PASCAL GERMAIN

# Algorithmes d'apprentissage automatique inspirés de la théorie PAC-Bayes

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de Maître ès Sciences (M. Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2009

# Résumé

Dans un premier temps, ce mémoire présente un théorème PAC-Bayes général, duquel il est possible d'obtenir simplement plusieurs bornes PAC-Bayes connues. Ces bornes permettent de calculer une garantie sur le risque d'un classificateur à partir de ses performances sur l'ensemble de données d'entraînement. Par l'interprétation du comportement de deux bornes PAC-Bayes, nous énonçons les caractéristiques propres aux classificateurs qu'elles favorisent. Enfin, une spécialisation de ces bornes à la famille des classificateurs linéaires est détaillée.

Dans un deuxième temps, nous concevons trois nouveaux algorithmes d'apprentissage automatique basés sur la minimisation, par la méthode de descente de gradient conjugué, de l'expression mathématique de diverses formulations des bornes PAC-Bayes. Le dernier algorithme présenté utilise une fraction de l'ensemble d'entraînement pour l'acquisition de connaissances *a priori*. Ces algorithmes sont aptes à construire des classificateurs exprimés par vote de majorité ainsi que des classificateurs linéaires exprimés implicitement à l'aide de la stratégie du noyau. Finalement, une étude empirique élaborée compare les trois algorithmes entre eux et révèle que certaines versions de ces algorithmes construisent des classificateurs compétitifs avec ceux obtenus par AdaBoost et les SVM.

# Abstract

At first, this master thesis presents a general PAC-Bayes theorem, from which we can easily obtain some well-known PAC-Bayes bounds. Those bounds allow us to compute a guarantee on the risk of a classifier from its achievements on the training set. We analyze the behavior of two PAC-Bayes bounds and we determine peculiar characteristics of classifiers favoured by those bounds. Then, we present a specialization of those bounds to the linear classifiers family.

Secondly, we conceive three new machine learning algorithms based on the minimization, by conjugate gradient descent, of various mathematical expressions of the PAC-Bayes bounds. The last algorithm uses a part of the training set to capture *a priori* knowledges. One can use those algorithms to construct majority vote classifiers as well as linear classifiers implicitly represented by the kernel trick. Finally, an elaborated empirical study compares the three algorithms and shows that some versions of those algorithms are competitive with both AdaBoost and SVM.

# Avant propos

D'autres vous le diront, effectuer une maîtrise de recherche n'est pas une mince affaire. Sans pour autant exagérer l'envergure de la tâche, il me paraît approprié d'affirmer ma fierté devant le travail accompli, ne serait-ce que pour souligner l'importance que j'accorde aux remerciements qui suivent. L'aventure aurait été drôlement plus compliquée sans la connivence des gens nommés ici-bas.

Je suis d'abord reconnaissant envers François Laviolette et Mario Marchand, respectivement mon directeur et mon codirecteur de recherche, de m'avoir accompagné et guidé dans ces travaux. J'ai apprécié la confiance que ces chercheurs passionnés ont témoignée à mon égard.

Je suis aussi redevable aux collègues qui ont peuplé le laboratoire du Groupe de recherche en apprentissage automatique de l'Université Laval (le sympathique GRAAL) pendant mes activités de recherche, auxquelles chacun a apporté sa contribution. Sébastien Boisvert a transformé les laboratoires informatiques du département en puissantes grilles de calculs. Alexandre Lacoste m'a conseillé sur les techniques de descente de gradient. Sara Shanian a été d'un encouragement constant. Jean-Francis Roy m'a aidé dans mes expérimentations. Je remercie particulièrement Alexandre Lacasse, qui a contribué à toutes les étapes du travail dont il sera question et qui a été une référence mathématique essentielle. Plusieurs de ces étudiants gradués ont aussi contribué au présent document par leurs commentaires et par leurs corrections. Outre l'apport de mes collègues, j'ai bénéficié des talents de correcteur et de la générosité légendaire de mon ami Daniel Gendron.

Enfin, je remercie Thérèse et Raymond, parents indéfectibles dans le support qu'ils apportent à leurs enfants.

# Table des matières

# Liste des tableaux

# Table des figures

# Chapitre 1

# Introduction

GÉRONTE : Que diable allait-il faire à cette galère ?

Les fourberies de Scapin (Pièce de Molière)

L'intelligence artificielle est un vaste domaine de recherche. On y regroupe plusieurs approches partageant l'ambition de reproduire, à l'aide d'un ordinateur, des comportements communément qualifiés d'intelligents lorsqu'ils sont observés chez un être vivant. Cela peut, entres autres, se manifester par la reconnaissance de la parole, la synthèse d'un texte, l'élaboration d'une stratégie gagnante aux échecs ou le processus de déduction permettant de combiner une série d'axiomes afin d'en tirer une conclusion.

L'apprentissage automatique est la branche de l'intelligence artificielle qui s'intéresse à la faculté d'apprendre à effectuer une tâche à partir de l'observation d'un environnement. Lorsqu'appliqué à des problèmes de classification, il s'agit d'apprendre à distinguer entre eux divers éléments de cet environnement par l'observation d'exemples. C'est par cette faculté qu'un enfant apprend à reconnaître entre elles chacune des vingt-six lettres de l'alphabet latin. Similairement, les algorithmes d'apprentissage automatique sont couramment utilisés pour construire des classificateurs de caractères manuscrits à l'aide d'une banque d'images de lettres de A à Z écrites par différents individus. Il ne s'agit que d'un exemple d'usage de l'apprentissage automatique, et il existe une multitude d'autres applications à une telle technologie. Notamment, on peut construire des classificateurs permettant de distinguer un champignon comestible d'un champignon vénéneux, de prévoir s'il pleuvra demain à partir des conditions météorologiques actuelles ou de produire un diagnostic médical (déterminer si un patient est porteur d'une maladie à partir d'un ensemble de symptômes).

Le défi considérable de l'apprentissage automatique est de développer un algorithme capable de résoudre efficacement des problèmes de classification de natures diverses. La

qualité d'un algorithme d'apprentissage dépend donc de sa capacité à construire un classificateur qui généralise le phénomène observé. En ce sens, on peut comparer un bon algorithme d'apprentissage à un bon élève, chez qui la compréhension de la matière va au-delà de la stricte mémorisation des exemples étudiés. Lorsqu'il apprend une nouvelle notion, cet élève sait en dégager les caractéristiques clés. Un bon algorithme d'apprentissage doit incorporer des mécanismes le dotant de capacités similaires. Chaque algorithme d'apprentissage suggère sa propre stratégie afin de généraliser les données représentant le phénomène à apprendre. Les algorithmes communément utilisés en pratique sont adoptés en vertu de leur efficacité empirique, à la lumière autant de leur rapidité d'exécution que de l'acuité des classificateurs qu'ils construisent.

Parallèlement au développement d'algorithmes efficaces, la recherche en apprentissage automatique s'intéresse à l'étude théorique du problème d'apprentissage. Les théories statistiques de l'apprentissage supposent que les exemples caractérisant chaque phénomène sont générés en accord avec une distribution de probabilité. De ce point de vue, la tâche d'un algorithme d'apprentissage est de construire une fonction (un classificateur) qui estime cette distribution. Ainsi, on parvient à énoncer des bornes supérieures sur le risque d'un classificateur, c'est-à-dire sur la probabilité qu'il classifie incorrectement un exemple. Par la suite, le calcul de ces bornes fournit une garantie sur la qualité d'un classificateur, que l'on formule habituellement ainsi : « Avec probabilité $1-\delta$, le risque que le classificateur $h$ classifie incorrectement un exemple $\mathbf{x}$ issu d'un phénomène donné est d'au plus $B$ » (où les valeurs de $\delta$ et $B$ sont situées entre 0 et 1).

En analysant les quantités intervenant dans les expressions mathématiques des bornes sur le risque, on peut dégager une intuition des caractéristiques propres aux bons classificateurs. Conséquemment, les bornes théoriques peuvent servir d'inspiration au développement d'algorithmes d'apprentissage. Plus encore, il est possible de concevoir un algorithme qui sélectionne, parmi une famille préétablie de classificateurs, le classificateur qui minimise l'expression mathématique d'une borne. Lorsque la borne est un bon indicateur du risque des classificateurs, elle devrait se traduire par un algorithme d'apprentissage de qualité. Pourtant, les algorithmes d'apprentissage actuels n'exploitent pas cette stratégie prometteuse. Cela s'explique à la fois par le manque de précision de la plupart des bornes sur le risque et par la complexité de leur expression mathématique qui rend difficile la conception d'algorithmes d'optimisation efficaces.

Au cours des dernières années, la théorie PAC-Bayes a fait l'objet de plusieurs travaux et a permis d'énoncer des bornes assez serrées sur le risque des classificateurs. L'acuité de ces bornes indique qu'il s'agit d'une théorie représentant bien le problème de l'apprentissage. Considérant cela, les travaux de recherche dont fait état le présent mémoire adoptent la théorie PAC-Bayes comme point de départ pour la conception de

nouveaux algorithmes d'apprentissage. Nous concentrons notre étude sur trois variantes de ces algorithmes. Ces derniers effectuent la minimisation, par la méthode de descente de gradient conjugué, de l'expression mathématique des bornes PAC-Bayes appliquées à la famille des classificateurs linéaires. Ils possèdent l'avantage de fournir une garantie sur le risque de chaque classificateur qu'ils construisent. Les expérimentations empiriques montrent que certains de ces algorithmes, bien que lents d'exécution, construisent des classificateurs compétitifs avec ceux obtenus par des algorithmes d'apprentissage communément utilisés. Nos résultats confirment le potentiel des algorithmes d'apprentissage basés sur la minimisation de bornes théoriques et motivent la poursuite des recherches dans cette voie.

Le contenu du mémoire est réparti ainsi :

- Le chapitre 2 (*Notions de base*) introduit les concepts clés de l'apprentissage automatique et de la classification. Les notions de classificateur (Decision Stump, vote de majorité, classificateur linéaire), d'algorithme d'apprentissage (AdaBoost, SVM) et de sélection de modèle (validation croisée, bornes sur l'ensemble test et bornes sur l'ensemble d'entraînement) y sont notamment présentés.
- Le chapitre 3 (*Éléments de la théorie PAC-Bayes*) expose les fondements de la théorie PAC-Bayes, qui permet d'obtenir des bornes sur le risque d'un classificateur à partir de ses performances sur l'ensemble d'entraînement. Il contient la démonstration d'un théorème PAC-Bayes général, duquel nous dérivons simplement deux bornes PAC-Bayes connues. Une attention particulière est accordée à l'analyse du comportement de ces deux bornes. Nous présentons ensuite une spécialisation des bornes PAC-Bayes aux classificateurs linéaires et nous étudions sa performance dans un contexte de sélection de modèle.
- Le chapitre 4 (*Algorithmes de descente de gradient PAC-Bayes*) présente des mécanismes de minimisation des bornes PAC-Bayes par descente de gradient conjugué. Nous y élaborons une méthode permettant de dédier une fraction des données d'apprentissage à l'acquisition de connaissances *a priori* sur le phénomène étudié. Nous présentons finalement trois nouveaux algorithmes d'apprentissage automatique. Ces algorithmes s'appliquent à la fois aux classificateurs par vote de majorité et aux classificateurs linéaires.
- Le chapitre 5 (*Expérimentations et analyse des résultats*) regroupe les résultats des expérimentations empiriques obtenus à l'aide de nos nouveaux algorithmes d'apprentissage automatique. Ces résultats sont comparés à ceux d'algorithmes d'apprentissage communément utilisés (AdaBoost et SVM) et analysés afin d'en expliquer les forces et les faiblesses.
- Enfin, le chapitre 6 (*Conclusion et travaux futurs*) discute d'avenues de recherche intéressantes afin de poursuivre le développement d'algorithmes basés sur la minimisation de bornes théoriques.

# Chapitre 2

# Notions de base

> 11:15 Restate my assumptions :
> 1. Mathematics is the language of nature.
> 2. Everything around us can be represented and understood through numbers.
> 3. If you graph these numbers, patterns emerge.
> Therefore : There are patterns everywhere in nature.
>
> Pi (film de Darren Aronofsky)

En apprentissage automatique, un problème de classification se présente sous la forme d'un ensemble de données, contenant des exemples issus de l'observation d'un phénomène. Chaque exemple est constitué d'une description et d'une étiquette. Un algorithme d'apprentissage analyse ces données afin de construire un classificateur. C'est à ce classificateur que revient ensuite la tâche d'étiqueter de nouveaux exemples à partir de leur description.

Afin d'étudier le problème de classification auquel nous nous intéresserons, ce chapitre définit en termes mathématiques les notions qui y sont reliées. De même, il présente quelques techniques couramment utilisées en apprentissage automatique.

## 2.1 Ensemble de données

Les problèmes d'apprentissage étudiés sont énoncés sous forme de données numériques. Ces données caractérisent une série d'instances du phénomène à apprendre, que l'on nomme *exemples*. Chaque exemple $\mathbf{z}$ est constitué d'une description $\mathbf{x}$ et d'une

étiquette $y$ [1].

$$\begin{array}{rrcl} & \mathbf{z} & \stackrel{\text{def}}{=} & (\mathbf{x}, y) \\ \text{où} & \mathbf{x} & \in & \mathcal{X} \subseteq \mathbb{R}^n \\ & y & \in & \mathcal{Y} = \{-1, +1\} \end{array}$$

On nomme respectivement les ensembles $\mathcal{X}$ et $\mathcal{Y}$ *l'espace d'entrée* et *l'espace de sortie*.

La description $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ de l'exemple s'avère un vecteur de valeurs réelles de dimension $n$, où $n$ est le nombre d'*attributs* de l'exemple. Selon le phénomène représenté, un attribut peut être numérique (par exemple, une température en degrés Celsius) ou nominal (une couleur, quand il a été déterminé arbitrairement que la valeur 0 équivaut à «rouge», 1 équivaut à «bleu» et 2 équivaut à «jaune»)

L'étiquette $y$ de l'exemple détermine sa *classe* d'appartenance. Les problèmes étudiés dans ce texte seront tous des problèmes de *classification binaire*, c'est-à-dire des problèmes de classification à seulement deux classes[2]. Pour chaque problème, on détermine arbitrairement que les exemples appartenant à une classe sont des exemples négatifs ($y = -1$) et que les autres exemples sont positifs ($y = +1$).

Plusieurs exemples d'un même phénomène sont regroupés dans un *ensemble de données*. On nomme *ensemble d'entraînement* l'ensemble des données dédiées à l'apprentissage. Cet ensemble est désigné par $S$ et le nombre d'exemples qu'il contient par $|S|$ ou $m$.

$$S \stackrel{\text{def}}{=} \{\mathbf{z}_i\}_{i=1}^{m} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$$

Fréquemment, on utilise un *ensemble test* afin d'évaluer la qualité d'un classificateur. Cet ensemble est désigné par $T$ et le nombre d'exemples qu'il contient par $|T|$. Les ensembles d'entraînement et de test sont composés d'exemples provenant d'observations distinctes.

Certains problèmes d'apprentissage s'intéressent aux situations où les exemples contenus dans l'ensemble d'entraînement ne sont pas étiquetés (*apprentissage non-supervisé*) ou ne sont étiquetés que partiellement (*apprentissage semi-supervisé*). En revanche, les travaux présentés dans ce texte s'intéressent aux problèmes *d'apprentissage supervisé*, où l'ensemble d'entraînement ne contient que des exemples étiquetés.

Chaque phénomène étudié est issu d'un environnement régi par ses propres règles.

[1] Dans ce texte, les variables en caractères gras représentent des vecteurs.

[2] Il est possible de diviser un problème multiclasse en un ensemble de sous-problèmes de classification binaire et de le résoudre par une série d'appels à un algorithme d'apprentissage dédié à la classification binaire. Certaines méthodes employées dans de telles circonstances sont décrites dans [17].

Les exemples observés sont générés en accord avec ces règles. C'est pourquoi on considère qu'il existe une distribution de probabilité $D$ sur $\mathcal{X} \times \mathcal{Y}$ qui génère les exemples. De ce point de vue, un ensemble de données est une collection de réalisations d'une variable aléatoire $\mathbf{Z}$ dont la distribution est $D$. Nous prenons pour acquis qu'il s'agit de réalisations *indépendantes et identiquement distribuées* (*iid*), c'est-à-dire que chaque exemple est généré selon la même distribution de probabilité $D$ et la probabilité d'observer un exemple est indépendante des autres exemples observés.

## 2.2 Classificateurs et concepts reliés

Un *classificateur* (aussi appelé *classificateur binaire*) $h : \mathcal{X} \to \{-1, +1\}$ est une fonction qui, étant donnée la description $\mathbf{x}$ d'un exemple, prédit sa classe d'appartenance :

$$h(\mathbf{x}) \;=\; y' \,.$$

Le classificateur idéal prédit $h(\mathbf{x}) = y$ pour tous les exemples $(\mathbf{x}, y)$ observables. Lorsque précisé, on a recours à un *classificateur à valeurs réelles* $h : \mathcal{X} \to \mathbb{R}$. Dans ce cas, la classe prédite par $h$ est donnée par $\mathrm{sgn}(y')$. Pour une valeur réelle $s$, on a :

$$\mathrm{sgn}(s) \stackrel{\text{def}}{=} \begin{cases} +1 & \text{si } s > 0 \\ -1 & \text{sinon.} \end{cases}$$

Habituellement, l'amplitude $|y'|$ représente la confiance du classificateur en sa prédiction.

### 2.2.1 Risque

Le *risque* (aussi désigné par *vrai risque* dans ce document) $R(h)$ d'un classificateur $h$ est la probabilité que ce dernier classe incorrectement un exemple tiré dans $\mathcal{X} \times \mathcal{Y}$ selon une distribution de probabilité inconnue $D$. Autrement dit :

$$R(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y) \,,$$

où $I$ est une fonction indicatrice : $I(a) = 1$ si l'énoncé $a$ est vrai et $I(a) = 0$ sinon.

Le *risque empirique* $R_S(h)$ d'un classificateur $h$ correspond au taux d'erreur qu'il réalise sur l'ensemble d'entraînement $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ :

$$R_S(h) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} I(h(\mathbf{x}_i) \neq y_i) \,.$$

La qualité d'un algorithme d'apprentissage est jugée par sa capacité à produire un classificateur à faible risque. Notons que le risque empirique d'un classificateur ne permet pas, à lui seul, d'en estimer le vrai risque. Pour s'en convaincre, considérons le classificateur $h_{\text{SA}}$ dont la sortie est donnée par :

$$h_{\text{SA}}(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} +1 & \text{si } (\mathbf{x}, +1) \in S \\ -1 & \text{sinon.} \end{cases} \tag{2.1}$$

Le classificateur $h_{\text{SA}}$ possède un risque empirique nul[3] alors que son vrai risque est possiblement très élevé. En effet, il classifie incorrectement tous les exemples positifs qui n'appartiennent pas à l'ensemble d'entraînement. On dit qu'il y a *surapprentissage* des données d'entraînement lorsqu'un classificateur de faible risque empirique possède un vrai risque élevé.

### 2.2.2 Decision Stumps

Le *decision stump* est un classificateur très simple. Pour classifier un exemple $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, le decision stump $h_{i,t,d}$ considère la valeur d'un seul attribut $x_i$. La sortie du classificateur est déterminée par la valeur de seuil $t \in \mathbb{R}$ et la direction $d \in \{-1, +1\}$ :

$$h_{i,t,d}(\mathbf{x}) = \begin{cases} +d & \text{si } x_i > t \\ -d & \text{sinon.} \end{cases}$$

### 2.2.3 Classificateurs linéaires

Un *classificateur linéaire* est défini par un hyperplan qui sépare en deux l'espace d'entrée $\mathcal{X} \subseteq \mathbb{R}^n$ des exemples et qui agit en tant que frontière de décision : les exemples situés d'un côté de l'hyperplan sont classés positivement et ceux situés de l'autre côté sont classés négativement. On caractérise cet hyperplan par un vecteur $\mathbf{w} \in \mathbb{R}^n$ (perpendiculaire à l'hyperplan) et une valeur de biais $b \in \mathbb{R}$. L'équation $\mathbf{w} \cdot \mathbf{x} = b$ définit l'hyperplan dans l'espace des exemples. La valeur de sortie du classificateur $h_{\mathbf{w},b}$ correspondant est donnée par :

$$h_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} - b) = \text{sgn}\left(\sum_{i=1}^{n} w_i x_i - b\right). \tag{2.2}$$

[3] Pour les fins de cet exemple, nous considérons que $\nexists \mathbf{x}$ tel que $(\mathbf{x}, -1) \in S$ et $(\mathbf{x}, +1) \in S$.

Pour simplifier la notation, on peut représenter implicitement le biais par l'ajout d'une composante aux vecteurs $\mathbf{x}$ et $\mathbf{w}$ pour obtenir les vecteurs $\mathbf{x}' \stackrel{\text{def}}{=} (1, \mathbf{x})$ et $\mathbf{w}' \stackrel{\text{def}}{=} (-b, \mathbf{w})$. Ainsi, la sortie du classificateur devient :

$$h_{\mathbf{w}'}(\mathbf{x}') = \text{sgn}(\mathbf{w}' \cdot \mathbf{x}') = \text{sgn}\left(\sum_{i=1}^{n+1} w'_i x'_i\right).$$

On dit qu'un ensemble d'entraînement est *linéairement séparable* lorsqu'il existe un hyperplan qui classifie correctement tous ses exemples. En d'autres termes, un ensemble d'entraînement $S$ est linéairement séparable s'il existe un classificateur linéaire $h_{\mathbf{w}'}$ dont le risque empirique $R_S(h_{\mathbf{w}'})$ est nul.

### 2.2.4 Marges d'un classificateur linéaire

Pour un ensemble d'entraînement donné, il existe une multitude de classificateurs linéaires possédant le même risque empirique. Afin de différencier ces classificateurs entre eux, on s'intéresse notamment à la distance entre la frontière de décision et les exemples de l'ensemble d'entraînement. On nomme cette mesure la *marge*. Nous distinguons trois types de marges différentes. Pour simplifier la notation, nous considérons que le biais est représenté implicitement dans le vecteur $\mathbf{w}$.

La *marge fonctionnelle* $\mu_{\mathbf{w}}$ correspond au produit scalaire entre les vecteurs $y\mathbf{x}$ et $\mathbf{w}$. C'est-à-dire :

$$\mu_{\mathbf{w}}(\mathbf{x}, y) \stackrel{\text{def}}{=} y\mathbf{w} \cdot \mathbf{x}\,.$$

La *marge géométrique* $\gamma_{\mathbf{w}}$ correspond à la distance euclidienne entre la frontière de séparation et un exemple :

$$\gamma_{\mathbf{w}}(\mathbf{x}, y) \stackrel{\text{def}}{=} \frac{y\mathbf{w} \cdot \mathbf{x}}{\|\mathbf{w}\|}\,.$$

La *marge normalisée* $\Gamma_{\mathbf{w}}$ correspond au cosinus de l'angle entre les vecteurs $y\mathbf{x}$ et $\mathbf{w}$. Sa valeur appartient à l'intervalle $[-1, +1]$ :

$$\Gamma_{\mathbf{w}}(\mathbf{x}, y) \stackrel{\text{def}}{=} \frac{y\mathbf{w} \cdot \mathbf{x}}{\|\mathbf{w}\|\|\mathbf{x}\|}\,. \tag{2.3}$$

Notons qu'un classificateur linéaire $h_{\mathbf{w}}$ classifie correctement un exemple $(\mathbf{x}, y)$ lorsque sa marge est positive et incorrectement lorsque sa marge est négative.

Lorsqu'un ensemble d'entraînement $S$ est linéairement séparable, la *marge d'un classificateur* sur cet ensemble désigne la marge de l'exemple situé le plus près de la frontière de décision. Par exemple, la marge géométrique du classificateur $h_{\mathbf{w}}$ sur l'ensemble d'entraînement $S$ équivaut à :

$$\gamma_{\mathbf{w}}(S) \stackrel{\text{def}}{=} \min_{(\mathbf{x}_i, y_i) \in S} [\gamma_{\mathbf{w}}(\mathbf{x}_i, y_i)] \,. \tag{2.4}$$

La *marge maximale* d'un ensemble d'entraînement $S$ linéairement séparable correspond à la plus grande marge qu'un classificateur peut obtenir sur $S$. Par exemple, la marge normalisée maximale sur $S$ équivaut à :

$$\Gamma(S) \stackrel{\text{def}}{=} \max_{\mathbf{w} \in \mathbb{R}^n} \left( \min_{(\mathbf{x}_i, y_i) \in S} [\Gamma_{\mathbf{w}}(\mathbf{x}_i, y_i)] \right) .$$

### 2.2.5 Espace des caractéristiques

Il existe plusieurs ensembles de données sur lesquels aucun classificateur linéaire ne possède un risque empirique nul. Autrement dit, ces ensembles ne sont pas linéairement séparables. Dans ce contexte, les classificateurs linéaires, tels qu'exprimés jusqu'à maintenant, peuvent s'avérer peu performants. Ainsi, il est parfois utile d'exprimer des classificateurs par des frontières de décision plus complexes qu'un simple hyperplan. Une façon de représenter ces séparateurs complexes est de transposer les données de l'espace d'entrée $\mathcal{X} \subseteq \mathbb{R}^n$ vers un espace de dimension plus élevée $\mathcal{K} \subseteq \mathbb{R}^{n'}$ (avec $n < n'$). L'espace $\mathcal{K}$ se nomme *l'espace des caractéristiques*. Par cette stratégie, chaque exemple $\mathbf{x} \in \mathcal{X}$ est transformé en un *vecteur de caractéristiques* $\boldsymbol{\phi}(\mathbf{x}) \in \mathcal{K}$ :

$$\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_{n'}(\mathbf{x})) \,.$$

Le vecteur $\mathbf{w}$ de dimension $n'$ caractérise un séparateur linéaire dans l'espace $\mathcal{K}$.

$$h_{\mathbf{w},b}(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) - b) = \operatorname{sgn}\left( \sum_{i=1}^{n'} w_i \phi_i(\mathbf{x}) - b \right) .$$

Ainsi, le classificateur $h_{\mathbf{w},b}$ résultant exprime un séparateur non-linéaire dans l'espace des exemples $\mathcal{X}$.

À titre d'exemple, étudions un ensemble de données à deux attributs $x_1, x_2$ que l'on transpose dans un espace des caractéristiques à cinq dimensions à l'aide de la transformation suivante :

$$\boldsymbol{\phi}\,(x_1, x_2) = \left(x_1^2,\ x_2^2,\ x_1 x_2,\ x_1,\ x_2\right) .$$

Par cette transformation, l'exemple $\mathbf{x}_a = (2, 7)$ est associé au vecteur de caractéristiques $\boldsymbol{\phi}(\mathbf{x}_a) = (4, 49, 14, 2, 7)$. Le classificateur linéaire $h_{\mathbf{w},b}$ dans l'espace des caractéristiques exprime un séparateur quadratique dans l'espace des exemples :

$$h_{\mathbf{w},b}(x_1, x_2) = \text{sgn}\left(w_1\, x_1^2 + w_2\, x_2^2 + w_3\, x_1 x_2 + w_4\, x_1 + w_5\, x_2 - b\right).$$

En particlulier si $\mathbf{w} = (1, 1, 0, 0, 0)$ et $b = 1$, la frontière de séparation entre les exemples classés positivement et ceux classés négativement est le cercle de rayon unité dans l'espace des exemples.

### 2.2.6 Noyaux

Le fait de représenter une frontière de décision complexe par un séparateur linéaire dans un espace de caractéristiques de très haute dimensionnalité peut engendrer un coût computationnel élevé. Fréquemment, l'étude des algorithmes d'apprentissage révèle que les seules opérations manipulant les vecteurs de caractéristiques consistent en des produits scalaires. La stratégie du noyau permet alors de substituer ces produits scalaires par une fonction $k(\mathbf{x}, \mathbf{x}')$ rapidement calculable. Le résultat de $k(\mathbf{x}, \mathbf{x}')$ doit être équivalent au produit scalaire entre $\boldsymbol{\phi}(\mathbf{x})$ et $\boldsymbol{\phi}(\mathbf{x}')$ dans un certain espace de caractéristiques $\mathcal{K}$. Ainsi, le *noyau $k$ est associé à $\boldsymbol{\phi}$* lorsque :

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}') = \sum_{i=1}^{n'} \phi_i(\mathbf{x})\phi_i(\mathbf{x}')\ . \tag{2.5}$$

Il est possible de représenter implicitement le vecteur $\mathbf{w}$ par une combinaison linéaire des vecteurs correspondant aux exemples d'entraînement $\{y_1\boldsymbol{\phi}(\mathbf{x}_1), y_2\boldsymbol{\phi}(\mathbf{x}_2), \ldots, y_m\boldsymbol{\phi}(\mathbf{x}_m)\}$ dans l'espace des caractéristiques. Le vecteur $\boldsymbol{\alpha} \in \mathbb{R}^m$ contient les poids associés à cette combinaison linéaire :

$$\mathbf{w} = \sum_{i=1}^{m} y_i\alpha_i\boldsymbol{\phi}(\mathbf{x}_i)\ . \tag{2.6}$$

Dans ce cas, on peut également recourir au noyau lors de la classification d'un exemple $\mathbf{x}$ :

$$\begin{aligned} h_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) - b) &= \text{sgn}\left(\sum_{i=1}^{m} y_i\alpha_i\boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}) - b\right) \\ &= \text{sgn}\left(\sum_{i=1}^{m} y_i\alpha_i k(\mathbf{x}, \mathbf{x}_i) - b\right). \end{aligned}$$

Dans ce texte, nous ferons référence à trois types de noyaux couramment utilisés. La figure 2.1 illustre le type de frontières de décision qu'ils permettent d'obtenir sur un exemple jouet d'ensemble de données.

– Le *noyau linéaire* est un simple produit scalaire :

$$k_{\text{LIN}}(\mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} \mathbf{x} \cdot \mathbf{x}' \,.$$

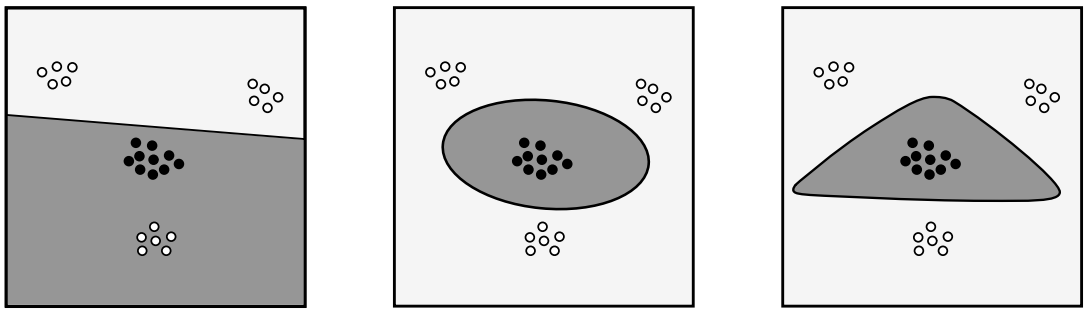
– Le *noyau polynomial* permet de représenter des frontières de décision par des polynômes de degré $p$ :

$$k_{\text{POL}}(\mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} (\mathbf{x} \cdot \mathbf{x}' + 1)^p \,.$$

– Le *noyau RBF* (de l'anglais *Radial Basis Function*) est un classificateur davantage complexe qui détermine la frontière de décision en effectuant une somme pondérée de fonctions gaussiennes centrées sur les exemples d'entraînement :

$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} \exp\left(-\gamma \left\|\mathbf{x} - \mathbf{x}'\right\|^2\right) . \tag{2.7}$$

Les classificateurs exprimés par un noyau linéaire sont équivalents aux séparateurs linéaires exprimés directement dans l'espace des exemples (sans utiliser la stratégie du noyau). Aussi, les variables $p$ et $\gamma$ des noyaux polynomial et RBF constituent des hyperparamètres des algorithmes d'apprentissage. Leur valeur modifie grandement le type de frontières de décision générées et, conséquemment, la qualité des classificateurs résultants.



(a) Linéaire (b) Polynomial de degré 2 (c) RBF

FIG. 2.1: Frontières de décision générées à l'aide de différents noyaux sur un ensemble de données à deux dimensions.

### 2.2.7 Classificateurs de Bayes et votes de majorité

Un *classificateur de Bayes* effectue un *vote de majorité* de plusieurs classificateurs distincts regroupés dans un ensemble $\mathcal{H}$. Chaque classificateur $h \in \mathcal{H}$ est un *classificateur de base* du vote de majorité. Il peut y avoir un nombre fini ou infini de classificateurs de base. De même, l'ensemble $\mathcal{H}$ peut être discret ou continu. Le vote de majorité est

pondéré par la distribution $Q$ sur l'ensemble $\mathcal{H}$. Autrement dit, un poids $Q(h)$ est associé au classificateur de base $h$ , déterminant ainsi son pouvoir de décision dans le vote de majorité.

Pour classifier un exemple $\mathbf{x}$, le classificateur de Bayes $B_Q$ considère le «vote» de chacun des classificateurs de base et retient l'opinion majoritaire (en accord avec la pondération $Q$). Ainsi, la prédiction $B_Q(\mathbf{x})$ est donnée par :

$$B_Q(\mathbf{x}) \stackrel{\text{def}}{=} \operatorname{sgn}\left[\mathop{\mathbf{E}}_{h\sim Q} h(\mathbf{x})\right].$$

Dans le cas où l'ensemble $\mathcal{H}$ des classificateurs de base est discret, cette définition revient à :

$$B_Q(\mathbf{x}) \stackrel{\text{def}}{=} \operatorname{sgn}\left[\sum_{h\in\mathcal{H}} Q(h)h(\mathbf{x})\right].$$

Le *risque de Bayes* d'une distribution $Q$ sur un ensemble de classificateur $\mathcal{H}$ est le risque du classificateur de Bayes associé à $Q$. Conformément aux définitions introduites à la section 2.2.1, l'expression du vrai risque de Bayes $R(B_Q)$ et l'expression du risque empirique de Bayes $R_S(B_Q)$ sont données par :

$$R(B_Q) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(B_Q(\mathbf{x}) \neq y)\,,$$

$$R_S(B_Q) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(B_Q(\mathbf{x}_i) \neq y_i)\,.$$

### 2.2.8 Classificateurs de Gibbs

À l'instar du classificateur de Bayes présenté à la section 2.2.7, le *classificateur de Gibbs* est défini par une distribution $Q$ sur un ensemble de classificateurs de base $\mathcal{H}$. Alors que le classificateur de bayes $B_Q$ est déterministe, le classificateur de Gibbs $G_Q$ est stochastique. Pour classifier un exemple $\mathbf{x}$, il pige aléatoirement un classificateur $h'$ selon la distribution $Q$. Il classifie ensuite l'exemple à l'aide de ce classificateur. Autrement dit, il retourne $h'(\mathbf{x})$.

Le *risque de Gibbs* d'une distribution $Q$ sur un ensemble de classificateur $\mathcal{H}$ est le risque du classificateur de Gibbs associé à $Q$. Le vrai risque de Gibbs et le risque

empirique de Gibbs sont définis ainsi :

$$R(G_Q) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} R(h) = \mathop{\mathbf{E}}_{h\sim Q} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y)\,, \tag{2.8}$$

$$R_S(G_Q) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} R_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathop{\mathbf{E}}_{h\sim Q} I(h(\mathbf{x}_i) \neq y_i)\,. \tag{2.9}$$

Dans le cas où l'ensemble $\mathcal{H}$ des classificateurs de base est discret, ces définitions reviennent à :

$$\begin{aligned} R(G_Q) &\stackrel{\text{def}}{=} \sum_{h\in\mathcal{H}} Q(h)R(h) = \sum_{h\in\mathcal{H}} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} Q(h)I(h(\mathbf{x}) \neq y)\,, \\ R_S(G_Q) &\stackrel{\text{def}}{=} \sum_{h\in\mathcal{H}} Q(h)R_S(h) = \frac{1}{m}\sum_{h\in\mathcal{H}}\sum_{i=1}^{m} Q(h)I(h(\mathbf{x}_i) \neq y_i)\,. \end{aligned}$$

Examinons maintenant la relation qu'entretiennent le risque de Bayes et le risque de Gibbs d'une même distribution de poids $Q$ sur un ensemble de classificateurs de base.

**Lemme 1.** *Pour toute distribution $Q$ sur un espace de classificateurs binaires $\mathcal{H}$ caractérisant un vote de majorité, on a :*

$$R(B_Q) \leq 2R(G_Q)\,.$$

*Démonstration.* Supposons d'abord que le classificateur de Bayes $B_Q$ classifie incorrectement l'exemple $\mathbf{z} = (\mathbf{x}, y)$. Donc, au moins la moitié du poids du vote de majorité est assignée à des classificateurs de base qui classifient incorrectement $\mathbf{z}$.

$$\begin{aligned} B_Q(\mathbf{x}) \neq y &\Rightarrow \Pr_{h\sim Q}(h(\mathbf{x}) \neq y) \geq 1/2 \\ &\Rightarrow \mathop{\mathbf{E}}_{h\sim Q} I\left(h(\mathbf{x}) \neq y\right) \geq 1/2 \end{aligned}$$

Notons par $R_{\{\mathbf{z}\}}(B_Q)$ et $R_{\{\mathbf{z}\}}(G_Q)$ le risque de $B_Q$ et de $G_Q$ sur l'exemple $\mathbf{z}$. Le risque de Gibbs pour $\mathbf{z}$ est d'au moins 50% :

$$R_{\{\mathbf{z}\}}(B_Q) = 1 \;\Rightarrow\; R_{\{\mathbf{z}\}}(G_Q) \geq 1/2\ .$$

Donc, peu importe si $B_Q$ classe l'exemple $\mathbf{z}$ correctement (auquel cas $R_{\{\mathbf{z}\}}(B_Q) = 0$) ou s'il classe l'exemple $\mathbf{z}$ incorrectement (auquel cas $R_{\{\mathbf{z}\}}(B_Q) = 1$), nous remarquons que :

$$R_{\{\mathbf{z}\}}(B_Q) \leq 2R_{\{\mathbf{z}\}}(G_Q)\,,$$

ce qui implique :

$$\mathop{\mathbf{E}}_{\mathbf{z}\sim D} R_{\{\mathbf{z}\}}(B_Q) \leq 2 \mathop{\mathbf{E}}_{\mathbf{z}\sim D} R_{\{\mathbf{z}\}}(G_Q)\,,$$

et conséquemment :

$$R(B_Q) \leq 2R(G_Q)\,.$$

□

Le lemme 1 indique que le risque de Bayes d'un vote de majorité est borné supérieurement par le double du risque de Gibbs associé au même vote de majorité. Cependant, il ne faut pas conclure que le classificateur de Gibbs est préférable au classificateur de Bayes. Bien qu'il existe des exemples où la borne est atteinte (c'est-à-dire que $R(B_Q) \approx 2R(G_Q)$ ), il est fréquent que le risque de Gibbs d'un vote de majorité soit supérieur à son risque de Bayes. Le tableau 2.1 illustre ces deux situations par un exemple simple. Puisque le recours à un classificateur par vote de majorité est davantage naturel que le recours à un classificateur stochastique, on favorise généralement l'utilisation du classificateur de Bayes. De plus, sur des problèmes réels, les classificateurs de Bayes sont la plupart du temps plus performant que les classificateurs de Gibbs possédant la même distribution de poids.

| $\underbrace{R(h_1) = 0}$ | $\underbrace{R(h_2) = 1}$ | | | |
|---|---|---|---|---|
| $\mathbf{Q(h_1)}$ | $\mathbf{Q(h_2)}$ | $\mathbf{R(B_Q)}$ | $\mathbf{R(G_Q)}$ | |
| $50\% - \epsilon$ | $50\% + \epsilon$ | 1 | $0.5 + \epsilon$ | $\Rightarrow R(B_Q) \approx 2R(G_Q)$ |
| $50\% + \epsilon$ | $50\% - \epsilon$ | 0 | $0.5 - \epsilon$ | $\Rightarrow R(B_Q) \ll R(G_Q)$ |

TAB. 2.1: Exemples de relations entre les risques de Bayes et de Gibbs d'un vote de majorité de deux classificateurs de base. Le classificateur $h_1$ classifie correctement tous les exemples et $h_2$ classifie incorectement tous les exemples.

### 2.2.9 Espace des votes de majorité

La section 2.2.7 définit un classificateur de Bayes comme un vote de majorité d'un ensemble $\mathcal{H}$ de classificateurs de base pondéré par une distribution de poids $Q$. La valeur de sortie du classificateur de Bayes est typiquement donnée par l'équation (2.2). Nous verrons maintenant qu'il est possible de représenter un classificateur de Bayes, ainsi que le classificateur de Gibbs associé, par un séparateur linéaire.

Tout d'abord, définissons l'*espace des votes de majorité* associé à un ensemble $\mathcal{H}$, qui correspond à l'ensemble de tous les classificateurs de Bayes qu'il est possible de construire à partir de $\mathcal{H}$ (autrement dit, toutes les distributions de poids $Q$ sur $\mathcal{H}$). Pour

représenter l'espace des votes de majorité, on a recours à un espace de caractéristiques (voir la section 2.2.5) où chaque fonction $\phi_i : \mathcal{X} \to \mathbb{R}$ correspond à un classificateur de base du vote de majorité (on a donc $\phi_i \in \mathcal{H}$). Ainsi, le vecteur de caractéristiques $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^{n'}$ contient la classe prédite pour l'exemple $\mathbf{x}$ par chacun des $n'$ classificateurs de base du vote de majorité.

Considérons maintenant un séparateur linéaire dans cet espace de caractéristiques, représenté par le vecteur $\mathbf{w} \in \mathbb{R}^{n'}$. Les éléments de ce vecteur correspondent aux poids associés aux $n'$ classificateurs de base du vote de majorité. Un séparateur linéaire $\mathbf{w}$ dans l'espace des votes de majorité s'exprime facilement en un vote de majorité (une distribution de poids $Q$ sur $\mathcal{H}$). En effet, le classificateur $\phi_i$ possède le poids $w_i$. Si on permet des valeurs de $w_i$ négatives, on considère que le poids $|w_i|$ est attribué au classificateur inverse $-\phi_i$ [4]. Pour obtenir une distribution de poids $Q$ normalisée, on divise chaque élément du vecteur $\mathbf{w}$ par $\sum_{i=1}^{n'} |w_i|$. Enfin, si on utilise une valeur de biais $b$ pour définir le séparateur linéaire, elle correspond dans le vote de majorité au poids affecté à un classificateur trivial dont la sortie est toujours négative.

À titre d'exemple, imaginons un classificateur de Bayes effectuant un vote de majorité de huit decision stumps de la forme $h_{i,t,d}$ (voir la section 2.2.2) sur un ensemble d'exemples à deux attributs $x_1, x_2$ :

$$\mathcal{H} = \{h_{1,5,1}, h_{1,5,-1}, h_{1,10,1}, h_{1,10,-1}, h_{2,5,1}, h_{2,5,-1}, h_{2,10,1}, h_{2,10,-1}\} .$$

Nous pouvons appliquer la transformation suivante à chaque exemple $\mathbf{x}$ de l'ensemble d'entraînement :

$$\boldsymbol{\phi}(\mathbf{x}) = (h_{1,5,1}(\mathbf{x}), h_{1,10,1}(\mathbf{x}), h_{2,5,1}(\mathbf{x}), h_{2,10,1}(\mathbf{x})) .$$

Remarquons que chaque élément du vecteur de caractéristiques $\boldsymbol{\phi}(\mathbf{x})$ représente deux classificateurs de base complémentaires. Par cette transformation, l'exemple $\mathbf{x}_a = (2, 7)$ est associé au vecteur de caractéristiques $\boldsymbol{\phi}(\mathbf{x}_a) = (-1, -1, +1, -1)$. De même, si le classificateur linéaire $h_{\mathbf{w}}$ (sans biais) est représenté par le vecteur $\mathbf{w} = (2, 0, -1, -1)$, il classifie négativement l'exemple $\mathbf{x}_a$ :

$$h_{\mathbf{w}}(\mathbf{x}_a) = \text{sgn}\left[\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_a)\right] = -1 .$$

[4] Dans ce cas, l'ensemble de classificateurs de base $\mathcal{H}$ doit être constitué de paires de classificateurs inverses.

## 2.3 Algorithmes d'apprentissage

Nous représentons un *algorithme d'apprentissage* par une fonction $A$ qui prend en entrée un ensemble d'entraînement $S$ et fournit en sortie un classificateur $h$ :

$$h = A(S)\,.$$

L'objectif de l'algorithme $A$ est de généraliser l'information contenue dans $S$ afin de construire un classificateur qui distingue un exemple positif d'un exemple négatif. Autrement dit, le classificateur $h$ doit prédire adéquatement la classe des exemples générés en accord avec la distribution de probabilité $D$.

Plusieurs algorithmes d'apprentissage requièrent en entrée d'autres paramètres que l'ensemble d'apprentissage. Ces valeurs, que l'on nomme *hyperparamètres*, doivent être spécifiées préalablement à l'exécution de l'algorithme. Elles conditionnent habituellement la capacité de généralisation du processus d'apprentissage.

Cette section présente deux algorithmes d'apprentissage communément utilisés. Ils serviront de point de comparaison pour évaluer les performances des algorithmes suggérés plus loin.

### 2.3.1 AdaBoost

Les algorithmes de type *Boosting* construisent des votes de majorité. Ils combinent des classificateurs obtenus d'un algorithme d'apprentissage faible (*weak learner*) en un classificateur de faible risque. Par algorithme d'apprentissage faible, on désigne un algorithme qui produit un classificateur dont le risque empirique est élevé, en autant qu'il soit inférieur à 50%.

*AdaBoost* [22, 23] est un algorithme de type Boosting largement utilisé. Il construit un vote de majorité de manière itérative. Au fil des itérations, il maintient une distribution de poids sur les exemples d'entraînement de telle sorte que les exemples mal classifiés voient leur poids augmenter et les exemples bien classifiés, leur poids diminuer. À chaque itération, l'algorithme d'apprentissage faible est entraîné avec l'ensemble d'exemples pondérés et le classificateur résultant est ajouté au vote de majorité. De cette manière, le nouveau classificateur tend à compenser les erreurs empiriques effectuées par les classificateurs sélectionnés précédemment.

L'algorithme 1 présente le pseudo-code d'AdaBoost. On désigne par $W$ la distribu-

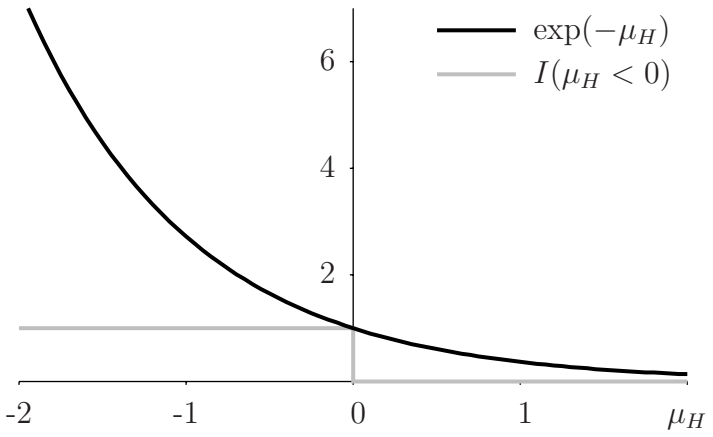



FIG. 2.2: Comparaison des valeurs du risque exponentiel et du vrai risque sur un exemple en fonction de la marge du vote de majorité $\mu_H(\mathbf{x}, y) = y \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$.

---

**Algorithme 1** AdaBoost$(S, T)$

---

**Entrée :** $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, l'ensemble d'entraînement.
**Entrée :** $T$, le nombre d'itérations.

Initialiser $W_1(i) \leftarrow 1/m$ pour $i = 1, \ldots, m$ .
**pour** $t = 1, \ldots, T$ **faire**
Entraîner l'algorithme d'apprentissage faible : $h_t \leftarrow A_{\text{weak}}(S, W_t)$ .
Calculer le risque empirique pondéré :

$$r_t \leftarrow \sum_{i=1}^{m} W_t(i)\, I(h_t(\mathbf{x}_i) \neq y_i)\,. \tag{2.10}$$

Calculer le poids du classificateur dans le vote de majorité :

$$\alpha_t \leftarrow \frac{1}{2} \ln\left[\frac{1 - r_t}{r_t}\right]. \tag{2.11}$$

Mettre à jour les poids des exemples ($Z_t$ est une constante de normalisation) :

$$W_{t+1}(i) \leftarrow \frac{W_t(i) \exp\left[-\alpha_t y_i h_t(\mathbf{x}_i)\right]}{Z_t}\ . \tag{2.12}$$

**fin pour**

**Sortie :** $H(\mathbf{x}) = \text{sgn}\left[\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right]$.

---

tion de poids sur les exemples et par $A_{\text{weak}}(S, W)$ l'algorithme d'apprentissage faible. Ce dernier retourne un classificateur à partir de l'ensemble $S$ et de la distribution $W$. Notons que le seul hyperparamètre nécessaire est le nombre d'itérations $T$ à effectuer. Le résultat de l'algorithme est généralement peu influencé par la valeur de $T$ choisie, du moment qu'elle soit suffisamment grande.

Il est montré dans [22] que AdaBoost minimise une borne supérieure sur le risque empirique du vote de majorité :

$$R_S(H) \leq \prod_{t=1}^{T} Z_t = \frac{1}{m} \sum_{i=1}^{m} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right).$$

On nomme cette fonction le *risque exponentiel*. La figure 2.2 donne un aperçu du risque exponentiel mesuré sur un exemple $\mathbf{x}$ donné.

Les decision stumps (voir la section 2.2.2) sont souvent les classificateurs de base utilisés avec l'algorithme AdaBoost. Dans ce cas, l'algorithme d'apprentissage faible $A_{\text{weak}}$ sélectionne à chaque itération le decision stump qui minimise le risque empirique pondéré (équation (2.10)).

### 2.3.2 Machine à vecteurs de support (SVM)

La *Machine à vecteurs de support* (*SVM*, de l'anglais *Support Vector Machine*) est un algorithme d'apprentissage qui produit un classificateur linéaire. Grâce à la stratégie des noyaux (voir la section 2.2.6), le SVM est en mesure de produire une frontière de décision complexe. Ses bonnes performances et sa rapidité d'exécution en font un algorithme très utilisé.

Examinons d'abord une première version de l'algorithme, le *SVM à marge rigide* [5], qui s'applique seulement aux ensembles d'entraînement linéairement séparables. Dans ce contexte, il existe différents classificateurs linéaires ayant un risque empirique nul. Le SVM à marge rigide trouve l'hyperplan $(\mathbf{w}, b)$ dont la marge géométrique est maximale :

$$\gamma \stackrel{\text{def}}{=} \max_{\mathbf{w},b} \left[\min_{i \in S} \frac{y_i[\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) - b]}{\|\mathbf{w}\|}\right].$$

On nomme *vecteurs de support* les exemples qui ont une marge d'exactement $\gamma$. Ces exemples définissent deux hyperplans, parallèles à la frontière de séparation, entre lesquels n'est situé aucun exemple (voir la figure 2.3) . Alors que plusieurs vecteurs de même direction définissent la frontière de séparation recherchée, le SVM choisit l'unique séparateur de marge maximale $(\mathbf{w}', b')$ tel que les vecteurs de support respectent l'équation

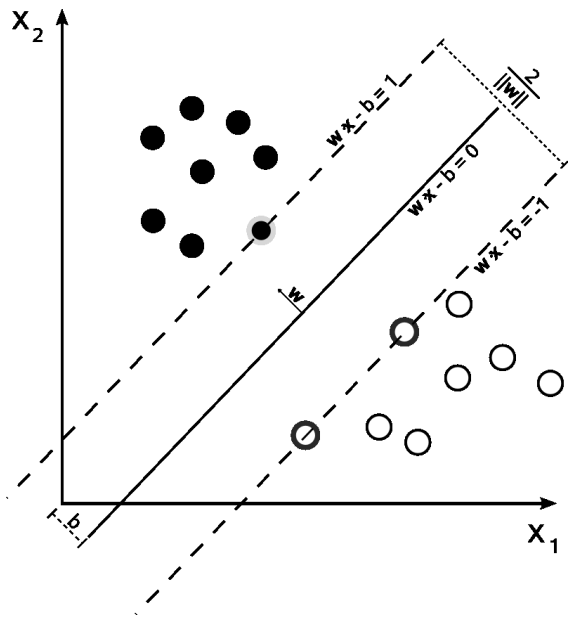



FIG. 2.3: Frontière de décision d'un SVM à marge rigide. Les trois exemples encerclés correspondent aux vecteurs de support. (Image : Wikipedia)

$|\mathbf{w}' \cdot \boldsymbol{\phi}(\mathbf{x}) - b'| = 1$. Autrement dit, la marge fonctionnelle des vecteurs de support vaut 1. Ce séparateur, que l'on nomme *hyperplan canonique*, possède une marge géométrique de $\|\mathbf{w}'\|^{-1}$ :

$$\gamma = \min_{i \in S} \frac{y_i[\mathbf{w}' \cdot \boldsymbol{\phi}(\mathbf{x}_i) - b']}{\|\mathbf{w}'\|} = \frac{1}{\|\mathbf{w}'\|} \ .$$

Le problème que solutionne le SVM à marge rigide se pose en un problème de programmation quadratique :

$$\begin{aligned} \text{Minimiser :} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{Sous les contraintes :} \quad & y_i[\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) - b] \geq 1 \quad \text{pour } i = 1, \ldots, m \end{aligned}$$

Il est possible de résoudre ce problème efficacement à l'aide de la méthode des multiplicateurs de Lagrange, tel que décrit dans [5].

Le *SVM à marge floue* [9] généralise le problème afin de permettre certaines erreurs de classification, ce qui permet d'utiliser l'algorithme sur des ensembles d'entraînement qui ne sont pas linéairement séparables. Pour ce faire, on associe une *variable d'écart* $\xi_i$ à chaque exemple d'entraînement $\mathbf{z}_i$. Cette variable attribue une pénalité aux exemples dont la marge est inférieure à $\|\mathbf{w}\|^{-1}$ selon la fonction de perte suivante, que l'on nomme le *hinge loss* (la figure 2.4 en donne un aperçu graphique) :

$$\xi_i = \max\left(0, 1 - y_i\left[\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) - b\right]\right) \ .$$

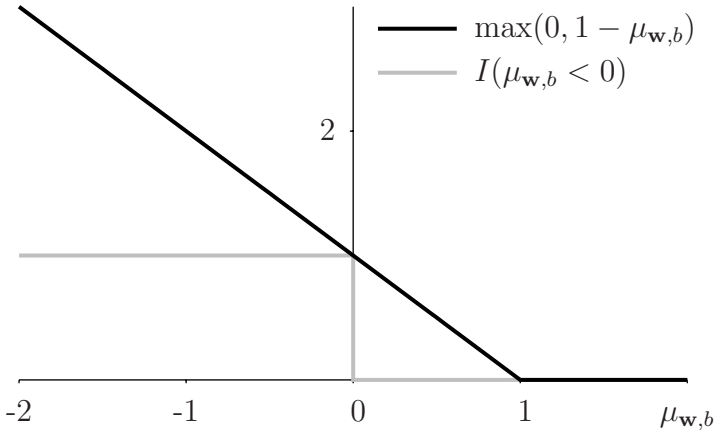



FIG. 2.4: Comparaison des valeurs du hinge loss et du vrai risque sur un exemple en fonction de la marge du plan séparateur $\mu_{\mathbf{w},b}(\mathbf{x}, y) = y\,[\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) - b]$.

La somme des variables d'écart $\sum_{i=1}^{m} \xi_i$ fournit une borne supérieure sur le risque empirique du classificateur. La valeur de chaque $\xi_i$ s'interprète ainsi :

- $\xi_i > 1$ : L'exemple $\mathbf{z}_i$ est mal classifié.
- $0 < \xi_i < 1$ : $\mathbf{z}_i$ est bien classifié, mais il est situé à l'intérieur de la marge du SVM.
- $\xi_i = 0$ : $\mathbf{z}_i$ est bien classifié et il est situé à l'extérieur de la marge du SVM.

Lors de l'apprentissage, le SVM à marge floue effectue un compromis entre la marge géométrique et la pénalité due aux variables d'écart. L'ampleur de ce compromis est dicté par un hyperparamètre de l'algorithme, le *paramètre de marge floue*, désigné par la constante $C$. L'objectif du SVM à marge floue est de minimiser la fonction suivante :

$$\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i \ . \tag{2.13}$$

Une petite valeur $C$ incite à tolérer les erreurs afin d'accentuer la marge. Lorsque $C$ est très grand ($C \to \infty$), l'algorithme se comporte comme un SVM à marge rigide et ne tolère aucune erreur. En pratique, on doit souvent exécuter l'algorithme avec une variété de valeurs de $C$ afin de trouver celle qui convient le mieux aux données étudiées.

Le problème que solutionne le SVM à marge floue s'énonce aussi sous la forme d'un problème quadratique :

$$\begin{array}{rll} \text{Minimiser :} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i & \\ \text{Sous les contraintes :} & y_i[\mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) - b] \geq 1 - \xi_i & \text{pour } i = 1, \ldots, m \\ & \xi_i \geq 0 & \text{pour } i = 1, \ldots, m \end{array}$$

Tel que démontré dans [9], il est important de remarquer qu'il s'agit d'un problème

d'optimisation convexe. Par conséquent, sa solution est unique. Plusieurs algorithmes, dont *SMO* [20], permettent de résoudre ce problème efficacement.

## 2.4 Sélection de modèle et estimation du risque

Confronté à un problème d'apprentissage automatique, on tente habituellement de le résoudre à l'aide de plusieurs algorithmes d'apprentissage. Aussi, on exécute à plusieurs reprises un même algorithme avec une variété d'hyperparamètres. On obtient alors plusieurs classificateurs différents. L'étape cruciale de la *sélection de modèle* consiste à sélectionner parmi cet ensemble de classificateurs celui qu'on croit le meilleur. Idéalement, nous désirerions estimer le vrai risque des classificateurs, ce qui permettrait d'en connaître la probabilité de succès. Cependant, on se contente parfois de sélectionner un classificateur que l'on préfère aux autres selon des critères préétablis.

### 2.4.1 Validation croisée

La *validation croisée k-fois* (ou *k-fold cross validation*) est une méthode couramment utilisée pour sélectionner les hyperparamètres d'un algorithme d'apprentissage. Elle consiste à partitionner l'ensemble d'entraînement $S$ en $k$ sous-ensembles disjoints $T_1, T_2, \ldots, T_k$ (idéalement de tailles identiques). Pour chaque combinaison d'hyperparamètres à évaluer, on exécute l'algorithme d'apprentissage $k$ fois. À chaque «fois», un sous-ensemble distinct $T_i$ est réservé pour tester le classificateur $h_i$ entraîné avec les autres données :

$$h_i = A(S \setminus T_i)\,.$$

On calcule ensuite le *risque de validation croisée* associé à ces hyperparamètres, c'est-à-dire la moyenne des risques calculés pour chaque sous-ensemble test :

$$R_{CV} = \frac{1}{k}\sum_{i=1}^{k} R_{T_i}(h_i)\,.$$

Parmi toutes les combinaisons d'hyperparamètres évaluées, on retient celle possédant le risque de validation croisée minimal. Ces hyperparamètres sont utilisés pour entraîner l'algorithme sur la totalité de l'ensemble d'entraînement $S$, ce qui fournit le classificateur final.

Cette méthode tente d'utiliser au maximum les données disponibles pour sélectionner les hyperparamètres adéquatement. Elle se révèle de grande utilité en pratique mais elle

possède deux désavantages. Premièrement, son utilisation ralentit considérablement le processus d'apprentissage[5]. Deuxièmement, malgré que l'on suppose qu'une combinaison d'hyperparamètres possédant un faible risque de validation croisée permet d'obtenir un bon classificateur lorsque l'algorithme d'apprentissage est executé sur toutes les données d'entraînement, la validation croisée ne fournit aucune garantie sur le vrai risque du classificateur final.

### 2.4.2 Bornes sur l'ensemble test

La méthode privilégiée pour estimer le vrai risque d'un classificateur $h$ consiste à le tester sur des exemples qui n'ont pas servi au processus d'entraînement. Pour ce faire, on réserve une portion des données étiquetées à notre disposition pour former un ensemble test $T$. Le risque $R_T(h)$ du classificateur $h$ sur l'ensemble test s'avère une estimation sans biais du vrai risque $R(h)$. En effet, comme on considère que chaque exemple de $T$ est une réalisation *iid* de la variable aléatoire $\mathbf{Z}$, la probabilité d'effectuer une erreur de classification sur un exemple de test est exactement $R(h)$. Ainsi, la probabilité d'effectuer $k = |T|R_T(h)$ erreurs de classification sur l'ensemble test est donnée par la loi binomiale :

$$\Pr_{\mathbf{Z}^{|T|}}\left(R_{\mathbf{Z}^{|T|}}(h) = \frac{k}{|T|}\right) = \binom{|T|}{k} R(h)^k (1 - R(h))^{|T|-k} . \tag{2.14}$$

Nous présentons ici une borne sur l'ensemble test suggérée par John Langford [15]. Il s'agit d'une borne exacte, en ce sens qu'elle est la plus serrée que l'on puisse obtenir en se basant uniquement sur le risque mesuré sur un ensemble test.

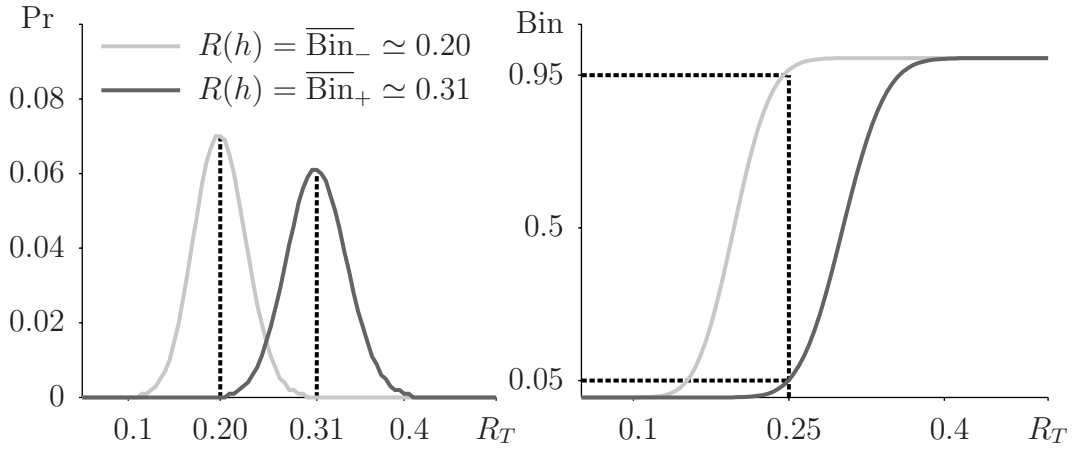
Définissons d'abord la probabilité qu'un classificateur de vrai risque $r$ fasse jusqu'à $k$ erreurs parmi $n$ exemples de test par la queue de la binomiale :

$$\mathrm{Bin}(n, k, r) \stackrel{\text{def}}{=} \sum_{i=0}^{k} \binom{n}{i} r^i (1 - r)^{n-i} . \tag{2.15}$$

Nous nous intéressons à l'inverse de la queue de la binomiale, $\overline{\mathrm{Bin}}_+(n, k, \delta)$, c'est-à-dire la plus grande valeur de risque $r$ telle que la probabilité est au moins $\delta$ de faire jusqu'à $k$ erreurs parmi $n$ exemples :

$$\overline{\mathrm{Bin}}_+(n, k, \delta) \stackrel{\text{def}}{=} \max\{r : \mathrm{Bin}(n, k, r) \geq \delta\} . \tag{2.16}$$

[5] Comparativement à un apprentissage simple sur la totalité des $m$ exemples d'entraînement, le temps d'exécution requis est multiplié par $k$ si on suppose que le temps d'exécution de l'algorithme utilisé est linéaire en $m$ et, plus généralement, par $1 + k(1 - \frac{1}{k})^n$ si le temps d'exécution est en $O(m^n)$.




(a) Loi binomiale (équation (2.14)) (b) Queue de la binomiale (équation (2.15))

FIG. 2.5: Illustration d'un calcul de bornes sur un ensemble test de cardinalité $|T| = 200$, sur lequel le classificateur $h$ fait 50 erreurs ($R_T(h) = 0.25$). Avec confiance $1 - \delta = 90\%$, le vrai risque $R(h)$ du classificateur se situe entre la borne inférieure $\overline{\text{Bin}}_-(200, 50, \frac{1}{20}) \simeq 0.20$ et la borne supérieure $\overline{\text{Bin}}_+(200, 50, \frac{1}{20}) \simeq 0.31$. Les figures ci-haut présentent les distributions de probabilité associées aux extrémités de cet intervalle.

Comme l'indique le théorème suivant, l'inverse de la queue de la binomiale nous donne une borne supérieure sur le vrai risque à partir du risque observé sur l'ensemble test.

**Théorème 2** (Langford 2005 [15])**.** *Pour tout classificateur $h$ de risque $R(h)$ et pour tout $\delta \in ]0, 1]$, nous avons*

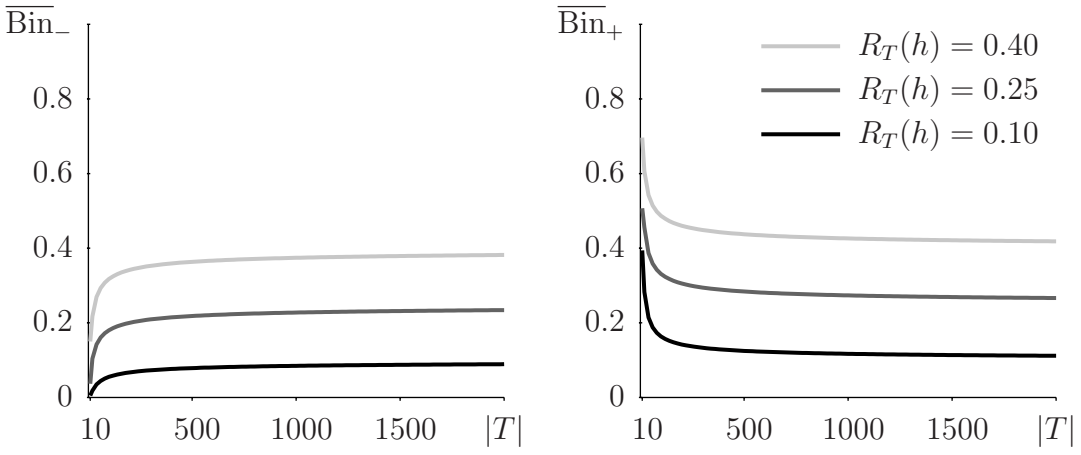
$$\Pr_{\mathbf{Z}^n}\left(R(h) \leq \overline{\text{Bin}}_+\big(n, nR_{\mathbf{Z}^n}(h), \delta\big)\right) \geq 1 - \delta \ .$$

De manière similaire, le théorème 3 ci-dessous donne une borne inférieure sur le vrai risque, où $\overline{\text{Bin}}_-(n, k, \delta)$ est la plus petite valeur de $r$ telle que la probabilité est au plus $1 - \delta$ de faire jusqu'à $k$ erreurs parmi $n$ exemples :

$$\overline{\text{Bin}}_-(n, k, \delta) \stackrel{\text{def}}{=} \min\big\{r : \text{Bin}(n, k, r) \leq 1 - \delta\big\} \ .$$

**Théorème 3** (Langford 2005 [15])**.** *Pour tout classificateur $h$ de risque $R(h)$ et pour tout $\delta \in ]0, 1]$, nous avons*

$$\Pr_{\mathbf{Z}^n}\left(R(h) \geq \overline{\text{Bin}}_-\big(n, nR_{\mathbf{Z}^n}(h), \delta\big)\right) \geq 1 - \delta \ .$$




(a) $\overline{\text{Bin}}_-\,(|T|, |T|R_T(h), \delta)$ selon $|T|$ (b) $\overline{\text{Bin}}_+\,(|T|, |T|R_T(h), \delta)$ selon $|T|$

FIG. 2.6: Valeurs des borne inférieures (2.6a) et supérieure (2.6b) sur l'ensemble test en fonction de $|T|$ (avec $\delta = \frac{1}{20}$) pour $R_T(h) \in \{0.10, 0.25, 0.40\}$.

En combinant les théorèmes 2 et 3 et en utilisant la borne de l'union[6], on obtient le corollaire 4 suivant qui permet d'établir un intervalle de confiance sur le vrai risque d'un classificateur. La figure 2.5 illustre par un exemple le calcul de cet intervalle de confiance.

**Corollaire 4.** *Pour tout classificateur $h$ de risque $R(h)$ et pour tout $\delta \in ]0, 1]$, nous avons*

$$\Pr_{\mathbf{Z}^n}\left(\overline{\text{Bin}}_-\big(n, nR_{\mathbf{Z}^n}(h), \delta/2\big) \leq R(h) \leq \overline{\text{Bin}}_+\big(n, nR_{\mathbf{Z}^n}(h), \delta/2\big)\right) \geq 1 - \delta\ .$$

Les intervalles de confiance obtenus par le calcul des bornes inférieures et supérieures sont utilisés pour comparer la performance de classificateurs entre eux. En particulier, nous dirons que le classificateur $h_1$ est *significativement meilleur* que le classificateur $h_2$ avec une confiance $1 - \delta$ lorsque :

$$\overline{\text{Bin}}_+(|T|, |T|R_T(h_1), \delta/2) < \overline{\text{Bin}}_-(|T|, |T|R_T(h_2), \delta/2)\,.$$

Les bornes obtenues sur l'ensemble test sont davantage serrées que le nombre $|T|$ d'exemples test est élevé (tel qu'illustré par la figure 2.6). Toutefois, comme chaque exemple étiqueté dans l'ensemble test est exclu du processus d'apprentissage de l'algorithme, l'augmentation de la taille de l'ensemble test se fait au détriment de la taille de

[6]Soit $A$ et $B$ deux évènements aléatoires. La borne de l'union stipule que :
$\Pr(A \cup B) \leq \Pr(A) + \Pr(B)$.

l'ensemble d'entraînement. Un algorithme entraîné avec moins d'exemples produira potentiellement un moins bon classificateur. Face à un problème d'apprentissage où il est coûteux (en temps ou en argent) d'étiqueter un grand nombre d'exemples, il y a donc un compromis à réaliser entre l'information utilisée pour l'apprentissage (taille de l'ensemble d'entraînement) et la précision de la garantie sur le vrai risque du classificateur obtenu (taille de l'ensemble test).

### 2.4.3 Bornes sur l'ensemble d'entraînement

Certaines bornes permettent d'obtenir une garantie sur le vrai risque d'un classificateur sans avoir recours à un ensemble test, mais en analysant plutôt ses performances sur l'ensemble d'entraînement. Comme le risque empirique à lui seul ne fournit aucune information sur le vrai risque, ces bornes doivent dépendre de la nature du classificateur $h = A(S)$ et de l'ensemble d'entraînement $S$. Une fonction $B(S, h, \delta)$ calculant, avec probabilité $1 - \delta$, une borne supérieure sur le vrai risque du classificateur $h$ respecte l'inégalité suivante :

$$\Pr_{\mathbf{Z}^m}\left(R(A(\mathbf{Z}^m)) \leq B(\mathbf{Z}^m, A(\mathbf{Z}^m), \delta)\right) \geq 1 - \delta \ .$$

Cela contraste avec la borne sur l'ensemble test qui dépend seulement du paramètre $\delta$, du risque mesuré sur l'ensemble test et de la cardinalité de ce dernier (voir l'équation (2.16)).

En pratique, les bornes supérieures sur l'ensemble d'entraînement fournissent généralement une garantie moins précise que la borne sur l'ensemble test présentée à la section 2.4.2. Typiquement, pour un ensemble d'entraînement $S$, un ensemble test $T$ et un classificateur $h_1 = A(S)$, la garantie sur le risque de $h_1$ obtenue par la borne de l'ensemble test $\overline{\text{Bin}}_+(|T|, |T|R_T(h_1), \delta)$ sera largement meilleure que celle obtenue par une borne de l'ensemble d'entraînement $B(S, h_1, \delta)$. Cependant, plusieurs raisons motivent l'élaboration de bornes sur l'ensemble d'entraînement les plus précises qui soient.

Premièrement, lorsqu'une borne sur l'ensemble d'entraînement est suffisamment serrée pour fournir une garantie que l'on juge «satisfaisante», il est possible d'utiliser tous les exemples étiquetés à notre disposition pour le processus d'apprentissage (sans se réserver d'ensemble test). Un classificateur $h_2 = A(S \cup T)$ obtenu ainsi sera possiblement meilleur que le classificateur $h_1$, puisqu'entraîné avec plus d'informations, tout en possèdant une garantie $B(S \cup T, h_2, \delta)$.

Deuxièment, une borne sur l'ensemble d'entraînement peut servir à effectuer une

sélection de modèle. On peut ainsi diminuer le temps requis par le processus d'apprentissage en remplaçant la méthode de la validation croisée pour sélectionner les hyperparamètres d'un algorithme. Notons que pour effectuer une sélection de modèle à l'aide d'une borne sur l'ensemble d'entraînement, le comportement de la borne doit refléter le comportement du vrai risque. Autrement dit, parmi un ensemble de classificateurs, il est important que celui associé à la plus faible valeur de borne puisse vraissemblablement être de ceux qui possèdent les meilleurs vrais risques.

Finalement, l'étude des diverses bornes sur l'ensemble d'entraînement permet d'énoncer des critères que doivent respecter un bon classificateur. Ainsi, elles contribuent à la compréhension du problème d'apprentissage. Cela permet à la fois d'expliquer la performance de certains algorithmes existants et de découvrir des avenues pour en élaborer de nouveaux. Voici une énumération de quelques concepts utilisés par certaines bornes sur l'ensemble d'entraînement et une explication intuitive de l'information qu'ils permettent d'obtenir sur la qualité des classificateurs :

– La *complexité* d'un classificateur est évaluée en fonction de la quantité d'information nécessaire pour le représenter. Un classificateur linéaire, exprimable seulement par le vecteur $\mathbf{w}$ et la valeur de biais $b$ (voir l'équation (2.2)), est de complexité relativement faible. Le risque empirique d'un tel classificateur est vraissemblablement un bon indicateur de son vrai risque, puisque sa simplicité le contraint à généraliser le phénomène observé. À l'inverse, un classificateur de complexité élevée peut être le résultat du surapprentissage des données d'entraînenement. C'est le cas du classificateur $h_{\text{SA}}$ définit par l'équation (2.1). Ce classificateur est très complexe, puisqu'il est représenté à l'aide de tous les exemples positifs de l'ensemble d'entraînement. Il s'agit aussi d'un exemple de classificateur possédant un risque empirique qui ne reflète pas son vrai risque. Ainsi, le critère de la complexité suggère de favoriser un classificateur simple, qui généralise habituellement mieux le phénomène observé qu'un classificateur complexe, même s'il possède un risque empirique légèrement plus élevé. Cette idée est exploitée par la borne du rasoir de Occam [4].

– La taille de la *compression des données* [10] est évaluée principalement par le nombre d'exemples d'entraînement nécessaires pour caractériser le classificateur (par exemple, le nombre de vecteurs de support d'un SVM). Cette mesure est reliée à la complexité d'un classificateur, puisqu'un classificateur qui s'exprime à l'aide de peu d'exemples est un classificateur simple.

– La connaissance d'un phénomène *a priori* permet de juger la vraissemblance d'un classificateur. En pratique, cette connaissance peut être représentée par une dis-

tribution sur une famille de classificateurs, où on accorde davantage de poids aux classificateurs en lesquels nous avons confiance. Les informations *a priori* ne reposent pas sur l'observation des exemples d'entraînement, alors que le classificateur est obtenu par l'apprentissage de ces exemples. Il s'agit d'un critère qui favorise les classificateurs cohérents avec la compréhension que nous avons du phénomène. Ce concept est essentiel à la théorie PAC-Bayes présentée au chapitre 3.

– Dans la version de la borne PAC-Bayes présentée par Lacasse et al. [13], on s'attarde à la *covariance des erreurs* entre les paires de classificateurs de base d'un vote de majorité. De deux votes de majorité possédant un faible risque empirique, on préfère celui dont les erreurs des classificateurs de base ont lieu sur des exemples différents.

– Pour un séparateur linéaire, les *marges de séparation* des exemples d'entraînement informent sur la robustesse du classificateur. Les classificateurs à grande marges sont moins susceptibles de faire des erreurs sur de nouveaux exemples. Inversement, si un exemple se situe près de la frontière de séparation, un faible déplacement de la position de l'exemple dans l'espace risque de modifier sa classification. Cette idée est exploitée dans la borne PAC-Bayes appliquée au classificateurs linéaires [14, 15] discutée dans les prochains chapitres.

# Chapitre 3

# Éléments de la théorie PAC-Bayes

> La nature a une perfection à elle, surprenante, et qui résulte d'une addition de limites. La nature est parfaite parce qu'elle n'est pas infinie. Si on comprend les limites, on comprend comment le mécanisme fonctionne.
>
> Océan Mer (roman d'Alessandro Baricco, traduction de Françoise Brun)

La recherche de bornes sur le risque des classificateurs calculées à partir de l'ensemble d'entraînement s'est historiquement divisée en deux grandes approches. D'une part, l'approche fréquentiste (ou *PAC*, pour *probablement approximativement correct*) dérive ses résultats de la supposition que les données observées sont générées de manière «iid» selon une distribution de probabilité, tel que nous l'avons présumé depuis la section 2.1. D'autre part, l'approche bayesienne consiste à évaluer la probabilité qu'un classificateur représente bien le phénomène observé en se basant sur les connaissances que nous en avons avant d'utiliser les exemples d'entraînement pour l'apprentissage. Cette information est typiquement représentée par une distribution de probabilité *a priori* sur l'ensemble des classificateurs possibles.

Ces deux avenues ont été incorporées dans la théorie PAC-Bayes par David McAllester [18]. Cette théorie permet d'obtenir des garanties pour le classificateur de Gibbs associé à un vote de majorité à partir de ses performances sur l'ensemble d'entraînement. Les bornes supérieures sur le risque de Gibbs d'un classificateur se convertissent directement en bornes supérieures sur le risque de Bayes, car le risque de Bayes d'un classificateur est inférieur ou égal au double de son risque de Gibbs (lemme 1).

## 3.1 Quelques bornes et théorèmes PAC-Bayes

Étant donné un classificateur de Gibbs $G_Q$ caractérisé par une distribution $Q$ sur un espace $\mathcal{H}$ de classificateurs de base, la théorie PAC-Bayes permet de borner le risque de Gibbs $R(G_Q)$ en se basant sur les informations suivantes :

- Le risque empirique de Gibbs $R_S(G_Q)$, mesuré sur l'ensemble d'entraînement ;
- Le nombre $m$ d'exemples d'entraînement ;
- La distribution $P$ sur l'espace $\mathcal{H}$, représentant les connaissances que nous avons du problème *a priori* (nommée le *prior*) ;
- La distribution $Q$ sur l'espace $\mathcal{H}$, représentant le classificateur obtenu par l'apprentissage (nommée le *posterior*). Autrement dit, il s'agit des connaissances que nous avons du problème *a posteriori* ;
- Le paramètre de confiance $\delta$, qui détermine avec quelle probabilité la borne calculée sera invalide.

Le choix de la distribution $P$ ne doit pas être influencé par les exemples de l'ensemble d'entraînement $S$. Ceux-ci sont pris en considération par l'algorithme d'apprentissage pour choisir la distribution $Q$. En pratique, il est nécessaire de déterminer la distribution $P$ préalablement à l'exécution de l'algorithme d'apprentissage. La précision de la borne obtenue est grandement affectée par la justesse de ce choix. Dans le théorème PAC-Bayes présenté à la section 3.1.1, on mesure la *divergence de Kullback-Leibler* entre les distributions $P$ et $Q$ :

$$\mathrm{KL}(Q\|P) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} \ln \frac{Q(h)}{P(h)} \,. \tag{3.1}$$

Cette fonction est nulle lorsque $Q \equiv P$ et sa valeur croît lorsque la distribution $Q$ «s'éloigne» de la distribution $P$. Dans le cas d'une distribution discrète, elle vaut $\ln \frac{1}{P(h_1)}$ lorsque tout le poids est concentré sur un seul classificateur ($Q(h_1) = 1$). La divergence de Kullback-Leibler quantifie donc la similarité entre les connaissances que nous avons d'un problème d'apprentissage *a priori* et *a posteriori*.

### 3.1.1 Théorème PAC-Bayes général

Dans la littérature, le théorème PAC-Bayes se décline en plusieurs versions, dont chacune possède sa propre démonstration. On les doit notamment aux travaux de McAllester [18], Langford et Seeger [14] et Catoni [8]. Toutefois, les différentes formulations du théorème PAC-Bayes sont semblables. Cette section introduit une formulation générale du théorème PAC-Bayes, à partir de laquelle il est possible de dériver simplement plusieurs versions connues du théorème.

Afin de démontrer le théorème PAC-Bayes général, nous évoquons l'inégalité de Markov et l'inégalité de Jensen. Nous nous contentons ici de formuler ces résultats mathématiques. Leur démonstration est présenté dans [21].

**Lemme 5** (Inégalité de Markov)**.** *Pour toute variable aléatoire non négative $X$ dont l'espérance est $\mu$ et pour tout $t \in \mathbb{R}^+$, on a :*

$$\Pr\ (X > t\mu) < \frac{1}{t}\ .$$

Nous dirons qu'une fonction $f : \mathcal{X} \to \mathbb{R}$ est convexe dans un domaine $\mathcal{X} \subseteq \mathbb{R}^n$ lorsque $\forall x_1, x_2 \in \mathcal{X}$ et $\forall \alpha \in [0,1]$ :

$$\alpha f(x_1) + (1-\alpha) f(x_2) \geq f(\alpha x_1 + (1-\alpha) x_2)\ .$$

Notons que cette définition implique que $\mathcal{X}$ doit être un ensemble convexe[1].

**Lemme 6** (Inégalité de Jensen)**.** *Soit une fonction $f$ d'une variable aléatoire $X$ distribuée selon $P$. Si $f$ est convexe dans le domaine $\mathcal{X} \subseteq \mathbb{R}^n$ où $P(x)$ est non nul, alors on a :*

$$\mathbf{E}\ f(X)\ \geq\ f\left(\mathbf{E}\ X\right).$$

Nous tirons aussi profit de la propriété des distributions de probabilité selon laquelle, pour toute fonction $f : \mathcal{H} \to \mathbb{R}$ et pour toutes distributions $P$ et $Q$ sur $\mathcal{H}$ où le support de $Q$ est inclus dans le support de $P$, on a :

$$\mathop{\mathbf{E}}_{h\sim P} f(h)\ =\ \mathop{\mathbf{E}}_{h\sim Q} \frac{P(h)}{Q(h)} f(h)\,. \tag{3.2}$$

Nous possédons maintenant tous les outils nécessaires à la compréhension du théorème PAC-Bayes général, qui s'énonce ainsi :

**Théorème 7.** *Pour toute distribution $D$, pour tout ensemble $\mathcal{H}$ de classificateurs, pour toute distribution $P$ sur $\mathcal{H}$, pour tout $\delta \in\ ]0,1]$ et pour toute fonction convexe $\mathcal{D} : [0,1] \times [0,1] \to \mathbb{R}$, on a :*

$$\Pr_{S\sim D^m}\left(\begin{array}{l}\forall\, Q \text{ sur } \mathcal{H}: \\ \mathcal{D}(R_S(G_Q), R(G_Q)) \leq \frac{1}{m}\left[\mathrm{KL}(Q\|P) + \ln\left(\frac{1}{\delta}\mathop{\mathbf{E}}_{S\sim D^m}\mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{D}(R_S(h),R(h))}\right)\right]\end{array}\right) \geq 1-\delta\ ,$$

où $\mathrm{KL}(Q\|P)$ est la divergence de Kullback-Leibler donnée par l'équation (3.1).

[1] On dit que l'ensemble $\mathcal{X}$ est convexe lorsque $x_1, x_2 \in \mathcal{X} \Rightarrow \alpha x_1 + (1-\alpha) x_2 \in \mathcal{X} \quad \forall\, \alpha \in [0,1]$.

*Démonstration.* Comme l'expression $\underset{h\sim P}{\mathbf{E}}\, e^{m\mathcal{D}(R_S(h),R(h))}$ est une variable aléatoire non négative, l'inégalité de Markov (lemme 5) donne :

$$\Pr_{S\sim D^m}\left(\underset{h\sim P}{\mathbf{E}}\, e^{m\mathcal{D}(R_S(h),R(h))} \leq \frac{1}{\delta}\underset{S\sim D^m}{\mathbf{E}}\,\underset{h\sim P}{\mathbf{E}}\, e^{m\mathcal{D}(R_S(h),R(h))}\right) \geq 1-\delta\ .$$

Puisque la fonction $\ln(\cdot)$ est croissante, on peut écrire :

$$\Pr_{S\sim D^m}\left(\ln\left[\underset{h\sim P}{\mathbf{E}}\, e^{m\mathcal{D}(R_S(h),R(h))}\right] \leq \ln\left[\frac{1}{\delta}\underset{S\sim D^m}{\mathbf{E}}\,\underset{h\sim P}{\mathbf{E}}\, e^{m\mathcal{D}(R_S(h),R(h))}\right]\right) \geq 1-\delta\ .$$

Nous transformons maintenant l'espérance sur $P$ en espérance sur $Q$ à l'aide de l'équation (3.2) :

$$\Pr_{S\sim D^m}\left(\forall Q : \ln\left[\underset{h\sim Q}{\mathbf{E}}\,\frac{P(h)}{Q(h)}e^{m\mathcal{D}(R_S(h),R(h))}\right] \leq \ln\left[\frac{1}{\delta}\underset{S\sim D^m}{\mathbf{E}}\,\underset{h\sim P}{\mathbf{E}}\, e^{m\mathcal{D}(R_S(h),R(h))}\right]\right) \geq 1-\delta\ .$$

Par une double application de l'inégalité de Jensen (lemme 6), d'abord sur la fonction concave $\ln(\cdot)$ puis sur la fonction convexe $\mathcal{D}(\cdot,\cdot)$, nous obtenons :

$$\begin{aligned}
\ln\left[\underset{h\sim Q}{\mathbf{E}}\,\frac{P(h)}{Q(h)}e^{m\mathcal{D}(R_S(h),R(h))}\right] &\geq \underset{h\sim Q}{\mathbf{E}}\,\ln\left[\frac{P(h)}{Q(h)}e^{m\mathcal{D}(R_S(h),R(h))}\right]\\
&= \underset{h\sim Q}{\mathbf{E}}\,\ln\left[\frac{P(h)}{Q(h)}\right] + \underset{h\sim Q}{\mathbf{E}}\,\ln\left[e^{m\mathcal{D}(R_S(h),R(h))}\right]\\
&= -\mathrm{KL}(Q\|P) + m\underset{h\sim Q}{\mathbf{E}}\,\mathcal{D}(R_S(h),R(h))\\
&\geq -\mathrm{KL}(Q\|P) + m\mathcal{D}(\underset{h\sim Q}{\mathbf{E}}\,R_S(h),\underset{h\sim Q}{\mathbf{E}}\,R(h))\\
&= -\mathrm{KL}(Q\|P) + m\mathcal{D}(R_S(G_Q),R(G_Q))\ .
\end{aligned}$$

La dernière étape ci-haut découle de la définition du risque de Gibbs (équations (2.8) et (2.9)). Nous obtenons donc :

$$\Pr_{S\sim D^m}\left(\forall Q : m\mathcal{D}(R_S(G_Q),R(G_Q)) - \mathrm{KL}(Q\|P) \leq \ln\left[\frac{1}{\delta}\underset{S\sim D^m}{\mathbf{E}}\,\underset{h\sim P}{\mathbf{E}}\, e^{m\mathcal{D}(R_S(h),R(h))}\right]\right) \geq 1-\delta\ .$$

À partir de ce point, une simple réorganisation des termes permet d'obtenir le théorème. □

En insérant dans l'expression du théorème PAC-Bayes général une fonction convexe $\mathcal{D} : [0,1]\times[0,1]\to\mathbb{R}$ adéquate, il est possible d'obtenir directement plusieurs bornes PAC-Bayes existantes. C'est le cas des deux bornes discutées aux sections suivantes, que l'on présente comme des corollaires du théorème 7. Il serait aussi possible d'utiliser ce théorème général comme un outil pour dériver de nouvelles bornes PAC-Bayes, mais cela va au-delà du sujet abordé par ce mémoire.

### 3.1.2 Borne PAC-Bayes Langford-Seeger

La première version de la borne PAC-Bayes présentée a d'abord été formulée par John Langford et Matthias Seeger [15, 24]. Nous déduisons ici leur résultat du théorème PAC-Bayes général. Pour ce faire, nous choisissons la fonction $\mathcal{D}$ du théorème 7 comme étant la divergence de Kullback-Leibler entre deux distributions de Bernoulli de probabilité de succès $p$ et $q$ :

$$\mathrm{kl}(q\|p) \stackrel{\text{def}}{=} q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p} \,. \tag{3.3}$$

**Corollaire 8.** *Pour toute distribution $D$, pour tout ensemble $\mathcal{H}$ de classificateurs, pour toute distribution $P$ sur $\mathcal{H}$ et pour tout $\delta \in ]0,1]$, on a :*

$$\Pr_{S\sim D^m}\left(\forall Q \text{ sur } \mathcal{H}\colon \mathrm{kl}\Big(R_S(G_Q)\Big\|R(G_Q)\Big) \leq \frac{\mathrm{KL}(Q\|P) + \ln\frac{\xi(m)}{\delta}}{m}\right) \geq 1-\delta \,,$$

où les divergences de Kullback-Leibler $\mathrm{kl}(Q\|P)$ et $\mathrm{KL}(Q\|P)$ sont données respectivement par les équations (3.3) et (3.1) et où :

$$\xi(m) \stackrel{\text{def}}{=} \sum_{k=0}^{m} \binom{m}{k} \left(\frac{k}{m}\right)^k \left(1-\frac{k}{m}\right)^{m-k} . \tag{3.4}$$

*Démonstration.* Le corollaire est obtenu par le théorème 7 en posant $\mathcal{D}(q,p) = \mathrm{kl}(q\|p)$. En effet :

$$\begin{aligned}
\mathop{\mathbf{E}}_{S\sim D^m}\mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{D}(R_S(h),R(h))} &= \mathop{\mathbf{E}}_{S\sim D^m}\mathop{\mathbf{E}}_{h\sim P} \exp\left[mR_S(h)\ln\frac{R_S(h)}{R(h)} + m\,(1-R_S(h))\ln\frac{1-R_S(h)}{1-R(h)}\right] \\
&= \mathop{\mathbf{E}}_{h\sim P}\mathop{\mathbf{E}}_{S\sim D^m} \left(\frac{R_S(h)}{R(h)}\right)^{mR_S(h)} \left(\frac{1-R_S(h)}{1-R(h)}\right)^{m(1-R_S(h))} \\
&= \mathop{\mathbf{E}}_{h\sim P} \sum_{k=0}^{m} \Pr_{S\sim D^m}\left(R_S(h)=\frac{k}{m}\right) \left(\frac{\frac{k}{m}}{R(h)}\right)^k \left(\frac{1-\frac{k}{m}}{1-R(h)}\right)^{m-k} .
\end{aligned}$$

Comme $R_S(h)$ est une variable aléatoire qui obéit à une loi binomiale d'espérance $R(h)$, on a :

$$\Pr_{S\sim D^m}\left(R_S(h)=\frac{k}{m}\right) = \binom{m}{k}\left(R(h)\right)^k\left(1-R(h)\right)^{m-k} .$$

Ce qui permet de simplifier l'équation de départ :

$$\mathop{\mathbf{E}}_{S\sim D^m}\mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{D}(R_S(h),R(h))} = \mathop{\mathbf{E}}_{h\sim P} \sum_{k=0}^{m} \binom{m}{k}\left(\frac{k}{m}\right)^k\left(1-\frac{k}{m}\right)^{m-k} = \xi(m) \,.$$

□

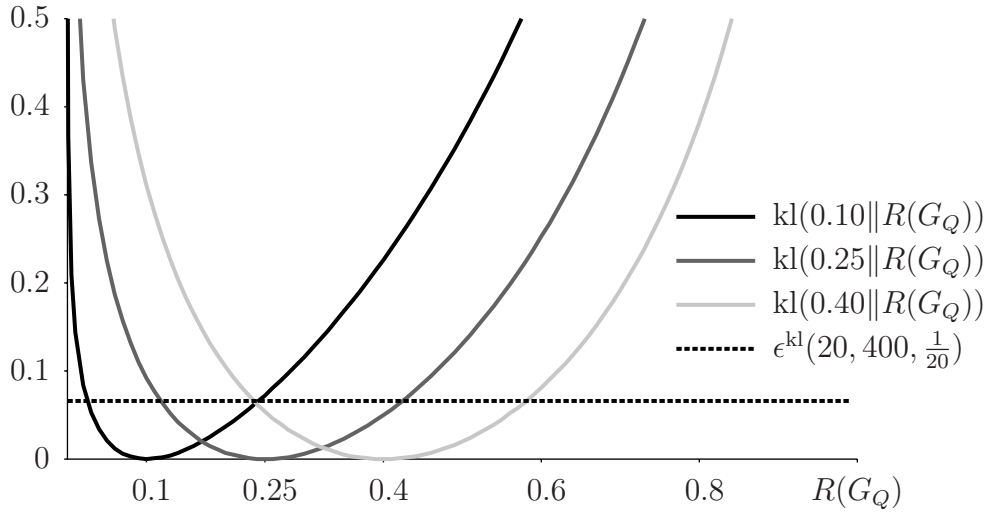



FIG. 3.1: Comportement de $\mathrm{kl}\Big(R_S(G_Q)\Big\|R(G_Q)\Big)$ en fonction de $R(G_Q)$, avec $R_S(G_Q) \in \{0.10, 0.25, 0.40\}$. La ligne pointillée représente une valeur $\epsilon^{\mathrm{kl}}(20, 400, \frac{1}{20}) \simeq 0.066$, définissant un intervalle de confiance dont les extrémités sont les bornes $B^{\mathrm{kl}}_-$ et $B^{\mathrm{kl}}_+$. Pour $R_S(G_Q) = 0.25$, on a $B^{\mathrm{kl}}_- \simeq 0.118$ et $B^{\mathrm{kl}}_- \simeq 0.424$.

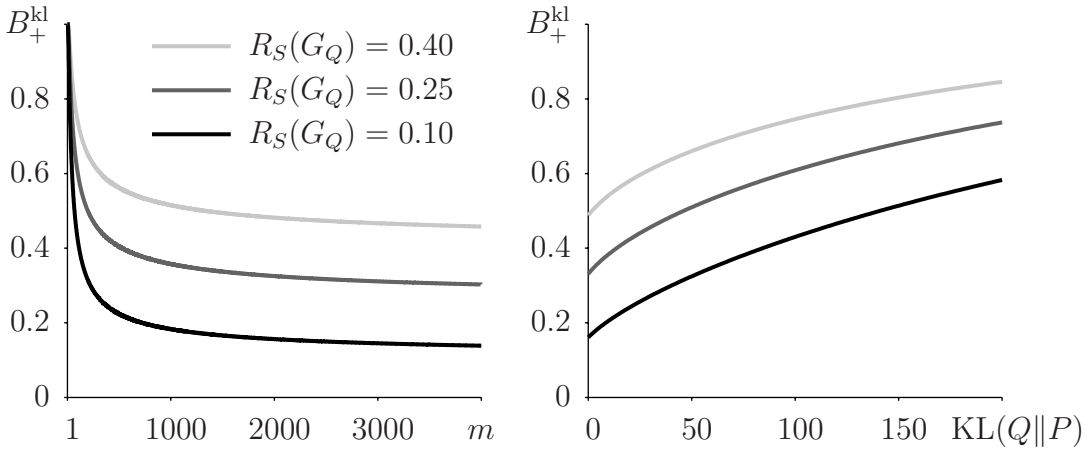
Notons que dans le corollaire 8 ci-dessus, le terme $m + 1$ du théorème original de Langford et Seeger [15] est remplacé par la fonction $\xi(m)$. Cela permet d'obtenir de meilleures garanties, car :

$$\xi(m) \;\leq\; \mathop{\mathbf{E}}_{h\sim P} \sum_{k=0}^{m} 1 = \mathop{\mathbf{E}}_{h\sim P} (m+1) = m+1 \;.$$

Nous désirons maintenant formuler une borne inférieure et une borne supérieure sur le risque d'un classificateur en se basant sur le corollaire 8. La fonction $\mathrm{kl}\Big(R_S(G_Q)\Big\|R(G_Q)\Big)$ quantifie la «distance»[2] entre le risque empirique de Gibbs et le vrai risque de Gibbs d'un classificateur. Un aperçu graphique de cette fonction pour différentes valeurs de risque est donné à la figure 3.1. Le corollaire 8 indique que cette «distance» est inférieure, avec une probabilité $1 - \delta$, à une certaine quantité qui dépend du nombre d'exemples d'entraînement, de la divergence entre $Q$ et $P$ et du paramètre de confiance $\delta$. Nous désignons cette quantité par $\epsilon^{\mathrm{kl}}$ :

$$\epsilon^{\mathrm{kl}}(\mathrm{KL}, m, \delta) \stackrel{\mathrm{def}}{=} \frac{\mathrm{KL} + \ln \frac{\xi(m)}{\delta}}{m} \;. \tag{3.5}$$

[2]La divergence de Kullback-Leibler est souvent présentée intuitivement comme une mesure de distance entre deux distributions de probabilités. Toutefois, il ne s'agit pas d'une véritable mesure de distance, puisqu'elle n'est pas symétrique ( $\exists Q, P$ tels que $\mathrm{KL}(Q\|P) \neq \mathrm{KL}(P\|Q)$ ).




(a) $B_+^{\mathrm{kl}}$ en fonction de $m$ (b) $B_+^{\mathrm{kl}}$ en fonction de $\mathrm{KL}(Q\|P)$

FIG. 3.2: Valeurs de la borne Langford-Seeger $B_+^{\mathrm{kl}}$ selon $m$ (avec $\mathrm{KL}(Q\|P) = 20$ et $\delta = \frac{1}{20}$) et selon $\mathrm{KL}(Q\|P)$ (avec $m = 400$ et $\delta = \frac{1}{20}$) pour $R_S(G_Q) \in \{0.10, 0.25, 0.40\}$.

Nous formulons ainsi l'intervalle de confiance où se situe le vrai risque, avec probabilité $1 - \delta$ :

$$\mathrm{kl}\Big(R_S(G_Q)\Big\|R(G_Q)\Big) \leq \epsilon^{\mathrm{kl}}(\mathrm{KL}(Q\|P), m, \delta) \ . \tag{3.6}$$

Nous obtenons à la fois une borne inférieure et une borne supérieure sur le risque en trouvant les valeurs de $R(G_Q)$ situées aux extrémités de cet intervalle :

$$B_-^{\mathrm{kl}}(S, G_Q, \delta, P) \stackrel{\mathrm{def}}{=} \inf\left\{r : \mathrm{kl}\Big(R_S(G_Q)\Big\|r\Big) \leq \epsilon^{\mathrm{kl}}(\mathrm{KL}(Q\|P), |S|, \delta)\right\}, \tag{3.7}$$

$$B_+^{\mathrm{kl}}(S, G_Q, \delta, P) \stackrel{\mathrm{def}}{=} \sup\left\{r : \mathrm{kl}\Big(R_S(G_Q)\Big\|r\Big) \leq \epsilon^{\mathrm{kl}}(\mathrm{KL}(Q\|P), |S|, \delta)\right\}. \tag{3.8}$$

Nous désignons ces résultats par *bornes Langford-Seeger*. Intéressons-nous davantage à l'étude de la borne supérieure $B_+^{\mathrm{kl}}$. Cette borne est d'autant précise que la valeur de $\epsilon^{\mathrm{kl}}(\mathrm{KL}(Q\|P), m, \delta)$ est faible. Examinons l'influence des trois paramètres de $\epsilon^{\mathrm{kl}}$ sur la précision de la borne et voyons comment nous pouvons interpréter intuitivement ces données :

- À valeurs $\mathrm{KL}(Q\|P)$ et $\delta$ fixes, la valeur de la borne s'approche du risque empirique avec l'augmentation du nombre $m$ d'exemples d'entraînement (tel qu'illustré à la figure 3.2a). Utiliser davantage d'exemples pour l'apprentissage améliore en effet la compréhension du phénomène observé. De plus, à la limite $m \to \infty$, la borne nous informe que le risque empirique $R_S(G_Q)$ converge vers le vrai risque $R(G_Q)$.
- À valeurs $m$ et $\delta$ fixes, la valeur de la borne s'éloigne du risque empirique lorsque la divergence entre $Q$ et $P$ augmente (tel qu'illustré à la figure 3.2b). En effet, lorsque nos connaissances *a priori* du phénomène observé sont cohérentes avec le

classificateur construit par l'algorithme d'apprentissage, on s'attend à ce que le risque empirique soit un bon indicateur du vrai risque. Toutefois, lorsque la distribution $Q$ est très différente de la distribution $P$, il y a un danger que l'algorithme ait «surappris» les données d'entraînement.

– À valeurs $m$ et $\mathrm{KL}(Q\|P)$ fixes, la valeur de la borne s'éloigne du risque empirique avec la diminution du paramètre $\delta$, c'est-à-dire avec l'augmentation du niveau de confiance $1-\delta$. Notons que, pour les valeurs de $\delta$ typiquement utilisés (entre 1% et 10%), l'influence de ce paramètre sur la valeur de la borne est minime comparativement aux deux paramètres discutés précédemment.

Afin de comprendre dans quel contexte s'applique la borne Langford-Seeger, étudions la forme du corollaire 8, qui est :

$$\forall\, D, \mathcal{H}, P, \delta : \ \Pr\left(\forall\, Q : \ \ldots\right) \geq 1-\delta\ .$$

Insistons sur le fait que la borne est valide, avec probabilité $1-\delta$, uniformément pour toute distribution $Q$. Cela permet de comparer les bornes obtenues de plusieurs pondérations d'un vote de majorité (plusieurs distributions $Q$) sur un ensemble de classificateurs $\mathcal{H}$ et de sélectionner la pondération associée à la borne minimale. Cependant, la borne n'est pas valide uniformément pour toute distribution $P$ caractérisant le prior. Ainsi, si on veut comparer les bornes obtenues par $k$ priors distincts, il faut faire appel à la borne de l'union. Chaque borne doit être calculée avec un paramètre de confiance $\frac{\delta}{k}$ afin de s'assurer que la borne minimale soit valide avec probabilité $1-\delta$ [3].

### 3.1.3 Borne PAC-Bayes hyperparamétrée

Afin d'obtenir une deuxième version de la borne PAC-Bayes à l'aide du théorème général, nous désirons appliquer le théorème 7 avec une fonction $\mathcal{D}(q,p)$ qui est linéaire en $q$ (le risque empirique). Nous cherchons une fonction de la forme $\mathrm{D}(q\|p,C) = \mathcal{F}(p) - C\cdot q$, où l'influence de $q$ est déterminée par un hyperparamètre $C \in \mathbb{R}^+$. Nous souhaitons que la fonction $\mathcal{F}$ soit convexe et indépendante de la variable $q$. Lors de la démonstration du corollaire 9, nous établirons qu'une fonction répondant à nos exigences est la suivante :

$$\mathrm{D}(q\|p,C) \stackrel{\text{def}}{=} -\ln\left[1-(1-e^{-C})p\right] - Cq\ . \tag{3.9}$$

[3] Pour un classificateur de risque empirique $R_S$, chacune des bornes $B_1, B_2, \ldots, B_k$ obtenues par les $k$ priors distincts est valide avec une probabilité $1-\frac{\delta}{k}$. La borne de l'union permet de calculer la probabilité que toutes ces bornes soient valides simultanément :

$$\Pr\left(\bigcup_{i=1}^{k}[R_S > B_i]\right) \leq \sum_{i=1}^{k}\Pr\left(R_S > B_i\right) = \sum_{i=1}^{k}\frac{\delta}{k} = \delta \iff \Pr\left(\bigcap_{i=1}^{k}[R_S \leq B_i]\right) \geq 1-\delta\ .$$

Le corollaire obtenu par cette démarche se révèle être un résultat énoncé originellement par Olivier Catoni [8].

**Corollaire 9.** *Pour toute distribution $D$, pour toute classe $\mathcal{H}$ de classificateurs, pour toute distribution $P$ sur $\mathcal{H}$, pour toute valeur réelle positive $C$ et pour tout $\delta \in ]0,1]$, on a :*

$$\Pr_{S\sim D^m}\left(\begin{array}{l}\forall\, Q \text{ sur } \mathcal{H}:\\ R(G_Q) \leq \frac{1}{1-e^{-C}}\left\{1-\exp\left[-\left(C\cdot R_S(G_Q)+\frac{1}{m}\left[\mathrm{KL}(Q\|P)+\ln\frac{1}{\delta}\right]\right)\right]\right\}\end{array}\right) \geq 1-\delta\ .$$

où $\mathrm{KL}(Q\|P)$ est la divergence de Kullback-Leibler donnée par l'équation (3.1).

*Démonstration.* Posons d'abord $\mathcal{D}(q,p) = \mathcal{F}(p) - C\cdot q$. Alors, on observe :

$$\begin{aligned}
\mathop{\mathbf{E}}_{S\sim D^m}\mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{D}(R_S(h),R(h))} &= \mathop{\mathbf{E}}_{h\sim P}\mathop{\mathbf{E}}_{S\sim D^m} e^{m\mathcal{F}(R(h))-CmR_S(h)}\\
&= \mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{F}(R(h))}\mathop{\mathbf{E}}_{S\sim D^m} e^{-CmR_S(h)}\\
&= \mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{F}(R(h))}\sum_{k=0}^{m}\Pr_{S\sim D^m}\left(R_S(h)=\frac{k}{m}\right)e^{-Ck}\\
&= \mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{F}(R(h))}\sum_{k=0}^{m}\binom{m}{k}R(h)^k(1-R(h))^{m-k}e^{-Ck}\\
&= \mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{F}(R(h))}\sum_{k=0}^{m}\binom{m}{k}(R(h)e^{-C})^k(1-R(h))^{m-k}\\
&= \mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{F}(R(h))}\left[R(h)e^{-C}+(1-R(h))\right]^m .
\end{aligned}$$

où la dernière égalité découle du binôme de Newton : $\displaystyle\sum_{k=0}^{m}\binom{m}{k}x^k y^{m-k} = (x+y)^m$.
Afin d'appliquer le théorème 7, nous désirons nous départir de l'espérance sur la distribution $P$. Nous procédons en rendant l'expression $e^{m\mathcal{F}(R(h))}\left[R(h)e^{-C}+(1-R(h))\right]^m$ indépendante de $h$. Plus particulièrement, nous choisissons la fonction $\mathcal{F}(p)$ telle que $e^{m\mathcal{F}(p)}\left[pe^{-C}+(1-p)\right]^m = 1$. Par les propriétés des logarithmes, nous obtenons :

$$\mathcal{F}(p) \;=\; -\ln\left[pe^{-C}+(1-p)\right] \;=\; -\ln\left[1-(1-e^{-C})p\right].$$

Nous avons donc maintenant :

$$\mathop{\mathbf{E}}_{S\sim D^m}\mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{D}(R_S(h),R(h))} \;=\; \mathop{\mathbf{E}}_{h\sim P} 1 \;=\; 1\ . \tag{3.10}$$

De la définition de la fonction $\mathcal{F}$, nous obtenons $\mathcal{D}(q,p) = \mathcal{F}(p) - C\cdot q = \mathrm{D}(q\|p,C)$, où D est définie par l'équation (3.9). Remarquons que la fonction $\mathcal{F}(p)$ est convexe en $p$. Conséquemment, $\mathrm{D}(q\|p,C)$ est convexe en $p$ et $q$, ce qui nous permet de l'insérer dans

l'énoncé du théorème 7, de pair avec l'équation (3.10). Ainsi, l'expression figurant à l'intérieur de la probabilité de l'énoncé du théorème 7 devient :

$$\begin{aligned} & -\ln\left[1 - R(G_Q)(1-e^{-C})\right] - C \cdot R_S(G_Q) \leq \frac{1}{m}\left[\mathrm{KL}(Q\|P) + \ln\frac{1}{\delta}\right] \\ \Leftrightarrow \quad & -\ln\left[1 - R(G_Q)(1-e^{-C})\right] \leq C \cdot R_S(G_Q) + \frac{1}{m}\left[\mathrm{KL}(Q\|P) + \ln\frac{1}{\delta}\right] \\ \Leftrightarrow \quad & R(G_Q) \leq \frac{1}{1-e^{-C}}\left\{1 - \exp\left[-\left(C \cdot R_S(G_Q) + \frac{1}{m}\left[\mathrm{KL}(Q\|P) + \ln\frac{1}{\delta}\right]\right)\right]\right\}. \end{aligned}$$

□

Le corollaire précédent nous permet d'énoncer une borne PAC-Bayes. Puisque sa valeur dépend d'un hyperparamètre $C$, nous la désignons par *borne hyperparamétrée* :

(3.11)
$$B_+^C(S, G_Q, \delta, P, C) \stackrel{\text{def}}{=} \frac{1}{1-e^{-C}}\left\{1 - \exp\left[-\left(C \cdot R_S(G_Q) + \frac{1}{m}\left[\mathrm{KL}(Q\|P) + \ln\frac{1}{\delta}\right]\right)\right]\right\}.$$

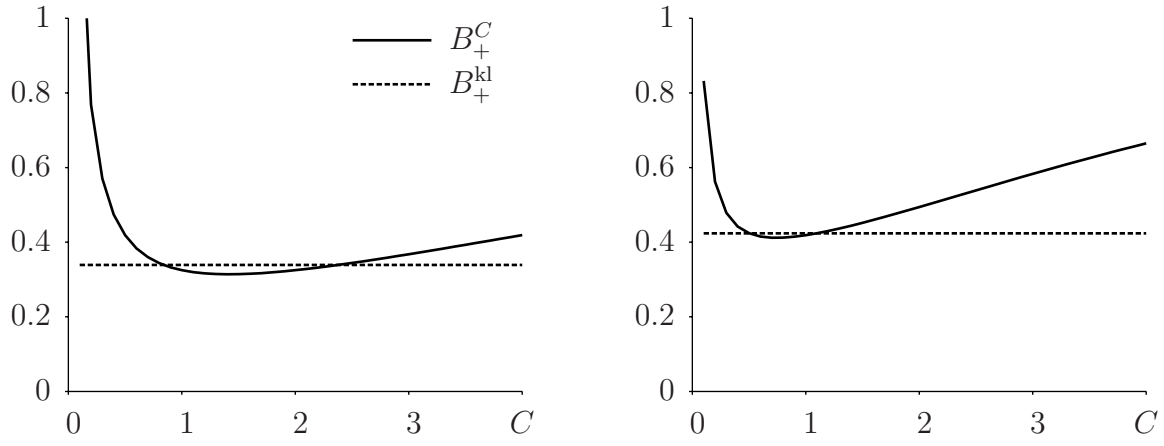
Notons que, alors que la borne de Langford-Seeger (dérivée du théorème 8) permet d'obtenir à la fois une borne inférieure et un borne supérieure sur le vrai risque, cette nouvelle expression ne nous fournit qu'une borne supérieure.

Comme l'illustre la figure 3.3, la valeur optimale de l'hyperparamètre $C$ fournissant la borne minimale dépend des autres paramètres de la borne ($R_S(G_Q)$, $m$ et $\mathrm{KL}(Q\|P)$). Il n'est donc pas possible de déduire *a priori* la valeur de $C$ optimale. La forme du corollaire 9 indique que la borne hyperparamétrée est valide uniformément pour toute distribution $Q$, mais pas pour toute valeur de $C$ :

$$\forall\, D, \mathcal{H}, P, C, \delta : \ \Pr\left(\forall\, Q : \ \ldots\right) \geq 1 - \delta\ .$$

Lorsque, pour un classificateur donné, nous désirons obtenir la borne hyperparamétrée de valeur minimale, il faut calculer la borne pour plusieurs valeurs prédéterminées de $C$ [4]. Afin de conserver la validité de la garantie, nous calculons chaque borne en remplaçant $\ln\frac{1}{\delta}$ par $\ln\frac{k}{\delta}$, où $k$ est le nombre de valeurs de $C$ (en vertu de la borne de l'union).

[4] Lors de nos tests empiriques, nous avons observé que le $C$ optimal se situe généralement dans l'intervalle $[0, 2]$. Afin de trouver une valeur de $C$ jugée satisfaisante, il est suffisant de calculer 20 valeurs de bornes correspondant à des valeurs de $C$ uniformément distribuées dans cet intervalle.




(a) $R_S(G_Q) = 0.10, m = 100, \mathrm{KL}(Q\|P) = 10$ (b) $R_S(G_Q) = 0.25, m = 400, \mathrm{KL}(Q\|P) = 20$

FIG. 3.3: Valeurs de la borne hyperparamétrée $B_+^C$ en fonction de $C$ pour différentes combinaisons de paramètres. La ligne pointillée indique la valeur de la borne $B_+^{\mathrm{kl}}$ pour les mêmes paramètres.

### 3.1.4 Comparaison des deux bornes PAC-Bayes

À la section 3.1.2, nous avons interprété l'expression de la borne Langford-Seeger comme une «distance» entre le risque empirique et le vrai risque. L'équation (3.6) évoque un intervalle de confiance à l'intérieur duquel est situé le vrai risque avec probabilité $1 - \delta$ :

$$\begin{aligned} & \mathrm{kl}(R_S(G_Q)\|R(G_Q)) \leq \epsilon^{\mathrm{kl}}(\mathrm{KL}(Q\|P), m, \delta)\,, \\ \text{où :}\quad & \mathrm{kl}(q\|p) \stackrel{\mathrm{def}}{=} q \ln \tfrac{q}{p} + (1-q) \ln \tfrac{1-q}{1-p}\,, \\ & \epsilon^{\mathrm{kl}}(\mathrm{KL}, m, \delta) \stackrel{\mathrm{def}}{=} \tfrac{\mathrm{KL} + \ln \frac{\xi(m)}{\delta}}{m}\,. \end{aligned} \tag{3.12}$$

Il est possible d'exprimer similairement la borne hyperparamétrée. L'inégalité suivante est obtenue en manipulant les termes de la borne hyperparamétrée (équation (3.11)) afin de mettre en évidence ceux formant la fonction D (équation (3.9)) :

$$\begin{aligned} & \mathrm{D}(R_S(G_Q)\|R(G_Q), C) \leq \epsilon^{\mathrm{D}}(\mathrm{KL}(Q\|P), m, \delta)\,, \\ \text{où :}\quad & \mathrm{D}(q\|p, C) \stackrel{\mathrm{def}}{=} -\ln\left[1 - (1 - e^{-C})p\right] - Cq\,, \\ & \epsilon^{\mathrm{D}}(\mathrm{KL}, m, \delta) \stackrel{\mathrm{def}}{=} \tfrac{\mathrm{KL} + \ln \frac{1}{\delta}}{m}\,. \end{aligned} \tag{3.13}$$

Les termes de droite des équations (3.12) et (3.13) ($\epsilon^{\mathrm{kl}}$ et $\epsilon^{\mathrm{D}}$) ont une formulation semblable. Leur valeur ne diffère que d'un terme $\frac{\ln \xi(m)}{m}$. Ceci permet de constater que

la précision de la borne hyperparamétrée est influencée par les paramètres $\mathrm{KL}(Q\|P)$, $m$ et $\delta$ de la même manière que la borne Langford-Seeger analysée à la section 3.1.2.

Afin de comparer les termes de gauche des mêmes équations (kl et D), examinons la valeur de $C$ qui donne la borne hyperparamétrée minimale. Il s'agit de la valeur de $C$ maximisant la fonction $\mathrm{D}(q\|p, C)$ pour $q$ et $p$ constants. Par la proposition suivante, on constate qu'avec ce $C$ les fonctions kl et D sont équivalentes (la figure 3.4 illustre ce résultat).

**Proposition 10** (Alexandre Lacasse[5])**.** *Pour toutes valeurs $q$ et $p$ telles que $q, p \in [0, 1[$ et $q \leq p$, on a :*

$$\max_{c \geq 0} \{\mathrm{D}(q\|p, c)\} = \mathrm{kl}(q\|p)\,.$$

*Démonstration.* Calculons les dérivées première et seconde de D selon $c$ :

$$\begin{aligned}
\frac{\partial \mathrm{D}(q\|p,c)}{\partial c} &= \frac{pe^{-c}}{1-p(1-e^{-c})} - q \;= \frac{p}{p+e^{c}(1-p)} - q\;, \\
\frac{\partial^2 \mathrm{D}(q\|p,c)}{\partial^2 c} &= p\frac{e^{c}(p-1)}{[p-e^{c}(1-p)]^2}\;.
\end{aligned}$$

Puisque $p \in ]0, 1]$, la dérivée seconde est toujours négative et la fonction D est concave en $c$. Son maximum se trouve au point $c_0$ où la dérivée première est nulle :

$$\frac{\partial \mathrm{D}(q\|p,c_0)}{\partial c} = 0 \iff q = \frac{p}{p+e^{c_0}(1-p)} \iff c_0 = \ln\left(\frac{qp-p}{qp-q}\right).$$

En introduisant la valeur de $c_0$ dans la fonction D, on obtient :

$$\begin{aligned}
\mathrm{D}(q\|p,c_0) &= -\ln\left[1-\left(1-\left(\frac{qp-q}{qp-p}\right)\right)p+1\right] - q\ln\left(\frac{qp-p}{qp-q}\right) \\
&= -\ln\left[\frac{(1-p)(q-1)+q(p-1)}{q-1}\right] - q\ln\left(\frac{p}{q}\right) - q\ln\left(\frac{q-1}{p-1}\right) \\
&= -\ln\left[\frac{p-1}{q-1}\right] + q\ln\left(\frac{q}{p}\right) - q\ln\left(\frac{q-1}{p-1}\right) \\
&= (1-q)\ln\left(\frac{1-q}{1-p}\right) + q\ln\left(\frac{q}{p}\right) \\
&= \mathrm{kl}(q\|p)\,.
\end{aligned}$$

□

Lorsqu'on calcule une garantie sur le risque d'un classificateur, on désire habituellement obtenir la borne supérieure la plus basse qui soit. De ce point de vue, la borne

[5] Résultat non publié à ce jour.

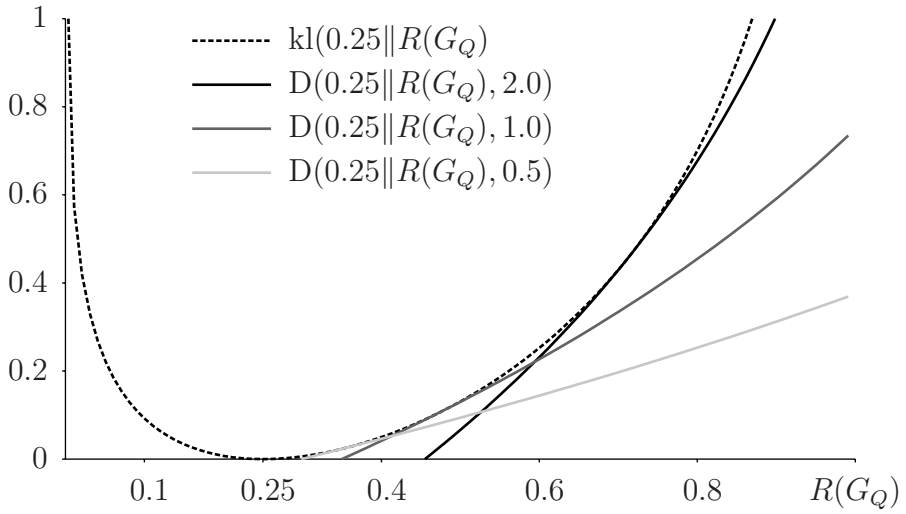



FIG. 3.4: Comparaison des fonctions $\mathrm{D}(\cdot\|\cdot, C)$ (avec $C \in \{0.5, 1.0, 2.0\}$) et $\mathrm{kl}(\cdot\|\cdot)$ pour un même risque empirique $R_S(G_Q) = 0.25$. Comme le montre la proposition 10, on observe que les deux fonctions sont équivalentes pour certaines valeurs de $C$.

hyperparamétrée est presque équivalente à la borne de Langford-Seeger. Leur différence provient uniquement des termes $\epsilon^{\mathrm{D}}$ et $\epsilon^{\mathrm{kl}}$. Rappelons que la recherche d'une bonne approximation de la borne hyperparamétrée minimale nécessite $k$ calculs de bornes avec un paramètre de confiance $\frac{\delta}{k}$. La borne hyperparamétrée est donc légèrement plus serrée que la borne Langford-Seeger lorsque $k < \xi(m)$, ce qui est le cas lorsque l'ensemble d'entraînement est suffisamment grand[6]. En contrepartie, la borne Langford-Seeger possède l'avantage d'être obtenue par un calcul plus direct.

## 3.2 Spécialisation aux classificateurs linéaires

La théorie PAC-Bayes s'applique aux classificateurs par vote de majorité[7], qu'on exprime par une distribution de probabilité $Q$ sur un espace de classificateurs de base $\mathcal{H}$. Cette formulation diffère grandement de la représentation classique des classificateurs linéaires. Cependant, John Langord et John Shawe-Taylor [14, 15] ont montré qu'en exprimant un classificateur linéaire comme un vote de majorité, la borne PAC-Bayes permet d'obtenir de bonnes garanties sur son risque. L'approche présentée ici s'inspire

[6] En supposant qu'on calcule la borne hyperparamétrée pour 20 valeurs de l'hyperparamètre $C$, la borne obtenue sera inférieure à la borne Langford-Seeger lorsque $m \geq 238$ (voir l'équation (3.4)).

[7] Rappelons que les bornes PAC-Bayes s'appliquent en fait aux classificateurs de Gibbs et que les bornes sur le risque de Bayes des classificateurs par vote de majorité correspondants sont obtenues par le biais du lemme 1.

fortement de leurs travaux.

Nous exprimerons un classificateur linéaire $h_{\mathbf{w}}$ par une distribution $Q_{\mathbf{w}}$ (section 3.2.1), ce qui permettra de calculer le risque empirique de Gibbs associé $R(G_{Q_{\mathbf{w}}})$ (section 3.2.2). Ensuite, nous déterminerons une distribution *a priori* $P_{\mathbf{w}',\sigma}$ (section 3.2.3), permettant de calculer la divergence de Kullback-Leibler $\mathrm{KL}(Q_{\mathbf{w}} \| P_{\mathbf{w}',\sigma})$ (section 3.2.4). En insérant les expressions du risque empirique de Gibbs et de la divergence de Kullback-Leibler dans les formules des bornes PAC-Bayes (équations (3.7), (3.8) et (3.11)), nous pourrons obtenir des garanties sur le risque des classificateurs linéaires.

Nous verrons à la section 3.3 que les expressions des bornes PAC-Bayes obtenues se révèlent des outils de sélection de modèle intéressants. Ces représentations mathématiques seront aussi à la base des nouveaux algorithmes d'apprentissage présentés au chapitre 4.

### 3.2.1 Choix du posterior

Rappelons qu'un classificateur linéaire dans un espace à $n$ dimensions est caractérisé par un vecteur $\mathbf{w} \in \mathbb{R}^n$ (pour simplifier la notation, on utilise ici la représentation implicite du biais). La sortie du classificateur linéaire $h_{\mathbf{w}}$ est donnée par :

$$h_{\mathbf{w}}(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x})\,.$$

Considérons maintenant un vote de majorité dont les classificateurs de base sont donnés par l'ensemble (continu) de tous les classificateurs linéaires dans $\mathbb{R}^n$ :

$$\mathcal{H} = \{h_{\mathbf{v}} | \mathbf{v} \in \mathbb{R}^n\}\,.$$

Pour représenter $h_{\mathbf{w}}$ par un vote de majorité, on désire construire une distribution $Q_{\mathbf{w}}$ sur l'ensemble $\mathcal{H}$ telle que la sortie du classificateur de Bayes associé soit la même que celle du classificateur $h_{\mathbf{w}}$. On dit alors que le classificateur $h_{\mathbf{w}}$ est *Bayes-équivalent* pour le vote de majorité $Q_{\mathbf{w}}$ :

$$h_{\mathbf{w}}(\mathbf{x}) = B_{Q_{\mathbf{w}}}(\mathbf{x}) \;= \mathrm{sgn}\left[\mathop{\mathbf{E}}_{\mathbf{v}\sim Q_{\mathbf{w}}} h_{\mathbf{v}}(\mathbf{x})\right] = \mathrm{sgn}\left[\mathop{\mathbf{E}}_{\mathbf{v}\sim Q_{\mathbf{w}}} \mathrm{sgn}(\mathbf{v}\cdot\mathbf{x})\right] \quad \forall\, \mathbf{x} \in \mathcal{X}\,.$$

Comme nous en informe la proposition 11 ci-dessous, une distribution $Q_{\mathbf{w}}$ *symétrique autour de* $\mathbf{w}$ est un cas particulier où le classificateur $h_{\mathbf{w}}$ est Bayes-équivalent. On dit qu'une distribution est symétrique autour de $\mathbf{w}$ lorsqu'elle respecte la propriété suivante :

$$\forall\, \mathbf{v}, \mathbf{u} \in \mathcal{H} : (\mathbf{w} - \mathbf{v}) = -(\mathbf{w} - \mathbf{u}) \Longrightarrow Q_{\mathbf{w}}(\mathbf{v}) = Q_{\mathbf{w}}(\mathbf{u})\,. \tag{3.14}$$

**Proposition 11.** *Pour toute distribution $Q_{\mathbf{w}}$ sur un ensemble $\mathcal{H}$ de classificateurs linéaires telle que $Q_{\mathbf{w}}$ est symétrique autour de $\mathbf{w}$ et non nulle dans un voisinage de $\mathbf{w}$, le classificateur $h_{\mathbf{w}}$ est Bayes-équivalent au vote de majorité $B_{Q_{\mathbf{w}}}$.*

*Démonstration.* Soit $h_{\mathbf{w}} \in \mathcal{H}$ un classificateur linéaire et $\mathbf{x}$ un exemple dans l'espace de ce classificateur. Posons $y' = h_{\mathbf{w}}(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x})$. Définissons ensuite les ensembles $\mathbb{A}$ et $\mathbb{D}$ , qui regroupent respectivement les vecteurs correspondant aux classificateurs en accord avec $h_{\mathbf{w}}$ et en désaccord avec $h_{\mathbf{w}}$. Définissons aussi l'ensemble $\hat{\mathbb{D}}$ «symétrique» à l'ensemble $\mathbb{D}$ par rapport à $\mathbf{w}$. Nous avons :

$$\begin{array}{rcl} \mathbb{A} & = & \{ \ \mathbf{v} \in \mathcal{H} \ \mid \ h_{\mathbf{v}}(\mathbf{x}) = \text{sgn}(\mathbf{v} \cdot \mathbf{x}) = \ y' \ \ \}, \\ \mathbb{D} & = & \{ \ \mathbf{v} \in \mathcal{H} \ \mid \ h_{\mathbf{v}}(\mathbf{x}) = \text{sgn}(\mathbf{v} \cdot \mathbf{x}) = -y' \ \ \}, \\ \hat{\mathbb{D}} & = & \{ \ \mathbf{u} \in \mathcal{H} \ \mid \ \mathbf{v} \in \mathbb{D}, (\mathbf{w} - \mathbf{v}) = -(\mathbf{w} - \mathbf{u}) \ \ \}. \end{array}$$

Considérons un vecteur $\mathbf{v} \in \mathbb{D}$ et son vecteur «symétrique» $\mathbf{u} = 2\mathbf{w} - \mathbf{v} \in \hat{\mathbb{D}}$. Notons que $\mathbf{w} \cdot \mathbf{x}$ et $\mathbf{v} \cdot \mathbf{x}$ possèdent des signes différents. Nous avons :

$$\begin{array}{rcl} h_{\mathbf{u}}(\mathbf{x}) = \text{sgn}(\mathbf{u} \cdot \mathbf{x}) & = & \text{sgn}\left[(2\mathbf{w} - \mathbf{v}) \cdot \mathbf{x}\right] \\ & = & \text{sgn}\left[2\mathbf{w} \cdot \mathbf{x} - \mathbf{v} \cdot \mathbf{x}\right] \\ & = & \text{sgn}\left[\mathbf{w} \cdot \mathbf{x}\right] \\ & = & h_{\mathbf{w}}(\mathbf{x}) = y' \, . \end{array}$$

Ainsi, $\mathbf{u} \in \mathbb{A}$, ce qui nous informe que $\hat{\mathbb{D}} \subset \mathbb{A}$ et $\mathbb{D} \cap \hat{\mathbb{D}} = \emptyset$. De plus, nous savions déjà que $\mathbb{A} \cap \mathbb{D} = \emptyset$.

Considérons maintenant une distribution $Q_{\mathbf{w}}$ sur l'ensemble $\mathcal{H}$ qui est symétrique autour de $\mathbf{w}$ (respectant l'expression (3.14)) et non nulle dans un voisinage de $\mathbf{w}$. Pour tout $\mathbf{v} \in \mathcal{H}$, on a $Q_{\mathbf{w}}(\mathbf{v}) = Q_{\mathbf{w}}(2\mathbf{w} - \mathbf{v})$. Nous pouvons désormais calculer la sortie du classificateur de Bayes associé à $Q_{\mathbf{w}}$ :

$$\begin{array}{rcl} B_{Q_{\mathbf{w}}}(\mathbf{x}) & = & \text{sgn}\left[\underset{\mathbf{v} \sim Q_{\mathbf{w}}}{\mathbf{E}} h_{\mathbf{v}}(\mathbf{x})\right] = \text{sgn}\left[\displaystyle\int_{\mathcal{H}} Q_{\mathbf{w}}(\mathbf{v}) h_{\mathbf{v}}(\mathbf{x}) d\mathbf{v}\right] \\ & = & \text{sgn}\left[y'\left(\displaystyle\int_{\mathbb{A}} Q_{\mathbf{w}}(\mathbf{v}) d\mathbf{v} - \int_{\mathbb{D}} Q_{\mathbf{w}}(\mathbf{v}) d\mathbf{v}\right)\right] \\ & = & \text{sgn}\left[y'\left(\displaystyle\int_{\mathbb{A} \setminus \hat{\mathbb{D}}} Q_{\mathbf{w}}(\mathbf{v}) d\mathbf{v} + \int_{\hat{\mathbb{D}}} Q_{\mathbf{w}}(\mathbf{v}) d\mathbf{v} - \int_{\mathbb{D}} Q_{\mathbf{w}}(\mathbf{v}) d\mathbf{v}\right)\right] \\ & = & \text{sgn}\left[y'\left(\displaystyle\int_{\mathbb{A} \setminus \hat{\mathbb{D}}} Q_{\mathbf{w}}(\mathbf{v}) d\mathbf{v} + \int_{\mathbb{D}} \left[Q_{\mathbf{w}}(2\mathbf{w} - \mathbf{v}) - Q_{\mathbf{w}}(\mathbf{v})\right] d\mathbf{v}\right)\right] \\ & = & \text{sgn}\left[y' \displaystyle\int_{\mathbb{A} \setminus \hat{\mathbb{D}}} Q_{\mathbf{w}}(\mathbf{v}) d\mathbf{v}\right] \\ & = & y' \ = \ h_{\mathbf{w}}(\mathbf{x}) \, . \end{array}$$

Donc, le classificateur $h_{\mathbf{w}}$ est Bayes-équivalent au vote de majorité $B_{Q_{\mathbf{w}}}$. □

Inspiré par Langford et Shawe-Taylor [14, 15], notre choix de distribution $Q_{\mathbf{w}}$, symétrique autour de $\mathbf{w}$, se porte sur une distribution gaussienne de variance unitaire dans chacune des $n$ dimensions et centrée sur le vecteur $\mathbf{w}$ :

$$Q_{\mathbf{w}}(\mathbf{v}) = \left(\frac{1}{\sqrt{2\pi}}\right)^n e^{-\frac{1}{2}\|\mathbf{v}-\mathbf{w}\|^2} = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(v_i - w_i)^2} . \tag{3.15}$$

Notons que la distribution $Q_{\mathbf{w}}$ ci-haut accorde un poids non-nul à tous les classificateurs $h_{\mathbf{v}} \in \mathcal{H}$. Il en sera de même pour la distribution *a priori* définie à la section 3.2.3. Comme nous le verrons à la section 3.2.4, ces choix permettent de quantifier aisément la similarité entre les distributions *a priori* et *a posteriori* par le calcul de la divergence de Kullback-Leibler.

### 3.2.2 Calcul du risque de Gibbs

Calculons le risque de Gibbs $R_S(G_{Q_{\mathbf{w}}})$ associé à la distribution $Q_{\mathbf{w}}$ de l'équation (3.15). Pour ce faire, calculons d'abord le risque de Gibbs sur un exemple $\mathbf{z} = (\mathbf{x}, y)$ :

$$\begin{aligned}
R_{\{\mathbf{z}\}}(G_{Q_{\mathbf{w}}}) &= \mathop{\mathbf{E}}_{\mathbf{v}\sim Q_{\mathbf{w}}} I(\mathrm{sgn}(\mathbf{v}\cdot\mathbf{x}) \neq y) \\
&= \mathop{\mathbf{E}}_{\mathbf{v}\sim Q_{\mathbf{w}}} I(y\mathbf{v}\cdot\mathbf{x} \leq 0) \\
&= \int_{\mathbb{R}^n} Q_{\mathbf{w}}(\mathbf{v})\, I(y\mathbf{v}\cdot\mathbf{x} \leq 0)\, d\mathbf{v} \\
&= \int_{\mathbb{R}^n} \left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(v_i - w_i)^2}\right) I(y\mathbf{v}\cdot\mathbf{x} \leq 0)\, d\mathbf{v} \\
&= \left(\frac{1}{\sqrt{2\pi}}\right)^n \int_{\mathbb{R}^n} \left(\prod_{i=1}^{n} e^{-\frac{1}{2}(v_i - w_i)^2}\right) I(y\mathbf{v}\cdot\mathbf{x} \leq 0)\, d\mathbf{v} .
\end{aligned}$$

Pour faciliter le calcul, nous choisissons la base dans laquelle on représente les vecteurs de sorte que le premier premier vecteur de cette base soit $\frac{y\mathbf{x}}{\|\mathbf{x}\|}$. Dans cette base, les vecteurs $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ et $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ ont comme premiers éléments les valeurs suivantes :

$$v_1 = \frac{y\mathbf{v}\cdot\mathbf{x}}{\|\mathbf{x}\|} , \qquad w_1 = \frac{y\mathbf{w}\cdot\mathbf{x}}{\|\mathbf{x}\|} .$$

Puisque $y\mathbf{v}\cdot\mathbf{x} = \|\mathbf{x}\|v_1$ et $\|\mathbf{x}\| > 0$, nous avons $I(y\mathbf{v}\cdot\mathbf{x} \leq 0) = I(v_1 \leq 0)$. Cela permet d'écrire :

$$\begin{aligned}
R_{\{\mathbf{z}\}}(G_{Q_\mathbf{w}}) &= \left(\frac{1}{\sqrt{2\pi}}\right)^n \int_{\mathbb{R}^n} \left(\prod_{i=1}^n e^{-\frac{1}{2}(v_i-w_i)^2}\right) I(v_1 \leq 0)\, d\mathbf{v} \\
&= \left(\frac{1}{\sqrt{2\pi}}\right)^n \int_{\mathbb{R}^n} \left(\prod_{i=2}^n e^{-\frac{1}{2}(v_i-w_i)^2}\right) e^{-\frac{1}{2}(v_1-w_1)^2} I(v_1 \leq 0)\, d\mathbf{v} \\
&= \left(\frac{1}{\sqrt{2\pi}}\right)^n \int_{\mathbb{R}} \left(\sqrt{2\pi}\right)^{n-1} e^{-\frac{1}{2}(v_1-w_1)^2} I(v_1 \leq 0)\, dv_1 \\
&= \left(\frac{1}{\sqrt{2\pi}}\right) \int_{\mathbb{R}} e^{-\frac{1}{2}(v_1-w_1)^2} I(v_1 \leq 0)\, dv_1 \ .
\end{aligned}$$

En posant $t = v_1 - w_1$, on obtient :

$$\begin{aligned}
R_{\{\mathbf{z}\}}(G_{Q_\mathbf{w}}) &= \left(\frac{1}{\sqrt{2\pi}}\right) \int_{\mathbb{R}} e^{-\frac{1}{2}t^2} I(t \leq -w_1)\, dt \\
&= \left(\frac{1}{\sqrt{2\pi}}\right) \int_{-\infty}^{-w_1} e^{-\frac{1}{2}t^2}\, dt \\
&= 1 - \left(\frac{1}{\sqrt{2\pi}}\right) \int_{w_1}^{\infty} e^{-\frac{1}{2}t^2}\, dt \ .
\end{aligned}$$

Remarquons que la dernière intégrale correspond à la fonction bien connue de répartition d'une loi normale de moyenne nulle et de variance unité. Pour simplifier la notation, définissons la fonction $\Phi(a)$ ainsi :

$$\Phi(a) \stackrel{\text{def}}{=} 1 - \Pr_{x\sim N(0,1)}(x < a)\,. \tag{3.16}$$

La formulation du risque de Gibbs calculé sur l'exemple $\mathbf{z}$ devient :

$$\begin{aligned}
R_{\{\mathbf{z}\}}(G_{Q_\mathbf{w}}) &= \Phi(w_1) = \Phi\left(\frac{y\mathbf{w}\cdot\mathbf{x}}{\|\mathbf{x}\|}\right) = \Phi\left(\|\mathbf{w}\|\frac{y\mathbf{w}\cdot\mathbf{x}}{\|\mathbf{w}\|\|\mathbf{x}\|}\right) \\
&= \Phi\left(\|\mathbf{w}\|\Gamma_\mathbf{w}(\mathbf{x}, y)\right).
\end{aligned}$$

où $\Gamma_\mathbf{w}(\mathbf{x}, y)$ est la marge normalisée du classificateur $h_\mathbf{w}$ sur l'exemple $\mathbf{z}$ telle que définie par l'équation (2.3). Nous pouvons maintenant définir le vrai risque de Gibbs et le risque empirique de Gibbs :

$$\begin{aligned}
R(G_{Q_\mathbf{w}}) &= \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} \Phi\left(\|\mathbf{w}\|\Gamma_\mathbf{w}(\mathbf{x}, y)\right), \\
R_S(G_{Q_\mathbf{w}}) &= \frac{1}{m}\sum_{i=1}^m \Phi\left(\|\mathbf{w}\|\Gamma_\mathbf{w}(\mathbf{x}_i, y_i)\right).
\end{aligned} \tag{3.17}$$

### 3.2.3 Choix du prior

Le calcul de la borne PAC-Bayes requiert une distribution *a priori* sur l'ensemble des classificateurs de base, qui représente nos connaissances sur le phénomène à apprendre avant d'observer les données d'entraînement. Le choix de ce prior est crucial puisqu'il amène la borne à préférer certains classificateurs. Puisque nous avons exprimé le posterior $Q_{\mathbf{w}}$ par une gaussienne, il semble naturel de faire de même pour le prior. Dans ce texte, nous distinguerons trois types de prior.

Le *prior non-informatif* (utilisé par Langford et Shawe-Taylor [14, 15]) s'avère une gaussienne centrée à l'origine et de variance unité dans toutes les directions. Il s'agit du prior préconisé lorsque nous ne possédons aucune information sur la nature d'un bon classificateur, car il accorde le même poids *a priori* à tous les hyperplans possibles :

$$P_{\mathbf{0},1}(\mathbf{v}) = \left(\frac{1}{\sqrt{2\pi}}\right)^n e^{-\frac{1}{2}\|\mathbf{v}\|^2} = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}v_i^2} \ . \tag{3.18}$$

Le *prior informatif isotrope* (utilisé par Ambroladze et al. [1]) s'avère une gaussienne centrée sur un vecteur $\mathbf{w}'$ et de variance unité dans toutes les directions. Le vecteur $\mathbf{w}'$ peut alors représenter un classificateur linéaire $h_{\mathbf{w}'}$ en lequel nous sommes confiant *a priori*.

$$P_{\mathbf{w}',1}(\mathbf{v}) = \left(\frac{1}{\sqrt{2\pi}}\right)^n e^{-\frac{1}{2}\|\mathbf{v}-\mathbf{w}'\|^2} = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(v_1-\|\mathbf{w}'\|)^2} \prod_{i=2}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}v_i^2} \ , \tag{3.19}$$

où $v_1$ correspond à la norme de la composante de $\mathbf{v}$ parallèle à $\mathbf{w}'$.

Le *prior informatif allongé*, introduit ici, généralise le prior précédent à une gaussienne centrée sur un vecteur $\mathbf{w}'$, de variance $\sigma$ dans la direction parallèle au vecteur $\mathbf{w}'$ et de variance unité dans les autres directions. Lorsque $\sigma \gg 1$, cela traduit une confiance en l'hyperplan caractérisé par $\mathbf{w}'$, sans pour autant préférer une norme $\|\mathbf{w}'\|$ particulière.

$$P_{\mathbf{w}',\sigma}(\mathbf{v}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(v_1-\|\mathbf{w}'\|)^2}{\sigma^2}} \prod_{i=2}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}v_i^2} \ . \tag{3.20}$$

Insistons sur le fait que le prior informatif allongé avec $\sigma = 1$ équivaut à un prior informatif isotrope. De même, un prior informatif isotrope avec $\mathbf{w}' = \mathbf{0}$ équivaut à un prior non informatif.

### 3.2.4 Calcul de la divergence de Kullback-Leibler

Calculons l'expression de la divergence de Kullback-Leibler entre le posterior $Q_{\mathbf{w}}$ (équation (3.15)) et le prior informatif allongé (équation (3.20)). Définissons les vecteurs $\mathbf{w}_{\parallel}$ et $\mathbf{w}_{\perp}$, respectivement la projection parallèle et la projection orthogonale du vecteur $(\mathbf{w} - \mathbf{w}')$ sur le vecteur $\mathbf{w}'$ (tel qu'illustré par la figure 3.5) :

$$\mathbf{w}_{\parallel} \stackrel{\text{def}}{=} \frac{(\mathbf{w} - \mathbf{w}') \cdot \mathbf{w}'}{\|\mathbf{w}'\|^2} \mathbf{w}' \,, \tag{3.21}$$

$$\mathbf{w}_{\perp} \stackrel{\text{def}}{=} (\mathbf{w} - \mathbf{w}') - \mathbf{w}_{\parallel} \,. \tag{3.22}$$

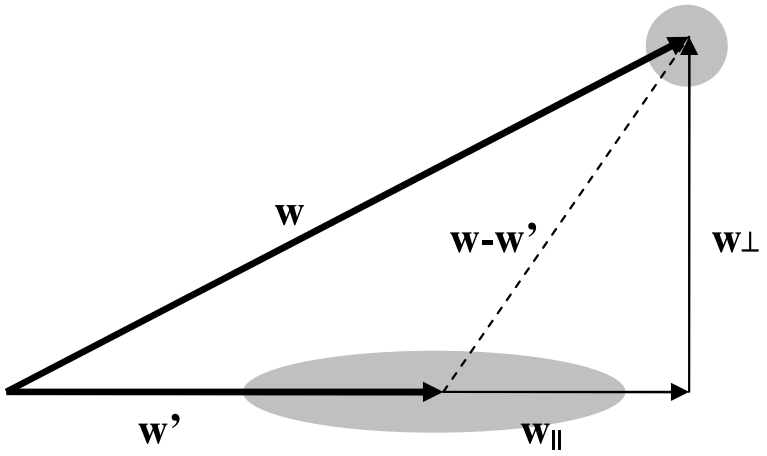



FIG. 3.5: Vecteurs intervenant dans le calcul de la divergence de Kullback-Leibler

Pour faciliter le calcul, nous choisissons la base dans laquelle on représente les vecteurs telle que les deux premiers vecteurs de cette base soient $\frac{\mathbf{w}_{\parallel}}{\|\mathbf{w}_{\parallel}\|}$ et $\frac{\mathbf{w}_{\perp}}{\|\mathbf{w}_{\perp}\|}$ respectivement. Dans cette base, les vecteurs $\mathbf{v}$, $\mathbf{w}$ et $\mathbf{w}'$ ont les formes suivantes :

$$\begin{array}{rcl}
\mathbf{v} & = & (\ v_1,\ v_2,\ v_3,\ \ldots,\ v_n\ ) \in \mathbb{R}^n \,, \\
\mathbf{w} & = & (\ w_1,\ w_2,\ 0,\ \ldots,\ 0\ ) \quad \text{avec } w_1^2 + w_2^2 = \|\mathbf{w}\|^2 \,, \\
\mathbf{w}' & = & (\ w'_1,\ 0,\ 0,\ \ldots,\ 0\ ) \quad \text{avec } w'_1 = \|\mathbf{w}'\| \,.
\end{array}$$

Nous pouvons désormais calculer la divergence entre $Q_{\mathbf{w}}$ et $P_{\mathbf{w}',\sigma}$ :

$$\begin{aligned}
\mathrm{KL}(Q_{\mathbf{w}}\|P_{\mathbf{w}',\sigma}) &= \mathop{\mathbf{E}}_{\mathbf{v}\sim Q_{\mathbf{w}}} \ln \frac{Q_{\mathbf{w}}(\mathbf{v})}{P_{\mathbf{w}',\sigma}(\mathbf{v})} \\
&= \int_{\mathbb{R}^n} Q_{\mathbf{w}}(\mathbf{v}) \ln \frac{Q_{\mathbf{w}}(\mathbf{v})}{P_{\mathbf{w}',\sigma}(\mathbf{v})} d\mathbf{v} \\
&= \int_{\mathbb{R}^n} Q_{\mathbf{w}}(\mathbf{v}) \ln \left[ \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(v_1-w_1)^2} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(v_2-w_2)^2} \prod_{i=3}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}v_i^2}}{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(v_1-w_1')^2}{\sigma^2}} \prod_{i=2}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}v_i^2}} \right] d\mathbf{v} \\
&= \int_{\mathbb{R}^n} Q_{\mathbf{w}}(\mathbf{v}) \ln \left[ \frac{e^{-\frac{1}{2}(v_1-w_1)^2} e^{-\frac{1}{2}(v_2-w_2)^2}}{\sigma^{-1} e^{-\frac{1}{2}\frac{(v_1-w_1')^2}{\sigma^2}} e^{-\frac{1}{2}v_2^2}} \right] d\mathbf{v} \\
&= \int_{\mathbb{R}^n} Q_{\mathbf{w}}(\mathbf{v}) \left[ \ln(\sigma) + \ln \left( \frac{e^{-\frac{1}{2}(v_1-w_1)^2}}{e^{-\frac{1}{2}\frac{(v_1-w_1')^2}{\sigma^2}}} \right) + \ln \left( \frac{e^{-\frac{1}{2}(v_2-w_2)^2}}{e^{-\frac{1}{2}v_2^2}} \right) \right] d\mathbf{v} \,.
\end{aligned}$$

Décomposons l'équation en trois parties telles que $\mathrm{KL}(Q_{\mathbf{w}}\|P_{\mathbf{w}',\sigma}) = \mathrm{KL}_A + \mathrm{KL}_B + \mathrm{KL}_C$ :

$$\begin{aligned}
\mathrm{KL}_A &= \int_{\mathbb{R}^n} Q_{\mathbf{w}}(\mathbf{v}) \ln(\sigma)\, d\mathbf{v} \\
&= \left( \frac{1}{\sqrt{2\pi}} \right)^n \int_{\mathbb{R}^n} e^{-\frac{1}{2}\|\mathbf{v}-\mathbf{w}\|^2} \ln(\sigma)\, d\mathbf{v} \\
&= \ln \sigma \ ,
\end{aligned}$$

$$\begin{aligned}
\mathrm{KL}_B &= \int_{\mathbb{R}^n} Q_{\mathbf{w}}(\mathbf{v}) \left[ \frac{1}{2} v_2^2 - \frac{1}{2}(v_2 - w_2)^2 \right] d\mathbf{v} \\
&= \left( \frac{1}{2\sqrt{2\pi}} \right) \int_{\mathbb{R}} e^{-\frac{1}{2}(v_2-w_2)^2} \left[ v_2^2 - (v_2 - w_2)^2 \right] dv_2 \\
&= \left( \frac{1}{2\sqrt{2\pi}} \right) \int_{\mathbb{R}} e^{-\frac{1}{2}(v_2-w_2)^2} \left[ 2v_2 w_2 - w_2^2 \right] dv_2 \\
&= \frac{1}{2} w_2^2 \ ,
\end{aligned}$$

$$\begin{aligned}
\mathrm{KL}_C &= \int_{\mathbb{R}^n} Q_{\mathbf{w}}(\mathbf{v}) \left[ \frac{1}{2} \frac{(v_1 - w_1')^2}{\sigma^2} - \frac{1}{2}(v_1 - w_1)^2 \right] d\mathbf{v} \\
&= \left( \frac{1}{2\sqrt{2\pi}} \right) \int_{\mathbb{R}} e^{-\frac{1}{2}(v_1-w_1)^2} \left[ \frac{(v_1 - w_1')^2}{\sigma^2} - (v_1 - w_1)^2 \right] dv_1 \\
&= \frac{1}{2} \left( \frac{1 + (w_1 - w_1')^2}{\sigma^2} - 1 \right).
\end{aligned}$$

En regroupant ces trois parties, nous obtenons :

$$\begin{aligned}
\mathrm{KL}(Q_{\mathbf{w}}\|P_{\mathbf{w}',\sigma}) &= \ln \sigma + \frac{1}{2} w_2^2 + \frac{1}{2} \left( \frac{1 + (w_1 - w_1')^2}{\sigma^2} - 1 \right) \\
&= \frac{1}{2} w_2^2 + \frac{1}{2} \left( \ln \sigma^2 + \frac{1}{\sigma^2} - 1 + \frac{(w_1 - w_1')^2}{\sigma^2} \right).
\end{aligned}$$

Remarquons que $w_2 = \|\mathbf{w}_\perp\|$, puisque la projection orthogonale de $\mathbf{w}$ sur $\mathbf{w}'$ est identique à la projection orthogonale de $(\mathbf{w} - \mathbf{w}')$ sur $\mathbf{w}'$. De même, $w_1 - \|\mathbf{w}_\parallel\| = w'_1$, puisque la différence des projections parallèles de $\mathbf{w}$ et $(\mathbf{w} - \mathbf{w}')$ sur $\mathbf{w}'$ équivaut au vecteur $\mathbf{w}'$ lui-même. On obtient ainsi l'expression générale du prior allongé (3.25), de laquelle on déduit simplement les expressions pour les cas particuliers que sont le prior non informatif (3.23) et le prior informatif isotrope (3.24) :

$$\text{(3.23)} \qquad \mathrm{KL}(Q_\mathbf{w} \| P_{\mathbf{0},1}) = \frac{1}{2}\|\mathbf{w}\|^2 ,$$

$$\text{(3.24)} \qquad \mathrm{KL}(Q_\mathbf{w} \| P_{\mathbf{w}',1}) = \frac{1}{2}\|\mathbf{w}_\perp\|^2 + \frac{1}{2}\|\mathbf{w}_\parallel\|^2 = \frac{1}{2}\|\mathbf{w} - \mathbf{w}'\|^2 ,$$

$$\text{(3.25)} \qquad \mathrm{KL}(Q_\mathbf{w} \| P_{\mathbf{w}',\sigma}) = \frac{1}{2}\|\mathbf{w}_\perp\|^2 + \frac{1}{2}\left(\ln \sigma^2 + \frac{1}{\sigma^2} - 1 + \frac{\|\mathbf{w}_\parallel\|^2}{\sigma^2}\right) .$$

En pratique, ces expressions de divergence de Kullback-Leibler ont tendance à donner des valeurs qui ne sont pas trop élevées. Rappelons qu'une divergence de Kullback-Leibler élevée a pour effet de dégrader la valeur des bornes PAC-Bayes.

## 3.3 Sélection de modèle par la borne

Il est naturel d'envisager la spécialisation de la borne PAC-Bayes aux classificateurs linéaires décrite à la section précédente comme un outil de sélection de modèle. Cette avenue a été explorée par Ambroladze et al. [1]. Plus spécifiquement, ces travaux étudient la possibilité d'utiliser la borne PAC-Bayes afin de sélectionner les hyperparamètres des SVM.

Le classificateur résultant du processus d'apprentissage des SVM est un séparateur linéaire $h_\mathbf{w}$. La classification d'un exemple par $h_\mathbf{w}$ est déterminée uniquement par la direction de $\mathbf{w}$, alors que la spécialisation de la borne PAC-Bayes accorde une signification à la norme $\|\mathbf{w}\|$. Pour obtenir une borne sur le risque de $h_\mathbf{w}$ à l'aide de la borne PAC-Bayes, il convient d'effectuer une recherche linéaire parmi tous les vecteurs $\mathbf{w}^*$ de même direction que $\mathbf{w}$ et de sélectionner celui qui donne la plus petite valeur de la borne PAC-Bayes. La garantie sur le risque de Gibbs obtenue ainsi est valide, car la borne PAC-Bayes est valide uniformément pour toute distribution $Q_\mathbf{w}$. De même, les vecteurs $\mathbf{w}$ et $\mathbf{w}^*$ engendrent des classificateurs de Bayes équivalents ($B_{Q_\mathbf{w}}(\mathbf{x}) = B_{Q_{\mathbf{w}^*}}(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{X}$). Nous avons donc, par le lemme 1, que le double de la borne sur le risque de Gibbs $R(G_{Q_{\mathbf{w}^*}})$ est une borne sur le risque de Bayes $R(B_{Q_\mathbf{w}}) = R(h_\mathbf{w})$.

### 3.3.1 Résultats empiriques

Étant donnés des ensembles de valeurs de $C$ et $\gamma$ pré-déterminés, les travaux de Ambroladze et al. [1] comparent trois méthodes différentes pour obtenir un classificateur linéaire à noyau RBF à partir des SVM :

1. Effectuer une validation croisée 10-fois afin de sélectionner la paire de d'hyperparamètres $(C, \gamma)$ qui possède le plus faible risque de validation croisée. Conformément à la procédure classique, ces hyperparamètres $(C, \gamma)$ sont utilisés pour entraîner le classificateur final sur la totalité de l'ensemble d'entraînement.
2. Entraîner les SVM pour chaque paire d'hyperparamètres $(C, \gamma)$ sur la totalité de l'ensemble d'entraînement et calculer la borne PAC-Bayes Langford-Seeger des classificateurs obtenus en utilisant un prior non informatif $P_{\mathbf{0},1}$. Le classificateur possédant la borne minimale est sélectionné.
3. Diviser l'ensemble d'entraînement en deux sous-ensembles $S_1$ et $S_2$. Pour chaque paire de d'hyperparamètres $(C, \gamma)$, les SVM sont entraînés deux fois :
   (a) Un premier SVM est entraîné avec le sous-ensemble $S_1$. Le vecteur $\mathbf{w}'$ caractérisant le classificateur obtenu est utilisé pour définir un prior informatif isotrope $P_{\mathbf{w}',1}$.
   (b) Un deuxième SVM est entraîné avec la totalité des exemples $S_1 \cup S_2$. Le vecteur $\mathbf{w}$ caractérisant le classificateur obtenu est utilisé pour définir un posterior $Q_{\mathbf{w}}$.

   La borne PAC-Bayes Langford-Seeger du classificateur obtenu en (b) est calculée à partir de son risque empirique de Gibbs sur le sous-ensemble $S_2$ en utilisant le prior $P_{\mathbf{w}',1}$ obtenu en (a). Pour ce faire, la borne est calculée pour plusieurs normes possibles $\|\mathbf{w}'\|$ et la borne minimale est retenue[8]. De tous les classificateurs obtenus en (b), celui possédant la borne minimale est alors sélectionné.

La dernière méthode (3) requiert de déterminer la taille des ensembles $S_1$ et $S_2$. Les résultats obtenus par Ambroladze et al. [1] suggèrent qu'il est généralement préférable de sélectionner la moitié des exemples d'entraînement pour former $S_1$ et l'autre moitié pour former $S_2$. Cela favorise à la fois la qualité du risque de Bayes du classificateur sélectionné et la précision de la borne qui lui est associée.

Les résultats obtenus montrent que les deux méthodes de sélection de modèle par la borne (2 et 3) permettent d'obtenir des classificateurs dont les risques de Bayes sont

[8] En vertu de la borne de l'union, il faut calculer la borne PAC-Bayes avec un paramètre de confiance $\frac{\delta}{k}$ lorsqu'on calcule la borne pour $k$ priors différents. Le prior informatif allongé $P_{\mathbf{w}',\sigma}$, introduit à la section précédente, a pour but d'obtenir le même effet (accorder davantage d'importance à la direction du vecteur $\mathbf{w}'$ plutôt qu'à sa norme) tout en ne nécessitant qu'un seul calcul de borne.

équivalents. Par contre, ces risques sont parfois légèrement moins bons que ceux des classificateurs obtenus par validation croisée (1). Notons cependant que le temps de calcul exigé par la validation croisée est grandement supérieur à celui requis par la sélection de modèle par la borne. Un autre avantage de la sélection de modèle par la borne est l'obtention d'une garantie sur le vrai risque du classificateur. À ce chapitre, l'utilisation d'un prior informatif (3) permet d'obtenir des bornes entre 10% et 50% plus faibles que l'utilisation d'un prior non informatif (2).

### 3.3.2 Interprétation des critères de sélection

Bien que la sélection de modèle par la borne ne soit pas aussi efficace que la validation croisée, les résultats empiriques d'Ambroladze et al. [1] montrent que la spécialisation de la borne PAC-Bayes aux classificateurs linéaires est un bon indicateur de leur performance. Ainsi, l'expression mathématique de la borne semble synthétiser adéquatement certains critères que doit respecter ce type de classificateur. Il est intéressant d'interpréter ces critères afin de cerner les caractéristiques des classificateurs qu'ils favorisent.

Pour un ensemble d'entraînement $S$ et un paramètre de confiance $\delta$ donnés, la valeur de la borne PAC-Bayes favorise les classificateurs possédant à la fois un faible risque empirique de Gibbs $R_S(G_Q)$ et une faible divergence de Kullback-Leibler $\mathrm{KL}(Q\|P)$. Dans la borne Langford-Seeger (utilisée pour la sélection de modèle décrite précédemment), l'importance relative de ces deux paramètres est déterminée par la fonction $\mathrm{kl}(\cdot\|\cdot)$.

Par la spécialisation de la borne aux classificateurs linéaires, le risque de Gibbs est donné par une fonction de perte sigmoïdale (équation (3.17)), illustrée par la figure 3.6. Cette fonction dépend de la quantité $\|\mathbf{w}\|\Gamma_{\mathbf{w}}(\mathbf{x}, y)$, c'est-à-dire de la norme du vecteur $\mathbf{w}$ et de la marge normalisée pour chaque exemple $(\mathbf{x}, y) \in S$. On remarque que la perte sigmoïdale tend vers 0 à partir d'une certaine valeur pour les exemples bien classifiés (lorsque $\|\mathbf{w}\|\Gamma_{\mathbf{w}}(\mathbf{x}, y) \gtrsim 3$) et vers 1 pour les exemples mal classifiés (lorsque $\|\mathbf{w}\|\Gamma_{\mathbf{w}}(\mathbf{x}, y) \lesssim -3$). Il existe donc un intervalle de valeurs de marge à l'intérieur duquel la valeur de la perte sigmoïdale varie. L'étendue de cet intervalle diminue avec l'augmentation de la norme $\|\mathbf{w}\|$. Lorsque la norme est très grande, la perte sigmoïdale est presque équivalente au risque de Bayes du vote de majorité défini par la distribution $Q_{\mathbf{w}}$ [9]. On peut donc considérer la fonction de perte sigmoïdale comme une approximation du risque de Bayes.

[9] Comme le classificateur linéaire $h_{\mathbf{w}}$ est Bayes-équivalent pour le vote de majorité $Q_{\mathbf{w}}$, nous avons $R(B_{Q_{\mathbf{w}}}) = R(h_{\mathbf{w}})$. Sur un exemple $\mathbf{z} = (\mathbf{x}, y)$ particulier, le risque d'un classificateur linéaire vaut 0 si sa marge est positive et vaut 1 si sa marge est négative. Il en est de même pour la fonction de perte sigmoïdale lorsque $\|\mathbf{w}\| \to \infty$.

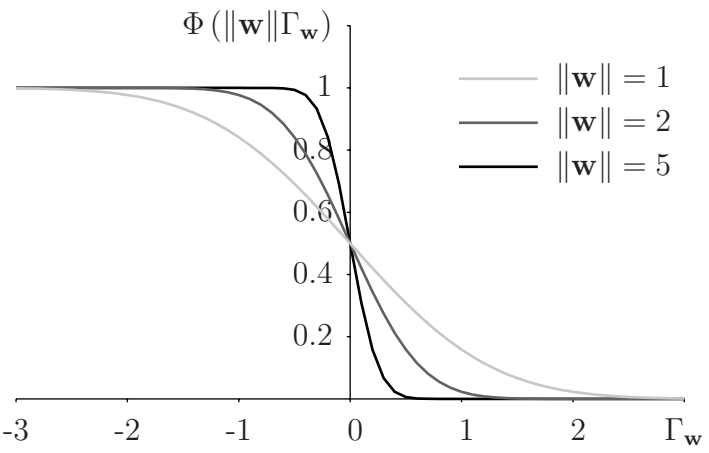



FIG. 3.6: Fonction de perte sigmoïdale $\Phi(\|\mathbf{w}\|\Gamma_{\mathbf{w}}(\mathbf{x}_i, y_i))$ en fonction de la marge normalisée $\Gamma_{\mathbf{w}}(\mathbf{x}_i, y_i)$ sur un exemple pour des valeurs $\|\mathbf{w}\| \in \{1, 2, 5\}$.

La valeur de la norme $\|\mathbf{w}\|$ influence aussi directement la valeur de la divergence $\mathrm{KL}(Q\|P)$. Examinons d'abord le cas du prior non informatif, où la borne incite à diminuer la quantité $\frac{1}{2}\|\mathbf{w}\|^2$. Pour un classificateur $h_{\mathbf{w}}$ possédant un petit risque empirique de Bayes, la borne est petite lorsqu'il est possible de sélectionner une faible valeur de norme $\|\mathbf{w}\|$ tout en conservant un petit risque empirique de Gibbs (c'est-à-dire qu'il est possible d'élargir l'intervalle où la perte sigmoïdale varie entre 0 et 1 en ne dégradant pas le risque empirique de Gibbs). Ainsi, l'expression de la borne favorise les classificateurs à grande marge de séparation, pour lesquels les exemples de l'ensemble d'entraînement sont éloignés de la frontière de décision. Le recours au prior informatif diminue l'importance accordée à la marge, en autant que les vecteurs correspondant au prior et au posterior possèdent des directions similaires.

Dans un contexte de sélection de modèle, l'expression des bornes PAC-Bayes appliquées aux classificateurs linéaires tend à favoriser les classificateurs possédant un faible risque empirique de Gibbs (et indirectement un faible risque empirique de Bayes) tout en possédant une grande marge de séparation. Ce dernier critère est représenté par l'expression $\|\mathbf{w}\|^2$ dans la formulation de la borne. Il est intéressant de remarquer que cette expression apparaît aussi dans l'équation (2.13) minimisée par les SVM. La minimisation de la norme $\|\mathbf{w}\|$ est donc une méthode préconisée autant par la spécialisation de la borne PAC-Bayes et que par les SVM afin de sélectionner des classificateurs à grande marge. On tend ainsi à éviter le surapprentissage des exemples d'entraînement.

# Chapitre 4

# Algorithmes de descente de gradient PAC-Bayes

> Dans la vie, l'essentiel est de porter sur tout des jugements a priori.
>
> Préface de l'Écume des jours
> (roman de Boris Vian)

Au chapitre précédent, nous avons présenté une méthode permettant d'appliquer les bornes PAC-Bayes aux classificateurs linéaires. Les travaux de Ambroladze et al. [1] montrent que les bornes ainsi calculées fournissent de bonnes garanties sur le risque des classificateurs obtenus grâce aux SVM, ce qui en fait de bons sélecteurs de modèle. Il semble donc que la théorie PAC-Bayes synthétise bien le compromis nécessaire à l'apprentissage. Cette constatation ouvre la voie au développement de nouveaux algorithmes d'apprentissage automatique directement inspirés de la théorie PAC-Bayes. Les travaux présentés ici se veulent un premier pas dans cette direction.

La borne Langford-Seeger et la borne hyperparamétrée se présentent toutes deux comme des expressions mathématiques associant à chaque classificateur d'un espace donné une borne sur son vrai risque. Notre objectif est de concevoir un algorithme permettant de trouver, parmi un espace de classificateurs donné (possiblement de taille infinie), celui minimisant ces expressions mathématiques. Nous procéderons à la minimisation de la borne PAC-Bayes par la méthode de descente de gradient.

L'algorithme présenté est nommé PBGD (de l'anglais *PAC-Bayes Gradient Descent*). Nous étudierons quelques variantes de cet algorithme, d'une part pour l'appliquer à plusieurs types de classificateur et d'autre part pour tirer profit des différentes possibilités que nous offrent les outils mathématiques énoncés jusqu'à maintenant.

## 4.1 Motivation

Bien qu'ils soient généralement inspirés de théories sur l'apprentissage, les algorithmes d'apprentissage conventionnels sont d'abord justifiés par des principes intuitifs et des considérations pratiques. À la suite de leur conception, le faible risque des classificateurs qu'ils construisent sur des données réelles stimule parfois le développement de théories permettant de les expliquer rigoureusement.

En développant un algorithme d'apprentissage directement basé sur la minimisation d'une borne, nous suggérons d'expérimenter l'approche inverse. La théorie PAC-Bayes possède de solides fondements statistiques et plusieurs travaux de recherche ont contribué à l'enrichir. Cela en fait un bon point de départ pour la conception de nouveaux algorithmes d'apprentissage automatique.

Il serait ambitieux de se fixer comme objectif immédiat le développement d'un algorithme dont l'efficacité rivalise avec les meilleures techniques d'apprentissage connues actuellement. Il est pourtant souhaitable que ces premières expérimentations mènent ultimement à un algorithme d'apprentissage compétitif basé sur la minimisation d'une borne sur le vrai risque, ce qui recèlerait plusieurs avantages. La garantie sur le vrai risque, fournie de pair avec chaque classificateur généré par un tel algorithme, permettrait d'utiliser tous les exemples étiquetés à notre disposition pour l'apprentissage, plutôt que de réserver un ensemble test pour calculer une garantie sur le risque (comme discuté à la section 2.4.3). De plus, alors que la plupart des algorithmes d'apprentissage conventionnels sont conditionnés par des hyperparamètres qui permettent d'ajuster leur capacité de généralisation au phénomène appris (le paramètre de marge floue des SVM [9], le nombre de couches et de neurones des réseaux de neurones [3] et, dans une moindre mesure, le nombre d'itérations d'AdaBoost [23]), les bornes sur le risque suggèrent habituellement elles-mêmes les compromis afin d'éviter le surapprentissage. Ainsi, un algorithme basé sur une borne serait dispensé de méthodes comme la validation croisée pour paramétrer l'algorithme d'apprentissage. Enfin, un algorithme d'apprentissage performant fondé sur des considérations théoriques consacrerait que celles-ci cernent bien le problème de l'apprentissage automatique.

Dans cet esprit, les algorithmes expérimentés dans ce mémoire s'avèrent aussi un moyen de mettre la théorie PAC-Bayes à l'épreuve. À la lumière des résultats obtenus, nous pourrons juger à quel point elle permet de bien comprendre le phénomène d'apprentissage, d'en cerner les lacunes et de la raffiner par la suite.

## 4.2 Espaces d'apprentissage

Nous nous intéressons à la conception d'un algorithme d'apprentissage produisant un classificateur linéaire. Préalablement à l'utilisation de l'algorithme, il faudra déterminer dans quel espace ce classificateur sera construit. Une possibilité est de considérer les classificateurs linéaires directement dans l'espace des exemples. Comme les classificateurs générés dans cet espace sont de complexité très limitée, nous préférons transformer chaque exemple $\mathbf{z} = (\mathbf{x}, y)$ en un vecteur de caractéristiques appartenant à un espace de complexité possiblement supérieure (voir la section 2.2.5) :

$$\boldsymbol{\phi}(\mathbf{x}) \stackrel{\text{def}}{=} (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_{n'}(\mathbf{x})) \in \mathbb{R}^{n'}.$$

Le classificateur linéaire associé à cet espace de caractéristiques consiste en un hyperplan défini par le vecteur $\mathbf{w} \in \mathbb{R}^{n'}$. On peut représenter les vecteurs $\mathbf{w}$ et $\boldsymbol{\phi}$ implicitement ou explicitement.

La *version primale* de l'algorithme utilise une représentation explicite des vecteurs de caractéristiques. Chaque élément du vecteur $\boldsymbol{\phi}$ est associé à un classificateur de base qui retourne une valeur réelle. L'algorithme construit donc un séparateur linéaire dans l'espace des votes de majorité, tel que décrit à la section 2.2.9.

La *version duale* de l'algorithme utilise une représentation implicite des vecteurs de caractéristiques. On tire profit du fait que tous les calculs manipulant ces vecteurs sont des produits scalaires afin d'appliquer la stratégie des noyaux, décrite à la section 2.2.6. La représentation duale du vecteur $\mathbf{w}$ est un vecteur $\boldsymbol{\alpha} \in \mathbb{R}^m$.

## 4.3 Apprentissage d'un prior

La spécialisation du théorème PAC-Bayes aux classificateurs linéaires présentée au chapitre précédent prévoit le recours à un prior informatif (voir la section 3.2.3). Ce prior prend la forme d'une distribution gaussienne (allongée ou isotrope) centrée sur un vecteur $\mathbf{w}'$. Lors de l'utilisation de l'algorithme d'apprentissage pour une application concrète, on possède parfois certaines connaissances du phénomène étudié *a priori* ne provenant pas de l'étude de l'ensemble d'entraînement. Dans ce cas, on peut traduire ces connaissances par le choix d'un vecteur $\mathbf{w}'$ qui marque la préférence envers un certain type de classificateur linéaire. Lors de l'apprentissage, l'algorithme sera alors amené à favoriser ce type de classificateur, dans la mesure où son risque empirique est raisonnablement faible.

Lorsque, au contraire, les seules informations que l'on possède sur le phénomène étudié proviennent des exemples d'entraînement, le choix adéquat s'avère le prior non informatif $\mathbf{w}' = \mathbf{0}$. En effet, ce prior accorde la même probabilité à tous les hyperplans possibles. Il marque aussi une préférence pour les classificateurs caractérisés par un vecteur $\mathbf{w}$ de faible norme. Comme expliqué à la section 3.3.2, en contraignant la norme $\|\mathbf{w}\|$ du classificateur final à une valeur faible, on favorise les classificateurs à grandes marges et on tend à éviter le surapprentissage. En contrepartie, cela réduit l'importance accordée à la diminution du risque empirique de Gibbs.

Afin d'accorder davantage d'importance à la diminution du risque empirique, nous étudierons la possibilité de réserver une fraction $p$ des exemples de l'ensemble d'entraînement $S$ pour apprendre le prior. L'algorithme s'exécutera donc en deux phases, chacune exploitant un sous-ensemble distinct de l'ensemble $S$ :

$$\begin{array}{lll} S_A \subseteq S & \text{avec} & m_A \stackrel{\text{def}}{=} |S_A| = \lfloor mp \rfloor \,, \\ S_B = S \setminus S_A & \text{avec} & m_B \stackrel{\text{def}}{=} |S_B| = m - m_A \,. \end{array}$$

Lors de la première phase d'apprentissage, nous minimisons la borne PAC-Bayes sur les exemples $S_A$ en utilisant un prior non informatif $\mathbf{w}'' = \mathbf{0}$. Cela fournit un classificateur caractérisé par un vecteur $\mathbf{w}'$. Dans la version duale de l'algorithme, nous représentons ce vecteur implicitement par un vecteur $\boldsymbol{\alpha}' \in \mathbb{R}^m$. Au cours de l'apprentissage, seules les composantes $\alpha'_i$ correspondant à des exemples $\mathbf{z}_i \in S_A$ sont modifiés. Les autres composantes sont nulles.

Lors de la deuxième phase d'apprentissage, nous minimisons la borne PAC-Bayes sur les exemples $S_B$ en centrant le prior sur le vecteur $\mathbf{w}'$. Dans la version duale de l'algorithme, nous utilisons aussi les exemples de l'ensemble $S_A$, mais seulement afin de conserver la représentation duale du vecteur $\mathbf{w}'$ par le vecteur $\boldsymbol{\alpha}'$. Pour accorder davantage d'importance à la direction de $\mathbf{w}'$, on peut recourir à un prior allongé qui pénalise moins les vecteurs en fonction de leur norme $\|\mathbf{w}'\|$ que le prior isotrope. Le résultat de cette deuxième phase d'apprentissage est le classificateur final caractérisé par le vecteur $\mathbf{w}$ (représenté implicitement par le vecteur $\boldsymbol{\alpha} \in \mathbb{R}^m$ dans la version duale de l'algorithme).

À chacune de ces deux étapes, le risque empirique de Gibbs est calculé seulement sur une fraction des exemples. Il y a donc un compromis à faire entre le nombre d'exemples $m_A$ utilisés pour apprendre le prior et le nombre d'exemples $m_B$ utilisés pour apprendre le posterior (le classificateur final). Cette méthode est optimale lorsque l'information contenue dans $S_A$ et $S_B$ est semblable, de telle sorte que l'observation des données lors de la deuxième phase d'apprentissage confirme les déductions faites lors la phase

d'apprentissage du prior. Dans cette situation, on peut entrevoir que les classificateurs finaux posséderont une très bonne borne sur le risque.

## 4.4 Définition des problèmes d'optimisation

Examinons la borne Langford-Seeger et la borne hyperparamétrée appliquées aux classificateurs linéaires en tant que fonctions du vecteur $\mathbf{w}$, en considérant constants les autres paramètres intervenant dans leur calcul. Comme elles proviennent des corollaires 8 et 9, les bornes PAC-Bayes sont valides uniformément pour toute distribution *a posteriori* $Q$, et donc pour tous les vecteurs $\mathbf{w}$. Ainsi, il est possible d'explorer le domaine de ces fonctions afin de trouver le vecteur $\mathbf{w}$ qui caractérise le classificateur possédant la borne sur le risque minimale. Considérant cela, nous concevons l'algorithme PBGD comme une procédure de minimisation d'une fonction. Énonçons d'abord les problèmes d'optimisation découlant de la borne Langford-Seeger et de la borne hyperparamétrée, et ce dans leurs versions primale et duale.

### 4.4.1 Borne Langford-Seeger

La borne PAC-Bayes Langford-Seeger est donnée par l'équation (3.8). Étant donnés un ensemble de données $S$ contenant $m$ exemples, un prior $P_{\mathbf{w}',\sigma}$ et un paramètre de confiance $\delta$, l'objectif est de trouver le vecteur $\mathbf{w}$ qui minimise la borne $B$ sous les contraintes suivantes :

$$\text{kl}\left(R_S(G_{Q_\mathbf{w}})||B\right) = \frac{1}{m}\left[\text{KL}(Q_\mathbf{w}||P_{\mathbf{w}',\sigma}) + \ln\left(\frac{\xi(m)}{\delta}\right)\right], \tag{4.1}$$

$$B > R_S(G_{Q_\mathbf{w}})\,. \tag{4.2}$$

Notons que la contrainte (4.1) est respectée pour la borne inférieure et la borne supérieure. La contrainte (4.2) limite le résultat à la borne supérieure.

Pour un vecteur $\mathbf{w}$ donné, il n'est pas possible d'exprimer ce problème en une fonction de $B$. On calcule plutôt $B$ en trouvant, parmi les deux racines de la fonction $f$ suivante, celle de valeur supérieure :

$$f(B) = \text{kl}\left(R_S(G_{Q_\mathbf{w}})||B\right) - \frac{1}{m}\left[\text{KL}(Q_\mathbf{w}||P_{\mathbf{w}',\sigma}) + \ln\left(\frac{\xi(m)}{\delta}\right)\right].$$

Pour ce faire, nous utilisons la méthode de Newton, qui converge vers la solution en approximant la fonction à l'aide de sa dérivée en un point précis :

$$f'(B) = \frac{\partial f(B)}{\partial B} = \frac{1 - R_S(G_{Q_\mathbf{w}})}{1 - B} - \frac{R_S(G_{Q_\mathbf{w}})}{B} = \frac{B - R_S(G_{Q_\mathbf{w}})}{B(1 - B)} \,.$$

L'algorithme 2 présente la méthode exacte utilisée pour le calcul de la borne. La valeur $B$ est initialisée à une valeur supérieure au risque empirique de Gibbs afin que la méthode de Newton converge vers la borne supérieure. On juge que la convergence est atteinte lorsque la progression effectuée entre deux itérations est inférieure à un certain seuil (ici, ce seuil est fixé à $10^{-12}$).

**Algorithme 2** BorneLangfordSeeger($R_S, \epsilon^{\text{kl}}$)

**Entrée :** $R_S = R_S(G_{Q_\mathbf{w}})$, le risque empirique de Gibbs.

**Entrée :** $\epsilon^{\text{kl}} = \frac{1}{m}\left[\text{KL}(Q_\mathbf{w}||P_{\mathbf{w}',\sigma}) + \ln\left(\frac{\xi(m)}{\delta}\right)\right]$, le terme de droite de l'équation (4.1).

Initialiser $B \leftarrow [1 + R_S]/2$ .
**répéter**
  $\Delta \leftarrow \frac{B(1-B)}{B-R_S} \cdot \left[\text{kl}\left(R_S||B\right) - \epsilon^{\text{kl}}\right]$ ,
  $B \leftarrow B - \Delta$ ,
**jusqu'à** $|\Delta| < 10^{-12}$ .

**Sortie :** $B$ .

### 4.4.2 Borne hyperparamétrée

La borne PAC-Bayes hyperparamétrée est donnée par l'équation (3.11). Étant donnés un ensemble de données $S$ contenant $m$ exemples, un prior $P_{\mathbf{w}',\sigma}$, un paramètre de confiance $\delta$ et une valeur $C$, l'objectif est de trouver le vecteur $\mathbf{w}$ qui minimise la borne $B$ donnée directement par la fonction suivante :

$$B = \frac{1}{1 - e^{-C}}\left\{1 - \exp\left[-\left(C \cdot R_S(G_{Q_\mathbf{w}}) + \frac{1}{m}\left[\text{KL}(Q_\mathbf{w}||P_{\mathbf{w}',\sigma}) + \ln\frac{1}{\delta}\right]\right)\right]\right\}. \tag{4.3}$$

Comme l'expression $\frac{1-e^{-m\beta}}{1-e^{-C}}$ est strictement croissante en $\beta$ pour $C > 0$ et $m > 0$, et que $\delta$ est constant, le vecteur $\mathbf{w}$ qui minimise la fonction $\beta$ suivante minimise également la borne hyperparamétrée :

$$\beta = C \cdot mR_S(G_{Q_\mathbf{w}}) + \text{KL}(Q_\mathbf{w}||P_{\mathbf{w}',\sigma})\,. \tag{4.4}$$

### 4.4.3 Espace primal

Afin de faciliter les calculs lors de l'exécution de l'algorithme PBGD dans sa version primale, nous créons pour chaque exemple $(\mathbf{x}_i, y_i)$ un vecteur $\boldsymbol{\psi}_i$ tel que :

$$\boldsymbol{\psi}_i \stackrel{\text{def}}{=} y_i \frac{\boldsymbol{\phi}(\mathbf{x}_i)}{\|\boldsymbol{\phi}(\mathbf{x}_i)\|} \ .$$

Nous écrivons $\psi_{i,j}$ pour désigner le $j$ème élément du vecteur $\boldsymbol{\psi}_i$.

En insérant cette définition dans l'équation (3.17), on obtient l'expression suivante permettant de calculer le risque empirique de Gibbs :

$$R_S(G_{Q_{\mathbf{w}}}) = \frac{1}{m}\sum_{i=1}^{m} \Phi\Big(\|\mathbf{w}\|\Gamma_{\mathbf{w}}(\boldsymbol{\phi}(\mathbf{x}_i), y_i)\Big) = \frac{1}{m}\sum_{i=1}^{m} \Phi\Big(\mathbf{w}\cdot\boldsymbol{\psi}_i\Big) \ . \tag{4.5}$$

Le calcul de la divergence entre les distributions $Q_{\mathbf{w}}$ et $P_{\mathbf{w}',\sigma}$ se fait par une simple application de l'équation (3.25) :

$$\begin{aligned} \mathrm{KL}(Q_{\mathbf{w}}\|P_{\mathbf{w}',\sigma}) &= \frac{1}{2}\|\mathbf{w}_\perp\|^2 + \frac{1}{2}\left(\ln\sigma^2 + \frac{1}{\sigma^2} - 1 + \frac{\|\mathbf{w}_\parallel\|^2}{\sigma^2}\right), \\ \text{où} \quad \|\mathbf{w}_\parallel\|^2 &= \left(\frac{(\mathbf{w}-\mathbf{w}')\cdot\mathbf{w}'}{\|\mathbf{w}'\|}\right)^2, \\ \|\mathbf{w}_\perp\|^2 &= \|\mathbf{w}-\mathbf{w}'\|^2 - \|\mathbf{w}_\parallel\|^2 \ . \end{aligned} \tag{4.6}$$

### 4.4.4 Espace dual

Les calculs de l'algorithme PBGD dans sa version duale utilisent la *matrice de Gram*, qui est définie comme suit :

$$G_{i,j} \stackrel{\text{def}}{=} y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \quad \forall\, i,j \in \{1,\ldots,m\}\,,$$

où $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ est le noyau permettant de calculer le produit scalaire entre deux exemples dans l'espace des carctéristiques (voir l'équation (2.5)). Remarquons que $G$ est une matrice symétrique ($G_{i,j} = G_{j,i}\ \forall\, i,j$).

Nous exprimons le vecteur $\mathbf{w}$ comme une combinaison linéaire des vecteurs de caractéristiques associés aux exemples d'entraînement. Les coefficients de cette combinaison linéaire sont les éléments du vecteur $\boldsymbol{\alpha} \in \mathbb{R}^m$ (voir l'équation (2.6)). Remarquons que

cette représentation duale limite le vecteur $\mathbf{w}$ au sous-espace engendré par les exemples d'entraînement à l'intérieur de l'espace des caractéristiques.

Nous calculons le produit scalaire entre le vecteur $\mathbf{w}$ et le vecteur de caractéristiques associé à l'exemple d'entraînement $(\mathbf{x}_i, y_i)$ ainsi :

$$\begin{aligned}\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i) &= \left(\sum_{j=1}^{m} y_j\alpha_j\boldsymbol{\phi}(\mathbf{x}_j)\right)\cdot\boldsymbol{\phi}(\mathbf{x}_i) = \sum_{j=1}^{m} y_j\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)\\ &= y_i\sum_{j=1}^{m}\alpha_j G_{i,j}\ .\end{aligned}$$

Cette formulation permet d'exprimer le risque empirique de Gibbs sous la forme suivante :

$$R_S(G_{Q_\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m}\Phi\left(\frac{y_i\mathbf{w}\cdot\boldsymbol{\phi}(\mathbf{x}_i)}{\|\boldsymbol{\phi}(\mathbf{x}_i)\|}\right) = \frac{1}{m}\sum_{i=1}^{m}\Phi\left(\frac{\sum_{j=1}^{m}\alpha_j G_{j,i}}{\sqrt{G_{i,i}}}\right). \tag{4.7}$$

Nous représentons également le vecteur $\mathbf{w}'$ implicitement à l'aide du vecteur $\boldsymbol{\alpha}' \in \mathbb{R}^m$. Comme ce vecteur correspond à une distribution *a priori* qui a été appris sur un sous-ensemble $S_A$ des exemples d'entraînement (voir la section 4.3), seules les composantes $\alpha'_i$ du vecteur $\boldsymbol{\alpha}'$ correspondant à des exemples $\mathbf{z}_i \in S_A$ qui ont servis à l'apprentissage du prior sont non nulles. Ainsi, le vecteur $\mathbf{w}'$ est limité au sous-espace engendré par les exemples de $S_A$ à l'intérieur de l'espace des caractéristiques.

Définissons la soustraction et le produit scalaire entre deux vecteurs $\mathbf{w}$ et $\mathbf{w}'$ représentés implicitement :

$$\mathbf{w}-\mathbf{w}' = \left(\sum_{i=1}^{m} y_i\alpha_i\boldsymbol{\phi}(\mathbf{x}_i)\right) - \left(\sum_{i=1}^{m} y_i\alpha'_i\boldsymbol{\phi}(\mathbf{x}_i)\right) = \sum_{i=1}^{m} y_i(\alpha_i-\alpha'_i)\boldsymbol{\phi}(\mathbf{x}_i)\ ,$$

$$\begin{aligned}\mathbf{w}\cdot\mathbf{w}' &= \left(\sum_{i=1}^{m} y_i\alpha_i\boldsymbol{\phi}(\mathbf{x}_i)\right)\cdot\left(\sum_{i=1}^{m} y_i\alpha'_i\boldsymbol{\phi}(\mathbf{x}_i)\right) = \sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j\alpha_i\alpha'_j\boldsymbol{\phi}(\mathbf{x}_i)\cdot\boldsymbol{\phi}(\mathbf{x}_j)\\ &= \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha'_j G_{i,j}\ .\end{aligned}$$

Nous pouvons désormais calculer la divergence entre les distributions $Q_{\mathbf{w}}$ et $P_{\mathbf{w}',\sigma}$ :

$$\mathrm{KL}(Q_{\mathbf{w}}\|P_{\mathbf{w}',\sigma}) = \frac{1}{2}\|\mathbf{w}_\perp\|^2 + \frac{1}{2}\left(\ln\sigma^2 + \frac{1}{\sigma^2} - 1 + \frac{\|\mathbf{w}_\parallel\|^2}{\sigma^2}\right), \tag{4.8}$$

$$\text{où} \quad \|\mathbf{w}_\parallel\|^2 = \left(\frac{(\mathbf{w}-\mathbf{w}')\cdot\mathbf{w}'}{\|\mathbf{w}'\|}\right)^2 = \frac{\left[\sum_{i=1}^m\sum_{j=1}^m(\alpha_i-\alpha'_i)\alpha'_jG_{i,j}\right]^2}{\sum_{i=1}^m\sum_{j=1}^m\alpha'_i\alpha'_jG_{i,j}},$$

$$\|\mathbf{w}_\perp\|^2 = \|\mathbf{w}-\mathbf{w}'\|^2 - \|\mathbf{w}_\parallel\|^2 = \left[\sum_{i=1}^m\sum_{j=1}^m(\alpha_i-\alpha'_i)(\alpha_j-\alpha'_j)G_{i,j}\right] - \|\mathbf{w}_\parallel\|^2.$$

## 4.5 Procédure d'optimisation

Nous avons établi les expressions mathématiques que les différentes versions de l'algorithme PBGD doivent minimiser. Les méthodes envisageables pour accomplir cette tâche sont limitées. Pour le comprendre, remarquons que l'expression du risque empirique de Gibbs (équation (3.16)), présente à la fois dans la borne Langford-Seeger et dans la borne hyperparamétrée, est donnée par une équation qui est non linéaire, non quadratique et non convexe. Cela dit, les expressions à minimiser sont dérivables en tout point, ce qui nous permet d'appliquer la méthode de descente de gradient.

### 4.5.1 Descente de gradient conjugué

Supposons que nous désirons minimiser une fonction $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ à $n$ variables. Le gradient de cette fonction au point $\mathbf{a}$ correspond à un vecteur contenant les dérivées partielles de $f$ selon chacune des $n$ variables évaluées au point $\mathbf{a}$ :

$$\nabla f(\mathbf{a}) \stackrel{\text{def}}{=} \left(\frac{\partial f}{\partial x_1}(\mathbf{a}), \frac{\partial f}{\partial x_2}(\mathbf{a}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{a})\right).$$

Le vecteur $\nabla f(\mathbf{a})$ indique la direction de plus forte pente. L'algorithme d'optimisation par *descente de gradient* est une méthode itérative qui consiste à parcourir le domaine d'une fonction, toujours en se dirigeant dans une direction qui permet de diminuer sa valeur. On initialise l'algorithme à un point quelconque $\mathbf{x}_0$. On effectue ensuite un «saut» dans la direction inverse au gradient, ce qui nous permet de progresser au point $\mathbf{x}_1$. On poursuit ensuite itérativement le processus de descente de gradient. Dans la version la plus simple de l'algorithme, on sélectionne le «saut» à effectuer à l'itération $t$ de la manière suivante :

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\nabla f(\mathbf{x}_t).$$

Ici, $\eta$ est une petite valeur positive nommée le *taux d'apprentissage*. Ce paramètre doit être sélectionné avec précaution selon la nature de la fonction à minimiser. Si la valeur de $\eta$ est trop élevée, les «sauts» seront trop grands et l'algorithme peinera à converger. Si la valeur de $\eta$ est trop faible, les «sauts» seront petits et la convergence sera très lente.

On arrête l'algorithme lorsque la progression est faible d'une itération à l'autre, indiquant que la procédure a convergé vers un minimum local de la fonction $f$. Ici, nous jugeons que la procédure a convergé lorsque $\|\nabla f(\mathbf{x}_t)\| < \epsilon_{abs}$ (avec un très petit $\epsilon_{abs}$). Ce critère d'arrêt est justifié par le fait que, au minimum $\mathbf{x}_{\min}$ de la fonction, le gradient $\|\nabla f(\mathbf{x}_{\min})\|$ possède une valeur nulle.

Pour l'algorithme PBGD, nous appliquons une méthode améliorée de la descente de gradient, dont la vitesse de convergence ne repose pas sur la sélection d'un taux d'apprentissage. La *descente de gradient conjugué* est présentée par l'algorithme 3. Elle introduit deux modifications à la descente de gradient décrite précédemment :

- La direction de descente $\mathbf{d}_t$ à l'itération $t$ est donnée par une combinaison linéaire du gradient au point courant et de la direction de descente à l'itération précédente (équation (4.10)).
- L'ampleur du «saut» effectué à l'itération $t$ est déterminée par une recherche linéaire dans la direction $\mathbf{d}_t$ afin de trouver le point qui minimise la fonction $f$ (équation (4.11)).

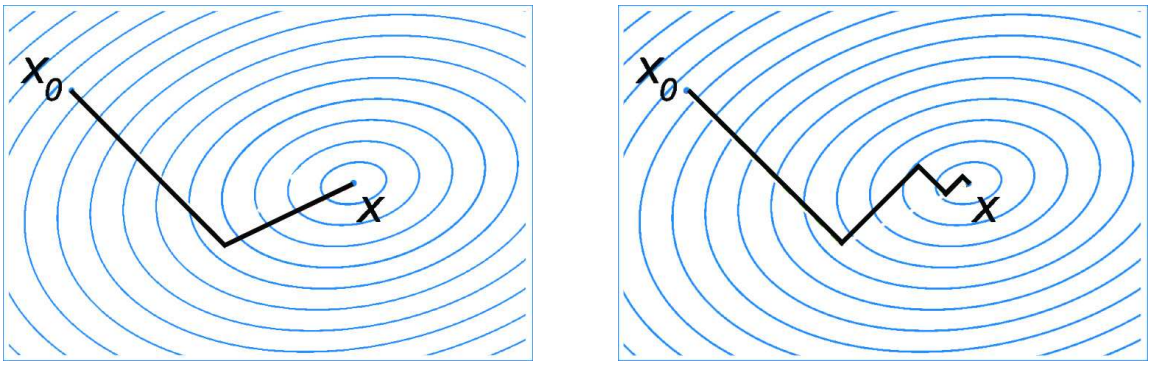
Il existe plusieurs équations dictant la valeur de la variable $\mu_t$ de l'équation (4.10). Notre choix se porte sur la forme *Polak-Ribière* (équation (4.9)), qui est reconnue pour son efficacité empirique [3].

Il est montré que, en présence d'une fonction $f$ quadratique à $n$ variables, cette méthode converge vers le minimum de la fonction en un maximum de $n$ itérations [19]. Sommairement, ce phénomène repose sur le fait que l'ensemble des directions choisies lors de la descente forme un ensemble de vecteurs linéairement indépendants qui sont *mutuellement conjugués* selon la matrice hessienne $\mathbf{H}(f)$ [1] :

$$\mathbf{d}_i^T \, [\mathbf{H}(f)] \, \mathbf{d}_j = 0 \quad \forall\, i, j \in \{0, \ldots, n-1\} \quad i \neq j \,.$$

La descente de gradient conjugué effectue alors des «sauts» optimaux dans chacune des directions pour atteindre le minimum de la fonction quadratique (tel qu'illustré par l'exemple de la figure 4.1).

[1] La matrice hessienne est carrée. L'élément $i, j$ est donné par $[\mathbf{H}(f)]_{i,j} \stackrel{\text{def}}{=} \frac{\partial^2 f}{\partial x_i \partial x_j}$. La matrice hessienne associée à une fonction quadratique n'est constituée que de termes constants.




(a) Descente de gradient conjugué (b) Descente de gradient « plus forte pente »

FIG. 4.1: Cet exemple illustre le fait que la descente de gradient conjugué d'une fonction quadratique à deux dimensions converge en deux itérations. Comparativement, une descente de gradient de type « plus forte pente » (qui favorise l'inverse du gradient comme direction de descente) initialisée au point $X_0$ converge en 5 itérations sur ce problème précis. (Image : Wikipedia)

---

**Algorithme 3** DescenteGradient( $\nabla f$, $\mathbf{x}_0$, $\epsilon_{abs}$ )

---

**Entrée :** $\nabla f$, l'expression du gradient de la fonction $f$ à minimiser.
**Entrée :** $\mathbf{x}_0$, le point initial où débute la descente de gradient.
**Entrée :** $\epsilon_{abs}$, le critère d'arrêt de l'algorithme.

Initialiser $t \leftarrow 0$ .
**tant que** $\|\nabla f(\mathbf{x}_t)\| \geq \epsilon_{abs}$ **faire**
Si $t \mod n = 0$, poser $\mu_t \leftarrow 0$. Sinon, calculer :

$$\mu_t \leftarrow \frac{\nabla f(\mathbf{x}_t) \cdot [\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_{t-1})]}{\|\nabla f(\mathbf{x}_{t-1})\|^2} \,. \tag{4.9}$$

Déterminer la direction de descente :

$$\mathbf{d}_t \leftarrow -\nabla f(\mathbf{x}_t) + \mu_t \mathbf{d}_{t-1} \,. \tag{4.10}$$

Effectuer un «saut» dans la direction $\mathbf{d}_t$ :

$$\mathbf{x}_{t+1} \leftarrow \underset{\mathbf{x}_t + \eta \mathbf{d}_t}{\operatorname{argmin}} \{f(\mathbf{x}_t + \eta \mathbf{d}_t) | \eta > 0\} \,. \tag{4.11}$$

$t \leftarrow t + 1$ .
**fin tant que**

**Sortie :** $\mathbf{x}_{t-1}$ .

---

En présence d'un problème non quadratique, la descente de gradient conjugué estime donc localement la fonction à minimiser par une fonction quadratique[2]. Afin d'éviter que cette estimation ne se déteriore au fil des itérations, la pratique courante consiste à «redémarrer» la descente de gradient à toutes les $n$ itérations (où $n$ est le nombre de variables de la fonction minimisée).

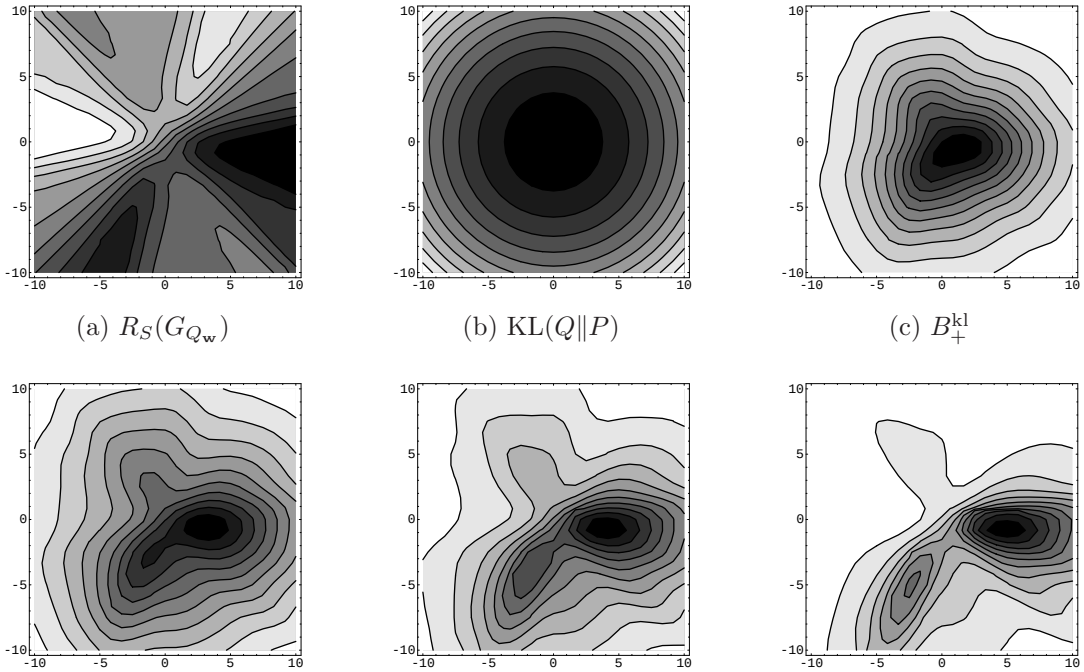
La convergence de la méthode de descente de gradient conjugué peut être lente sur des problèmes mal conditionnés, où l'approximation quadratique de la fonction à optimiser n'est pas appropriée. En contrepartie, cette méthode est simple d'implémentation et d'utilisation.

### 4.5.2 Minimums locaux et redémarrages aléatoires

Les expressions mathématiques des bornes PAC-Bayes que l'on désire optimiser sont des fonctions non convexes (en raison de la forme du risque de Gibbs). Conséquemment, ces fonctions possèdent possiblement plusieurs minimums locaux. Quelques expérimentations empiriques ont permis de constater qu'il en est bien ainsi. Il fut observé que la borne Langford-Seeger possède habituellement un seul minimum local. Toutefois, la borne hyperparamétrée présente plusieurs minimums locaux lorsque la valeur de l'hyperparamètre $C$ est élevée, comme l'illustre l'exemple de la figure 4.2. Ces considérations font de la minimisation de la borne hyperparamétrée un problème d'optimisation davantage complexe que celui de la minimisation de la borne Langford-Seeger. Cependant, nous verrons aux sections 4.6.2 et 4.6.3 que la borne hyperparamétrée permet de formuler des algorithmes d'apprentissage s'adaptant davantage aux particularités de l'ensemble d'entraînement.

Lors de l'optimisation d'une fonction possédant plusieurs minimums, on désire idéalement trouver le minimum global, c'est-à-dire le minimum absolu de la fonction. Hélas, les méthodes de descente de gradient convergent vers un minimum local de la fonction à optimiser. Le minimum trouvé par l'algorithme dépend grandement du point initial $\mathbf{x}_0$. En présence d'une fonction possédant plusieurs minimums locaux, il faut effectuer plusieurs descentes de gradient, chacune possédant un point de départ différent, tel que décrit par l'algorithme 4. À défaut d'avoir la certitude d'obtenir le minimum global de la fonction, nous trouvons ainsi un minimum local dont la valeur est suffisamment petite.

[2]Notons que, lorsque l'expression de la fonction le permet, une stratégie équivalente serait d'estimer la fonction $f$ au point $\mathbf{x}_t$ par une fonction quadratique en calculant la matrice hessienne $[\mathbf{H}(f)](\mathbf{x}_t)$ et d'effectuer un déplacement vers le minimum de cette fonction quadratique. On évite cette technique puisque le calcul de la matrice hessienne nécessiterait un temps de calcul exagéré.



(a) $R_S(G_{Q_{\mathbf{w}}})$ (b) $\mathrm{KL}(Q\|P)$ (c) $B_+^{\mathrm{kl}}$

(d) $B_+^C$ $(C = 5)$ (e) $B_+^C$ $(C = 10)$ (f) $B_+^C$ $(C = 20)$

FIG. 4.2: Courbes de niveaux des bornes PAC-Bayes sur un ensemble jouet contenant 50 exemples générés aléatoirement. Le niveau de gris correspond à la valeur mesurée pour un classificateur à noyau linéaire sans biais caractérisé par un vecteur $\mathbf{w} = (x_1, x_2)$. Les bornes PAC-Bayes combinent le risque empirique de Gibbs (4.2a) et la divergence de Kullback-Leibler (4.2b). La pondération de ces quantités est déteminée automatiquement dans la borne Langford-Seeger (4.2c) et par le biais de l'hyperparamètre $C$ dans la borne hyperparamétrée (4.2d, 4.2e, 4.2f).

---

**Algorithme 4** DescenteGradientMultiple( $\nabla f$, $\kappa$, $\omega$, $\epsilon_{abs}$ )

---

**Entrée :** $\nabla f$, l'expression du gradient de la fonction $f$ à minimiser.
**Entrée :** $\kappa$, le nombre de descentes de gradient à effectuer.
**Entrée :** $\omega$, la norme maximale du vecteur initial.
**Entrée :** $\epsilon_{abs}$, le critère d'arrêt de l'algorithme.

**pour** $i = 1, \ldots, \kappa$ **faire**
  Initialiser $\mathbf{a}$ aléatoirement tel que $0 \leq \|\mathbf{a}\| \leq \omega$ .
  $\mathbf{x}_i \leftarrow$ DescenteGradient$(\nabla f, \mathbf{a}, \epsilon_{abs})$ .
**fin pour**

**Sortie :** $\underset{\mathbf{x}_i}{\operatorname{argmin}} \{f(\mathbf{x}_i) \,|\, i = 1, \ldots, \kappa\}$ .

---

Ainsi, l'algorithme de descente de gradient conjugué est exécuté $\kappa$ fois. À chaque exécution, le point initial est déterminé en sélectionnant aléatoirement un vecteur dans un rayon $\omega$ de l'origine. Parmi l'ensemble des résultats obtenus par les $\kappa$ exécutions, on retient celui pour lequel la valeur de $f$ est minimale.

### 4.5.3 Calcul des gradients dans l'espace primal

Dans la version primale de l'algorithme PBGD, nous désirons obtenir le gradient de la borne $B$ en fonction du vecteur $\mathbf{w}$. Celui-ci est donné par :

$$\nabla_{\mathbf{w}} B = \left( \frac{\partial B}{\partial w_1}, \frac{\partial B}{\partial w_2}, \ldots, \frac{\partial B}{\partial w_{n'}} \right).$$

Calculons d'abord une dérivée partielle $\frac{\partial B}{\partial w_k}$ de la borne Lanford-Seeger. Comme il n'est pas possible d'exprimer la valeur de $B$ directement en une fonction de $\mathbf{w}$, nous calculons d'abord la dérivée de chaque terme de l'égalité (4.1) :

$$\frac{\partial}{\partial w_k} \mathrm{kl}\left(R_S(G_{Q_{\mathbf{w}}}) || B\right) = \frac{\partial}{\partial w_k} \left[ \frac{1}{m} \left( \mathrm{KL}(Q_{\mathbf{w}}, P_{\mathbf{w}',\sigma}) + \ln \frac{\xi(m)}{\delta} \right) \right].$$

Pour calculer le terme de droite, la notation est simplifiée en posant $\mathrm{KL} = \mathrm{KL}(Q_{\mathbf{w}}, P_{\mathbf{w}',\sigma})$ :

$$\frac{\partial}{\partial w_k} \left[ \frac{1}{m} \left( \mathrm{KL} + \ln \frac{\xi(m)}{\delta} \right) \right] = \frac{1}{m} \frac{\partial \mathrm{KL}}{\partial w_k}.$$

Pour calculer le terme de gauche, la notation est simplifiée en posant $R_S = R_S(G_{Q_{\mathbf{w}}})$ :

$$\begin{aligned}
\frac{\partial}{\partial w_k} \mathrm{kl}\left(R_S || B\right) &= \frac{\partial}{\partial w_k} \left[ R_S \ln \frac{R_S}{B} + (1 - R_S) \ln \frac{1 - R_S}{1 - B} \right] \\
&= \left[ \ln \frac{R_S}{B} \right] \frac{\partial R_S}{\partial w_k} + \left[ \frac{R_S}{B} \right] \frac{\partial B}{\partial w_k} - \left[ \ln \frac{1 - R_S}{1 - B} \right] \frac{\partial R_S}{\partial w_k} + \left[ \frac{1 - R_S}{1 - B} \right] \frac{\partial B}{\partial w_k} \\
&= \left[ \ln \frac{R_S(1 - B)}{B(1 - R_S)} \right] \frac{\partial R_S}{\partial w_k} + \left[ \frac{B - R_S}{B(1 - B)} \right] \frac{\partial B}{\partial w_k}.
\end{aligned}$$

On peut ensuite isoler $\frac{\partial B}{\partial w_k}$ :

$$\begin{aligned}
\left[ \frac{B - R_S}{B(1 - B)} \right] \frac{\partial B}{\partial w_k} &= \frac{1}{m} \frac{\partial \mathrm{KL}}{\partial w_k} - \left[ \ln \frac{R_S(1 - B)}{B(1 - R_S)} \right] \frac{\partial R_S}{\partial w_k} \\
\iff \qquad \frac{\partial B}{\partial w_k} &= \frac{1}{m} \frac{B(1 - B)}{B - R_S} \left[ \frac{\partial \mathrm{KL}}{\partial w_k} + \left( \ln \frac{B(1 - R_S)}{R_S(1 - B)} \right) \cdot m \frac{\partial R_S}{\partial w_k} \right].
\end{aligned}$$

Pour un vecteur $\mathbf{w}$ donné, le gradient de la borne hyperparamétrée (équation (4.3)) possède la même direction que le gradient de la fonction $\beta$ défini par l'équation (4.4). La dérivée de cette dernière est donnée par :

$$\begin{aligned}\frac{\partial \beta}{\partial w_k} &= \frac{\partial}{\partial w_k}\left[C \cdot m R_S + \mathrm{KL}\right] \\ &= C \cdot m \frac{\partial R_S}{\partial w_k} + \frac{\partial \mathrm{KL}}{\partial w_k} \ .\end{aligned}$$

Calculons maintenant $\Phi'(x)$, la dérivée de la fonction $\Phi$ (équation (3.16)) évaluée au point $x$ :

$$\Phi'(x) = \frac{\partial}{\partial x}\Phi(x) = -\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \ . \tag{4.12}$$

On trouve la dérivée partielle du risque empirique de Gibbs en dérivant l'équation (4.5) :

$$\begin{aligned}\frac{\partial R_S}{\partial w_k} &= \frac{1}{m}\sum_{i=1}^{m} \Phi'\Big(\mathbf{w} \cdot \boldsymbol{\psi}_i\Big) \frac{\partial \mathbf{w} \cdot \boldsymbol{\psi}_i}{\partial w_k} \\ &= \frac{1}{m}\sum_{i=1}^{m} \Phi'\Big(\mathbf{w} \cdot \boldsymbol{\psi}_i\Big) \psi_{i,k} \ .\end{aligned}$$

Enfin, calculons l'expression de la dérivée partielle de la divergence de Kullback-Leibler (définie à l'équation (4.6)) :

$$\begin{aligned}\frac{\partial \mathrm{KL}}{\partial w_k} &= \frac{1}{2}\left[\frac{\partial \|\mathbf{w}_\perp\|^2}{\partial w_k} + \frac{1}{\sigma^2}\frac{\partial \|\mathbf{w}_\parallel\|^2}{\partial w_k}\right] \\ &= \frac{1}{2}\left[\frac{\partial \|\mathbf{w} - \mathbf{w}'\|^2}{\partial w_k} + \left(\frac{1}{\sigma^2} - 1\right)\frac{\partial \|\mathbf{w}_\parallel\|^2}{\partial w_k}\right] \\ &= \frac{1}{2}\left[2(w_k - w'_k) + \left(\frac{1}{\sigma^2} - 1\right)\frac{-2w'_k}{\|\mathbf{w}'\|}\frac{(\mathbf{w} - \mathbf{w}') \cdot \mathbf{w}'}{\|\mathbf{w}'\|}\right] \\ &= w_k - w'_k + w'_k\left(1 - \frac{1}{\sigma^2}\right)\left[\frac{\mathbf{w} \cdot \mathbf{w}'}{\|\mathbf{w}'\|^2} - 1\right] \\ &= w_k - w'_k\left[\frac{\mathbf{w} \cdot \mathbf{w}'}{\|\mathbf{w}'\|^2}\left(1 - \frac{1}{\sigma^2}\right) + \frac{1}{\sigma^2}\right] .\end{aligned}$$

### 4.5.4 Calcul des gradients dans l'espace dual

Dans la version duale de l'algorithme PBGD, nous désirons obtenir le gradient de la borne $B$ en fonction du vecteur dual $\boldsymbol{\alpha}$, c'est-à-dire :

$$\nabla_{\boldsymbol{\alpha}} B = \left(\frac{\partial B}{\partial \alpha_1}, \frac{\partial B}{\partial \alpha_2}, \ldots, \frac{\partial B}{\partial \alpha_m}\right) .$$

En suivant les mêmes étapes de calcul qu'à la section précédente, nous obtenons d'abord l'expression d'une dérivée $\frac{\partial B}{\partial \alpha_k}$ pour la borne Langford-Seeger :

$$\frac{\partial B}{\partial \alpha_k} = \frac{1}{m}\frac{B(1-B)}{B-R_S}\left[\frac{\partial \mathrm{KL}}{\partial \alpha_k} + \left(\ln\frac{B(1-R_S)}{R_S(1-B)}\right)\cdot m\frac{\partial R_S}{\partial \alpha_k}\right].$$

Nous obtenons ensuite l'expression d'une dérivée $\frac{\partial \beta}{\partial \alpha_k}$ associée à la borne hyperparamétrée :

$$\frac{\partial \beta}{\partial \alpha_k} \;=\; C\cdot m\frac{\partial R_S}{\partial \alpha_k} + \frac{\partial \mathrm{KL}}{\partial \alpha_k}\,.$$

Nous trouvons la dérivée partielle du risque empirique de Gibbs en dérivant l'équation (4.7) (La fonction $\Phi'$ est définie par l'équation (4.12)) :

$$\begin{aligned}\frac{\partial R_S}{\partial \alpha_k} &= \frac{1}{m}\sum_{i=1}^{m}\Phi'\left(\frac{\sum_{j=1}^m \alpha_j G_{j,i}}{\sqrt{G_{i,i}}}\right)\frac{\partial}{\partial \alpha_k}\left[\frac{\sum_{j=1}^m \alpha_j G_{j,i}}{\sqrt{G_{i,i}}}\right]\\ &= \frac{1}{m}\sum_{i=1}^{m}\Phi'\left(\frac{\sum_{j=1}^m \alpha_j G_{j,i}}{\sqrt{G_{i,i}}}\right)\frac{G_{k,i}}{\sqrt{G_{i,i}}}\,.\end{aligned}$$

La divergence de Kullback-Leibler est définie par l'équation (4.8). Sa dérivée devient :

$$\begin{aligned}\frac{\partial \mathrm{KL}}{\partial \alpha_k} &= \frac{1}{2}\left[\frac{\partial\|\mathbf{w}-\mathbf{w}'\|^2}{\partial \alpha_k} + \left(\frac{1}{\sigma^2}-1\right)\frac{\partial\|\mathbf{w}_\|\|^2}{\partial \alpha_k}\right]\\
&= \frac{1}{2}\left[\left(\frac{\partial}{\partial \alpha_k}\sum_{i=1}^m\sum_{j=1}^m(\alpha_i-\alpha_i')(\alpha_j-\alpha_j')G_{i,j}\right)+2\left(\frac{1}{\sigma^2}-1\right)\frac{(\mathbf{w}-\mathbf{w}')\cdot\mathbf{w}'}{\|\mathbf{w}'\|}\left(\frac{\partial}{\partial \alpha_k}\frac{\mathbf{w}\cdot\mathbf{w}'}{\|\mathbf{w}'\|}\right)\right]\\
&= \frac{1}{2}\left[2\sum_{i=1}^m(\alpha_i-\alpha_i')G_{i,k} + 2\left(\frac{1}{\sigma^2}-1\right)\frac{(\mathbf{w}-\mathbf{w}')\cdot\mathbf{w}'}{\|\mathbf{w}'\|}\left(\frac{\partial}{\partial \alpha_k}\frac{\sum_{i=1}^m\sum_{j=1}^m\alpha_i\alpha_j' G_{i,j}}{\|\mathbf{w}'\|}\right)\right]\\
&= \sum_{i=1}^m(\alpha_i-\alpha_i')G_{i,k} + \left(\frac{1}{\sigma^2}-1\right)\frac{(\mathbf{w}-\mathbf{w}')\cdot\mathbf{w}'}{\|\mathbf{w}'\|^2}\sum_{i=1}^m\alpha_i' G_{i,k}\\
&= \sum_{i=1}^m\left[\alpha_i-\alpha_i'+\alpha_i'\left(\frac{1}{\sigma^2}-1\right)\left[\frac{\mathbf{w}\cdot\mathbf{w}'}{\|\mathbf{w}'\|^2}-1\right]\right]G_{i,k}\\
&= \sum_{i=1}^m\left[\alpha_i-\alpha_i'\left[\frac{\sum_{j=1}^m\sum_{l=1}^m\alpha_j\alpha_l' G_{j,l}}{\sum_{j=1}^m\sum_{l=1}^m\alpha_j'\alpha_l' G_{j,l}}\left(1-\frac{1}{\sigma^2}\right)+\frac{1}{\sigma^2}\right]\right]G_{i,k}\,.\end{aligned}$$

## 4.6 Algorithmes d'apprentissage

Les sections précédentes détaillent un certain nombre d'outils mathématiques dans le but d'élaborer un algorithme de minimisation de la borne PAC-Bayes, que l'on nomme PBGD. On peut imaginer plusieurs techniques pour combiner ces outils, chacune menant à une variante de l'algorithme d'apprentissage PBGD. Cette section présente trois de ces variantes, désignées par PBGD1, PBGDcv et PBGD2. Pour éviter les ambiguïtés, chaque version est détaillée à l'aide d'un pseudo-code.

Les expressions mathématiques à minimiser, de même que leur gradient, proviennent des résultats des calculs obtenus dans les deux sections précédentes. Ces expressions sont formulées directement si on représente le vecteur $\mathbf{w}$ explicitement (espace primal) ou implicitement à l'aide d'un vecteur $\boldsymbol{\alpha}$ (espace dual). Dans le texte qui suit, nous désignons par le symbole $(\mathcal{P})$ les expressions propres à l'espace primal et par le symbole $(\mathcal{D})$ celles propres à l'espace dual.

### 4.6.1 PBGD1 : Apprentissage en une phase

La première version de l'algorithme envisagée est nommée *PBGD1*. Elle consiste à minimiser la borne Langford-Seeger à l'aide d'un prior non informatif. Ainsi, tous les exemples d'entraînement sont utilisés pour déterminer le vecteur $\mathbf{w}$. Il s'agit donc de minimiser la valeur $B$ dans le système d'équations suivant :

$$\begin{aligned} \mathrm{kl}\left(R_S(G_{Q_\mathbf{w}})||B\right) &= \frac{1}{m}\left[\mathrm{KL}(Q_\mathbf{w}||P_{\mathbf{0},1}) + \ln\left(\frac{\xi(m)}{\delta}\right)\right], \\ B &> R_S(G_{Q_\mathbf{w}})\,, \end{aligned} \tag{4.13}$$

où :

$$(\mathcal{P}) \quad R_S(G_{Q_\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m}\Phi\left(\mathbf{w}\cdot\boldsymbol{\psi}_i\right) \quad \text{et} \quad \mathrm{KL}(Q_\mathbf{w}||P_{\mathbf{0},1}) = \tfrac{1}{2}\|\mathbf{w}\|^2\,,$$

$$(\mathcal{D}) \quad R_S(G_{Q_\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m}\Phi\left(\frac{\sum_{j=1}^{m}\alpha_j G_{j,i}}{\sqrt{G_{i,i}}}\right) \quad \text{et} \quad \mathrm{KL}(Q_\mathbf{w}||P_{\mathbf{0},1}) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j G_{i,j}\,.$$

L'algorithme 5 présente le pseudo-code de PBGD1. Il s'agit de la version de l'algorithme nécessitant le moins d'hyperparamètres.

**Algorithme 5** PBGD1$(S, \delta, \kappa, \omega, \epsilon_{abs})$

**Entrée :** $S$, l'ensemble d'entraînement.

**Entrée :** $\delta$, le paramètre de confiance.

**Entrée :** $(\kappa, \omega, \epsilon_{abs})$, les paramètres de la descente de gradient conjugué.

Minimiser l'équation (4.13) : $\mathbf{w} \leftarrow$DescenteGradientMultiple$(\nabla B, \kappa, \omega, \epsilon_{abs})$ .
Les composantes du gradient $\nabla B$ sont données par :

$$(\mathcal{P})\ \frac{\partial B}{\partial w_k} = \frac{1}{m}\frac{B(1-B)}{B-R_S}\left[w_k + \left(\ln\frac{B(1-R_S)}{R_S(1-B)}\right)\sum_{i=1}^{m}\Phi'\Big(\mathbf{w}\cdot\boldsymbol{\psi}_i\Big)\psi_{i,k}\right],$$

$$(\mathcal{D})\ \frac{\partial B}{\partial \alpha_k} = \frac{1}{m}\frac{B(1-B)}{B-R_S}\left[\sum_{i=1}^{m}\alpha_k G_{i,k} + \left(\ln\frac{B(1-R_S)}{R_S(1-B)}\right)\sum_{i=1}^{m}\Phi'\left(\frac{\sum_{j=1}^{m}\alpha_j G_{j,i}}{\sqrt{G_{i,i}}}\right)\frac{G_{k,i}}{\sqrt{G_{i,i}}}\right].$$

où $B$ est trouvé par la méthode de Newton (algorithme 2 de la section 4.4.1).

**Sortie :** $\mathbf{w}$ ou $\boldsymbol{\alpha}$, le classificateur final dans sa représentation primale ou duale.

**Sortie :** $B(\mathbf{w})$, la borne sur le risque correspondante.

### 4.6.2 PBGDcv : Apprentissage par validation croisée

À l'instar de l'algorithme PBGD1, la seconde version de l'algorithme a recours à un prior non informatif, mais minimise cette fois la borne hyperparamétrée. Il s'agit de minimiser la valeur de $\beta$ donnée par l'expression suivante :

$$\begin{aligned}(\mathcal{P})\quad \beta = &\ C\sum_{i=1}^{m}\Phi\left(\mathbf{w}\cdot\boldsymbol{\psi}_i\right) + \frac{1}{2}\|\mathbf{w}\|^2,\\ (\mathcal{D})\quad \beta = &\ C\sum_{i=1}^{m}\Phi\left(\frac{\sum_{j=1}^{m}\alpha_j G_{j,i}}{\sqrt{G_{i,i}}}\right) + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j G_{i,j}\ .\end{aligned} \tag{4.14}$$

La valeur $C$ est un hyperparamètre de l'algorithme. L'algorithme 6 présente la procédure de minimisation pour un $C$ donné. Afin de sélectionner une bonne valeur pour cet hyperparamètre, nous devons effectuer de la sélection de modèle. Il serait possible d'exécuter l'algorithme avec plusieurs valeurs de $C$ et de retenir celle permettant d'obtenir la borne hyperparamétrée minimale. Cependant, avec une bonne sélection de valeurs, le classificateur obtenu sera le même que celui obtenu par la minimisation de la borne Langford-Seeger de l'algorithme PBGD1. Comme expliqué à la section 3.1.4, la valeur de la borne serait possiblement légèrement inférieure à celle obtenue par PBGD1, mais au prix de plusieurs exécutions de l'algorithme. Il n'est donc pas justifié de recourir à cette méthode de sélection de modèle par la borne PAC-Bayes alors que l'algorithme PBGD1 nécessite une seule exécution tout en produisant un résultat équivalent.

L'algorithme présenté ici explore la possibilité d'utiliser la technique de la sélection de modèle par validation croisée, telle que décrite à la section 2.4.1, plutôt que de miser uniquement sur une procédure de minisation de la borne. Cette méthode, nommée *PBGDcv*, est décrite par l'algorithme 7. La valeur de $C$ est sélecrionnée parmi un ensemble $\mathcal{C}$ de valeurs possibles par validation croisée $k$-fois. On remarque dans l'équation (4.14) que l'hyperparamètre $C$ permet de modifier l'importance du risque empirique de Gibbs par rapport à la norme du vecteur $\mathbf{w}$. Une grande valeur de $C$ favorise la diminution du risque empirique de Gibbs, caractérisé par la fonction de perte sigmoïdale. Dans ce cas, la valeur de la borne hyperparamétrée associée sera aussi très grande et peu informative, voir triviale. Conséquemment, l'algorithme ne retourne pas de garantie théorique de pair avec le classificateur construit. Ainsi, PBGDcv s'avère un algorithme inspiré de la borne PAC-Bayes plutôt qu'un algorithme de minimisation de borne. L'analyse des résultats de cet algorithme s'annonce néanmoins intéressante pour deux raisons :

- Dans l'algorithme PBGD1, la borne Langford-Seeger dicte elle-même, par le biais de la fonction $\mathrm{kl}(\cdot\|\cdot)$, le compromis entre la norme de $\mathbf{w}$ et le risque empirique de Gibbs. L'algorithme PBGDcv sélectionne ce compromis par validation croisée. La comparaison des résultats des algorithmes PBGD1 et PBGDcv permettra donc de juger si le compromis suggéré par la borne Langford-Seeger est adéquat.

- Le compromis dicté par l'équation (4.14) est fort semblable à celui préconisé par le SVM (voir l'équation (2.13)), qui minimise une combinaison du hinge loss et de la norme du vecteur $\mathbf{w}$. La comparaison de l'algorithme PBGDcv avec les SVM sera donc intéressante afin de déterminer si la fonction de risque sigmoïdale est préférable au hinge loss.

En résumé, PBGDcv évacue la plupart des avantages recherchés chez les algorithmes de minimisation d'une borne, car il ne fournit pas de garantie sur le vrai risque et il effectue une sélection de modèle par validation croisée. Ainsi, son exécution nécessite la spécification d'un ensemble $\mathcal{C}$ de valeurs pour l'hyperparamètre $C$ et le nombre $k$ utilisé par la validation croisée $k$-fois. Cependant, comme la procédure de validation croisée est généralement une méthode efficace de sélection de modèle, on peut s'attendre à ce qu'elle produise de bons classificateurs. Son étude permettra de surcroît d'évaluer la pertinence du recours au risque sigmoïdal ainsi que l'optimalité du compromis suggéré par la borne Langford-Seeger.

**Algorithme 6** PBGDcvUneFois$(S, C, \kappa, \omega, \epsilon_{abs})$

**Entrée :** $S$, l'ensemble d'entraînement.

**Entrée :** $C$, le paramètre de la borne hyperparamétrée.

**Entrée :** $(\kappa, \omega, \epsilon_{abs})$, les paramètres de la descente de gradient conjugué.

Minimiser l'équation (4.14) : $\mathbf{w} \leftarrow$DescenteGradientMultiple$(\nabla\beta, \kappa, \omega, \epsilon_{abs})$ .
Les composantes du gradient $\nabla\beta$ sont données par :

$$(\mathcal{P}) \quad \frac{\partial\beta}{\partial w'_k} = C\sum_{i=1}^{m} \Phi'\Big(\mathbf{w}' \cdot \boldsymbol{\psi}_i\Big)\psi_{i,k} + w'_k \ ,$$

$$(\mathcal{D}) \quad \frac{\partial\beta}{\partial \alpha'_k} = C\sum_{i=1}^{m} \Phi'\left(\frac{\sum_{j=1}^{m}\alpha'_j G_{j,i}}{\sqrt{G_{i,i}}}\right)\frac{G_{k,i}}{\sqrt{G_{i,i}}} + \sum_{i=1}^{m'}\alpha'_k G_{i,k} \ .$$

**Sortie :** $\mathbf{w}$ ou $\boldsymbol{\alpha}$, le classificateur final dans sa représentation primale ou duale.

**Algorithme 7** PBGD1cv$(S, \mathcal{C}, k, \kappa, \omega, \epsilon_{abs})$

**Entrée :** $S$, l'ensemble d'entraînement.

**Entrée :** $\mathcal{C}$, un ensemble de valeurs pour le paramètre C.

**Entrée :** $k$, le nombre de divisions de l'ensemble $S$ lors de la validation croisée.

**Entrée :** $(\kappa, \omega, \epsilon_{abs})$, les paramètres de la descente de gradient conjugué.

Diviser aléatoirement $S$ en $k$ sous-ensembles distincts : $S = T_1 \cup T_2 \cup \ldots \cup T_k$ .
Initialiser $R(\overline{C}) \leftarrow 1$ .
**pour tout** $C \in \mathcal{C}$ **faire**
  $i \leftarrow 0$ ; $R(C) \leftarrow 0$ .
  **tant que** $i \leq k$ et $R(C) < R(\overline{C})$ **faire**
    $i \leftarrow i + 1$ ; $S' \leftarrow S \setminus T_i$ .
    Trouver $\mathbf{w}$ par l'algorithme 6 : $\mathbf{w} \leftarrow$PBGDcvUneFois$(S', C, \kappa, \omega, \epsilon_{abs})$ .
    Mettre à jour le risque de validation croisée : $R(C) \leftarrow R(C) + \frac{1}{k}R_{T_i}(h_{\mathbf{w}})$ .
  **fin tant que**
  **si** $R(C) < R(\overline{C})$ **alors**
    $\overline{C} \leftarrow C$ .
  **fin si**
**fin pour**
Trouver le classificateur final : $\overline{\mathbf{w}} \leftarrow$PBGDcvUneFois$(S, \overline{C}, \kappa, \omega, \epsilon_{abs})$ .

**Sortie :** $\overline{\mathbf{w}}$ ou $\overline{\boldsymbol{\alpha}}$, le classificateur final dans sa représentation primale ou duale.

### 4.6.3 PBGD2 : Apprentissage en deux phases

La dernière version de l'algorithme envisagée tente de conserver les avantages des algorithmes PBGD1 et PBGDcv. D'une part, on désire éviter la sélection de modèle par validation croisée et produire un classificateur possédant une bonne garantie sur le vrai risque. D'autre part, on désire ajuster le compromis entre la diminution du risque empirique de Gibbs et la norme du vecteur $\mathbf{w}$ selon les particularités de l'ensemble d'entraînement. Pour ce faire, on tire profit de la stratégie d'apprentissage d'un prior décrite à la section 4.3. L'algorithme prend en paramètre un ensemble $\mathcal{C}$ de valeurs possibles pour le paramètre $C$. Le processus d'apprentissage se divise en deux phases pour chaque valeur de $C$. Ainsi, cet algorithme est nommé *PBGD2*.

La première phase d'apprentissage consiste à apprendre un prior $\mathbf{w}'$ à l'aide d'un sous-ensemble $S_A$ formé par une proportion $p$ des exemples de l'ensemble d'entraînement $S$. On minimise la borne hyperparamétrée avec une valeur de $C$ donnée. On procède donc à la minimisation de la valeur $\beta$ en calculant le risque empirique de Gibbs sur les exemples appartenant à $S_A$ :

$$(\mathcal{P}) \quad \beta = \; C \sum_{\mathbf{z}_i \in S_A} \Phi\left(\mathbf{w}' \cdot \boldsymbol{\psi}_i\right) + \frac{1}{2}\|\mathbf{w}'\|^2 \,,$$

(4.15)

$$(\mathcal{D}) \quad \beta = \; C \sum_{\mathbf{z}_i \in S_A} \Phi\left(\frac{\sum_{j=1}^{m} \alpha'_j G_{j,i}}{\sqrt{G_{i,i}}}\right) + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha'_i \alpha'_j G_{i,j}$$
$$\text{avec } \alpha'_k = 0 \text{ lorsque } \mathbf{z}_k \notin S_A \,.$$

Rappelons que, dans la version duale de l'algorithme, le classificateur linéaire est exprimé par une combinaison linéaire des exemples d'entraînement. Comme l'apprentissage du prior considère les exemples du sous-ensemble $S_A$, le vecteur $\mathbf{w}'$ doit être une combinaison linéaire des exemples de ce sous-ensemble uniquement.

La seconde phase d'apprentissage produit un classificateur $\mathbf{w}$ (le posterior) associé à une valeur de $C$ donnée. Pour ce faire, on minimise la borne Langford-Seeger calculée sur les exemples du sous-ensemble $S_B = S \backslash S_A$ avec un prior informatif caractérisé par le vecteur $\mathbf{w}'$ obtenu par la première phase d'apprentissage. L'hyperparamètre $\sigma$ détermine le degré d'élongation de ce prior. En définitive, la seconde phase d'apprentissage consiste à minimiser la valeur $B$ dans le système d'équation suivant :

$$\text{kl}\left(R_{S_B}(G_{Q_\mathbf{w}})||B\right) = \frac{1}{m_B}\left[\text{KL}(Q_\mathbf{w}||P_{\mathbf{w}',\sigma}) + \ln\left(\frac{\xi(m_B)}{\delta}\right) + \ln|\mathcal{C}|\right], \tag{4.16}$$

où :

$$(\mathcal{P}) \quad R_{S_B}(G_{Q_\mathbf{w}}) = \frac{1}{m_B}\sum_{\mathbf{z}_i \in S_B} \Phi\left(\mathbf{w}\cdot\boldsymbol{\psi}_i\right) \qquad \text{et } \text{KL}(Q_\mathbf{w}||P_{\mathbf{w}',\sigma}) \text{ est défini par (4.6),}$$

$$(\mathcal{D}) \quad R_{S_B}(G_{Q_\mathbf{w}}) = \frac{1}{m_B}\sum_{\mathbf{z}_i \in S_B} \Phi\left(\frac{\sum_{j=1}^{m}\alpha_j G_{j,i}}{\sqrt{G_{i,i}}}\right) \quad \text{et } \text{KL}(Q_\mathbf{w}||P_{\mathbf{w}',\sigma}) \text{ est défini par (4.8).}$$

Le terme $\ln|\mathcal{C}|$ dans l'équation précédente provient du fait que chaque borne est calculée avec un niveau de confiance de $\frac{\delta}{|\mathcal{C}|}$. Cela est justifié par la borne de l'union.

On obtient donc un classificateur pour chaque valeur de $C \in \mathcal{C}$. De tous ces classificateurs, l'algorithme sélectionne celui qui possède la borne la plus faible à la suite de la deuxième phase d'apprentissage. La deuxième phase d'apprentissage sert donc à la fois à valider le résultat obtenu par la première phase sur le sous-ensemble $S_A$, à l'ajuster en fonction des exemples appartenant au deuxième sous-ensemble $S_B$ et à fournir une borne sur le risque d'apprentissage du classificateur généré.

L'algorithme 8 présente le pseudo-code de PBGD2. Par rapport à PBGD1, cet algorithme nécessite plusieurs hyperparamètres, soit l'ensemble $\mathcal{C}$, la proportion $p$ et le paramètre d'élongation du prior $\sigma$. Dans le chapitre suivant, nous étudierons l'influence de ces hyperparamètres afin de déterminer lesquels conviennent le mieux à l'algorithme.

**Algorithme 8** PBGD2($S, \mathcal{C}, p, \sigma, \delta, \kappa, \omega, \epsilon_{abs}$)

**Entrée :** $S$, l'ensemble d'entraînement.
**Entrée :** $\mathcal{C}$, un ensemble de valeurs pour le paramètre C.
**Entrée :** $p$, la proportion de l'ensemble $S$ dédiée à l'apprentissage du prior.
**Entrée :** $\sigma$, l'écart-type du prior.
**Entrée :** $\delta$, le paramètre de confiance.
**Entrée :** $(\kappa, \omega, \epsilon_{abs})$, les paramètres de la descente de gradient conjugué.

Diviser aléatoirement $S$ en deux sous-ensembles $S_A$ et $S_B$ tels que $|S_A| = \lfloor mp \rfloor$ .
Initialiser $\overline{\mathbf{w}} \leftarrow \mathbf{0}$ et $B(\overline{\mathbf{w}}) \leftarrow 1$ .
**pour tout** $C \in \mathcal{C}$ **faire**
Minimiser l'équation (4.15) : $\mathbf{w}' \leftarrow$DescenteGradientMultiple($\nabla\beta, \kappa, \omega, \epsilon_{abs}$) .
Les composantes du gradient $\nabla\beta$ sont données par :

$$(\mathcal{P})\quad \frac{\partial\beta}{\partial w'_k} = C \sum_{\mathbf{z}_i \in S_A} \Phi'\Big(\mathbf{w}' \cdot \boldsymbol{\psi}_i\Big)\psi_{i,k} + w'_k \qquad \text{pour } k = 1, \ldots, n' ,$$

$$(\mathcal{D})\quad \frac{\partial\beta}{\partial \alpha'_k} = C \sum_{\mathbf{z}_i \in S_A} \Phi'\left(\frac{\sum_{j=1}^m \alpha'_j G_{j,i}}{\sqrt{G_{i,i}}}\right)\frac{G_{k,i}}{\sqrt{G_{i,i}}} + \sum_{i=1}^m \alpha'_k G_{i,k} \quad \begin{array}{l}\text{pour } k \text{ tq } \mathbf{z}_k \in S_A \\ \text{et } \alpha'_k = 0 \text{ si } \mathbf{z}_k \notin S_A .\end{array}$$

Minimiser l'équation (4.16) : $\mathbf{w} \leftarrow$DescenteGradient($\nabla B, \mathbf{w}', \epsilon_{abs}$) .
Les composantes du gradient $\nabla B$ sont données par :

$$(\mathcal{P})\quad \frac{\partial B}{\partial w_k} = \frac{1}{m_B}\frac{B(1-B)}{B-R_S}\left[\begin{array}{l} w_k - w'_k\left[\dfrac{\mathbf{w}\cdot\mathbf{w}'}{\|\mathbf{w}'\|^2}\left(1-\dfrac{1}{\sigma^2}\right)+\dfrac{1}{\sigma^2}\right] \\ + \left(\ln\dfrac{B(1-R_S)}{R_S(1-B)}\right)\displaystyle\sum_{\mathbf{z}_i\in S_B}\Phi'\Big(\mathbf{w}\cdot\boldsymbol{\psi}_i\Big)\psi_{i,k}\end{array}\right],$$

$$(\mathcal{D})\quad \frac{\partial B}{\partial \alpha_k} = \frac{1}{m_B}\frac{B(1-B)}{B-R_S}\left[\begin{array}{l} \displaystyle\sum_{i=1}^m\left[\alpha_i - \alpha'_i\left[\frac{\sum_{j=1}^m\sum_{l=1}^m \alpha_j\alpha'_l G_{j,l}}{\sum_{j=1}^m\sum_{l=1}^m \alpha'_j\alpha'_l G_{j,l}}\left(1-\frac{1}{\sigma^2}\right)+\frac{1}{\sigma^2}\right]\right]G_{i,k} \\ + \left(\ln\dfrac{B(1-R_S)}{R_S(1-B)}\right)\displaystyle\sum_{\mathbf{z}_i\in S_B}\Phi'\left(\frac{\sum_{j=1}^m \alpha_j G_{j,i}}{\sqrt{G_{i,i}}}\right)\frac{G_{k,i}}{\sqrt{G_{i,i}}}\end{array}\right],$$

où $B$ est trouvé par la méthode de Newton (algorithme 2 de la section 4.4.1).
**si** $B(\mathbf{w}) < B(\overline{\mathbf{w}})$ **alors**
$\overline{\mathbf{w}} \leftarrow \mathbf{w}$ .
**fin si**
**fin pour**

**Sortie :** $\overline{\mathbf{w}}$ ou $\overline{\boldsymbol{\alpha}}$, le classificateur final dans sa représentation primale ou duale.
**Sortie :** $B(\overline{\mathbf{w}})$, la borne sur le risque correspondante.

# Chapitre 5

# Expérimentations et analyse des résultats

> The idea of waiting for something makes it more exciting.
>
> Andy Warhol

Ce chapitre présente une synthèse des résultats empiriques obtenus par l'exécution des différentes versions de l'algorithme PBGD sur plusieurs problèmes d'apprentissage. Les résultats sont comparés à ceux obtenus par AdaBoost et les SVM, deux algorithmes d'apprentissage reconnus pour leur excellente performance sur des données réelles. Cette étude permet de cerner les forces et les faiblesses de l'algorithme PBGD. Particulièrement, nous déterminons quelles versions de l'algorithme favorisent la minimisation du vrai risque et de la borne sur le vrai risque.

## 5.1 Méthodologie

Afin d'évaluer la performance des différentes versions de l'algorithme PBGD, nous les avons exécutées sur plusieurs problèmes de classification binaire. La même méthodologie est appliquée pour chacun des problèmes de classification. D'abord, les données étiquetées disponibles sont divisées aléatoirement en un ensemble d'entraînement $S$ et un ensemble test $T$ [1]. Ensuite, tous les algorithmes d'apprentissage sont entraînés sur le même ensemble $S$. Pour chaque classificateur obtenu, nous calculons le risque sur

[1] Précisons que la subdivision des ensembles $S$ et $T$ est effectuée une seule fois. Ainsi, la même subdivision est utilisée pour toutes les expérimentations.

l'ensemble test $T$ et un intervalle de confiance sur le vrai risque à l'aide de la borne sur l'ensemble test présentée à la section 2.4.2. Enfin, ces intervalles de confiance permettent de déterminer si les écarts des risques observés sur un problème d'apprentissage particulier sont statistiquement significatifs. Pour toutes les expérimentations décrites dans ce chapitre, nous utilisons un paramètre de confiance $\delta = \frac{1}{20}$ lors du calcul des bornes sur le risque.

### 5.1.1 Ensemble de données

La majorité des ensembles de données utilisés lors des tests empiriques proviennent du répertoire «UCI», maintenu par le «Center for Machine Learning and Intelligent Systems» de l'université de Californie [2][2] . Il s'agit d'une sélection de problèmes variés qui sont fréquemment utilisés afin d'évaluer la qualité des algorithmes d'apprentissage. Deux ensembles de données proviennent d'une autre source :

- L'ensemble «MNIST» contient des images (28 pixels par 28 pixels) de chiffres manuscrits. L'étiquette de chaque exemple correspond à un chiffre de 0 à 9 représenté par l'image. Cet ensemble a d'abord été utilisé par Yann LeCun [16][3].
- L'ensemble «Ringnorm» est un ensemble de données synthétiques. Les exemples de chacune des deux classes sont générés par une distribution de données spécifique (deux lois normales multivariés de moyennes et de variances distinctes). Cet ensemble a d'abord été utilisé par Leo Breiman [7][4].

Certains ensembles de données regroupent plus de deux classes, tels «MNIST» (10 classes) et «Letter» (26 classes). Nous les avons divisés afin de créer plusieurs problèmes d'apprentissage binaire. Pour ce faire, nous avons conservé pour chaque problème d'apprentissage les exemples appartenant à deux classes du problème d'origine. Ainsi, l'ensemble de données «MNIST:0vs8» contient les images des chiffres 0 et 8. Similairement, nous avons créé les ensembles «MNIST:1vs7», «MNIST:1vs8», «MNIST:2vs3» et «Letter:AB», «Letter:DO», «Letter:OQ».

Chaque ensemble de données a été divisé aléatoirement en un ensemble de données d'entraînement et un ensemble de données test. L'objectif de nos expérimentations n'est pas de résoudre des problèmes d'apprentissage précis mais plutôt de comparer

[2]Le «UCI Machine Learning Repository» est accessible à l'adresse http://archive.ics.uci.edu/ml. Le lecteur est invité à s'y référer pour obtenir davantage d'informations sur la nature des ensembles de données utilisés.

[3]L'ensemble «MNIST» est disponible à l'adresse : http://yann.lecun.com/exdb/mnist/

[4]Le code source utilisé pour générer l'ensemble «Ringnorm» provient du site web http://www.cs.toronto.edu/~delve/data/ringnorm/desc.html

des algorithmes d'apprentissage entre eux. Dans ce contexte, aucun pré-traitement des données ne s'impose. Les seuls ensembles modifiés sont «Credit-A» et «Adult», puisque les attributs présentent de grandes valeurs qui peuvent causer des problèmes de précision lors des calculs impliquant le noyau RBF (à cause de l'expression exponentielle). Pour ce faire, la valeur de chaque attribut $x_1, x_2, \ldots, x_n$ de tous les exemples a été renormalisée en utilisant la technique de la tangente hyperbolique :

$$x_i' = \tanh\left[\frac{x_i - \bar{x}_i}{\sigma_i}\right],$$

où $\bar{x}_i$ et $\sigma_i$ sont respectivement la moyenne et l'écart type des valeurs du $i$ème attribut mesurés sur l'ensemble d'entraînement.

Remarquons que les ensembles «Ringnorm» et «Waveform» sont constitués de données synthétiques, c'est-à-dire qu'ils ont été générés par un programme informatique en accord avec une distribution de probabilité préétablie. Les autres ensembles de données proviennent tous de l'observation de phénomènes réels.

### 5.1.2 Apprentissage d'un vote de majorité de decision stumps

Il serait possible d'expérimenter la version primale de l'algorithme PBGD avec plusieurs types de classificateurs de base. Nous nous limitons ici aux decision stumps (voir la section 2.2.2), qui sont souvent utilisés de pair avec l'algorithme d'apprentissage AdaBoost. Préalablement à l'exécution de l'algorithme d'apprentissage, il est nécessaire de déterminer l'ensemble $\mathcal{H}$ de tous les decision stumps constituant le vote de majorité. Pour un ensemble d'entraînement $S$ dont les exemples possèdent $n$ attributs, nous créons 10 decisions stumps par attribut. Les valeurs de seuil de ces 10 decision stumps sont distribuées également entre la valeur minimale $x_i^{\min}$ et la valeur maximale $x_i^{\max}$ du $i$ème attribut sur l'ensemble d'entraînement[5] :

$$\mathcal{H} = \left\{ h_{i,t,1} \;\middle|\; \begin{array}{l} i = 1, 2, \ldots, n \quad k = 1, 2, \ldots, 10 \\ t = \frac{k}{10} \cdot x_i^{\min} + (1 - \frac{k}{10}) \cdot x_i^{\max} \end{array} \right\}.$$

Rappelons que l'algorithme d'apprentissage peut affecter un poids négatif à un classificateur de base, auquel cas on considère que le poids est affecté au classificateur inverse. Les calculs seront effectués avec un vecteur $\mathbf{w}$ de dimension $10n$, mais représentant implicitement les poids de $20n$ classificateurs.

[5]Notons que la borne PAC-Bayes est valide lorsque l'ensemble $\mathcal{H}$ des classificateurs de base est déterminé indépendamment de l'observation de l'ensemble d'entraînement. Ici, nous supposons que nous possédons une connaissance *a priori* minimale de chaque problème, c'est-à-dire que nous connaissons l'intervalle des valeurs de possibles pour chaque attributs.

| **Ensembles de données** | | | | | **AdaBoost** | | **SVM** (Noyau RBF) | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Diminutif** | **Nom complet dans UCI** | $\|T\|$ | $\|S\|$ | $n$ | $R_T$ | $R_S$ | $R_T$ | $R_S$ | $\gamma$ | $C$ |
| Usvotes | Congressional Voting Records | 200 | 235 | 16 | 0.055 | 0.026 | 0.055 | 0.009 | 0.03125 | 5 |
| Liver | Liver-disorders | 175 | 170 | 6 | 0.320 | 0.224 | 0.314 | 0.141 | 0.00130 | 5 |
| Credit-A | Australian Credit | 300 | 353 | 15 | 0.170 | 0.096 | 0.183 | 0.062 | 0.00833 | 500 |
| Glass | Glass | 107 | 107 | 9 | 0.178 | 0.056 | 0.178 | 0.112 | 0.50000 | 2 |
| Haberman | Haberman's Survival | 150 | 144 | 3 | 0.260 | 0.208 | 0.280 | 0.236 | 0.00340 | 0.02 |
| Heart | Heart-disease | 147 | 150 | 13 | 0.259 | 0.033 | 0.197 | 0.067 | 0.15385 | 1 |
| Sonar | Sonar | 104 | 104 | 60 | 0.231 | 0.000 | 0.164 | 0.038 | 0.40833 | 2 |
| BreastCancer | Breast Cancer | 340 | 343 | 9 | 0.053 | 0.000 | 0.038 | 0.017 | 0.00347 | 0.5 |
| Tic-tac-toe | Tic-Tac-Toe Endgame | 479 | 479 | 9 | 0.357 | 0.286 | 0.081 | 0.000 | 0.22222 | 10 |
| Ionosphere | Ionosphere | 175 | 176 | 34 | 0.120 | 0.000 | 0.097 | 0.011 | 0.13235 | 10 |
| Wdbc | Wisconsin Breast Cancer | 284 | 285 | 30 | 0.049 | 0.000 | 0.074 | 0.039 | 0.00026 | 0.5 |
| MNIST:0vs8 | - | 1916 | 500 | 784 | 0.008 | 0.000 | 0.003 | 0.000 | 0.01594 | 2 |
| MNIST:1vs7 | - | 1922 | 500 | 784 | 0.013 | 0.000 | 0.011 | 0.002 | 0.01020 | 5 |
| MNIST:1vs8 | - | 1936 | 500 | 784 | 0.025 | 0.000 | 0.011 | 0.000 | 0.04082 | 1 |
| MNIST:2vs3 | - | 1905 | 500 | 784 | 0.047 | 0.000 | 0.021 | 0.000 | 0.02296 | 5 |
| Letter:AB | Letter Recognition | 1055 | 500 | 16 | 0.010 | 0.000 | 0.001 | 0.000 | 0.28125 | 0.02 |
| Letter:DO | Letter Recognition | 1058 | 500 | 16 | 0.036 | 0.000 | 0.014 | 0.002 | 0.00781 | 20 |
| Letter:OQ | Letter Recognition | 1036 | 500 | 16 | 0.038 | 0.000 | 0.015 | 0.004 | 0.03125 | 1 |
| Adult | Adult | 10000 | 1809 | 14 | 0.149 | 0.135 | 0.159 | 0.114 | 0.03571 | 100 |
| Mushroom | Mushrooms | 4062 | 4062 | 22 | 0.000 | 0.000 | 0.000 | 0.000 | 0.02273 | 10 |
| Waveform | Waveform | 4000 | 4000 | 21 | 0.085 | 0.078 | 0.068 | 0.063 | 0.02381 | 0.5 |
| Ringnorm | - | 3700 | 3700 | 20 | 0.043 | 0.036 | 0.017 | 0.005 | 0.02500 | 1 |

TAB. 5.1: Ensembles de données utilisés lors des expérimentations. Spécifie le nom complet provenant du répertoire de UCI, le nombre d'exemples présents dans l'ensemble test ($|T|$) et dans l'ensemble d'entraînement ($|S|$), le nombre d'attributs ($n$), le risque sur l'ensemble test ($R_T$) et le risque empirique ($R_S$) des classificateurs obtenus par les algorithmes AdaBoost et SVM, les hyperparamètres des SVM ($\gamma$ et $C$) qui sont choisis par validation croisée.

Nous comparons le risque des classificateurs obtenus par les algorithmes de type PBGD dans l'espace primal avec ceux obtenus par AdaBoost (voir la section 2.3.1). Pour ce faire, l'ensemble des classificateurs utilisé est exactement le même pour tous les algorithmes. Le nombre d'itérations $T$ de AdaBoost est fixé à 200, car ce paramètre a permis d'obtenir des classificateurs à faible risque lors de nos expérimentations.

### 5.1.3 Apprentissage avec un noyau RBF

Les expérimentations avec la version duale de l'algorithme sont effectuées en utilisant un noyau RBF (équation (2.7)). Le choix du paramètre de noyau $\gamma$ se fait parmi l'ensemble de valeurs suggérées par [1] :

$$\gamma \in \left\{ \frac{a^2}{2n} \;\middle|\; a = \tfrac{1}{8}, \tfrac{1}{7}, \tfrac{1}{6}, \tfrac{1}{5}, \tfrac{1}{4}, \tfrac{1}{3}, \tfrac{1}{2}, 1, 2, 3, 4, 5, 6, 7, 8 \right\}. \tag{5.1}$$

où $n$ est le nombre d'attributs des exemples de l'ensemble de données sur lequel on exécute l'algorithme.

Nous comparons le risque des classificateurs obtenus par les algorithmes de type PBGD dans l'espace dual avec ceux obtenus par les SVM à marge floue (voir la section 2.3.2). Le paramètre de marge floue $C$ des SVM est choisi parmi un ensemble de valeurs suggérées par [1] :

$$C \in \{0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 3, 10, 20, 50, 100, 200, 500, 1000\} . \tag{5.2}$$

Pour sélectionner les hyperparamètres $\gamma$ et $C$ des SVM, nous effectuons une validation croisée 10-fois. Les résultats présentés dans ce document sont obtenus en utilisant le programme SVM$^{light}$, dont l'implémentation est décrite dans [12][6].

## 5.2 Expérimentations avec PBGD1

L'algorithme PBGD1, décrit à la section 4.6.1, consiste à utiliser tous les exemples d'entraînement lors d'une unique phase d'apprentissage. Il minimise la borne Langford-Seeger en ayant recours à un prior non informatif $P_{\mathbf{0},1}$. Le classificateur $h_{\mathbf{w}}$ minimisant la borne réalise un compromis entre la divergence de Kullback-Leibler $\mathrm{KL}(Q_{\mathbf{w}}||P_{\mathbf{0},1})$ et le risque empirique $R_S(G_{Q_{\mathbf{w}}})$. L'ampleur de ce compromis est dictée par la fonction $\mathrm{kl}(\cdot\|\cdot)$ (équation (3.3)).

Malgré le fait que la fonction à minimiser n'est pas convexe, les expérimentations ont permis de constater qu'elle possède rarement plusieurs minimums locaux. C'est d'ailleurs ce que suggère l'exemple jouet présenté à la figure 4.2. Ainsi, il est superflu de redémarrer la descente de gradient à plusieurs points différents. On se contente donc d'effectuer une seule étape de descente de gradient en initialisant le point de départ sur le prior. Les paramètres de l'algorithme utilisés lors des expérimentations sont les suivants :

| | |
|---|---|
| Niveau de confiance ($\delta$) | 0.05 |
| Nombre de redémarrages ($\kappa$) | 1 |
| Norme du vecteur initial ($\omega$) | 0 |
| Critère d'arrêt ($\epsilon_{abs}$) | $10^{-4}$ |

Le tableau 5.2 présente les résultats empiriques obtenus par la version primale de l'algorithme. Il contient aussi une borne sur le risque de Gibbs des classificateurs linéaires

[6] Le programme SVM$^{light}$ est téléchargeable à l'adresse http://svmlight.joachims.org/

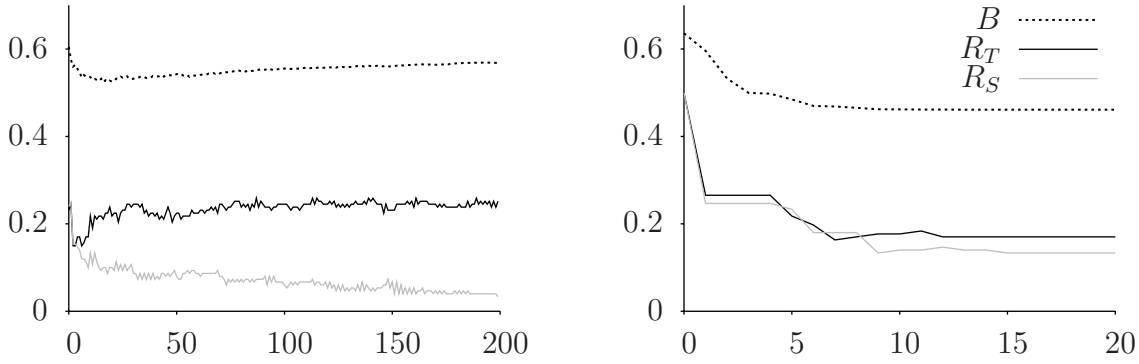
| | **AdaBoost** ($A$) | | | **PBGD1** ($P_1$) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Ensemble** | $R_T$ | $R_S$ | $B$ | $R_T$ | $R_S$ | $B$ | $G_T$ | $G_S$ | $\|\mathbf{w}\|$ | Diff. sign. |
| Usvotes | **0.055** | 0.026 | 0.346 | 0.085 | 0.038 | 0.207 | 0.103 | 0.069 | 4.77 | |
| Liver | **0.320** | 0.224 | 0.614 | 0.383 | 0.429 | 0.587 | 0.426 | 0.432 | 2.24 | |
| Credit-A | **0.170** | 0.096 | 0.504 | 0.177 | 0.153 | 0.375 | 0.243 | 0.214 | 5.48 | |
| Glass | **0.178** | 0.056 | 0.636 | 0.196 | 0.150 | 0.562 | 0.346 | 0.334 | 3.39 | |
| Haberman | **0.260** | 0.208 | 0.590 | 0.273 | 0.236 | 0.422 | 0.283 | 0.251 | 2.64 | |
| Heart | 0.259 | 0.033 | 0.569 | **0.170** | 0.133 | 0.461 | 0.250 | 0.241 | 4.43 | |
| Sonar | **0.231** | 0.000 | 0.644 | 0.269 | 0.154 | 0.579 | 0.376 | 0.342 | 3.54 | |
| BreastCancer | 0.053 | 0.000 | 0.295 | **0.041** | 0.017 | 0.129 | 0.058 | 0.035 | 4.96 | |
| Tic-tac-toe | 0.357 | 0.286 | 0.483 | **0.294** | 0.257 | 0.462 | 0.384 | 0.339 | 4.10 | |
| Ionosphere | **0.120** | 0.000 | 0.602 | **0.120** | 0.142 | 0.425 | 0.223 | 0.214 | 4.81 | |
| Wdbc | 0.049 | 0.000 | 0.447 | **0.042** | 0.046 | 0.272 | 0.099 | 0.101 | 6.27 | |
| MNIST:0vs8 | **0.008** | 0.000 | 0.528 | 0.015 | 0.008 | 0.191 | 0.052 | 0.048 | 8.68 | |
| MNIST:1vs7 | **0.013** | 0.000 | 0.541 | 0.020 | 0.006 | 0.184 | 0.055 | 0.043 | 8.82 | |
| MNIST:1vs8 | **0.025** | 0.000 | 0.552 | 0.037 | 0.028 | 0.247 | 0.097 | 0.081 | 8.96 | |
| MNIST:2vs3 | 0.047 | 0.000 | 0.558 | **0.046** | 0.032 | 0.264 | 0.118 | 0.098 | 8.60 | |
| Letter:AB | 0.010 | 0.000 | 0.254 | **0.009** | 0.002 | 0.180 | 0.050 | 0.045 | 8.38 | |
| Letter:DO | **0.036** | 0.000 | 0.378 | 0.043 | 0.022 | 0.314 | 0.124 | 0.109 | 10.20 | |
| Letter:OQ | **0.038** | 0.000 | 0.431 | 0.061 | 0.064 | 0.357 | 0.170 | 0.159 | 9.17 | |
| Adult | **0.149** | 0.135 | 0.394 | 0.168 | 0.167 | 0.270 | 0.196 | 0.193 | 6.64 | $A < P_1$ |
| Mushroom | **0.000** | 0.000 | 0.200 | 0.046 | 0.043 | 0.130 | 0.065 | 0.062 | 13.52 | $A < P_1$ |
| Waveform | 0.085 | 0.078 | 0.215 | **0.080** | 0.078 | 0.183 | 0.113 | 0.109 | 12.31 | |
| Ringnorm | **0.043** | 0.036 | 0.346 | 0.048 | 0.044 | 0.254 | 0.117 | 0.112 | 21.25 | |

TAB. 5.2: Résultats de PBGD1 dans l'espace primal. Contient le risque de Bayes sur l'ensemble test ($R_T$), le risque empirique de Bayes ($R_S$), le risque de Gibbs du vote de majorité $Q_{\mathbf{w}}$ sur l'ensemble test ($G_T$), le risque empirique de Gibbs du vote de majorité ($G_S$), la valeur de la borne Langford-Seeger ($B$), la norme du vecteur caractérisant le séparateur linéaire ($\|\mathbf{w}\|$) et les différences statistiquement significatives calculées par la borne sur l'ensemble test (Diff. sign.).

construits par AdaBoost. Cette borne est calculée par l'équation (4.13) de la borne Langford-Seeger que minimise PBGD1 (avec $\delta = \frac{1}{20}$). Remarquons que l'expression de la borne accorde une signification particulière à la norme du vecteur $\mathbf{w}$ caractérisant le classificateur, alors que la sortie du classificateur est seulement déterminée par la direction de $\mathbf{w}$. Considérant cela, la borne rapportée dans le tableau est obtenue en sélectionnant, parmi tous les vecteur $\mathbf{w}^*$ de même direction que $\mathbf{w}$, celui qui donne la plus petite valeur de la borne Langford-Seeger[7]. La borne calculée pour AdaBoost par cette méthode est systématiquement plus élevée que celle de PBGD1 (58% plus élevée en moyenne). Le type de classificateur construit par les deux algorithmes diffère donc de manière importante.

Malgré la faible valeur de leur borne, les classificateurs construit par PBGD1 possèdent un risque de Bayes sur l'ensemble test inférieur ou égal à celui des classificateurs obtenus par AdaBoost pour seulement 8 ensembles de données sur 22. De plus, le calcul

[7]Il s'agit de la méthode introduite lors de la sélection de modèle par la borne à la section 3.3.




(a) AdaBoost (b) PBGD1

FIG. 5.1: Comportement de AdaBoost et de PBGD1 sur l'ensemble «Heart» au fil des itérations.

des intervalles de confiance nous informe que le risque de AdaBoost est significativement meilleur que celui de PBGD1 sur deux ensembles de données («Adult» et «Mushroom»). Nous concluons que AdaBoost est un algorithme d'apprentissage préférable à PBGD1.

Il est toutefois intéressant de remarquer que le risque empirique de Bayes (mesuré sur l'ensemble d'entraînement $S$) de PBGD1 est presque toujours inférieur à celui de AdaBoost, peu importe lequel des deux algorithmes possède le meilleur vrai risque. Cela met en évidence que AdaBoost accorde davantage d'importance à la minimisation du risque empirique, si bien que les classificateurs qu'il construit ont souvent un risque empirique nul. Le danger d'une telle approche est de surapprendre les données d'entraînement. Bien que ce ne soit généralement pas le cas, la figure 5.1a illustre un exemple où AdaBoost favorise la minimisation du risque empirique au détriment du vrai risque qui se détériore au fil des itérations. On voit par la figure 5.1b que le processus d'apprentissage de PBGD1 sur le même ensemble est davantage cohérent. En effet, le risque empirique et le vrai risque demeurent semblables au fil des itérations. Il s'agit du comportement typique de PBGD1, qui a été observé sur la majorité des ensembles de données.

Dans l'espace dual, le noyau RBF requiert un hyperparamètre $\gamma$. Lors des expérimentations, nous avons exécuté l'algorithme pour chacune des 15 valeurs de $\gamma$ identifiées par l'ensemble (5.1) et nous avons sélectionné le classificateur possédant la plus faible borne. Afin que la borne demeure valide, un paramètre de confiance de $\frac{\delta}{15}$ est utilisé pour chaque exécution de l'algorithme (en vertu de la borne de l'union). Les expérimentations ont montré que la borne permet de sélectionner efficacement le paramètre

de noyau. Les résultats obtenus ainsi se sont révélés aussi bon qu'en sélectionnant le paramètre $\gamma$ par validation croisée.

| | **SVM** ($S$) | | | **PBGD1** ($P_1$) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Ensemble** | $R_T$ | $R_S$ | $B$ | $R_T$ | $R_S$ | $B$ | $G_T$ | $G_S$ | $\|\mathbf{w}\|$ | $\gamma$ | **Diff. sign.** |
| Usvotes | **0.055** | 0.009 | 0.370 | 0.080 | 0.038 | 0.244 | 0.117 | 0.080 | 5.59 | 0.03125 | |
| Liver | **0.314** | 0.141 | 0.593 | 0.326 | 0.218 | 0.548 | 0.412 | 0.347 | 4.00 | 0.00231 | |
| Credit-A | 0.183 | 0.062 | 0.591 | **0.150** | 0.105 | 0.341 | 0.196 | 0.169 | 6.32 | 0.13333 | |
| Glass | 0.178 | 0.112 | 0.571 | **0.168** | 0.121 | 0.539 | 0.349 | 0.279 | 4.32 | 1.38889 | |
| Haberman | **0.280** | 0.236 | 0.423 | **0.280** | 0.229 | 0.417 | 0.285 | 0.247 | 2.60 | 0.00260 | |
| Heart | 0.197 | 0.067 | 0.513 | **0.190** | 0.160 | 0.441 | 0.236 | 0.233 | 4.08 | 0.03846 | |
| Sonar | **0.163** | 0.038 | 0.599 | 0.250 | 0.125 | 0.560 | 0.379 | 0.308 | 3.96 | 0.53333 | |
| BreastCancer | **0.038** | 0.017 | 0.146 | 0.044 | 0.017 | 0.132 | 0.056 | 0.034 | 5.22 | 0.00617 | |
| Tic-tac-toe | **0.081** | 0.000 | 0.555 | 0.365 | 0.328 | 0.426 | 0.369 | 0.332 | 2.23 | 0.00087 | $S < P_1$ |
| Ionosphere | **0.097** | 0.011 | 0.531 | 0.114 | 0.057 | 0.395 | 0.242 | 0.175 | 5.30 | 0.23529 | |
| Wdbc | **0.074** | 0.039 | 0.400 | **0.074** | 0.042 | 0.366 | 0.204 | 0.158 | 6.99 | 0.00026 | |
| MNIST:0vs8 | **0.003** | 0.000 | 0.257 | 0.009 | 0.002 | 0.202 | 0.053 | 0.045 | 9.55 | 0.01020 | |
| MNIST:1vs7 | **0.011** | 0.002 | 0.216 | 0.014 | 0.004 | 0.161 | 0.045 | 0.031 | 8.67 | 0.01594 | |
| MNIST:1vs8 | **0.011** | 0.000 | 0.306 | 0.014 | 0.008 | 0.204 | 0.066 | 0.050 | 9.26 | 0.01594 | |
| MNIST:2vs3 | **0.020** | 0.000 | 0.348 | 0.038 | 0.018 | 0.265 | 0.112 | 0.087 | 9.36 | 0.01020 | $S < P_1$ |
| Letter:AB | **0.001** | 0.000 | 0.491 | 0.005 | 0.002 | 0.170 | 0.043 | 0.041 | 8.28 | 0.00781 | |
| Letter:DO | **0.014** | 0.002 | 0.395 | 0.017 | 0.008 | 0.267 | 0.095 | 0.073 | 10.50 | 0.03125 | |
| Letter:OQ | **0.015** | 0.004 | 0.332 | 0.029 | 0.018 | 0.299 | 0.130 | 0.105 | 9.81 | 0.03125 | |
| Adult | **0.159** | 0.114 | 0.535 | 0.173 | 0.163 | 0.274 | 0.198 | 0.191 | 7.33 | 0.14286 | |
| Mushroom | **0.000** | 0.000 | 0.213 | 0.007 | 0.005 | 0.119 | 0.032 | 0.029 | 20.59 | 0.02273 | $S < P_1$ |
| Waveform | **0.068** | 0.063 | 0.169 | 0.072 | 0.070 | 0.156 | 0.095 | 0.093 | 11.11 | 0.02381 | |
| Ringnorm | **0.016** | 0.005 | 0.252 | 0.023 | 0.012 | 0.229 | 0.095 | 0.073 | 25.11 | 0.02500 | |

TAB. 5.3: Résultats de PBGD1 dans l'espace dual. Contient les mêmes informations que le tableau 5.2, en plus du paramètre de noyau $\gamma$ sélectionné par la borne.

Le tableau 5.3 présente les résultats empiriques obtenus par la version duale de l'algorithme. Il contient aussi la valeur de la borne sur le risque de Gibbs des classificateurs construits par les SVM, obtenue en sélectionnant, parmi tous les vecteur $\mathbf{w}^*$ de même direction que le vecteur $\mathbf{w}$ de la solution des SVM, celui qui donne la plus petite valeur de la borne Langford-Seeger. Les bornes calculées pour les SVM sont en moyenne 38% plus élevées que celles de PBGD1. Malgré cela, le risque de Bayes sur l'ensemble test des classificateurs construits par PBGD1 est inférieur ou égal à celui des SVM pour seulement 5 ensembles de données sur 22. De plus, le risque des SVM est significativement meilleur que celui de PBGD1 sur trois ensembles de données («Tic-tac-toe», «MNIST:2vs3» et «Mushroom»). Les SVM possèdent donc un net avantage sur PBGD1.

Similairement à ce qui est observé dans l'espace primal, le risque empirique de Bayes de PBGD1 est généralement plus élevé que celui des SVM. Donc, PBGD1 semble un algorithme peu enclin au surapprentissage. En contrepartie, ses performances décevantes s'expliquent possiblement par le peu d'importance accordé à la diminution du risque empirique.

## 5.3 Expérimentations avec PBGDcv

L'algorithme PBGDcv, décrit à la section 4.6.2, détermine le compromis entre la divergence de Kullback-Leibler $\mathrm{KL}(Q_{\mathbf{w}}||P_{\mathbf{0},1})$ et le risque empirique $R_S(G_{Q_{\mathbf{w}}})$ en choisissant le l'hyperparamètre $C$ de la borne hyperparamétrée par validation croisée. Il est particulièrement intéressant de comparer les fonctions de perte minimisées par AdaBoost, les SVM et PBGDcv :

$$\begin{aligned}
\text{AdaBoost :} \quad & \sum_{i=1}^{m} \exp\left(-y_i \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i)\right), \\
\text{SVM :} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \max\left(0, 1 - y_i \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i)\right), \\
\text{PBGDcv :} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \Phi\left(\frac{y_i \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i)}{\|\boldsymbol{\phi}(\mathbf{x}_i)\|}\right).
\end{aligned}$$

Pour les SVM et PBGDcv, la norme du vecteur $\mathbf{w}$ agit comme un régularisateur qui empêche le processus d'apprentissage de minimiser uniquement le risque empirique. L'influence de ce régularisateur est déterminée par le paramètre $C$. La seule différence entre ces deux algorithmes provient de la fonction de perte. Quant à lui, AdaBoost ne possède pas de tel régularisateur. Dans ce cas, le surapprentissage est évité en utilisant des classificateurs faibles dans le vote de majorité et en limitant le nombre d'itérations éffectuées par l'algorithme. Remarquons que le risque exponentiel de AdaBoost et le hinge loss des SVM pénalisent énormément les exemples mal classifiés situés loin de la frontière de séparation, alors que la pénalité du risque sigmoïdal, obtenue par l'expression du risque de Gibbs $\Phi(\cdot)$, sature à la valeur 1 (voir les figures 2.2, 2.4 et 3.6 illustrant ces fonctions de perte).

Considérant la similarité entre les fonctions optimisées par PBGDcv et les SVM, nous utilisons pour ces deux algorithmes la même sélection de valeurs du paramètre $C$ (présentée en (5.2)) même si conceptuellement, il s'agit de deux paramètres différents[8]. Nous ajoutons toutefois à l'ensemble de valeurs de PBGDcv les valeurs 5000 et 10000, car les valeurs du risque sigmoïdal sont généralement plus petites que celles du hinge loss :

$$C \in \{0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 3, 10, 20, 50, 100, 200, 500, 1000, 5000, 10000\}. \tag{5.3}$$

Les expérimentations ont permis de constater que la quantité de minimums locaux de l'expression de la borne hyperparamétrée augmentent avec la valeur de $C$. Afin d'explorer convenablement le domaine des fonctions, il convient de redémarrer un grand

[8]Rappelons que, dans le cas des SVM, $C$ s'avère le paramètre de marge floue. Dans le cas de PBGDcv, $C$ est l'argument de la borne hyperparamétrée.

nombre de fois la descente de gradient. Les paramètres de l'algorithme PBGDcv utilisés lors des expérimentations sont les suivants :

| | |
|---|---|
| Ensemble de valeurs pour $C$ ($\mathcal{C}$) | (voir (5.3)) |
| Divisions lors de la validation croisée ($k$) | 10 |
| Nombre de redémarrages ($\kappa$) | 100 |
| Norme maximale du vecteur initial ($\omega$) | 50 |
| Critère d'arrêt ($\epsilon_{abs}$) | $10^{-4}$ |

Le tableau 5.4 présente les résultats empiriques obtenus par la version primale de l'algorithme. Sur la majorité des ensembles de données (16 sur 22), le risque sur l'ensemble test de PBGDcv est inférieur à celui de AdaBoost. Le risque de PBGDcv est significativement inférieur à celui de AdaBoost sur deux ensembles de données («Tic-tac-toe» et «Ringnorm»). La situation inverse est observée sur un ensemble de données («Mushroom»). Ainsi, les classificateurs obtenus à l'aide de PBGDcv sont donc généralement de qualité supérieure aux classificateurs obtenus par AdaBoost.

| | $A$ | | $P_1$ | | | **PBGDcv** ($P_{cv}$) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Ensemble** | $R_T$ | $B$ | $R_T$ | $B$ | $\|\mathbf{w}\|$ | $R_T$ | $R_S$ | $B$ | $G_T$ | $G_S$ | $\|\mathbf{w}\|$ | $C$ | Diff. sign. |
| Usvotes | **0.055** | 0.346 | 0.085 | 0.207 | 4.77 | 0.060 | 0.021 | 0.261 | 0.057 | 0.021 | 56.38 | 10000 | |
| Liver | 0.320 | 0.614 | 0.383 | 0.587 | 2.24 | **0.314** | 0.259 | 0.593 | 0.394 | 0.375 | 5.35 | 2 | |
| Credit-A | 0.170 | 0.504 | 0.177 | 0.375 | 5.48 | **0.143** | 0.079 | 0.420 | 0.159 | 0.113 | 19.26 | 20 | |
| Glass | 0.178 | 0.636 | 0.196 | 0.562 | 3.39 | **0.150** | 0.121 | 0.581 | 0.226 | 0.188 | 11.08 | 10 | |
| Haberman | **0.260** | 0.590 | 0.273 | 0.422 | 2.64 | 0.273 | 0.236 | 0.424 | 0.386 | 0.368 | 0.9 | 0.05 | |
| Heart | 0.259 | 0.569 | 0.170 | 0.461 | 4.43 | **0.184** | 0.113 | 0.473 | 0.214 | 0.160 | 8.74 | 5 | |
| Sonar | 0.231 | 0.644 | 0.269 | 0.579 | 3.54 | **0.125** | 0.000 | 0.622 | 0.209 | 0.020 | 29.22 | 100 | |
| BreastCancer | 0.053 | 0.295 | 0.041 | 0.129 | 4.96 | **0.044** | 0.006 | 0.190 | 0.048 | 0.006 | 29.23 | 1000 | |
| Tic-tac-toe | 0.357 | 0.483 | 0.294 | 0.462 | 4.10 | **0.207** | 0.186 | 0.474 | 0.217 | 0.194 | 55.52 | 200 | $P_{cv} < A, P_1$ |
| Ionosphere | 0.120 | 0.602 | 0.120 | 0.425 | 4.81 | **0.103** | 0.000 | 0.557 | 0.125 | 0.015 | 31.34 | 100 | |
| Wdbc | 0.049 | 0.447 | 0.042 | 0.272 | 6.27 | **0.035** | 0.007 | 0.319 | 0.051 | 0.029 | 16.56 | 20 | |
| MNIST:0vs8 | 0.008 | 0.528 | 0.015 | 0.191 | 8.68 | **0.006** | 0.000 | 0.262 | 0.011 | 0.000 | 33.94 | 1000 | |
| MNIST:1vs7 | **0.013** | 0.541 | 0.020 | 0.184 | 8.82 | 0.016 | 0.002 | 0.233 | 0.017 | 0.002 | 40.03 | 5000 | |
| MNIST:1vs8 | 0.025 | 0.552 | 0.037 | 0.247 | 8.96 | **0.018** | 0.000 | 0.305 | 0.037 | 0.007 | 26.25 | 50 | $P_{cv} < P_1$ |
| MNIST:2vs3 | 0.047 | 0.558 | 0.046 | 0.264 | 8.60 | **0.034** | 0.002 | 0.356 | 0.048 | 0.005 | 38.74 | 200 | |
| Letter:AB | 0.010 | 0.254 | 0.009 | 0.180 | 8.38 | **0.007** | 0.000 | 0.180 | 0.044 | 0.038 | 9.09 | 2 | |
| Letter:DO | 0.036 | 0.378 | 0.043 | 0.314 | 10.20 | **0.024** | 0.002 | 0.360 | 0.038 | 0.015 | 30.06 | 50 | |
| Letter:OQ | **0.038** | 0.431 | 0.061 | 0.357 | 9.17 | 0.042 | 0.000 | 0.454 | 0.049 | 0.003 | 81.09 | 1000 | |
| Adult | **0.149** | 0.394 | 0.168 | 0.270 | 6.64 | 0.159 | 0.093 | 0.364 | 0.160 | 0.103 | 159.64 | 1000 | |
| Mushroom | **0.000** | 0.200 | 0.046 | 0.130 | 13.52 | 0.002 | 0.002 | 0.150 | 0.004 | 0.003 | 46.5 | 50 | $A < P_{cv} < P_1$ |
| Waveform | 0.085 | 0.215 | 0.080 | 0.183 | 12.31 | **0.078** | 0.074 | 0.184 | 0.097 | 0.092 | 17.22 | 2 | |
| Ringnorm | 0.043 | 0.346 | 0.048 | 0.254 | 21.25 | **0.028** | 0.013 | 0.306 | 0.036 | 0.025 | 90.89 | 100 | $P_{cv} < A, P_1$ |

TAB. 5.4: Résultats de PBGDcv dans l'espace primal. Contient les mêmes informations que le tableau 5.2, en plus du paramètre $C$ sélectionné par validation croisée.

De plus, les performances de PBGDcv sont nettement supérieures à celles de PBGD1. En effet, le risque de PBGDcv est significativement inférieur à celui de PBGD1 sur

quatre ensembles de données («Tic-tac-toe», «MNIST:1vs8», «Mushroom» et «Ringnorm»). Cela indique que le compromis entre la norme du vecteur $\mathbf{w}$ et le risque empirique suggéré par la borne Langford-Seeger n'est pas optimal dans un contexte de minimisation de la borne. On remarque que les valeurs du paramètre $C$ choisies par la validation croisée lors de l'exécution de PBGDcv sont généralement élevées, ce qui permet d'accorder davantage d'importance à la minimisation du risque empirique. D'ailleurs, les normes $\|\mathbf{w}\|$ des classificateurs sont nettement plus grandes pour PBGDcv que pour PBGD1 (5.6 fois plus grandes en moyenne).

Le tableau de résultats présente une borne sur le risque de Gibbs des classificateurs linéaires obtenue par l'algorithme PBGDcv. Comme les valeurs du paramètre $C$ sélectionnées par validation croisée sont grandes, les valeurs de la borne hyperparamétrée associée sont élevées et de peu d'utilité. Nous calculons plutôt la borne Langford-Seeger, avec un prior non informatif, en sélectionnant parmi tous les vecteur $\mathbf{w}^*$ de même direction que le vecteur $\mathbf{w}$, celui qui donne la plus petite valeur de la borne Langford-Seeger. (il s'agit de la même méthode utilisée pour calculer les bornes des classificateurs obtenus par les SVM et AdaBoost). Évidemment, ces bornes sont plus élevées que celles obtenues avec PBGD1 (18% plus élevées en moyenne). Remarquons cependant que les bornes de PBGDcv sont plus petites que celles obtenues par AdaBoost (20% plus faibles en moyenne). Cela suggère que les classificateurs construits par PBGDcv ressemblent davantage à ceux de PBGD1 que ceux de AdaBoost.

Dans l'espace dual, l'algorithme PBGDcv doit sélectionner le paramètre de noyau $\gamma$ en plus du paramètre $C$. Pour ce faire, l'algorithme 7 est modifié afin de sélectionner la valeur de $\gamma$ par validation croisée. Autrement dit, nous minimisons la borne hyperparamétrée avec la combinaison de paramètres $(C, \gamma)$ possédant le plus faible risque de validation croisée.

Le tableau 5.5 présente les résultats obtenus par la version duale de l'algorithme. Le risque de Bayes observé sur l'ensemble test de PBGDcv est inférieur à celui des SVM sur environ la moitié des ensembles de données (11 sur 21). Le calcul des intervalles de confiance sur le vrai risque ne permet pas de départager significativement les deux algorithmes d'apprentissage. Nous concluons que les classificateurs construits par les SVM et PBGDcv sont de qualités équivalentes et que le hinge loss et la perte sigmoïdale sont des fonctions de perte également appropriées.

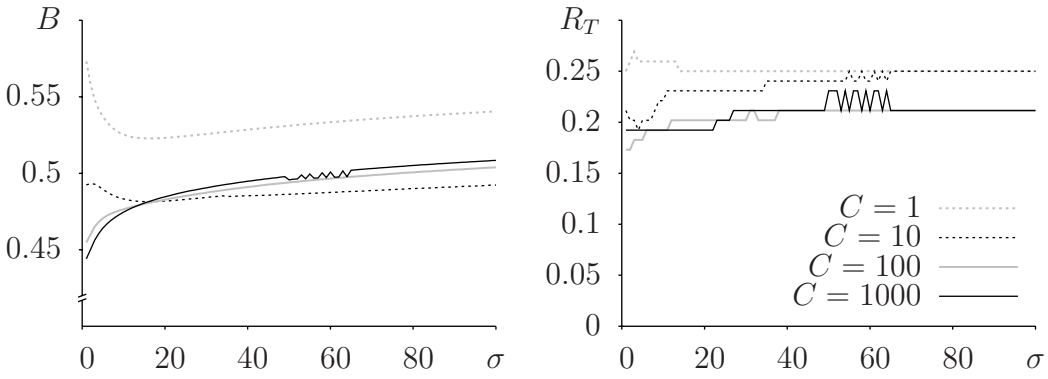
Comme dans l'espace primal, on remarque que PBGDcv est un algorithme préférable à PBGD1, car il permet d'accorder davantage d'importance à la diminution du risque empirique. Cela se traduit par une norme $\|\mathbf{w}\|$ généralement plus élevée pour les classificateurs construits par PBGDcv (6.8 fois plus élevée en moyenne). Aussi, les

| | $S$ | | $P_1$ | | | **PBGDcv ($P_{cv}$)** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Ensemble** | $R_T$ | $B$ | $R_T$ | $B$ | $\|\mathbf{w}\|$ | $R_T$ | $R_S$ | $B$ | $G_T$ | $G_S$ | $\|\mathbf{w}\|$ | $\gamma$ | C | **Diff. sign.** |
| Usvotes | **0.055** | 0.370 | 0.080 | 0.244 | 5.59 | 0.075 | 0.030 | 0.332 | 0.085 | 0.053 | 11.91 | 0.00781 | 10 | |
| Liver | **0.314** | 0.593 | 0.326 | 0.548 | 4.00 | 0.320 | 0.106 | 0.589 | 0.342 | 0.139 | 26.44 | 0.00130 | 50 | |
| Credit-A | 0.183 | 0.591 | 0.150 | 0.341 | 6.32 | **0.160** | 0.091 | 0.375 | 0.267 | 0.224 | 5.39 | 0.30000 | 0.5 | |
| Glass | 0.178 | 0.571 | 0.168 | 0.539 | 4.32 | **0.168** | 0.112 | 0.541 | 0.316 | 0.227 | 5.94 | 1.38889 | 2 | |
| Haberman | 0.280 | 0.423 | 0.280 | 0.417 | 2.60 | **0.253** | 0.153 | 0.555 | 0.250 | 0.160 | 130.92 | 0.00340 | 10000 | |
| Heart | **0.197** | 0.513 | 0.190 | 0.441 | 4.08 | **0.197** | 0.013 | 0.520 | 0.246 | 0.077 | 14.36 | 0.15385 | 10 | |
| Sonar | 0.163 | 0.513 | 0.250 | 0.560 | 3.96 | **0.144** | 0.019 | 0.585 | 0.243 | 0.096 | 13.42 | 0.53333 | 10 | |
| BreastCancer | **0.038** | 0.146 | 0.044 | 0.132 | 5.22 | 0.047 | 0.017 | 0.162 | 0.051 | 0.029 | 7.74 | 0.00222 | 1000 | |
| Tic-tac-toe | 0.081 | 0.555 | 0.365 | 0.426 | 2.23 | **0.077** | 0.000 | 0.548 | 0.107 | 0.003 | 71.14 | 0.22222 | 5 | $P_{cv} < P_1$ |
| Ionosphere | 0.097 | 0.531 | 0.114 | 0.395 | 5.30 | **0.091** | 0.017 | 0.465 | 0.165 | 0.038 | 16.77 | 0.13235 | 500 | |
| Wdbc | **0.074** | 0.400 | 0.074 | 0.366 | 6.99 | **0.074** | 0.042 | 0.367 | 0.210 | 0.166 | 6.61 | 0.00026 | 20 | |
| MNIST:0vs8 | **0.003** | 0.257 | 0.009 | 0.202 | 9.55 | 0.004 | 0.000 | 0.320 | 0.011 | 0.000 | 50.01 | 0.03125 | 1 | |
| MNIST:1vs7 | 0.011 | 0.216 | 0.014 | 0.161 | 8.67 | **0.010** | 0.000 | 0.250 | 0.012 | 0.000 | 47.06 | 0.01594 | 5000 | |
| MNIST:1vs8 | 0.011 | 0.306 | 0.014 | 0.204 | 9.26 | **0.010** | 0.000 | 0.291 | 0.024 | 0.000 | 30.13 | 0.04082 | 5000 | |
| MNIST:2vs3 | **0.020** | 0.348 | 0.038 | 0.265 | 9.36 | 0.023 | 0.000 | 0.326 | 0.036 | 0.000 | 36.03 | 0.02296 | 500 | |
| Letter:AB | **0.001** | 0.491 | 0.005 | 0.170 | 8.28 | **0.001** | 0.000 | 0.485 | 0.408 | 0.339 | 6.01 | 0.28125 | 1000 | |
| Letter:DO | 0.014 | 0.395 | 0.017 | 0.267 | 10.50 | **0.013** | 0.002 | 0.350 | 0.031 | 0.013 | 29.11 | 0.00781 | 0.5 | |
| Letter:OQ | 0.015 | 0.332 | 0.029 | 0.299 | 9.81 | **0.014** | 0.000 | 0.329 | 0.045 | 0.016 | 23.18 | 0.03125 | 50 | |
| Adult | **0.159** | 0.535 | 0.173 | 0.274 | 7.33 | 0.164 | 0.076 | 0.372 | 0.174 | 0.099 | 41.12 | 0.32143 | 20 | |
| Mushroom | **0.000** | 0.213 | 0.007 | 0.119 | 20.59 | **0.000** | 0.000 | 0.167 | 0.001 | 0.000 | 63.02 | 0.02273 | 500 | $P_{cv} < P_1$ |
| Waveform | 0.068 | 0.169 | 0.072 | 0.156 | 11.11 | - | - | - | - | - | - | - | - | |
| Ringnorm | **0.016** | 0.252 | 0.023 | 0.229 | 25.11 | 0.017 | 0.002 | 0.281 | 0.021 | 0.002 | 109.11 | 0.02500 | 1000 | |

TAB. 5.5: Résultats de PBGDcv dans l'espace dual. Contient les mêmes informations que le tableau 5.2, en plus des paramètres $\gamma$ et $C$ sélectionnés par validation croisée. Les résultats sur l'ensemble «Waveform» sont absents en raison du trop grand temps de calcul exigé.

bornes Langford-Seeger sur le risque de Gibbs calculées pour PBGDcv sont plus élevées que celles de PBGD1 (32% plus élevées en moyenne) et elles sont en général légèrement plus basses que celles calculées pour les SVM (5% plus basses en moyenne).

Ainsi, l'algorithme d'apprentissage PBGDcv construit des classificateurs légèrement meilleurs que ceux obtenus par AdaBoost et équivalents à ceux obtenus par les SVM. Son exécution s'avère par contre très lente, à cause du processus de validation croisée et de la procédure de minimisation au cours de laquelle la descente de gradient est redémarrée à plusieurs reprises. Contrairement à PBGDcv, les algorithmes AdaBoost et SVM ne font pas face au problème des multiples minimums locaux car ils optimisent une fonction convexe. De plus, les SVM s'énoncent en un problème quadratique qu'il est possible de résoudre efficacement, ce qui permet d'avoir recours à la validation croisée tout en conservant un temps d'exécution convenable. Ces considérations d'ordre pratique font de PBGDcv un algorithme peu intéressant pour résoudre des problèmes d'apprentissage de grande taille.




(a) Variation de la borne

(b) Variation du risque

FIG. 5.2: Variation de la borne $B$ et du risque sur l'ensemble test $R_T$ du classificateur obtenu par PBGD2 en fonction de l'élongation du prior $\sigma$ sur l'ensemble «Sonar» dans l'espace primal.

## 5.4 Expérimentations avec PBGD2

L'algorithme PBGD2, décrit à la section 4.6.3, mise sur l'apprentissage d'un prior sur une portion des données d'entraînement à l'aide de la borne hyperparamétrée. Le classificateur final est obtenu par l'apprentissage d'un posterior sur l'autre portion des données à l'aide de la borne Langford-Seeger.

L'exécution de cet algorithme nécessite plusieurs hyperparamètres susceptibles de modifier son résultat. Un grand nombre d'expérimentations empiriques a été mené afin de déterminer les valeurs à favoriser pour chacun de ces hyperparamètres. Voici un résumé de nos observations :

1. Il est préférable de dédier la moitié des exemples d'entraînement pour l'apprentissage du prior et l'autre moitié pour l'apprentissage du classificateur final ($p = 0.5$). Il s'agit généralement de la proportion qui permet d'obtenir les classificateurs possédant le meilleur risque de Bayes. De plus, ces classificateurs sont associés à des bornes très petites. Ces observations vont dans le même sens que celles formulées par Ambroladze et al. [1] lors de la sélection de modèle par la borne PAC-Bayes (voir la section 3.3.1).
2. Le recours à un prior allongé a un impact variable selon la valeur du paramètre $C$, tel qu'illustré à la figure 5.2 par un exemple représentatif du comportement observé sur la plupart des ensembles de données. Lorsque la valeur de $C$ est faible ($C \lesssim 10$), une légère élongation du prior ($1 < \sigma \lesssim 20$) permet généralement de diminuer la valeur de la borne associée au classificateur obtenu. Toutefois, lorsque

la valeur de $C$ est élevée ($C \gtrsim 100$), la borne dégrade avec l'élongation du prior. En utilisant un prior isotrope ($\sigma = 1$), on observe qu'il existe presque systématiquement une grande valeur de $C$ permettant d'obtenir une meilleure borne que celle obtenue par une petite valeur de $C$ dont le prior allongé est optimal. De plus, peu importe la valeur de $C$, l'impact de l'élongation du prior sur le vrai risque du classificateur obtenu est imprévisible et négligeable, c'est-à-dire que les variations du risque observées ne sont pas statistiquement significatives. Considérant cela, la stratégie du prior allongé paraît de peu d'utilité. On préfère donc conserver un prior isotrope lors de nos expérimentations.

3. La première phase d'apprentissage de PBGD2 minimise l'expression de la borne hyperparamétrée. Comme pour l'algorithme PBGDcv, on observe la présence de plusieurs minimums locaux lorsque la valeur du paramètre $C$ est élevée. Il est donc nécessaire de redémarrer la descente de gradient un grand nombre de fois ($\kappa = 100$).
4. Il importe d'avoir recours à un ensemble étendu de valeurs pour le paramètre $C$, car la première phase d'apprentissage doit produire des priors de natures différentes. L'utilisation d'un $C$ élevé occasionne parfois un surapprentissage des données mais, advenant cette situation, l'obtention d'une borne élevée lors de la deuxième phase d'apprentissage permet de détecter qu'il s'agit d'un mauvais prior. Conséquemment, nous utilisons l'ensemble de valeurs suivant :

$$\begin{array}{ll} (\mathcal{P}) & C \in \{10^a \,|\, a = 0, 1, \ldots, 6\}\,, \\ (\mathcal{D}) & C \in \{1, 10, 100, 500, 1000\}\,. \end{array} \tag{5.4}$$

   Lors des expérimentations dans l'espace dual $(\mathcal{D})$, les valeurs maximales du paramètre $C$ sont moins élevées que dans l'espace primal $(\mathcal{P})$. Cela s'explique par l'énorme quantité de minimums locaux de la borne hyperparamétrée lors du recours au noyau RBF avec une valeur de $C$ très élevée. Comme il faudrait augmenter le nombre de redémarrages aléatoires pour explorer adéquatement le domaine des fonctions correspondantes et que cela nécessiterait un temps de calcul trop élevé, nous préférons limiter les valeurs de ce paramètre.

Ainsi, les paramètres suivants ont été retenus lors des expérimentations avec l'algorithme PBGD2 :

| | |
|---|---|
| Ensemble de valeurs pour $C$ ($\mathcal{C}$) | (voir (5.4)) |
| Proportion des exemples dédiée à l'apprentissage du prior ($p$) | 0.5 |
| Élongation du prior ($\sigma$) | 1.0 |
| Niveau de confiance ($\delta$) | 0.05 |
| Nombre de redémarrages ($\kappa$) | 100 |
| Norme maximale du vecteur initial ($\omega$) | 50 |
| Critère d'arrêt ($\epsilon_{abs}$) | $10^{-4}$ |

Le tableau 5.6 présente les résultats empiriques obtenus par la version primale de l'algorithme. Le risque de Bayes obtenu sur l'ensemble test est inférieur ou égal à celui de AdaBoost sur environ la moitié des ensembles de données (12 sur 22). Dans un cas, ce risque est significativement meilleur que celui de AdaBoost («Tic-tac-toe») et il est significativement moins bon dans deux cas («Adult» et «Mushroom»). Nous concluons que les deux algorithmes sont pratiquement équivalents.

| | $A$ | $P_1$ | | $P_{cv}$ | **PBGD2** ($P_2$) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Ensemble** | $R_T$ | $R_T$ | $B$ | $R_T$ | $R_T$ | $R_S$ | $B$ | $G_T$ | $G_S$ | $\|\mathbf{w}\|$ | $C$ | Diff. sign. |
| Usvotes | **0.055** | 0.085 | 0.207 | 0.060 | 0.060 | 0.030 | 0.165 | 0.058 | 0.028 | 49.63 | 1e6 | |
| Liver | **0.320** | 0.383 | 0.587 | 0.314 | 0.337 | 0.200 | 0.522 | 0.343 | 0.199 | 111.89 | 1e4 | |
| Credit-A | **0.170** | 0.177 | 0.375 | 0.143 | 0.187 | 0.082 | 0.272 | 0.191 | 0.093 | 64.86 | 1e3 | |
| Glass | 0.178 | 0.196 | 0.562 | 0.150 | **0.168** | 0.075 | 0.395 | 0.176 | 0.085 | 102.98 | 1e6 | |
| Haberman | **0.260** | 0.273 | 0.422 | 0.273 | 0.267 | 0.153 | 0.465 | 0.287 | 0.179 | 18.53 | 1e2 | |
| Heart | 0.259 | 0.170 | 0.461 | 0.184 | **0.190** | 0.087 | 0.379 | 0.205 | 0.087 | 57.22 | 1e4 | |
| Sonar | 0.231 | 0.269 | 0.579 | 0.125 | **0.173** | 0.135 | 0.547 | 0.168 | 0.141 | 65.68 | 1e5 | |
| BreastCancer | 0.053 | 0.041 | 0.129 | 0.044 | **0.047** | 0.012 | 0.104 | 0.054 | 0.017 | 14.68 | 1e2 | |
| Tic-tac-toe | 0.357 | 0.294 | 0.462 | 0.207 | **0.207** | 0.186 | 0.302 | 0.208 | 0.187 | 93.62 | 1e4 | $P_2 < A, P_1$ |
| Ionosphere | 0.120 | 0.120 | 0.425 | 0.103 | **0.109** | 0.068 | 0.347 | 0.129 | 0.077 | 32.09 | 1e3 | |
| Wdbc | **0.049** | 0.042 | 0.272 | 0.035 | **0.049** | 0.018 | 0.147 | 0.048 | 0.021 | 69.59 | 1e6 | |
| MNIST:0vs8 | **0.008** | 0.015 | 0.191 | 0.006 | 0.011 | 0.000 | 0.062 | 0.016 | 0.005 | 26.51 | 1e3 | |
| MNIST:1vs7 | **0.013** | 0.020 | 0.184 | 0.016 | 0.015 | 0.002 | 0.050 | 0.016 | 0.003 | 51.41 | 1e4 | |
| MNIST:1vs8 | **0.025** | 0.037 | 0.247 | 0.018 | 0.027 | 0.012 | 0.087 | 0.030 | 0.014 | 75.03 | 1e6 | |
| MNIST:2vs3 | 0.047 | 0.046 | 0.264 | 0.034 | **0.040** | 0.024 | 0.105 | 0.044 | 0.022 | 74.22 | 1e5 | |
| Letter:AB | 0.010 | 0.009 | 0.180 | 0.007 | **0.007** | 0.002 | 0.065 | 0.011 | 0.005 | 34.64 | 1e4 | |
| Letter:DO | 0.036 | 0.043 | 0.314 | 0.024 | **0.033** | 0.010 | 0.090 | 0.039 | 0.013 | 62.51 | 1e4 | |
| Letter:OQ | **0.038** | 0.061 | 0.357 | 0.042 | 0.053 | 0.018 | 0.106 | 0.053 | 0.019 | 110.42 | 1e6 | |
| Adult | **0.149** | 0.168 | 0.270 | 0.159 | 0.169 | 0.118 | 0.209 | 0.169 | 0.119 | 322.96 | 1e6 | $A < P_2$ |
| Mushroom | **0.000** | 0.046 | 0.130 | 0.002 | 0.016 | 0.017 | 0.030 | 0.017 | 0.017 | 218.64 | 1e4 | $A, P_{cv} < P_2 < P_1$ |
| Waveform | 0.085 | 0.080 | 0.183 | 0.078 | **0.080** | 0.074 | 0.125 | 0.092 | 0.085 | 24.56 | 1e1 | |
| Ringnorm | 0.043 | 0.048 | 0.254 | 0.028 | **0.038** | 0.020 | 0.060 | 0.041 | 0.024 | 154.48 | 1e3 | |

TAB. 5.6: Résultats de PBGD2 dans l'espace primal. Contient les mêmes informations que le tableau 5.2, en plus du paramètre $C$ sélectionné par la borne.

Le risque de Bayes obtenu par PBGD2 sur l'ensemble test est généralement inférieur ou égal à celui de PBGD1 (16 sur 22) et il est statistiquement significativement meilleur que celui de PBGD1 dans deux cas («Tic-tac-toe» et «Mushroom»). Toutefois, le risque de Bayes de PBGD2 est inférieur ou égal à celui de PBGDcv sur seulement 5 ensembles. Le risque de Bayes de PBGDcv est statistiquement significativement inférieur à celui de PBGD2 dans un cas («Mushroom»).

Soulignons l'amélioration notable des garanties théoriques obtenues de pair avec les classificateurs de PBGD2. En effet, la borne sur le risque de Gibbs associée à PBGD2 est en moyenne de 41% inférieure à celle de PBGD1. Le gain est d'autant plus grand sur les ensembles de données possédant beaucoup d'exemples d'entraînement.

Dans l'espace dual, nous exécutons l'algorithme PBGD2 pour chacune des 15 valeurs

du paramètre $\gamma$ du noyau RBF avec un paramètre de confiance de $\frac{\delta}{15}$ (en vertu de la borne de l'union). Le classificateur possédant la borne minimale est ensuite sélectionné.

| | $S$ | $P_1$ | | $P_{cv}$ | **PBGD2** ($P_2$) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Ensemble** | $R_T$ | $R_T$ | $B$ | $R_T$ | $R_T$ | $R_S$ | $B$ | $G_T$ | $G_S$ | $\|\mathbf{w}\|$ | $\gamma$ | C | **Diff. sign.** |
| Usvotes | 0.055 | 0.080 | 0.244 | 0.075 | **0.050** | 0.017 | 0.153 | 0.050 | 0.023 | 41.76 | 0.00781 | 1e3 | |
| Liver | **0.314** | 0.326 | 0.548 | 0.320 | 0.326 | 0.182 | 0.537 | 0.366 | 0.263 | 10.70 | 0.00130 | 1e1 | |
| Credit-A | 0.183 | 0.150 | 0.341 | 0.160 | **0.150** | 0.088 | 0.248 | 0.152 | 0.097 | 52.97 | 0.03333 | 1e3 | |
| Glass | **0.178** | 0.168 | 0.539 | 0.168 | 0.215 | 0.140 | 0.430 | 0.232 | 0.165 | 36.34 | 0.05556 | 5e2 | |
| Haberman | **0.280** | 0.280 | 0.417 | 0.253 | 0.327 | 0.132 | 0.444 | 0.323 | 0.148 | 43.99 | 0.00667 | 1e3 | |
| Heart | 0.197 | 0.190 | 0.441 | 0.197 | **0.184** | 0.093 | 0.400 | 0.190 | 0.112 | 23.03 | 0.03846 | 1e2 | |
| Sonar | **0.163** | 0.250 | 0.560 | 0.144 | 0.173 | 0.077 | 0.477 | 0.231 | 0.094 | 26.98 | 0.53333 | 1e3 | |
| BreastCancer | **0.038** | 0.044 | 0.132 | 0.047 | 0.041 | 0.017 | 0.101 | 0.046 | 0.023 | 15.00 | 0.00113 | 1e2 | |
| Tic-tac-toe | **0.081** | 0.365 | 0.426 | 0.077 | 0.173 | 0.071 | 0.287 | 0.193 | 0.084 | 53.62 | 0.22222 | 5e2 | $S, P_{cv} < P_2 < P_1$ |
| Ionosphere | **0.097** | 0.114 | 0.395 | 0.091 | 0.103 | 0.045 | 0.376 | 0.151 | 0.082 | 24.44 | 0.13235 | 5e2 | |
| Wdbc | 0.074 | 0.074 | 0.366 | 0.074 | **0.067** | 0.042 | 0.298 | 0.119 | 0.070 | 30.53 | 0.00026 | 1e3 | |
| MNIST:0vs8 | **0.003** | 0.009 | 0.202 | 0.004 | 0.007 | 0.000 | 0.058 | 0.015 | 0.004 | 24.59 | 0.01020 | 1e3 | |
| MNIST:1vs7 | 0.011 | 0.014 | 0.161 | 0.010 | **0.009** | 0.002 | 0.052 | 0.015 | 0.003 | 21.78 | 0.01594 | 5e2 | |
| MNIST:1vs8 | **0.011** | 0.014 | 0.204 | 0.010 | **0.011** | 0.000 | 0.060 | 0.019 | 0.004 | 26.69 | 0.01594 | 1e3 | |
| MNIST:2vs3 | **0.020** | 0.038 | 0.265 | 0.023 | 0.028 | 0.002 | 0.096 | 0.043 | 0.014 | 29.87 | 0.01594 | 1e3 | |
| Letter:AB | **0.001** | 0.005 | 0.170 | 0.001 | 0.003 | 0.002 | 0.064 | 0.009 | 0.006 | 23.21 | 0.00781 | 1e3 | |
| Letter:DO | **0.014** | 0.017 | 0.267 | 0.013 | 0.024 | 0.006 | 0.086 | 0.030 | 0.012 | 46.02 | 0.00781 | 1e3 | |
| Letter:OQ | **0.015** | 0.029 | 0.299 | 0.014 | 0.019 | 0.006 | 0.078 | 0.032 | 0.010 | 37.73 | 0.03125 | 1e3 | |
| Adult | **0.159** | 0.173 | 0.274 | 0.164 | 0.180 | 0.104 | 0.224 | 0.181 | 0.109 | 81.26 | 0.32143 | 5e2 | $S, P_{cv} < P_2$ |
| Mushroom | **0.000** | 0.007 | 0.119 | 0.000 | 0.001 | 0.000 | 0.011 | 0.003 | 0.001 | 54.42 | 0.02273 | 5e2 | $P_2 < P_1$ |
| Waveform | **0.068** | 0.072 | 0.156 | - | 0.073 | 0.072 | 0.109 | 0.075 | 0.074 | 138.47 | 0.00049 | 1e3 | |
| Ringnorm | **0.016** | 0.023 | 0.229 | 0.017 | 0.017 | 0.008 | 0.045 | 0.028 | 0.013 | 78.84 | 0.02500 | 1e3 | |

TAB. 5.7: Résultats de PBGD2 dans l'espace dual. Contient les mêmes informations que le tableau 5.2, en plus des paramètres $\gamma$ et $C$ sélectionnés par la borne.

Le tableau 5.7 présente les résultats obtenus par la version duale de l'algorithme. Le risque de Bayes obtenu par PBGD2 est rarement meilleur ou égal à ceux des algorithmes SVM (6 cas sur 22) et PBGDcv (6 cas sur 21). De plus, le risque des SVM et de PBGDcv est significativement meilleur que celui de PBGD2 sur deux ensembles («Tic-tac-toe» et «Adult»). En contrepartie, le risque de Bayes sur l'ensemble test de PBGD2 est généralement meilleur ou égal à celui de PBGD1 (17 cas sur 22). Le risque de PBGD2 est significativement meilleur que celui de PBGD1 sur deux ensembles («Tic-tac-toe» et «Mushroom»).

Comme dans l'espace primal, les bornes sur le risque de Gibbs obtenues par PBGD2 sont particulièrement basses. En effet, elles sont de 40% inférieures en moyenne à celles obtenues à l'aide de PBGD1.

Rétrospectivement, l'algorithme d'apprentissage PBGD2 construit des classificateurs possédant de meilleurs risques de Bayes que PBGD1. La stratégie d'apprentissage d'un prior permet d'accorder davantage d'importance à la diminution du risque empirique, comme en témoignent les normes $\|\mathbf{w}\|$ très élevées des classificateurs générés

par PBGD2. Les bornes obtenues par PBGD2 sont beaucoup plus serrées que celles de PBGD1, ce qui suggère que l'amélioration de la qualité des bornes sur le risque permet le développement de meilleurs algorithmes d'apprentissage. Cela dit, les classificateurs construits par les algorithmes SVM et PBGDcv, effectuant une sélection de modèle par validation croisée, sont plus efficaces que ceux de PBGD2.

Ainsi, les algorithmes expérimentés dans ce mémoire ne permettent pas de substituer la validation croisée par une méthode directe de minimisation d'une borne. Cependant, le gain obtenu de PBGD1 à PBGD2 laisse présager qu'il est possible d'améliorer les résultats en formulant le problème d'apprentissage différemment. Les algorithmes présentés jusqu'ici se sont limités à l'optimisation de deux formulations de la borne PAC-Bayes (la borne Langford-Seeger et la borne hyperparamétrée), alors que le théorème général permet d'en formuler beaucoup d'autres. De même, il serait intéressant d'explorer d'autres types de priors informatifs que le prior isotrope et le prior allongé. La combinaison de ces stratégies est susceptible de mener à plusieurs autres variantes de l'algorithme PBGD.

# Chapitre 6

# Conclusion et travaux futurs

Scapin : Cette galère lui tient à coeur.

Les fourberies de Scapin (Pièce de Molière)

Les travaux présentés dans ce mémoire constituent une première tentative d'élaboration d'un algorithme d'apprentissage basé sur la minimisation d'une borne sur le risque d'un classificateur. Notre étude s'est ici concentrée sur la théorie PAC-Bayes, en raison des bornes relativement basses qu'elle permet d'obtenir. Nous avons d'abord formulé un théorème PAC-Bayes général (théorème 7) qui a permis de démontrer simplement deux versions de la borne PAC-Bayes, soit la borne Langford-Seeger et la borne hyperparamétrée. Nous avons ensuite spécialisé ces bornes aux classificateurs linéaires. Cela permet de les appliquer à la fois aux séparateurs linéaires dans l'espace des votes de majorité (espace primal) et aux séparateurs linéaires dans un espace exprimé implicitement par la stratégie du noyau (espace dual). Nous avons conçu trois algorithmes d'apprentissage optimisant l'expression mathématique des bornes PAC-Bayes par descente de gradient conjugué :

- L'algorithme PBGD1 minimise l'expression de la borne Langford-Seeger en utilisant un prior non informatif ;
- L'algorithme PBGDcv minimise l'expression de la borne hyperparamétrée en sélectionnant le compromis entre la divergence Kullback-Leibler et le risque empirique par validation croisée ;
- L'algorithme PBGD2 minimise l'expression de la borne Langford-Seeger en utilisant un prior informatif. Une portion de l'ensemble de données est préalablement dédiée à l'apprentissage de ce prior par minimisation de la borne hyperparamétrée.

Comparativement aux algorithmes d'apprentissage conventionnels, PBGD1 et PBGD2 présentent l'avantage de fournir, de pair avec le classificateur construit, une borne sur son vrai risque. Les expérimentations empiriques montrent que les classifica-

teurs construits par PBGD2 possèdent de meilleurs risques de Bayes que ceux construits par PBGD1. De même, la stratégie d'apprentissage du prior de PBGD2 permet d'obtenir des valeurs de bornes nettement plus basses.

L'algorithme PBGDcv ne retourne pas de borne sur le risque mais permet d'obtenir de meilleurs classificateurs que PBGD1 et PBGD2. Les expérimentations montrent que ces classificateurs possèdent un risque de Bayes équivalent à ceux obtenus par les SVM et meilleurs que ceux obtenus par AdaBoost. Par contre, l'algorithme est lent d'exécution, en raison du processus de validation croisée et des redémarrages de la descente de gradient rendus nécessaires par les multiples minimums locaux de l'expression de la borne hyperparamétrée. Il n'est donc pas envisageable d'utiliser PBGDcv sur des problèmes réels de grande taille.

La qualité des classificateurs construits par PBGDcv suggère que la recherche d'un compromis entre la fonction de perte sigmoïdale et la divergence de Kullback-Leibler est une stratégie qui permet d'obtenir de bons classificateurs. Ainsi, la forme des bornes PAC-Bayes obtenue par la spécialisation aux classificateurs linéaires semble adéquate. Il convient de continuer les recherches dans cette voie dans le but d'élaborer un algorithme qui permettrait de bien sélectionner le compromis par la borne plutôt que par validation croisée. Une piste de solution serait de raffiner le processus d'apprentissage du prior. Lors des expérimentations, nous avons exploré la possibilité d'utiliser un prior informatif allongé. Cette approche s'est avérée superflue lorsque l'apprentissage du prior est effectué par la minimisation de la borne hyperparamétrée. Il serait envisageable d'adopter une distribution *a priori* répartie entre les multiples minimums locaux de la borne hyperparamétrée. Cela permettrait d'accorder une confiance à plusieurs types de classificateurs à la fois. La difficulté de cette approche réside dans le calcul de la divergence de Kullback-Leibler. En effet, si le calcul de la divergence entre deux distributions gaussiennes s'avère relativement simple, il en est autrement pour des distributions complexes dans des espaces de haute dimensionalité.

Afin de créer un algorithme de qualité, il importe aussi de diminuer le temps d'exécution exigé par la minimisation de la borne. Il convient donc d'appliquer une méthode d'optimisation davantage efficace que la descente de gradient conjugué. Pour les problèmes présentés dans ce mémoire, les techniques d'optimisation envisageables étaient limitées par la non convexité des expressions mathématiques étudiées. Cela s'explique par la forme sigmoïdale du risque de Gibbs découlant de la spécialisation de la borne PAC-Bayes aux classificateurs linéaires. Il serait souhaitable de modifier la formulation de la borne PAC-Bayes afin de l'exprimer sous la forme d'une expression convexe. Des travaux précédents de Germain et al. [11] ont permis d'élaborer une version de la borne PAC-Bayes applicable aux fonctions de pertes générales. Par exemple, on peut utiliser

cette méthode pour obtenir une borne sur le risque exponentiel d'un vote de majorité (plutôt que sur son risque de Gibbs). De même, on peut recourir à une fonction de perte de forme quadratique. Cette méthode permettrait possiblement d'obtenir une expression convexe de la borne PAC-Bayes, dont le minimum se calculerait rapidement. Il n'est toutefois pas garanti que le risque de Bayes des classificateurs obtenus par une telle méthode serait satisfaisant.

Une autre perspective, pour l'instant inexplorée, est l'utilisation de la stratégie d'apprentissage du prior au sein d'un algorithme d'apprentissage de type *enligne*. Dans ce mémoire, nous avons discuté d'algorithmes d'apprentissage de type *batch*, c'est-à-dire d'algorithmes qui ont accès simultanément à tous les exemples d'entraînement lors de l'apprentissage. Le classificateur obtenu ainsi est destiné à être utilisé sans modification subséquente. Quant à eux, les algorithmes d'apprentissage de type enligne reçoivent graduellement les exemples d'entraînement par sous-ensembles. L'apprentissage sur chaque sous-ensemble d'exemples permet de raffiner le classificateur. Ce dernier peut être utilisé entre temps pour classifier des exemples non étiquetés. En plus d'améliorer le classificateur au besoin en lui fournissant davantage d'exemples d'entraînement, cette approche permet d'exécuter l'algorithme sur un ensemble de données volumineux en le divisant en plusieurs sous-ensembles. La plupart des algorithmes d'apprentissage courants sont de type batch, tels les SVM et AdaBoost[1]. La stratégie d'apprentissage du prior se prête facilement à la conception d'algorithmes enlignes. En effet, le vecteur définissant un classificateur peut en tout temps servir de prior lors de l'apprentissage d'un nouveau sous-ensemble de données. Il s'agit d'une approche qui mérite d'être approfondie.

On constate que plusieurs avenues demeurent à explorer afin de perfectionner les algorithmes d'apprentissage présentés dans ce mémoire et d'en développer de nouveaux. Les bons résultats empiriques obtenus par la minimisation de la borne PAC-Bayes suggèrent que la poursuite de ces recherches est susceptible de mener au développement d'algorithmes de qualité. De plus, ces résultats motivent le perfectionnement de la théorie statistique de l'apprentissage, puisque la comparaison entre les algorithmes PBGD1 et PBGD2 suggère que l'obtention de bornes plus serrées peut se traduire en l'apprentissage de classificateurs possédant de meilleurs risques de Bayes. Ultimement, il est à espérer qu'un futur algorithme d'apprentissage fournira, de pair avec le classificateur construit, une borne sur son risque suffisamment serrée pour éliminer la nécessité de le valider sur un ensemble test. La possibilité de dédier la totalité des exemples étiquetés au processus d'apprentissage constituerait un avancement majeur dans la pratique de l'apprentissage automatique.

[1] Récemment, Bradley et Schapire [6] ont proposé une version enligne de l'algorithme AdaBoost.

# Bibliographie


[1] Ambroladze, A., E. Parrado-Hernández et J. Shawe-Taylor, *Tighter pac-bayes bounds*, *Advances in neural information processing systems 19*, Schölkopf, B., J. Platt et T. Hoffman, éditeurs, p. 9–16, MIT Press, Cambridge, MA, 2007.

[2] Asuncion, A. et D. Newman, *UCI machine learning repository*, 2007.

[3] Bishop, C. M., *Neural networks for pattern recognition*, Oxford University Press, Oxford, UK, 1996.

[4] Blumer, A., A. Ehrenfeucht, D. Haussler et M. Warmuth, « Occam's razor », *Information Processing Letters*, 24, p. 377–380, 1987.

[5] Boser, B. E., I. M. Guyon et V. N. Vapnik, « A training algorithm for optimal margin classifiers », *Proceedings of the 5th annual acm workshop on computational learning theory*, p. 144–152, ACM Press, 1992.

[6] Bradley, J. K. et R. Schapire, *Filterboost : Regression and classification on large datasets*, *Advances in neural information processing systems 20*, Platt, J., D. Koller, Y. Singer et S. Roweis, éditeurs, p. 185–192, MIT Press, Cambridge, MA, 2008.

[7] Breiman, L., *Bias, variance, and arcing classifiers*, rapport technique, 1996.

[8] Catoni, O., *Pac-bayesian supervised classification : The thermodynamics of statistical learning*, 2007.

[9] Cortes, C. et V. Vapnik, « Support-vector networks », *Machine Learning*, 20(3), p. 273–297, 1995.

[10] Floyd, S. et M. Warmuth, « Sample compression, learnability, and the Vapnik-Chervonenkis dimension », *Machine Learning*, 21(3), p. 269–304, 1995.

[11] Germain, P., A. Lacasse, F. Laviolette et M. Marchand, *A pac-bayes risk bound for general loss functions*, *Advances in neural information processing systems 19*, Schölkopf, B., J. Platt et T. Hoffman, éditeurs, p. 449–456, MIT Press, Cambridge, MA, 2007.

[12] Joachims, T., *Making large-scale support vector machine learning practical*, MIT Press, Cambridge, MA, 1998.

[13] Lacasse, A., F. Laviolette, M. Marchand, P. Germain et N. Usunier, *Pac-bayes bounds for the risk of the majority vote and the variance of the gibbs classifier*, *Advances in neural information processing systems 19*, Schölkopf, B., J. Platt et T. Hoffman, éditeurs, p. 769–776, MIT Press, Cambridge, MA, 2007.

[14] Langford, J. et J. Shawe-Taylor, *PAC-Bayes & margins*, *Advances in neural information processing systems 15*, S. Becker, S. T. et K. Obermayer, éditeurs, p. 423–430, MIT Press, Cambridge, MA, 2003.

[15] Langford, J., « Tutorial on practical prediction theory for classification », *Journal of Machine Learning Research*, 6, p. 273–306, 2005.

[16] Lecun, Y., Y. Bengio et P. Haffner, « Gradient-based learning applied to document recognition », *Proceedings of the ieee*, p. 2278–2324, 1998.

[17] Marchand, M., *Notes de cours : Apprentissage automatique (ift-65764), chapitre 1*, Université Laval, 2007.

[18] McAllester, D., « Some PAC-Bayesian theorems », *Machine Learning*, 37, p. 355–363, 1999.

[19] McCormick, G. P., *Nonlinear programming : theory, algorithms, and applications*, John Wiley & Sons, Inc., New York, NY, USA, 1983.

[20] Platt, J., *Fast training of support vector machines using sequential minimal optimization*, chapitre 12, p. 185–208, MIT Press, 1999, C. J. C. Burges, and A. Smola editors, Advances in Kernel Methods : Support Vector Learning.

[21] Ross, S. M., *Initiation aux probabilités, traduction de la quatrième édition américaine*, Presses polytechniques et universitaires romandes, 1994.

[22] Schapire, R. E. et Y. Singer, « Improved boosting algorithms using confidence-rated predictions », *Mach. Learn.*, 37(3), p. 297–336, December 1999.

[23] Schapire, R., « The boosting approach to machine learning : An overview », 2001.

[24] Seeger, M., « PAC-Bayesian generalization bounds for gaussian processes », *Journal of Machine Learning Research*, 3, p. 233–269, 2002.