexed by the internal nodes of T. A quite natural way to define a distribution on the set of models $\mathcal{M}$ is to choose a distribution on the tree structure T, then on K_T given T, then on A_T given (T, K_T). In the sequel we will denote by $\tau = (T, K_T)$ the tree with its internal nodes labeled with the "type" of question to ask (i.e., on which coordinate of X the question should be asked).

In our case we will choose the two first distributions in a deterministic (data-independent) way which we will denote by $\pi(\tau)$. Now for the last one we notice that in fact the value of the estimator $Q^E_m(y_1|x_1, z_2^L)$ depends only on the statistic $m(x_1), \dots, m(x_n)$, that is, on the way the sample x_1^N is "split" across the tree. In other words, we can group the possible values of A_T into statistically (with regard to the sample) equivalent sets of questions, corresponding to the different possible splittings of the sample x_1^N (A "splitting" is here a mapping from the set $\{x_i; i = 1, \dots, N\}$ to the set $\partial\tau$ of the leaves of the tree τ). Let $\Delta^\tau(x_1^N)$ be the set of all such possible splittings when A_T varies in $\mathbb{R}^{|\partial\tau|}$. What we propose is simply to give equal weight to each one of these splittings. We thus define the data-dependent mixture probability

$$\widetilde{Q}^E_{\text{mix.}}(y_1^L|x_1^L) = \sum_{\tau=(T,K_T)} \pi(\tau) \sum_{\sigma \in \Delta^\tau(x_1^N)} \frac{1}{|\Delta^\tau(x_1^N)|} \prod_{i=1}^{L} Q^E_{\tau,\sigma}(y_i|x_i),$$

where $Q^E_{\tau,\sigma}$ denotes the common value of the estimators Q^E_m for the tree τ and the splitting σ.

It is important to note here that with this definition, the value of the mixture probability on a single point $\widetilde{Q}^E_{\text{mix.}}(y_i|x_1^L)$ still depends on the *whole* X-sample x_1^N, since the data-dependent prior still depends on the number of different ways we can split the *entire* X-sample, and that is why the conditioning will always involve x_1^L. (Recall that on the other hand, the dependency of $\widetilde{Q}^E_{\text{mix.}}$ with respect to the "estimation" examples ranging from $L+1$ to N, is denoted by the superscript E).

We can then define the data-dependent progressive mixture estimator $\widetilde{Q}_\pi$ by equation (2.1) just replacing $Q^E_{\text{mix.}}$ with $\widetilde{Q}^E_{\text{mix.}}$ and introducing the suitable modification as for the conditioning, that is:

$$\widetilde{Q}^E_\pi \left(dY_1|x_1, z_2^L\right) = \frac{1}{L} \sum_{M=1}^{L} \widetilde{Q}^E_{\text{mix.}} \left(dY_1|x_1^L, y_2^M\right). \tag{2.9}$$

We now give a result similar to theorem 1:

Theorem 3. *With the above definitions, the progressive mixture estimator with data-dependent prior $\widetilde{Q}_\pi$ satisfies, for any exchangeable probability distribution P_N,*

$$\begin{aligned} &E_{P_N} H\left(P_N\left(Y_1 \in \cdot\, |X_1, Z_2^N\right), \widetilde{Q}_\pi\left(Y_1 \in \cdot\, |X_1, Z_2^N\right)\right) \\ &\leq \inf_{m=(\tau, A_T)\in\mathcal{M}} \left\{ E_{P_N} H\left(P_N\left(Y_1 \in \cdot\, |X_1, Z_2^N\right), Q_m\left(Y_1 \in \cdot\, |X_1, Z_{L+1}^N\right)\right) \right. \\ &\qquad\qquad \left. + \frac{1}{L}\log\frac{1}{\pi(\tau)} + \frac{|\partial m|\log(N+1)}{L} \right\}, \end{aligned} \tag{2.10}$$

where $|\partial m|$ denotes the number of terminal nodes of the tree model m.

If we choose for the estimators Q_m the same conditional Laplace estimators as in the previous section, then using theorem 2 we immediately deduce

Corollary 2. *Under the same hypotheses the estimator $\widetilde{Q}_\pi$ satisfies*

$$\begin{aligned} &E_{P_N} H\left(P_N\left(Y_1 \in \cdot\, |X_1, Z_2^N\right), \widetilde{Q}_\pi\left(Y_1 \in \cdot\, |X_1, Z_2^N\right)\right) \\ &\leq \inf_{m,\theta^m} \left\{ E_{P_N} H\left(P_N\left(Y_1 \in \cdot\, |X_1, Z_2^N\right), M_{m,\theta^m}\left(Y_1|X_1\right)\right) \right. \\ &\qquad\qquad \left. + \frac{\log\pi(m)^{-1}}{L} + \frac{c\,|\partial m|}{N-L+1} + \frac{|\partial m|\log(N+1)}{L} \right\}. \end{aligned} \tag{2.11}$$

where M_{m,θ^m} is the conditional multivariate distribution associated with the tree model m and the set of parameters θ^m, defined in the section 2.3.

Proof of theorem 3. We can apply the same first steps as in the proof of theorem 1, (however, in this case, since we are dealing with discrete distributions, we can skip the considerations about relative densities). Namely, let m be a given model, we want to upper-bound the quantity

$$
\begin{aligned}
-E_{P_N} \log \frac{\widetilde{Q}_{\pi}^{E}\left(Y_1|X_1, Z_2^L\right)}{Q_m^E\left(Y_1|X_1\right)} &\leq -\frac{1}{L}\sum_{M=1}^{L} E_{P_N} \log \frac{\widetilde{Q}_{\text{mix.}}^{E}(Y_1|X_1^L, Y_2^M)}{Q_m^E(Y_1|X_1)} \\
&= -\frac{1}{L}\sum_{M=1}^{L} E_{P_N} \log \frac{\widetilde{Q}_{\text{mix.}}^{E}(Y_{M+1}|X_1^L, Y_2^M)}{Q_m^E(Y_{M+1}|X_{M+1})} \\
&= -\frac{1}{L} E_{P_N} \log \frac{\widetilde{Q}_{\text{mix.}}^{E}(Y_1^L|X_1^L)}{Q_m^E(Y_1^L|X_1^L)},
\end{aligned}
$$

where the first inequality follows from the concavity of the logarithm, the following equality is obtained by swapping the role of (X_1, Y_1) and (X_{M+1}, Y_{M+1}) using the exchangeability of P_N (replacing (X_{L+1}, Y_{L+1}) by (X_1, Y_1) if necessary), and the last one by the chain rule.

Finally the following inequality holds:

$$
-\frac{1}{L} E_{P_N} \log \frac{\widetilde{Q}_{\text{mix.}}^{E}\left(Y_1^L|X_1^L\right)}{Q_m^E\left(Y_1^L|X_1^L\right)} \leq -\frac{\log \pi(\tau)}{L} + \frac{E_{P_N} \log |\Delta^\tau(x_1^N)|}{L}.
$$

It can be derived in the following way: for our fixed model $m = (\tau, A_T)$ there exists a splitting $\sigma_0 \in \Delta^\tau(x_1^N)$ which separates the data exactly in the same way as does the tree model m. Thus $Q_m^E(X_1^L|Y_1^L) = Q_{\tau,\sigma_0}^E(X_1^L|Y_1^L)$, and keeping only the corresponding term in the sum defining $\widetilde{Q}_{\text{mix.}}^{E}$ as in the proof of theorem 1, we get the desired inequality.

Now the sample reaching an internal node s is at most of size N and thus can be split at most in $N+1$ ways at this node since we only make use of order statistics for a certain coordinate. Thus $|\Delta^\tau(x_1^N)| \leq (N+1)^{|\text{int}(m)|}$ where $|\text{int}(m)|$ denotes the number of internal nodes of the model tree m; but $|\text{int}(m)| \leq |\partial m|$ since for a complete binary tree $|\partial m| = |\text{int}(m)| + 1$; thus we get to the desired conclusion.

$\square$

This result can be enlarged in several ways. First, one can notice that we can easily extend this result in a more general framework. Namely, assume we have a possibly uncountable family of models $\mathcal{M}$, but that every model can be written in the from $m = (\tau, \lambda_\tau)$ where τ takes its values in some countable set. In the same way as above, we can choose a fixed prior distribution for τ and a data-dependent distribution for λ_τ, grouping the values of λ_τ into statistically equivalent sets to which we assign an equal weight. We can then apply the same reasoning and the main remaining point is to get a bound for $E_{P_N} \log |\Delta^\tau(x_1^N)|$. For instance, if instead of questions of the type "$x^k < a$?" we choose at each node a set of questions of Vapnik-Cervonenkis dimension bounded by D, then we easily come to $\log |\Delta^\tau(x_1^N)| \leq C|\partial m| D \log N$ where C is a real constant.

Secondly, in practice it seems uneasy to compute $|\Delta^\tau(x_1^N)|$. Instead of a uniform distribution on all the global possible splittings, we would like to take the distribution given by the product of the uniform distributions on all local possible splits at each node. To state it in a more formal (and probably clearer) way, let us denote by σ a given partitioning of the sample over the tree τ and by I_s^σ the number of examples reaching an internal node s using the partitioning σ. If we choose uniformly at each internal node between the $I_s^\sigma + 1$ different possible splits, then the new distribution ω on the possible partitionings is given by

$$\omega(\sigma) = \prod_{s \in \mathrm{int(T)}} \frac{1}{I_s^\sigma + 1}.$$

The advantage of this distribution is that it can be computed easily in a recursive way for every σ. Furthermore, the bound of theorem 3 still holds true since

$$-\log \omega(\sigma) = \sum_{s \in \mathrm{int}(T)} \log(I_s^\sigma + 1) \leq |\partial m| \log(N+1).$$

Finally, notice that the bound in theorem 3 is not as good as in theorem 1, namely, there is a term of order $\log(N)/L$ in the new bound whereas the penalty term was only of order $1/L$ in theorem 1. If we take $L = kN$, $0 < k < 1$, this means that if the real conditional distribution $P_N(Y_1^N | X_1^N)$ is in the model (that is, is of the form $M_{m,\theta^m}^{\otimes N}$), the modified estimator converges towards it at a rate of order $\log N/N$, to be compared with $1/N$ in theorem 1. This is because we dropped the hypothesis of a countable family of models. However, in the particular case considered in this section, namely, conditional multivariate distributions M_{m,θ^m} depending on the tree model m and the set of parameters θ^m, the following theorem states that this rate of convergence is actually the minimax rate within the set of models, and therefore cannot be significantly improved.

Theorem 4. *Assume X is a real random variable drawn according to the uniform distribution U in $[0,1]$, then with the set of tree models $\mathcal{M}$ defined in this section, there exists a real constant C such that for any estimator $Q(y|x, z_1^N)$ depending of a sample of size N, the following lower bound holds true for sufficiently big N:*

$$\sup_{m,\theta^m} E\, H(M_{m,\theta^m}(Y|X), Q(Y|X, Z_1^N)) \geq C\frac{\log N}{N},$$

where the expectation on Z_1^N is taken with respect to the product distribution $(U(dX).M_{m,\theta^m}(dY|X))^{\otimes N}$ (in other words, when Z_1^N is drawn i.i.d. according to $U(dX).M_{m,\theta^m}(dY|X)$).

To prove this theorem we will make use of the following version of Fano's lemma (see e.g. [9], corollary 2.9):

Lemma 1. *Let $\mathcal{P}$ be a finite set of probability distributions on a measurable space $\mathcal{X}$ such that $|\mathcal{P}| = J \geq 4$, and $\psi : \mathcal{P} \to F$ a function taking its values in some metric space (F, d). Assume there exist positive constants K, γ, t such that*

$$\sup_{P,Q\in\mathcal{P}} H(P,Q) \leq K, \tag{i}$$

$$\inf_{\substack{P,Q\in\mathcal{P}\\ P\neq Q}} d(\psi(P), \psi(Q)) > t > 0, \tag{ii}$$

$$K + \log 2 \leq (1-\gamma)\log(J-1), \tag{iii}$$

then for any function $\Phi : \mathcal{X} \to F$ and any $p > 0$ the following lower bound holds

$$\sup_{P\in\mathcal{P}} E_P\left[d^p(\Phi(X), \psi(P))\right] \geq \gamma\left(\frac{t}{2}\right)^p.$$

Proof of theorem 4. We can restrict ourselves to the case $c = 1$ since the minimax rate of convergence can only increase with c. Thus the distributions we are dealing with are conditional Bernoulli, which we will denote by $B_{m,\theta^m}(dY|X)$.

We will apply Fano's lemma in the following framework: as the metric space F we will take the set of all distributions on $[0,1]\times\{0,1\}$ and for d the Hellinger distance on E. Below we will construct the set $\mathcal{P}$ as a finite subset of the the set of product probability distributions $\mathcal{D} = \left\{(U(dX).B_{m,\theta^m}(dY|X))^{\otimes N}\right\}$ on $([0,1]\times\{0,1\})^N$. A natural function $\psi : \mathcal{D} \to F$ is $\psi : P^{\otimes N} \mapsto P$ and any estimator depending on a sample of size N is indeed a function $\Phi : ([0,1]\times\{0,1\})^N \to F$.

Let us choose $\alpha, \beta \in [0,1]$ which will remain fixed for the rest of the proof. Let us then consider the conditional probability of Y given by a Bernoulli distribution of parameter β if X belongs to some subinterval $I \subset [0,1]$ of length η, and of parameter α if X does not belong to I. This distribution clearly belongs to our model since it can be represented as a tree with two questions corresponding to the endpoints of I. If $P_1^{\otimes N}, P_2^{\otimes N}$ are N-th direct products of two such distributions on (X,Y) (drawing X uniformly on $[0,1]$), obtained with two disjoint intervals I_1 and I_2 of the same length η, then we get easily

$$\begin{aligned} H(P_1^{\otimes N}, P_2^{\otimes N}) &\leq N\eta(H(\beta,\alpha) + H(\alpha,\beta)), \\ d^2(P_1, P_2) &= \eta(\sqrt{\alpha} - \sqrt{\beta})^2. \end{aligned}$$

Now we can find as many as $J = \left\lfloor \frac{1}{\eta} \right\rfloor$ such disjoint intervals in $[0,1]$ and take for $\mathcal{P}$ the set of distributions thus obtained. Taking

$$\eta = \frac{1}{H(\alpha,\beta) + H(\beta,\alpha)} \frac{\log N}{2N},$$

the hypotheses of Fano's lemma are now satisfied for $K = \frac{1}{2}\log N$, $\gamma = 1/4$, $t = C_1\sqrt{\log N/N}$ for a constant $C_1 = |\sqrt{\alpha} - \sqrt{\beta}|/\sqrt{(2H(\alpha,\beta) + H(\beta,\alpha))}$, and N big enough so that (iii) is satisfied.

This yields the desired result if we choose $p = 2$ in the inequality obtained, and finally apply the well-known inequality $H(P,Q) \geq 2d^2(P,Q)$ as a final step. □

Remark: Note that the only probability distributions for Y used in this proof are Bernoulli with parameter α or β which can be arbitrarily fixed. Therefore the extra $\log N$ factor in the rate of convergence does not come from the "pathologic" behavior of the Kullback-Leibler distance for parameter values near 0 and 1, as one might have thought at first glance.

2.6 Approximate computation using a Monte-Carlo method

Let us have a new look on equation (2.1) and focus our attention on one particular term $Q^E_{\text{mix.}}(dY_1|x_1, z_2^M)$ in the Cesàro sum defining Q^N_π. Since we will still be working in this section in the framework of classification trees, the space

$\mathcal{Y}$ is discrete and we can identify probability distributions with their densities. Let us write

$$\begin{aligned} Q^E_{\text{mix.}}(y_1|x_1, z_2^M) &= \frac{\displaystyle\sum_{m\in\mathcal{M}} \pi(m) Q^E_m\left(y_1^M|x_1^M\right)}{\displaystyle\sum_{m'\in\mathcal{M}} \pi(m') Q^E_{m'}\left(y_2^M|x_2^M\right)} \\ &= \sum_{m\in\mathcal{M}} Q^E_m\left(y_1|x_1\right) \frac{\pi(m) Q^E_m\left(y_2^M|x_2^M\right)}{\displaystyle\sum_{m'\in\mathcal{M}} \pi(m') Q^E_{m'}\left(y_2^M|x_2^M\right)} \\ &= \sum_{m\in\mathcal{M}} Q^E_m\left(y_1|x_1\right) \omega^E(m|z_2^M), \end{aligned}$$

where ω^E is, from a Bayesian point of view, the *a posteriori* distribution of the models given the data subset z_2^M of the test set, when the estimation set is fixed. We have shown in section 4 that in the case of a fixed-design tree, and when the prior π is the distribution of the genealogy tree of a branching process with constant parameter, this sum can be recursively computed in an efficient way.

In the general case however, the tree design is not fixed because we have to choose between several questions at each node, and we did not find any similar factorization for the computations. We therefore propose to simulate the *a posteriori* distribution of the models using a Monte-Carlo algorithm.

In order to do this we will need to define a reversible transition kernel $\Gamma(m, m')$ on the set of models $\mathcal{M}$, such that the associated stationary distribution is precisely ω^E. Γ should therefore be irreducible and acyclic and satisfy for any $m, m' \in \mathcal{M}$:

$$\Gamma(m, m') = \Gamma(m', m) = 0$$

or

$$\frac{\Gamma(m', m)}{\Gamma(m, m')} = \frac{\omega^E(m|z_2^M)}{\omega^E(m'|z_2^M)} = \frac{\pi(m)}{\pi(m')} \cdot \frac{Q^E_m(y_2^M|x_2^M)}{Q^E_{m'}(y_2^M|x_2^M)}. \tag{2.12}$$

Our goal is that at each transition of the MCMC chain there should be as few computations as possible to perform. For that purpose we will allow $\Gamma(m, m') \neq 0$ only if m' is a “neighbor” of m. We will call $m = (T, F_T)$ a neighbor of $m' = (T', F'_{T'})$ if the trees T and T' only differ by two added or removed terminal nodes (leafs), i.e. if T' is obtained from T by turning one of its leafs into an internal node and adjoining to it two leafs, or vice-versa, and if $F_{T\cap T'} = F'_{T\cap T'}$, or in other words, if the questions attached to the common internal nodes of T and T' are the same.

As in the previous section we will construct our prior π in the form

$$\pi(m) = \pi(T, F_T) = \pi(T)\pi(F_T|T).$$

Once again we will choose for the first marginal $\pi(T)$ the probability distribution of the genealogy tree of a branching process with a constant parameter $\rho \leq \frac{1}{2}$, restricted (and normalized) to the set of trees of depth less than d (this a is slightly different prior than the one considered in section 2.4, which was the distribution of a *stopped* branching process, though the difference is quite negligible in practice). The advantage of this choice is that an MCMC chain simulating this distribution can easily be built using the following algorithm for the transitions:

- Start at the root node, then follow the branches choosing for each internal node you reach one of its sons with a probability of $\frac{1}{2}$, until you reach a terminal node f;
- If f's brother is also a leaf, then destroy them both with probability $\frac{1}{2}$, or give two sons to f with probability $\rho(1-\rho)/2$ (unless you are at the maximum depth d, in which case you do nothing), or do nothing with the remaining probability.
- If f's brother is an internal node, then give two sons to f with probability $\rho(1-\rho)/2$, or do nothing with the remaining probability.

It can readily be seen that the transition kernel thus defined on the trees is irreducible and aperiodic and that it is reversible w.r.t. its stationary distribution, which is the desired one. Again, like in section 2.4, this can be extended easily to a more general prior where the parameter ρ is not a constant but depends on the node s considered (for example one can actually simulate the exact same distribution as in section 2.4 if wanted).

The next step is to see how to define $\pi(F_T|T)$. Recall that we have in mind to construct a Markov Chain Monte Carlo, and therefore we are interested at the first place in transition probabilities from one tree to one of his neighbors. We thus want to compute in a simple way the ratio

$$\frac{\pi(F'_{T'}|T')}{\pi(F_T|T)} \tag{2.13}$$

when (T, F_T) and $(T', F'_{T'})$ are neighbor labeled trees. (To be sure that this ratio is well-defined we will assume here and for the rest of this section that all trees have strictly positive prior probability.)

To make things clearer, let us assume that we want to perform a transition from T to T', where T is a subtree of T' and that the leaf s of T has been turned into an internal node in T', so a new question must be attached to s in T'. We would like to define a *local rule* $R(f_s|F_T)$ giving us the probability distribution of this new question given the rest of the tree (which we will use as transition probability for the MCMC chain). Unfortunately, the ratio (2.13) is *not* necessarily a probability distribution on the new question, because of the different dependencies on T' and T respectively for the numerator and the denominator.

This motivates the introduction of the following definitions and hypotheses: let us write $F_T \prec F_{T'}$ if $T \subset T'$ and if $F_{T'}$ and F_T coincide on the internal nodes of T. Let us then define the marginal

$$\pi(F_{T'}|T) = \sum_{F_T : F_{T'} \prec F_T} \pi(F_T|T).$$

In order the ratio (2.13) to be a probability distribution, we will make the assumption that

$$\forall T' \subset T \qquad \pi(F_{T'}|T) = \pi(F_{T'}|T'). \tag{2.14}$$

As a natural consequence, if we define $\widetilde{\pi}$ as a distribution on the questions attached to the maximal complete tree T^* of depth d by $\widetilde{\pi}(F_{T^*}) = \pi(F_{T^*}|T^*)$ and $\widetilde{\pi}$'s marginal

$$\widetilde{\pi}(F_T) \stackrel{\Delta}{=} \sum_{F_{T^*} : F_T \prec F_{T^*}} \pi(F_{T^*}|T^*) = \pi(F_T|T^*),$$

then by assumption (2.14) we have

$$\widetilde{\pi}(F_T) = \pi(F_T|T).$$

Therefore any non-vanishing distribution on the questions attached to the maximal tree T^* gives rise to an associated local rule defined by

$$R(f_s|F_T) = \frac{\widetilde{\pi}(F'_{T'})}{\widetilde{\pi}(F_T)} \tag{2.15}$$

(we recall that s is the leaf of T which becomes an internal node of T', and f_s is the question attached to node s.). R is now a true conditional distribution on the question f_s.

The natural reciprocal problem to be asked now is, what are the minimal conditions to be satisfied by a set of local rules R (that is to say, a set of conditional distributions on the question f_s to be asked at node s given the questions asked on the internal nodes of a tree T such that s is a leaf of T), so that there exists a distribution $\widetilde{\pi}(F_T)$ such that (2.15) is satisfied? Obviously, a "local coherence" condition must be satisfied, namely, take a tree T, let s_1 and s_2 be two different leafs of T; T_1, T_2, T_{12} be the trees obtained by adjoining two sons to s_1, s_2, and both s_1 and s_2, respectively; $F_T, F^1_{T_1}, F^2_{T_2}$ and $F^{12}_{T_{12}}$ be sets of questions attached to these trees such that they match on the common internal nodes of two of them, then R must satisfy

$$\frac{\widetilde{\pi}(F_{T_{12}})}{\widetilde{\pi}(F_T)} = R(f_{s_2}|F^1_{T_1}).R(f_{s_1}|F_T) = R(f_{s_1}|F^2_{T_2}).R(f_{s_2}|F_T) \qquad (*)$$

(this means that the probability distribution of the two new questions given the initial tree T must not depend on the order they were constructed).

The following theorem states that this necessary condition is also sufficient:

Theorem 5. *Let a set local rules R be given such such that condition $(*)$ is satisfied for any choice of (T, s_1, s_2), then there exists a distribution $\widetilde{\pi}(F_T)$ on the set of questions attached to the maximal tree T^* satisfying (2.15).*

Proof. This is actually quite straightforward. Let (T, F_T) be a model tree. Choose some way of building T node by node, namely a sequence $\{\emptyset\} = T_0 \subsetneq T_1 \subsetneq \ldots \subsetneq T_k = T$ and a sequence $\emptyset = F^0_{T_0}, F^1_{T_1}, \ldots, F^k_{T_k} = F_T$ such that for every $i < k$, $(T_i, F^i_{T_i})$ and $(T_{i+1}, F^{i+1}_{T_{i+1}})$ are neighbors (thus $F^i_{T_i}$ is necessarily the set of questions F_T restricted to the internal nodes of T_i). We will call such a sequence an *construction path* for T. Necessarily $\widetilde{\pi}(F_T)$ should be defined the following way:

$$\widetilde{\pi}(F_T) = \prod_{i=1}^{k} R(f_{s_i}|F^{i-1}_{T_{i-1}}),$$

where s_i is of course the leaf of T_{i-1} turned into an internal node of T_i, and f_{s_i} the question attached to s_i.

It is clear now that we should only make sure that this definition does not depend on the path we chose. Obviously the construction paths are in natural bijection with the set of permutations $\mathfrak{S} = (s_0 = \emptyset, s_1, \dots, s_n)$ of all the internal nodes of T, satisfying the property that for any $i < j$, s_j is not an ancestor of s_i (i.e. $\mathfrak{S}$ is compatible with the order “be ancestor of”). Such a permutation will be called *admissible* and gives us the order of construction of T node by node. Now relation $(*)$ tells us that if s_i is not the father of s_{i+1}, swapping them in the permutation does not change the value of $\widetilde{\pi}(F_T)$ above defined (and the new permutation is still admissible). We will say that the new permutation is equivalent to the first one.

Now it is easy to see that actually all admissible permutations are equivalent using a simple recursion on the number of internal nodes of T. Namely, let $\mathfrak{S}_1$ and $\mathfrak{S}_2$ be two admissible permutations. Let s^* be an internal node of T such that his two sons are both leafs. Thus no internal node is a descendant of s^* and therefore we can swap s^* in $\mathfrak{S}_1$ and $\mathfrak{S}_2$ successively with all the nodes built after him. We thus obtain two admissible permutations $\mathfrak{S}_1'$ and $\mathfrak{S}_2'$ respectively equivalent to $\mathfrak{S}_1$ and $\mathfrak{S}_2$ and such that s^* is the last node to be built in both cases. We can thus proceed recursively, since the case of a single internal node is obvious. $\square$

In particular, it is clear that we can choose any local rule R such that the probability distribution of the question at a given node only depends on his ancestors. In this case $(*)$ is obviously satisfied.

We now study the last factor in the right side of (2.12). Let $m_1 = (T_1, F^1_{T_1})$ and $m_2 = (T_2, F^2_{T_2})$ be two neighbor models such that $T_1 \subsetneq T_2$, u is the leaf of T_1 being turned into an internal node of T_2, and f_u is the new question attached to u. As in section 4 we define for any internal or terminal node s the counters $b^i_s, i = 0, \dots, c$ as the number of examples of the subsample z_2^M that cross or reach node s and are of class i. Since the estimators $Q^E_m(\,\cdot \mid \cdot\,)$ are conditional multivariate distributions whose vector parameters θ_s at each leaf s are estimated using the estimation set z^N_{L+1} (here we don't make any hypothesis on the way these parameters are estimated, it can be a Laplace estimator or any other estimator), we get

$$
\begin{aligned}
K(m_1, u, f_u) \triangleq \frac{Q^E_{m_1}(y_2^M | x_2^M)}{Q^E_{m_2}(y_2^M | x_2^M)} &= \frac{\displaystyle\prod_{s \in \partial T_1} \prod_{i=0}^{c} (\theta_{s,i})^{b_s^i}}{\displaystyle\prod_{s \in \partial T_2} \prod_{i=0}^{c} (\theta_{s,i})^{b_s^i}} \\
&= \prod_{i=0}^{c} \frac{(\theta_{u,i})^{b_u^i}}{(\theta_{u0,i})^{b_{u0}^i} (\theta_{u1,i})^{b_{u1}^i}}.
\end{aligned}
$$

Here $u0$ and $u1$ denote of course the two sons of u, and $\theta_{s,i}$ is the i-th coordinate of θ_s. Thus this factor only depends on the counters at nodes $u, u0, u1$ and is therefore easy to compute.

To put it together, here is an algorithmic description of an iteration of our MCMC algorithm for the step concerning the transition towards a new model:

1. Choose a leaf of the current model m and choose whether you adjoin to it two new sons, or destroy it and his brother, or do nothing, according to the algorithm already described sooner.

2. If you chose to delete two leafs, denote by u their father, then accept the new truncated model m' with probability $K(m', u, f_u) \wedge 1$;

3. If you chose to add two leafs, denote by u the happy father of the twins, choose a question f_u to be attached to it according to the local distribution R, then accept the new model m' with probability $1/K(m, u, f_u) \wedge 1$.

It is now clear that the transition kernel $\Gamma(m, m')$ thus defined satisfies (2.12). Furthermore, the number of counters to be updated at each step is of most $c + 1$ (in the case you create two new leafs).

What is left is now to perform the Cesàro sum of the Bayesian estimators while the test set is growing. To do this, we suggest that the examples could actually be introduced progressively while the Monte-Carlo chain is running, so that there is the same number of steps performed in the chain for each new example introduced. However, one could think of other possible methods to achieve a tradeoff between accuracy and computation complexity ; we are planning to perform several tests using various methods in the next future.

2.7 Discussion

The point of view we adopted in this paper is to give some basic theoretical results about the progressive mixture estimator used in regression estimation. Actually this estimator is a modification of the classical Bayesian analysis, for which we divide the sample into two subsets and perform a Cesàro sum over Bayesian mixtures of available estimators as we let the size of the "test set" grow. Restricting our attention on the case of regression trees, we then proposed that the computation of the mixture could be approximated by a MCMC chain, which is quite standard in Bayesian analysis.

We want to discuss here the relation of this work with that of Geman and Amit [2, 4] who put forward, in a written character recognition problem, the construction of a classification tree using a local entropy minimization rule. More precisely, the basic algorithm they propose is to grow a tree by selecting at each node the question which reduces the most the empirical entropy at this node (i.e. a local maximum likelihood selection), stopping the construction when reaching a fixed maximum depth or a minimum size of the sample at a given node. (In practice, the optimization is in fact performed on a randomly selected small subset of the available set of questions at each node). We want to point out the similarity of this procedure with ours: in an informal sense the local entropy reduction procedure can be compared with a MCMC chain (or, to fit the comparison better, a Gibbs sampler) at "zero temperature". In our algorithm the number of examples in the test set plays the role of inverse temperature (as the bigger the test set is, the less likely we are to select questions which are locally "irrelevant"). In this regard, introducing the test examples progressively can be seen as a kind of simulated annealing algorithm.

We hope that allowing more liberty in the search in the tree space can lead to visit trees achieving a better performance. Moreover, the *a priori* distribution acts here as a penalty over the complexity of the models, and thus as a "natural" stopping rule included in the algorithm. However as what we want here is to compute a sum over the models by simulating their *a posteriori* distribution, and not merely to select one of them, the complexity of the computation will of course be much higher than for the local optimization algorithm. We are currently trying to find some answers to this problem; one could think for example of performing a MCMC trajectory once and for all, and then just re-use it to estimate every new example.

We would also like to make some reference to Chipman et al.'s work [25], of which we became aware only recently, which also extensively presents the use of a MCMC algorithm to explore the space of tree models with a Bayesian prior. There is a strong similarity between their procedure and ours; we therefore want to highlight a few points on which we hope our point of view might be relevant. First we started from a theoretical point of view, deriving strong upper bounds for the risk of the progressive mixture estimator in a general framework and for the particular case of regression trees. This has two main consequences on the design of our algorithm: first, our goal is really to compute a sum, and not to select a single tree as that was the case in [25]. In this regard we follow the Bayesian principle more strictly. It is relevant to note that Catoni [24] proved on a simple example that model selection may behave worse than model mixture. Besides, Geman and Amit also show that performance turns out to be much better when they perform a weighted sum over several different trees obtained with the "randomized" version of the local entropy reduction algorithm.

Secondly, because of our construction we divide our sample into two sets and suggest to introduce the test examples progressively. We believe that this could give more liberty in the movements of the MCMC chain at the beginning like in a simulated annealing algorithm (see above). On the other hand, the MCMC implementation we exposed here is still rough and quite sketchy; practical evidence in [25] strongly suggests that allowing more liberty in the chain transitions (allowing to change the question asked at an internal node; swapping the questions of two neighbor nodes) significantly improves the performance of the algorithm. We plan to make more practical experiments in this direction in the next future.

Chapitre 3

A new algorithm for Bayesian MCMC CART sampling

Abstract.

Recent work ([25], [31]) has shown the interest of using Metropolis procedures in a Bayesian framework to search for good classification trees. For a particular class of prior distributions on the trees, we introduce a new algorithm for MCMC sampling of trees, similar to a Gibbs sampler, based on Willems et al. "tree weighting" algorithm [59], which results in a dramatic increase of the number of models actually taken into account. The sampled tree models are then averaged to produce an aggregate estimator or classifier. We present results of simulations on three benchmark datasets, that show the practical interest of this procedure.

Index terms: CART model, multiple classifiers, Bayesian MCMC sampling, tree weighting procedure, mixture models.

3.1 Introduction

The use of classification and regression tree models (CART, [21]) enjoys a wide popularity for a broad range of classification problems involving a complex structure of features. Recently a lot of attention has been focused on using multiple tree classifiers: a "forest" of classification trees is built using some protocol, and the different outputs of the trees are usually just averaged (possibly using some weighted averaging). Use of such "aggregated classifiers" has been proven to be largely more efficient than a single tree, especially in the context of classification.

Various protocols to build multiple trees include randomization procedures when building a tree, such as proposed by Amit and Geman in the context of handwritten digit recognition [2, 4], bootstrap resampling of the training set (Breiman's "bagging", [17]), and "boosting" (Freund and Shapire, [36]). Another approach is to consider tree sampling by the means of a Markov Chain Monte Carlo procedure, in a Bayesian context. More precisely, one can define a prior probability on tree structures and use a MCMC procedure to simulate the posterior distribution on trees. This idea has already been proposed and tested by several authors, see Chipman et al. [25], Denison et al. [31], Blanchard [14].

The two main points of this article are the following: first, we introduce a new algorithm for MCMC sampling of trees, similar to a Gibbs sampler, that allows a more efficient simulation of the posterior over the space of trees. This algorithm is valid for a particular, yet quite broad, class of prior distributions, and makes use of the "tree weighting algorithm" introduced by Willems et al. [59].

As a second point, our goal is to truly average the tree outputs over the posterior. Most authors using MCMC chains of trees so far were mainly aimed at finding one single "good" tree by maximizing the posterior. Two distinct reasons led us to be eager to perform an average over the sampled trees: on the one hand, theoretical results on the performance of the related, Bayesian-flavored "progressive mixture" estimator [14]; and on the other, to be able to present this algorithm as a legitimate of its own procedure for building "forests" of CARTs, as presented above. Our experiments show that actually keeping only a limited number of trees (of the order of 30 in the examples studied) among a long run of the MCMC is sufficient to achieve good classification rates.

We further introduced in our experiments an "inverse temperature" parameter that results in a modified posterior. This parameter, which we show has a dramatic influence on the results of the aggregated classifier, only makes sense in the context of averaging several trees. In particular, we show that it is closely related to the tradeoff between the average individual performance of one tree and the average conditional dependence of pairs of trees, which is put forward as a key quantity by Amit et al. [1] in the context of multiple randomized classifiers.

3.2 The Bayesian CART estimator

3.2.1 General framework

We will be focusing throughout this paper on a general classification problem. We consider a couple of random variables (X, Y) where X is the "observation" taking his values in some measurable space $\mathcal{X}$ and Y the "class" of X taking its values in a finite set of classes $\mathcal{Y} = \{1, \ldots, c\}$. The general method for classification will be to compute an estimator for the conditional probability $P(Y|X)$: the associated classifier is then $C(X) = \mathrm{Argmax_Y P(Y|X)}$. This means our results can easily be extended to a more general regression framework as well. Also, we can design other rules for inferring the classifier from the estimator, for instance if the misclassification loss function actually depends on the class.

The abstract framework and notations will be the following. We assume that we have a set of training examples $Z_1^N = (X_1, Y_1), \ldots, (X_N, Y_N)$ which we divide into two sets, the "learning set" $\Lambda = Z_1^L$ and the "estimation set" $E = Z_{L+1}^N$. The choice of the parameter L will be studied later. To perform the estimation, suppose that we first have to choose between a countable family of available models $\mathcal{M}$. With every choice of a model $m \in \mathcal{M}$, comes an associated estimator $Q_m^E(Y|X)$, that is, a conditional probability of Y given X, that depends on the estimation set E. We will detail the construction of the models and estimators in the next section.

The main problem regards of course how we should choose a "good" model m, that is, whose risk, measured through a certain loss function, is as low as possible. Instead of actually picking one model in $\mathcal{M}$, we work in the following Bayesian setting: let π be a prior on the set $\mathcal{M}$. We define the *mixture* (conditional) probability on the set $\mathcal{M} \times \mathcal{Y}^L$ by

$$Q_{\text{mix.}}^E(m, Y_1^L|X_1^L) \triangleq \pi(m) \prod_{i=1}^{L} Q_m^E(Y_i|X_i). \tag{3.1}$$

The term "mixture" refers to the following marginal distribution

$$Q_{\text{mix.}}^E(Y_1^L|X_1^L) = \sum_{m \in \mathcal{M}} \pi(m) \prod_{i=1}^{L} Q_m^E(Y_i|X_i).$$

When the estimation set E is kept fixed, each model m defines a (conditional) product probability $\prod_{i=1}^{L} Q_m^E(Y_i|X_i)$ on Λ, so that the marginal distribution defined in the above equation appears as a weighted mixture of product probabilities on the learning set.

We finally define the (predictive) Bayesian estimator Q_π

$$\begin{aligned} Q_\pi(Y_1|X_1, Z_2^N) &\triangleq Q^E_{\text{mix.}}(Y_1|X_1, Z_2^L) \\ &= \sum_{m\in\mathcal{M}} \pi(m) \frac{\prod_{i=1}^L Q^E_m(Y_i|X_i)}{\sum_{m'\in\mathcal{M}} \pi(m') \prod_{j=2}^L Q_{m'}(Y_j|X_j)}. \end{aligned} \tag{3.2}$$

Please note that defined this way, Q_π is an estimator of the conditional probability of the class Y_1 given observation X_1, using the knowledge of the rest of the sample $(X_2, Y_2), \ldots, (X_N, Y_N)$. The fact that we perform the estimation on the observation having index 1 is purely a notational convenience and should not be misleading.

3.2.2 Relation to previous work

In [14], following ideas first introduced by Catoni [24] we have presented theoretical bounds for the Kullback-Leibler loss of the "progressive mixture" estimator defined by the following Cesàro mean of Bayesian estimators

$$\begin{aligned} Q_{\text{prog.}}(Y_1|X_1, Z_2^N) &\triangleq \frac{1}{L} \sum_{K=1}^L \sum_{m\in\mathcal{M}} \pi(m) \frac{\prod_{i=1}^K Q^E_m(Y_i|X_i)}{\sum_{m'\in\mathcal{M}} \pi(m') \prod_{j=2}^K Q^E_{m'}(Y_j|X_j)} \\ &= \frac{1}{L} \sum_{K=1}^L Q^E_{\text{mix.}}(Y_1|X_1, Z_2^K). \end{aligned} \tag{3.3}$$

Let P_N be the joint probability of the variables $(Z_i)_{1\le i\le N}$, and let us denote by $R(Q)$ the Kullback-Leibler risk of an estimator $Q(Y_1|X_1, Z_2^N)$ with respect to P_N:

$$R(Q) = E_{P_N} D(P_N(Y_1|X_1, Z_2^N), Q(Y_1|X_1, Z_2^N)),$$

where D denotes the Kullback-Leibler divergence, i.e.

$$D(P, Q) = \begin{cases} E_P \log dP/dQ & \text{if } P \ll Q \\ +\infty & \text{otherwise.} \end{cases}$$

Then the following bound holds:

Theorem 6. *[14] If the joint probability P_N of the variables $(Z_i)_{1\le i\le N}$ is exchangeable,*

$$R(Q_{\text{prog.}}) \le \inf_{m\in\mathcal{M}} \left\{ R(Q_m) + \frac{1}{L} \log \frac{1}{\pi(m)} \right\} \tag{3.4}$$

This bound tells us that the K-L risk of the progressive mixture estimator $Q_{\text{prog.}}$ is not bigger than the risk of any model m plus a penalty term of order $\mathcal{O}(1/N)$, depending on m (when L is chosen to be of the form $L = \lfloor kN \rfloor$). We refer the reader to [14] for more detail.

For the sake of simplicity in issues concerning practical implementation, we have decided to restrict our attention throughout this paper to the experimental study of the Bayesian estimator Q_π (which is also the last term of the above Cesàro sum). Although bound (3.4) does not apply directly to this estimator, we hope that their behavior are close enough in practice.

3.2.3 The CART model

We now define the model space $\mathcal{M}$ and estimators $Q_m^E(Y|X)$ as CART estimators [21]. We assume that we have a certain "library" $\mathcal{Q}$ of available questions on the observation X. A question is a measurable function $f : \mathcal{X} \to \{0,1\}$. For instance, a very typical setup is when $\mathcal{X} = \mathbb{R}^n$ and the questions are a set of separating hyperplanes $f_{v,b} : x \mapsto \mathbf{1}\{(x|v) \geq b\}$, where $\mathbf{1}$ denotes the indicator function of a set.

A classification tree is a tree whose nodes are labeled with questions $f \in \mathcal{Q}$. Given some observation X, we start from the root and choose at each node the left or right branch depending on the answer given to the question attached to the node. We continue until we reach a leaf; each leaf contains a certain distribution on Y that will be our estimator $Q_m^E(Y|X)$. The distributions contained in the leaves are estimated using the estimation set E.

Formally, a CART model m is defined by a complete binary tree whose internal nodes are labeled by elements of $\mathcal{Q}$ (by *complete* tree we mean that each node either is a leaf or has two descendants, there are no nodes having a single descendant). So $m = (T, K)$, where T is a binary tree structure, and K is the set of questions assigned to the internal nodes of T.

We denote by $m(X)$ the terminal node of T reached by letting the observation X follow the branches of the tree, as described earlier. Putting, for s a terminal node of T,

$$n_s(i) = \sum_{(X,Y)\in E} \mathbf{1}\{m(X) = s, Y = i\}\ ,\ i = 1, \ldots, c\ ,$$

we define the estimator Q_m through the distributions attached to the leaves, namely

$$Q_m^E(y|m(x) = s) = \frac{n_s(y) + 1}{\sum_i n_s(i) + c}.$$

The main reason for which we chose a Laplace estimator at each terminal node, and not the simpler frequentist estimator, is that in the final Bayesian estimator (3.2), each model is weighted according to its posterior distribution on the learning set Λ and we don't want to allow this posterior to vanish. Also, simple bounds can be derived for the Kullback-Leibler risk of the Laplace estimator that do not hold for the frequentist estimator, and make sense in the setting recalled in section 3.2.2.

Of course, in order for the set of models $\mathcal{M}$ to be countable, the set of questions $\mathcal{Q}$ has to be countable in the first place. However, it can be shown that a bound similar to (3.4) holds in a more general setting when $\mathcal{Q}$ is of finite Vapnik-Cervonenkis dimension. Without getting into detail, the general principle is that we can regroup questions that divide the sample Z_1^N in the same way. For fixed N, we thus get a finite number of sets of regrouped questions and therefore a countable set of equivalent models on which we can define a prior in a suitable way [14].

3.3 Approximate computation through a MCMC procedure

3.3.1 Notations and hypotheses

The exact computation of the estimator (3.2) appears untractable because of the high number of models m involved. As an alternative method, common in Bayesian analysis, we will make use of a MCMC procedure, simulating the posterior on the set $\mathcal{M}$, to provide an approximate computation for the estimator. The idea of a MCMC simulation on classification trees in a Bayesian framework has already be suggested and studied by several authors [25], [31], and in the previous paper [14].

With the definition of a model m as a couple (T, K), an intuitively natural way to define a MCMC setup is to allow a move from m to m' by modifying either T or K. As a matter of fact, MCMC procedures proposed so far in the literature for this particular setting seem to always involve stochastic movement on the tree structure itself: at each step of the MCMC algorithm, a pair of terminal nodes can be added or removed from the current tree. Our contribution to this point is to suggest a different algorithm for which the structure of the tree is not altered, only the questions at the interior nodes of the complete tree of depth d are changed at each step. The partial sum over all subtrees of this complete tree can then be computed in an exact and computationally efficient way at each step of the Monte-Carlo chain. We hope this algorithm to be more efficient and quick in practice than the previous ones.

We will first need to make a slight change in notation. Let us first choose a maximal depth d for all the trees. From now on, when writing $m = (T, K)$ we will assume that K actually stands for a set of questions attached to the internal nodes of the *maximal* complete binary tree of depth d, even though some of them are not used in the actual tree structure T. By maximal tree we mean the tree whose leaves are all at depth d. This just amounts to splitting the old original models into several equivalent models (namely, all models that have the same questions as the original one on the actual internal nodes of the structure T). The reason for this will become clearer quite soon.

The following set of assumptions, denoted (H), will be necessary and assumed to hold in the sequel:

- The trees have a maximum depth d; in other words, the trees deeper than d have a prior probability zero.

- We assume that the prior π can be written as a product

$$\pi(m) = \pi(T, K) = \pi_1(T)\pi_2(K). \tag{3.5}$$

  This means that the questions are independent of the structure for the prior distribution.

- The prior π_1 on the tree structure T is exactly the distribution of the genealogy tree of a binary branching process of given parameter ρ, stopped at depth d (each individual gives birth to two descendants with probability ρ, or dies without children with probability $1 - \rho$, except for individuals of the dth generation who never have children). We will show later that this hypothesis can be extended to a more general prior, namely a branching process where the parameter ρ can actually depend on the location of the node in the tree.

3.3.2 A Gibbs sampler for trees

The key point of the algorithm is that, under (H), when the set of questions K is fixed, a sum of the form

$$S(K, Z_1^P) = \sum_{T \in \mathcal{T}_d} \pi_1(T) \prod_{i=1}^{P} Q^E_{(T,K)}(Y_i|X_i), \tag{3.6}$$

(where $\mathcal{T}_d$ denotes the set of all complete binary tree structures of depth less than d), can be computed analytically and in a "fast" way, using the recursive "tree-weighting" algorithm designed by Willems et al. [59] in a data compression framework. We describe the algorithm in Appendix 3.6.1.

To take advantage of this algorithm, we develop the sum (3.2) defining Q_π into two stages, namely a sum over all the possible values of the set of questions K, and a sum over the tree structure $T \in \mathcal{T}_d$ when K is fixed. The principle is that the tree weighting algorithm allows us to compute efficiently this second stage, while a Gibbs sampler algorithm only acting on the values of K is used to perform the first stage.

More formally, let $\mathcal{Q}_d$ denote the set of all possible values of K, that is, all possible assignments of questions to the internal nodes of the maximal complete binary tree of depth d.

First we note that from the definition of $Q_{\text{mix.}}$ (3.1) we have

$$Q^E_{\text{mix.}}(K, Y_1^P|X_1^P) = \sum_T Q^E_{\text{mix.}}(\,(T,K), Y_1^P|X_1^P) = \pi_2(K)S(K, Z_1^P) \tag{3.7}$$

Now, we develop the definition (3.2) of Q_π into a sum over K only, getting

$$\begin{aligned} Q_\pi(Y_1|X_1, Z_2^N) &= Q^E_{\text{mix.}}(Y_1|X_1, Z_2^L) \\ &= \sum_{K \in \mathcal{Q}_d} Q^E_{\text{mix.}}(K|Z_2^L) Q^E_{\text{mix.}}(Y_1|X_1, Z_2^L, K) \end{aligned} \tag{3.8}$$

For every K, using equation (3.7) the second factor can be written

$$Q^E_{\text{mix.}}(Y_1|X_1, Z_2^L, K) = \frac{Q^E_{\text{mix.}}(K, Y_1^L|X_1^L)}{Q^E_{\text{mix.}}(K, Y_2^L|X_2^L)} = \frac{S(K, Z_1^L)}{S(K, Z_2^L)}. \tag{3.9}$$

As for the first factor, which is the marginal posterior probability of K, using again relation (3.7), we obtain

$$Q^E_{\text{mix.}}(K|Z_2^L) = \frac{Q^E_{\text{mix.}}(K, Y_2^L|X_2^L)}{\sum_{K'} Q^E_{\text{mix.}}(K', Y_2^L|X_2^L)} = \frac{\pi_2(K)S(K, Z_2^L)}{\sum_{K'} \pi_2(K')S(K', Z_2^L)} \tag{3.10}$$

To sum things up, for every K we are able to compute the second factor of (3.8) through the tree weighting algorithm. Therefore, we only need to simulate the marginal of the posterior of K on $\mathcal{Q}_d$.

The quantities $Q^E_{\text{mix.}}(Y_1|X_1, Z_2^L, K)$ are actually a family of estimators indexed by K; K can therefore be seen as a super-model index on which the MCMC transitions will operate. Once K is fixed, we will call informally $Q^E_{\text{mix.}}(Y_1|X_1, Z_2^L, K)$ an "s-tree" estimator defined by K. An s-tree estimator is obtained by averaging according to the posterior over all the possible subtree structures of the maximal tree of depth d when the questions K are fixed.

The state space of our MCMC algorithm will be the space of s-trees. The posterior distribution over K is given by equation (3.10)

$$Q^E_{\text{mix.}}(K|Z_2^L) \propto \pi_2(K)S(K, Z_2^L). \tag{3.11}$$

This gives rise to the following Gibbs sampler algorithm:

1. Initialize randomly the questions at the internal nodes of the maximal tree of depth d.

2. Choose an internal node s at random, according to some fixed in advance probability distribution, whose support includes all the nodes.

3. Choose a new question f for this node according to the distribution $\pi_2(\cdot|K_s)$ on $\mathcal{Q}$, where K_s is the set of questions attached to the internal nodes of the tree except for node s. Denote by K' the new set of questions (i.e. question attached to node s being replaced by f).

4. Accept the new set of questions K' with probability
$$1 \wedge \frac{S(K', Z_2^L)}{S(K, Z_2^L)}. \tag{3.12}$$

5. Record the new s-tree obtained, and go to step 2.

The MCMC algorithm then consists in averaging the terms given by equation (3.9) over the sample of s-trees obtained by this algorithm.

Note that alternatively, for step 2, one can sweep successively through all the internal nodes according to a fixed deterministic order, as in a classical Gibbs sampler. However, with this method half of the computation time will be used to change questions at the deepest internal nodes, which does not appear as a good strategy since the most relevant nodes are the top ones. For the same reason, a uniform random choice between the internal nodes is probably not optimal. The solution we adopted is to draw a depth p uniformly between 1 and $d-1$, and then to draw a node uniformly among nodes at depth p. This ensures that the questions at the top nodes will have a reasonable variability, which certainly is a crucial point if we are to expect the algorithm to have good mixing properties.

The complexity of the algorithm can be studied as follows: at each step of the algorithm we have to compute the sum $S(K', Z_2^L)$ which is done in less than $\mathcal{O}(2^d)$ operations (more precisely, if we choose to change node s which is at depth h, we have to update the entire subtree rooted at s and then update the nodes on the branch from the root to s, resulting in $\mathcal{O}(2^{d-h}+h)$ operations. This a straightforward consequence of the form of the tree weighting algorithm, see appendix 3.6.1); we then may also possibly have to compute $S(K', Z_1^L)$ in the case the transition is accepted, but this one is done in $\mathcal{O}(d)$ operations since we only have to update the counters along the branch followed by observation X_1.

3.3.3 Variants and extensions

Prior on the tree structure

As announced previously, the prior on the tree structure T can be made more flexible if the following way: the parameter ρ of the Galton-Watson process (i.e. the probability for a node to have two descendants) can be a function depending on the node itself, this appears clearly when looking at the recursive algorithm in Appendix 3.6.1.

This way, we have some control over the prior; for example, we can tune the parameters ρ so that trees of a given size have a higher prior, lower the prior of the shallow trees which we don't expect to be very good by taking ρ close to 1 for the root node and its descendants, etc.

Other MCMC moves

Chipman et al. suggest in [25] that in addition to changing the question at a given node, allowing to swap the questions between one internal node and his parent yields significant improvement in terms of speed of convergence. Of course this move can be added to our version as well—it is not a Gibbs sampler any more in this case though.

Continuous set of questions

The algorithm applies in principle in cases when the set of questions $\mathcal{Q}$ is finite or countable. However, in can be extended in cases where it is a continuous set as studied in [14]: namely we can regroup the questions into sets of statistically equivalent questions with respect to the sample, that is to say, questions that divide the whole sample Z_1^N in the same way. Then we draw uniformly between these sets (which are of course in a finite number) and pick any question of the chosen set.

In the case where the space of features is $\mathbb{R}^n$ and the questions are hyperplanes orthogonal to one of the coordinate axes, we can do the following: choose a coordinate number p at random between 1 and n and then choose uniformly between the possible splits of the sample data along the p-th coordinate.

Same estimation and learning sets

While the separation between an estimation and a learning set allowed us, in [14], to derive bounds on the Kullback-Leibler risk for the "progressive estimator" given by equation (3.3), we are naturally led, in practice, to try to use the same set for estimation and learning. This is, of course, going one more step away from the theory. Our experiments on standard databases show that this, heuristically, usually yields a noticeable improving of results.

"Temperature" parameter

Because we wanted to get more variability in the Monte Carlo chain used to simulate the posterior, we introduced a "inverse temperature" parameter β and replaced the probability of accepting a new question in equation (3.12) by

$$1 \wedge \left(\frac{S(K', Z_2^L)}{S(K, Z_2^L)} \right)^{\beta} . \tag{3.13}$$

Of course, doing this we are not simulating the true posterior anymore; rather, we will get an approximate computation for the modified estimator

$$Q_{\pi,\beta}(Y_1 | X_1, Z_2^N) = \sum_{K \in \mathcal{Q}_d} \frac{\pi_2(K) S(K, Z_2^L)^{\beta}}{\sum_{K'} \pi_2(K') S(K', Z_2^L)^{\beta}} \left(\frac{S(K, Z_1^L)}{S(K, Z_2^L)} \right) . \tag{3.14}$$

(The introduction of this parameter and this modified version of the posterior has been suggested to us by the work of Catoni [23] on "Gibbs estimators").

Data Set	Size	Nb. features	Nb. classes
Breast Cancer	683	9	2
Pima Diabetes	768	8	2
Yeast	1484	8	10

TAB. 3.1: *The datasets used in the experiments.*

It appeared experimentally that usually taking some suitable $\beta \in (0,1)$ provided noticeably better classifiers than the original "plain" algorithm (equivalent to taking $\beta = 1$). This our opinion, which we will discuss later in the paper, than this improved performance is not (or at least not only), as one could think at first glance, due to faster convergence of the Monte-Carlo chain (and thus better approximation) when $\beta < 1$ (i.e. "higher" temperature), but is an intrinsic property of the estimator $Q_{\pi,\beta}$.

3.4 Experimental results

We tested the algorithm on three "classical" datasets taken from the UCI database (available at `ftp.ics.uci.edu/pub/machine-learning-databases`), namely "Breast-cancer-Wisconsin", "Pima-Indian-Diabetes", and "Yeast". A summary of these datasets is given in table 3.1. All the features are real-valued. For the Breast Cancer database, 16 examples with missing data (out of an original 699) where thrown away.

3.4.1 Base protocol

The base protocol makes use of the extension mentioned above under the paragraph "continuous set of questions". Half of the examples of the sets are used for training and the others are kept as a test set. The default parameters used in the experiments are the following: 5000 iterations of "burn-in" and then 5000 recorded iterations; the training examples are split into estimation and learning sets of equal sizes; the branching parameter ρ is 0.4; the maximal tree depth is 8. The same 5000 recorded iterations are then used for estimation and classification of all examples of the test set. Each experiment is repeated 200 times (re-drawing randomly the training and test sets each time) and the error bars in the graphics show a 95% confidence interval of the estimated quantities for a Gaussian approxi-

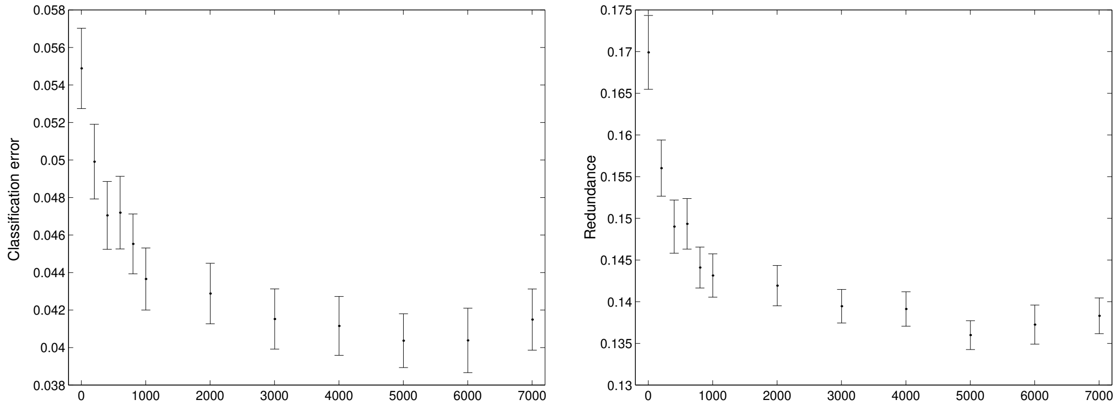


FIG. 3.1: *Classification error and redundancy vs. number of iterations (after burn-in) obtained on the breast cancer dataset.*

mation. We report classification errors which are our primary interest, but also the empirical redundancy[1] of the classifier, since the classifier was built in [14] to minimize this loss criterion. One can see on the figures that these two criterions are closely related in behavior except in some particular cases.

3.4.2 Number of iterations

The choice of the number of iterations for a Metropolis algorithm is always delicate. Due to the complex structure of the space considered, it is difficult to give *a priori* estimates of the number of steps required. The results in figure 3.1 show the evolution of the error rates as the number of iterations is increased for two of the datasets. While the error rate looks stationary after 5000 steps (a first 5000 steps are performed before as a "burn-in"), it is not necessarily an evidence that the MCMC has reached its stationary distribution, because the space considered obviously has many local extrema. However, the mixing properties of this chain are certainly better than those of a chain where the tree structure itself is changed at each step.

1. For two discrete probability distributions P, Q the redundancy of Q with respect to P is defined as $E_P[-\log(Q)]$. It is, up to the constant $E_P[-\log(P)]$, the same quantity as the K-L divergence. We estimate it by replacing P by the empirical distribution of the test set.

Data Set	2 set	1 set
Breast Cancer	3.8%	2.7%
Pima Diabetes	25.3%	24.3%
Yeast	44.3%	41.2%

TAB. 3.2: *Dividing (equally) the training set into estimating and learning set versus using the same set (the full training set) for both (table shows best results wrt. choice of the inverse temperature β in each case)*

3.4.3 Repartition between learning and estimation sets

The choice of the parameter L, that is, of the respective size of the estimation and learning sets, is a question arising naturally. Figure 3.2 shows the error rates drawn with respect to the ratio L/N, as well as the mean size of a tree and the average depth reached by a test example[2]. The conclusion is that in general the algorithm is quite robust with respect to the choice of L, although we can guess that the best value is for L/N probably a little over 0.5. It is interesting to note as well that in this case the best performance also corresponds to the larger average size of the trees.

While this results give us more insight onto the general behavior of the algorithm, this issue is however of reduced practical interest, because when we choose to use the full set Z_1^N both as estimation and learning set as suggested in the previous section, we get systematically better classifications rates (see table 3.2). This is the protocol we will use from now on.

3.4.4 Branching parameter and other priors

Another parameter we have to choose is ρ — the parameter of the binary branching process. As shown in figure 3.3, while as a rule of thumb $\rho \approx 0.4$ seems to be a good value in general, the algorithm appears to be quite robust with respect to the choice of ρ. In our opinion, this reduces the interest of an in depth discussion about the choice of the prior π_1. We also tried, in experiments not shown, the extended prior suggested in the previous section, namely, since we can actually let ρ depend on the node considered in the tree, it seems a reasonable try to choose ρ_s to be close to 1 for the nodes s close to the root in order to

2. A description of how to compute the average size of the tree models and the average depth reached by an example on a given s-tree is given in Appendix 3.6.2

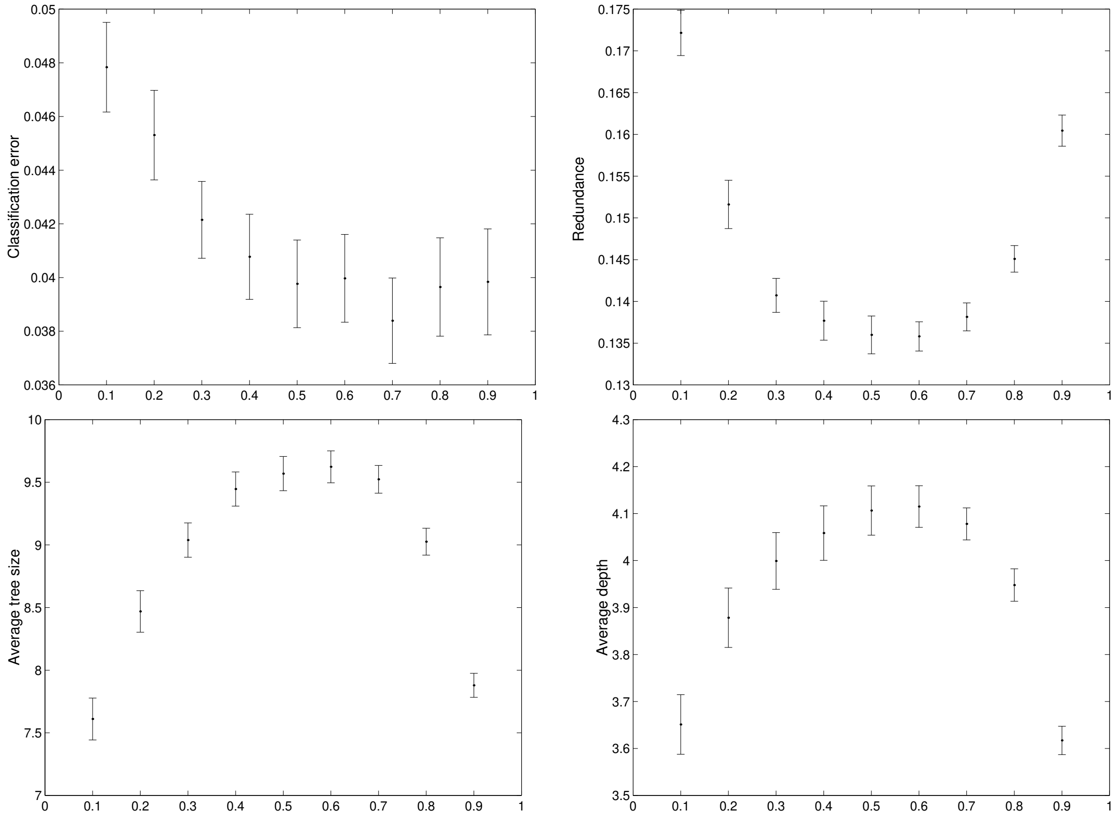


FIG. 3.2: *Influence of the repartition of training examples between learning and estimation on various characteristic quantities. X axis: fraction of the training sample used for estimation. (Breast Cancer dataset)*

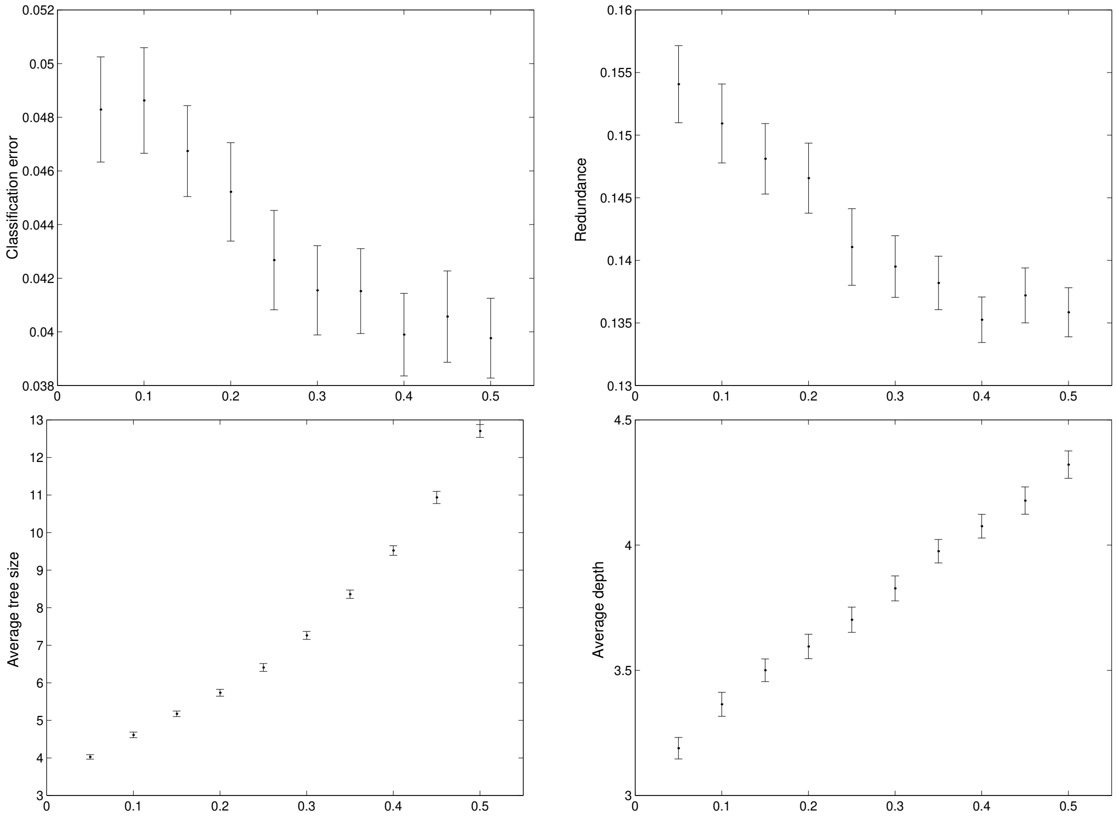


FIG. 3.3: *Influence of the branching parameter ρ. (Breast Cancer dataset)*

down-weight the very shallow trees (which are given a high prior if ρ is fixed). Nevertheless, using this kind of prior we noticed a but very slight improvement of performance, of little statistical significance, which strengthens the evidence that the algorithm is robust with respect to the choice of the prior. It is noticeable though that the parameter has an impact on the average size of the trees, which was to be expected (A larger ρ gives a relatively higher *a priori* probability to large trees). Although it seems profitable at first to rise the weight of larger trees, it seems that for $\rho > 0.4$ the classification error stays stationary while ρ and the average tree size are still growing.

3.4.5 Reduction of the number of trees

An obvious drawback of the method proposed it that it is computationally intensive. In our experiments, we usually drew a single sample path of the chain using the training and estimation sets, recorded it and then used the average of all the distributions predicted by the trees along this path to classify all the examples of the test set. This usually means a few thousand trees. On the other hand, other authors dealing with some other algorithms to generate random trees (see e.g. [2]) usually achieve their best performance with only a few dozens of trees. In order to reduce the number of trees used, we tried the following procedure: sample k independent paths using the Metropolis algorithm, every time keeping only the last tree. Then for classification of the test set use only the k trees thus built. On the databases used, we were able to obtain results as good as averaging over one single sample path with only 20 to 30 trees. Since we have to run 30 times the Metropolis algorithm, of course the part involving the tree building is now even more computationally intensive. But the important point is that once the classifier (the k trees) is built, we only have to deal with a reduced number of trees for effective classification (and this is the part where we are more interested to reduce the computation time in the first place). (An alternative method which we did not test would be to keep only a fixed number of trees on a single sample path, for example one every thousand.)

3.4.6 Temperature parameter

The temperature parameter β introduced in the previous section was found to have a strong influence on the results. Figure 3.4 shows curves of classification rates vs. β. These appear to be coarsely convex for $\beta \in [0, 1]$, and usually have a minimum for some intermediate value of β. Surprisingly enough, for the Breast Cancer data set, the value $\beta = 0$ is better than $\beta = 1$ — while $\beta = 1$ corresponds to the standard Bayesian estimator studied so far, $\beta = 0$ is equivalent to drawing questions completely randomly at every node of the tree. (In practice, the new question drawn at a node of the tree is always accepted at each step of the Metropolis chain.) Note, however, that at each step, once the questions are fixed, the weighted average over all the subtrees of the complete maximal tree is still done according to the "true" posterior, that is to say, the parameter β only influences the posterior distribution of the *questions* while the posterior on the *tree structure*, given the questions, remains the same. In other words, β has an influence on the distribution simulated on the space of s-trees, but not on the way an s-tree estimator is computed. We have tried to introduce this temperature parameter for the posterior on the tree structure as well, but it yielded a dramatic deterioration of the classification rate for $\beta < 1$, negating the beneficial effects on the posterior

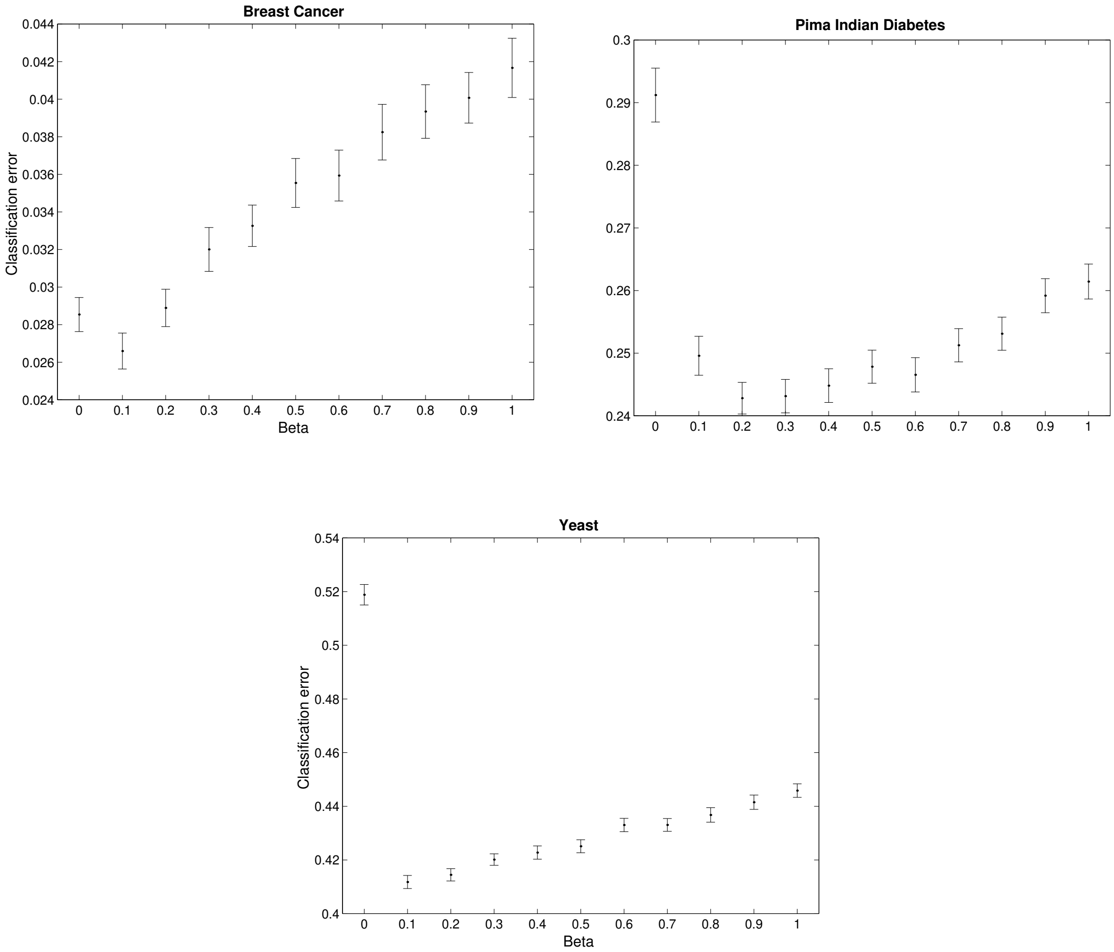


FIG. 3.4: *Influence of the inverse temperature β on classification error*

on the questions.

How can we explain these results? Obviously, a lesser value for β has a "flattening" effect on the posterior — in some sense, it reduces artificially the size of a test sample from L to βL. So why do we get better results? The first thought that comes in mind is that the number of iterations of our Metropolis-Hastings chain is not sufficient for $\beta = 1$ and that it keeps being caught in some local minima. When $\beta < 1$, the energy landscape is flatter and the chain would reach its stationary state sooner. This would compensate for the formal reduction of the sample size. To check for this hypothesis we did the following experiment: start by performing the modified algorithm with $\beta < 1$ chosen to be heuristically the "best" one (from the previous table of results). Then, switch to $\beta = 1$ and perform a few more thousand iterations. Keep the last tree obtained, and start over again in order to obtain 25 trees. Remember that in the previous paragraph we showed that drawing a limited number of trees from such independent Metropolis chains does as well as averaging over thousands of trees along one path of the chain.

If the explanation was right, the iterations made at $\beta = 1$ should not drive the chain too far away from the region it reached with a lower β and thus we should get equivalent or better results than those obtained with a fixed $\beta < 1$. But his was not the case: by acting this way we got back the same performance results that with the "plain" $\beta = 1$ algorithm. This must show, as claimed earlier in the paper, that the better performance obtained with low β is not caused (or at least not mainly) by better convergence rates of the MCMC chain.

We suggest two tracks for explaining the better classification rate at "higher" temperature. First, as noted earlier, the idea of introducing such a temperature parameter was suggested by Catoni's "Gibbs estimators" [23]. In this paper Catoni presents an in-depth study of a very general form of such modified Bayesian estimators where the likelihood part of the posterior is modified by an similar inverse temperature exponent β. However, as pointed out earlier, a full Gibbs estimator in our case would involve to use the β exponent also when computing the partial sums over all subtrees when the questions are fixed, and when we tried to do so the classification rates deteriorated.

A more heuristic explanation relies on the work of Geman and Amit [4], and more recently Amit, Blanchard and Wilder [1] where they suggest that "aggregated trees" (i.e. precisely a classifier obtained by an average of trees) perform better when the classifiers are *weakly dependent conditional on class.* On a very broad class

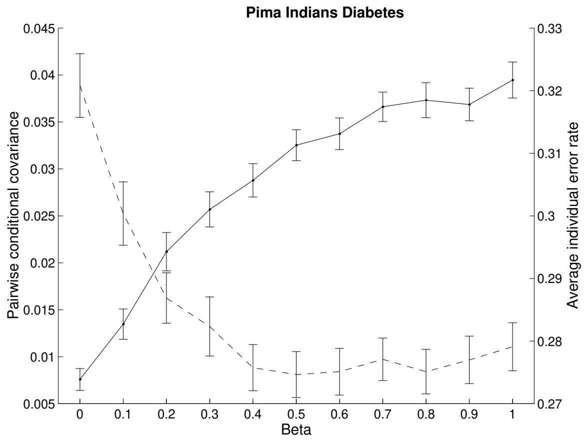


FIG. 3.5: *Solid line shows pairwise conditional covariance of s-trees, dashed line shows average error rate of one single s-tree versus β.*

of algorithms generating multiple classifiers, there is actually a tradeoff between this weak dependence and the individual performance of each tree.

We show in figure 3.5 the average covariances (conditional on class) between pairs of s-trees when the parameter β varies, and notice that the dependence is indeed weaker when β gets smaller. From this point of view, it makes more sense to use some $\beta < 1$ for the posterior probability of the questions, but not when performing the sums over subtrees once questions are fixed: we want to have more freedom on the choice of questions since it will very probably decrease the dependence between trees. But if we fix the questions, then all the classifiers obtained as subtrees of the complete tree are likely to be very correlated whatsoever, so our better choice is probably to concentrate on the better ones, i.e. to have a more peaked (conditional) posterior, rather than a spread out posterior.

3.5 Discussion

Generation of classifier trees by MCMC procedures has received increased interest recently. One of the main problems encountered so far is that very often, when the chain movements concern the tree structure itself, the chain can get caught into local minima of the energy landscape (see for example [25]). The algorithm we propose has several advantages: we only have transitions on the questions asked at the internal nodes of the tree. Taking into account the different possible tree structures is made automatically and in an exact way. As a consequence we avoid many of the problems linked to the chain being caught locally, and obtain better mixing properties. Also the number of "trees" truly averaged is much greater in sheer number (remember that for a fixed set of questions on a complete tree of depth d, there are of the order of 2^d subtrees which are all taken into account). The classification rates obtained on the "breast-cancer" and "diabetes" datasets (see table 3.2) are similar or better than those of various protocols generating multiple tree classifiers to be found in e.g. [19]. (The third dataset "Yeast" does not appear in this last reference)

In terms of computing power needed, one step of the chain requires of course more computation than when single trees are considered. The structure averaging algorithm is very fast and efficient however. On the examples studied, we were able to run 10000 iterations of the chain in the order of one minute on a SPARC Ultra5. For practical classification problems, the critical point should be the computation time needed to "ask" a question on a training example. Namely, when we change a question at an internal node s of the tree we need to collect all training examples present at that node and let them follow the branches again until they reach a leaf. This means asking $d - d_s$ questions per training example present at node s, where d_s is the depth of node s. In the examples studied here, asking a question does not require any computation so this was not much of a problem; in other situations (for example the complicated queries taken into consideration by Amit and Geman [2] for handwritten digit recognition), this might be a serious bottleneck. Note however that this problem has always been present in previous Bayesian trees MCMC procedures in the literature so far.

On the other hand, if true that constructing the chain is relatively computationally intensive, classifying a new test example is actually very fast if we only finally keep 30 trees.

Finally, we want to point out once again the key role of the inverse temperature parameter β. In practice it works as a tuning parameter that allows the user to balance individual classifier performance versus pairwise conditional dependence. It controls the spread of the distribution on the space of s-trees in a straightforward way, as recommended by Amit et al. [1] for multiple randomized classifiers. In practice a good value of β can be determined by cross-validation, but a more theoretical explanation on why and how to choose the parameter is probably an interesting track for future research.

3.6 Appendix

3.6.1 The tree weighting algorithm

We review here briefly Willem, Shtarkov and Tjalken's recursive "tree weighting" algorithm [59] for efficient exact calculation of sums of the form

$$S(f,g) = \sum_{T \in \mathcal{T}_d} \left(\prod_{s \in \delta T} f(s) \prod_{s \in T \setminus \delta T} g(s) \right) \tag{3.15}$$

where $\mathcal{T}_d$ is the set of all complete binary trees of depth less than d, δT is the set of terminal nodes (leafs) of the tree T, f is some real function defined on the nodes (both internal and terminal) of the maximal complete binary tree of depth d, denoted by $\mathcal{M}_d$, and g is another real function defined on $\mathcal{M}_d \setminus \delta\mathcal{M}_d$. A node of a tree is defined as a finite binary chain, element of $\{\emptyset\} \cup \{0,1\}^*$ ($\emptyset$ is the tree root; if s is a node, by definition its two descendants are $s0$ and $s1$). A tree T is defined as the collection of its nodes.

Let us define recursively the function σ on the nodes of $\mathcal{M}_d$ by:

$$\sigma(s) = \begin{cases} f(s) & \text{if } s \text{ is a terminal node of } \mathcal{M}_d \\ f(s) + g(s)\sigma(s0)\sigma(s1) & \text{otherwise.} \end{cases} \tag{3.16}$$

Then it can easily be shown that $\sigma(\emptyset) = S(f,g)$, and so we can compute $S(f,g)$ in $\mathcal{O}(2^d)$ operations.

Now it is easy to see that the quantity $S(K, Z_1^P)$ defined in equation (3.6) is of the form $S(f,g)$ for a certain couple (f,g). On the one hand, the specific prior π_1 on the tree structure can be written as

$$\pi_1(T) = \rho^{|T \setminus \delta T|}(1-\rho)^{|\delta T \setminus \delta\mathcal{M}_d|}.$$

On the other hand, let us define the counters $n_s(i)$ as the number of examples of class i of the estimation set E whose path down the tree (recall that the set of questions K attached to the nodes is fixed) goes through node s, and $k_s(i)$ as the similar counters for the set Z_1^P (a subset of the learning set Λ), then we have

$$\prod_{i=1}^{P} Q^E_{(T,K)}(Y_i|X_i) = \prod_{s \in \delta T} \prod_{i=0}^{c} \left(\frac{n_s(i)+1}{\sum_j n_s(j) + c + 1} \right)^{k_s(i)},$$

so that we finally get the announced property if we define the functions f, g as

$$f(s) = \begin{cases} (1-\rho)\prod_{i=0}^{c}\left(\frac{n_s(i)+1}{\sum_j n_s(j)+c+1}\right)^{k_s(i)} & \text{for } s \in \mathcal{M}_d \setminus \delta\mathcal{M}_d, \\ \prod_{i=0}^{c}\left(\frac{n_s(i)+1}{\sum_j n_s(j)+c+1}\right)^{k_s(i)} & \text{for } s \in \delta\mathcal{M}_d, \end{cases}$$

$$g(s) = \rho \text{ for all } s \in \mathcal{M}_d \setminus \delta\mathcal{M}_d.$$

3.6.2 Calculation of the average tree size

During the run of the MCMC chain we want to keep trace of relevant quantities such as the average number of leaves in the tree $|\delta T|$ (or the average total number of nodes in T which is $2|\delta T| - 1$ for a complete binary tree). When the set of questions K on the tree are fixed, the average number of leaves (conditional on these questions) is

$$L(K) = \frac{1}{Z_0(K)} \sum_{T \in \mathcal{T}_d} \pi_1(T)|\delta T| \prod_{s \in \delta T} Q^E_{(T,K)}(Y^s|X^s)$$

where $|\delta T|$ denotes the set of leaves (terminal nodes) of the tree T, Z_0 is the normalizing constant, and the superscript s means that we take the subset of examples that have reached node s: $X^s = \{X_i | T(X_i) = s\}$.

Although this sum cannot be directly factorized as in appendix 3.6.1, we have

$$\begin{aligned} L(K) &= \frac{1}{Z_0(K)} \frac{\partial}{\partial \xi} \left(\sum_{T \in \mathcal{T}_d} \pi_1(T)(1+\xi)^{|\delta T|} \prod_{s \in \delta T} Q^E_{(T,K)}(Y^s|X^s) \right)\Bigg|_{\xi=0} \\ &= \frac{\partial}{\partial \xi} \left(\log Z_\xi(K)\right)\Big|_{\xi=0} \end{aligned}$$

where

$$Z_\xi(K) = \sum_{T \in \mathcal{T}_d} \pi_1(T) \prod_{s \in \delta T} (1+\xi) Q^E_{(T,K)}(Y^s|X^s) \tag{3.17}$$

which now can be computed using the factorization method given in appendix 3.6.1, so that we can approximate $L(K)$ by $(Z_\xi(K) - Z_0(K))/\xi Z_0(K)$ for ξ small enough. To control the precision we can write

$$
\begin{aligned}
|(1+\xi)^n - 1 - n\xi| &= \int_0^{\xi} x \left.\frac{\partial^2}{\partial u^2}(1+u)^n\right|_{u=x} dx \\
&\leq \frac{n(n-1)}{2}\xi^2(1+\xi)^{n-2} \\
&\leq \frac{(n\xi)^2}{2}\exp{(n\xi)}
\end{aligned}
$$

so that in practice for example with $n \leq 256$ (i.e. for a maximum tree depth of 8) $\xi = 1.10^{-3}$ gives us a .05 accurate value (note that since we are integrating $(1+\xi)^{|\delta T|}$ with respect to a probability distribution we get the same precision on $L(K)$). We can then average this quantity over the run of the MCMC chain.

Similarly, we can compute the empirical average depth reached by the test examples: given one test example X, when the questions K are fixed, the average depth reached by X can be approximated by the same principle as above if we replace the factor $(1+\xi)$ in (3.17) by the function

$$
h(s,\xi) = \begin{cases} (1+\xi)^{|s|} & \text{if } s \text{ is on the branch followed by } X \\ 1 & \text{otherwise} \end{cases}
$$

where $|s|$ denotes the depth of the node s.

Chapitre 4

Multiple Randomized Classifiers

Joint work with Pr. Yali Amit, University of Chicago.

Résumé

We attempt to provide a unifying explanation to the success of aggregation protocols of multiple classifiers. We show that all successful protocols are producing weakly dependent classifiers conditional on class. This is pointed out as an important reason both for decrease in the bias error of the aggregate classifier relative to each individual one, as well as for the reduction in variance with respect to the training sample. We argue that all protocols including deterministic Boosting can be viewed as producing a sample from some distribution on the space of classifiers. There is a tradeoff in terms of two key expectations with respect to this distribution: the margin and the average conditional covariance. We study these points using a simple running example on which we also illustrate many of the properties we derive.

4.1 Introduction

In recent years a multitude of algorithms have been suggested in which a large number of classifiers are trained on the same training sample and then combined to provide an improved *aggregate classifier*, [3, 2, 17, 54, 19, 44, 34, 14, 33, 39, 20]. Mostly the ideas have been presented in the context of classification trees but similar procedures can be implemented with other classifiers. We argue that in all cases the classifiers can be viewed as a random sample from a distribution $\mathbf{Q}$ on the space of classifiers, which depends on the protocol and on the training set, hence the acronym Multiple Randomized Classifiers (MRCL). In many cases the sample is truly i.i.d. For example in [2] the split at each node of a tree is chosen as the best from among a small random sample of all possible splits, this is revisited in [20]. The protocols in [17, 19, 33, 14, 43], suggest other forms of randomization. We also include in our setting the ADABoost algorithm [54], although it was originally defined as a completely deterministic process: one the one hand, any convex combination can actually be considered as a probability distribution. On the other hand, we argue that Boosting is a dynamical process in the space of classifiers which, at least in certain cases, seems to asymptotically mimic a limiting distribution, much like an ergodic process.

Intuitively randomization is a method to obtain multiple points of view or "experts", which are then aggregated using some form of voting. The intuition behind Boosting is the need for experts which concentrate on the difficult examples, or alternatively experts who actually "disagree" on these difficult examples.

The success of MRCL stems from the fact that a sample from a very rich and diverse set of classifiers produces on average *weakly dependent classifiers conditional on class.* We will show that this weak conditional dependence among the classifiers drawn from $\mathbf{Q}$ leads to reductions in both bias and variance relative to the individual classifiers. There is a clear tradeoff in terms of two expectations calculated with respect to $\mathbf{Q}$. The first, called the *average conditional margin* (ACM), is the difference between the expected weight on the correct class and the expected weight on the other classes, conditional on the correct class. The second, called the *average conditional covariance* (ACC), expresses the degree to which two classifiers sampled independently from $\mathbf{Q}$ have a low conditional covariance. Experiments clearly show that a significant reduction in the ACM, which implies a reduction in the individual classification rates, can lead to a stronger aggregate classifier if the dependence between the classifiers, i.e. ACC is reduced.

The paper is organized as follows. In section 4.2 we present an overview of some recent work on the subject. In section 4.3 we offer a description of the various protocols. In section 4.4 we define the average conditional margin and average conditional covariance, and bound the classification error with an elementary Chebychev-type inequality making use of the two quantities above. We present an overview of the experimental results obtained, focusing on the role played by these two quantities.

In section 4.5 we present the experimental results with more detail and discussion. We introduce a simple synthetic running example on which we can easily visualize the distribution **Q** on the space of classifiers. We discuss the distributions built by different protocols and compare their performance, also introducing a new protocol, explicitly designed to perform a sequential minimization of the aggregate classifier covariance. We finally devote a separate subsection to Boosting and the specific problems raised by this protocol

In section 4.6 we present some more experimental evidence for the claims we make in the context of shape recognition, and in section 4.7 we list some of the main conclusions.

4.2 Related work

The emphasis on weak conditional dependence presented in this paper has appeared in [3, 2, 19]. In these papers the role of weak conditional dependence in bias reduction is discussed in the context of shape recognition. This concept has not been addressed in most of the recent discussions of aggregation, e.g. [17, 19, 54, 39]. Only recently has it been studied in [33], [20] and [49]. In the first, this dependence is measured in an *unconditioned* way. It is therefore unclear what this type of dependence is measuring and how it connects to the performance of the aggregation methods. In [20] a measure of dependence is suggested which overcomes this limitation and in the two class case is very similar to a measure of conditional dependence. The conclusions are very similar to those of [2]. In [49] the basic paradigm is that of modeling the conditional dependence between the classifiers, but there is an attempt to define this in terms conditional exponential moments. Given these moments one obtains more precise error bounds, however the moments are very hard to estimate robustly.

We review here other explanations for the success of aggregation methods. In the papers [19, 17], Breiman puts the emphasis on the reduction of variance with respect to the training set. In some sense it is argued that ensemble methods are supposed to mimic a semi-ideal situation where one had a large number of independent training sets of the same size, each of which would be used to produce one classifier. These would then be aggregated. However Breiman himself notes that this is only part of the story. For example if the base classifier is very stable, this variance reduction scheme alone will yield negligible improvement.

In particular, we show in some simulated experiments (see section 4.6) that drawing a new random training set to produce each classifier may not produce the same gains in classification rates as the other protocols.

Concerning the now popular Boosting algorithm, all analyses of this method proposed in the literature clearly point to the fact that Boosting is effectively reducing bias by constructing a compound classifier in the space of convex combinations of base classifiers, which is of course richer than the space of individual classifiers themselves. This appears in the initial motivations for Boosting in [38] as well as in recent attempts to provide an interpretation of Boosting in the framework of generalized additive models [39]. Although there are pretty clear theoretical reasons why Boosting dramatically decreases the classification error on training, there is still a lack of theoretical explanation as to why Boosting also does so well on the test set.

The explanation proposed by [54] is based on an extension of a Vapnik-type inequality in the space of convex combinations of classifiers, and on the empirical distribution of the margins. The key property employed in this explanation is then that Boosting actively reduces the empirical "margin" of the aggregate classifier. However [18] shows experimentally, using ad hoc algorithms designed for this purpose, that concentrating only on maximizing the margins can lead to over-fitting and degradation in performance.

In [39] a more transparent derivation of the weighting scheme of Boosting is provided. They show that in the two class case where the two classes are labeled $-1, 1$ and the classifier outputs a positive or negative value, Boosting can be viewed as a procedure to iteratively minimize a cost function of the form

$$C_{n+1} = E \exp[-Y(H^n(X) + \beta_{n+1} h_{n+1}(X))],$$

where H^n is the current aggregate, h_{n+1} is the new classifier, Y is the class label, and β_{n+1} is a free parameter chosen to yield maximum decrease of the cost once h_{n+1} has been chosen. It has more recently be proven in [27] that the Boosting procedure (in the ideal case where the base classifier having least weighted error is picked at each iteration) indeed leads to minimizing asymptotically the cost function $\mathcal{C}(H) = E[\exp(-YH(X))]$, where H is any combination with positive coefficients of base classifiers and the expectation is the empirical expectation on the training set.

The problem with this explanation is that it is entirely dedicated to the training set. In the context of a simple running example used below we show that C_n

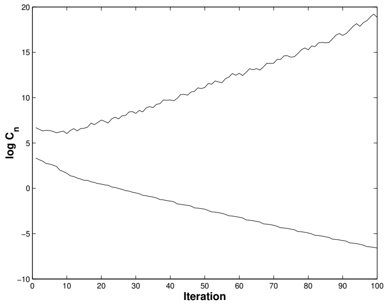


FIG. 4.1: $\log(C_n)$ *as estimated from training data (lower curve) and from test data.*

diverges exponentially on the test set while converging exponentially to zero on the training set. This is demonstrated graphically in figure 4.1 where $\log(C_n)$ is plotted both for the training sample and the test sample in a run of ADAboost for the running example defined in section 4.5.1. Moreover, we argue in section 4.5.4 that the sole minimization of the above cost function cannot explain the success of Boosting and that its *dynamical* behavior is more important.

While we acknowledge that Boosting is indeed constructing a compound classifier by making full use of the rich structure of convex combinations of base classifiers, the alternative and in a sense more "naive" point of view we emphasize in this paper is to study this convex combination as a probability distribution on the space of classifiers and to concentrate only on elementary, but robust, properties of this distribution (i.e. the average conditional margin and the average conditional covariance of two classifiers drawn independently from the distribution). We show that these properties are found to be common to all successful algorithms used to build multiple classifiers.

We reiterate: the comparable performances of ARCing, Boosting, randomization etc, and the various combinations thereof means that the explanation must lie in something which is common to all protocols, not the particular details of any one of them. We believe it is due to the effective sampling from a distribution in the space of classifiers with a good balance between the average conditional margin and average conditional covariance.

4.3 Protocols for generating multiple classifiers

The notations used are the following: X denotes an object to be classified, Y is its associated class belonging to the set $\{1, \dots, K\}$. We assume that we have at hand a (finite of infinite) set $\mathcal{H}$ of classifiers. A classifier $h \in \mathcal{H}$ is a function associating to every object X a probability distribution over $\{1, \dots, K\}$, that is, a vector of sum one belonging to $[0,1]^K$, that will be denoted by $h(X,d), d = 1, \dots, K$. In our experiments we will concentrate on classification trees but we will refer to abstract classifiers whenever the explicit form of the classifiers is unimportant.

Given a training set $\mathcal{L}$ for a classification problem with K classes we produce a sequence of classifiers $h_1, \dots, h_N$ using one of several protocols.

1. **Bootstrap and AGgregate, BAGGING.** To build each classifier a bootstrap sample from the training data is produced. This weighted sample is then used to build a classifier [17].

2. **Bayesian sampling.** A prior distribution on classifier space is determined. The posterior distribution on classifier is determined by multiplying the prior by a likelihood which is given by the probability assigned to the data by a classifier. A sampling procedure such as a MCMC samples from this posterior and the aggregate classifier is given by the average over sampled classifiers. See [25, 14] for explicit construction in the case of classification trees.

3. **Error correcting optimal coding, ECOC.** The classes are randomly partitioned into two "super classes". A classifier is grown to classify between the super classes [34].

4. **Boosting.** An initial uniform weight vector W_1 for the training data is set, $W_1(l) = 1/L, l = 1, \dots, L$. A classifier h_1 is built to fit the training data. The training error ε_1 of the classifier is obtained. The misclassified points in the training set have their weight increased by a factor of $\alpha \cdot (1 - \varepsilon_1)/\varepsilon_1$. The weights of the training data are renormalized to produce the new weight vector W_2. After $n-1$ classifiers $h_1, \dots, h_{n-1}$ have been produced and given an updated weight vector W_n, a new classifier h_n is built to fit the weighted training sample. The weighted training error ε_n of T_n is evaluated in terms of the weighted training data, the weights on the misclassified points are multiplied by $\alpha \cdot (1 - \varepsilon_n)/\varepsilon_n$, and the entire weight vector normalized to produce W_{n+1}. In [54], the original ADAboost uses $\alpha = 1$, but we keep this as a free parameter. The value $\alpha = 0$ will denote no reweighting.

Additionally, in the specific case when classifiers are decision trees, the following protocols are available:

5. **Randomized splits (a), RSa.** At each node a random subset of size **S** of all candidate splits is entertained. The best split from this subset is chosen. See for example [3, 2, 58].

6. **Randomized splits (b), RSb.** At each node the top M splits are found and one split is chosen at random from among them [33].

Finally, also in the case of classification trees we present experiments using a "randomized Boosting" procedure, which is the Boosting protocol in which we use at each iteration the **RSa** method above to build a tree from weighted data. In this case we recall that we keep as free parameters the coefficient α (see above) and **S**, i.e. the size of the random subset of features sampled at each node. When **S** is taken equal to the total number of splits, there is no randomization and we have a deterministic Boosting procedure.

In all the protocols, the aggregate classifier is then produced by averaging the output vectors $h_n(X,\cdot)$ over the sequence of classifiers, namely given a set of weights $(\beta_i)_{i=1,\dots,N}$, we define the aggregated distribution vector $H^N(X,\cdot)$ and classifier $C^N(X)$ as

$$H^N(X,d) = \sum_{n=1}^{N} \beta_n h_n(X,d) \text{ and } C^N(X) = \text{Argmax}_d H^N(X,d). \tag{4.1}$$

The β_i can be uniform weights or specific weights produced from the protocol. For the original Boosting,

$$\beta_n = \frac{1}{2}\log\left((1-\varepsilon_n)/\varepsilon_n\right), \tag{4.2}$$

where ε_n is the weighted empirical error of h_n. In our experiments, weights are given by in equation (4.2) for the **RB** protocol, and uniform of sum one for the other protocols.

4.4 Asymptotic analysis in the number of classifiers

In this section we present an analysis of MRCL asymptotically in the number of classifiers. In other words we assume that the process $h_1, h_2, \dots, h_N, \dots$ is ergodic and the empirical average as given in equation (4.1), can be replaced by an expected value with respect to the underlying distribution $\mathbf{Q}$ on the space of classifiers which is determined by the protocol and by the training sample.

We therefore replace H^N defined in equation (4.1) and the associated classifier by

$$H_{\mathbf{Q}}(X) = E_{\mathbf{Q}}[h(X)], \text{ and } C_{\mathbf{Q}}(X) = \text{Argmax}_d H_{\mathbf{Q}}(X, d). \tag{4.3}$$

In the case of randomized splits for classification trees, i.e. $\alpha = 0$, the ergodicity assumption follows directly from the fact that the trees are sampled independently from $\mathbf{Q}$. For each tree, the random sample of splits from which the optimal one is chosen is sampled independently. Similarly, for bagging each bootstrap sample is drawn independently so that the classifiers are sampled independently from $\mathbf{Q}$; this is also the case for the **ECOC** protocol while the Bayesian sampling protocol also produces an ergodic process by construction.

For the Boosting protocol and in particular deterministic Boosting, one can wonder if the sequence of classifiers can be viewed as an ergodic average as well. We present a more detailed discussion on this point in section 4.5. In our experiments, Boosting seemed to behave like an ergodic process; this fact has also been reported by other authors [20]. On the other hand, it is known ([39, 27]) that, in the two class case, assuming that the class variable Y and the base classifiers take their values in $\{-1; 1\}$, Boosting aims at minimizing the empirical loss function $E_N[\exp(-YF(X))]$, where E_N is the expectation over the training set, and F belongs the set of all linear combinations with nonnegative coefficients of base classifiers. If this minimum is attained for a unique F, then the aggregate classifier at step N, H^N, should converge towards F, which, once suitably renormalized, can be seen as a convex combination of base classifiers. So in any case we can consider the limit behavior of H^N as an average of base classifiers with respect to some distribution $\mathbf{Q}$.

4.4.1 A variance decomposition

In the analysis to come we will make use of three different probability distributions:

- The probability distribution of the sample $\mathcal{L}$, denoted by $P_{\mathcal{L}}$;
- The probability distribution $\mathbf{Q}$ on the space of classifiers (depending on $\mathcal{L}$ through the classifier production protocol);

– The conditional probability distribution of a new object X given $Y = c$, denoted by P_c.

The associated expectation and variance operators will be denoted with the appropriate indices, i.e. $E_c, E_{\mathcal{L}}, E_{\mathbf{Q}}$, etc. Multiple indices indicate that these operators are taken with respect to the corresponding product distributions.

For a fixed training set $\mathcal{L}$ define the vectors of average margins on a new example of class c as $M_c(d) = E_c[H_{\mathbf{Q}}(X, d)]$, for $d = 1, \ldots, K$. The *average conditional margin* (ACM) for class c is now defined as

$$\theta_c = \min_{d \neq c}(M_c(c) - M_c(d)). \tag{4.4}$$

We assume that $\theta_c > 0$: in our experiments we found that this was always satisfied in cases of interest. This assumption is however quite natural: one would not expect good classification results when it is violated.

Assuming $\theta_c > 0$ the error rate for class c depends on the extent to which the aggregate classifier $H(X, d)$ is concentrated around $M_c(d)$ for each $d = 1, \ldots, K$. The simplest measure of concentration is the variance of $H(X, d)$ with respect to the distribution P_c. Using Chebyshev's inequality we write a coarse bound on the misclassification probability with respect to P_c as follows.

$$\begin{aligned} P_c(C_{\mathbf{Q}}(X) \neq c) &\leq P_c(H_{\mathbf{Q}}(X, c) < M_c(c) - \theta_c/2) \\ &\quad + \sum_{d \neq c} P_c(H_{\mathbf{Q}}(X, d) > M_c(d) + \theta_c/2) \\ &\leq \sum_{d=1}^{K} P_c(|H_{\mathbf{Q}}(X, d) - M_c(d)| > \theta_c/2) \\ &\leq \frac{4}{\theta_c^2} \sum_d Var_c[H_{\mathbf{Q}}(X, d)]. \end{aligned} \tag{4.5}$$

Of course Chebyshev's inequality is coarse and will not give very sharp results in itself, be we state it here as a landmark pointing to the relative importance of margin and variance.

We rewrite each of the variance terms of the last equation as

$$\begin{aligned} Var_c[E_{\mathbf{Q}} h(X, d)] &= E_c[E_{\mathbf{Q}} h(X, d)]^2 - (E_c E_{\mathbf{Q}} h(X, d))^2 \\ &= E_{\mathbf{Q} \otimes \mathbf{Q}} E_c[h_1(X, d) h_2(X, d)] - E_{\mathbf{Q} \otimes \mathbf{Q}} \left[E_c[h_1(X, d)] E_c[h_2(X, d)]\right] \\ &= E_{\mathbf{Q} \otimes \mathbf{Q}} Cov_c[h_1(X, d), h_2(X, d)] \doteq \gamma_{c,d}, \end{aligned} \tag{4.6}$$

where the notation $E_{\mathbf{Q}\otimes\mathbf{Q}}$ means that h_1, h_2 are two classifiers sampled independently from the distribution $\mathbf{Q}$. We can therefore interpret this variance term as the conditional covariance of two classifiers independently sampled from $\mathbf{Q}$. We call this quantity *Average conditional covariance* (ACC) in the sequel.

In table 4.1 we show on a synthetic 2-class example (described in more detail in section 4.5.1) and for various protocols the values of this term, averaged over training sets, that is, $\widetilde{\gamma}_1 = E_{\mathcal{L}} Var_1[H^N(1) - H^N(2)]$ (and vice-versa for class two). Note that $\widetilde{\gamma}_1$ is a more natural quantity to look at in the two class case since it is exactly the average variance of the conditional margin θ_1 (so that we actually have $\widetilde{\gamma}_1 = 4\gamma_{1,2}$, as defined in eq. (4.6)).

Note how this quantity decreases as a function of the number N of classifiers. For a small number of classifiers the variance of the aggregate classifier is dominated by the variances of the individual classifiers. The magnitude of these can be read off the variance columns in the first row, where only 1 classifier is produced. As the number of classifiers increases the variance becomes dominated by the ACC term, which is the asymptotic value as shown in equation (4.6). The same phenomenon is observed in table 4.2 where we show experiments in character recognition on a synthetically generated dataset. In all tables $\alpha = 1$ corresponds to Boosting, $\alpha = 0$ corresponds to no Boosting, ($\alpha = -1$ denotes a sequential covariance minimization protocol described below in section 4.5). $\mathbf{S}$ denotes the number of random planes sampled to produce the classifier, $\mathbf{S} = 110$ denotes all possible planes and corresponds to deterministic Boosting. It appears that it is of interest to reduce the ACC to get better classification error, even if the average margin is lower. When the margin is too low, the classification rate also deteriorates, so there appears to be a compromise to reach between the two quantities. Tables 4.1 and 4.2 also show the standard deviation of the margins with respect to the training set, i.e. distribution $P_{\mathcal{L}}$. This quantity appears to decrease together with the ACC term, which suggests that there is a second advantage in reducing the ACC, namely, increased stability with respect to the training set (this is similar to what is usually called the “variance” term in the literature, e.g. [19]), which should also result in lower estimation error.

In order to test the relevance of the bound obtained by the Chebychev inequality, we performed some elementary regression calculations on the numbers presented in table 4.1. We first performed a standard least squares regression, in log scale, of the classification error with respect to the margin M and the variance

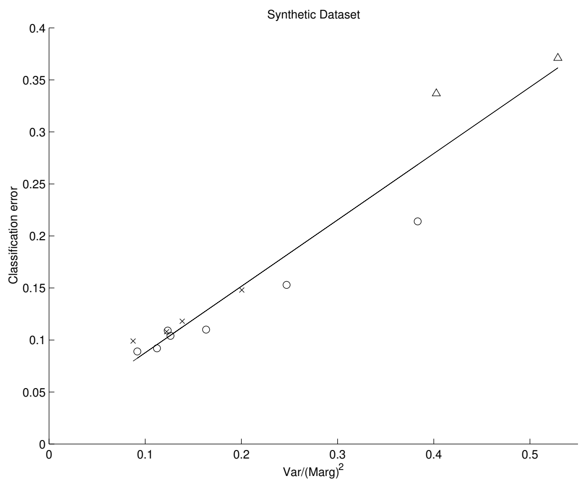


FIG. 4.2: *Error versus V/M^2 ratio with standard least squares regression line. Points are taken from table 4.1 with the point corresponding to one single base classifier removed as out of range. Circles correspond to Randomized Boosting, triangles to Randomized Splits, crosses to* **SCM** *protocol.*

V (both averaged over the two classes). We obtained a coefficient (thus to be interpreted as an exponent) of -.88 for margin and .47 for variance for a 80% R-statistic. The corresponding 5% confidence intervals (not extremely relevant *per se* since there is no reason the errors should be Gaussian) are very loose ($\pm$.7 for margin and $\pm$.15 for variance), but it is interesting to point out that the exponent of the margin is very close to twice the exponent of the variance in absolute value. We re-performed the same regression with the point corresponding to one single base classifier removed (for this point, the ratio M/V^2 is greater than 4, a value that can be considered as well out of range of the other points for which the ratio ranges between .1 and .5) and although we obtained quite different exponents (-1.7 for margin, .8 for variance for a R-statistic of 92% in log-scale), it is striking to note that the ratio between them is still very close to 2.

Finally, we plotted in figure 4.2 the error versus the ratio V/M^2 with the least square regression line obtained in linear scale: the fit is actually surprisingly good.

We don't want to draw over-interpretative conclusions from these obviously limited regression results (made for a single specific classification problem and with only 13 points). It is clear that the information given by the margin/variance alone can allow for very different actual distributions, one can even suggest from fig. 4.2 that different protocols may result in slightly different behaviors (the points corresponding to the Randomized Boosting and Sequential Covariance Minimization seem to be nicely aligned; the two points corresponding to Randomized Splits seem somewhat out of alignment with the others). However, in our opinion these results suggest that the V/M^2 ratio would in practice be a relevant quantity.

4.4.2 Conditional and unconditional dependence

It should be emphasized that two classifiers, provided that they achieve reasonable classification rate (that is, better than just picking a class at random) *will not be unconditionally independent.* If we do not know the class label of a point, and vector $(h_1(X,i))_i$ is large at class c, then we actually change our expectations regarding $(h_2(X,i))_i$. On the other hand if we were given in advance the class label Y, then knowing $h_1(X)$ would hardly affect our guess about $h_2(X)$. This is the motivation behind the notion of weak conditional dependence.

This is in contrast to the measure of dependence introduced in [33], which involves the unconditional covariance. The κ statistic used there is

$$\kappa(h_1, h_2) = \frac{\sum_d Cov[h_1(X,d), h_2(X,d)]}{1 - \sum_d Eh_1(X,d)Eh_2(X,d)},$$

A simple decomposition of the numerator yields:

$$Cov[h_1(X,d), h_2(X,d)] = ECov[h_1(X,d), h_2(X,d)|Y] + Cov[E[h_1(X,d)|Y], E[h_2(X,d)|Y]]. \tag{4.7}$$

The first term is related to equation (4.6) and should therefore be small on average over classifiers. However the second term can be relatively large since on average we expect both classifiers to produce similar results in terms of the conditional expectation $E_c h_i(X,d), d = 1, \ldots, K$. Also this second term in no way affects classification rates.

In [20] a different measure of dependence is used which although unconditional is very closely related to the conditional one defined here. It is defined as

$$\begin{aligned} &Var\left[H_{\mathbf{Q}}(X,Y) - \max_{d \neq Y} H_{\mathbf{Q}}(X,d)\right] \\ &= E_{\mathbf{Q}\otimes\mathbf{Q}} Cov\left(h_1(X,Y) - \max_{d\neq Y} E_{\mathbf{Q}} h(X,d)\ ;\ h_2(X,Y) - \max_{d\neq Y} E_{\mathbf{Q}} h(X,d)\right). \end{aligned} \tag{4.8}$$

Here $\mathbf{Q}$ denotes some distribution on classifiers. In this case the second term in the covariance decomposition of equation (4.7) will be very small, because all that is observed is the expected difference between the weight on the true class and the mode outside the true class, this should be more or less the same for all classes. Another attraction of this measure of dependence is that

$$P\left[H_{\mathbf{Q}}(X,Y) - \max_{d\neq Y} H_{\mathbf{Q}}(X,d) < 0\right],$$

is precisely the misclassification probability. However this condition is not given directly in terms of the relationship between two independently sampled classifiers. It depends in a complex way on the expected classifier.

4.5 Experimental results

4.5.1 Artificial example

In order to illustrate our statements we make use of an artificial example with two classes of points in R^2, using stumps (depth one trees) produced from a finite number of splits determined by a finite number of separating lines in the plane. A line is defined by a couple (θ, τ), where θ is an angle and τ a scalar. Setting v_θ the unit vector in direction θ, the associated classifier predicts class 1 for the "positive" half-plane $\{x \in R^2 | \langle v_\theta, x\rangle - \tau > 0\}$ and class 2 for the "negative" half-plane. Specifically we define 10 uniformly spaced angles with 11 uniformly spaced thresholds. Thus in total there are 110 possible features or splits.

In order to make the problem non-trivial the two classes are distributed according to a mixture of Gaussians. The first class has means at $(2,2)$ and $(-2,-2)$, with weights $1/4, 3/4$ respectively, and the second has means at $(-2,2)$ and $(2,-2)$ with weights $1/2, 1/2$ respectively. The skewed weights on one of the classes are specifically chosen to demonstrate certain properties of Boosting. We purposely pick a small size training set, 20 points per class or on average 10 points per cluster, to study variance properties of these classifiers with respect to the training set. Figure 4.3 shows a training sample of 20 points per-class alongside a test sample of 500 points per class.

4.5.2 Comparing the different protocols

The use of small subsets of randomized splits, which corresponds to the **RS** protocol, is discussed in [2] in a shape recognition context where there is a virtually infinite number of splits, each corresponding to a spatial arrangement of local

Protocol	Classification Rate	Average Margin(1)	Intraclass Variance(1)	Stdv. wrt. Training(1)	Average Margin(2)	Intraclass Variance(2)	Stdv. wrt. Training(2)
$\mathbf{S} = 2, N = 1$	.590	.420	.129	.559	-.057	.146	.647
$\mathbf{S} = 2, N = 10$	.786	.212	.017	.075	.177	.012	.082
$\mathbf{S} = 2, N = 20$	.847	.190	.011	.055	.181	.006	.054
$\mathbf{S} = 2, N = 50$	.896	.177	.004	.032	.156	.003	.031
$\mathbf{S} = 2, N = 100$	.911	.166	.003	.028	.164	.002	.020
$\mathbf{S} = 2, N = 1000$	.908	.135	.002	.027	.132	.002	.025
$\mathbf{S} = 50, N = 1000$	.891	.222	.006	.056	.219	.006	.052
$\mathbf{S} = 110, N = 1000$	.890	.253	.010	.064	.242	.010	.060
Randomized Boosting, $\alpha = 1$							
$\mathbf{S} = 2, N = 1$	.590	.420	.129	.559	-.057	.146	.647
$\mathbf{S} = 2, N = 20$	.629	.417	.025	.167	-.038	.013	.181
$\mathbf{S} = 2, N = 1000$	.663	.382	.021	.071	.004	.009	.071
Randomized splits, $\alpha = 0$							
$\mathbf{S} = 2, \gamma = .05$	.852	.082	.002	.011	.091	.001	.02
$\mathbf{S} = 10, \gamma = .05$	.892	.142	.003	.023	.144	.002	.031
$\mathbf{S} = 110, \gamma = .05$	.882	.160	.004	.029	.158	.003	.041
$\mathbf{S} = 10, \gamma = .02$	.901	.105	.001	.023	.109	.001	.029
Sequential covariance minimization, $\alpha = -1, N = 1000$							

TAB. 4.1: *Results for various protocols. 20 samples per class in training, and 500 per class in testing. $\alpha = -1$ denotes the sequential covariance minimization protocol, $\alpha = 0$ corresponds to randomized splits, $\alpha = 1$ denotes Boosting.* $\mathbf{S}$ *is the number of randomized splits (maximum number is 110). "Average Margin (c)" corresponds to $E_{\mathcal{L}} E_c[H^N(X,c) - H^N(X,c')]$, where c' is the other class. "Intraclass variance(c)" corresponds to $E_{\mathcal{L}} Var_c[H^N(X,c) - H^N(X,c')]$. "Stdv. wrt. training(c)" corresponds to the square root of $Var_{\mathcal{L}} E_c[H^N(c) - H^N(c')]$. The expectations $E_{\mathcal{L}}$ and $Var_{\mathcal{L}}$ over training samples were calculated using 50 independently produced training samples.*

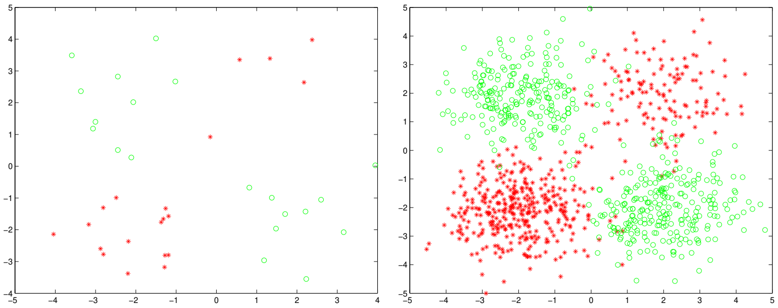


FIG. 4.3: *Left: A training sample of 20 data points per class. The first class is a mixture of two Gaussians centered at* $(-2,-2),(2,2)$ *with weights* $3/4, 1/4$ *respectively. The second class is an equal mixture with centers at* $(-2,2),(2,-2)$*. Right: A test sample of 500 per class.*

image features. Two random splits drawn from this huge pool will typically be very weakly correlated. Also, at a given node of the tree there is a variety of possible splits achieving good performance, so that the **RS** protocol will naturally build very different — and hence not very correlated — trees. This seems the basis of the success of simple randomization in that context, i.e. $\alpha = 0$ and **S** small. This is also discussed in [20]. There it is suggested that in many other contexts one should generate large numbers of more complex features from the initial input features in order to create such a pool of very weakly correlated splits.

By contrast, the 110 features we use in the synthetic example are not weakly correlated. The linear classifier that separates the lower left hand cluster of class 1 from class 2 clearly outperforms all the other base classifiers in terms of (individual) classification error. Therefore, whenever more than one split is sampled and the best is chosen, classifiers that are close to this unique best one tends to be picked and consequently, the limit distribution on the classifier space concentrates very quickly around it, see for example the middle panel of figure 4.5. But as a consequence, the upper right hand cluster of class 1 is ignored and the aggregate classifier performs poorly. This is expressed in the high variance of the aggregate as summarized in table 4.1 (entries with $\alpha = 0$). As mentioned above one could create more complex features, or grow depth 2 trees and then the problem would become trivial. There is still however much to be learned from this very constrained situation. In other contexts it is not always possible to create more complex decision boundaries within the framework of the base classifier. Interestingly, we observe that in this same setup, Boosting, in particular randomized Boosting, produces good classification rates, and the variance of the aggregate is much smaller than without Boosting, see table 4.1, entries with $\alpha = 1$. The corresponding decision boundaries are shown in figure 4.5.

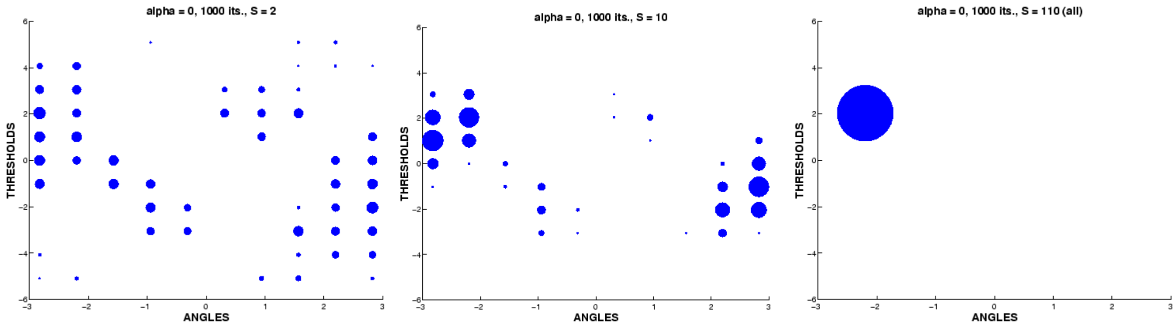
The distributions produced by the different protocols on the set of 110 planes are shown in figure 4.4, all using the same training set. The first row shows the distribution produced by randomized splits with $\mathbf{S} = 2, 10, 110$. Note how the supports differ for each value of $\mathbf{S}$. The second row shows the same for Boosting, i.e. $\alpha = 1$. Note how Boosting puts more weight on the upper right hand region, which apparently takes care of the smaller cluster of the first class.

However, Boosting does not always produce better classifiers; in particular, *deterministic* Boosting can produce over fitting, and is especially sensitive to small training sets (see also [33]). At times, strange decision boundaries are produced, where the process concentrates on a cycle in the space of classifiers as exhibited in the right hand panel in the middle row of figure 4.4. The Boosting process cycles between the three planes shown in the figure; the corresponding decision boundaries are shown in the left hand panel of 4.5. In less extreme cases $\mathbf{Q}$ is concentrated on a small number of planes. Typically better classification rates are obtained with the randomized Boosting protocol: it is interesting to point out that this also corresponds to lower values of the average conditional covariance and of the training variance: both decrease as $\mathbf{S}$ decreases (see table 4.1), and the best rates are observed for $\mathbf{S} = 2$. The same phenomena are exhibited in the synthetic character set (see section 4.6 below for a description) and summarized in table 4.2. Note that in the experiments on the NIST digits data base, with 1000 samples per class, deterministic Boosting did not over-fit but did perform as well as randomized Boosting. In this context the main drawback of deterministic Boosting is the substantially longer training time.

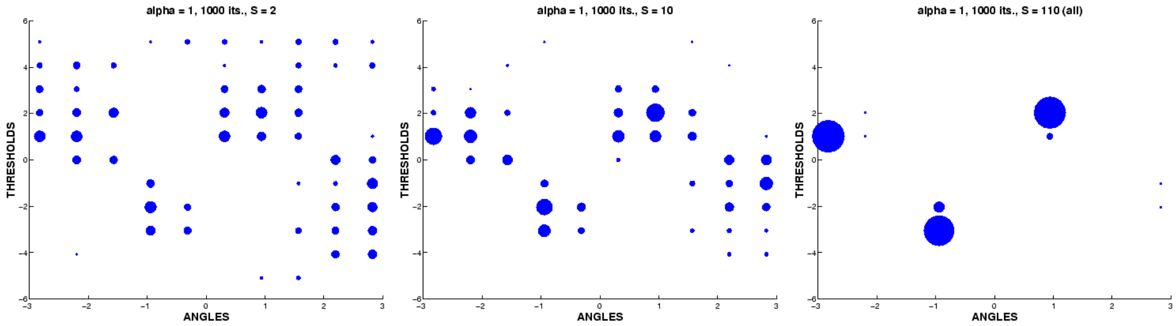
4.5.3 Covariance process and sequential covariance minimization

When the aggregate classifier is obtained by uniform weighting, we can rewrite our quantity of interest, the variance of the aggregate classifier H^n, as the following sum: let $F^n = n\widehat{H}^n = \sum_{l=1}^{n} h_l$, then we easily obtain by recursion that

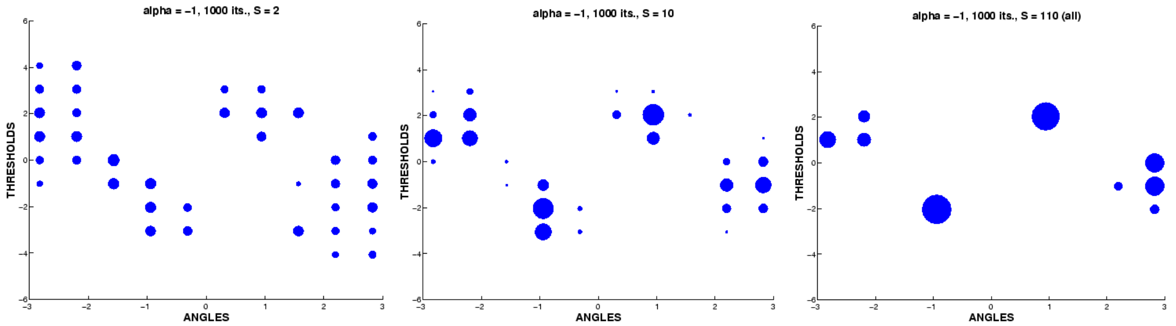
$$Var_c(H^N) = \frac{1}{N^2}\sum_n Var_c(h_n) + \frac{2}{N^2}\sum_n Cov_c(h_n, F^{n-1}). \tag{4.9}$$



Randomized splits, **S**=2,10,110



Randomized Boosting, **S**=2,10,110



Sequential covariance minimization, **S**=2,10,110, $\gamma = .05$

FIG. 4.4: *Distributions on the space of planes given by different protocols. The x-axis is the angle θ and the y-axis is the threshold τ. The area of the circles is proportional to the probability of the corresponding point. Decision regions associated to panels 2,4 and 6 are shown in fig. 4.5.*

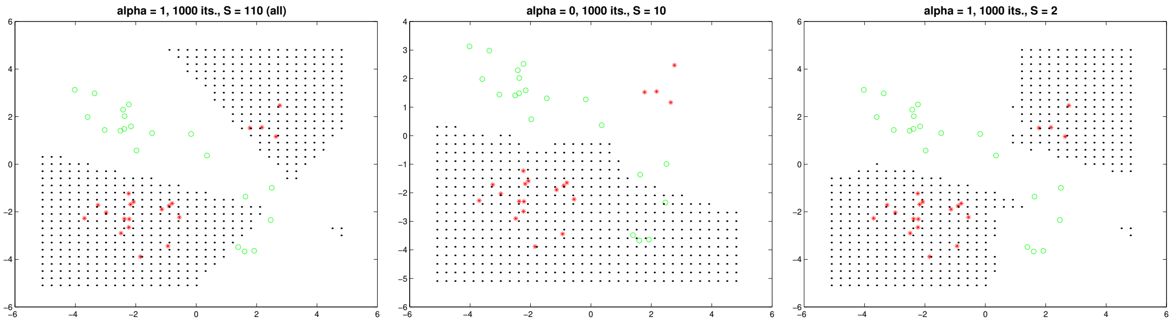


FIG. 4.5: *Left: Decision boundaries obtained after 1000 iterations of the (deterministic) Boosting algorithm. A 3-cycle has been reached. Middle: Decision boundaries obtained after 1000 iterations by the randomized splits algorithm* $\alpha = 0, \mathbf{S} = 10$*: the small cluster of class one is 'ignored'. Right: Decision boundaries obtained by the randomized Boosting algorithm* $\alpha = 1, \mathbf{S} = 2$*. These decision regions correspond to distributions 6,2 and 4 shown on fig. 4.4.*

Therefore if the terms in the second sum are close to zero this variance will be very small.

This remark led us to try to run a protocol where each new classifier is chosen *explicitly* to minimize the covariance with the current aggregate, subject to having reasonable classification rate. We call this protocol *sequential covariance minimization* (**SCM**). Specifically we select $\mathbf{S}$ random planes $f^1, \ldots, f^{\mathbf{S}}$, and let $h^1, \ldots, h^{\mathbf{S}}$ denote the corresponding depth one trees. Evaluate

$$j^* = \mathrm{Argmax}_j \left\{ E[Cov_c(h^j, H^n)] + \gamma e(h^j) \right\}, \tag{4.10}$$

where the expectation is to be understood as an average over the class parameter c, $e(h^j)$ is the error of classifier j, and γ is a free parameter. The next classifier h_{n+1} is taken to be h^{j^*}. It is interesting to notice that the results from this protocol are very similar to those from randomized Boosting as shown in table 4.1 (entries with $\alpha = -1$).

Remember that for the Boosting protocol, each new classifier h_n is built depending on the current (unnormalized) aggregate H^n (through the weighting of the training sample, which in the two class case (see [39]) is proportional to $\exp(-YH^n(X))$ assuming the class and the base classifiers take their values in $\{-1; 1\}$). It therefore appeared interesting to take a look at the covariance process $Cov_c(h_{n+1}, H^n)$ for the Boosting protocol as well (note however that eq. (4.9) is not valid without modifications for Boosting since the weighting of the classifiers is not uniform. The covariance process nevertheless gives some information on the step-by-step behavior of the Boosting algorithm). The experimental results on the artificial example showed that the Boosting algorithm tends to keep the covariance of each new classifier h_n with the current aggregate H^{n-1} low or negative on training, and that on the test data the covariances follow a very similar pattern (thus inducing a low variance of the aggregate). This is in contrast to the divergence of the exponential moments illustrated in section 4.2.

Although these quantities are calculated in terms of the training set, second moments are much more reliably estimated than exponential moments, moreover all we need is for these moments to be small. The reason why Boosting yields low covariances between the current aggregate and the next classifier is not completely clear. We give in section 4.5.4 below a rough comparison between Boosting and **SCM**, namely we point out that this protocol is similar to Boosting with re-weighting of the training data using a linear function of the current error.

To conclude, for each protocol, for example fixed α, there is a "randomization parameter", e.g. $\mathbf{S}$, which controls the dependence between the classifiers. Typically as randomization is increased the average conditional covariance is decreased and a price is payed in terms of the average conditional margin. It is hard to predict where the optimal combination lies, and it is definitely worth exploring this payoff in more detail: an interesting lead is pointed to by the regression results shown in section 4.4.1 which suggest that it is of interest to try minimizing the ratio appearing in the Chebychev inequality.

The next paragraph is devoted to more specific remarks and discussions about the Boosting protocol.

4.5.4 More about Boosting

Boosting as a dynamical process

Let $\mathcal{H}$ denote the set of all base classifiers, and $\mathcal{W}$ be the set of all probability measures on the training set. Then deterministic Boosting is a dynamical process on the product space $\mathcal{H} \times \mathcal{W}$: to a given couple $(h_n, W_n) \in \mathcal{H} \times \mathcal{W}$ it associates a new couple (h_{n+1}, W_{n+1}), as described in item 4 of section 4.3. In the case where we employ randomized Boosting $\mathbf{S} < \mathbf{S_t}$ the process is a Markov chain on $\mathcal{H} \times \mathcal{W}$. When $\mathbf{S} = \mathbf{S_t}$ the process is deterministic.

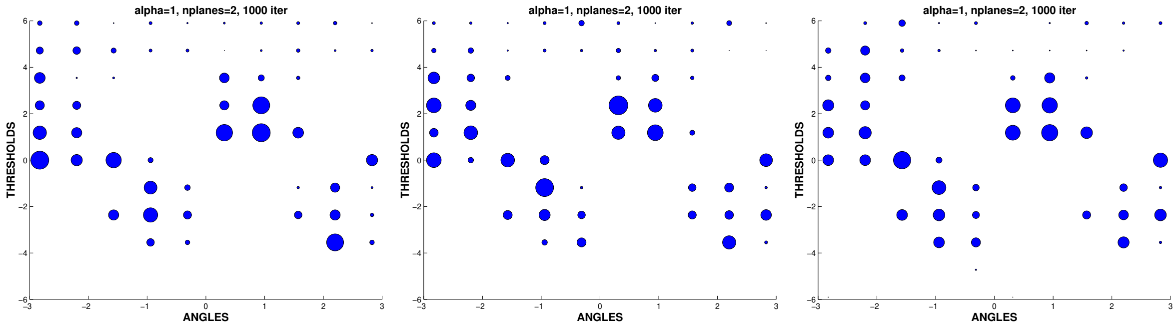
Experimentally on the synthetic classification problem we have observed the following properties which point to the idea of Boosting as an ergodic process.

- The weights β_i in the definition of the aggregate classifier can be taken as uniform, with little change in performance.
- The same classification rates are obtained even when starting with *a random initial* W_1. In other words instead of initializing W_1 as the uniform distribution on $\mathcal{L}$ we draw a weight for each training point uniformly from the interval $(0,1)$ and normalize.
- After a large enough number of iterations, a sample from this process actually mimics some non-trivial invariant distribution which again appears insensitive to initial conditions, see figure 4.6 below.
- It is possible to drop the beginning of the sequence of classifiers and obtain the same results.

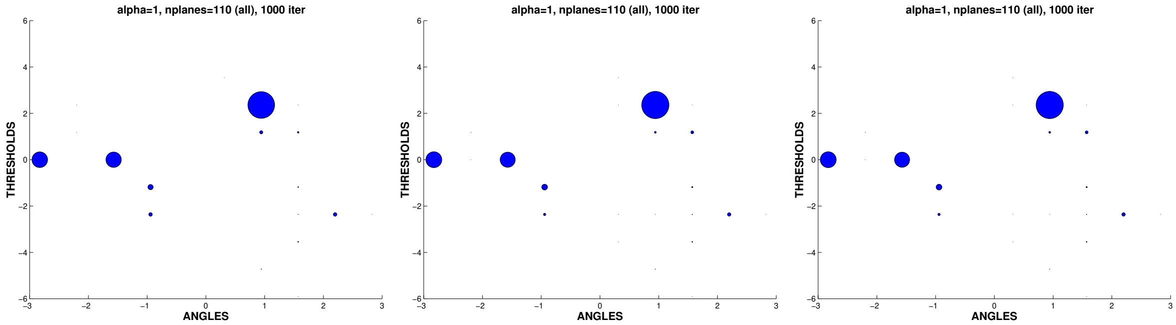
Figure 4.6 is an illustration of the Boosting algorithm mimicking a distribution on classifier space which seems independent from initial conditions. The top three panels show $\mathbf{Q}$, with $\mathbf{S} = 2$, for the same training set on three different runs with 1000 iterations. Both the initial conditions are randomized (each point is assigned a uniformly sampled weight in $(0,1)$) and the random planes in each run are sampled with different seeds. The bottom three panels show the same results for deterministic Boosting, i.e. $\mathbf{S} = 110$, where randomization is only in the initial weights.

It therefore appears appropriate to include Boosting in the general probabilistic setting in which the classifier is an average with respect to some distribution on the space of base classifiers.

However, these experimental results may seem surprising if one has in mind the fact that the Boosting algorithm aims at minimizing a certain cost function. We give here a few comments. As mentioned in section 4.2, in a 2-class problem, assuming Y and the classifiers $h \in \mathcal{H}$ take their values in $\{-1; 1\}$, it has been shown that Boosting is minimizing the cost function $\mathcal{C}(H) = E[\exp(-YH(X))]$, where H is any combination with positive coefficients of base classifiers and the expectation is the empirical expectation on the training set. This fact was already suggested in [54], then in [39] it was shown that Boosting is similar to a Newton algorithm for minimizing this empirical cost iteratively; finally it has been shown [27] that Boosting will always asymptotically drive the cost function to its infimum



Randomized Boosting, $\alpha = 1, \mathbf{S} = 2$



Deterministic Boosting, $\alpha = 1, \mathbf{S} = 110$

FIG. 4.6: *Top: Invariant distributions of Boosting protocol on same training set with three randomized initial conditions. The x-axis is the angle θ and the y-axis is the threshold τ. The area of the circles is proportional to the probability of the corresponding point.*

$\mathcal{C}^*$. This "gradient-descent" interpretation of Boosting seems at odds with our previous claims: the weights β_i are important for this purpose; one would think that dropping the first iterations would completely change the aggregate classifier obtained asymptotically.

To cast some light on these apparent contradictions, let us first notice that, when there exists *some* convex combination of base classifiers that perfectly separates the training data, then the minimum of the cost function $\mathcal{C}(H)$ defined above is $\mathcal{C}^* = 0$: since the aggregate classifier in the cost function is *unnormalized*, one can take any convex combination separating the data and multiply it by an arbitrarily large constant to make the cost function tend to 0. Therefore, in this case, the fact that the cost function goes down to $\mathcal{C}^*$ is not really saying much about the asymptotical behavior of the aggregate classifier apart from the fact that it will separate the training data. Remember one of the main reasons

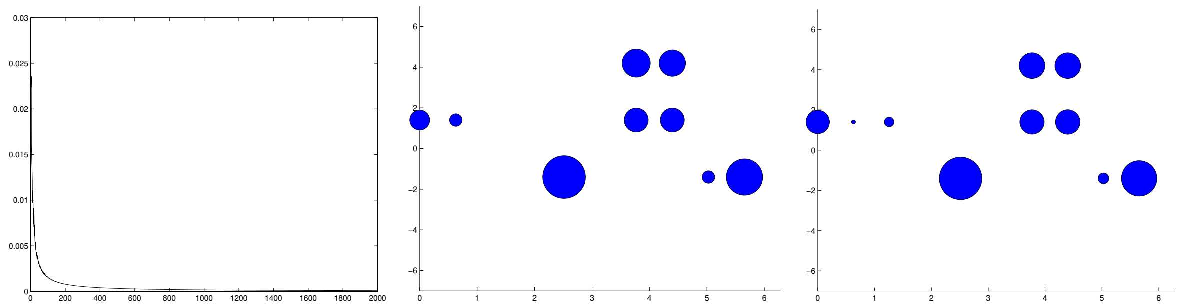


FIG. 4.7: *Deterministic Boosting. Left: evolution of coefficients β_i. Middle: distribution of first 1000 classifiers. Right: distribution of classifiers 1000-2000. The distributions are drawn taking into account the factors β_i, and renormalized to be of sum one in each case. (The sum of coefficients is 47 for 1000 first classifiers, 6.4 for classifiers 1000-2000.)*

for the success of Boosting is precisely that even when the training error has been drawn to zero for the aggregate, the *test* error is known to continue to decrease when one goes on iterating the algorithm (see e.g. [54]). This can't be explained just by the asymptotic property that $\mathcal{C}(H_n) \to 0$. Moreover, one can easily see that in this case there is no minimizer H^* of $\mathcal{C}$: as we pointed it out, $\mathcal{C}(H_n) \to 0$ implies that some of the coefficients of the combination defining H_n must go to infinity. In our experiments, for separable data the coefficients β_i stayed most of the time in the same range of values, from 1 to 4 in terms of scale (thus their sum goes to infinity. This also explains why the final classifier stays roughly the same when uniform weights are used instead). Therefore there is no contradiction with Boosting asymptotically mimicking some distribution on the space of classifiers. More than that, the *dynamical* behavior of Boosting is then more important, as far as generalization error is concerned, than the fact that it draws the cost function to zero.

When $\mathcal{C}^* \neq 0$, the situation is more difficult to analyze. The function $\mathcal{C}$ may or may not admit a minimizer with finite coefficients H^*. Let us report the following interesting experiment made with the artificial example in a case where the training set is not separable (but still drawn from the distribution described in 4.5.1): see fig. 4.7. The coefficients β_i are going to zero, therefore it is true that in the average over the first 2000 iterations of Boosting, the 1000 last ones have in fact only little contribution. Nevertheless, it is *still* true that when we drop the 1000 first iterations, we obtain a very similar result. One can see on the figure that the distributions on classifier space for iterations 0-1000 and 1000-2000 are very much alike. In the case where there is no finite minimizer of the cost function, the sum of the β_i still has to go to infinity (our belief is that this is the case in fig. 4.7): then again the long-term behavior of the algorithm is of importance, even if it does no seem so at first sight.

Our conclusion from these experiments is that there is more to Boosting than minimizing a certain cost function. Once again the *dynamical* behavior of Boosting is essential to be understood, and it seems that little is known about it so far.

Boosting and sequential covariance minimization

As mentioned in section 4.2 for a two-class problem where the two classes are labeled $-1, 1$, and the outputs of the base classifiers are accordingly negative or positive, the Boosting procedure weights a training example X proportional to $e^{-YH(X)}$, where H is the current aggregate classifier:

$$H = H^N = \sum_{i=1}^{N} \beta_i h_i$$

Let h be a new candidate classifier. We assume that the output $h(X)$ of classifier h is either 1 or -1. Consider a slightly more general "Boosting" setup, in which h is chosen in order to minimize (at least approximately) a weighted error function

$$\mathcal{E}(h) = E[W(-HY)(1 - hY)],$$

where the expectation is taken wrt. the empirical distribution, and W is some weight function, supposed to be strictly increasing and convex. (Note that $\mathcal{E}$ is not a normalized error.) For example for original Boosting, $W(x) = \exp(x)$.

Now this can be rewritten as

$$\mathcal{E}(h) = Cov[-W(-HY), hY] + e(h)E[W(-HY)]$$

where $e(h)$ is the empirical error of classifier h. Thus the normalized weighted error function $\alpha(h)$ is given by

$$\alpha(h) = Cov\left[\frac{-W(-HY)}{E[W(-HY)]}, hY\right] + e(h).$$

It is interesting to notice the analogy with the sequential minimization protocol given by eq. (4.10). Boosting tries to minimize at each step a criterion which is a compromise between the error of the next classifier and its covariance with some convex deformation of the current aggregate. Formally, we can retrieve a protocol very close to **SCM** if we take uniform weights of sum one for the aggregate and choose the linear weight function $W_n(x) = x + C$ in which case we obtain

$$\mathcal{E}(h) = Cov[H^N Y, hY] + e(h)\left(C - E[H^N Y]\right) = Cov[H^N Y, hY] + \gamma(H^N)e(h).$$

This is essentially the same as the "sequential covariance minimization " cost function of equation (4.10) except that γ depends on the current normalized aggregate H^N, and the average conditional covariance has been replaced by $Cov(H^N Y, hY)$ (this quantity should nevertheless have a behavior similar to conditional covariance, see related discussion in section 4.4.2).

However, the reason why the choice of an exponential function for W should result in better performance is unclear. Note that other weight functions have already been proposed by other authors [52, 39].

4.6 LaTeX and NIST databases

The LaTeX database is described in detail in [2]. It is of interest for several reasons. The randomness is synthetic so that training and test sets of various sizes can be reproduced at will. On the other hand it deals with a 'real' problem of shape recognition which is very interesting for numerous computer vision applications. This database also tests algorithms in the presence of large numbers of class e.g. 293. The experiments reported below only deal with the first ten classes (digits) since the aim is to illustrate the issues raised in the previous sections. Again in order to study robustness to the size of the training set we use a very small number (8) samples per class, producing shallow trees of depth 2. We test with 100 samples per class. See figure 4.8 for some examples of randomly perturbed digits. Note that the standard Boosting protocol runs into a problem when the weighted classification rate is below 0.5. This is often the case in multi-class problems. In [39] a number of alternative Boosting protocols have been suggested for the multi-class problem which are rather complex. We have simply modified the protocol to update the weights by $1/e$ instead of $(1-e)/e$. This appeared to work fine.

The features we use in this experiment are simpler than those described in [2]. We assume the images are *centered* in a 32×32 grid. We detect edges of 8 orientations and spread each detected edge to a 7×7 neighborhood. In other words if edge type e is detected at location x, y all locations in a 5×5 neighborhood of

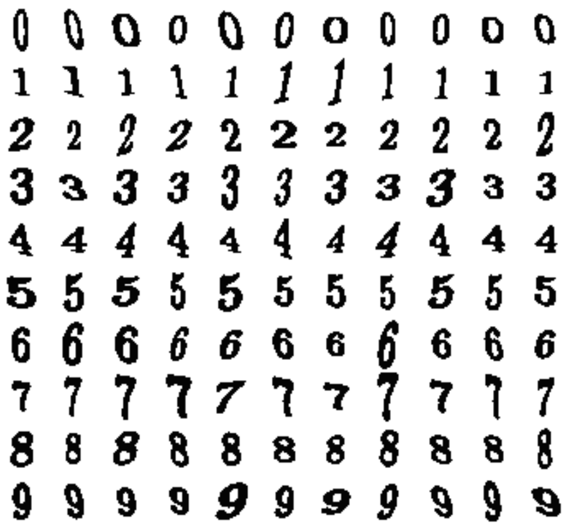


FIG. 4.8: *Examples of the randomly perturbed LATEX digits*

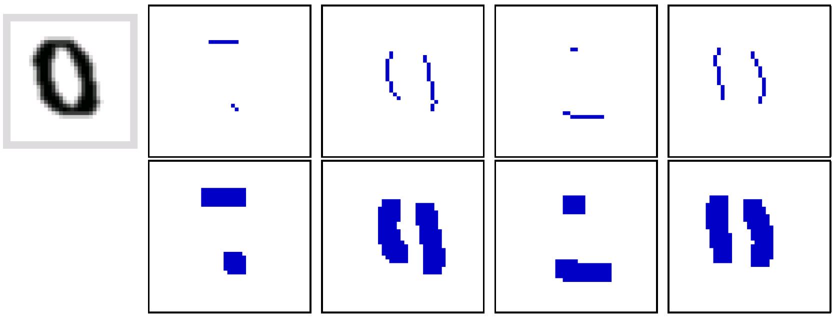


FIG. 4.9: *Top from left to right: Example of a deformed LATEX '0'. Detected edges of angle* 0*, edges of angle* 90*, edges of angle* 180*, edges of angle* 270*. Bottom from left to right: same edges spread in a* 5×5 *neighborhood.*

(x, y) will be turned on for that edge type. This explicitly introduces a large degree of invariance to local deformations.

Table 4.2 summarizes the results for experiments for various protocols. We have added in 'ideal bagging' for the sake of comparison. For each tree a new random training set is produced, and the tree is subsequently grown with no randomization, i.e. all possible splits ($32 \cdot 32 \cdot 8 = 8192$.) The same patterns are observed here as in the synthetic example. Bias and variance are reduced with the number of trees. Randomization has a significant effect on reducing the average conditional covariance, at the cost of reduction in the average conditional margin however often leading to an increase in the classification rate. In this particular

Protocol	Class. Rate	Ave. Marg.	Intraclass Variance	Stdv. wrt. Training
$\alpha = 0, \mathbf{S} = 20, N = 1$	.276	-.087	.055	.028
$\alpha = 0, \mathbf{S} = 20, N = 5$	.469	.020	.013	.012
$\alpha = 0, \mathbf{S} = 20, N = 10$	.664	.043	.008	.009
$\alpha = 0, \mathbf{S} = 20, N = 200$	.963	.097	.004	.005
$\alpha = 0, \mathbf{S} = 200, N = 200$	.944	.132	.007	.007
$\alpha = 1, \mathbf{S} = 20, N = 200$	.960	.059	.001	.0038
$\alpha = 1, \mathbf{S} = 200, N = 200$	.954	.092	.003	.006
$\alpha = 1, \mathbf{S} = 8192(all), N = 200$	.807	.096	.022	.024
Ideal bagging, **S**=8192, N=200	.463	.064	.019	.007

TAB. 4.2: *Results for various protocols on the LATEX database. Here the quantities are not presented for each class but averaged over all ten classes.*

setting randomized Boosting had no advantage relative to simple randomization. Note however the low performance of deterministic Boosting and ideal bagging in comparison. The asymptotic value of the variance which is essentially the ACC is too large relative to the margin. In both cases the ACC is an order of magnitude larger than the ACC for other protocols. We also noted that for some training sets deterministic Boosting produced very low classification rates, this is expressed in the high training standard deviation of the margin. In the case of ideal bagging, one can expect that the top splits will be quite robust to variations in the training set, so that the distribution on trees is rather concentrated, and therefore average conditional covariance is relatively high.

Finally we show some interesting results from the NIST hand written digit database. Using 1000 samples per class to train and 1000 per class to test we experimented with the randomized Boosting protocols. The features were the same as those used for LATEX digits, except that the images were initially downscaled to 16×16 grids, together with a simple slant correction algorithm. Given the smaller size of the images the edges were only spread in a 3×3 region. 50 trees were grown for each protocol. Despite the randomization very little variation is observed between different runs in terms of classification rates. The results are shown in table 4.3. We show only classification rates. The trees are deep in this case - average depth 9. Here Boosting always does slightly better than simple randomized splits, and deterministic Boosting also does very well, although it

Protocol	Class. Rate
$\alpha = 1, \mathbf{S} = 5, N = 50$	.9716
$\alpha = 0, \mathbf{S} = 5, N = 50$	.9600
$\alpha = 1, \mathbf{S} = 20, N = 50$	.9762
$\alpha = 0, \mathbf{S} = 20, N = 50$	.9707
$\alpha = 1, \mathbf{S} = 200, N = 50$	.9786
$\alpha = 0, \mathbf{S} = 200, N = 50$	.9700
$\alpha = 1, \mathbf{S} = 2048\ (all), N = 50$	.9784

TAB. 4.3: *Classification rates for various protocols on the NIST handwritten digit database. 10000 training samples, 10000 test samples.*

takes massive amounts of time to generate each tree (approximately 15 minutes on an Ultra Sparc 2.)

4.7 Discussion

We believe the main reason behind the success of various protocols of MRCL lies in the reduction of the average conditional covariance with respect to the limit distribution **Q** produced on the space of classifiers. Such a limit distribution appears to exist even in the context of deterministic Boosting. The low average conditional covariance reduces the variance of the margin around its mean and is also related to lower estimation errors, namely variability with respect to the training set.

The role of randomization is to produce a large variety of weakly dependent classifiers, using randomly selected subsets of features. We have discussed randomization more particularly in the context of decision trees, however randomization can also be applied to other classification methods. It is always possible to produce the base classifier using a random subset of features. In the context of feed-forward neural nets the connections can be randomly selected. In this sense randomization tends to produce 'different' experts and Boosting produces experts which 'disagree'. Randomized Boosting seems to perform better and be more stable than any of the two extremes: deterministic Boosting or only randomization without Boosting.

Chapitre 5

Generalization error bounds for aggregate classifiers

5.1 Introduction

The use of multiple classifiers has raised much interest in the statistical learning community in the past few years. In particular, the Boosting algorithm has recently received much attention because of its good reported practical results. Several authors have reported that, while a perfect aggregate classifier (that is, having zero training error) can be reached very quickly by the algorithm, going on iterating the algorithm usually still results in a noticeable reduction of test error. Since this property can obviously not be explained by just using training error statistics, people have tried to introduce more accurate statistics on training to explain these good results of Boosting. A particularly interesting lead has been opened by Freund, Shapire, Bartlett and Lee [54] using the notion of empirical margin, which we introduce below.

In this chapter for simplicity we will restrict our attention to 2-class classification problems, the class variable Y will take its values in $\{-1;1\}$. We suppose we are given a set of classifiers $\mathcal{H}$, supposed, according to the cases, finite or of finite VC dimension (we will often think of the finite case as a first step towards the finite VC dimension d case, with $M = |\mathcal{H}|$ to be replaced by N^d). A classifier $h \in \mathcal{H}$ will take its values in $\{-1;1\}$ (in some cases we will allow that $h \in \mathcal{H}$ takes its values in the full range $[-1;1]$. This will be precised accordingly).

We focus our interest on aggregate classifiers, in other words convex combinations of classifiers. Such a convex combination can be identified with a corresponding probability distribution $\alpha \in \mathcal{M}_1(\mathcal{H})$ (we won't elaborate on measurability issues, assuming that the space $\mathcal{H}$ is nice enough). In the sequel, we will denote by H_α the aggregate classifier obtained through the convex combination α. The *margin* M_α associated to an example $Z = (X, Y)$ and a probability distribution α is defined as

$$M_\alpha(Z) = H_\alpha(X)Y.$$

Thus it is equivalent to say that an example is misclassified or that its margin is negative. However, margins give more information about the behavior of the classifier with respect to the data than just training error. The goal of this chapter is to propose several possible methods to infer bounds on the generalization error from statistics coming from the *empirical* margin distribution. We will propose three different point of views.

In section 5.2, we present a bound on generalization error based on the empirical cumulative distribution function of the margins. It can be seen as a interpolation result between the bounds given in the original work of Shapire et al. [54] and a bound proven by Breiman [18] based on the minimum empirical margin. The result we present uses the same kind of proof technique and allows to retrieve, up to multiplicative factor, both of these previous bounds as special cases.

In section 5.3, we give bounds on generalization error based on the ideas exposed in the previous chapter, namely through a Chebychev inequality. We derive uniform controls of mean and variance of the margin function for any convex combination of base classifiers, based on their empirical counterparts. These bounds mainly involve very standard tools for control of empirical processes (Hoeffding's and Bernstein's inequalities). One of the point we will discuss is whether low empirical variance indeed leads to a more accurate estimation.

Finally, in section 5.4 we will concentrate on the minimax margin, also introduced in [18] and suggested in [54]. Namely, in this setting one tries to find the convex combination of base classifiers for which the minimum empirical margin on the training set is the highest possible. We recall the relation of this problem with game theory and support vector machines, as already mentioned in the articles cited above. We show how the well-known Blackwell algorithm coming from game theory can be used to find an approximate solution to this problem. Then we propose a notion of "support vectors" in this setting allowing to derive a bound on generalization error based on a sparsity argument.

5.2 Bounds using the empirical margin distribution

5.2.1 Relation to previous work

In [54], Shapire et al. proved the following theorem:

Theorem 7 (Freund,Bartlett,Shapire,Lee). *Let Z_1^N be an i.i.d. training sample drawn according to some distribution P on $\mathcal{X} \times \mathcal{Y}$. Assume $|\mathcal{H}| = L$, and that a classifier $h \in \mathcal{H}$ takes its values in $\{-1;1\}$. Let $\delta > 0$, then except on a set of samples of P_N-probability less than δ, for all $\theta > 0$ and all $\alpha \in \mathcal{M}_1(\mathcal{H})$ the following inequality holds:*

$$P[M_\alpha(Z) \leq 0] \leq \widehat{P}_N[M_\alpha(Z) \leq \theta] + \frac{C}{\sqrt{N}} \left(\frac{\log L \log N}{\theta^2} + \log(1/\delta) \right)^{1/2}, \qquad (5.1)$$

where $\widehat{P}_N$ is the empirical probability based on the sample Z_1^N, and C is an universal constant.

This theorem thus permits to bound the generalization error by a function depending on the empirical margin distribution on the sample. The same authors proved that under the hypothesis that $\mathcal{H}$ is of finite VC dimension $d \geq 1$, the same inequality holds when one replaces $\log L$ by $d \log(N/d)$ whenever $N \geq d$. Furthermore, the same paper gives empirical and theoretical evidence that the Boosting algorithm tends to make the empirical margins as high as possible. Intuitively, one can say that at each iteration, the Boosting algorithm will mainly concentrate on those examples that have the lower margins with respect to the current aggregate classifier.

Based on this results, one can think of specifically trying to maximize over convex combinations of classifiers the lowest observed margin on the training set. This is what we call the "minimax margin" principle, discussed in more detail in section 5.4. Our point of interest here is the following bound showed by Breiman [18] based on the minimum empirical margin:

Theorem 8 (Breiman). *Let Z_1^N be an i.i.d. training sample drawn according to some distribution P on $\mathcal{X} \times \mathcal{Y}$. Assume $|\mathcal{H}| = L$, and that a classifier $h \in \mathcal{H}$ takes its values in $[-1;1]$. Define $\underline{M_\alpha}$ as*

$$\underline{M_\alpha} \doteq \inf_{1 \leq i \leq N} M_\alpha(Z_i)$$

Let $\delta > 0$, then except on a set of samples of P_N-probability less than δ, for all $\Delta > 0$, $\alpha \in \mathcal{M}_1(\mathcal{H})$ the following inequality holds:

$$P[M_\alpha(Z) \leq \underline{M_\alpha} - \Delta] \leq \frac{C}{N}\left(\frac{\log L \log N}{\Delta^2} + \log(1/\delta)\right), \tag{5.2}$$

where C is an universal constant.

We thus get a bound on generalization error by taking $\Delta = \underline{M_\alpha}$ in the inequality above. The interesting point is to compare with bound (5.1) when $\theta = \underline{M_\alpha}$: the bound obtained by Breiman is of the order of magnitude of the square of the Shapire et al. bound, and therefore much more accurate. On the other hand, inequality (5.1) is more general since we can choose any value for the margin level θ and not only the minimum margin. It is likely that the minimum of the right hand side of (5.1) is obtained for some $\theta > \underline{M_\alpha}$ and it is not clear in this case whether (5.2) gives a better bound or not.

This situation is to be compared with the well-known classification setting in which one applies the ERM principle on a classifier set of finite VC dimension d: in general, the rate of convergence for the classical VC type bounds of the generalization error are of the order of $\sqrt{d \log(N)/N}$. However, if one adds the hypothesis that there exists a classifier having zero generalization error, then the rate of convergence can be shown to be of the order of $d \log(N)/N$. Recently, interpolation bounds between these two situations have been proven [47], that explicitly show how the transition between the two regimes takes place. The principle of the proof is to use classical VC bounds for relative, instead of absolute, deviations of empirical processes from their means.

In our case, inequality (5.2) in which one focuses on the minimum margin is somehow the analog of the "perfect classifier" case above. The main result of this section is an interpolation bound between inequalities (5.1) and (5.2) which renders the regime transition explicit. The proof of this inequality is mainly based on the same ideas used by Shapire et al. and Breiman to prove theorems 7 and 8, and on the use of VC bounds for relative deviations.

5.2.2 An interpolation bound

We now give the main result of this section:

Theorem 9. *Let $\mathcal{H} = \{h_1, \dots, h_L\}$ be a finite set of classifiers taking their values in $[-1;1]$. Let $Z_1, \dots, Z_N$, where $Z_i = (X_i, Y_i) \in \mathcal{X} \times \{-1;1\}$ be an i.i.d. training sample drawn according to some distribution P.*

For $\alpha \in S_{L-1}$ belonging to the $(L-1)$-dimensional simplex, denote $H_\alpha = \sum_{i=1}^M \alpha_i h_i$ the composite classifier with weights given by α, and for $Z = (X, Y)$, let $M_\alpha(Z) = H_\alpha(X)Y$ be the margin function.

Let F_α denote the cumulative distribution function of $M_\alpha(Z)$ and $\widehat{F}_\alpha^N$ denote the empirical cdf of the same variable, based on sample Z_1^N.

Define finally the two following quantities:

$$R_{L,N} = \frac{48\log(2L)}{N\Delta^2} + 4\sqrt{\frac{2\log(2L)}{N\Delta^2}}\sqrt{\widehat{F}_\alpha^N(\theta)}, \tag{5.3}$$

$$r_N = \frac{4}{N}\log\left(\frac{8N+4}{\delta}\right) \tag{5.4}$$

Then, except on a set Ω_δ of training samples of probability less than δ, for any $\theta \in [-1;1]$ and $\Delta \in [0;2]$, for all $\alpha \in \mathcal{M}_1(\mathcal{H})$ the following inequality holds:

$$F_\alpha(\theta - \Delta) \leq \widehat{F}_\alpha^N(\theta) + R_{L,N}\left(\log(R_{L,N}^{-1}) + 3\right) + \sqrt{\widehat{F}_\alpha^N(\theta)}\sqrt{r_N} + \frac{3}{2}r_N. \tag{5.5}$$

Before proceeding to the proof, let us give the following corollary to give a more straightforward idea of the order of convergence obtained:

Corollary 3. *Under the hypotheses of the previous theorem, except on a set Ω_δ of training samples of probability less than δ, for any $\theta \in [-1;1]$ and $\Delta \in [0;2]$, for all $\alpha \in \mathcal{M}_1(\mathcal{H})$ the following inequality holds:*

$$\begin{aligned} F_\alpha(\theta - \Delta) \leq \widehat{F}_\alpha^N(\theta) + C_1\sqrt{\widehat{F}_\alpha^N(\theta)}\left(\frac{\log L \log N}{\Delta^2 N} + \log(1/\delta)\right)^{1/2} \\ + C_2\left(\frac{\log L \log N}{\Delta^2 N} + \log(1/\delta)\right), \end{aligned} \tag{5.6}$$

where C_1, C_2 are universal constants.

Proof. Fix $\alpha \in S_{L-1}$ and let K be a positive integer.

Consider α as a probability distribution on $\{1, \dots, L\}$ and let $i_1, \dots, i_K$ be K independent draws from this distribution. Let $(n_1, \dots, n_L)$ be the associated vector of counts, $n_k = \sum_{j=1}^K \mathbf{1}\{i_j = k\}$. We define this way the random variable

$$\widetilde{\alpha} = \left(\frac{n_1}{K}, \dots, \frac{n_L}{K}\right)$$

taking its values in S_{L-1}^K, the simplex discretized through a grid of scale $1/K$.

In the sequel, we will denote by P the probability distribution of a new example Z, by $P_{\widetilde{\alpha}}$ the probability distribution of the random variable $\widetilde{\alpha}$ defined above, and by $P_N = P^{\otimes N}$ the probability distribution of the training sample $Z_1, \ldots, Z_N$. $E, E_{\widetilde{\alpha}}, E_N$ will denote the corresponding expectation operators.

We start with the following inequality:

$$\begin{aligned} F_\alpha(\theta - \Delta) &= P[M_\alpha(Z) \le \theta - \Delta] && (5.7) \\ &\le PP_{\widetilde{\alpha}}\left[M_\alpha(Z) \le \theta - \Delta; M_{\widetilde{\alpha}}(Z) - M_\alpha(Z) < \Delta/2\right] \\ &\quad + PP_{\widetilde{\alpha}}\left[M_\alpha(Z) \le \theta - \Delta; M_{\widetilde{\alpha}}(Z) - M_\alpha(Z) \ge \Delta/2\right] \\ &\le EP_{\widetilde{\alpha}}[M_{\widetilde{\alpha}}(Z) < \theta - \Delta/2] + E_{\widetilde{\alpha}}P[M_{\widetilde{\alpha}}(Z) - M_\alpha(Z) \ge \Delta/2]. && (5.8) \end{aligned}$$

We first deal with the second term of (5.8). For each fixed Z, $M_{\widetilde{\alpha}}(Z) = \frac{1}{K}\sum_{j=1}^K h_{i_j}(X)Y$, is the average of K i.i.d. variables of mean $M_\alpha(Z)$ and bounded by 1 in absolute value. Therefore by Hoeffding's inequality:

$$P_{\widetilde{\alpha}}[M_{\widetilde{\alpha}}(Z) - M_\alpha(Z) \ge \Delta/2] \le \exp\left(-\frac{K\Delta^2}{8}\right). \qquad (5.9)$$

The first term in majoration (5.8) is equal to $E_{\widetilde{\alpha}}[F_{\widetilde{\alpha}}(\theta - \Delta/2)]$. We will derive a bound on $F_{\widetilde{\alpha}}(\theta - \Delta/2)$ holding uniformly over $\widetilde{\alpha} \in S_{L-1}^K$ with P_N-probability greater than δ and then perform integration over $\widetilde{\alpha}$.

For the moment, let $\widetilde{\alpha}$ be fixed. We will apply Vapnik and Chervonenkis' relative deviation bound:

Theorem 10 (Vapnik, Chervonenkis). *Let μ be a probability distribution over a set $\mathcal{Z}$, and μ_N be the empirical probability distribution on $\mathcal{Z}$ obtained through an i.i.d. draw from μ of size N. Let $\mathcal{A}$ be a collection of subsets of $\mathcal{Z}$ and $S_{\mathcal{A}}(n)$ be the shatter coefficient of $\mathcal{A}$. Then the following inequality holds:*

$$P\left[\sup_{A\in\mathcal{A}} \frac{\mu(A) - \mu_N(A)}{\sqrt{\mu(A)}} > \varepsilon\right] \le 4S_{\mathcal{A}}(2N)\exp\left(-\frac{N\varepsilon^2}{4}\right) \qquad (5.10)$$

In our case, consider $\mathcal{Z} = \mathbb{R}$, μ the probability distribution of the variable $M_{\widetilde{\alpha}}$, and $\mathcal{A} = \{(-\infty, t] | t \in \mathbb{R}\}$. $\mathcal{A}$ is of VC-dimension 1 so that by the Vapnik-Chervonenkis-Sauer lemma, $S_{\mathcal{A}}(N) \le N + 1$. Applying inequality (5.10) we therefore get that for a fixed $\widetilde{\alpha}$,

$$P_N\left[\sup_{t\in\mathbb{R}} \frac{F_{\widetilde{\alpha}}(t) - \widehat{F}_{\widetilde{\alpha}}^N(t)}{\sqrt{F_{\widetilde{\alpha}}(t)}} > \varepsilon\right] \le 4(2N+1)\exp\left(-\frac{N\varepsilon^2}{4}\right),$$

where we recall that $\widehat{F}^N_{\widetilde{\alpha}}$ is the empirical cdf of the variable $M_{\widetilde{\alpha}}$.

Since $|S^K_{L-1}| < L^K$, we get by the union bound

$$P_N\left[\sup_{\widetilde{\alpha}\in S^K_{L-1}}\sup_{t\in\mathbb{R}}\frac{F_{\widetilde{\alpha}}(t)-\widehat{F}^N_{\widetilde{\alpha}}(t)}{\sqrt{F_{\widetilde{\alpha}}(t)}}>\varepsilon\right]\le 4(2N+1)L^K\exp\left(-\frac{N\varepsilon^2}{4}\right). \tag{5.11}$$

Finally, let us choose ε_K such that the right-hand side of the last inequality is equal to $2^{-K}\delta$. Solving this equation yields

$$\varepsilon_K^2=\frac{4}{N}\left(\log\left(\frac{8N+4}{\delta}\right)+K\log(2L)\right); \tag{5.12}$$

with this choice, again using the union bound, we therefore get that, except for a set Ω_δ of training samples of P_N-probability less than δ,

$$\forall K>0\quad \forall\widetilde{\alpha}\in S^K_{L-1}\quad \forall t\in\mathbb{R}\quad F_{\widetilde{\alpha}}(t)-\widehat{F}^N_{\widetilde{\alpha}}(t)\le\varepsilon_K\sqrt{F_{\widetilde{\alpha}}(t)}.$$

From now on suppose that $Z_1^N\notin\Omega_\delta$. The last relation then implies, by solving the quadratic inequality, that

$$\forall K>0\quad \forall\widetilde{\alpha}\in S^K_{L-1}\quad \forall t\in\mathbb{R}\quad F_{\widetilde{\alpha}}(t)\le\widehat{F}^N_{\widetilde{\alpha}}(t)+\varepsilon_K\sqrt{\widehat{F}^N_{\widetilde{\alpha}}(t)}+\varepsilon_K^2. \tag{5.13}$$

Putting $t=\theta-\Delta/2$ and taking expectation with respect to $\widetilde{\alpha}$, we thus get

$$\begin{aligned}E_{\widetilde{\alpha}}[F_{\widetilde{\alpha}}(\theta-\Delta/2)]&\le E_{\widetilde{\alpha}}[\widehat{F}^N_{\widetilde{\alpha}}(\theta-\Delta/2)]+\varepsilon_K E_{\widetilde{\alpha}}[\sqrt{\widehat{F}^N_{\widetilde{\alpha}}(\theta-\Delta/2)}]+\varepsilon_K^2\\&\le E_{\widetilde{\alpha}}[\widehat{F}^N_{\widetilde{\alpha}}(\theta-\Delta/2)]+\varepsilon_K\sqrt{E_{\widetilde{\alpha}}[\widehat{F}^N_{\widetilde{\alpha}}(\theta-\Delta/2)]}+\varepsilon_K^2,\end{aligned} \tag{5.14}$$

where we have used Jensen's inequality.

Now let us bound the expectation terms with respect to $\widetilde{\alpha}$:

$$\begin{aligned}E_{\widetilde{\alpha}}[\widehat{F}^N_{\widetilde{\alpha}}(\theta-\Delta/2)]&=E_{\widetilde{\alpha}}\left[\frac{1}{N}\sum_{i=1}^N\mathbf{1}\{F_{\widetilde{\alpha}}(Z_i)\le\theta-\Delta/2\}\right]\\&=\frac{1}{N}\sum_{i=1}^N\Big(P_{\widetilde{\alpha}}[F_{\widetilde{\alpha}}(Z_i)\le\theta-\Delta/2;F_\alpha(Z_i)-F_{\widetilde{\alpha}}(Z_i)\le\Delta/2]\\&\qquad+P_{\widetilde{\alpha}}[F_{\widetilde{\alpha}}(Z_i)\le\theta-\Delta/2;F_\alpha(Z_i)-F_{\widetilde{\alpha}}(Z_i)>\Delta/2]\Big)\\&\le\frac{1}{N}\sum_{i=1}^N\Big(\mathbf{1}\{F_\alpha(Z_i)\le\theta\}+P_{\widetilde{\alpha}}[F_\alpha(Z_i)-F_{\widetilde{\alpha}}(Z_i)>\Delta/2]\Big)\\&\le\widehat{F}^N_\alpha(\theta)+\exp\left(-\frac{K\Delta^2}{8}\right),\end{aligned} \tag{5.15}$$

where we have used again Hoeffding's inequality for every term of the sum. Finally (5.14),(5.15) lead to,

$$E_{\widetilde{\alpha}}[F_{\widetilde{\alpha}}(\theta-\Delta/2)] \leq \widehat{F}_\alpha^N(\theta)+\varepsilon_K\sqrt{\widehat{F}_\alpha^N(\theta)}+\varepsilon_K^2+\varepsilon_K\exp\left(-\frac{K\Delta^2}{16}\right)+\exp\left(-\frac{K\Delta^2}{8}\right)$$
$$\leq \widehat{F}_\alpha^N(\theta)+\varepsilon_K\sqrt{\widehat{F}_\alpha^N(\theta)}+\frac{3}{2}\varepsilon_K^2+\frac{3}{2}\exp\left(-\frac{K\Delta^2}{8}\right), \tag{5.16}$$

where we have used the inequalities $\sqrt{A+B}\leq\sqrt{A}+\sqrt{B}$ and $AB\leq\frac{1}{2}(A^2+B^2)$. Now putting together (5.8),(5.9) and (5.16) we get that for $Z_1^N\notin\Omega_\delta$, and ε_K given by (5.12),

$$\forall K>0 \quad F_\alpha(\theta-\Delta)\leq\widehat{F}_\alpha^N(\theta)+\varepsilon_K\sqrt{\widehat{F}_\alpha^N(\theta)}+\frac{3}{2}\varepsilon_K^2+\frac{5}{2}\exp\left(-\frac{K\Delta^2}{8}\right). \tag{5.17}$$

Now define

$$R_{L,N}=\frac{48\log(2L)}{N\Delta^2}+4\sqrt{\frac{2\log(2L)}{N\Delta^2}}\sqrt{\widehat{F}_\alpha^N(\theta)}$$

and choose

$$K=\left\lfloor\frac{8}{\Delta^2}\log(CR_{L,N}^{-1})\right\rfloor \text{ with } C=\frac{5\sqrt{e}}{2},$$

so that

$$\frac{5}{2}\exp\left(-\frac{K\Delta^2}{8}\right)\leq C^{-1}\frac{5}{2}\exp\left(\frac{\Delta^2}{8}\right)R_{N,L}\leq R_{N,L},$$

since $\Delta\in[0;2]$. For more compact notation define

$$r_N=\frac{4}{N}\log\left(\frac{8N+4}{\delta}\right)$$

so that (5.12) becomes

$$\varepsilon_K^2=\frac{4K}{N}\log(2L)+r_N\leq\frac{32}{N\Delta^2}\log(2L)\log(CR_{L,N}^{-1})+r_N, \tag{5.18}$$

and that we also have

$$\varepsilon_K\leq 4\sqrt{2}\sqrt{\frac{\log(2L)}{N\Delta^2}}\log(CR_{L,N}^{-1})+\sqrt{r_N}, \tag{5.19}$$

where we have assumed that $R_{L,N} \leq 1$ to remove the square root on the logarithmic term (if this is not the case, the result is trivial). Using these various tidbits back into (5.17) yields

$$\begin{aligned} F_\alpha(\theta-\Delta) \leq \widehat{F}_\alpha^N(\theta) &+ 4\sqrt{2}\sqrt{\frac{\log(2L)}{N\Delta^2}}\sqrt{\widehat{F}_\alpha^N(\theta)}\log(CR_{L,N}^{-1}) + \sqrt{r_N}\sqrt{\widehat{F}_\alpha^N(\theta)} \\ &+ \frac{48}{N\Delta^2}\log(2L)\log(CR_{L,N}^{-1}) + \frac{3}{2}r_N + R_{L,N} \\ &\leq \widehat{F}_\alpha^N(\theta) + R_{L,N}\left(\log(R_{L,N}^{-1})+3\right) + \sqrt{\widehat{F}_\alpha^N(\theta)}\sqrt{r_N} + \frac{3}{2}r_N. \end{aligned}$$

□

Proof of Corollary 3. The corollary is not *stricto sensu* a consequence of the inequality given in the theorem. Instead or using inequality (5.19), use directly the square root of (5.18) into (5.17) to obtain

$$\begin{aligned} F_\alpha(\theta-\Delta) \leq \widehat{F}_\alpha^N(\theta) &+ A\sqrt{\widehat{F}_\alpha^N(\theta)}\left(\frac{\log(2L)}{N\Delta^2}\log(CR_{L,N}^{-1}) + r_N\right)^{1/2} \\ &+ B\left(\frac{\log(2L)\log(CR_{L,N}^{-1})}{N\Delta^2} + r_N + R_{L,N}\right), \end{aligned}$$

where A, B are constants. Then using the fact that $R_{L,N}^{-1} \leq N$ leads to the announced inequality. □

Using exactly the same argument as in [54] we can extend theorem 9 in the case where $\mathcal{H}$ is of finite dimension d, but with the additional hypothesis that the classifiers take their values in $\{-1;1\}$:

Theorem 11. *Consider the same hypotheses as in theorem 9, except $\mathcal{H}$ is only supposed to be of finite VC dimension d, and classifiers $h \in \mathcal{H}$ take their values in $\{-1;1\}$. Then except on a set Ω_δ of training samples of probability less than δ, for any $\theta \in [-1;1]$ and $\Delta \in [0;2]$, for all $\alpha \in \mathcal{M}_1(\mathcal{H})$ the following inequality holds:*

$$\begin{aligned} F_\alpha(\theta-\Delta) \leq \widehat{F}_\alpha^N(\theta) &+ C_1'\sqrt{\widehat{F}_\alpha^N(\theta)}\left(\frac{d\log^2 N + \log(1/\delta)}{\Delta^2 N}\right)^{1/2} \\ &+ C_2'\left(\frac{d\log^2 N + \log(1/\delta)}{\Delta^2 N}\right), \end{aligned}$$

where C_1', C_2' are universal constants.

Proof. We proceed exactly as in the proof of theorem 9, except that we want to directly apply theorem 10 to the variable $Z = (X, Y) \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and the set

$$\mathcal{A}_K = \left\{ \{Z | F_{\widetilde{\alpha}}(Z) \leq t\}, t \in \mathbb{R}, \widetilde{\alpha} \in S_{\mathcal{H}}^K \right\}, \tag{5.20}$$

where $S_{\mathcal{H}}^K$ is the set of unweighted averages of K elements taken from $\mathcal{H}$. We now have to bound $\mathcal{S}_{\mathcal{A}_K}(N)$. By Sauer's lemma and the hypothesis on the VC dimension of $\mathcal{H}$,

$$\text{Card}\left\{(h(Z_1), \ldots, h(Z_N)), h \in \mathcal{H}\right\} \leq (N+1)^d;$$

then, since each $\widetilde{\alpha} \in S_{\mathcal{H}}^K$ is composed of K elements from $\mathcal{H}$, we have

$$\text{Card}\left\{(h_{\widetilde{\alpha}}(Z_1), \ldots, h_{\widetilde{\alpha}}(Z_N)), h \in \mathcal{H}, \widetilde{\alpha} \in S_{\mathcal{H}}^K\right\} \leq (N+1)^{Kd}.$$

Finally, since $F_{\widetilde{\alpha}}(Z)$ can take only values of the form $(2k/K) - 1$, $k = 0, \ldots, K$, we can restrict the range of t in the above definition of $\mathcal{A}$ to these $(K+1)$ discretized values, hence

$$\mathcal{S}_{\mathcal{A}_K}(2N) \leq (K+1)(2N+1)^{Kd} \leq (2N+1)^{Kd+1},$$

if we ensure that $K \leq 2N$, which is not a constraint since K is to be chosen of the order of $\log(N)$. Hence (5.12) is replaced by

$$\varepsilon_K^2 = \frac{4}{N}\left(\log\left(\frac{4}{\delta}\right) + (Kd+1)\log(2N+1)\right);$$

Now defining

$$R_{d,N} = C\frac{d\log(N)}{N\Delta^2} + C'\sqrt{\frac{d\log(N)}{N\Delta^2}}\sqrt{\widehat{F}_\alpha^N(\theta)},$$

with suitable constants C and C', and applying the same steps as in the previous proof we get the announced bound. □

5.2.3 Discussion

It is easy to see that from inequality (5.6) we can retrieve both inequalities (5.1) and (5.2), up to a constant factor. Theorem 9 therefore indeed shows how the transition between the two different rates or convergence takes place. Moreover, one can hope that inequality (5.5) may give a better bound on generalization error in some cases, especially in the transition region. In particular, one can think of the case where there is a small proportion of misclassified training points, or more generally a small proportion of training points having a very low margin. Then we can both take advantage of increasing the threshold margin θ while $\widehat{F}_\alpha^N(\theta)$ is still low.

About the restriction that classifiers take their values in $\{-1;1\}$ in theorem 11, note that it plays a crucial role in the proof. More precisely, if we allow values in $[-1;1]$ for the base classifiers, then the set $\mathcal{A}_K$ defined by eq. (5.20) may shatter any sample. Consider the case where $X \in \mathbb{R}$, $\mathcal{H}$ is made of increasing or decreasing functions, and therefore of VC dimension 2. Then the corresponding $\mathcal{A}_2$ can shatter any sample since a sum of an increasing and a decreasing function can take any specified values on a finite set of points (with the restriction that these functions take their values in $[-1;1]$, one can still show easily that there exist samples totally shattered by $\mathcal{A}_2$). This just shows that the argument we give in the proof would fail to overcome this restriction without additional refining. However, this hypothesis is not extremely restricting as far as practical applications are concerned.

5.3 Bounds based on the empirical mean and variance of the margin

In this section, we investigate how one can derive generalization error bounds based on the estimation of the margin mean and variance like was suggested in the previous chapter. Since this control is based on Chebychev's inequality, it is clear that from an asymptotic point of view, when the size of the training set tends to infinity, Chebychev-based bounds won't be able to compete with the type of bounds shown in the previous section, since an inequality such as (5.6) makes use of information coming from the full empirical distribution on the margin, whereas the Chebychev-based bound only makes use of first and second moment statistics. However, our hope is that for small training sets, it may be more favorable in practice to only take into account these simpler statistics because their estimation is more robust.

The bound obtained by Chebychev's inequality is the following: for any $\alpha \in \mathcal{M}_1(\mathcal{H})$,

$$P[\mathrm{Error}(H_\alpha)] = P[M_\alpha(Z) \leq 0] \leq \frac{Var[M_\alpha(Z)]}{E[M_\alpha(Z)]^2} \tag{5.21}$$

(since we deal with a two-class case, this bound based on the margin variable is more convenient to handle than the bound given in the previous chapter and based on conditional variance given class; however it is very similar in spirit).

The goal of this section is to give exact inequalities for the control of mean and variance of margin distributions based on their empirical counterparts: in other words, confidence intervals. This will allow to get from (5.21) a bound on generalization error based purely on empirical quantities.

In the previous chapter, we suggested methods like the "sequential covariance minimization" for building multiple classifiers based on empirical mean and variance of the corresponding margin distributions, the variance of the aggregate being interpreted as the average covariance between two classifiers. This is why we need to control estimation accuracy uniformly over all convex combinations of base classifiers. This way, the control obtained from (5.21) will be valid for any α chosen based on the actual data used for estimation.

The results presented in this section will be derived in the case of finite $\mathcal{H}$. We only make use of very classical tools and the inequalities obtained should be considered as fairly standard. We will present successively two types of bounds, control of absolute deviations in paragraph 5.3.1 and control of relative deviations (that is, taking the variance into account) in paragraph 5.3.2. One of the questions we have in mind is how the bounding quantities scale with the number of classifiers considered.

5.3.1 Absolute deviations

We give here a control of absolute deviations of empirical mean and variance uniform over convex combinations of a finite family of bounded random variables.

Proposition 5. *Let $X^1, \dots, X^L$ be L random variables of joint probability P and such that $|X^i| \leq 1$, $i = 1, \dots, L$. For $\alpha \in S_{L-1}$ belonging to the $(L-1)$-dimensional simplex, denote $X^\alpha = \sum_i \alpha_i X^i$. Let $M_\alpha = E[X^\alpha]$, and $V_\alpha = Var[X^\alpha]$ for $\alpha \in S_{L-1}$.*

Consider an i.i.d. sample of size N of $X^1, \dots, X^L$ and for $\alpha \in S_{L-1}$, let $\widehat{M}_\alpha$ be the associated empirical mean of variable X_α, $\widehat{M}_{\alpha\alpha}$ the empirical mean of variable X_α^2, and $\widehat{V}_\alpha = \widehat{M}_{\alpha\alpha} - \widehat{M}_\alpha^2$.

Then, except on a set of samples of probability less than $\delta > 0$, the following *inequalities hold for all $\alpha \in S_{L-1}$:*

$$|M_\alpha - \widehat{M}_\alpha| \leq \sqrt{\frac{2\log(2(L+1)^2/\delta)}{N}}$$

and

$$|V_\alpha - \widehat{V}_\alpha| \leq \sqrt{\frac{32\log(2(L+1)^2/\delta)}{N}}$$

Proof. Put $M_i = E[X^i]$, $M_{ij} = E[X^i X^j]$ and $V_{ij} = Cov(X^i, X^j) = M_{ij} - M_i M_j$. Define $\widehat{M}_i$, $i = 1, \dots, L$ as the empirical average of variable X^i, $\widehat{M}_{ij}$ as the empirical average of variable $X^i X^j$, $i, j = 1, \dots, L$, and $\widehat{V}_{ij} = \widehat{M}_{ij} - \widehat{M}_i \widehat{M}_j$.

Using Hoeffding's inequality for these variables, we get that, except on a set of samples of probability less than $2(L + L^2)\exp(-N\varepsilon^2/2) = \delta$, we have simultaneously for all $i = 1, \dots, L$:

$$|M_i - \widehat{M}_i| \leq \varepsilon, \tag{5.22}$$

and for all $i, j = 1, \dots, L$:

$$|M_{ij} - \widehat{M}_{ij}| \leq \varepsilon. \tag{5.23}$$

Summing inequalities (5.22) after reweighting each one of them by α_i yields the first announced inequality. Now multiplicating inequalities (5.22) for indices i and j we get

$$|(\widehat{M}_i - M_i)(\widehat{M}_j - M_j)| \leq \varepsilon^2,$$

hence, assuming $\varepsilon \leq 1$,

$$|\widehat{M}_i \widehat{M}_j - M_i M_j| \leq \varepsilon^2 + M_i|M_j - \widehat{M}_j| + M_i|M_i - \widehat{M}_i| \leq 3\varepsilon. \tag{5.24}$$

Using (5.24) and (5.23) we get for all i, j:

$$|V_{ij} - \widehat{V}_{ij}| \leq |M_{ij} - \widehat{M}_{ij}| + |M_i M_j - \widehat{M}_i \widehat{M}_j| \leq 4\varepsilon.$$

Now summing these inequalities after reweighting by $\alpha_i \alpha_j$ yields the second inequality of the proposition. □

5.3.2 Relative deviations

It was suggested in the previous chapter that reducing the variance of the aggregate classifier (or, equivalently, the average covariance of selected base classifiers) had two interests: first, getting a better bound on generalization error through eq. (5.21); secondly, reducing estimation error (a kind of "second order" effect). In relation to this idea, we derive here a control of empirical mean and variance, relative to variance, uniform over convex combinations of a finite family of bounded random variables.

We will start with a useful result concerning the "poissonian" behavior of empirical variance for a bounded random variable. The result we obtain has most probably been known for a long time (although we don't have a precise reference at hand), but we give it here as a straightforward consequence of a concentration inequality obtained recently by Boucheron, Lugosi and Massart [16]. More precisely we will make use of the following corollary of their theorem:

Theorem 12 (Boucheron,Lugosi,Massart). *Let $\{X_i, i \in \mathcal{I}\}$ be a finite family of independent random variables of distribution P. Define $X = (X_j)_{j\in\mathcal{I}}$ and $X^{(i)} = (X_j)_{J\in\mathcal{I}\setminus\{i\}}$. Let $Z = \zeta(X)$ be some positive and measurable function of X. Assume that for every $i \in \mathcal{I}$ there exists a measurable function $Z^{(i)}$ of $X^{(i)}$ such that*

$$0 \leq Z - Z^{(i)} \leq 1, \tag{5.25}$$

and that

$$\sum_{i\in\mathcal{I}} (Z - Z^{(i)}) \leq Z. \tag{5.26}$$

Then the following inequality holds: for every positive number t,

$$P[Z \leq E[Z] - t] \leq \exp\left(-\frac{t^2}{2E[Z]}\right).$$

Using this result we derive the following proposition:

Proposition 6. *Let ξ be a real random variable of distribution P such that $|\xi| \leq 1$. Let $\xi_1, \ldots, \xi_N$ be an i.i.d. sample from P and $\widehat{C}_N$ be the empirical variance*

$$\widehat{C}_N = \frac{1}{N}\sum_{i=1}^{N} \left(\xi_i - \overline{\xi}_N\right)^2,$$

where $\overline{\xi}_N$ denotes the empirical mean of the sample. Then the following inequality holds:

$$P\left[\widehat{C}_N \leq \frac{N-1}{N}Var[\xi] - \varepsilon\right] \leq \exp\left(-\frac{N\varepsilon^2}{8Var[\xi]}\right) \tag{5.27}$$

Proof. Let

$$Z = \sum_{i=1}^{N} \left(\xi_i - \overline{\xi}_N\right)^2.$$

Z is a positive function of the sample ξ_1^N. Next define for $k = 1, \dots, N$:

$$Z^{(k)} = \sum_{i \neq k} \left(\xi_i - \overline{\xi}_N^{(k)}\right),$$

where $\overline{\xi}_N^{(k)}$ denotes the empirical mean of the sample $\xi_1^N \setminus \{\xi_k\}$. Then

$$\begin{aligned} Z - Z^{(k)} &= \left(\xi_k - \overline{\xi}_N\right)^2 + \sum_{i \neq k} \left[\left(\xi_i - \overline{\xi}_N\right)^2 - \left(\xi_i - \overline{\xi}_N^{(k)}\right)^2\right] \\ &= \left(\xi_k - \overline{\xi}_N\right)^2 + (N-1)\left(\overline{\xi}_N - \overline{\xi}_N^{(k)}\right)^2 \end{aligned}$$

Moreover

$$\begin{aligned} \left(\overline{\xi}_N - \overline{\xi}_N^{(k)}\right)^2 &= \left[\overline{\xi}_N - \left(\frac{N}{N-1}\overline{\xi}^N - \frac{1}{N-1}\xi_k\right)\right]^2 \\ &= \left(\frac{1}{N-1}\xi_k - \frac{1}{N-1}\overline{\xi}_N\right)^2 = \frac{1}{(N-1)^2}\left(\xi_k - \overline{\xi}_N\right)^2, \end{aligned}$$

so that

$$Z - Z^{(k)} = \frac{N}{N-1}\left(\xi_k - \overline{\xi}_N\right)^2.$$

Since $|\xi| \leq 1$ we get

$$0 \leq Z - Z^{(k)} \leq \frac{4N}{N-1}$$

and

$$\sum_i \left(Z - Z^{(i)}\right) = \frac{N}{N-1}Z.$$

So that $Z' = (N-1)Z/4N$ satisfies the hypotheses of theorem 12; since $\widehat{C}^N = 4Z'/(N-1)$ and $E[\widehat{C}] = \frac{N-1}{N}Var[\xi]$ the desired inequality is obtained. □

We can now state the following proposition:

Proposition 7. *With the same notations as in proposition 5, except on a set of training samples of probability less than δ, the following inequalities hold for all $\alpha \in S_{L-1}$:*

$$V_\alpha - \widehat{V}_\alpha \leq \sqrt{\frac{8L\log(2N/\delta)}{N}}\sqrt{\widehat{V_\alpha}} + \frac{8 + 20L\log(2N/\delta)}{N} \tag{5.28}$$

and

$$|M_\alpha - \widehat{M_\alpha}| \leq \sqrt{\frac{L\log(2N/\delta)}{N}}\sqrt{V_\alpha} + \frac{1 + 3L\log(2N/\delta)}{2N}. \tag{5.29}$$

Proof. As in the proof of theorem 9, let $K > 0$ be an integer and let $\widetilde{\alpha}$ be a random variable obtained as the empirical distribution of K independent draws $\xi_1, \dots, \xi_K$ from α. $\widetilde{\alpha}$ is taking its values in S^K_{L-1}, the simplex discretized through a grid of scale $1/K$. Let also $(n_1, \dots, n_L)$ denote the empirical counts for the sample $\xi_1, \dots, \xi_K$.

We can write

$$V_\alpha - \widehat{V}_\alpha = E_{\widetilde{\alpha}}[V_\alpha - \widehat{V}_\alpha] = E_{\widetilde{\alpha}}[V_\alpha - V_{\widetilde{\alpha}}] + E_{\widetilde{\alpha}}[V_{\widetilde{\alpha}} - \widehat{V}_{\widetilde{\alpha}}] + E_{\widetilde{\alpha}}[\widehat{V}_{\widetilde{\alpha}} - \widehat{V}_\alpha]. \tag{5.30}$$

We have

$$\begin{aligned}
E_{\widetilde{\alpha}}[V_{\widetilde{\alpha}}] &= E_{\widetilde{\alpha}}[Var[X^{\widetilde{\alpha}}]] \\
&= E_{\widetilde{\alpha}}\left[\frac{1}{K^2}Var\left[\sum_i n_i X^i\right]\right] \\
&= \frac{1}{K^2}E_{\widetilde{\alpha}}\left[\sum_{i,j} n_i n_j Cov[X^i, X^j]\right] \\
&= \frac{1}{K^2}E_{\widetilde{\alpha}}\left[\sum_{i,j=1}^{L}\left(\sum_{\ell=1}^{K}\mathbf{1}\{\xi_\ell = i\}\right)\left(\sum_{m=1}^{K}\mathbf{1}\{\xi_m = j\}\right)Cov[X^i, X^j]\right] \\
&= \frac{1}{K^2}\left(K(K-1)\sum_{i,j}\alpha_i\alpha_j Cov[X^i, X^j] + K\sum_i \alpha_i Var[X^i]\right) \\
&= \frac{K-1}{K}V_\alpha + \frac{1}{K}\sum_i \alpha_i Var[X^i]
\end{aligned}$$

So that

$$E_{\widetilde{\alpha}}[V_{\widetilde{\alpha}} - V_\alpha] = \frac{1}{K}\left(\sum_i \alpha_i Var[X^i] - V_\alpha\right).$$

The same type of relation is valid for $E_{\widetilde{\alpha}}[\widehat{V}_{\widetilde{\alpha}} - \widehat{V}_\alpha]$ (replace variances and covariances by their empirical counterparts in the above derivation), hence using the fact that $|X^i| \leq 1$ for all i, we get from (5.30):

$$V_\alpha - \widehat{V}_\alpha \leq \frac{4}{K} + E_{\widetilde{\alpha}}[V_{\widetilde{\alpha}} - \widehat{V}_{\widetilde{\alpha}}].$$

Now using inequality (5.27), we get that, except on a set of samples of probability less than $\delta/2M$, one has for a fixed $\widetilde{\alpha} \in S_{L-1}^K$:

$$V_{\widetilde{\alpha}} - \widehat{V}_{\widetilde{\alpha}} \leq \frac{1}{N} + \sqrt{\frac{8V_{\widetilde{\alpha}}}{N} \log\left(\frac{2M}{\delta}\right)}; \tag{5.31}$$

now since $|S_{L-1}^K| \leq K^L$, putting $M = K^L$ and using the union bound, we get that the inequality above is valid for any $\widetilde{\alpha} \in S_{L-1}^K$ except on a set of samples of probability less than $\delta/2$. Integrating with respect to $\widetilde{\alpha}$ and using Jensen's inequality, we therefore get

$$V_\alpha - \widehat{V}_\alpha \leq \frac{4}{K} + \frac{1}{N} + \sqrt{\frac{8L\log(2K/\delta)}{N}\left(\left(1 - \frac{1}{K}\right)V_\alpha + \frac{1}{K}\right)}.$$

Choosing $K = N$, we get

$$V_\alpha - \widehat{V}_\alpha \leq \frac{5}{N} + \sqrt{\frac{8L\log(2N/\delta)}{N}}\left(\sqrt{V_\alpha} + \frac{1}{\sqrt{N}}\right);$$

inverting this last quadratic inequality leads to (5.28). For the second inequality, we proceed similarly, writing

$$|M_\alpha - \widehat{M}_\alpha| = |E_{\widetilde{\alpha}}[M_{\widetilde{\alpha}} - \widehat{M}_{\widetilde{\alpha}}]| \leq E_{\widetilde{\alpha}}[|M_{\widetilde{\alpha}} - \widehat{M}_{\widetilde{\alpha}}|], \tag{5.32}$$

then, for each fixed $\widetilde{\alpha}$, Bernstein's inequality implies that except on a set of probability less than $\exp(-u)$, one has

$$|\widehat{M}_{\widetilde{\alpha}} - M_{\widetilde{\alpha}}| \leq \sqrt{\frac{uV_{\widetilde{\alpha}}}{N}} + \frac{2u}{3N}, \tag{5.33}$$

so that for $u = L\log(2K/\delta)$ this inequality is true simultaneously for all $\widetilde{\alpha} \in S_{L-1}^K$ except for a set of samples of probability less than $\delta/2$. Putting this back in (5.32) with the choice $K = N$ and using Jensen's inequality leads to (5.29). □

Note that by using inequality (5.28) into (5.29), we obtain a control of the relative deviations of the empirical mean based only on empirical quantities.

5.3.3 Discussion

Scaling in the number of classifiers

The important point to be noted is that the order of magnitude in the number of classifiers L is very different in the absolute and relative control bounds: it is of the order of $\sqrt{\log(L)}$ for inequalities obtained in proposition 5 for absolute deviations, and of the order of $\sqrt{L}$ in proposition 7 for relative deviations. This means that proposition 5 would carry over easily to the case where $\mathcal{H}$ is of finite VC dimension d: it would suffice to apply Vapnik and Chervonenkis' classical inequality for absolute deviations instead of Hoeffding's inequality. One would then obtain the same type of control with $d\log(2N+1)$ instead of L. In proposition 7, L already acts as a dimension parameter. It is easy to see that it is not possible to get an order of $\sqrt{\log(L)}$ for a relative deviation inequality in this setting. Namely, consider the case where $L = N$ and $X^1, \ldots, X^L$ are i.i.d. Rademacher variables, for each of which an N-sample is drawn. Let $\widehat{M}_1, \ldots, \widehat{M}_L$ be the corresponding empirical means. If N is big enough, then with probability bounded away from zero, there exists a proportion of, say, $1/8$ among these empirical means that are greater than $1/(4\sqrt{N})$. Then the uniform average of these is also greater than $1/(4\sqrt{N})$, but the variable X^α corresponding to this convex combination has its variance of the order $1/N$. This therefore casts a serious doubt on the idea that the "second order" effect of variance reduction could carry over in the case where $\mathcal{H}$ is of finite VC dimension.

One simple remark has to be made however to conclude on this point: one can see from the proof of proposition 7 that the "limiting factor" is the approximation of an arbitrary convex combination α by a discretized version $\widetilde{\alpha} \in S^K_{L-1}$. But in practice, it is often the case that people only build a "small" number of classifiers on which they perform a uniform average. For example in Amit et al. [2], the authors build for handwritten digit recognition about 20 classification trees, which are then uniformly averaged. Therefore, it can be argued that in practice, one actually needs only take into account discretized convex combinations $\widetilde{\alpha} \in S^K_{L-1}$ (the set of uniform averages of K elements of $\mathcal{H}$). For this purpose we can just use the union bound over all $\widetilde{\alpha} \in S^K_{L-1}$, and, since $|S^K_{L-1}| \leq L^K$, using inequalities (5.31) and (5.33) we can state the following:

Proposition 8. *With the same notations as above, except on a set of training* *samples of probability less than δ, the following inequalities hold for all $\widetilde{\alpha} \in S_{L-1}^K$:*

$$V_{\widetilde{\alpha}} - \widehat{V}_{\widetilde{\alpha}} \leq \sqrt{\frac{8K\log(2L/\delta)}{N}}\sqrt{\widehat{V}_{\widetilde{\alpha}}} + \frac{2 + 10K\log(2L/\delta)}{N} \tag{5.34}$$

and

$$|M_{\widetilde{\alpha}} - \widehat{M}_{\widetilde{\alpha}}| \leq \sqrt{\frac{K\log(2L/\delta)}{N}}\sqrt{V_{\widetilde{\alpha}}} + \frac{2K\log(2L/\delta)}{3N}. \tag{5.35}$$

This means that if one averages over a fixed (or only slowly increasing with N) number K of classifiers, the "second-order" effect of the variance still takes place. The scale $\sqrt{\log(L)}$ leads to think that it should still be the case when $\mathcal{H}$ is of finite VC dimension.

About a slight refinement

One can actually obtain a strictly better bound than inequality (5.21), still only using first and second moment statistics. Namely, one has for a random variable $X \in [-1;1]$

$$P[X \leq 0] = P[(X-a) \leq -a] \leq \frac{E[(X-a)^2]}{a^2};$$

choosing $a = E[X^2]/E[X]$ (instead of $a = E[X]$ previously) leads to

$$P[X \leq 0] \leq \frac{Var(X)}{E[X^2]}.$$

Note that when $Var[X]$ is small, which is what we look for, the improvement gained with respect to (5.21) is small anyway. In any case, it is possible to derive similar control bounds for $E[X^2]$ using exactly the same techniques presented in this section. For deviation bounds of the empirical mean of X^2 relative to its variance, we can actually express the bound in terms of the variance of X since

$$\begin{aligned} Var[X^2] &= \frac{1}{2}E[(X^2 - X'^2)^2] \\ &= \frac{1}{2}E[(X+X')^2(X-X')^2] \\ &\leq 2E[(X-X')^2] = 4Var[X]. \end{aligned}$$

where X, X' are two independent copies of X and we have used the fact that $|X| \leq 1$.

5.4 Minimax margin principle

5.4.1 Setting

The bounds on generalization error based on margin distributions presented in section 5.2 have led authors to propose another principle for selecting an aggregate classifier: namely, find the convex combination of classifiers for which the minimum observed margin on the training sample is the highest possible. Namely, we define for any $\alpha \in \mathcal{M}_1(\mathcal{H})$ the quantity

$$\underline{M_\alpha} = \inf_{1 \le i \le N} M_\alpha(Z_i),$$

we want to find α^* such that

$$\alpha^* = \operatorname*{Argmax}_{\alpha \in \mathcal{M}_1(\mathcal{H})} \underline{M_\alpha}.$$

We will suppose from now on that $\mathcal{H}$ is finite, $\mathcal{H} = \{h_1, \dots, h_L\}$. The preceding definition of α^* thus becomes

$$\alpha^* = \operatorname*{Argmax}_{\alpha \ge 0, \|\alpha\|_1 = 1} \min_{1 \le k \le N} \left(\sum_{i=1}^{L} \alpha_i M_i(Z_k) \right), \tag{5.36}$$

where $M_i(Z_k) = h_k(X_i)Y_i$. Then it can easily be seen that up to rescaling, finding α^* is equivalent to the problem

$$\text{Find } \alpha \in \mathbb{R}^L \text{ minimizing } \|\alpha\|_1 \tag{5.37}$$

$$\text{subject to } \alpha_i \ge 0, \quad 1 \le i \le L, \tag{5.38}$$

$$\text{and } \sum_{i=1}^{L} \alpha_i M_i(Z_k) \ge 1, \quad 1 \le k \le N. \tag{5.39}$$

As pointed out by Shapire et al.[54], this problem bears an analogy with the classical Support Vector Machine setting (in the case of separable data) where the optimization problem to be solved is

$$\begin{aligned} &\text{Find } \omega \in H, b \in \mathbb{R} \text{ minimizing } \|\omega\|_2 \\ &\qquad \text{subject to } \left(\langle \omega | \sigma(X_k) \rangle - b\right) Y_k \ge 1, \quad 1 \le k \le N, \end{aligned} \tag{5.40}$$

where H is some Hilbert space, and σ is some mapping from $\mathcal{X}$ to H. If ω^*, b^* are the solutions of (5.40), the resulting classifier H is then defined as

$$H(X) = \operatorname{sign}\left(\langle \omega^* | \sigma(X) \rangle - b^*\right).$$

The analogy with the minimax margin setting is apparent if one puts $\sigma(x) = (h_i(x))_{i=1,\ldots,L}$, the parameter b is dropped and the minimization is performed on the 1-norm instead of the 2-norm, so the quadratic programming problem of the SVM setting is replaced by a linear programming problem[1].

An other point of view about the minimax margin principle is to interpret it as in the framework of game theory, as proposed by Breiman [18]. The framework of game theory has actually been one of the initial motivations for the derivation of the Boosting algorithm (see [37]). Namely, define the gain matrix Λ by $\Lambda_{ij} = M_i(X_j)$. Then starting from (5.36), we can write

$$\begin{aligned}\min_{\substack{\|\alpha\|_1=1\\ \alpha\geq 0}} \max_{1\leq k\leq N} \left(\sum_{i=1}^{L} \alpha_i M_i(X_k)\right) &= \min_{\substack{\|\alpha\|_1=1\\ \alpha\geq 0}} \max_{1\leq k\leq N} (\alpha\Lambda)_k \\ &= \min_{\substack{\|\alpha\|_1=1\\ \alpha\geq 0}} \max_{\substack{\|\omega\|_1=1\\ \omega\geq 0}} (\alpha\Lambda\omega) \end{aligned} \tag{5.41}$$

Therefore, the convex combination α^* solution of optimization problem (5.37) can also be seen as an optimal mixed strategy in a (zero-sum) game where the first player plays a classifier h from $\mathcal{H}$, the second player plays an example X from the training sample and the payoff (gain of player 1) is given by the margin $h(X)Y$.

With these different formulations in mind, we will be able to derive bounds on the generalization error using a support vector type inequality; we also show in the last paragraph how Blackwell's strategy, a classical algorithm of game theory, can be used to obtain an approximate solution.

5.4.2 A sparsity bound for the minimax margin problem

In the Support Vector Machine setting, one can show that the vector $\omega^* \in H$ solution of problem (5.40) can be written as a linear combination with positive coefficients of the vectors $\sigma(X_i)$. The support vectors are defined as examples of the training set which are given strictly positive weight in this expansion. It can then easily be shown that non-support points do not have an influence on the solution ω^* in the sense that, if one removes from the training set a point X_k that is not a support vector, the solution of the SVM problem with this reduced

1. Authors in the SVM community have proposed "linear support vector machines" which are also based on a minimization of the 1-norm. However, the constraints are of somewhat different nature than in the minimax margin setting, see e.g. [26], p.112

training set is still ω^*. This fact has leads to the following bound in expectation:

Proposition 9 (Vapnik). *For a separable dataset, let ℓ_N be the number of support vectors for a classifier H_N corresponding to vector ω^*_N obtained by solving optimization problem (5.40) from a training set of size N. Then the following inequality holds:*

$$E_N E[\mathrm{error}(H_N)] \leq E_{N+1}[\ell_{N+1}/(N+1)], \tag{5.42}$$

where E_N (resp. E_{N+1}) denotes expectation with respect to a training set of size N (resp. $N+1$) and E expectation with respect to a new example.

This inequality can easily be shown by the "leave-one-out" principle: the average error of H_N built on sample X_1^N on a new example X_{N+1} is the same as the average of the errors of classifiers H_N^i where $\{X_1^{N+1}\} \setminus \{X_i\}$ is used to build the classifier and X_i is used for test. But under the hypothesis that the data is separable, classifier H_{N+1} built with the sample X_1^{N+1} classifies perfectly all examples; furthermore for any support vector X_i of H_{N+1} we have $H_{N+1} = H_N^i$ and therefore H_N^i makes no error on point X_i.

This fact was pointed out by the first authors having defined support vector machines (see [55]), hence the name. However, this bound is of limited practical interest since the bounding quantity involves an expectation and therefore is not accessible through empirical quantities. More recently, Littlestone and Warmuth [46] (also cited in [26]) have derived a bound depending only on the observed numbers of support vectors using a general "compression" argument:

Proposition 10 (Littlestone, Warmuth). *Let $X_1^N \mapsto A(X_1^N)$ be an algorithm for building a classifier from a training sample X_1^N. Let ℓ be the minimum value of* $\mathrm{card}(S)$ *over subsamples $S \subset X_1^N$ such that classifier $A(S)$ still classifies the whole sample X_1^N correctly.*

Then except on a set of training samples of probability less than δ, the following inequality holds:

$$P[\mathrm{error}(A(X_1^N))] \leq \frac{1}{N-\ell}\left(\ell \log\left(\frac{eN}{\ell}\right) + \log\frac{N}{\delta}\right). \tag{5.43}$$

In particular, in the SVM setting the number of support vectors is a majorant of the quantity ℓ above and inequality (5.43) therefore holds when ℓ is replaced by the number of support vectors.

We now want to show that these bounds can be carried over to the minimax margin setting, provided the "support vectors" are suitably defined. Remember the only property of a support point that really matters for deriving bounds (5.42) and (5.43) is that the solution of the optimization problem is not changed if one removes the training point in question.

Having in mind the game-theoretic interpretation (5.41) of the minimax margin optimization problem, one can be tempted to define support points as the support of the optimal strategy ω^* of player 2. However, several problems arise: first, this optimal strategy ω^* is not necessarily unique (and neither is the optimal convex combination α^*). Furthermore, some points may not belong to the support of any optimal strategy ω^*, and yet have an influence on the set of solutions. Consider the following very simple counterexample: the gain matrix is given by $\left(\begin{smallmatrix} 0 & 1 \\ 2 & 1 \end{smallmatrix}\right)$. Then the optimal strategies for player 1 are given by all combinations of the form $(t, 1-t)$ for $t \in [0; 1/2]$. The only optimal strategy for player 2 is to play column 2. Yet if column 1 is removed, then all possible strategies become optimal for player 1. We want to rule that case out since we require that all optimal strategies of player 1 for the reduced game (with non-support "columns" removed) are still optimal strategies for the full game.

This leads to a definition of support vectors as the set columns which are a *best answer* (for which the value of the game is attained) of player 2 for at least one optimal strategy of player 1:

Definition 1. *Let Λ be an (L, N) gain matrix of value V. Define $\mathcal{E}^i_\Lambda$ as the set of equilibrium mixed strategies of player i, $i = 1, 2$. Define the set of support columns of player 2 as*

$$SV_2(\Lambda) = \left\{ j \in \{1, \dots, N\} | \exists \alpha^* \in \mathcal{E}^1_\Lambda : \quad (\alpha^* \Lambda)_j = V \right\}.$$

With this definition we have the property:

Proposition 11. *Let $\widetilde{\Lambda}$ be the gain matrix of the reduced game obtained by deleting all columns not belonging to $SV_2(\Lambda)$. Then*

$$\mathcal{E}^1_{\widetilde{\Lambda}} = \mathcal{E}^1_\Lambda.$$

Proof. It is clear that since we are deleting columns, the value $\widetilde{V}$ of $\widetilde{\Lambda}$ is greater or equal to V. Since all equilibrium strategies ω^* of player 2 are still available, we actually have $\widetilde{V} = V$, and $\mathcal{E}^1_\Lambda \subset \mathcal{E}^1_{\widetilde{\Lambda}}$. Now, consider any $\alpha \notin \mathcal{E}^1_\Lambda$ and assume *ab absurdo* that $\alpha \in \mathcal{E}^1_{\tilde{\Lambda}}$. Let α_0 be the projection of α on the closed convex set $\mathcal{E}^1_\Lambda$. Then for all $0 < t < 1$, $\alpha_t = t\alpha + (1-t)\alpha_0 \notin \mathcal{E}^1_\Lambda$. Let ω_t be a best answer of player 2 to α_t: then $\alpha_t \Lambda \omega_t < V$ for $t > 0$. But for t small enough, for any $i \notin SV_2(\Lambda)$, $(\alpha_t \Lambda)_i > V$ by definition of $SV_2(\Lambda)$. Therefore the support of ω_t has empty intersection with $SV_2(\Lambda)$ and ω_t is still an available strategy in the reduced game, which implies that $\alpha_t \notin \mathcal{E}^1_{\tilde{\Lambda}}$, in contradiction with the convexity of $\mathcal{E}^1_{\tilde{\Lambda}}$. $\square$

This definition of the support vector set thus ensures that any convex combination α^* solution of problem (5.37) with a training set reduced to support training points is also a solution when the full sample is taken into account. Therefore, if one builds an algorithm A giving as an output an optimal strategy $\alpha^* \in \mathcal{E}^1_\Lambda$ (no matter which one exactly), then when the data is separable, bounds (5.42) and (5.43) hold for the associate classifier and $\ell = |SV_2(\Lambda)|$. Let us point out that the interest of this type of bound is that it does not depend directly on the size of $\mathcal{H}$ (like was the case in the previous sections) but only of the number of support points in the training set. Thus the "complexity" of the set $\mathcal{H}$ is actually directly read on the data points (through the number of support points).

In practice, it is however difficult to compute ℓ. In the case where the solution α^* is unique, the following additional property is valid:

Proposition 12. *For a mixed strategy ω, define* $\mathrm{supp}(\omega)$ *as the set of nonzero coordinates of ω.*

If $|\mathcal{E}^1_\Lambda| = 1$, then

$$SV_2(\Lambda) = \bigcup_{\omega \in \mathcal{E}^2_\Lambda} \mathrm{supp}(\omega);$$

additionally,

$$\exists \omega \in \mathcal{E}^2_\Lambda : \quad SV_2(\Lambda) = \mathrm{supp}(\omega).$$

Proof. We use the following result coming from game theory: define

$$I_1 = \bigcup_{\omega \in \mathcal{E}^2_\Lambda} \mathrm{supp}(\omega)$$

and

$$I_2 = \bigcap_{\alpha \in \mathcal{E}^1_\Lambda} \{ i \in \{1, \dots, N\} | (\alpha\Lambda)_i = V \};$$

then $I_1 = I_2$ (see [15],p.54, theorem 1). Since we assume that $|\mathcal{E}^1| = 1$, by definition $I_2 = SV_2(\Lambda)$, hence the result. The second property follows easily from the fact that the set of equilibrium strategies is convex. □

This implies in particular that if the equilibrium strategies (α^*, ω^*) are unique for both players, then $SV_2(\Lambda) = \mathrm{supp}(\omega^*)$, which can simplify the computation of $SV_2(\Lambda)$.

5.4.3 Blackwell's strategy

We conclude this section with an iterative algorithm coming from game theory, that allows an approximate computation of equilibrium strategies. This algorithm initially found by Blackwell [12] has been recently shown to be part of a more general class of algorithms of "adaptive strategies" [41]. The algorithm produces a succession of mixed strategies for player 2, that are played in the game repeated iteratively. At each iteration k, the "regret vector" R_k for player 2 is computed: for each pure strategy i of player 2, the regret R_k^i is defined as the difference between the gain of player 1 until now (that is, in the first k games) and the gain he would have obtained if player 2 had played strategy i systematically until now (leaving the strategies played by player 1 unchanged). Thus when the R_k^i is positive it can be said that player 2 "regrets" not to have played strategy i more often.

Blackwell's algorithm, interpreted in terms of classification, can be described this way:

1. Initialize uniform weights weights $w_1(X_i) = 1/N$, $i = 1, \ldots, N$ on the training sample. Initialize regret vector $R_1^i = 0$, $i = 1, \ldots, N$.

2. Find a classifier h_k having minimum weighted error on training set. (That is: find $j^* = \mathrm{Argmax}_j(\Lambda\omega)_j$).

3. If m_k is the weighted average margin of h_k, i.e. $m_k = \sum_{i=1}^N w_k(X_i) Y_i h_k(X_i)$, update regret vector $R_{k+1}^i = R_k^i + m_k - Y_i h_k(X_i)$ $(= R_k^i + (\Lambda\omega)_{j^*} - \Lambda_{j^*i})$.

4. Update weight vector $\omega_{k+1} \propto (R_{k+1})_+$ and renormalize to 1, then go to step 2.

The aggregate classifier is then taken as the uniform average of the h_k's.

Breiman proposed in [18] several iterative algorithms that are shown to converge towards a classifier α^* attaining the minimax margin. The interest of mentioning here additionally Blackwell's strategy is its simplicity and the fact that it also belongs to what Breiman calls "ARCing" algorithms (for Adaptive Reweighting and Combining), like the Boosting algorithm.

The algorithm presented here can be shown to be such that the distance between the average empirical strategy of player 1 and the set of equilibrium strategies tends to zero. Moreover, Blackwell's strategy actually has a more general property, namely that whatever the strategy played by player 1, the distance between the average regret vector R^k/k and the negative quadrant tends to zero. When player 1 plays a best answer at each stage, as is the case above (step (2) of the algorithm), this entails convergence to the set of equilibrium strategies. However, much like Boosting, this algorithm can also be applied when step (2) is replaced by any suitable procedure for building a classifier from weighted data (not necessarily the best one). In this case, the average classifier will not necessarily converge to a minimax margin classifier; however the property concerning the asymptotic behavior of the average regret vector is still satisfied. This can thus still give information on the asymptotic behavior of the average classifier. As for now we only have started to explore the perspectives thus offered.

5.5 Conclusion and perspectives

We have shown in this chapter how bounds on generalization error of aggregate classifiers could be obtained using only empirical quantities related to margin distributions or to support training points.

However, even if they point in interesting directions, these bounds remain quite loose when applied to practical situations. An important number of points are open to future research. Breiman [18] gives some empirical evidence that the minimax margin principle may not perform as well as Boosting. An important question is therefore, what is the optimal "margin profile" for a given training set and a fixed set $\mathcal{H}$ of classifiers? What are the margin profiles selected by Boosting, and is the precise form of Boosting important for that purpose? Also, our presentation of the minimax margin principle gives rise to specific questions as well, in particular, if it is possible to compute in practice the number of support vectors, or if the uniqueness of the equilibrium strategies can be established in some cases. Finally, the behavior of the aggregate classifier obtained by Blackwell's strategy in the general case where only a suboptimal classifier is build at each stage is yet to be studied. (Note that this question actually applies to the Boosting algorithm as well, since most of the studies on the behavior of Boosting assume that the best classifier, in terms of weighted error, is chosen at each stage)

Bibliographie

[1] Y. Amit, G. Blanchard, and K. Wilder. Multiple randomized classifiers: MRCL. Technical report, University of Chicago, nov 1999. available at `http://galton.uchicago.edu/~amit/Papers/mrcl.ps.gz`.

[2] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.

[3] Y. Amit, D. Geman, and K. Wilder. Recognizing shapes from simple queries about geometry. Technical report, University of Chicago, July 1995. Available at `http://galton.uchicago.edu/~amit/Papers/digits.ps.gz`.

[4] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Trans. PAMI*, 19(11):1300–1306, 1997.

[5] A. Barron. Are Bayes rules consistent in information? In T.M. Cover and B. Gopinath, editors, *Open Problems in Communication and Computation*, pages 85–91. Springer-Verlag, 1987.

[6] A. Barron, L. Birgé, and P. Massart. Risk bounds for model selection via penalization. *Probability theory and related fields*, 113:301–413, 1999.

[7] A. Barron and Y. Yang. Information theoretic determination of minimax rates of convergence. *The Annals od Statistics*, 27:1564–1599, 1999.

[8] R. Barron and T. M. Cover. Minimum complexity density estimation. *IEEE transactions on information theory*, 37(4):1034–1054, 1991.

[9] L. Birgé. Approximation dans les espaces métriques et théorie de l'approximation. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 65:181–237, 1983.

[10] L. Birgé and P. Massart. From model selection to adaptative estimation. In E. Torgensen, D. Pollard, and Yang. G., editors, *Festschrift for Lucien Le Cam: research papers in probability and statistics*, pages 55–87. Springer, New York, 1997.

[11] L. Birgé and P. Massart. Minimum contrast estimators on sieves. *Bernoulli*, 4:329–375, 1998.

[12] D. Blackwell. An analog of the minmax theorem for vector payoffs. *Pacific Journal of Mathermatics*, (6):1–8, 1956.

[13] G. Blanchard. L'estimateur de «mélange progressif» appliqué aux arbres de décision. *C.R.A.S.*, 328, Série I:925–928, 1999.

[14] G. Blanchard. The "progressive mixture" estimator for regression trees. *Annales de l'I.H.P.*, 35(6):793–820, 1999.

[15] H. F. Bohnenbust, S. Karlin, and L. S. Shapley. Solutions of finite two-person games. In Kuhn H. W. and A. W. Tucker, editors, *Contribution to the theory of games*. Princeton University Press, 1950.

[16] S. Boucheron, G. Lugosi, and P. Massart. A sharp concentration inequality with applications. Technical report, Université Paris-Sud Orsay, apr 1999.

[17] L. Breiman. Bagging predictors. *Machine Learning*, 26:123–140, 1996.

[18] L. Breiman. Prediction games and arcing algorithms. Technical report, Statistics department, University of California at Berkeley, December 1997. Available at `ftp://ftp.stat.berkeley.edu/pub/users/breiman/games-.ps.Z`.

[19] L. Breiman. Arcing classifiers. *The annals of statistics*, 26(3):801–849, 1998.

[20] L. Breiman. Random forests - random features. Technical report, University of California, Berkeley, 1999.

[21] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression Trees*. Wadsworth, Belmont, California, 1984.

[22] O. Catoni. A mixture approach to universal model selection. Preprint of the E.N.S., available at `http://www.dma.ens.fr/edition/publis/index.html`, 1997.

[23] O. Catoni. Gibbs estimators. *Preprint of the Laboratoire de Mathématiques de l'Ecole Normale Supérieure*, LMENS-98(21):1–23, may 1998. available at `http://www.dma.ens.fr/edition/publis/index.html`.

[24] O. Catoni. "Universal" aggregation rules with exact bias bounds. *Preprint of the laboratoire de Probabilités et modèles aléatoires, Université Pierre et Marie Curie*, 1999. available at `http://www.proba.jussieu.fr/mathdoc/preprints/index.html#1999` (to appear in *Annals of Statistics*).

[25] H. Chipman, E. I. George, and E. McCulloch. Bayesian CART model search. *JASA*, 93:935–947, September 1998.

[26] N. Christianini and J. Shawe-Taylor. *An introduction to Support Vector Machines*. Cambridge University Press, 2000.

[27] M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. In *Proceedings of the thirteenth annual conference on computational learning theory*, 2000.

[28] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley series in telecommunications. J. Wiley, 1991.

[29] L. D. Davisson. Universal noiseless coding. *IEEE transactions on information theory*, IT-19(6):783–795, november 1973.

[30] L. D. Davisson and A. Leon-Garcia. A source matching approach to finding minimax codes. *IEEE transactions on information theory*, IT-26(2):166–174, march 1980.

[31] D. Denison, B. Mallick, and A. F. M. Smith. A bayesian CART algorithm. *Biometrika*, (85-2):363–377, June 1998.

[32] L. Devroye and L. Györfi. *Nonparametric density estimation: the L^1 view*. Wiley, New York, 1985.

[33] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization. *Machine Learning*, 2000.

[34] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting outut codes. *J. Artificial Intell. Res.*, 2:263–286, 1995.

[35] M. Feder and N. Merhav. Hierarchical universal coding. *IEEE transactions on information theory*, 42(5):1354–1364, September 1996.

[36] Y. Freund and R. Schapire. Experiments on a new boosting algorithm. *Machine Learning: proceedings of the 13th international conference*, pages 148–156, 1996.

[37] Y. Freund and R. E. Schapire. Game theory, on-line prediciton and Boosting. In *Proceedings of the 9th annual conference on computational learning theory*, 1996.

[38] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55:119–139, 1997.

[39] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28:337–374, 2000.

[40] R. Gribonval. *Approximation non-linéaires pour l'analyse de signaux sonores.* PhD thesis, Université Paris 9 Dauphine, 1999.

[41] S. Hart and A. Mas-Colell. A general class of adaptive strategies. *Journal of Economic Litterature*.

[42] D. Helmbold and R. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27:51–68, 1997.

[43] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. PAMI*, 20(8):832–844, 1998.

[44] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE trans. PAMI*, 16:66–75, 1994.

[45] R. E. Krichevsky and V. K. Trofimov. The performance of universal encoding. *IEEE transactions on information theory*, IT-27:199–207, march 1981.

[46] N. Littlestone and M. Warmuth. Relating data compression and learnability. Technical report, University of California, Santa Cruz, 1986.

[47] G. Lugosi. Lectures on statistical learning theory. Presented at the Garchy Seminar on Mathematical Statistics and Applications, available at `http://www.econ.upf.es/~lugosi`, 2000.

[48] P. Massart. Some applications of concentration inequalities in statistics. Technical report, Université de Paris-Sud Orsay, 2000.

[49] A. Murua. Upper bounds for error rates associated to linear combination of classifiers. Technical Report 354, University of Washington, 1999.

[50] J. Rissanen. A universal data compression system. *IEEE transactions on information theory*, IT-32(5):656–664, September 1983.

[51] J. Rissanen. Stochastic complexity and modeling. *The annals of statistics*, 14(3):1080–1100, 1986.

[52] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 2000. To appear.

[53] B. Y. Ryabko. Twice-universal coding. *Problems of information transmission*, 20:24–28, 1984.

[54] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margins: a new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

[55] V. N. Vapnik. *Estimation of dependences based on empirical data.* Springer series in statistics. Springer-Verlag, 1982.

[56] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, 1995.

[57] V. N. Vapnik. *Statistical learning theory.* Wiley, 1998.

[58] K. Wilder. *Decision trees for shape recognition.* PhD thesis, University of Massachusets, Amherst, 1998.

[59] F. M. J. Willems, Y. M. Shtarkov, and T J. Tjalkens. The context-tree weighting method: basic properties. *IEEE transactions on information theory*, 41(3):653–664, 1995.

[60] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. Context weighting for general finite-context sources. *IEEE transactions on information theory*, 42(5):1514–1520, September 1996.