Aix*Marseille Université

Ecole Doctorale Mathématiques et Informatique
(ed 184)

Laboratoire d'Informatique Fondamentale de Marseille
(lif, umr 7279 - cnrs)

---

# Thèse

présentée pour obtenir le grade de
Docteur de l'Université d'Aix*Marseille
*Spécialité : Informatique*

par

Thomas PEEL

# Algorithmes de Poursuite Stochastiques et Inégalités de Concentration Empiriques pour l'Apprentissage Statistique

Thèse soutenue le 29 novembre 2013 devant le jury composé de :

| | | | |
|---|---|---|---|
| M. | Laurent DAUDET | Université Paris Diderot - Paris 7 | *Rapporteur* |
| M. | Nicolas VAYATIS | Ecole Normale Supérieure de Cachan | *Rapporteur* |
| M. | Eric DEBREUVE | CNRS, Université Nice Sophia Antipolis | *Examinateur* |
| M. | François DENIS | Aix*Marseille Université | *Examinateur* |
| M. | Matthieu KOWALSKI | Université Paris-Sud - Paris 11 | *Examinateur* |
| M$^{me}$ | Sandrine ANTHOINE | CNRS, Aix*Marseille Université | *Directrice* |
| M. | Liva RALAIVOLA | Aix*Marseille Université | *Directeur* |

# Table des matières

Table des matières **v**

Notations **1**

Introduction **3**

1 Notions élémentaires **9**
- 1.1 Apprentissage supervisé . . . 11
- 1.2 Inégalités de concentration . . . 13
- 1.3 Algorithme du perceptron . . . 15
  - 1.3.1 Perceptron linéaire . . . 15
  - 1.3.2 Perceptron à noyau . . . 15
- 1.4 Régression linéaire . . . 16
  - 1.4.1 Cadre . . . 16
  - 1.4.2 Régression linéaire régularisée . . . 17
  - 1.4.3 Lasso . . . 18
- 1.5 Vers la représentation parcimonieuse de signaux . . 18

**I Algorithmes gloutons accélérés et application à la classification multiclasse 21**

2 Echantillonage aléatoire pour l'accélération d'algorithmes gloutons **23**
- 2.1 Matching Pursuit et ses variantes . . . 27
  - 2.1.1 Notations . . . 27
  - 2.1.2 Matching Pursuit . . . 27
- 2.2 Matching Pursuit avec sélection aléatoire . . . 29
  - 2.2.1 Algorithme . . . 29
  - 2.2.2 Garanties théoriques . . . 31
- 2.3 Applications numériques . . . 39
  - 2.3.1 Données synthétiques . . . 39
  - 2.3.2 Données réelles : declipping d'un signal audio . . . 44

Conclusion . . . 46

3 Algorithmes gloutons régularisés pour la classification multi-classe avec codes correcteurs : sélection de variables avec parcimonie structurée **49**
- 3.1 Régression linéaire régularisée pour l'apprentissage multi-classes . . . 53
  - 3.1.1 Apprentissage multi-classes et codes correcteurs d'erreurs . 53

3.1.2 Notations et formulation du problème . . . . . 54
3.2 Représentations parcimonieuses simultanées . . . . . 55
3.3 OMP multi-signaux régularisé et parcimonie groupée 56
3.3.1 Présentation du problème et travail relié . . . . . 56
3.3.2 Présentation et analyse de l'algorithme . . . . . 57
3.3.3 Implémentation efficace . . . . . 60
3.3.4 Remarques et extensions . . . . . 61
3.4 Applications . . . . . 62
3.4.1 Problèmes jouets . . . . . 63
3.4.2 20 Newsgroups . . . . . 63
3.5 Conclusion . . . . . 66

**II Inégalités de Bernstein empiriques 67**

4 Inégalités de Bernstein empiriques pour les U-statistiques **69**
4.1 Contexte . . . . . 73
4.1.1 Quelques notations . . . . . 73
4.1.2 U-statistique . . . . . 73
4.1.3 Inégalités de concentration pour les U-statistiques . . . . . 74
4.2 Inégalités de Bernstein empiriques pour les U-statistiques . . . . . 76
4.3 Optimisation des calculs d'une U-statistique . . . . . 78
4.3.1 Ordonnancement bipartite . . . . . 79
4.3.2 Au-delà de l'ordonnancement bipartite . . . . . 82
4.4 Application au ranking bipartite . . . . . 84
4.4.1 Bornes sur la taille de l'échantillon de test . . . . . 84
4.4.2 Ordonnancement en ligne et algorithmes empiriques de course 85
4.5 Conclusion . . . . . 87

5 Inégalités de Bernstein empiriques pour l'apprentissage en ligne **89**
5.1 Préliminaires . . . . . 93
5.1.1 Martingales . . . . . 93
5.1.2 Inégalité d'Azuma-Hoeffding . . . . . 93
5.1.3 Inégalité de Bernstein pour les martingales . . . . . 94
5.2 Inégalités de Bernstein empiriques pour les martingales . . . . . 95
5.3 Inégalités de Bernstein empiriques pour l'apprentissage en ligne . . . . . 98
5.3.1 Estimateur instantané de la variance conditionnelle pour l'apprentissage en ligne . . . . . 98
5.3.2 Bornes théoriques pour l'apprentissage en ligne . . . . . 98
5.3.3 Cas d'une fonction de perte convexe . . . . . 100
5.4 Elaboration d'un algorithme d'apprentissage en ligne . . . . . 101
5.4.1 Pegasos . . . . . 102
5.4.2 Une nouvelle règle de mise à jour . . . . . 103
5.4.3 Utilisation d'un noyau de Mercer . . . . . 105
5.5 Résultats expérimentaux . . . . . 107

5.6 Conclusion . . . . . 110

Conclusion **113**

Bibliographie **117**

# Notations

### Vecteurs et matrices

| | |
|---|---|
| $\boldsymbol{x}$ | Le vecteur $\boldsymbol{x}$ |
| $\mathbf{A}$ | La matrice $\mathbf{A}$ |
| $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ | Le produit scalaire usuel |
| $\|\boldsymbol{x}\|_0$ | La norme $\ell_0$ du vecteur $\boldsymbol{x}$ : $\|\boldsymbol{x}\|_0 := card(\{i \,\|\, x_i \neq 0\})$ |
| $\|\boldsymbol{x}\|_1$ | La norme $\ell_1$ du vecteur $\boldsymbol{x}$ : $\|\boldsymbol{x}\|_1 := \sum_i \|x_i\|$ |
| $\|\boldsymbol{x}\|_2$ | La norme $\ell_2$ du vecteur $\boldsymbol{x}$ : $\|\boldsymbol{x}\|_2 := \sqrt{\sum_i x_i^2}$ |
| $\|\mathbf{A}\|_1$ | L'extension matricielle de la norme $\ell_1$ : $\|\mathbf{A}\|_1 := \sum_{i,j} \|\mathbf{A}_{i,j}\|$ |
| $\|\mathbf{A}\|_F$ | La norme de Frobenius de la matrice $\mathbf{A}$ : $\|\mathbf{A}\|_F := \sqrt{\sum_{i,j} (\mathcal{A}_{i,j})^2}$ |
| $\langle \mathbf{U}, \mathbf{V} \rangle_F$ | Le produit scalaire associé à la norme de Frobenius : $\langle \mathbf{U}, \mathbf{V} \rangle_F := \sum_{i,j} (\mathbf{U})_{i,j} (\mathbf{V})_{i,j}$ |

### Variables aléatoires

| | |
|---|---|
| $\{X_n\}$ | La suite de variables aléatoires $X_1, \ldots, X_n$ |
| $\mathbb{E}[X]$ | L'espérance de la variable aléatoire $X$ |
| $\mathbb{V}[X]$ | La variance de la variable aléatoire $X$ |
| $\mathcal{D}$ | Une distribution de probabilité |

# Introduction

Cette thèse se place à l'interface entre deux domaines : l'apprentissage automatique et le traitement du signal. Si le premier a longtemps été considéré comme une discipline de l'intelligence artificielle, la richesse et la variété des problématiques qu'il traite en font à présent une discipline à part entière. Son essor est fortement lié à la facilité grandissante avec laquelle nous collectons des données. L'apprentissage automatique se concentre sur l'analyse de celles-ci afin de proposer un modèle capable d'en extraire des caractéristiques, voire d'en prédire l'évolution. Le second domaine, le traitement du signal, est lui aussi extrêmement vaste. Qu'il s'agisse d'un son, d'une image, de l'information produite par un capteur, bref, d'un signal, cette discipline englobe toutes les techniques visant à interpréter, à transmettre, à améliorer ou encore à catégoriser celui-ci. Il existe un lien fort entre ces deux domaines, que nous allons essayer de mettre en avant tout au long de ce manuscrit. Au-delà des outils mathématiques sur lesquels ils s'appuient (statistiques, théorie des probabilités, optimisation et bien d'autres), ils sont parfois proches dans la manière qu'ils ont de formuler les problèmes auxquels ils tentent d'apporter des réponses.

Abordons tout d'abord à travers deux exemples quelques notions, liées au sujet de cette thèse, et qui relèvent de ces deux domaines. La première problématique est née dans les années 1980 aux Etats-Unis, au sein des services postaux américains. Devant l'afflux sans cesse grandissant de lettres, l'U.S. Postal a souhaité automatiser la reconnaissance des codes postaux afin d'accélérer le tri du courrier. Pour cela, une base de quelques milliers d'images numériques représentant des chiffres manuscrits a été constituée. Chacun de ces *exemples* (image) a ensuite été *étiqueté* par un expert humain, c'est-à-dire identifié comme étant un 1, un 2, ou tout autre chiffre. L'objectif a ensuite été d'élaborer un prédicteur capable d'associer la bonne classe, c'est-à-dire le bon chiffre, à une nouvelle image de chiffre manuscrit, non-présente dans l'ensemble ayant servi à la conception de ce modèle. En d'autres termes, les services postaux souhaitaient obtenir un *classifieur* capable de *généraliser* ce qu'il a *appris* à de nouvelles instances. Nous profitons de cet exemple pour aborder deux notions auxquelles nous nous intéressons dans cette thèse. La première est celle du *risque d'erreur* d'une telle machine qui mesure sa capacité à prédire correctement la classe d'un nouvel exemple. Une manière naturelle d'estimer ce risque est de tester cette machine sur un nouvel échantillon de chiffres manuscrits afin d'obtenir une information statistique telle qu'un intervalle de confiance sur le nombre d'erreurs commises par le modèle. La seconde notion est celle de l'*apprentissage en ligne* qui englobe les algorithmes capables de mettre à jour le classifieur à chaque nouvel exemple rencontré. Quid du

traitement du signal ? Prenons le modèle des *k plus proches voisins* [Cover et Hart 1967] qui attribue à chaque nouveau chiffre à identifier l'étiquette majoritaire parmi celles des $k$ exemples les plus ressemblants dans la base de données étiquetées. La qualité de cette procédure repose en grande partie sur la notion de ressemblance utilisée par l'algorithme pour comparer deux exemples. Doit-il analyser la moindre caractéristique, le moindre pixel de l'image ? Pas selon le principe du rasoir d'Occam, dont une formulation couramment utilisée est : « les hypothèses les plus simples sont souvent les meilleures ». Selon ce principe, une représentation simple (et non simpliste) des chiffres est préférable à une représentation complexe ; le concept de *représentation parcimonieuse* en traitement de signal en est un exemple de représentation simple. Dans la pratique, choisir une représentation parcimonieuse adéquate des chiffres manuscrits conduit à une comparaison 1) plus rapide du fait du petit nombre de caractéristiques et 2) plus robuste aux petites variations (translation, échelle, etc). Considérons maintenant un moteur de recherche. Ce type de système nous permet d'introduire la dernière notion que nous souhaitons illustrer dans cette introduction, celle d'*ordonnancement* (ranking en anglais). Supposons qu'un utilisateur interroge un moteur de recherche afin d'obtenir les derniers résultats de son équipe favorite. Parmi les milliers de pages web donnant des résultats sportifs, seulement quelques-unes, les pages « pertinentes », fournissent l'information que cherche l'utilisateur. Afin de satisfaire au mieux l'utilisateur (et le fidéliser), le moteur de recherche va s'efforcer de présenter ces pages pertinentes le plus haut possible dans la liste des résultats de recherche. Pour réaliser cela, une solution est de concevoir une fonction capable d'associer un score à chaque page. Le moteur de recherche affichera alors les pages selon l'ordre induit par ce score. L'apprentissage d'une telle *fonction de score* s'inscrit directement comme une problématique d'ordonnancement. Nous détaillons à présent les contributions apportées par cette thèse.

La première problématique considérée est celle de la représentation parcimonieuse de signaux. Nous nous fixons un ensemble de caractéristiques que nous appelons *atomes* avec lesquelles nous souhaitons représenter un signal. Le principe de parcimonie nous amène à chercher une représentation compacte des signaux, c'est-à-dire faisant l'usage d'un petit nombre d'atomes. Tout signal n'admet pas nécessairement une représentation exacte et parcimonieuse. Il faut alors trouver un compromis entre la parcimonie recherchée et l'approximation du signal obtenue, le but étant bien sûr que les caractéristiques du signal nécessaires à son traitement soient conservées. Le problème combinatoire qui se pose afin de réaliser un tel compromis de manière optimale est difficile. Dans cette thèse nous nous concentrons particulièrement sur les algorithmes dits *gloutons* permettant de trouver une solution approchée à ce problème. Ces algorithmes itératifs suivent un principe simple : à chaque itération, identifier une caractéristique *optimale* et mettre à jour la représentation courante du signal. Cette caractéristique est identifiée selon un *critère de sélection* et la mise à jour s'effectue sans remettre en cause les choix faits au cours des itérations passées. A chaque étape, l'approximation du signal est ainsi raffinée. L'algorithme le plus connu dans ce domaine est sans doute *Matching Pursuit (MP)* [Mallat et Zhang 1993], dont de nombreux algorithmes découlent. Tantôt ils apportent une

amélioration à l'étape de mise à jour comme *Orthogonal Matching Pursuit (OMP)* [Pati et al. 1993], tantôt c'est l'étape de sélection qui est modifiée. En effet, la recherche du « meilleur » atome (caractéristique optimale) peut nécessiter un coût de calcul important lorsque l'ensemble des atomes disponibles, appelé *dictionnaire*, est grand. Afin de faire diminuer le coût de cette recherche, on peut utiliser le fait que la sélection d'une « bonne » caractéristique est parfois suffisante à l'obtention d'une approximation de qualité [*Approximate Weak Greedy Algorithms,* Gribonval et Nielsen 2001]. Pour ce faire, Tjoa et Liu [2011] (*Approximate Matching Pursuit*) mettent en œuvre un critère de sélection approximatif rapide à calculer. Popovici et al. [2005] et Moussallam et al. [2012] proposent plutôt de conserver le critère de MP mais d'extraire aléatoirement à chaque itération un petit ensemble de caractéristiques « candidates » sur lesquelles le critère est évalué. Notre première contribution emboîte le pas de ces travaux dont nous partageons la problématique. Nous proposons de considérer un critère de sélection affaibli, appliqué à un sous-ensemble des atomes du dictionnaire renouvelé à chaque itération, afin de réduire davantage le temps de calcul de l'étape de sélection. Bien entendu, réduire ce coût de calcul n'aurait aucun sens si la qualité de l'approximation du signal de départ, comparativement à celle produite par Matching Pursuit, n'était pas maintenue : nous montrons, sur les plans pratique et théorique, que la procédure que nous proposons, *Matching Pursuit avec Sélection Aléatoire (MP-SA)*, implémente de manière effective ce compromis entre vitesse d'exécution et approximation du signal.

La deuxième problématique abordée dans ce document est celle de la représentation parcimonieuse *simultanée* de signaux avec pour objectif la résolution de problèmes multi-classes. Le cadre est identique à celui décrit au début du paragraphe précédent mis à part le fait que nous ajoutons la contrainte de sélectionner un petit ensemble de caractéristiques communes à tous les signaux. Il n'existe pas de manière unique de procéder à cette sélection. Ainsi, dans les travaux sur le sujet, la notion de « meilleure » caractéristique peut désigner un atome *corrélé* à de nombreux signaux [*Simultaneous Orthogonal Matching Pursuit*, Tropp et al. 2006], même si cette corrélation est relativement faible. Elle peut au contraire viser une caractéristique corrélée à un ensemble moins important de signaux, privilégiant l'amplitude moyenne de ces corrélations [Cotter et al. 2005]. Il existe également des travaux montrant que la sélection d'un atome, sous-optimal selon ces stratégies, permet pourtant d'obtenir une approximation de qualité [Leviatan et Temlyakov 2006]. Notre contribution sur le sujet est double. Dans un premier temps, nous présentons un nouvel algorithme pour répondre à cette problématique. Celui-ci, pour lequel nous montrons qu'il peut s'interpréter comme un algorithme de descente de gradient par blocs de coordonnées, permet ainsi de considérer une famille de contraintes structurées au cours de chaque étape de sélection d'un atome. Ensuite, nous proposons de l'utiliser pour résoudre des problèmes de classification *multi-classes* grâce à une formulation adéquate de tels problèmes, basée sur l'utilisation de codes correcteurs d'erreurs [Dietterich et Ghulum 1995] pour représenter chaque classe. Notre algorithme est alors en mesure de produire des classifieurs utilisant un petit nombre de caractéristiques « clés » et

rivalisant avec la plupart des méthodes « état de l'art ».

La troisième problématique présente dans ce manuscrit relève de l'apprentissage statistique. Nous avons mentionné précédemment l'importance de pouvoir quantifier les performances d'un classifieur que nous notons $h$. Dans de nombreux cas, cela passe par une estimation statistique du *risque* (réel) associé à celui-ci. Le risque $R(h) = \mathbb{E}_{X,Y}\left[\ell(h(X), Y)\right]$ du classifieur $h$ est défini comme l'espérance de la variable aléatoire $\ell(h(X), Y)$ où $\ell$ est une *fonction de perte* qui quantifie l'écart entre $Y$ et la prédiction $h(X)$ faite par $h$ pour l'observation $X$. Le risque étant inconnu, on utilise un échantillon $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ pour calculer un risque empirique $\hat{R}(h, \underline{Z}_n) = \frac{1}{n}\sum_{i=1}^n \ell(h(\boldsymbol{x}_i), y_i)$ (où les $(\boldsymbol{x}_i, y_i)$ sont des réalisations i.i.d de $(X, Y)$) que nous souhaitons relier à $R(h)$. Des *inégalités de concentration* classiques qui s'appliquent pour des variables aléatoires indépendantes (Hoeffding, Bernstein) permettent de faire ce lien et nous conduisent, pour un prédicteur $h$ donné, à des *bornes sur le risque* du type

$$\begin{aligned} R(h) &\leq \hat{R}(h, \underline{Z}_n) + f(n) && \text{(Hoeffding)}, \\ R(h) &\leq \hat{R}(h, \underline{Z}_n) + f(\mathbb{V}\left[\ell\right], n) && \text{(Bernstein)}, \end{aligned}$$

où $f$ est une fonction décroissante du nombre de données $n$ [pour une étude détaillée, voir Boucheron et al. 2005] [1]. Dans le cas où la variance $\mathbb{V}\left[\ell\right]$ de $\ell(X, Y)$ est petite, la borne de type Bernstein est plus précise. Cependant, cette quantité est rarement accessible et l'on est souvent obligé de recourir à une majoration pour pouvoir calculer en pratique cette borne. Les inégalités de Bernstein empiriques ont pour but de remplacer cette majoration, souvent lâche, par l'introduction d'un estimateur empirique de la variance [Maurer et Pontil 2009]. Cependant, dans le cas de la problématique du ranking, la fonction de perte considère deux exemples : $\ell(h, (\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j))$. Le risque empirique est alors défini comme

$$\hat{R}(h, \underline{Z}_n) = \frac{1}{n(n-1)} \sum_{i \neq j} \ell(h, (\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)).$$

On constate que les termes de cette somme ne sont pas des variables aléatoires indépendantes (bien que les paires $(\boldsymbol{x}_i, y_i)$ le soient) et l'utilisation des inégalités de concentration évoquées ci-dessus n'est donc plus possible. Pour faire le lien discuté précédemment entre le risque empirique et son espérance, il convient alors d'utiliser la théorie des *U-statistiques* [Hoeffding 1948] comme le proposent Clémençon et al. [2008] ; les inégalités de concentration citées précédemment y disposent de leur équivalent adapté aux dépendances statistiques portées par les U-statistiques [Hoeffding 1963, Arcones 1995]. Dans ce cadre de travail, notre première contribution est l'établissement de nouvelles inégalités de Bernstein *empiriques* pour les U-statistiques, dont nous tirons profit dans le cadre du ranking. Nous proposons également un algorithme pour calculer efficacement la statistique $\hat{R}$ dans ce cas.

1. Notons que nous fournissons dans cet exemple introductif une borne valable pour un prédicteur $h$ choisi *a priori*. Des arguments plus élaborés faisant par exemple intervenir les capacités de classes de fonctions devraient être invoqués pour obtenir des bornes simultanément valables pour tous les classifieurs d'une classe donnée.

La quatrième et dernière problématique que nous traitons relève de l'apprentissage *en ligne*. Cette notion désigne la famille des algorithmes d'apprentissage dont le mode de fonctionnement repose sur la mise à jour d'un classifieur courant à chaque nouvel exemple considéré. L'exécution de ce type d'algorithme produit donc un ensemble de classifieurs (nous utilisons également le terme *hypothèse*) que nous notons $\{h_1, \ldots, h_T\}$. Une question naturelle est alors de pouvoir caractériser le risque moyen associé à ces hypothèses. A nouveau, l'utilisation d'inégalités de concentration est au cœur des résultats que nous présentons. Les dépendances qui existent entre ces hypothèses générées successivement interdisent de considérer celles-ci comme des variables aléatoires indépendantes. Cesa-Bianchi et al. [2004] proposent d'utiliser des inégalités de concentration issues de la théorie des martingales afin de pallier ce problème. L'atout majeur de leur résultat est l'usage d'un estimateur *instantané* du risque, calculable à moindre coût au cours d'un processus d'apprentissage en ligne. Plus tard, ils ont proposé une amélioration de leur travail grâce à l'utilisation d'une inégalité de concentration du second ordre [Cesa-Bianchi et Gentile 2008]. Notre contribution principale est une nouvelle inégalité de concentration empirique du second ordre pour les martingales qui repose sur un estimateur instantané de la variance. Nous l'utilisons dans le cadre de l'apprentissage en ligne et obtenons une borne plus précise que l'état de l'art, à la fois en théorie et en pratique. Elle nous a ensuite conduit à élaborer un algorithme dont les règles de mise à jour prennent en compte cet estimateur conjointement à l'estimateur instantané du risque usuel.

Certains travaux présentés dans cette thèse ont fait l'objet de publications dans des conférences internationales [Peel et al. 2010; 2012]. D'autres travaux annexes ont été publiés dans des conférences internationales [Machart et al. 2011a] et nationales [Machart et al. 2011b] et ne sont pas présents dans ce manuscrit.

Cette thèse comporte 2 parties et 5 chapitres. Le Chapitre 1 présente les concepts clés de l'apprentissage statistique, introduit quelques notations et détaille les problématiques abordées dans cette thèse. La Partie I dont le thème majeur est la représentation parcimonieuse de signaux regroupe 2 chapitres. Le Chapitre 2 est consacré à la présentation de l'algorithme *Matching Pursuit avec Sélection Aléatoire*. Nous y établissons les garanties théoriques associées à celui-ci et exposons des résultats intéressants de sa mise en pratique. Le Chapitre 3 est quant à lui dédié à la description de la procédure *Greedy Block Coordinate Descent for Regularized Least Squares*. Son application à la résolution de problèmes de classification multi-classes est ensuite abordée et la pertinence de notre approche est confortée par les résultats de simulations numériques. La Partie II traite du thème des inégalités de concentrations empiriques et est scindée en deux chapitres. Nous présentons dans le Chapitre 4 de nouvelles inégalités de concentration empiriques du second ordre pour les U-statistiques. Le calcul rapide des estimateurs impliqués dans celles-ci est ensuite abordé avant que nous décrivions de potentielles applications de notre résultat. Le Chapitre 5 vient clore cette partie par la description d'une nouvelle inégalité de concentration empirique du second ordre pour les martingales. Cette dernière est alors utilisée pour

caractériser le risque moyen des hypothèses apprises au cours d'un processus d'apprentissage en ligne et motive un nouvel algorithme considérant à la fois un estimateur instantané du risque et de la variance. Nous comparons enfin à travers une série d'expériences la qualité de notre inégalité et de notre algorithme par rapport à l'état de l'art. La conclusion de cette thèse en synthétise les contributions et propose des perspectives ouvertes par notre travail.

# Notions élémentaires

1

Sommaire


1.1 Apprentissage supervisé . . . . . 11
1.2 Inégalités de concentration . . . . . 13
1.3 Algorithme du perceptron . . . . . 15
1.3.1 Perceptron linéaire . . . . . 15
1.3.2 Perceptron à noyau . . . . . 15
1.4 Régression linéaire . . . . . 16
1.4.1 Cadre . . . . . 16
1.4.2 Régression linéaire régularisée . . . . . 17
1.4.3 Lasso . . . . . 18
1.5 Vers la représentation parcimonieuse de signaux . . 18


De nos jours, les données sont omniprésentes et leur acquisition de plus en plus aisée couplée à des capacités de stockage grandissantes font de celles-ci des ressources prisées par de nombreux acteurs (administration, régies publicitaires, etc.). Cependant, les données brutes en elles-mêmes sont d'une utilité limitée et il faut à ces acteurs des moyens d'extraire de l'information à partir des données qu'ils récoltent. L'apprentissage automatique désigne un ensemble de méthodes algorithmiques et statistiques permettant l'analyse de ces données et qui conduisent à la production de modèles prédictifs. Un exemple du quotidien souvent cité dans les travaux d'apprentissage automatique est celui du filtrage des e-mails indésirables. Considérons un système dont le but est de déterminer à partir d'un ensemble de caractéristiques simples (expéditeur, objet, etc) si un e-mail est désirable ou non. Pour arriver à ses fins, le système dispose d'un ensemble d'e-mails déjà reçus par un utilisateur, pour lesquels celui-ci a fourni une étiquette (indésirable ou non). A partir de cet ensemble de données étiquetées, le système va élaborer une règle de décision capable de classer de manière automatique un nouvel e-mail et dont un objectif est qu'elle se trompe rarement. On dit que le système va généraliser son apprentissage à de nouveaux exemples. Au début de la procédure, l'échantillon d'exemples étiquetés auquel le système a accès est petit et celui-ci peut donc commettre beaucoup d'erreurs de classification. L'utilisateur sera alors amené à corriger celles-ci et le système pourra utiliser ces corrections pour parfaire son apprentissage. Au bout d'un certain nombre d'e-mails

vus, le système sera enfin en mesure de ne commettre que très peu d'erreurs.

Dans ce chapitre, nous allons poser un cadre mathématique autour de la notion d'apprentissage statistique. Nous en profitons pour introduire les concepts-clés de ce domaine et une partie des notations qui seront utilisées dans ce manuscrit. Consacré en majeure partie à l'apprentissage automatique, la fin de ce chapitre introduira quelques notions utiles pour la représentation et l'approximation de signaux.

## 1.1 Apprentissage supervisé

D'un point de vue mathématique, le problème de la classification supervisée peut se formaliser de la façon suivante [Vapnik 1999]. Une donnée étiquetée est définie comme la réalisation d'un couple

$$(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$$

de variables aléatoires de loi de probabilité $\mathcal{D}$, a priori inconnue, sur l'espace produit $\mathcal{X} \times \mathcal{Y}$. Sauf mention contraire, on supposera dans cette thèse que l'espace de description $\mathcal{X}$ est simplement $\mathbb{R}^M$ :

$$\boldsymbol{x} := (x_1, \ldots, x_M) \in \mathcal{X} = \mathbb{R}^M.$$

Suivant l'espace $\mathcal{Y}$ considéré, on distingue plusieurs types de problèmes en apprentissage supervisé :

- Si $\mathcal{Y}$ est un ensemble discret, on parle de *classification* ou *catégorisation*. On cherche à apprendre à associer une classe à une donnée. Par exemple, on cherche à classer des e-mails en spam / non-spam ou alors à reconnaître des chiffres manuscrits. Dans le cas particulier où $card(\mathcal{Y}) = 2$ (par exemple $\mathcal{Y} = \{-1, +1\}$), on parle de classification *binaire*.
- Si $\mathcal{Y}$ est continu, on parle alors de *régression*. On cherche alors à apprendre à associer une valeur à une donnée. Par exemple, on cherche à prédire la quantité de pluie étant donnée une collection de facteurs météorologiques ou alors on cherche à prédire la taille d'un individu à partir de certaines données physiologiques.

Etant donné un ensemble d'apprentissage $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ constitué de couples de variables aléatoires indépendamment et identiquement distribuées suivant $\mathcal{D}$, on cherche une hypothèse $h : \mathcal{X} \to \tilde{\mathcal{Y}}$ (i.e., une fonction) appartenant à une famille $\mathcal{H}$. L'espace de décision $\tilde{\mathcal{Y}}$ associé au classifieur $h$ n'est pas nécessairement $\mathcal{Y}$. En classification binaire ($\mathcal{Y} = \{-1, +1\}$), il est par exemple courant que $\tilde{\mathcal{Y}} = \mathbb{R}$. L'étiquette prédite pour l'exemple $\boldsymbol{x}$ est dans ce cas

$$\tilde{y} = signe(h(\boldsymbol{x})).$$

Cette fonction de prédiction doit être capable de bien classer une donnée qui n'est pas présente dans l'ensemble $\underline{Z}_n$ (sous réserve qu'elle provienne de la même distribution $\mathcal{D}$) : cette capacité est appelée *généralisation*. Pour définir cette capacité, on introduit la notion de *fonction de perte*.

**Définition 1.1** ***(Fonction de perte)*** *Une fonction de perte $\ell$ est telle que*

$$\ell : \tilde{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}^+. \tag{1.1}$$

Pour un couple $(\boldsymbol{x}, y)$, une telle fonction permet de quantifier à quel point la prédiction $h(\boldsymbol{x})$ réalisée par $h$ est proche de l'étiquette $y$. En classification, on peut souhaiter compter le nombre d'erreurs commises par un classifieur et donc vouloir que cette perte renvoie 0 lorsque le classifieur prédit la bonne

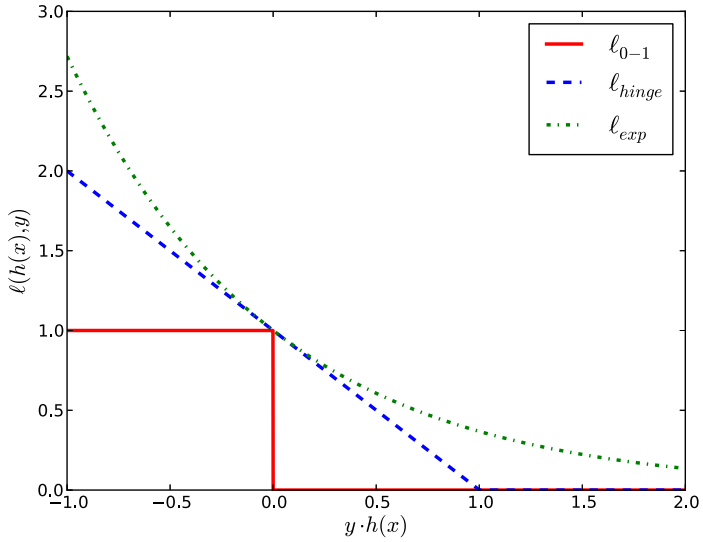



FIGURE 1.1 – *Quelques fonctions de perte classiques en classification binaire.*

étiquette et qu'elle renvoie 1 sinon. Cette fonction de perte particulière porte le nom de perte 0-1 et on la note

$$\ell_{0-1}(h(\boldsymbol{x}), y) := \begin{cases} 0 & \text{si } signe(h(\boldsymbol{x})) = y, \\ 1 & \text{sinon.} \end{cases}$$

Cependant, une telle fonction n'est ni convexe ni différentiable (nous comprendrons par la suite pourquoi ces propriétés sont importantes). C'est pourquoi on préfère utiliser en pratique d'autre pertes, telles que la perte *hinge* et la perte *exponentielle*, illustrées dans la Figure 1.1 et définies par :

$$\ell_{hinge}(h(\boldsymbol{x}), y) := \max(0,\, 1 - y \cdot h(x)),$$
$$\ell_{exp}(h(\boldsymbol{x}), y) := \exp(-y \cdot h(x)).$$

Ces pertes peuvent être envisagées comme des approximations convexes de la perte 0-1.

**Remarque 1.1** *Dans le Chapitre 4, les fonctions de perte que nous considérons prennent la forme $\ell(h, (\boldsymbol{x}, y), (\boldsymbol{x}', y'))$. Cette notation particulière vient du fait que les problématiques auxquelles nous nous attaquons dans ce chapitre visent à apprendre des classifieurs capables d'induire un ordre sur les exemples. On cherche dans ce cas $h$ tel que*

$$h(\boldsymbol{x}) \geq h(\boldsymbol{x}') \text{ si } y \geq y'.$$

*Ces problématiques portent le nom de* ranking *ou* scoring.

Etant donnée une fonction de perte, on peut définir le *risque réel* $R_\ell(h)$ associé à un classifieur comme étant l'espérance de $\ell(h(X), Y)$ pour un couple $(X, Y)$ de loi $\mathcal{D}$.

**Définition 1.2** ***(Risque réel)*** *Soit $\mathcal{D}$ une distribution de probabilité sur $\mathcal{X} \times \mathcal{Y}$, $h : \mathcal{X} \to \tilde{\mathcal{Y}}$ une fonction de classification et $\ell : \tilde{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}^+$ une fonction de perte. Le*

*risque réel $R_\ell(h)$ du classifieur $h$ est défini comme*

$$R_\ell(h) := \mathbb{E}_{(X,Y)\sim\mathcal{D}}\left[\ell(h(X),Y)\right]. \tag{1.2}$$

C'est cette quantité qu'un processus d'apprentissage va tenter de contrôler. Or, la distribution $\mathcal{D}$ étant supposée inconnue il n'est pas possible de calculer $R(h)$ (et donc encore moins de le contrôler). On définit alors le *risque empirique* $\hat{R}_n(h)$ d'une hypothèse $h$, un estimateur du risque $R(h)$ basé sur l'échantillon $\underline{Z}_n$.

**Définition 1.3** ***(Risque empirique)*** *Soit $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ un ensemble d'apprentissage, $h : \mathcal{X} \to \tilde{\mathcal{Y}}$ une fonction de classification et $\ell : \tilde{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}^+$ une fonction de perte. Le risque empirique $\hat{R}_n(h)$ du classifieur $h$ est défini comme*

$$\hat{R}_n(h) := \frac{1}{n}\sum_{i=1}^{n} \ell(h(\boldsymbol{x}_i), y_i). \tag{1.3}$$

Cet estimateur converge en probabilité vers le risque :

$$\lim_{n\to\infty} \mathbb{P}\left[|\hat{R}_n(h) - R(h)| > \epsilon\right] = 0, \forall\epsilon > 0,$$

et amène naturellement une approche visant à minimiser le risque empirique pour construire un classifieur performant. Cependant, ce résultat asymptotique n'apporte qu'une information limitée sur la qualité d'un classifieur appris avec un ensemble de données de taille finie. Les inégalités de concentration sont un outil clé pour quantifier la déviation de l'estimateur.

## 1.2 Inégalités de concentration

Dans cette section, nous présentons deux inégalités de concentration couramment utilisées en apprentissage statistique et dont nous reparlerons souvent par la suite. Avant toute chose, nous présentons un lemme que nous aurons l'occasion d'utiliser à de nombreuses reprises dans ce manuscrit.

**Lemme 1.1** ***(Lemme de renversement d'inégalité)*** *Soit $X$ une variable aléatoire. S'il existe $a, b > 0$ d'une part, $c, d \geq 0$ non nuls ensemble simultanément d'autre part, tels que*

$$\forall\varepsilon > 0,\ \mathbb{P}_X\left[|X| \geq \varepsilon\right] \leq a\exp\left\{-\frac{b\varepsilon^2}{c + d\varepsilon}\right\}, \tag{1.4}$$

*alors, $\forall\delta \in ]0, 1]$,*

$$\mathbb{P}\left[|X| \leq \sqrt{\frac{c}{b}\ln\frac{a}{\delta}} + \frac{d}{b}\ln\frac{a}{\delta}\right] \geq 1 - \delta. \tag{1.5}$$

*Démonstration.* En résolvant pour $\varepsilon$ l'équation du second degré de telle façon que le membre de droite de (1.4) soit égal à $\delta$ on obtient :

$$\varepsilon = \frac{1}{2b}\left(d\ln\frac{a}{\delta} + \sqrt{d^2\ln^2\frac{a}{\delta} + 4bc\ln\frac{a}{\delta}}\right).$$

Ensuite, l'utilisation de l'inégalité $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ lorsque $a, b \geq 0$ donne une borne supérieure sur $\varepsilon$ et fournit le résultat. □

L'inégalité de Hoeffding [Hoeffding 1963] fournit une borne supérieure sur la probabilité qu'une somme de variables aléatoires dévie de son espérance.

**Théorème 1.1** ***(Inégalité de Hoeffding)*** *Soient les variables aléatoires indépendantes $X_1, \ldots, X_n$ telles que $a_i \leq X_i \leq b_i$ pour $1 \leq i \leq n$. Posons $S_n = \sum_{i=1}^n X_i$, alors pour tout $\epsilon > 0$ :*

$$\mathbb{P}\left[S_n - \mathbb{E}\left[S_n\right] \geq \epsilon\right] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right). \tag{1.6}$$

L'inégalité de Bernstein fait intervenir un moment de second ordre dans la caractérisation de la déviation d'une somme de variables aléatoires par rapport à son espérance. Il existe de nombreuses formulation de cette inégalité dont l'une d'entre elles, proposée dans [Bennett 1962], est rappelée dans le théorème suivant.

**Théorème 1.2** ***(Inégalité de Bernstein)*** *Soient les variables aléatoires indépendantes $X_1, \ldots, X_n$ telles que $|X_i - \mathbb{E}\left[X_i\right]| \leq M$ pour $1 \leq i \leq n$. Posons $S_n = \sum_{i=1}^n X_i$ et $\sigma^2 = \sum_{i=1}^n \mathbb{V}\left[X_i\right]$, alors pour tout $\epsilon > 0$ :*

$$\mathbb{P}\left[S_n - \mathbb{E}\left[S_n\right] \geq \epsilon\right] \leq \exp\left(-\frac{\epsilon^2}{2\sigma^2 + \frac{2}{3}M\epsilon}\right). \tag{1.7}$$

En supposant que la fonction de perte est telle que $\ell \in [0, 1]$, les deux inégalités ci-dessus permettent d'écrire, en utilisant le Lemme 1.1, pour une hypothèse $h$ fixée :

$$R(h) \leq \hat{R}_n(h) + \sqrt{\frac{2}{n}\ln\left(\frac{1}{\delta}\right)}, \tag{Hoeffding}$$

$$R(h) \leq \hat{R}_n(h) + \sqrt{\frac{2\mathbb{V}\left[\ell(h(X), Y)\right]}{n}\ln\left(\frac{1}{\delta}\right)} + \frac{2}{3n}\ln\left(\frac{1}{\delta}\right), \tag{Bernstein}$$

avec une probabilité au moins $1 - \delta$ pour tout $0 < \delta \leq 1$.

Ces équations invitent à développer une stratégie visant à minimiser le risque empirique. Cependant, une telle stratégie peut conduire dans la pratique à un algorithme apprenant par cœur les étiquettes des exemples présents dans l'ensemble d'apprentissage et donnant lieu à un classifieur incapable de classer correctement une nouvelle donnée, on parle alors de *sur-apprentissage*. La propension d'un algorithme à apprendre par cœur est typiquement liée à la capacité de la classe de fonction $\mathcal{H}$ considérée. Une analyse des travaux associant inégalités de concentration et capacité des classes de fonction est présentée par Boucheron et al. [2005]. Pour pallier le problème du sur-apprentissage, on peut considérer l'apprentissage d'un classifieur $h$ comme un problème d'optimisation régularisé

$$\min_{h \in \mathcal{H}} \hat{R}_n(h) + \lambda\Omega(h),$$

où $\Omega(h)$ est un terme de *régularisation* ayant pour but de contrôler la complexité de $h$. Nous reviendrons un peu plus tard dans ce chapitre sur la notion de régularisation. Nous présentons pour l'heure un premier algorithme d'apprentissage pour la classification, le Perceptron, qui va nous permettre d'introduire la notion d'apprentissage en ligne.

**Algorithme 1 :** Perceptron

**Entrée :** $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
**Sortie :** $(\mathbf{w}^t, b^t)$
**Initialisation :** $t \leftarrow 0, (\mathbf{w}^0, b^0) \leftarrow (\mathbf{0}, 0)$
  **tant que** $\exists i, y_i(\langle \mathbf{w}^t, \boldsymbol{x}_i \rangle + b^t) \leq 0$ **faire**
    $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + y_i \boldsymbol{x}_i$
    $b^{t+1} \leftarrow b^t + y_i$
    $t \leftarrow t + 1$
  **fin tant que**

## 1.3 Algorithme du perceptron

### 1.3.1 Perceptron linéaire

Le Perceptron [Rosenblatt 1958] est un algorithme itératif de classification adapté à la classification binaire ($\mathcal{Y} = \{-1, 1\}$). Il produit un classifieur linéaire, c'est-à-dire que

$$h(\boldsymbol{x}) = signe(\langle \mathbf{w}, \boldsymbol{x} \rangle + b),$$

où $b$ est un terme de biais et $\mathbf{w}$ est le vecteur normal à un hyperplan *séparateur* (d'équation $\langle \mathbf{w}, \boldsymbol{x} \rangle + b = 0$). Le Perceptron est un algorithme simple qui fonctionne de manière itérative en considérant les exemples de l'ensemble d'apprentissage $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ les uns après les autres. Tant qu'il existe un exemple $(\boldsymbol{x}, y)$ dans l'ensemble d'apprentissage pour lequel $y(\langle \mathbf{w}^t, \boldsymbol{x} \rangle + b) < 0$, l'algorithme procède à une mise à jour de la forme

$$\begin{array}{rcl} \mathbf{w}^{t+1} & \leftarrow & \mathbf{w}^t + y\boldsymbol{x} \\ b^{t+1} & \leftarrow & b^t + y \end{array},$$

où $\mathbf{w}^0, \ldots, \mathbf{w}^t, \ldots$ est la séquence des vecteurs de coefficients produits au cours de l'apprentissage. En traitant un exemple à la fois de manière séquentielle sur $\underline{Z}_n$, le Perceptron peut être vu comme un algorithme d'*apprentissage en ligne*. L'Algorithme 1 résume les différentes étapes de la procédure. Une contrainte pour que l'algorithme du Perceptron converge est que l'échantillon d'apprentissage $\underline{Z}_n$ soit linéairement séparable, c'est-à-dire qu'il existe un hyperplan tel que $y_i(\langle \mathbf{w}, \boldsymbol{x}_i \rangle + b) \geq 0, \forall 1 \leq i \leq n$. Afin de traiter une classe de problèmes plus large que celle des problèmes linéairement séparables, on peut utiliser l'astuce du noyau.

### 1.3.2 Perceptron à noyau

L'idée de l'astuce du noyau [popularisée par Boser et al. 1992, Cortes et Vapnik 1995] est la suivante. Par le biais d'une application $\Phi$, les données sont plongées dans un espace de Hilbert $\mathcal{H}_k$ (de dimension généralement supérieure à M) muni d'un produit scalaire tel que $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle = k(\boldsymbol{x}, \boldsymbol{x}')$ où $k$ est appelé noyau. Le classifieur recherché $h$ est alors de la forme $h(\boldsymbol{x}) = signe(\langle \mathbf{w}, \Phi(\boldsymbol{x}) \rangle)$. Le vecteur $\mathbf{w}$ possède la propriété de pouvoir s'écrire comme une combinaison linéaire des images $\Phi(\boldsymbol{x}_i)$ de nos données dans le nouvel espace considéré [Théorème du représentant, Kimeldorf et

**Algorithme 2 :** Perceptron à noyau

**Entrée :** $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
**Sortie :** $\boldsymbol{\alpha}^t \in \mathbb{R}^n$
**Initialisation :** $t \leftarrow 0, \boldsymbol{\alpha}^0 \leftarrow \mathbf{0}$
**tant que** $\exists i, y_i \sum_{j=1}^n \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq 0$ **faire**
  $\alpha_i^{t+1} \leftarrow \alpha_i^t + y_i$
  $t \leftarrow t + 1$
**fin tant que**

Wahba 1971, Schölkopf et al. 2001]

$$\mathbf{w} = \sum_i \alpha_i \Phi(\boldsymbol{x}_i). \tag{1.8}$$

Le produit scalaire dans $\mathcal{H}_k$ entre $\mathbf{w}$ et $\Phi(\boldsymbol{x})$ s'écrit donc

$$\begin{aligned} \langle \mathbf{w}, \Phi(\boldsymbol{x}) \rangle &= \langle \sum_i \alpha_i \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}) \rangle \\ &= \sum_i \alpha_i \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}) \rangle \\ &= \sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}). \end{aligned} \tag{1.9}$$

L'objectif de l'apprentissage est alors de trouver le bon vecteur de coefficients $\boldsymbol{\alpha}$. Un exemple de noyau rencontré fréquemment en apprentissage automatique est le noyau Gaussien défini par

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\sigma^2}\right). \tag{1.10}$$

Nous ne développons pas plus les méthodes à noyau ici mais le lecteur intéressé pourra se tourner vers les travaux de Schölkopf et Smola [2001], Hofmann et al. [2008] pour une revue détaillée de ces méthodes. En utilisant l'astuce du noyau, il est alors possible de pouvoir classer un jeu de données non linéairement séparable avec un Perceptron. L'Algorithme 2 synthétise les étapes du Perceptron à noyau. Nous continuons notre survol de l'apprentissage automatique par la régression.

## 1.4 Régression linéaire

### 1.4.1 Cadre

Comme évoqué précédemment, la régression met en jeu des couples $(\boldsymbol{x}, y)$ tels que $\boldsymbol{x} \in \mathbb{R}^M$ et $y \in \mathbb{R}$. On parle de régression linéaire lorsque l'on cherche à prédire la valeur $y$ associée à $\boldsymbol{x}$ en utilisant une fonction linéaire

$$h(\boldsymbol{x}) = \sum_{i=1}^M w_i x_i = \langle \mathbf{w}, \boldsymbol{x} \rangle$$

appliquée à la donnée $\boldsymbol{x}$. En régression, il est courant d'utiliser l'*erreur aux moindres carrés* pour évaluer la performance de la fonction $h$. Celle-ci se définit comme

$$\ell_{mse}(h(\boldsymbol{x}), y) = (y - h(\boldsymbol{x}))^2.$$

En considérant l'ensemble d'apprentissage $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, nous cherchons à apprendre une fonction $f$ dont l'erreur empirique

$$\hat{R}_n(h) = \frac{1}{n}\sum_{i=1}^{n}(y_i - h(\boldsymbol{x}_i))^2$$

sur cet ensemble est faible. En notant $\mathbf{X} \in \mathbb{R}^{n\times M}$ la matrice contenant les exemples et $\boldsymbol{y} \in \mathbb{R}^n$ le vecteur contenant les étiquettes sous la forme

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{pmatrix} \text{ et } \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix},$$

le problème de régression peut s'écrire comme le problème d'optimisation suivant :

$$\min_{\mathbf{w}\in\mathbb{R}^M} \|\mathbf{X}\mathbf{w} - \boldsymbol{y}\|_2^2, \tag{1.11}$$

où

$$\|\boldsymbol{u}\|_2 = \sqrt{\sum_i u_i^2}.$$

Ce dernier, bien connu, admet pour solution le vecteur $\mathbf{w}^*$ défini comme

$$\mathbf{w}^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y},$$

lorsque la matrice $\mathbf{X}^\top\mathbf{X}$ est inversible (on peut utiliser la pseudo-inverse de Moore-Penrose $(\mathbf{X}^\top\mathbf{X})^+$ dans le cas contraire). Cependant, pour que la fonction $h$ puisse prédire correctement l'étiquette de nouvelles données (pour se prémunir du sur-apprentissage), on peut introduire un terme de régularisation dans ce problème.

### 1.4.2 Régression linéaire régularisée

Le premier terme de régularisation que nous considérons provient de la pénalisation de la norme $\ell_2$ du vecteur de coefficients $\mathbf{w}$ recherché. Le problème de l'Equation (1.11) devient alors

$$\min_{\mathbf{w}\in\mathbb{R}^M} \|\mathbf{X}\mathbf{w} - \boldsymbol{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2. \tag{1.12}$$

Celui-ci porte le nom de régression *ridge* et admet pour solution le vecteur $\mathbf{w}^*$ tel que

$$\mathbf{w}^* = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbb{I})^{-1}\mathbf{X}^\top\boldsymbol{y},$$

où $\mathbb{I}$ est la matrice identité :

$$\mathbb{I} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}.$$

Au-delà de l'aspect *généralisation*, un apport de ce nouveau terme de pénalisation réside dans l'amélioration du conditionnement de la matrice $(\mathbf{X}^\top\mathbf{X} + \lambda\mathbb{I})$ devant être inversée. D'autres pénalisations sont possibles, le terme présenté dans la section suivante permet par exemple de sélectionner un petit nombre de caractéristiques dans les données.

### 1.4.3 Lasso

Le Lasso [Tibshirani 1996] est une formulation du problème de régression faisant intervenir la norme $\ell_1$ du vecteur de coefficients $\mathbf{w}$ :

$$\min_{\mathbf{w}\in\mathbb{R}^M} \|\mathbf{X}\mathbf{w} - \boldsymbol{y}\|_2^2 + \lambda \|\mathbf{w}\|_1 , \tag{1.13}$$

où

$$\|\boldsymbol{u}\|_1 = \sum_i |u_i|.$$

La particularité de cette formulation par rapport à celle de l'Equation (1.12) réside dans la structure du vecteur $\mathbf{w}^*$ solution de ce problème. En effet, ce dernier possède seulement un petit nombre de coefficients non nuls (plus $\lambda$ est grand, plus le nombre de coefficients nuls augmente). Ainsi, la fonction de régression $h$ ne va s'appuyer que sur quelques coordonnées pour mener à bien sa tâche. Cette solution possède le double avantage (i) de pouvoir être calculée plus rapidement puisque seuls quelques coefficients de $\mathbf{w}$ ne sont pas nuls et (ii) de pouvoir donner lieu à une interprétation aisée lorsque les coefficients de $\mathbf{w}$ correspondent à des caractéristiques des exemples.

## 1.5 Vers la représentation parcimonieuse de signaux

Il est d'usage en traitement du signal de représenter un signal $\boldsymbol{y} \in \mathbb{R}^M$ en utilisant un ensemble de $K$ signaux élémentaires $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^K\}$ appelés *atomes*. Le *dictionnaire*

$$\mathbf{X} = \left(\boldsymbol{x}^1 \cdots \boldsymbol{x}^K\right) \in \mathbb{R}^{M\times K}$$

formé à partir de ces atomes est la matrice dont chaque colonne est un de ces éléments. La décomposition du signal $\boldsymbol{y}$ sur le dictionnaire $\mathbf{X}$ se fait alors *via* un vecteur de coefficients $\mathbf{w}$ de telle sorte que

$$\boldsymbol{y} = \mathbf{X}\mathbf{w} = \sum_{j=1}^{K} w_j \boldsymbol{x}^j.$$

Nous nous intéressons dans cette thèse à représenter les signaux en utilisant un nombre restreint d'atomes, cela revient à chercher un vecteur $\mathbf{w}$ *creux* ou *parcimonieux* (beaucoup de coefficients de $\mathbf{w}$ sont nuls). Parmi les applications naturellement visées par la représentation parcimonieuse de signaux, on pense par exemple à la compression ou la classification de signaux, toutes deux tirant parti de la concision avec laquelle l'information est alors représentée. Nous nous plaçons dans le cas où le dictionnaire $\mathbf{X}$ est *redondant*, c'est-à-dire que le nombre d'atomes $K$ qui le composent est supérieur à la dimension $M$ de ceux-ci. Les propriétés de tels dictionnaires pour la représentation parcimonieuse ont largement été étudiées par Aharon [2006]. La redondance implique qu'il n'existe pas une unique représentation de $\boldsymbol{y}$ dans $\mathbf{X}$. Restreindre le nombre de coefficients non nuls de $\mathbf{w}$ apporte en plus des bénéfices mentionnés précédemment une manière de lever cette indétermination. Ce problème peut se formuler de la façon suivante :

$$\min_{\mathbf{w}\in\mathbb{R}^K} \|\mathbf{X}\mathbf{w} - \boldsymbol{y}\|_2^2 \text{ t.q. } \|\mathbf{w}\|_0 \leq K_0 , \tag{1.14}$$

où

$$\|\boldsymbol{u}\|_0 = card(\{i \,|\, u_i \neq 0\}),$$

et $K_0$ est une contrainte sur la parcimonie de $\mathbf{w}$. Cependant, le problème combinatoire énoncé à l'Equation (1.14) est très difficile à résoudre. Dans le chapitre suivant, nous dressons un rapide état de l'art des techniques utilisées avant de présenter nos contributions sur le sujet. On s'aperçoit d'ores et déjà que cette formulation est très proche du problème de la régression linéaire régularisée. Ces liens seront détaillés dans le Chapitre 3.

# Première partie

# Algorithmes gloutons accélérés et application à la classification multiclasse

# 2 Echantillonage aléatoire pour l'accélération d'algorithmes gloutons

Sommaire


2.1 Matching Pursuit et ses variantes . . . . . 27
2.1.1 Notations . . . . . 27
2.1.2 Matching Pursuit . . . . . 27
2.2 Matching Pursuit avec sélection aléatoire . . . . . 29
2.2.1 Algorithme . . . . . 29
2.2.2 Garanties théoriques . . . . . 31
2.3 Applications numériques . . . . . 39
2.3.1 Données synthétiques . . . . . 39
2.3.2 Données réelles : declipping d'un signal audio . . . . . 44
Conclusion . . . . . 46


Nous nous intéressons ici à la représentation parcimonieuse approchée $\mathbf{w} \in \mathbb{R}^K$ d'un signal $\boldsymbol{y} \in \mathbb{R}^M$ dans un *dictionnaire* $\mathbf{X} \in \mathbb{R}^{M \times K}$ telle que

$$\boldsymbol{y} \approx \mathbf{X}\mathbf{w} \approx \sum_{j=1}^{K} w_j \boldsymbol{x}^j \tag{2.1}$$

où $M \leq K$, et où le nombre d'éléments non nuls $\|\mathbf{w}\|_0$ jouant un rôle dans la représentation $\mathbf{w}$ est petit, c'est-à-dire $\|\mathbf{w}\|_0 \ll K$. Les colonnes du dictionnaire $\mathbf{X}$ sont habituellement appelées *atomes* et $\|\mathbf{w}\|_0$ est aussi appelée parcimonie de $\mathbf{w}$.

Retrouver la décomposition la plus parcimonieuse (c'est-à-dire faisant intervenir le plus petit nombre de coefficients non nuls) d'un signal observé en utilisant un dictionnaire fixé a priori est un problème connu pour être NP-difficile [Davis et al. 1997]. L'algorithme Matching Pursuit (MP) et ses descendants [Mallat et Zhang 1993, Pati et al. 1993, Blumensath et Davies 2008] permettent d'obtenir une approximation de cette représentation. Ces algorithmes fonctionnent de manière itérative en alternant deux étapes, une étape de *sélection* et une étape de *mise à jour*. Ces deux étapes font

intervenir une quantité cruciale appelée *résidu*, que nous noterons $\mathbf{r}$. Celle-ci n'est autre que la part du signal qui n'est pas encore représentée à l'itération courante : $\mathbf{r}^t = \boldsymbol{y} - \mathbf{X}\mathbf{w}^t$. A chaque itération, les algorithmes cités précédemment choisissent une colonne du dictionnaire, optimisant un critère local impliquant le résidu, afin de raffiner la représentation du signal. Ils utilisent ensuite cette colonne pour mettre à jour le vecteur $\mathbf{w}$ sans jamais revenir sur le choix des atomes déjà sélectionnés. Ils appartiennent ainsi à la famille des algorithmes *gloutons* (Cormen et al. [2009], Chapitre 16). Les performances des algorithmes gloutons dans la résolution de problèmes d'approximation parcimonieuse ont largement été étudiées [Tropp 2004, Gribonval et Vandergheynst 2006] et ont contribué à leur renommée.

Ces algorithmes d'approximation demandent malgré tout un coût de calcul important porté en grande partie par la phase de sélection. En effet, le critère local évoqué précédemment implique en général un grand nombre de calculs, souvent proportionnel à la taille du dictionnaire (voir par exemple Gribonval et Nielsen [2001] pour la complexité de calcul induite par certain dictionnaires courants). L'utilisation de dictionnaires de grandes dimensions a donc conduit la communauté à proposer des solutions afin de baisser le coût de calcul de cette étape de sélection. Cela consiste essentiellement à éviter le produit matrice-vecteur $\mathbf{X}^\top\mathbf{r}$, complet et naïf, entre la transposée de la matrice du dictionnaire et le vecteur courant des résidus. Pour des dictionnaires spécifiques, possédant une structure adéquate, l'utilisation de transformées rapides permet de contourner ce calcul naïf. Mallat et Zhang [1993] proposent ainsi dans leur article une implémentation efficace dans le cas de dictionnaires de Gabor. Cependant, de telles transformations n'existent pas dans le cas de dictionnaires génériques ne disposant pas de telles propriétés. Par exemple, l'utilisation de dictionnaires formés de patchs appris à partir d'images [Mairal et al. 2008, Elad et Aharon 2006] peut devenir problématique quand la taille et le nombre de ces derniers est important. Parmi les contributions concernant le large public des dictionnaires sans structure identifiable permettant de calculer rapidement les corrélations, nous trouvons deux catégories. La première consiste à introduire une structure dans ces dictionnaires. On peut par exemple citer les travaux de Jost et al. [2006] (*Tree-Based Pursuit*) qui proposent de construire un dictionnaire hiérarchique formé de représentants des atomes du dictionnaire de base. Ces représentants sont ensuite adossés à une structure d'arbre afin d'accélérer la recherche de corrélations. La deuxième catégorie est portée par les travaux de Temlyakov [2000] puis Gribonval et Nielsen [2001] qui étudient une autre solution dans le cas de tels dictionnaires : utiliser un atome qui ne réalise pas l'optimum du critère mais qui s'en approche. On parle ici de versions *faibles* des algorithmes classiques. On distingue principalement trois approches :

1. Projeter les atomes du dictionnaire et le signal dans un espace de plus petite dimension où les produits scalaires sont plus rapides à calculer. Cette idée est plus connue sous le nom d'acquisition compressée (compressed sensing) et a donné lieu à de nombreux travaux. Elle a notamment été étudiée dans les articles publiés par Donoho [2006] et Candès et al. [2006].
2. Utiliser une fonction de corrélation rapide à calculer en lieu et place du produit scalaire classique. Cette approche a par exemple été abordée

dans les travaux de Cotton et Ellis [2009] et de Tjoa et Liu [2011] en utilisant un schéma de hachage local (LSH).

3. Utiliser une version réduite du dictionnaire obtenue en sélectionnant aléatoirement un petit groupe d'atomes. Cette solution a été adoptée par Popovici et al. [2005] et Moussallam et al. [2012].

D'un autre côté, le coût important induit par le calcul des produits scalaires a guidé la communauté vers une autre piste d'amélioration : une fois l'atome sélectionné, extraire le maximum d'information contenue dans celui-ci. Cette idée vise principalement à minimiser l'erreur d'approximation de ces algorithmes pour un nombre d'itérations fixé, au détriment de la complexité de calcul de ces itérations. Par exemple, les performances sont meilleures — en terme de décroissance par itération de la norme du *résidu* — avec Orthogonal Matching Pursuit [Mallat et Zhang 1993, Pati et al. 1993] qu'avec Matching Pursuit, cela au prix d'une étape de mise à jour coûteuse en temps de calcul. S'est alors posée la question de l'accélération de l'étape de mise à jour, étudiée en profondeur dans Blumensath et Davies [2008]. Les auteurs proposent une analyse détaillée de la complexité calculatoire de MP et OMP ainsi qu'une nouvelle famille d'algorithmes : les algorithmes de poursuite directionnelle utilisant le gradient dans la phase de mise à jour. Leurs algorithmes Gradient Pursuits (GP) obtiennent de meilleures performances que MP avec un coût calculatoire de l'étape de mise à jour moindre que dans OMP. Enfin, la dernière amélioration dont nous parlerons a été proposée dans les travaux de Blumensath et Davies [2009], de Needell et Tropp [2010] et de Donoho et al. [2012]. L'idée portée par ceux-ci est toujours ici de rentabiliser au maximum le calcul des corrélations. Les auteurs proposent alors de ne pas sélectionner seulement l'atome le plus corrélé au résidu à chaque itération mais un ensemble d'atomes corrélés au-delà d'un certain seuil.

Dans ce chapitre, nous proposons une nouvelle approche afin d'accélérer la phase de sélection en généralisant le principe de sélection aléatoire proposé récemment par Moussallam et al. [2012]. Les principales contributions présentées dans ce chapitre sont :

1. une nouvelle famille d'algorithmes reposant sur une phase de sélection aléatoire et qui sont des variantes basées sur les versions classiques (MP, OMP) bien ancrées dans la communauté,
2. des résultats théoriques sur la qualité des atomes sélectionnés au cours des itérations,
3. des simulations numériques mettant en avant la précision, comparable à celle des versions *classiques*, des solutions obtenues via ces nouveaux algorithmes pour un coût de calcul moindre.

Le travail présenté ici a fait l'objet d'une publication [Peel et al. 2012] lors de la conférence EUSIPCO 2012.

Le chapitre est organisé comme suit. La section 2.1 introduit quelques notations spécifiques à ce chapitre, dresse un rapide état de l'art des algorithmes gloutons existants et motive notre travail. Dans la section 2.2, nous présentons et analysons la procédure de sélection aléatoire que nous mettons en œuvre. En particulier, nous montrons les liens existants entre

notre travail et les algorithmes gloutons *faibles*. La section 2.3 contient des simulations numériques montrant la pertinence de notre approche. Enfin, les questions posées par nos travaux sont discutées dans la conclusion.

## 2.1 Matching Pursuit et ses variantes

### 2.1.1 Notations

Soit $\boldsymbol{y}$ un signal dans $\mathbb{R}^M$ ; $\boldsymbol{y}$ est une observation (possiblement) bruitée d'un signal $\mathbf{Xw}$ encodé de manière parcimonieuse avec le dictionnaire $\mathbf{X}$. Nous considérons essentiellement dans ce chapitre le modèle non bruité

$$\boldsymbol{y} = \mathbf{Xw},$$

dans lequel :

- $\mathbf{X} \in \mathbb{R}^{M \times K}$ est un dictionnaire composé de $K$ *atomes* : $\mathbf{X} = [\boldsymbol{x}^1 \dots \boldsymbol{x}^K]$. Chaque $\boldsymbol{x}^i$ est un vecteur de $\mathbb{R}^M$ dont la norme $l_2$ est unitaire ($\|\boldsymbol{x}^i\|_2 = 1$).
- $\mathbf{w} \in \mathbb{R}^K$ est une représentation de $\boldsymbol{y}$ dans $\mathbf{X}$.

De plus, nous supposons que $\boldsymbol{y}$ possède un *support* de taille $K_0$ c'est-à-dire qu'il existe une représentation de $\boldsymbol{y}$ n'impliquant que $K_0$ atomes :

$$\exists \mathbf{w}^* \in \mathbb{R}^K \text{ tel que } \|\mathbf{w}^*\|_0 = K_0 \text{ et } \boldsymbol{y} = \mathbf{Xw}^*.$$

Notre travail se base sur la sélection aléatoire de $m$ lignes et $k$ colonnes du dictionnaire. Par conséquent, nous notons :

- $\mathcal{M} = \{i_1, \dots, i_m\}$ l'ensemble des lignes (coordonnées) sélectionnées,
- $\mathcal{K} = \{j_1, \dots, j_k\}$ l'ensemble des colonnes (atomes) sélectionnées,
- $\boldsymbol{y}_\mathcal{M} \in \mathbb{R}^\mathcal{M}$ la restriction du vecteur $\boldsymbol{y}$ aux coordonnées indicées par l'ensemble $\mathcal{M}$,
- $\langle \boldsymbol{y}, \boldsymbol{y}' \rangle_\mathcal{M} = \langle \boldsymbol{y}_\mathcal{M}, \boldsymbol{y}'_\mathcal{M} \rangle$ le produit scalaire restreint à ces coordonnées.

### 2.1.2 Matching Pursuit

Matching Pursuit [Mallat et Zhang 1993] est un algorithme glouton développé dans les années 90 dont le but est de fournir une solution approchée au problème NP-difficile de trouver la décomposition la plus parcimonieuse du signal $\boldsymbol{y}$ sur le dictionnaire $\mathbf{X}$. Comme décrit dans l'Algorithme 3, Matching Pursuit produit une approximation de la solution au problème

$$\arg\min_{\mathbf{w} \in \mathbb{R}^K} \|\mathbf{w}\|_0 \text{ t.q. } \|\boldsymbol{y} - \mathbf{Xw}\|_2 \leq \epsilon, \tag{2.2}$$

où $\epsilon \geq 0$ ; $\epsilon > 0$ correspond au cas bruité et $\epsilon = 0$ correspond au cas *exactement parcimonieux* qui nous intéresse.

Matching Pursuit est une procédure itérative qui améliore l'estimation du vecteur parcimonieux des coefficients $\mathbf{w}$ de manière gloutonne. Ainsi, à l'itération $t$, le vecteur des coefficients courant $\mathbf{w}^t$ conduit à l'approximation $\mathbf{Xw}^t$ du signal $\boldsymbol{y}$. Nous appelons $\mathbf{r}^t = \boldsymbol{y} - \mathbf{Xw}^t$ le *résidu* de cette décomposition (la part de $\boldsymbol{y}$ pas encore codée par $\mathbf{w}^t$). A cet instant, Matching Pursuit sélectionne le meilleur atome à choisir, c'est-à-dire l'atome $\boldsymbol{x}^{*t+1}$ qui minimise la norme du résidu à l'étape suivante :

$$\boldsymbol{x}^{*t+1} = \arg\min_{\boldsymbol{x} \in \{\boldsymbol{x}^1, \dots, \boldsymbol{x}^K\}} \left\| \mathbf{r}^t - \langle \mathbf{r}^t, \boldsymbol{x} \rangle \boldsymbol{x} \right\|_2. \tag{2.3}$$

Dans la suite de ce chapitre, l'itération sera sous-entendue et nous noterons cet atome $\boldsymbol{x}^*$. Le vecteur de coefficients $\mathbf{w}^{t+1}$ et le résidu $\mathbf{r}^{t+1}$ sont ensuite

**Algorithme 3 :** Matching Pursuit (MP)

**Entrées :** $\mathbf{X} \in \mathbb{R}^{M \times K}, \boldsymbol{y} \in \mathbb{R}^{M}$.
**Sorties :** $\mathbf{w} \in \mathbb{R}^{K}$.
**Initialisation :** $\mathbf{w} \leftarrow \mathbf{0}, \quad \mathbf{r} \leftarrow \boldsymbol{y}$.

**répéter**
Sélection : $j^* = \arg\max_{j \in \{1,\dots,K\}} |\langle \mathbf{r}, \boldsymbol{x}^j \rangle|$
$\boldsymbol{x}^* = \boldsymbol{x}^{j^*}$
Mise à jour : $\mathbf{w}_{j^*} = \mathbf{w}_{j^*} + \langle \mathbf{r}, \boldsymbol{x}^* \rangle$
$\mathbf{r} = \mathbf{r} - \langle \mathbf{r}, \boldsymbol{x}^* \rangle \boldsymbol{x}^*$
**jusqu'à** satisfaire un critère d'arrêt.

mis à jour en conséquence et la procédure se poursuit jusqu'à ce qu'un critère d'arrêt soit satisfait (voir l'Algorithme 3 pour les détails de l'algorithme). La formulation du problème proposée à l'Equation (2.2) amène naturellement l'utilisation d'un critère d'arrêt basé sur l'erreur d'approximation. L'algorithme est ainsi stoppé lorsque la précision souhaitée est atteinte (lorsque la norme du résidu est petite $\left\|\mathbf{r}^t\right\|_2 \leq \epsilon$). Un autre critère d'arrêt couramment utilisé est de mettre fin à la procédure après un nombre $T$ d'itérations défini à l'avance. Dans ce cas, l'algorithme cherche à obtenir la meilleure approximation possible avec un nombre d'atomes fixé et répond à la formulation du problème de la décomposition parcimonieuse suivante :

$$\min_{\mathbf{w} \in \mathbb{R}^K} \ \|\boldsymbol{y} - \mathbf{X}\mathbf{w}\|_2 \ \text{ t.q. } \ \|\mathbf{w}\|_0 \leq T. \tag{2.4}$$

Sauf mention contraire, c'est cette formulation qui sera retenue dans la suite de ce manuscrit.

Quand le dictionnaire considéré ne possède pas de structure particulière, la complexité de l'algorithme Matching Pursuit provient en grande partie de la phase de sélection. Plus précisément, elle provient du calcul des produits scalaires $\langle \boldsymbol{x}^j, \mathbf{r} \rangle$ entre le résidu et tous les atomes du dictionnaire, effectué grâce au produit matriciel $\mathbf{X}^\top \mathbf{r}$. Cette étape implique le calcul de $K$ produits scalaires entre des signaux de taille $M$ ce qui prend $\mathcal{O}(MK)$ opérations (on néglige le parcours de la liste des résultats ($\mathcal{O}(K)$) afin d'extraire l'indice du *meilleur* atome). D'un autre côté, l'étape de mise à jour nécessite seulement $\mathcal{O}(M)$ opérations.

Afin d'accélérer Matching Pursuit, la communauté a principalement suivi trois grandes directions. Premièrement, quand le dictionnaire possède une structure particulière — par exemple lorsqu'il correspond à une transformation rapide comme la transformée de Gabor — il est possible de tirer parti des propriétés de cette structure pour faire diminuer la complexité de calcul des produits scalaires à $\mathcal{O}(K \log M)$. Cependant, dans un cadre plus général, il n'est pas rare que le dictionnaire redondant mis en jeu ne possède pas de telles propriétés (voir Figure 2.1). Il est alors nécessaire d'envisager d'autres stratégies d'accélération. A cette fin, une deuxième voie empruntée a été de se focaliser sur l'efficacité de l'étape de mise à jour : une fois que l'atome a été sélectionné, le but est d'implémenter une stratégie de mise à



FIGURE 2.1 – *Exemple d'un dictionnaire de patchs de taille* $8 \times 8$ *pixels (droite) appris à partir de l'image « Lena » (gauche). Il n'existe pas de moyen rapide de calculer les produits scalaires* $\mathbf{X}^\top \mathbf{r}$ *avec ce type de dictionnaire.*

jour telle que la norme du résidu décroisse plus rapidement que dans le cas de Matching Pursuit. C'est l'objet de Orthogonal Matching Pursuit (OMP, proposé en même temps que Matching Pursuit) et de nombreux autres algorithmes (Gradient Pursuit [Blumensath et Davies 2008]). Ces mises à jour de meilleure qualité (au sens de la décroissance de la norme du résidu) possèdent cependant un coût de calcul qui prend le pas sur le coût de calcul des produits scalaires. La troisième direction s'attaque directement à réduire la complexité de la phase de sélection tout en gardant la stratégie de mise à jour proposée par Matching Pursuit, simple et néanmoins efficace. C'est la stratégie adoptée par de nombreux travaux [Temlyakov 2000, Gribonval et Nielsen 2001, Gribonval et Vandergheynst 2006, Candès et al. 2006, Donoho 2006, Jost et al. 2006, Tjoa et Liu 2011, Moussallam et al. 2012] et c'est également sur celle-ci que nous basons notre algorithme.

Il est important de noter que nous ne faisons aucune hypothèse sur les propriétés du dictionnaire utilisé et que notre but est de conserver une faible complexité de calcul pour chaque étape de notre version modifiée de Matching Pursuit, de façon à ce que chaque itération se fasse rapidement. Comme nous allons le voir, nous proposons une étape de sélection approchée dont le résultat peut être une sélection sous-optimale des atomes. Cette version met donc en œuvre des mises à jour qui amènent une décroissance de la norme du résidu potentiellement moins importante à chaque itération qu'avec Matching Pursuit. Cependant, on peut espérer que le gain obtenu dans le temps de calcul de chaque itération vienne contrebalancer la sous-optimalité des mises à jour, résultant en un algorithme plus rapide dans sa globalité. Avant de présenter l'algorithme, nous voulons préciser que bien que notre procédure de *sélection aléatoire* soit présentée dans le cadre de Matching Pursuit, il est évident qu'elle est applicable à tout algorithme de poursuite reposant sur une étape de sélection basée sur le calcul de corrélations comme c'est le cas dans OMP, GP et bien d'autres.

## 2.2 Matching Pursuit avec sélection aléatoire

### 2.2.1 Algorithme

Notre algorithme de Matching Pursuit avec sélection aléatoire (MP-SA) diffère de la version classique seulement au niveau de l'étape de sélection.

**Algorithme 4 :** Matching Pursuit avec Sélection Aléatoire (MP-SA)

**Entrées :** $\mathbf{X} \in \mathbb{R}^{M \times K}, \boldsymbol{y} \in \mathbb{R}^{M}$, $m \in \{1, \dots, M\}$, $k \in \{1, \dots, K\}$.
**Sorties :** $\mathbf{w} \in \mathbb{R}^{K}$.
**Initialisation :** $\mathbf{w} \leftarrow \mathbf{0}, \quad \mathbf{r} \leftarrow \boldsymbol{y}$.

**répéter**
Tirer aléatoirement :
- un ensemble $\mathcal{M} \subseteq \{1, \dots, M\}$ de $m$ indices de ligne,
- un ensemble $\mathcal{K} \subseteq \{1, \dots, K\}$ de $k$ indices de colonne.

Sélection : $\tilde{j} = \arg\max_{j \in \mathcal{K}} |\langle \mathbf{r}, \boldsymbol{x}^j \rangle_{\mathcal{M}}|$
$\tilde{\boldsymbol{x}} = \boldsymbol{x}^{\tilde{j}}$

Mise à jour : $\mathbf{w}_{\tilde{j}} = \mathbf{w}_{\tilde{j}} + \langle \mathbf{r}, \tilde{\boldsymbol{x}} \rangle$
$\mathbf{r} = \mathbf{r} - \langle \mathbf{r}, \tilde{\boldsymbol{x}} \rangle \tilde{\boldsymbol{x}}$

**jusqu'à** satisfaire un critère d'arrêt.

La sélection aléatoire que nous proposons ici réduit le coût de la recherche classique des corrélations en procédant à deux réductions de dimension :

1. *Sélection aléatoire de colonnes.* Nous considérons uniquement un sous-dictionnaire de $\mathbf{X}$ constitué de $k$ atomes choisis aléatoirement et uniformément parmi les $K$ atomes originaux.
2. *Sélection aléatoire de lignes.* Nous considérons l'approximation $\langle \mathbf{r}, \boldsymbol{x}^j \rangle_{\mathcal{M}}$ de chacun des produits scalaires $\langle \mathbf{r}, \boldsymbol{x}^j \rangle$ basée seulement sur un échantillon de $m$ coordonnées des signaux, choisies elles aussi aléatoirement et uniformément parmi $\{1, \dots, M\}$.

L'algorithme MP-SA prend donc en entrée deux paramètres : $k$ et $m$. On notera ainsi par la suite MP-SA$(k, m)$ une instance particulière de notre algorithme. Une fois le sous-dictionnaire extrait, on en sélectionne l'atome dont le produit scalaire approché avec le résidu courant possède la plus grande amplitude. L'étape de mise à jour est exactement la même que dans MP : une fois l'atome $\tilde{\boldsymbol{x}}$ sélectionné, on utilise le produit scalaire exact $\langle \mathbf{r}, \tilde{\boldsymbol{x}} \rangle$ afin de mettre à jour le vecteur de coefficients et le résidu soit

$$\mathbf{w}_{\tilde{j}}^{t+1} = \mathbf{w}_{\tilde{j}}^{t} + \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle, \tag{2.5}$$

$$\mathbf{r}^{t+1} = \mathbf{r}^t - \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle \tilde{\boldsymbol{x}}. \tag{2.6}$$

Le critère d'arrêt reste aussi identique à la version classique de MP. Les étapes de l'algorithme sont détaillées dans l'Algorithme 4.

L'algorithme proposé par Moussallam et al. [2012] est un cas particulier de notre algorithme dans lequel on ne sélectionne aléatoirement qu'un sous-ensemble des colonnes. Lorsqu'on utilise seulement un sous-ensemble aléatoire des atomes de $\mathbf{X}$, on ne peut être assuré de sélectionner le meilleur atome. Cependant, on espère que le meilleur candidat du sous-ensemble considéré appartient au support de $\boldsymbol{y}$. L'ajout dans notre algorithme du sous-échantillonnage des coordonnées implique que nous n'avons pas accès aux véritables corrélations, y compris celles entre le résidu et les atomes du sous-dictionnaire extrait. Cependant, en renouvelant la sélection aléatoire des atomes et des coordonnées considérées à chaque itération, on permet

une exploration complète du dictionnaire $\mathbf{X}$. Nous verrons dans la suite de ce chapitre que le choix de renouveler le sous-dictionnaire est crucial afin d'obtenir une décomposition de bonne qualité.

L'étape de sélection aléatoire proposée dans l'Algorithme 4 coûte $\mathcal{O}(mk)$ opérations, là où Matching Pursuit en nécessitait $\mathcal{O}(MK)$. Le gain en temps de calcul de chaque itération est donc de l'ordre de $MK/mk$. Ce gain est toutefois à mettre en regard de la perte de précision due à la sous-optimalité des atomes choisis au cours de la procédure. Dans la section 2.3, nous montrons cependant des résultats de simulations numériques qui mettent en avant le gain global en terme de temps de calcul obtenu par l'introduction de sélections aléatoires dans Matching Pursuit.

### 2.2.2 Garanties théoriques

Le résultat principal de cette section établit un lien entre la procédure de sélection aléatoire que nous avons présentée et l'algorithme Weak-$\alpha$-MP [Mallat et Zhang 1993, Temlyakov 2000; 2002, Gribonval et Nielsen 2001, Gribonval et Vandergheynst 2006]. Nous montrons que sous certaines conditions simples, notre algorithme sélectionne avec une grande probabilité un atome pertinent, au sens de Weak-$\alpha$-MP, à chaque itération. Ce résultat est énoncé à la Proposition 2.2. Nous commençons cette section par la présentation de l'algorithme Weak-$\alpha$-MP puis nous montrons deux résultats intermédiaires concernant deux cas particuliers de notre procédure. Dans le premier cas, nous considérons l'algorithme MP-SA$(k, M)$ procédant à un échantillonnage restreint aux atomes. Dans le second cas, nous considérons l'algorithme MP-SA$(K, m)$ procédant à un échantillonnage restreint aux coordonnées. Enfin, nous terminons cette section en assemblant les éléments afin de construire le résultat évoqué ci-dessus.

#### Weak-$\alpha$-MP

La sélection sous-optimale d'atomes a été évoquée à de nombreuses reprises dans les travaux concernant Matching Pursuit. Dès le papier originel [Mallat et Zhang 1993], les auteurs montrent que leur procédure converge ($\|\mathbf{r}^t\|_2 \to 0$ quand $t \to \infty$) même lorsque l'atome sélectionné à chaque itération de l'algorithme n'est pas celui le plus corrélé au résidu. En d'autre termes, pour que la procédure converge il suffit que pour un certain $0 < \alpha \leq 1$ fixé, l'atome $\tilde{\boldsymbol{x}}$ sélectionné à chaque itération $t$ satisfasse

$$|\langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle| \geq \alpha \, |\langle \mathbf{r}^t, \boldsymbol{x}^* \rangle|, \tag{2.7}$$

où

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^K\}}{\arg\max} \; |\langle \mathbf{r}^t, \boldsymbol{x} \rangle|. \tag{2.8}$$

Nous utilisons le terme *$\alpha$-sous-optimal* pour désigner un tel atome dans la suite de ce chapitre. Bien entendu, la vitesse de convergence de l'algorithme dépend de la valeur de $\alpha$. Plus la valeur de $\alpha$ est petite (plus l'atome sélectionné peut être sous-optimal) et plus l'algorithme sera lent à converger. Concernant la parcimonie de la solution, une trop faible valeur de $\alpha$ signifie l'introduction potentielle dans la solution de composantes n'appartenant

pas au support du signal. Nous allons maintenant voir que l'algorithme que nous proposons, dans le cas où l'on considère toutes les coordonnées mais seulement quelques atomes, est une version faible de Matching Pursuit.

### Echantillonnage des atomes

Nous étudions dans un premier temps l'algorithme MP-SA$(k, M)$. Cette version est en tout point similaire à l'algorithme SASMP proposé dans la thèse de Moussallam [2012]. L'auteur propose une caractérisation de cette procédure dans le cas de dictionnaires quasi-incohérents en utilisant des statistiques d'ordre. Nous proposons ici une nouvelle approche probabiliste dans le cas où le dictionnaire considéré possède une certaine cohérence. Nous commençons par rappeler la notion de redondance structurelle d'un dictionnaire [Frossard et Vandergheynst 2001] puis nous introduisons les notions de $\gamma$-*voisinage* et de $p$-$\gamma$-*cohérence* qui sont au coeur de notre analyse.

**Définition 2.1** **(Redondance structurelle)** *La redondance structurelle $\beta$ d'un dictionnaire $\mathbf{X} \in \mathbb{R}^{M\times K}$ est définie par :*

$$\beta = \inf_{\boldsymbol{a}, \|\boldsymbol{a}\|_2=1} \left\{ \sup_{\boldsymbol{x}\in\{\boldsymbol{x}^1,\dots,\boldsymbol{x}^K\}} |\langle \boldsymbol{a}, \boldsymbol{x}\rangle| \right\}. \tag{2.9}$$

Dans un dictionnaire d'atomes unitaires, la redondance structurelle $\beta$ peut être interprétée comme le cosinus de l'angle maximal possible entre n'importe quelle direction de $\mathbb{R}^M$ et l'atome du dictionnaire le plus proche. Les auteurs pointent du doigt que cette quantité tend à se rapprocher de la valeur 1 quand la taille du dictionnaire augmente. Nous définissons maintenant la notion de $\gamma$-voisinage afin de pouvoir caractériser la similarité existant entre les atomes du dictionnaire.

**Définition 2.2** **($\gamma$-voisinage)** *Le $\gamma$-voisinage d'un atome $\boldsymbol{x} \in \{\boldsymbol{x}^1, \dots, \boldsymbol{x}^K\}$, noté $v_\gamma(\boldsymbol{x})$, est l'ensemble des atomes $\boldsymbol{x}'$ du dictionnaire qui vérifient $|\langle \boldsymbol{x}, \boldsymbol{x}'\rangle| \geq \gamma$ :*

$$v_\gamma(\boldsymbol{x}) = \boldsymbol{x} \cup \{\boldsymbol{x}' \in \{\boldsymbol{x}^1, \dots, \boldsymbol{x}^K\} \mid \boldsymbol{x} \neq \boldsymbol{x}' \text{ et } |\langle \boldsymbol{x}, \boldsymbol{x}'\rangle| \geq \gamma\}. \tag{2.10}$$

Cette définition permet d'introduire une condition équivalente à l'Equation (2.7) pour caractériser une version $\alpha$-faible de l'algorithme MP.

**Proposition 2.1** *Soient $\mathbf{X}$ un dictionnaire de redondance structurelle $\beta$ et $A$, un algorithme de poursuite sélectionnant à chaque itération $t$ un atome $\tilde{\boldsymbol{x}} \in \mathbf{X}$ tel que*

$$\tilde{\boldsymbol{x}} \in v_\gamma(\boldsymbol{x}^*). \tag{2.11}$$

*L'algorithme $A$ sélectionne un atome $\alpha$-sous-optimal si $\beta, \gamma \in ]0, 1]$ sont tels que $\gamma^2 + \beta^2 > 1$ avec*

$$\alpha = \gamma\left(\gamma\beta - \sqrt{(1-\gamma^2)(1-\beta^2)}\right). \tag{2.12}$$

*Démonstration.* Considérons les vecteurs suivants : le résidu $\mathbf{r}$, l'atome optimal $\boldsymbol{x}^*$ et l'atome $\tilde{\boldsymbol{x}}$ sélectionné par l'algorithme A. En appliquant un procédé d'orthonormalisation de Gram-Schmidt au triplet $(\boldsymbol{x}^*, \mathbf{r}, \tilde{\boldsymbol{x}})$, nous obtenons

une base orthonormée $\mathcal{B} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ du sous-espace engendré par ces vecteurs telle que

$$\begin{aligned} \boldsymbol{u}_1 &= \boldsymbol{x}^*, & \boldsymbol{e}_1 &= \frac{\boldsymbol{u}_1}{\|\boldsymbol{u}_1\|_2}, \\ \boldsymbol{u}_2 &= \mathbf{r} - \langle \mathbf{r}, \boldsymbol{x}^* \rangle \boldsymbol{x}^*, & \boldsymbol{e}_2 &= \frac{\boldsymbol{u}_2}{\|\boldsymbol{u}_2\|_2}, \\ \boldsymbol{u}_3 &= \tilde{\boldsymbol{x}} - \langle \tilde{\boldsymbol{x}}, \boldsymbol{x}^* \rangle \boldsymbol{x}^* - \frac{\langle \tilde{\boldsymbol{x}}, \boldsymbol{u}_2 \rangle}{\|\boldsymbol{u}_2\|_2} \boldsymbol{u}_2 & \boldsymbol{e}_3 &= \frac{\boldsymbol{u}_3}{\|\boldsymbol{u}_3\|_2}. \end{aligned}$$

Avec cette base, les vecteurs qui nous intéressent s'écrivent $\boldsymbol{x}^* = (1, 0, 0)$, $\mathbf{r} = (\cos\rho \cdot \|\mathbf{r}\|_2, \sin\rho \cdot \|\mathbf{r}\|_2, 0)$ et $\tilde{\boldsymbol{x}} = (\cos\phi\cos\phi', \cos\phi\sin\phi', \sin\phi)$ si $\rho$ est l'angle entre $\mathbf{r}$ et $\boldsymbol{x}^*$, $\phi$ celui entre $\tilde{\boldsymbol{x}}$ et le plan contenant $\mathbf{r}$ et $\boldsymbol{x}^*$ et enfin $\phi'$ l'angle entre la projection de $\tilde{\boldsymbol{x}}$ sur ce plan et $\boldsymbol{x}^*$. Remarquons que si la dimension du sous-espace engendré par ces trois vecteurs est inférieure à trois, il est possible de trouver $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3$ tels que l'écriture ci-dessus reste valable. Les propriétés du dictionnaire nous donnent

$$|\langle \mathbf{r}, \boldsymbol{x}^* \rangle| \geq \beta \cdot \|\mathbf{r}\|_2 \Rightarrow |\cos\rho| \geq \beta \text{ et } |\sin\rho| \leq \sqrt{1-\beta^2}, \tag{2.13}$$

et l'hypothèse de l'Equation (2.11) permet d'écrire

$$\begin{aligned} |\langle \tilde{\boldsymbol{x}}, \boldsymbol{x}^* \rangle| \geq \gamma &\Rightarrow |\cos\phi\cos\phi'| \geq \gamma \\ &\Rightarrow |\cos\phi|, |\cos\phi'| \geq \gamma \text{ et } |\sin\phi|, |\sin\phi'| \leq \sqrt{1-\gamma^2}\,. \end{aligned} \tag{2.14}$$

On peut alors écrire

$$\begin{aligned} |\langle \mathbf{r}, \tilde{\boldsymbol{x}} \rangle| &= |\cos\rho\cos\phi\cos\phi' + \sin\rho\cos\phi\sin\phi'| \cdot \|\mathbf{r}\|_2 \\ &= |\cos\phi|\,|\cos\rho\cos\phi' + \sin\rho\sin\phi'| \cdot \|\mathbf{r}\|_2 \end{aligned} \tag{2.15}$$

En utilisant les équations (2.13) et (2.14) on montre que

$$\begin{aligned} |\cos\rho\cos\phi' + \sin\rho\sin\phi'| &\geq \big||\cos\rho\cos\phi'| - |\sin\rho\sin\phi'|\big| \\ &\geq |\cos\rho\cos\phi'| - |\sin\rho\sin\phi'| \qquad (2.16) \\ &\geq \left(\gamma\beta - \sqrt{(1-\gamma^2)(1-\beta^2)}\right). \qquad (2.17) \end{aligned}$$

Cela nous donne donc la valeur minimale suivante pour l'amplitude de la corrélation entre $\tilde{\boldsymbol{x}}$ et $\mathbf{r}$

$$|\langle \mathbf{r}, \tilde{\boldsymbol{x}} \rangle| \geq \gamma \left(\gamma\beta - \sqrt{(1-\gamma^2)(1-\beta^2)}\right) \cdot \|\mathbf{r}\|_2\,. \tag{2.18}$$

Supposons maintenant que

$$\alpha = \gamma \left(\gamma\beta - \sqrt{(1-\gamma^2)(1-\beta^2)}\right) > 0$$

alors en utilisant $|\langle \mathbf{r}, \boldsymbol{x}^* \rangle| \leq \|\mathbf{r}\|_2$ on est en mesure d'écrire

$$|\langle \mathbf{r}, \tilde{\boldsymbol{x}} \rangle| \geq \alpha \cdot \|\mathbf{r}\|_2 \geq \alpha |\langle \mathbf{r}, \boldsymbol{x}^* \rangle|.$$

On termine la preuve en constatant que $\alpha$ est strictement positif pour

$$\begin{aligned} & \gamma\beta > \sqrt{1-\gamma^2}\sqrt{1-\beta^2} \\ \Leftrightarrow \quad & \gamma^2\beta^2 > (1-\gamma^2)(1-\beta^2) \\ \Leftrightarrow \quad & \gamma^2+\beta^2 > 1 \end{aligned} \tag{2.19}$$

□

On définit maintenant une notion qui caractérise les similarités entre les atomes présents dans le dictionnaire $\mathbf{X}$. Cette notion est plus riche que la notion de cohérence définie habituellement dans les travaux liés aux algorithmes gloutons pour la représentation parcimonieuse de signaux.

**Définition 2.3** **(p-$\gamma$-cohérence)** *Un dictionnaire $\mathbf{X} \in \mathbb{R}^{M\times K}$ est dit p-$\gamma$-cohérent si pour tout atome $\boldsymbol{x}$ appartenant au dictionnaire, la taille de son $\gamma$-voisinage est au moins p :*

$$\forall \boldsymbol{x} \in \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^K\},\ card\Big(v_\gamma(\boldsymbol{x})\Big) \geq p. \tag{2.20}$$

A partir des définitions précédentes, nous pouvons établir un résultat simple sur la probabilité que la procédure MP-SA$(k, M)$ sélectionne à l'itération $t$ un atome $\tilde{\boldsymbol{x}}$ appartenant au $\gamma$-voisinage du meilleur atome $\boldsymbol{x}^*$ (Equation (2.8)). Autrement dit, le résultat suivant quantifie la propension de la procédure MP-SA à sélectionner un atome $\alpha$-sous-optimal dans le cas d'une sélection aléatoire restreinte aux atomes uniquement. Il est important de noter que contrairement à la plupart des travaux sur le sujet qui supposent une faible cohérence du dictionnaire, notre approche tire parti de la propriété inverse.

**Proposition 2.2** *Soit $\mathbf{X} \in \mathbb{R}^{M\times K}$ un dictionnaire p-$\gamma$-cohérent de redondance structurelle $\beta$. Si $\beta^2+\gamma^2 > 1$ alors la probabilité que l'algorithme MP-SA$(k, M)$ sélectionne à chacune de ses $T$ itérations un atome $\alpha$-sous-optimal est telle que*

$$\mathbb{P}\left[\forall t \in \{1,\ldots,T\}, \tilde{\boldsymbol{x}} \in v_\gamma(\boldsymbol{x}^*)\right] \geq 1 - T\prod_{i=0}^{k-1}\left[1-\frac{p}{K-i}\right], \tag{2.21}$$

*avec*

$$\alpha = \gamma\left(\gamma\beta - \sqrt{(1-\gamma^2)(1-\beta^2)}\right). \tag{2.22}$$

*Démonstration.* On commence par calculer la probabilité qu'aucun des $p$ atomes du $\gamma$-voisinage de $\boldsymbol{x}^*$ ne soit sélectionné à l'itération $t$. Cette probabilité s'obtient en considérant le processus de tirage aléatoire sans remise suivant. On commence par regarder la probabilité de ne pas sélectionner un de ces $p$ atomes dans l'ensemble du dictionnaire : $1-\frac{p}{K}$. Puis on calcule la probabilité de ne pas tirer un atome du $\gamma$-voisinage de $\boldsymbol{x}^*$ parmi les $K-1$ atomes restants dans le dictionnaire : $1-\frac{p}{K-1}$. On répète cette procédure jusqu'à avoir tiré $k$ atomes dans le sous-dictionnaire et on obtient

$$\mathbb{P}\left[\tilde{\boldsymbol{x}} \notin v_\gamma(\boldsymbol{x}^*)\right] = \prod_{i=0}^{k-1}\left[1-\frac{p}{K-i}\right].$$

En supposant que la procédure réalise $T$ itérations et en utilisant une borne de l'union, on montre que la probabilité que MP-SA$(k, M)$ ne sélectionne pas un atome appartenant au $\gamma$-voisinage de l'atome optimal à au moins une des itérations est

$$\mathbb{P}\left[\exists t \in \{1,\dots,T\}, \tilde{\boldsymbol{x}} \notin v_\gamma(\boldsymbol{x}^*)\right] \leq T \prod_{i=0}^{k-1}\left[1 - \frac{p}{K-i}\right].$$

Il suffit alors de prendre la probabilité de l'évènement contraire pour terminer la preuve de la proposition. □

**Remarque 2.1** *Il suffit de prendre $k = K - p + 1$ pour être assuré de sélectionner un atome appartenant au $\gamma$-voisinage de $\boldsymbol{x}^*$ et donc de sélectionner un atome $\alpha$-sous-optimal avec $\alpha$ tel qu'à l'équation (2.22).*

### Echantillonage des coordonnées

Nous regardons maintenant le cas de figure où la procédure sélectionne tous les atomes du dictionnaire à chaque itération mais seulement une partie des coordonnées. Nous notons cette procédure MP-SA$(K, m)$. Nous aimerions tout d'abord attirer l'attention sur d'autres travaux qui eux aussi permettent de caractériser notre procédure. Il s'agit des travaux de Drineas et al. [2006] sur l'échantillonnage par importance (« Importance sampling ») et les propriétés des algorithmes stochastiques pour approcher le produit de deux matrices. Leur méthode s'applique presque directement dans le cas de figure qui nous intéresse mais leur borne est plus lâche que la nôtre. De plus, leur algorithme suppose un échantillonnage avec remise là où nous utilisons explicitement le fait que notre procédure effectue un tirage aléatoire sans remise. Le théorème suivant montre que le calcul des produits scalaires partiels (utilisant un sous-ensemble des coordonnées) entre le résidu et les atomes est une bonne approximation des produits scalaires réels.

**Théorème 2.1** *Soient un vecteur $\mathbf{r} \in \mathbb{R}^M$, une matrice $\mathbf{X} = [\boldsymbol{x}^1 \dots \boldsymbol{x}^K] \in \mathbb{R}^{M\times K}$, $\mathcal{M}$ un sous-ensemble de coordonnées de taille $|\mathcal{M}| = m$, $\mu = m/M$ et $\delta \in ]0,1]$. Alors, avec probabilité au moins $1-\delta$, $\forall j \in \{1,\dots,K\}$ :*

$$\left|\frac{\langle \mathbf{r}, \boldsymbol{x}^j\rangle_{\mathcal{M}}}{\mu} - \langle \mathbf{r}, \boldsymbol{x}^j\rangle\right| \leq \max_j\left(\|\boldsymbol{x}^j\|_\infty\right)\|r\|_\infty \sqrt{\frac{2\ln\left(\frac{2K}{\delta}\right)(M-m+1)}{\mu}}. \quad (2.23)$$

*Démonstration.* La preuve fait intervenir une inégalité de concentration classique présentée dans [Serfling 1974] concernant la somme de variables aléatoires tirées sans replacement. Cette inégalité est rappelée dans le lemme suivant.

**Lemme 2.1** *On considère le tirage aléatoire sans replacement à partir d'une liste $v_1,\dots,v_N$ de valeurs telles que $a \leq v_i \leq b, \forall i \in \{1,\dots,N\}$. On note $V_1,\dots,V_n$ les valeurs d'un échantillon de taille $n$ tiré aléatoirement sans remise et on définit les sommes $S_n = \sum_{i=1}^n V_i$ et $\bar{v}_N = \frac{1}{N}\sum_{i=1}^N v_i$, respectivement la somme de l'échantillon et la moyenne de la liste complète. En*

*notant* $f_n^* = \frac{(n-1)}{N}$ *on a*

$$\mathbb{P}\left[|S_n - n\bar{v}_N| \geq \epsilon\right] \leq 2\exp\left(\frac{-2n\epsilon^2}{(1-f_n^*)(b-a)^2}\right). \tag{2.24}$$

Pour un couple $(\boldsymbol{x}, \mathbf{r}) \in \mathbb{R}^M \times \mathbb{R}^M$ de vecteurs, considérons la liste de valeurs $v_1, \ldots, v_M$ telle que $v_i = \boldsymbol{x}_{(i)} \times \mathbf{r}_{(i)}$ ($v_i$ est la contribution de la $i^{eme}$ coordonnée au produit scalaire $\langle \boldsymbol{x}, \mathbf{r} \rangle$). Il est facile de voir que $|v_i| \leq \|\boldsymbol{x}\|_\infty \|\mathbf{r}\|_\infty$. En notant

$$\frac{\delta}{K} = \exp\left(\frac{-2n\epsilon^2}{(1-f_n^*)(b-a)^2}\right),$$

une utilisation directe du Lemme 2.1 sur la liste $x_1, \ldots, x_M$ donne le résultat suivant pour un atome :

$$\mathbb{P}\left[\left|\frac{\langle \mathbf{r}, \boldsymbol{x} \rangle_{\mathcal{M}}}{m} - \frac{\langle \mathbf{r}, \boldsymbol{x} \rangle}{M}\right| \geq \|\boldsymbol{x}\|_\infty \|r\|_\infty \sqrt{\frac{2\ln\left(\frac{2K}{\delta}\right)(M-m+1)}{Mm}}\right] \leq \frac{\delta}{K}.$$

On termine la preuve en utilisant le fait que $\forall j \in \{1, \ldots, K\}, \|\boldsymbol{x}^j\|_\infty \leq \max_j\left(\|\boldsymbol{x}^j\|_\infty\right)$, couplé à une borne de l'union sur les $K$ atomes du dictionnaire. □

**Remarque 2.2** *La qualité de cette approximation dépend de la fraction $\mu$ des coordonnées utilisées pour le calcul du produit scalaire partiel. Cependant, on peut remarquer que lorsque $m$ tend vers $M$ ($\mu$ tend vers 1), c'est-à-dire lorsque toutes les coordonnées tendent à être considérées, le membre de droite de l'inégalité (2.23) ne s'évanouit pas mais tend vers*

$$\max_j\left(\|\boldsymbol{x}^j\|_\infty\right) \|r\|_\infty \sqrt{2\ln\left(\frac{2K}{\delta}\right)} \neq 0.$$

*Cela est malheureusement inhérent à ce type d'inégalité de concentration et aux techniques de preuve utilisées. La Figure 2.2 illustre le comportement de la borne sur un exemple dans le cas où $\max_j\left(\|\boldsymbol{x}^j\|_\infty\right) = \|r\|_\infty = 1$. On y remarque le comportement évoqué ci-dessus, à savoir que la courbe se rapproche d'une erreur nulle sans pour autant l'atteindre même lorsque toutes les coordonnées ont été vues.*

Malgré cette remarque, il faut retenir que le théorème précédent porte le message qu'il est possible, avec une grande probabilité, de contrôler simultanément la précision des approximations des produits scalaires calculés à partir d'un échantillon aléatoire de coordonnées. Il est évident que plus la taille de l'échantillon est grande, meilleure est la qualité des approximations. Ce constat met en évidence le compromis qui existe entre temps de calcul et précision de la solution recherchée. Il est cependant normal d'espérer qu'avec une grande probabilité un atome sous-optimal mais néanmoins pertinent soit sélectionné à chaque itération tout en maintenant une réduction significative du temps de calcul. Nous quantifions à présent la pertinence de cette sélection, en tissant un lien entre notre algorithme et la version faible de Matching Pursuit présentée dans [Mallat et Zhang 1993].

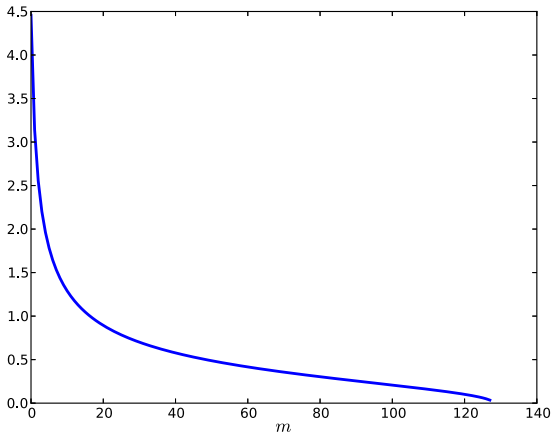



FIGURE 2.2 – *Comportement de la borne de l'Equation* (2.23) *en fonction de* $m$ *avec les valeurs suivantes :* $M = 128$, $K = 512$, $\delta = 0.05$ *et* $\max_j \left(\|\boldsymbol{x}^j\|_\infty\right) = \|r\|_\infty = 1$.

**Proposition 2.3** *Posons (borne obtenue au Théorème 2.1)*

$$\epsilon = \max_j \left(\|\boldsymbol{x}^j\|_\infty\right) \|r\|_\infty \sqrt{\frac{2\ln\left(\frac{2K}{\delta}\right)(M-m+1)}{\mu}}. \tag{2.25}$$

*Alors, MP-SA*$(K, m)$ *réalisant* $T$ *itérations sélectionne avec probabilité au moins* $(1-\delta)^T$ *un atome* $\alpha$*-sous-optimal à chaque itération* $t$ *si on a*

$$\left|\langle \mathbf{r}^t, \tilde{\boldsymbol{x}}\rangle_{\mathcal{M}}\right| \geq \frac{\mu(1+\alpha)}{1-\alpha}\epsilon. \tag{2.26}$$

En notant la dépendance en $m$ de $\epsilon$, cette inégalité peut être interprétée de deux manières. La première est la suivante : en évaluant à chacune des $T$ itérations les quantités mises en jeu, on est en mesure de trouver le plus petit $\alpha$ au cours de la procédure tel que l'Equation (2.26) soit vraie, cet $\alpha$ permet alors de caractériser avec grande probabilité la qualité de la solution obtenue par l'algorithme MP-SA$(K, m)$ après $T$ itérations par rapport à celle qu'aurait obtenue MP. La deuxième interprétation est comme suit : on fixe une valeur de $\alpha$ que l'on souhaite obtenir pour notre procédure, le but est atteint avec grande probabilité si l'on est en mesure d'adapter $m$ à chaque itération de sorte que l'équation précédente soit vérifiée pour chacune d'entre elles. Ce deuxième point de vue relevant d'une version « adaptative » de notre procédure, différente de celle proposée jusqu'ici, nous nous focalisons sur le premier.

*Démonstration.* Rappelons dans un premier temps l'inégalité de concentration exposée au Théorème 2.1. Nous avons montré qu'avec une probabilité au moins égale à $1-\delta$, pour $\delta \in ]0, 1]$, on a l'inégalité suivante

$$\left|\frac{\langle \mathbf{r}, \boldsymbol{x}^j\rangle_{\mathcal{M}}}{\mu} - \langle \mathbf{r}, \boldsymbol{x}^j\rangle\right| \leq \max_j \left(\|\boldsymbol{x}^j\|_\infty\right) \|r\|_\infty \sqrt{\frac{2\ln\left(\frac{2K}{\delta}\right)(M-m+1)}{\mu}},$$

simultanément pour tous les atomes $\boldsymbol{x}^j$. Soit $\boldsymbol{x}^*$ le meilleur atome à l'itération $t$ courante (voir Equation (2.8)) et $\tilde{\boldsymbol{x}}$ l'atome le plus proche du résidu courant

dans l'espace restreint aux coordonnées dans $\mathcal{M}$ :

$$\tilde{\boldsymbol{x}} = \underset{\boldsymbol{x}^j \in \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^K\}}{\arg\max} \left( \left| \langle \mathbf{r}^t, \boldsymbol{x}^j \rangle_{\mathcal{M}} \right| \right). \tag{2.27}$$

En utilisant l'Equation (2.23) et l'inégalité triangulaire $| \, |a| - |b| \, | \leq |a - b|$, on peut écrire avec grande probabilité

$$\frac{1}{\mu} \left| \langle \mathbf{r}^t, \boldsymbol{x}^* \rangle_{\mathcal{M}} \right| - \epsilon \leq \left| \langle \mathbf{r}^t, \boldsymbol{x}^* \rangle \right| \leq \frac{1}{\mu} \left| \langle \mathbf{r}^t, \boldsymbol{x}^* \rangle_{\mathcal{M}} \right| + \epsilon, \tag{2.28}$$

$$\frac{1}{\mu} \left| \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle_{\mathcal{M}} \right| - \epsilon \leq \left| \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle \right| \leq \frac{1}{\mu} \left| \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle_{\mathcal{M}} \right| + \epsilon. \tag{2.29}$$

Pour que MP-SA$(K, m)$ sélectionne un atome $\alpha$-sous-optimal, il suffit que l'atome sélectionné par notre algorithme à chaque itération soit proche de l'atome qui aurait été sélectionné par Matching Pursuit. Précisément, nous voulons pour $0 < \alpha \leq 1$

$$\left| \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle \right| \geq \alpha \left| \langle \mathbf{r}^t, \boldsymbol{x}^* \rangle \right|. \tag{2.30}$$

On voit par l'Equation (2.28) qu'une condition suffisante pour avoir avec grande probabilité (2.30) est

$$\left| \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle \right| \geq \alpha \left( \frac{1}{\mu} \left| \langle \mathbf{r}^t, \boldsymbol{x}^* \rangle_{\mathcal{M}} \right| + \epsilon \right). \tag{2.31}$$

Nous pouvons maintenant utiliser la borne inférieure donnée par l'Equation (2.29) sur le membre de gauche de l'équation précédente pour introduire le produit scalaire partiel de l'atome $\tilde{\boldsymbol{x}}$. La condition suffisante pour obtenir (2.30) devient alors

$$\frac{1}{\mu} \left| \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle_{\mathcal{M}} \right| - \epsilon \geq \alpha \left( \frac{1}{\mu} \left| \langle \mathbf{r}^t, \boldsymbol{x}^* \rangle_{\mathcal{M}} \right| + \epsilon \right). \tag{2.32}$$

Par définition de $\tilde{\boldsymbol{x}}$ on a $\left| \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle_{\mathcal{M}} \right| \geq \left| \langle \mathbf{r}^t, \boldsymbol{x}^* \rangle_{\mathcal{M}} \right|$. En injectant ce résultat dans l'équation précédente, on obtient que dès que

$$\epsilon \leq \frac{1 - \alpha}{1 + \alpha} \left( \frac{1}{\mu} \left| \langle \mathbf{r}^t, \tilde{\boldsymbol{x}} \rangle_{\mathcal{M}} \right| \right), \tag{2.33}$$

alors une itération de notre algorithme entre dans le paradigme de Weak-$\alpha$-MP avec une probabilité au moins $1 - \delta$. Pour pouvoir quantifier la qualité de la solution trouvée par l'algorithme MP-SA$(K, m)$ par rapport à MP, il suffit que cette condition soit vraie à chaque itération, d'où l'apparition de la probabilité $(1 - \delta)^T$ qui vient terminer la preuve. □

### Echantillonnage des atomes et des coordonnées

Nous sommes maintenant en mesure de montrer les conditions suffisantes pour que la procédure MP-SA sélectionne à chacune de ses itérations un atome $\alpha$-sous-optimal.

**Théorème 2.2** *Soit* $\mathbf{X}$ *un dictionnaire p-*$\gamma$*-cohérent de redondance structurelle* $\beta$*. Si* $\epsilon$*, comme défini à l'Equation (2.25), est tel que pour chaque itération la condition de l'Equation (2.33) est vérifiée et si* $\gamma^2 + \beta^2 > 1$ *alors l'algorithme*

*MP-SA sélectionne à chacune de ses $T$ itérations un atome $\alpha$-sous-optimal avec une probabilité au moins égale à*

$$\begin{cases} (1-\delta)^T \left(1 - \prod_{i=0}^{k-1} \left[1 - \frac{p}{K-i}\right]\right)^T & \text{si } k < K-p+1, \\ (1-\delta)^T & \text{sinon,} \end{cases} \tag{2.34}$$

*avec*

$$\alpha = \gamma \left(\gamma\beta - \sqrt{(1-\gamma^2)(1-\beta^2)}\right).$$

*Démonstration.* Ce théorème s'obtient simplement en notant que MP-SA sélectionne un atome $\alpha$-sous-optimal si, à chaque itération, un atome appartenant au $\gamma$-voisinage de $\boldsymbol{x}^*$ est présent dans le sous-dictionnaire considéré et que c'est cet atome qui est sélectionné sur la base du produit scalaire partiel. □

Cette condition est bien sûr une condition dans le « pire » des cas et elle conduit à une probabilité qui décroît rapidement quand le nombre d'itérations augmente. Une analyse plus fine du processus aléatoire mis en jeu ici permettrait sans doute d'obtenir un résultat théorique moins pessimiste. Car en pratique, comme nous allons le voir dans la section suivante, MP-SA permet dans de nombreux cas d'obtenir une approximation comparable à celle obtenue par MP.

## 2.3 Applications numériques

Dans cette section, nous présentons les résultats de simulations numériques réalisées dans le but de conforter notre approche. Dans une première expérience, nous générons un dictionnaire aléatoire redondant et étudions la qualité de l'approximation fournie par notre algorithme en évaluant la décroissance de l'énergie du résidu en fonction du nombre d'itérations et du temps. Nous comparons notre approche de Matching Pursuit avec sélections aléatoires (MP-SA) avec la version standard de l'algorithme de Matching Pursuit (MP). Nous utilisons ensuite dans une autre expérience un dictionnaire DCT afin de valider la capacité de l'approche (MP-SA) à retrouver de façon exacte le support parcimonieux d'un signal. Enfin, une troisième expérience montrera comment l'introduction de l'étape de sélection aléatoire dans l'algorithme OMP permet d'accélérer le *declipping* d'un signal audio.

### 2.3.1 Données synthétiques

Dans les deux sous-sections qui suivent, nous utilisons des signaux de longueur $M = 128$ et deux types de dictionnaires. D'un côté, nous générons un dictionnaire aléatoire $\mathbf{X} \in \mathbb{R}^{M \times K}$ : les entrées de la matrice $\mathbf{X}$ sont des variables aléatoires indépendantes et identiquement distribuées à partir d'une distribution Gaussienne. La matrice est ensuite normalisée afin que les atomes soient de norme $l_2$ unitaire. D'un autre côté, on utilise des dictionnaires DCT IV standards. Dans ce qui suit, le vecteur de coefficients recherché $\mathbf{w}^*$ est un vecteur aléatoire dont les $K_0$ entrées non-nulles sont des variables aléatoires Gaussiennes i.i.d.

### Décroissance de l'énergie du résidu

Dans cette expérience, on mesure la décroissance de l'énergie du résidu $\|\mathbf{r}\|_2^2$ par rapport à l'énergie du signal $\|\boldsymbol{y}\|_2^2$ en fonction des itérations de l'algorithme et du temps passé par celui-ci dans l'étape de sélection. On représente cette décroissance dans la Figure 2.3 et la Figure 2.4 en utilisant une mesure proche du rapport signal-à-bruit

$$\mathrm{SNR}_{dB}(\boldsymbol{y}, \mathbf{r}^t) = 10 \log_{10} \left( \frac{\|\mathbf{r}^t\|_2^2}{\|\boldsymbol{y}\|_2^2} \right).$$

Nous comparons notre approche avec l'algorithme Matching Pursuit standard et une variante de notre approche avec sélection aléatoire que nous appelons MP-$S_0$. Dans cette variante, un sous-dictionnaire est extrait une fois pour toute au début de l'algorithme (en sélectionnant aléatoirement de manière uniforme un même ratio des lignes et des colonnes du dictionnaire entier). La spécificité de cette version réside dans le fait que ce sous-dictionnaire n'est pas renouvelé au cours des itérations. Une fois qu'un atome a été sélectionné, nous effectuons les mêmes mises à jour que celles proposées dans les Algorithmes 3 et 4 (i.e. nous utilisons toute l'information contenue dans l'atome). La complexité de la phase de recherche du meilleur atome de cette variante est strictement identique à celle obtenue par notre procédure aléatoire MP-SA. Elle joue donc le rôle de version de référence qui nous permet de démontrer l'utilité de renouveler le sous-dictionnaire considéré à chaque itération.

Pour la procédure MP-SA, on note $\mu$ (resp. $\kappa$) le ratio $m/M$ (resp. $k/K$). Nous avons conduit des expériences en utilisant un large spectre de configurations ($\mu, \kappa \in [0.2, 1.0]$). De grandes valeurs de ces paramètres impliquent une décroissance du résidu à chaque itération proche de celle obtenue avec MP mais un faible gain en temps de calcul. De petites valeurs impliquent quant à elles une moindre décroissance par itération mais un gain plus important sur le temps de calcul de chaque itération. Nous reportons uniquement dans les Figures 2.3 et 2.4 les résultats pour les cas où $\mu \times \kappa = 0.36$ et $K = 512$. Nous utilisons trois configurations de MP-SA, toutes les trois possédant une complexité calculatoire identique. La première configuration tire parti de tous les atomes du dictionnaire mais ne considère que 36% des coordonnées à chaque itération ($m = 0.36 \times M$). La deuxième considère toutes les dimensions d'un sous-dictionnaire composé de seulement $k = 0.36 \times K$ atomes. Enfin, la dernière configuration échantillonne le dictionnaire de la même manière sur les colonnes et sur les lignes : $\mu = \kappa = 0.6$. Pour chaque jeu de paramètres, on moyenne les performances obtenues par les différents algorithmes sur une série de 20 répétitions.

Il est clair sur les courbes obtenues lors de ces expériences que la procédure MP-$S_0$ devient rapidement incapable de faire décroître l'énergie du résidu. Celle-ci atteint rapidement une asymptote, résultant en une approximation de qualité médiocre de la solution. Ce constat vient mettre en avant l'échec de l'approche MP-$S_0$ et, par conséquent, l'importance dans notre procédure de renouveler le sous-dictionnaire à chaque itération. Intuitivement,

l'information contenue dans le sous-dictionnaire est rapidement consommée jusqu'à manquer, résultant en une stabilisation de l'approximation autour d'une solution peu convaincante : la projection sur le sous-espace engendré par le sous-dictionnaire. Au contraire, avec une complexité calculatoire par itération identique, MP-SA tire avantageusement parti du renouvellement de l'échantillon extrait à chaque itération et ne souffre en aucun cas de ce phénomène d'information limitée.

La Figure 2.3 expose les résultats des expériences menées avec une représentation dont le support est de taille $K_0 = 32$. Comme nous pouvons le constater, avec MP-SA, l'énergie du résidu diminue moins rapidement au cours des itérations qu'avec Matching Pursuit (Figure 2.3 haut). Ce comportement s'explique facilement par le fait que MP-SA ne sélectionne pas toujours le meilleur atome (le plus corrélé, celui conduisant à la décroissance du résidu la plus importante) au cours des itérations, alors que c'est précisément le but poursuivi par MP. Cependant, comme les itérations de MP-SA nécessitent moins d'opérations que celles de MP, on peut remarquer que l'énergie du résidu décroît plus vite dans le temps avec notre nouvelle procédure aléatoire (Figure 2.3 bas). On peut remarquer que les performances obtenues par MP-SA sont meilleures que celles obtenues par MP quel que soit le paramétrage utilisé. Cet avantage est ici encore plus marqué pour des valeurs de $\mu$ et $\kappa$ telles que $\mu = 1$ et $\kappa = 0.36$, c'est-à-dire avec un calcul complet des produits scalaires pour un sous-ensemble d'atomes. Sur la figure, on peut noter qu'avec cette version de MP-SA une décroissance de l'énergie du résidu de l'ordre de 120dB ($\|\mathbf{r}\|_2 \,/\, \|\boldsymbol{y}\|_2 = 10^{-6}$) est obtenue plus de deux fois plus rapidement qu'avec MP. Cette amélioration constatée est en accord avec le fait que l'on n'effectue que 36% des opérations effectuées par MP à chaque itération. Notons enfin que cette version de MP-SA correspond précisément à l'algorithme proposé par Moussallam et al. [2012].

Le comportement est quelque peu différent avec le paramétrage utilisé pour générer la Figure 2.4. Ici, on se place dans le cas d'un support extrêmement réduit puisque l'on fixe à $K_0 = 2$ le nombre de coefficients non-nuls dans la représentation de $\boldsymbol{y}$. Dans cette configuration, la version de MP-SA usant de tous les atomes mais ne considérant que 36% des coordonnées à chaque itération est la seule compétitive avec MP. Ici, seuls 2 atomes entrent dans la représentation du signal et le prix à payer lorsque ces atomes ne sont pas présents dans le sous-dictionnaire considéré à chaque itération est très important. Dans le même temps, on se rend compte que 36% des coordonnées permettent de discriminer efficacement les bons atomes des mauvais dans ce cas de figure. L'atome sélectionné par MP-SA est identique à celui sélectionné par MP. Il en résulte une mise à jour quasi-identique à celle effectuée par MP au cours des itérations (les courbes de décroissance du résidu semblent superposées) mais, ici encore, MP-SA atteint bien plus rapidement le même niveau de performance que son rival. Ici, l'approche de Moussallam et al. [2012] ($\mu = 1$ et $\kappa = 0.36$) éprouve de grandes difficultés à faire décroître l'énergie du résidu. Celle-ci semble, pour ce type de problème, trop sensible à l'absence répétée dans le sous-dictionnaire d'un atome du support. On observe le même type de comportement avec le dernier paramétrage ($\mu = \kappa = 0.6$). Les performances légèrement meilleures de cette dernière

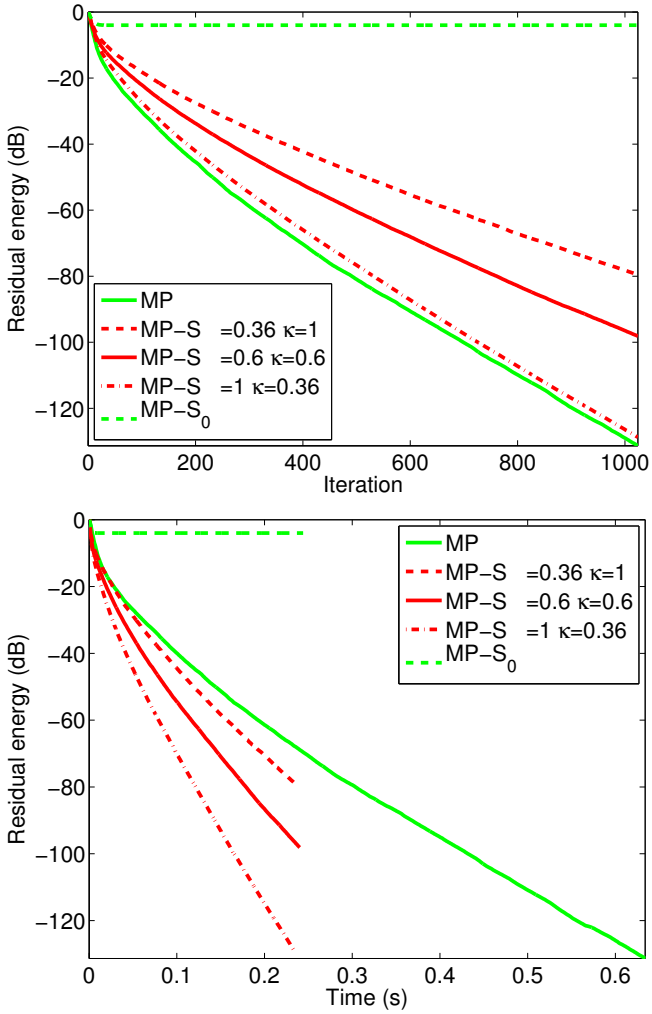



FIGURE 2.3 – *Décroissance de l'énergie du résidu (dictionnaire aléatoire, $K_0 = 32$).*

version confortent l'hypothèse d'une hyper-sensibilité de la procédure MP-SA, dans le cas d'un vecteur de coefficients extrêmement parcimonieux, à la présence dans le sous-dictionnaire d'atomes du support.

### Identification exacte du support

Dans cette section, nous évaluons la capacité de notre approche à retrouver le support d'un signal et la comparons à l'approche standard. Nous nous intéressons seulement au problème de l'identification *exacte* d'un support parcimonieux et nous faisons encore l'hypothèse que le signal n'est pas bruité : $\boldsymbol{y} = \mathbf{X}\mathbf{w}$. Le dictionnaire utilisé dans cette section est un dictionnaire DCT IV. En d'autre termes, chaque entrée de la matrice $\mathbf{X} \in \mathbb{R}^{M \times K}$ constituant le dictionnaire est de la forme

$$x_{i,j} = cos\left(\frac{\pi}{M}\left(i + \frac{1}{2}\right)\left(j + \frac{1}{2}\right)\right),$$

avant que chaque colonne soit normalisée. Afin de visualiser plus facilement les conditions de la réussite de notre approche, nous représentons les résultats sous la forme de diagrammes de transition de phase. Le critère

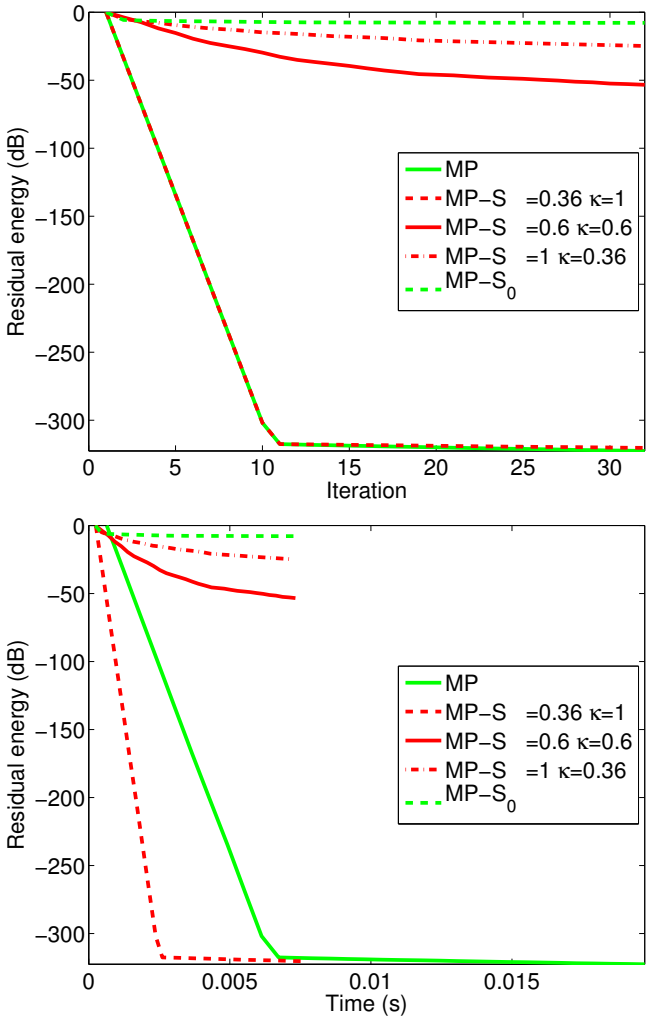


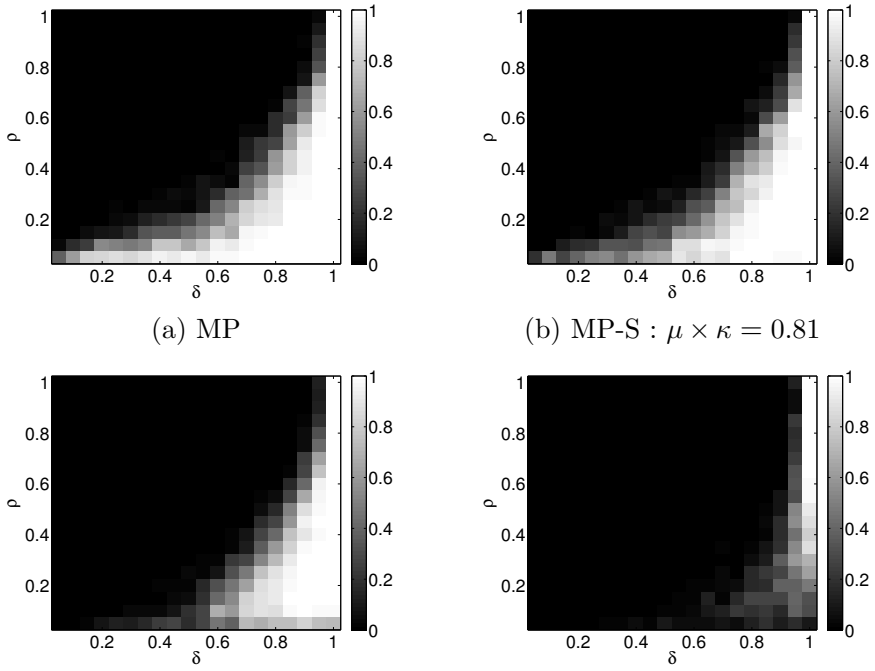

FIGURE 2.4 – *Décroissance de l'énergie du résidu (dictionnaire aléatoire, $K_0 = 2$).*

de performance que nous utilisons est la corrélation normalisée entre les vecteurs de coefficients estimés et véritables. On considère que l'algorithme retrouve de façon exacte le support si cette corrélation est au-dessus de 99%. Nous avons testé de nombreuses configurations et reporté les succès obtenus en fonction de la redondance du dictionnaire $\delta = M/K$ et de la parcimonie $\rho = K_0/M$ du vecteur recherché. Pour chaque couple $(\delta, \rho)$, nous avons lancé l'expérience à 50 reprises en stoppant la procédure avant d'insérer une $(K_0+1)^{ème}$ valeur non nulle dans le vecteur de coefficients. On représente la proportion de succès au cours des répétitions avec une échelle de couleur allant du blanc (succès pour les 50 répétitions) au noir (échec pour toutes les répétitions). Ici, nous ne cherchons pas à optimiser le choix des paramètres pour la procédure MP-SA et choisissons le paramétrage médian $\mu = \kappa$.

La Figure 2.5 montre les diagrammes de transition de phase pour plusieurs valeurs de $\mu \times \kappa$. Premièrement, la différence est très légère entre MP et MP-SA lorsque $\mu \times \kappa = 0.81$ (voir (a) et (b)) c'est-à-dire lorsque le sous-dictionnaire considéré est encore assez proche du dictionnaire com-

plet utilisé par MP. On peut noter que les différences apparaissent dans le régime de dictionnaires très redondants. Ici, beaucoup d'atomes sont quasiment identiques et un atome n'appartenant pas au support peut facilement prendre la place d'un atome du support dont il est très proche. Notre approche semble souffrir un peu plus de ce phénomène que Matching Pursuit à cause sans doute de l'utilisation du produit scalaire approché. Un paramétrage plus fin de l'algorithme permettrait sans doute d'approcher (voire d'égaler) les performances de MP.

La redondance du dictionnaire semble à nouveau être un paramètre crucial dans le succès de notre procédure lorsqu'on fait décroitre le nombre de lignes et de colonnes considérées (voir (c) et (d)). Une analyse théorique plus poussée de l'algorithme permettrait sans doute d'expliquer un peu plus précisément cette sensibilité.




(a) MP (b) MP-S : $\mu \times \kappa = 0.81$

(c) MP-S : $\mu \times \kappa = 0.36$ (d) MP-S : $\mu \times \kappa = 0.09$

FIGURE 2.5 – *Identification du support en fonction de la redondance du dictionnaire et de la parcimonie du support pour différentes valeurs du produit* $\mu \times \kappa$.

### 2.3.2 Données réelles : declipping d'un signal audio

Dans le but d'évaluer le comportement de notre approche sur des données réelles, nous avons implémenté l'étape de sélection aléatoire au sein de l'algorithme OMP que nous avons utilisé dans le cadre d'un problème de declipping d'un signal audio. Ce problème a pour but la reconstruction des parties manquantes du spectre d'un signal audio qui a été écrêté. La Figure 2.6 illustre un tel problème.

Nous nous sommes placés dans le cadre de l'expérience conduite sur un signal de parole échantillonné à 8kHz proposée par Adler et al. [2012] en

| Clipping | OMP | OMP-$S_{0.8}$ | OMP-$S_{0.6}$ | d-OMP | d-OMP-$S_{0.8}$ | d-OMP-$S_{0.6}$ |
|---|---|---|---|---|---|---|
| 0.4 | 8.3 | 8.2 | 8.2 | 14.5 | 13.5 | 14.0 |
| 0.6 | 13.2 | 12.7 | 12.5 | 18.2 | 18.2 | 18.0 |
| 0.8 | 18.5 | 18.1 | 17.4 | 23.9 | 23.9 | 23.1 |
| Temps (s) | 293 | 238 | 193 | 321 | 264 | 220 |

TABLE 2.1 – *Performance (SNR en dB) sur la tâche de declipping audio pour plusieurs algorithmes (colonnes) et plusieurs niveaux d'écrêtage (lignes) : OMP, declipping OMP (d-OMP) et leurs variantes avec sélections aléatoires (OMP-$S_*$, d-OMP-$S_*$). Dernière ligne : temps de traitement moyen d'un signal.*

utilisant le code fourni par les auteurs. Les conditions expérimentales ont été reproduites à l'identique et nous comparons les résultats obtenus par la version classique de l'algorithme OMP, la version d'OMP optimisée pour la tâche de declipping proposée par les auteurs (d-OMP) et leurs variantes avec sélections aléatoires. Nous utilisons pour ces dernières versions les paramétrages $\mu = \kappa = 0.8$ et $\mu = \kappa = 0.6$.

La mesure classique du rapport signal à bruit (SNR) obtenu par les différentes méthodes est reportée dans le Tableau 2.1 conjointement au temps de calcul mesuré. Ici, le temps de calcul présenté est le temps total de la procédure (pas seuleument la phase de sélection). On observe par exemple dans ce tableau que la version classique d-OMP met environ 20% de temps en plus que sa variante d-OMP-$S_{0.8}$ pour atteindre la même performance. En général, on observe une diminution très légère (voire nulle) des performances lors de l'utilisation des versions avec sélections aléatoires pour un temps de calcul bien moins important. Cette expérience nous en dit plus sur certaines propriétés intéressantes de la sélection aléatoire de sous-dictionnaires. La méthode proposée :

1. peut être appliquée dans des problématiques faisant intervenir des données réelles et *bruitées* ;
2. est utilisable avec toute sorte d'algorithme de poursuite (ici (d-)OMP) ;
3. permet un gain de temps notable même dans le cas d'un algorithme de poursuite dont la phase de mise à jour est plus coûteuse que la phase de sélection.

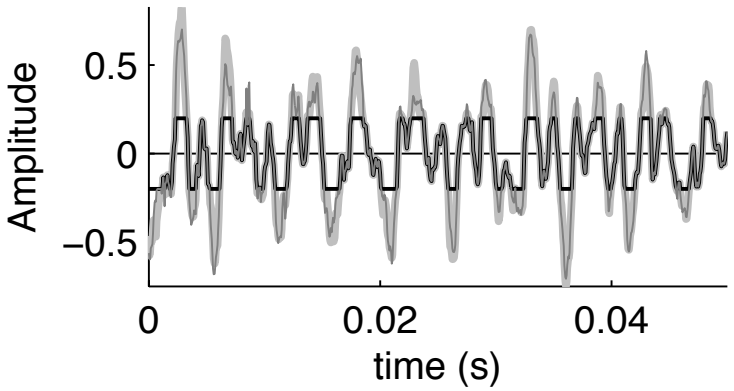



FIGURE 2.6 – *Exemple d'un signal audio dans sa version d'origine (gris large), écrêtée (noir) puis reconstruite (gris fin).*

## Conclusion du chapitre

Dans ce chapitre, nous avons proposé *Matching Pursuit avec Sélection Aléatoire (MP-SA)*, une stratégie d'échantillonnage aléatoire rapide et efficace permettant d'accélérer l'étape de sélection d'un atome dans l'algorithme Matching Pursuit. Notre procédure repose sur un double échantillonnage aléatoire du dictionnaire à chaque itération afin de sélectionner un atome (éventuellement sous-optimal). Elle conduit ainsi à l'utilisation d'un sous-dictionnaire renouvelé à chaque itération. L'ensemble des colonnes sélectionnées correspond à l'ensemble des atomes candidats à être ajoutés dans la représentation du signal considéré. Cela signifie qu'au lieu de passer en revue tous les atomes disponibles, nous restreignons la recherche à un petit sous-ensemble d'atomes sélectionnés aléatoirement d'une manière identique à Moussallam et al. [2012]. Nous étendons cette approche avec une caractéristique supplémentaire : au lieu de calculer les corrélations exactes entre les atomes sélectionnés et le résidu courant nous utilisons une estimation de ces corrélations basée sur un sous-échantillonnage aléatoire des coordonnées considérées dans l'opération.

Nous avons exposé des garanties théoriques de notre procédure. D'une part, en définissant une nouvelle propriété quantifiant les similarités entre les atomes d'un dictionnaire, nous avons pu mettre en avant la pertinence de la sélection aléatoire de sous-dictionnaire à chaque itération. Contrairement aux analyses classiques mettant en avant l'incohérence du dictionnaire, le régime de prédilection de notre procédure est justement celui de dictionnaires relativement cohérents. D'autre part, nous avons montré en utilisant notamment l'inégalité de concentration proposée par Serfling [1974] que cette sélection aléatoire de coordonnées est théoriquement fondée. Comme le montrent de nombreuses simulations réalisées, cette approche est efficace en pratique puisqu'elle permet de maintenir une qualité de représentation comparable à l'algorithme classique du Matching Pursuit tout en nécessitant un temps de calcul inférieur. Notre algorithme permet de plus une paramétrisation fine du compromis entre la complexité de calcul et la précision désirées. Nous avons enfin vu que le choix de cette paramétrisation peut être guidé notamment par les propriétés du dictionnaire et par un a priori sur le taille du support recherché.

Une remarque d'importance sur notre travail réside dans le vaste champ d'application de l'approche proposée ici. Bien que nous nous sommes focalisés sur l'algorithme de Matching Pursuit dans ce chapitre, il est clair que notre procédure s'applique à toute sorte d'algorithme de poursuite incluant une phase de sélection basée sur le calcul de corrélations. Notre approche soulève ensuite de nombreuses questions auxquelles nous espérons répondre dans un futur proche.

Premièrement, nous avons présenté dans une section de ce chapitre une analyse théorique de notre approche. Notre analyse s'attelle notamment à quantifier l'approximation réalisée à chaque étape par notre procédure. Celle-ci fait appel à la notion de p-$\gamma$-cohérence d'un dictionnaire et aussi à des inégalités de concentrations issues de travaux de Serfling [Serfling 1974].

Ces garanties ne reflètent malheureusement pas la qualité des résultats obtenus en pratique. Ainsi, une première piste pourrait être d'améliorer la précision des bornes présentées, notamment en utilisant des inégalités de concentration matricielles (par exemple les travaux de Drineas et al. [2006]). Nous espérons que l'étude des propriétés de notre procédure à un niveau matriciel permette de nous passer de l'utilisation d'une borne de l'union indispensable dans notre approche « atomique ». Ensuite, il serait utile de remplacer cette même technique de preuve au niveau des itérations et d'affiner l'analyse théorique de celles-ci, notamment les implications sur les itérations à venir du choix d'un atome. Enfin, une analyse plus spécifique de la capacité de notre procédure de sélection aléatoire à identifier le support réel d'un signal pourrait être intéressante.

Ensuite, il apparaît bien évidemment la question des connexions entre notre travail et les travaux utilisant des projections aléatoires dans le but de réduire la taille des objets intervenant dans les calculs. La sélection aléatoire de coordonnées des atomes du dictionnaire peut en effet être vue comme une projection aléatoire sur un espace de plus petite dimension. Vue ainsi, la stratégie de sélection aléatoire uniforme des coordonnées possède certainement des propriétés (orthogonalité de l'espace de projection par exemple) qui demandent à être éclaircies. Au-delà de ces simples propriétés, on est amené à considérer les travaux de Needell et Tropp [2010] sur l'algorithme CoSaMP, une version de Matching Pursuit basée sur une approche « compressed sensing ». Ici encore, la différence principale réside dans le fait que notre procédure renouvelle le sous-espace défini à partir du sous-échantillon des coordonnées à chaque itération.

Des travaux futurs pourraient résider dans l'élaboration de bornes plus strictes pouvant être utilisées au coeur de notre procédure pour guider le choix des sous-échantillons à analyser. A la manière des poursuites dynamiques évoquées dans la thèse de Moussallam [2012] ou des *screening* tests (articles de Xiang et al. [2011] et de Xiang et Ramadge [2012] par exemple), il pourrait s'avérer très profitable d'identifier très tôt au cours des itérations les atomes non pertinents afin de les exclure de la procédure d'échantillonnage. Pour terminer, l'utilisation de notre algorithme sur des taches de grande ampleur, par exemple en recherche d'information multimédia, apporterait encore plus de crédit à notre travail.

# 3 Algorithmes gloutons régularisés pour la classification multi-classe avec codes correcteurs : sélection de variables avec parcimonie structurée

Sommaire


3.1 Régression linéaire régularisée pour l'apprentissage multi-classes . . . 53
3.1.1 Apprentissage multi-classes et codes correcteurs d'erreurs . 53
3.1.2 Notations et formulation du problème . . . 54
3.2 Représentations parcimonieuses simultanées . . . 55
3.3 OMP multi-signaux régularisé et parcimonie groupée 56
3.3.1 Présentation du problème et travail relié . . . 56
3.3.2 Présentation et analyse de l'algorithme . . . 57
3.3.3 Implémentation efficace . . . 60
3.3.4 Remarques et extensions . . . 61
3.4 Applications . . . 62
3.4.1 Problèmes jouets . . . 63
3.4.2 20 Newsgroups . . . 63
3.5 Conclusion . . . 66


Nous nous sommes intéressés dans le chapitre précédent à la décomposition parcimonieuse d'un signal sur un dictionnaire redondant. Une extension naturelle de cette problématique et la décomposition de plusieurs signaux en utilisant le même dictionnaire. Celle-ci peut être effectuée de façon indépendante en lançant un algorithme de représentation pour chacun des signaux. Cependant, dans certaines situations que nous détaillons par la suite, il peut être intéressant de ne pas traiter les signaux indépendamment les uns des autres et de préférer les représenter avec les mêmes

atomes. Nous faisons référence à cette approche en utilisant les termes *représentation parcimonieuse simultanée*. Dans ce chapitre, nous allons voir comment l'application de cette technique populaire dans la communauté du traitement du signal permet de s'attaquer conjointement aux problématiques de la classification multi-classes et de la sélection de variables. Nous commençons ce chapitre en exposant le cadre dans lequel nous nous plaçons.

**Classification multi-classes** En classification multi-classe, on souhaite associer à un exemple une classe (catégorie) choisie parmi un ensemble discret de cardinalité supérieure à deux. On travaille avec un ensemble d'apprentissage $\underline{Z}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ où généralement $\boldsymbol{x}_i \in \mathbb{R}^M$ et $y_i \in \{1, \ldots, Q\}$. Ce problème apparaît par exemple dans le cadre de la reconnaissance de chiffres manuscrits (on considère alors les classes $0, \ldots, 9$) ou encore dans des tâches d'indexation multimédia (classification de photographies par thème ou encore identification du genre d'une chanson). Nous excluons de ce cadre les problématiques de classification multi-labels où plusieurs classes sont associées à une donnée (par exemple, la tâche d'identifier des objets dans une photographie est un problème multi-labels qui sort du cadre d'étude qui nous intéresse). La plupart des algorithmes destinés à la classification binaire ($\mathcal{Y} = \{-1, +1\}$ ou $\mathcal{Y} = \{0, 1\}$) sont extensibles au cadre de la classification multi-classes. Certains le sont naturellement (arbres de décisions, $k$ plus proches voisins), d'autres nécessitent quelques modifications afin de prendre en considération les propriétés spécifiques de cette tâche [*Multi-class SVM* de Weston et Watkins 1998]. Il existe de nombreux algorithmes (ou extension d'algorithmes binaires existants) *dédiés* à la classification multi-classes. Une autre manière classique de considérer un problème multi-classe afin d'utiliser directement les versions binaires des classifieurs usuels est de séparer la tâche en un ensemble de problèmes binaires. Plusieurs classifieurs sont ainsi appris (un par sous-problème binaire) et la classe d'un exemple est donnée en *combinant* les sorties de chacun. Les deux principales approches rencontrées dans cette méthode sont les approches *un-contre-tous* et *un-contre-un*. Une généralisation de cette pratique en représentant les classes par des « mots de code » a été proposée par Dietterich et Ghulum [1995]. Ce travail étend celui de Sejnowski et Rosenberg [1987] et établit les propriétés essentielles que doivent posséder les codes notamment en s'inspirant des codes correcteurs d'erreurs issus de la théorie de l'information. Ces codes sont dans un premier temps binaire jusqu'aux travaux de Allwein et al. [2000] qui proposent une extension de cette méthode *via* l'utilisation de codes ternaires. Nous reviendrons plus en détail sur les spécificités de la classification multi-classe à l'aide de codes correcteurs d'erreurs dans la suite de ce chapitre.

**Sélection de variables** La notion de sélection de variables prend tout son sens lorsque la tâche de classification ne vise pas uniquement l'obtention d'un bon classifieur (dans le sens d'un risque réel faible) mais considère aussi la simplicité du modèle utilisé par le classifieur. A travers celle-ci à améliorer l'interprétabilité du modèle, le temps de calcul nécessaire pour classer une nouvelle donnée ou comme nous l'avons vu précédemment, les propriétés de généralisation du classifieur. L'article de Guyon et Elisseeff

[2003] propose une introduction détaillée sur le sujet. Pour illustrer les buts recherchés par la sélection de variables, prenons le problème de la classification de textes. Nous souhaitons identifier la catégorie à laquelle appartient un article (sport, économie, science, etc) en analysant les mots qu'il contient. La manière habituelle de procéder est de représenter chaque article par un vecteur où chaque coefficient représente la fréquence d'apparition de chaque mot dans l'article (en appliquant une normalisation appropriée) et dont la longueur est la taille du dictionnaire utilisé. Une limitation évidente d'une telle représentation est qu'elle conduit à des vecteurs dont la taille est souvent de plusieurs milliers de mots (même après élagage des mots de liaison non informatifs) alors que ces vecteurs sont bien souvent extrêmement creux. Lorsque le nombre d'articles à classer est important, la tâche devient alors longue et fastidieuse. Pouvoir sélectionner un sous-ensemble de mots pertinents pour mener à bien celle-ci dans un temps acceptable revêt alors une importance capitale. Au-delà de cet aspect calculatoire, une telle approche permet ensuite d'obtenir une représentation interprétable de chaque article, sous forme d'une liste réduite de mots-clés. Nous avons déjà vu dans les chapitres précédents qu'un moyen d'obtenir des modèles de ce type était de contraindre le problème d'optimisation mis en jeu en y ajoutant un terme de régularisation.

A première vue, il n'est pas aisé de faire le lien entre les algorithmes régularisés présentés dans le chapitre précédent et les deux problématiques évoquées ci-dessus. Pourtant, nous présentons dans ce chapitre une procédure très simple à mettre en œuvre afin de transposer les algorithmes gloutons de représentation parcimonieuse au cadre de l'apprentissage multi-classes. Notre travail n'est pas le seul à combiner représentation parcimonieuse et codes correcteurs d'erreurs, par exemple Hsu et al. [2009] proposent de résoudre une tâche de prédiction multi-labels en utilisant conjointement du *compressed sensing* et des codes correcteurs.

La suite de ce chapitre se présente de la manière suivante. Dans la Section 3.1 nous présentons l'utilisation des codes correcteurs dans le cadre de l'apprentissage multi-classes et posons le problème de régression qui nous intéresse. La Section 3.2 dresse un rapide état de l'art des algorithmes de représentation parcimonieuse simultanée. Nous exposons dans la Section 3.3 notre approche, une version légèrement modifiée de l'algorithme OMP, pour résoudre un problème de classification multi-classe tout en sélectionnant des variables communes à toutes les classes. Nous présentons aussi dans cette section les propriétés de cette approche et la confrontons à d'autres méthodes « état de l'art ».

## 3.1 Régression linéaire régularisée pour l'apprentissage multi-classes

### 3.1.1 Apprentissage multi-classes et codes correcteurs d'erreurs

La classification multi-classe intervient lorsque les données $\boldsymbol{x}_i$ auxquelles nous nous intéressons peuvent appartenir à un nombre $Q > 2$ de classes : $y_i \in \{1, \ldots, Q\}$. Pour traiter ce type de problèmes, il existe deux grandes approches dans la communauté.

La première consiste à séparer le problème multi-classes en plusieurs sous-problèmes, chacun opposant uniquement deux classes ou bien deux ensembles de classes, et à apprendre un classifieur binaire usuel pour chaque sous-problème. Les deux stratégies les plus communes sont les stratégies

1. *un-contre-tous (OAA pour one-against-all)* : pour chaque classe, on apprend un classifieur permettant de distinguer cette classe des autres ;
2. *un-contre-un (OAO pour one-against-one)* : un classifieur est appris pour chaque paire de classes.

La résolution de chacun de ces problèmes fournit alors un ensemble de classifieurs ($Q$ pour l'approche $OAA$ et $Q(Q-1)/2$ pour l'approche $OAO$). Il ne reste ensuite plus qu'à combiner les règles de décision associées à ces classifieurs pour classer une nouvelle donnée. Dans le cas $OAA$, si l'on note $\mathbf{w}_q$ le vecteur définissant l'hyperplan séparant la classe $q$ des autres, une manière de procéder est d'associer à un nouvel exemple $\boldsymbol{x}$ la classe $\hat{q}$ telle que

$$\hat{q} = \underset{q \in \{1,\ldots,Q\}}{\arg\max} \langle \mathbf{w}_q, \boldsymbol{x} \rangle,$$

c'est à dire la classe pour laquelle la donnée $\boldsymbol{x}$ est du bon côté de l'hyperplan séparateur et le plus loin possible de celui-ci. Dans le cas $OAO$, on peut procéder à un vote de majorité. On associe alors à une donnée la classe la plus représentée parmi les sorties des $Q(Q-1)/2$ classifieurs.

La deuxième stratégie consiste à étendre un algorithme initialement destiné à la classification binaire. Cette extension est parfois triviale, comme par exemple dans le cas des arbres de décision [par exemple l'algorithme *C4.5*, Quinlan 1993] qui sont intrinsèquement des algorithmes de classification multi-classes puisque chaque feuille peut être associée à une classe différente. Elle peut aussi passer par la modification d'un problème d'optimisation associé initialement à une problématique binaire, on peut par exemple penser au travail de Weston et Watkins [1998] pour les SVM.

Nous laissons de côté cette dernière approche pour nous intéresser aux approches reposant sur la décomposition de la tâche en sous-problèmes. Nous nous focalisons plus particulièrement dans ce chapitre sur l'utilisation de codes correcteurs d'erreurs dans le cadre de l'apprentissage multi-classe. Cette utilisation est largement présentée dans les travaux de Dietterich et Ghulum [1995] et Allwein et al. [2000]. Les premiers proposent de représenter chaque classe $q \in \{1, \ldots, Q\}$ par un mot de code binaire $\boldsymbol{c}_q \in \{0, 1\}^L$. Les seconds étendent cette représentation en associant à chaque classe $q$ un mot de code ternaire $\boldsymbol{c}_q \in \{-1, 0, +1\}^L$. Chaque bit du mot de code correspond

à un problème binaire spécifique. De cette façon, un exemple appartenant à la classe $q$ sera considéré comme positif dans le problème $t$ si le $t^{eme}$ bit de son mot de code est $+1$. Il sera considéré comme négatif si ce bit vaut $-1$ et la valeur 0 indique de ne pas considérer l'exemple dans ce problème binaire. L'utilisation de mots de codes permet donc une partition plus fine des exemples que les stratégies *OAA* et *OAO* qu'ils généralisent. Ces mots de code sont souvent regroupés dans une matrice appelée *livre de codes*. Un exemple de livre de codes équivalent à l'approche un-contre-tous et un-contre-un pour un problème à trois classes est présenté dans la Figure 3.1. Ces livres de code peuvent être donnés a priori ou alors être construits en fonction du jeu de données considéré. Nous orientons le lecteur intéressé vers les travaux théoriques de Crammer et Singer [2002] sur la conception de livres de code.

| | $p_1$ | $p_2$ | $p_3$ |
|---|---|---|---|
| $c_1$ | 1 | $-1$ | $-1$ |
| $c_2$ | $-1$ | 1 | $-1$ |
| $c_3$ | $-1$ | $-1$ | 1 |

| | $p_1$ | $p_2$ | $p_3$ |
|---|---|---|---|
| $c_1$ | 1 | 1 | 0 |
| $c_2$ | $-1$ | 0 | 1 |
| $c_3$ | 0 | $-1$ | $-1$ |

FIGURE 3.1 – *Exemple d'un livre de code un-contre-tous (gauche) et un-contre-un (droite) pour un problème à trois classes. Chaque colonne de la matrice correspond à un problème de classification binaire.*

Dans l'exemple un-contre-tous ci-dessus, la première colonne de la matrice notée $p_1$ correspond au problème de trouver un classifieur séparant les données de la classe 1, considérées comme positives, de celles des classes 2 et 3, considérées comme négatives. La deuxième colonne correspond au problème de séparer la classe 2 des deux autres classes et ainsi de suite. On résout alors chaque problème en utilisant n'importe quel algorithme d'apprentissage binaire (SVM, Perceptron, régression linéaire, etc...) afin d'obtenir un ensemble de $L$ classifieurs. Pour un nouvel exemple $\boldsymbol{x}_i$, on calcule la sortie de chacun de ces $L$ classifieurs pour former le mot de code associé à cet exemple. On attribue alors à l'exemple la classe possédant le code le plus proche suivant une certaine distance (Hamming, $\ell_1$, $\ell_2$).

### 3.1.2 Notations et formulation du problème

Nous utilisons dans ce chapitre le formalisme suivant. Nous considérons un échantillon d'apprentissage $\underline{Z}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ où $\boldsymbol{x}_i \in \mathbb{R}^K$ et $y_i \in \{1, \ldots, Q\}$ ainsi qu'un livre de code $\mathbf{M} \in \mathbb{R}^{Q \times L}$ tel que chaque ligne $\boldsymbol{m}_q$ représente le code de la classe $q$,

$$\boldsymbol{m}_q \in \mathbb{R}^L,\ 1 \leq q \leq Q.$$

On forme ensuite la matrice $\mathbf{X} \in \mathbb{R}^{n \times K}$ dont chaque ligne représente un exemple et la matrice $\mathbf{C} \in \mathbb{R}^{n \times L}$ contenant les codes des classes correspondantes où chaque ligne $\boldsymbol{c}_i$ est telle que

$$\boldsymbol{c}_i = \boldsymbol{m}_{y_i}.$$

Dans la suite de ce chapitre, nous étendons la norme $\ell_1$ au cas matriciel :

$$\|\mathbf{A}\|_1 = \sum_{i,j} |\mathbf{A}_{i,j}|.$$

Nous nous intéressons au problème d'apprendre un ensemble de vecteurs de coefficients $\{\mathbf{w}^1, \ldots, \mathbf{w}^L\}$ tels que $\mathbf{X}\mathbf{w}^j \approx \boldsymbol{c}^j$. Avec les notations introduites ci-dessus, ce problème peut alors s'écrire sous la forme matricielle

$$\min_{\mathbf{W} \in \mathbb{R}^{K \times L}} \|\mathbf{X}\mathbf{W} - \mathbf{C}\|_F^2 + \Omega(\mathbf{W}). \tag{3.1}$$

$\Omega(\mathbf{W})$ est un terme de régularisation. A partir de cette formulation générale du problème, différentes fonctions de régularisation ont donné lieu aux travaux suivants :

- Lorsque $\Omega(\mathbf{W}) = \|\mathbf{W}\|_1$ on obtient la formulation *Lasso* [Tibshirani 1996].
- Lorsque $\Omega(\mathbf{W}) = \lambda_1 \|\mathbf{W}\|_1 + \lambda_2 \|\mathbf{W}\|_2$, on parle d'*Elastic Net* [Zou et Hastie 2005].
- Lorsque $\Omega(\mathbf{W}) = \|\mathbf{W}\|_2$ on est alors dans le cadre d'un problème de régression aux moindres carrés régularisée autrement appelée *Ridge Regression* [Hoerl et Kennard 1970].

Une fois obtenue la solution $\mathbf{W}^*$ à ce problème, on attribue à un nouvel exemple $\boldsymbol{x}$ la classe $\hat{q}$ calculée selon

$$\hat{q} = \arg\min_{q \in \{1,\ldots,Q\}} \|\boldsymbol{x}\mathbf{W}^* - \boldsymbol{m}_q\|_2^2 . \tag{3.2}$$

**Remarque 3.1** *Nous ne mentionnons dans ce chapitre que l'utilisation de la norme $\ell_2$ pour attribuer une classe à un nouvel exemple. Cette utilisation est motivée par la formulation « moindres carrés » dans (3.1) ayant justement pour but de minimiser l'écart quadratique moyen sur l'ensemble d'apprentissage entre les codes prédits et les codes réels associés aux exemples.*

## 3.2 Représentations parcimonieuses simultanées

Reprenons le formalisme du chapitre précédent avec maintenant un ensemble $\{\boldsymbol{y}^i\}_{i=1}^L$ de $L$ signaux dans $\mathbb{R}^M$. Nous souhaitons décomposer ces signaux sur un dictionnaire $\mathbf{X} \in \mathbb{R}^{M \times K}$ en utilisant un ensemble, toujours le plus petit possible, d'atomes communs à tous les signaux. Cette hypothèse d'utiliser des atomes communs prend tout son sens par exemple lorsque les signaux $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L$ sont des versions bruitées d'un signal de base $\boldsymbol{y}^*$. Dans ce cas de figure, le fait de coder les signaux simultanément permet de se focaliser sur le support du signal $\boldsymbol{y}^*$ en essayant de faire abstraction des artefacts dûs au bruit. On peut par exemple penser au problème de débruitage d'un électroencéphalogramme (EEG). En effet, de par leur processus d'acquisition (électrodes posées sur le cuir chevelu), les signaux EEG sont fortement bruités. Une manière de procéder pour nettoyer ces acquisitions est typiquement de chercher une représentation de celles-ci utilisant des atomes communs.

En notant $\mathbf{Y} = [\boldsymbol{y}^1 \cdots \boldsymbol{y}^L]$ la matrice dont chaque colonne est un signal, on cherche alors une matrice de coefficients $\mathbf{W}$, dont peu de lignes comportent une valeur non-nulle, telle que $\mathbf{X}\mathbf{W} \approx \mathbf{Y}$. Nous utilisons les notations $\mathbf{W}_{E,\cdot}$ (resp. $\mathbf{W}_{\cdot,E}$) pour désigner les lignes (resp. colonnes) de la matrice $\mathbf{W}$ indicées par l'ensemble $E$. Lorsque $E$ ne possède qu'un élément, nous notons $\mathbf{w}_i$ (resp. $\mathbf{w}^j$) la $i$-ème ligne (resp. $j$-ème colonne) de la matrice

$\mathbf{W}$ et $w_{i,j}$ l'élément se trouvant à l'intersection. Chaque colonne $\mathbf{w}^j$ est ainsi la représensation du signal $\boldsymbol{y}^j$ sur le dictionnaire $\mathbf{X}$ :

$$\mathbf{X}\mathbf{w}^j \approx \boldsymbol{y}^j.$$

On définit la pseudo-norme $\|\mathbf{W}\|_{\bullet,0}$ de la matrice $\mathbf{W}$ comme

$$\|\mathbf{W}\|_{\bullet,0} = card(\,\{i \in \{1,\ldots,K\} \,|\, \exists j \text{ tel que } w_{i,j} \neq 0\}\,), \tag{3.3}$$

cela correspond au nombre de lignes non nulles de la matrice. Avec ces notations, le problème évoqué ci-dessus peut s'écrire

$$\min_{\mathbf{W}\in\mathbb{R}^{K\times L}} \|\mathbf{X}\mathbf{W}-\mathbf{Y}\|_F^2 \text{ t.q. } \|\mathbf{W}\|_{\bullet,0} \leq T. \tag{3.4}$$

Le problème (3.4) est difficile à résoudre. L'utilisation de la pseudo-norme $\|\cdot\|_{\bullet,0}$ ne permet pas d'adopter une approche naïve séparant ce problème en $L$ sous-problèmes et utilisant un algorithme de poursuite séparément sur chacun des signaux. En effet, la contrainte que les signaux soient décomposés suivant des atomes communs impose un traitement joint des colonnes de $\mathbf{W}$. Cette fois-ci, les algorithmes de poursuite auxquels nous nous intéressons doivent choisir un atome en prenant en considération toutes les corrélations entre les atomes et les différents résidus. En notant $\mathbf{W}^t$ la matrice de coefficients à l'itération $t$ et $\mathbf{R}^t = \mathbf{X}\mathbf{W}^t - \mathbf{Y}$ la matrice des résidus, ces algorithmes choisissent un atome suivant la règle suivante :

$$\max_{\boldsymbol{x}\in\{\boldsymbol{x}^1,\ldots,\boldsymbol{x}^K\}} \left\|\boldsymbol{x}^\top \mathbf{R}^t\right\|_p \tag{3.5}$$

Intuitivement, de faibles valeurs de $p$ promeuvent la sélection d'un atome *généraliste*, c'est-à-dire un atome corrélé à de nombreux signaux, même si l'amplitude de ces corrélations est relativement faible. A l'inverse, des valeurs plus élevées favorisent la sélection d'un atome dont l'amplitude des corrélations est élevée, privilégiant ainsi un atome *spécifique* à un petit groupe de signaux. Dans le cas présent, nous cherchons à coder les signaux avec le même ensemble d'atomes et nous sommes donc intéressés par de faibles valeurs de $p$. On trouve dans la littérature plusieurs versions de cette étape de sélection portées par les travaux de Tropp et al. [2006] ($p = 1$), Cotter et al. [2005] ($p = 2$) et les versions faibles de Leviatan et Temlyakov [2006] ($p = 2$ et $p = \infty$). Enfin, on peut citer les travaux de Chen et Huo [2006] qui montrent que pour $p \geq 1$, OMP permet de retrouver la décomposition la plus creuse.

## 3.3 OMP multi-signaux régularisé et parcimonie groupée

### 3.3.1 Présentation du problème et travail relié

L'algorithme OMP (mono-signal) a déjà été utilisé pour faire de la sélection de variables [Tropp 2004, Zhang 2009]. Cette approche porte souvent le nom de *régression aux moindres carrés gloutonne*. Nous nous intéressons à une extension de celle-ci, la *régression aux moindres carrés régularisée*

*gloutonne* [par exemple Pahikkala et al. 2010] associée au problème d'optimisation

$$\min_{\mathbf{w}\in\mathbb{R}^K} \|\mathbf{X}\mathbf{w}-\boldsymbol{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2 \text{ t.q. } \|\mathbf{w}\|_0 \leq T. \tag{3.6}$$

Le terme de régularisation supplémentaire $\lambda\|\mathbf{w}\|_2$ renforce les contraintes appliquées à $\mathbf{w}$ et donc restreint davantage la complexité de la représentation que l'on s'autorise.

A partir des équations (3.1), (3.4) et (3.6) nous proposons de formuler le problème de la classification multi-classes avec codes correcteurs, sous la contrainte de ne représenter les données qu'avec un petit nombre fixé de caractéristiques, comme le problème d'optimisation suivant :

$$\min_{\mathbf{W}\in\mathbb{R}^{K\times L}} \frac{1}{2}\|\mathbf{X}\mathbf{W}-\mathbf{Y}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2 \text{ t.q. } \|\mathbf{W}\|_{\bullet,0} \leq T. \tag{3.7}$$

Cette formulation a déjà été étudiée par Naula et al. [2011] dans un cadre un peu différent de celui que nous traitons. Ils proposent de résoudre ce problème comme une poursuite multi-signaux dont l'atome sélectionné à chaque itération minimise la moyenne des erreurs *leave-one-out* [Stone 1974] de chaque $\mathbf{w}^j$. Ce critère de sélection est différent de ceux évoqués dans la section précédente, basés uniquement sur l'étude des corrélations entre les atomes et les résidus des différents signaux.

Nous proposons de résoudre le problème (3.7) grâce à une version modifiée de l'algorithme OMP multi-canaux en intégrant le paramètre de régularisation $\lambda$ dans l'étape de sélection qui reste basée sur le calcul de corrélations. Nous montrons que ce critère revient à sélectionner la direction de descente restreinte à une ligne de la matrice $\mathbf{W}$ possédant la plus grande pente. En cela, notre algorithme est un algorithme de descente de gradient par coordonnée proche de celui proposé par Friedman et al. [2010].

### 3.3.2 Présentation et analyse de l'algorithme

Dans cette section, nous présentons et analysons les propriétés de l'algorithme *Greedy Block Coordinate Descent for Regularized Least Squares (GBCD-RLS)* que nous développons pour résoudre le problème (3.7). Le pseudo-code de notre procédure est donné dans l'Algorithme 5. Dans un premier temps, nous nous focalisons sur l'étape de sélection puis nous étudions l'étape de mise à jour afin de montrer que l'algorithme ne sélectionne jamais deux fois la même variable. Dans ce qui suit, nous faisons l'hypothèse que les colonnes de la matrice $\mathbf{X}$ sont de norme $\ell_2$ unitaire ($\|\boldsymbol{x}^j\|_2 = 1$).

Le critère de sélection que nous utilisons revient à identifier la direction de plus grande pente du gradient lorsqu'on contraint $\mathbf{W}$ à ne varier que d'une seule ligne par itération. En effet, le gradient de la fonction

$$f(\mathbf{W}) := \frac{1}{2}\|\mathbf{X}\mathbf{W}-\mathbf{Y}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2,$$

pris en $\mathbf{W}^t$ est donné par

$$\nabla^t = \nabla_{\mathbf{W}} f(\mathbf{W}^t) = \mathbf{X}^\top\left(\mathbf{X}\mathbf{W}^t - \mathbf{Y}\right) + \lambda\mathbf{W}^t. \tag{3.8}$$

**Algorithme 5 :** Greedy Block Coordinate Descent for Regularized Least Squares (GBCD-RLS)

**Entrée :** $\mathbf{X} \in \mathbb{R}^{n\times K}, \mathbf{Y} \in \mathbb{R}^{n\times L}$ et $T, \lambda \geq 0$.
**Sortie :** $\mathbf{W}^t \in \mathbb{R}^{K\times L}$ t.q. $\|\mathbf{W}^t\|_{\bullet,0} \leq T$.
**Initialisation :** $\mathbf{W} \leftarrow \mathbf{0}$, $\mathbf{R}^0 \leftarrow \mathbf{Y}$, $\Lambda^0 \leftarrow \emptyset$ et $t \leftarrow 0$

**tant que** $t \leq T$ et $\mathbf{R}^t \neq \mathbf{0}$ **faire**
  Sélection :
    $\mathbf{S} \leftarrow \mathbf{X}^\top \mathbf{R}^t + \lambda \mathbf{W}^t$
    $j^* \leftarrow \arg\max_{j\in\{1,\dots,K\}} \|\mathbf{S}_j\|_2$
  Mise à jour :
    $\Lambda^{t+1} \leftarrow \Lambda^t \cup \{j^*\}$
    $\mathbf{G}^{t+1} \leftarrow (\mathbf{X}_{.,\Lambda^{t+1}})^\top \mathbf{X}_{.,\Lambda^{t+1}}$
    $\mathbf{W}^{t+1}_{\Lambda^{t+1},.} \leftarrow \left(\mathbf{G}^{t+1} + \lambda \mathbb{I}^{t+1}\right)^{-1} (\mathbf{X}_{.,\Lambda^{t+1}})^\top \mathbf{Y}$
    $\mathbf{R}^{t+1} \leftarrow \mathbf{X}_{.,\Lambda^{t+1}} \mathbf{W}^{t+1}_{\Lambda^{t+1},.} - \mathbf{Y}$
  $t \leftarrow t+1$
**fin tant que**

Parmi l'ensemble

$$\mathcal{D} = \{\mathbf{D} = \begin{pmatrix} 0 \\ \vdots \\ \boldsymbol{d} \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{K\times L} \mid \|\mathbf{D}\|_{\bullet,0} = 1 \text{ et } \|d\|_2 = 1\},$$

des directions de descente que nous considérons, la direction de descente de plus grande pente $\mathbf{D}^*$ est obtenue par la résolution du problème d'optimisation suivant

$$\mathbf{D}^* = \arg\max_{\mathbf{D}\in\mathcal{D}} \langle \mathbf{D}, \nabla^t \rangle_F \tag{3.9}$$

où

$$\langle \mathbf{U}, \mathbf{V} \rangle_F = \sum_{i,j} (\mathbf{U})_{i,j} (\mathbf{V})_{i,j}\,.$$

En notant $\nabla^t_{i,.}$ la $i^{ème}$ ligne du gradient, le problème précédent revient à trouver l'indice $j^*$ tel que

$$\begin{aligned} j^* &= \arg\max_{j\in\{1,\dots,K\}} \left( \max_{\boldsymbol{d}\in\mathbb{R}^L, \|\boldsymbol{d}\|_2=1} \langle \boldsymbol{d}, \nabla^t_{j,.} \rangle \right) \\ &= \arg\max_{j\in\{1,\dots,K\}} \left\| \nabla^t_{j,.} \right\|_2, \end{aligned} \tag{3.10}$$

qui peut s'écrire en utilisant l'Equation (3.8)

$$j^* = \arg\max_{j\in\{1,\dots,K\}} \left\| \mathbf{X}_j^\top (\mathbf{X}\mathbf{W}^t - \mathbf{Y}) + \lambda \mathbf{W}^t_j \right\|_2. \tag{3.11}$$

Une fois cette coordonnée sélectionnée, elle est ajoutée à l'ensemble des coordonnées déjà choisies : $\Lambda^{t+1} = \Lambda^t \cup \{j^*\}$. La mise à jour est alors le

résultat du problème d'optimisation réduit

$$\mathbf{W}^{t+1}_{\Lambda^{t+1},.} = \underset{\mathbf{U}\in\mathbb{R}^{t+1\times L}}{\arg\min}\ \frac{1}{2}\left\|\mathbf{X}_{.,\Lambda^{t+1}}\mathbf{U}-\mathbf{Y}\right\|_2^2 + \frac{1}{2}\lambda\left\|\mathbf{U}\right\|_2^2, \tag{3.12}$$

où $\mathbf{W}^{t+1}_{\Lambda^{t+1},.}$ désigne l'ensemble des lignes de $\mathbf{W}^{t+1}$ dont les indices sont contenus dans l'ensemble $\Lambda^{t+1}$. La solution $\mathbf{W}^{t+1}_{\Lambda^{t+1},.}$ de ce problème de régression régularisée est bien évidemment donnée par

$$\mathbf{W}^{t+1}_{\Lambda^{t+1},.} = \left(\mathbf{G}^{t+1}+\lambda\mathbb{I}^{t+1}\right)^{-1}(\mathbf{X}_{.,\Lambda^{t+1}})^\top\mathbf{Y}, \tag{3.13}$$

où nous avons noté $\mathbf{G}^{t+1} = (\mathbf{X}_{.,\Lambda^{t+1}})^\top\mathbf{X}_{.,\Lambda^{t+1}}$ la matrice de Gram du dictionnaire $\mathbf{X}$ réduit aux colonnes dont les indices sont dans $\Lambda^{t+1}$ et où $\mathbb{I}^{t+1}$ est la matrice identité de taille $(t+1)\times(t+1)$. Nous montrons maintenant qu'une colonne de la matrice $\mathbf{X}$ (une variable) ne peut pas être sélectionnée deux fois par notre algorithme.

**Proposition 1.** *Supposons qu'à l'instant $t$, $j^*$ soit la solution de (3.11) c'est-à-dire que*

$$j^* = \underset{j\in\{1,\dots,K\}}{\arg\max}\ \left\|\mathbf{X}_j^\top(\mathbf{X}\mathbf{W}^t-\mathbf{Y})+\lambda\mathbf{W}^t_j\right\|_2. \tag{3.14}$$

*Si $j^*\in\Lambda^t$ alors :*

*(a)* $\left\|\mathbf{X}_{j^*}^\top(\mathbf{X}\mathbf{W}^t-\mathbf{Y})+\lambda\mathbf{W}^t_{j^*}\right\|_2 = 0$ *;*

*(b)* $\mathbf{W}^t$ *est un minimum global du problème d'optimisation (3.6) avec* $\left\|\mathbf{W}^t\right\|_{\bullet,0} = t$.

*Démonstration.* Notons que $(b)$ est une conséquence de $(a)$. En effet, si

$$\left\|\mathbf{X}_{j^*}^\top(\mathbf{X}\mathbf{W}^t-\mathbf{Y})+\lambda\mathbf{W}^t_{j^*}\right\|_2 = 0,$$

alors, par définition de $j^*$ on a

$$\forall j, \left\|\mathbf{X}_j^\top(\mathbf{X}\mathbf{W}^t-\mathbf{Y})+\lambda\mathbf{W}^t_j\right\|_2 = 0,$$

c'est-à-dire

$$\nabla^t = \nabla_{\mathbf{W}} f(\mathbf{W}^t) = 0,$$

qui est une condition suffisante (et nécessaire) pour que $\mathbf{W}^t$ soit une solution du problème d'optimisation (3.6) dont la fonctionnelle est fortement convexe. Il nous reste maintenant à montrer $(a)$. Nous observons que

$$\begin{aligned}\mathbf{X}_{j^*}^\top(\mathbf{X}\mathbf{W}^t-\mathbf{Y})+\lambda\mathbf{W}^t_{j^*} &= \mathbf{X}_{j^*}^\top(\mathbf{X}_{.,\Lambda^t}\mathbf{W}^t_{\Lambda^t,.}-\mathbf{Y})+\lambda\mathbf{W}^t_{j^*}\\ &= \mathbf{X}_{j^*}^\top\mathbf{X}_{.,\Lambda^t}\mathbf{W}^t_{\Lambda^t,.}+\lambda\mathbf{W}^t_{j^*}-\mathbf{X}_{j^*}^\top\mathbf{Y},\end{aligned}$$

la première ligne venant du fait que les lignes de $\mathbf{W}^t$ dont les indices ne sont pas dans $\Lambda^t$ sont nulles. Il apparaît ensuite que $\mathbf{X}_{j^*}^\top\mathbf{X}_{.,\Lambda^t}\mathbf{W}^t_{\Lambda^t,.}+\lambda\mathbf{W}^t_{j^*}$ et $\mathbf{X}_{j^*}^\top\mathbf{Y}$ sont respectivement les $j^{*\text{-ème}}$ entrées de $(\mathbf{G}^t+\lambda\mathbb{I}^t)\mathbf{W}^t_{\Lambda^t,.}$ et $\mathbf{X}_{.,\Lambda^t}^\top\mathbf{Y}$. On a donc par définition de $\mathbf{W}^t_{\Lambda^t,.}$ (voir Equation (3.13))

$$\begin{aligned}\mathbf{X}_{j^*}^\top(\mathbf{X}\mathbf{W}^t-\mathbf{Y})+\lambda\mathbf{W}^t_{j^*} &= \left[(\mathbf{G}^t+\lambda\mathbb{I}^t)\mathbf{W}^t_{\Lambda^t,.}\right]_{j^*} - \left[\mathbf{X}_{.,\Lambda^t}^\top\mathbf{Y}\right]_{j^*}\\ &= \left[(\mathbf{G}^t+\lambda\mathbb{I}^t)(\mathbf{G}^t+\lambda\mathbb{I}^t)^{-1}\mathbf{X}_{.,\Lambda^t}^\top\mathbf{Y}\right]_{j^*} - \left[\mathbf{X}_{.,\Lambda^t}^\top\mathbf{Y}\right]_{j^*}\\ &= \left[\mathbf{X}_{.,\Lambda^t}^\top\mathbf{Y}\right]_{j^*} - \left[\mathbf{X}_{.,\Lambda^t}^\top\mathbf{Y}\right]_{j^*}\\ &= 0,\end{aligned}$$

ce qui vient terminer la preuve. □

**Algorithme 6 :** GBCD-RLS-Cholesky

**Entrée :** $\mathbf{X} \in \mathbb{R}^{n \times K}, \mathbf{Y} \in \mathbb{R}^{n \times L}$ et $T, \lambda \geq 0$.
**Sortie :** $\mathbf{W}^t \in \mathbb{R}^{K \times L}$ t.q. $\|\mathbf{W}^t\|_{\bullet,0} \leq T$.
**Initialisation :**
$\mathbf{W}^0 \leftarrow \mathbf{0}$, $\mathbf{R}^0 \leftarrow \mathbf{C}$, $\Lambda^0 \leftarrow \emptyset$, $\mathbf{A} \leftarrow \mathbf{X}^\top \mathbf{Y}$, $\mathbf{L} = \sqrt{1+\lambda}$ et $t \leftarrow 0$

**tant que** $t \leq T$ et $\mathbf{R}^t \neq \mathbf{0}$ **faire**
Sélection :
$\mathbf{S} \leftarrow \mathbf{X}^\top \mathbf{R}^t + \lambda \mathbf{W}^t$
$j^* \leftarrow \arg\max j \in \{1, \ldots, K\} \|\mathbf{S}_j\|_2$
Mise à jour :
**si** $t > 1$ **alors**
$\boldsymbol{u} :=$ Résoudre pour $\boldsymbol{v}$ $\left\{\mathbf{L}\boldsymbol{v} = \mathbf{X}^\top_{\cdot,\Lambda^t} \boldsymbol{x}^{j^*}\right\}$
$\mathbf{L} \leftarrow \begin{pmatrix} \tilde{\mathbf{L}} & \mathbf{0} \\ \boldsymbol{u}^\top & \sqrt{1+\lambda-\boldsymbol{u}^\top \boldsymbol{u}} \end{pmatrix}$
**fin si**
$\Lambda^{t+1} \leftarrow \Lambda^t \cup \{j^*\}$
$\mathbf{W}^{t+1}_{\Lambda^{t+1},\cdot} :=$ Résoudre pour $\mathbf{B}$ $\left\{\mathbf{L}\mathbf{L}^\top \mathbf{B} = \mathbf{A}_{\Lambda^{t+1}}\right\}$
$\mathbf{R}^{t+1} \leftarrow \mathbf{X}_{\cdot,\Lambda^{t+1}} \mathbf{W}^{t+1}_{\Lambda^{t+1},\cdot} - \mathbf{Y}$
$t \leftarrow t+1$
**fin tant que**

### 3.3.3 Implémentation efficace

Une implémentation naïve de l'étape d'inversion de l'Algorithme 5, $\mathbf{W}^{t+1}_{\Lambda^{t+1},\cdot} \leftarrow \left(\mathbf{G}^{t+1} + \lambda \mathbb{I}^{t+1}\right)^{-1} \mathbf{X}^\top_{\cdot,\Lambda^{t+1}} \mathbf{Y}$, coûte $\mathcal{O}(t^3)$ opération à chaque itération ce qui représente un coût global de $\mathcal{O}(T^4)$ pour la procédure totale. L'utilisation d'une décomposition incrémentale de Cholesky comme celle présentée dans le travail de Rubinstein et al. [2008] permet de contenir la complexité de la procédure en $\mathcal{O}(T^3)$ tout en améliorant la stabilité numérique de la procédure. La subtilité repose sur le fait que la matrice semi-définie positive $\mathbf{G}^{t+1} + \lambda \mathbb{I}^{t+1}$ est mise à jour à chaque itération en ajoutant seulement une ligne et une colonne. Sa factorisation de Cholesky peut alors être obtenue à partir de celle de $\mathbf{G}^t + \lambda \mathbb{I}^t$ et ne requiert que la résolution d'un système linéaire de taille $t$. En effet, il est établi qu'à partir de la factorisation de Cholesky d'une matrice $\tilde{\mathbf{A}} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top \in \mathbb{R}^{n-1 \times n-1}$, la factorisation de la matrice

$$\mathbf{A} = \begin{pmatrix} \tilde{\mathbf{A}} & \boldsymbol{v} \\ \boldsymbol{v}^\top & c \end{pmatrix} \in \mathbb{R}^{n \times n}$$

est donnée par $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$, avec

$$\mathbf{L} = \begin{pmatrix} \tilde{\mathbf{L}} & \mathbf{0} \\ \boldsymbol{u}^\top & \sqrt{c - \boldsymbol{u}^\top \boldsymbol{u}} \end{pmatrix}, \boldsymbol{u} = \tilde{\mathbf{L}}^{-1} \boldsymbol{v}.$$

L'Algorithme 6 détaille une implémentation de la procédure présentée dans la section précédente en utilisant la propriété ci-dessus.

### 3.3.4 Remarques et extensions

Dans les algorithmes présentés, nous faisons l'hypothèse que les colonnes de la matrice $\mathbf{X}$ sont de norme $\ell_2$ unitaire. Cela est rarement le cas en pratique. Pour pallier ce problème, il est donc nécessaire d'ajouter une étape de normalisation au préalable. Cependant, cette normalisation dépend fortement de l'échantillon d'apprentissage considéré, ce qui va à l'encontre de notre but qui est de pouvoir prédire les étiquettes de nouveaux exemples pas encore observés. Il convient donc d'ajouter également une « dé-normalisation » de notre vecteur de coefficients à la fin de l'algorithme. Nous résolvons donc le problème avec une version normalisée de la matrice $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{N}$ de notre matrice d'exemples $\mathbf{X}$ où $\mathbf{N}$ est la matrice

$$\mathbf{N} = \begin{pmatrix} \frac{1}{\|\boldsymbol{x}^1\|_2^2} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \frac{1}{\|\boldsymbol{x}^K\|_2^2} \end{pmatrix}.$$

Nous obtenons de cette façon la matrice de coefficients $\tilde{\mathbf{W}}$ telle que $\tilde{\mathbf{X}}\tilde{\mathbf{W}} \approx \mathbf{Y}$. Par identification, nous pouvons voir que la matrice de coefficients que nous recherchons est donc $\mathbf{W} = \mathbf{N}\tilde{\mathbf{W}}$.

Au-délà de cette remarque, nous souhaitons discuter de deux extensions de notre approche. La première fait écho aux travaux de Yuan et Lin [2006] (*Group Lasso*) sur la sélection groupée de variables. Considérons la partition $\mathcal{G} = \{g_1, \ldots, g_p\}$ des lignes de $\mathbf{W}$ où chaque $g_i$ est un sous-ensemble d'indices pris parmi $\{1, \ldots, K\}$. Le problème de sélectionner un petit nombre de groupes peut alors s'écrire comme

$$\min_{\mathbf{W} \in \mathbb{R}^{K \times L}} \frac{1}{2} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 + \frac{\lambda}{2} \|\mathbf{W}\|_F^2 \text{ t.q. } \|\mathbf{W}\|_{\mathcal{G},0} \leq T, \tag{3.15}$$

où

$$\|\mathbf{W}\|_{\mathcal{G},0} = card(\{g \in \mathcal{G} \,|\, \exists\, k \in g, \exists j \in \{1, \ldots, L\} \text{ tel que } w_{k,j} \neq 0\}). \tag{3.16}$$

Ce problème peut alors être traité par l'algorithme que nous avons présenté moyennant une petite modification au niveau de l'étape de sélection qui revient alors à sélectionner le groupe $g^*$ tel que

$$g^* = \arg\max_{g \in \mathcal{G}} \left\| \mathbf{X}_{g,.}^\top (\mathbf{X}\mathbf{W}^t - \mathbf{Y}) + \lambda \mathbf{W}_{g,.}^t \right\|_2. \tag{3.17}$$

Les coordonnées indexées par les éléments de $g^*$ sont ensuite ajoutées aux coordonnées déjà choisies, $\Lambda^{t+1} = \Lambda^t \cup g^*$, et l'étape de mise à jour demeure inchangée.

La deuxième extension que nous souhaitons aborder est celle de sélectionner des variables communes à des sous-ensembles de signaux avec un budget commun à ces sous-ensembles. Pour motiver cette approche, reprenons l'exemple des signaux EEG. Imaginons que nous souhaitons obtenir une représentation parcimonieuse d'un certain nombre de signaux issus de plusieurs tâche (reconnaître une image familière, reconnaître un son familier, etc) avec un ensemble commun d'atomes. Chaque tâche donne lieu à plusieurs acquisitions et ainsi nous nous retrouvons avec un ensemble de signaux

que nous pouvons regrouper par tâche. On peut aussi penser à une unique tâche effectuée par plusieurs patients pour laquelle on peut alors regrouper les signaux par patient. Dans les deux cas, il peut être intéressant de ne rechercher les variables pertinentes qu'à travers ces sous-ensemble de signaux. Ici encore, on peut traiter ce cas de figure avec une légère modification de notre algorithme. Considérons la partition $\mathcal{G} = \{g_1, \ldots, g_p\}$ des colonnes de $\mathbf{W}$ où chaque $g_i$ est un sous-ensemble d'indices pris parmi $\{1, \ldots, L\}$. On cherche alors à chaque itération la variable $j^*$ et le groupe $g^*$ tels que

$$(j^*, g^*) = \underset{j \in \{1,\ldots,K\},\, g \in \mathcal{G}}{\arg\max} \left\| \left[ \mathbf{X}_{j,\cdot}^\top (\mathbf{X}\mathbf{W}^t - \mathbf{Y}) + \lambda \mathbf{W}_{j,\cdot}^t \right]_g \right\|_2 , \tag{3.18}$$

où

$$\left[ \mathbf{X}_{j,\cdot}^\top (\mathbf{X}\mathbf{W}^t - \mathbf{Y}) + \lambda \mathbf{W}_{j,\cdot}^t \right]_g \tag{3.19}$$

désigne la $j^{ème}$ ligne du gradient restreinte au groupe (tâche ou patient) $g$. On met alors à jour l'ensemble des variables sélectionnées pour ce groupe par $\Lambda_{g^*}^{t+1} = \Lambda_{g^*}^t \cup \{j^*\}$ et les entrées de $\mathbf{W}$ correpondantes. On répète cette procédure tant que $card(\Lambda_{g_1}) + \ldots + card(\Lambda_{g_p}) \leq T$ où $T$ est le budget commun défini au préalable.

## 3.4 Applications

Avant de commencer, nous introduisons quelques mesures d'erreur communes en recherche d'information. Pour un classifieur $h$, le rappel pour la classe $i$ mesure le nombre d'exemples de cette classe ayant été correctement identifiés parmi tous les exemples de la classe $i$ d'un échantillon $\underline{Z}_n$

$$rappel_i(h) = \frac{card(\{(\boldsymbol{x}, y) \in \underline{Z}_n \,|\, h(\boldsymbol{x}) = i \text{ et } y = i\})}{card(\{(\boldsymbol{x}, y) \in \underline{Z}_n \,|\, y = i\})} . \tag{3.20}$$

Le rappel du classifieur $h$ est simplement la moyenne des rappels de ce classifieurs sur l'ensemble des classes

$$rappel(h) = \frac{1}{Q} \sum_{i=1}^{Q} rappel_i(h). \tag{3.21}$$

La précision de $h$ pour la classe $i$ mesure le nombre de documents appartenant réellement à cette classe parmi tous les documents ayant été prédits comme appartenant à celle-ci

$$precision_i(h) = \frac{card(\{(\boldsymbol{x}, y) \in \underline{Z}_n \,|\, h(\boldsymbol{x}) = i \text{ et } y = i\})}{card(\{(\boldsymbol{x}, y) \in \underline{Z}_n \,|\, h(\boldsymbol{x}) = i\})} . \tag{3.22}$$

De manière identique au rappel, la précision du classifieur $h$ est la moyenne des précisions sur chaque classe

$$precision(h) = \frac{1}{Q} \sum_{i=1}^{Q} precision_i(h). \tag{3.23}$$

Enfin, la dernière mesure d'erreur que nous souhaitons présenter est la $f_1$-mesure qui combine les deux mesures ci-dessus de la façon suivante

$$f_1\text{-mesure} = 2 \cdot \frac{\text{précision} \cdot \text{rappel}}{\text{précision} + \text{rappel}}. \tag{3.24}$$

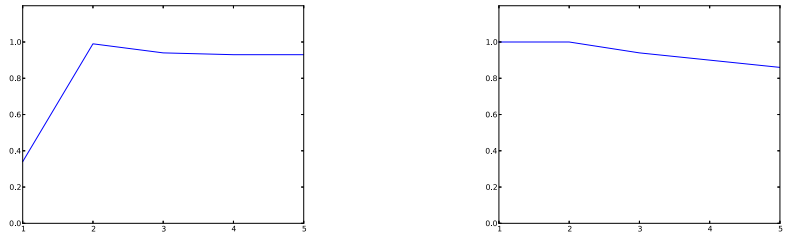


FIGURE 3.2 – *Gauche : $f_1$-mesure en fonction du nombre de variables sélectionnées ; Droite : ratio moyen de la somme des coefficients associés aux deux premières variables sur la somme totale des coefficients.*

### 3.4.1 Problèmes jouets

Nous commençons cette section par un jeu de données jouet afin de mettre en avant les capacités de notre algorithme à identifier des variables clés. Nous générons des données dans $\mathbb{R}^{1024}$ pour lesquelles seules les 2 premières dimensions sont pertinentes. Nous assignons à un exemple $\boldsymbol{x}$ la classe donnée par le Tableau 3.1. Le livre de codes que nous utilisons est le livre de

| | $x_1 \geq 0$ | $x_1 < 0$ |
|---|---|---|
| $x_2 \geq 0$ | 0 | 1 |
| $x_2 < 0$ | 2 | 3 |

TABLE 3.1 – *Tableau d'assignation des classes.*

*un-contre-tous* présenté dans la première section de ce chapitre. Nous reportons dans la Figure 3.2 les résultats obtenus par notre algorithme en apprenant sur 1000 données et en testant sur 1000 autres données n'ayant pas été utilisées pour l'apprentissage. Pour cette expérience, la valeur du paramètre $\lambda$ ne semblant pas déterminante elle a été arbitrairement fixée à 0.1. On peut voir dans la courbe de gauche de la Figure 3.2 que les performances du classifieur appris augmentent jusqu'à avoir sélectionné deux variables puis diminuent au fur et à mesure que l'on ajoute d'autres variables. La courbe de droite représente la contribution des coefficients associés aux deux premières coordonnées par rapport à la somme des coefficients de $\mathbf{W}$. On voit clairement que les deux coordonnées pertinentes sont sélectionnées aux deux premières itérations de l'algorithme. Ensuite les nouvelles variables ajoutées viennent bruiter la solution optimale obtenue en ne sélectionnant que ces deux premières coordonnées. Ces résultats montrent que notre algorithme glouton sélectionne les bonnes variables dans le cas où les données ne sont pas bruitées. Nous allons voir dans l'expérience suivante que pour des données réelles, la paramètre $\lambda$ jouen parfaitement son rôle de régularisation de la solution.

### 3.4.2 20 Newsgroups

Nous reportons les expériences menées sur le jeux de données *20 Newsgroups* [Lang 1995]. Ce jeu de données contient environ 20000 documents issus de newsgroups répartis à peu près équitablement entre 20 catégories. Nous cherchons à savoir i) si notre approche est valide pour résoudre ce problème de classification en identifiant un sous-ensemble de mots com-

muns à plusieurs classes permettant de les discriminer et ii) comment elle se comporte vis-à-vis d'algorithmes de référence en classification multi-classes. Nous comparons une implémentation de notre algorithme dans le langage Python à d'autres approches implémentées dans le module Scikit-Learn [Pedregosa et al. 2011]. Nous utilisons pour la comparaison le protocole établi dans la documentation du module dont les détails peuvent être trouvés ici : `http://scikit-learn.org/stable/auto_examples/document_classification_20newsgroups.html`. Les données sont représentées sous forme d'une matrice TF-IDF (de l'anglais Term Frequency - Inverse Document Frequency). Pour un ensemble de documents $\mathcal{C} = \{d_1, \ldots, d_n\}$ appelé *corpus* et un ensemble de termes $\{t_1, \ldots, t_k\}$, $\mathrm{TF}_{i,j}$ mesure le nombre d'occurrences du terme $t_i$ dans le document $d_j$

$$\mathrm{TF}_{i,j} = card(\{t \in d_j \,|\, t = t_i\}). \tag{3.25}$$

$\mathrm{IDF}_i$ est une mesure de l'importance du terme $t_i$ dans l'ensemble du corpus

$$\mathrm{IDF}_i = \log\left(\frac{n}{card(\{d \in \mathcal{C} \,|\, t_i \in d\})}\right). \tag{3.26}$$

La matrice TF-IDF est alors définie comme

$$\text{TF-IDF}_{i,j} = \mathrm{TF}_{i,j} \cdot \mathrm{IDF}_i. \tag{3.27}$$

Le schéma TF-IDF vise ainsi à donner un poids plus important aux termes les moins fréquents dans le corpus. Une partie des exemples est laissée de côté au cours du processus d'apprentissage et permet d'évaluer les performances des différents algorithmes mis en jeu. Dans le reste de cette section, les codes utilisés sont ceux de l'approche un-contre-tous.

Nous reportons dans la Figure 3.3 les résultats obtenus par notre algorithme sur l'échantillon d'apprentissage et sur l'échantillon de test pour plusieurs valeurs de $\lambda$. On voit dans un premier temps que plus les valeurs de $\lambda$ augmentent et plus les performances sur l'échantillon d'apprentissage se détériorent. Au contraire, sur l'échantillon de test, les performances augmentent avec $\lambda$ avant de diminuer ensuite quand les valeurs du terme de régularisation deviennent trop importantes. On peut donc noter que l'ajout du second terme de régularisation sur la matrice de coefficients permet d'améliorer les performances de classification.

Nous reportons ensuite dans le Tableau 3.2 les performances de notre algorithme avec les performances d'algorithmes état de l'art dans le domaine de la classification multi-classe. Ces algorithmes utilisent également l'approche un-contre-tous. Nous reportons les performances, le temps de calcul nécessaire à l'apprentissage et aussi le nombre de variables pertinentes sélectionnées. Nous commençons par remarquer que le temps de calcul n'est pas à l'avantage de notre approche qui est à peine plus rapide que l'approche moindre-carrés régularisée classique tandis que les autres algorithmes testés sont bien plus rapides. Il faut cependant nuancer ce constat par le fait que notre implémentation en Python n'est pas optimale, une grande partie du temps de calcul provenant notamment des accès mémoires. Une comparaison des algorithmes en terme de nombre d'opérations effectuées serait sans

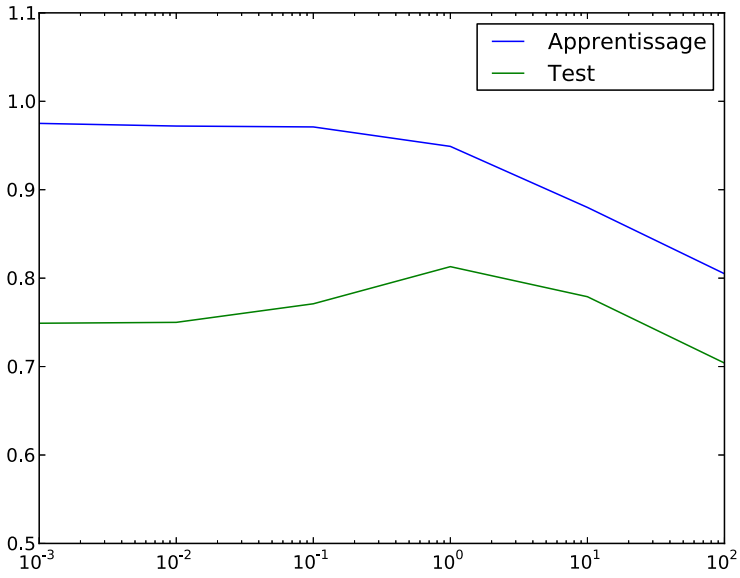



FIGURE 3.3 – *Evolution de la $f_1$-mesure sur l'échantillon d'apprentissage et sur l'échantillon de test en fonction de $\lambda$.*

| Algorithme | $f_1$-mesure | Temps (s) | # Variables |
|---|---|---|---|
| Ridge Regression | $\mathbf{0,858}$ | $386,8$ | 55946 |
| GBCD-RLS | $0,813$ | $381,4$ | **2000** |
| Perceptron | $0,803$ | $5,6$ | 42674 |
| Passive-Agressive | $0,850$ | 6.2 | 55707 |
| SVM-$\ell_1$ | 0.828 | $7,6$ | 4457 |
| SVM-$\ell_2$ | $0,856$ | $9,53$ | 56122 |
| SGDC-$\ell_1$ | 0.793 | $13,1$ | 2437 |
| SGDC-$\ell_2$ | 0.854 | $\mathbf{5,0}$ | 55926 |
| SGDC-Elastic Net | 0.802 | $13,7$ | 3367 |

TABLE 3.2 – *Résultats obtenus pour plusieurs algorithmes.*

doute plus informative. Concernant la sélection de variables, nous avons limité l'algorithme *GBCD-RLS* à n'utiliser que 2000 d'entre elles. Ainsi, notre approche permet d'obtenir le classifieur le plus parcimonieux du classement. Les algorithmes incluant une pénalisation $\ell_1$ sont très proches en terme de parcimonie alors que les procédures à base de régularisation $\ell_2$ produisent des classifieurs considérant quasiment toutes les variables disponibles. Cependant, en terme de performances de classification que nous mesurons grâce à la $f_1$-mesure, ces dernières possèdent les meilleurs résultats. On peut toutefois noter que notre approche se comporte plutôt bien dans une tâche qui ne semble pas favoriser la sélection de variables. Notre algorithme réalise des performances souvent meilleures que les approches $\ell_1$ tout en sélectionnant moins de variables. Par exemple, la comparaison avec l'approche basée sur une descente de gradient stochastique d'un problème avec une régularisation elastic-net (SGDC-Elastic Net) montre qu'avec moins de variables (2000 contre 3367) nous réussissons à obtenir de meilleurs résultats en $f_1$-

mesure (0,813 contre 0,802). Un constat plus général concerne le fait qu'il semble difficile de trouver des mots discriminants pour *toutes* les classes considérées. Une version alternative de notre algorithme cherchant des mots communs à des sous-ensembles de classes, comme évoquée dans la section précédente, obtiendrait peut-être de meilleurs résultats.

## 3.5 Conclusion

Dans ce chapitre nous avons présenté un nouvel algorithme glouton pour résoudre le problème de la régression moindre carrés régularisée et donné une implémentation efficace de celui-ci. Cet algorithme est exprimé dans un cadre général, celui de la représentation parcimonieuse simultanée de signaux. Nous avons ensuite montré comment appliquer ce dernier pour résoudre un problème de classification multi-classes *via* l'utilisation de codes correcteurs d'erreurs.

Notre algorithme est très proche d'algorithmes déjà existants, notamment des travaux de Naula et al. [2011]. La différence majeure réside dans le fait que notre procédure utilise à chaque itération la direction de plus forte pente, restreinte à une ligne, du gradient de la fonction objectif. Ce choix nous permet de démontrer que notre algorithme ne peut sélectionner deux fois la même variable, à moins d'avoir atteint le vecteur de coefficients optimal. Nous avons enfin proposé une implémentation efficace de ce dernier, permettant de gagner un ordre de grandeur sur la complexité de calcul par rapport à une implémentation naïve.

Grâce à l'utilisation de codes correcteurs d'erreurs, nous avons montré sur un problème jouet non bruité de classification multi-classes que notre approche est pertinente pour identifier des variables singulières. Sur des données réelles de grande dimension, nous avons constaté que ses performances sont proches de l'état de l'art.

Les perspectives de ce travail sont nombreuses. La première a déjà été mentionnée dans ce chapitre et concerne la sélection d'atomes communs à des sous-ensembles de signaux. Cela permettrait alors de s'attaquer à des problématiques pour lesquelles l'*a priori* d'atomes communs à tous les signaux considérés n'est pas vérifiée. Ensuite, nous avons vu qu'une limitation de l'application de notre algorithme à des problèmes multi-classes de grande taille est le temps de calcul nécessaire par rapport aux approches de référence dans ce domaine. L'utilisation de sous-dictionnaires aléatoires, comme proposée au chapitre précédent, semble être une piste intéressante à explorer pour diminuer ce temps de calcul. Enfin, travailler avec d'autres types de codes et / ou d'autres distances (Hamming, $\ell_1$) au moment d'associer une classe à un code estimé amènerait sans doute à considérer de nouvelles stratégies de sélection afin de prendre en compte ces spécificités.

## Deuxième partie

# Inégalités de Bernstein empiriques

# 4 Inégalités de Bernstein empiriques pour les U-statistiques

Sommaire


4.1 Contexte . . . 73
4.1.1 Quelques notations . . . 73
4.1.2 U-statistique . . . 73
4.1.3 Inégalités de concentration pour les U-statistiques . . . 74
4.2 Inégalités de Bernstein empiriques pour les U-statistiques . . . 76
4.3 Optimisation des calculs d'une U-statistique . . . 78
4.3.1 Ordonnancement bipartite . . . 79
4.3.2 Au-delà de l'ordonnancement bipartite . . . 82
4.4 Application au ranking bipartite . . . 84
4.4.1 Bornes sur la taille de l'échantillon de test . . . 84
4.4.2 Ordonnancement en ligne et algorithmes empiriques de course 85
4.5 Conclusion . . . 87


Les problèmes de classification ou de régression étant déjà fort bien étudiés, la communauté de l'apprentissage automatique se tourne de plus en plus vers des problèmes plus riches. Parmi ces problèmes, un en particulier a motivé le travail présenté dans ce chapitre. Il s'agit du problème d'ordonnancement qui a pour but d'apprendre une fonction capable de prédire un ordre précis sur des objets en se basant sur les informations pertinentes les concernant. La résolution de tels problèmes implique généralement l'utilisation d'une fonction de coût (ou fonction de perte) différente de la perte 0-1 de mauvaise classification comme, par exemple, la perte 0-1 de *mauvais ordonnancement* [Clémençon et al. 2008] qui permet de compter le nombre de *paires discordantes* ou alors une perte *surrogée*.[1] Pour deux couples $(\boldsymbol{x}, y)$ et $(\boldsymbol{x}', y')$ appartenant à l'espace $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ (où par exemple, $\mathcal{X} = \mathbb{R}^d$ et $\mathcal{Y} = \mathbb{R}$) la perte de mauvais ordonnancement $\ell^{ord}$ et une perte convexe $\ell^{sur}$

1. Ce terme est une francisation du terme anglais « *surrogate* » utilisée par la communauté de l'apprentissage automatique. Le terme correct serait plutôt « *subrogée* ».

peuvent être définies pour une fonction de score $f \in \mathcal{Y}^{\mathcal{X}}$ comme les fonctions suivantes :

$$\ell^{ord}(f,(\boldsymbol{x},y),(\boldsymbol{x}',y')) := \mathbb{1}_{[(y-y')(f(\boldsymbol{x})-f(\boldsymbol{x}'))<0]} \tag{4.1}$$
$$\ell^{sur}(f,(\boldsymbol{x},y),(\boldsymbol{x}',y')) := \left[1-(y-y')(f(\boldsymbol{x})-f(\boldsymbol{x}'))\right]^2 \tag{4.2}$$

Etant données de telles fonctions de perte ou, plus généralement, une fonction de perte $\ell : \mathcal{Y}^{\mathcal{X}} \times \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$, et un échantillon d'apprentissage $\underline{Z}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ composé de copies *indépendantes* d'une variable aléatoire $Z := (X,Y)$ distribuée selon une distribution de probabilité $\mathcal{D}$, la tâche d'apprentissage d'une fonction $f \in \mathcal{Y}^{\mathcal{X}}$ est telle que l'espérance du risque (ou erreur de généralisation) de la fonction $f$ basé sur la perte $\ell$, notée $R_\ell(f)$ et définie par

$$R_\ell(f) := \mathbb{E}_{Z,Z'\sim\mathcal{D}}\left[\ell(f,Z,Z')\right] = \mathbb{E}_{Z,Z'\sim\mathcal{D}}\left[\ell(f,(X,Y),(X',Y'))\right]$$

soit la plus petite possible. En pratique, cela se traduit par la minimisation d'une estimation empirique $\hat{R}_\ell(f,\underline{Z}_n)$ de l'erreur de généralisation $R_\ell(f)$ sur l'échantillon $\underline{Z}_n$. On appelle $\hat{R}_\ell(f,\underline{Z}_n)$ l'erreur empirique de la fonction d'ordonnancement $f$ sur l'échantillon d'apprentissage $\underline{Z}_n$ définie comme

$$\hat{R}_\ell(f,\underline{Z}_n) := \frac{1}{n(n-1)}\sum_{i\neq j}\ell(f,(\boldsymbol{x}_i,y_i),(\boldsymbol{x}_j,y_j)), \tag{4.3}$$

qui se trouve être une *U-statistique* [Hoeffding 1948, Clémençon et al. 2008].

Une question importante est de caractériser précisément la façon dont l'erreur d'ordonnancement empirique $\hat{R}_\ell(f,\underline{Z}_n)$ est reliée à l'erreur réelle $R_\ell(f)$ et, plus spécifiquement, on aimerait pouvoir majorer $R_\ell(f)$ par un majorant exprimé en terme de $\hat{R}_\ell(f,\underline{Z}_n)$ et d'autres quantités comme une mesure de la capacité de la classe de fonctions à laquelle $f$ appartient et la taille $n$ de $\underline{Z}_n$ – en d'autres mots, on aimerait dériver des bornes de généralisation [Boucheron et al. 2005]. Des outils majeurs pour l'obtention de telles bornes sont les *inégalités de concentration*, lesquelles prédisent avec quelle probabilité une fonction de plusieurs variables indépendantes est amenée à prendre des valeurs éloignées de son espérance ; bien sûr, plus l'inégalité de concentration est précise et plus la caractérisation de la relation entre l'estimation empirique et l'espérance de la fonction étudiée le sera. Il est donc très important d'avoir en notre possession des inégalités de concentration très précises ; il est également important que ces inégalités reposent autant que possible sur des quantités qui peuvent être mesurées à partir de $\underline{Z}_n$.

Dans ce chapitre, nous proposons de nouvelles *inégalités empiriques de type Bernstein* pour les U-statistiques. Comme l'indique leur nom (i) nos résultats sont des inégalités de type Bernstein et donc utilisent l'information obtenue sur la variance des variables considérées, (ii) au lieu d'être des inégalités mettant en jeu une certaine connaissance à propos de cette variance (qui est rarement disponible pour de réelles problématiques), elles ne reposent que sur des quantités empiriques et (iii) elles s'appliquent

aux U-statistiques. Nos nouvelles inégalités, dont l'obtention repose sur des arguments simples, généralisent celles obtenues dans Audibert et al. [2007] et dans Maurer et Pontil [2009], lesquelles fournissent également les points (i) et (ii) (mais *pas* (iii)). A notre connaissance, ce sont les premiers résultats pour lesquels les points (i), (ii) et (iii) sont tous vérifiés. Ainsi, ces nouvelles inégalités donnent lieu à certaines applications, liées aux tâches d'ordonnancement, que nous allons décrire dans le paragraphe suivant.

Les applications que nous allons présenter ne sont que quelques possibilités qui nous sont offertes par les résultats présentés. La première application que nous évoquerons sera l'établissemment de bornes impliquant la taille de l'échantillon de test dans le cadre d'une tâche d'ordonnancement bipartite. Ces bornes interviennent, par exemple, dans le but de sélectionner un modèle $f$ appris sur une certaine proportion de l'ensemble $\underline{Z}_n$ réservée à l'apprentissage et testé sur le reste des exemples non sélectionnés. On s'intéresse alors à la taille de l'intervalle de confiance associé à l'estimation de l'erreur de généralistaion de $f$ en fonction de la taille de l'échantillon de test. La seconde application sera liée à la notion d'ordonnancement en ligne. Cette notion signifie que les exemples contenus dans $\underline{Z}_n$ apparaissent au fur et à mesure et donnent lieu à l'apprentissage de $n$ modèles de la façon suivante : le modèle $f_{i-1}$ est mis à jour lors de l'arrivée de l'exemple $Z_i$ et donne lieu à l'émergence du modèle $f_i$, lui même utilisé avec l'exemple $Z_{i+1}$ et ainsi de suite. La question est alors de savoir comment utiliser au mieux tous les modèles appris.

Le travail présenté ici a fait l'objet d'une publication [Peel et al. 2010] lors de la conférence NIPS 2010.

Le chapitre est organisé comme suit. La section 4.1 commence par l'introduction des notations utilisées tout au long du chapitre et rappelle brièvement les bases concernant aussi bien les U-statistiques que les inégalités de concentration déjà existantes sur lesquelles sont basés nos résultats. Nos inégalités empiriques de type Bernstein sont présentées dans la section 4.2 alors que nous montrons dans la section 4.3 un moyen efficace de calculer la variance empirique quand les U-statistiques considérées sont basées sur la perte de mauvais ordonnancement $\ell^{ord}$ de l'équation (4.1). La section 4.4 fournit deux exemples applicatifs pour les résultats théoriques présentés. La première partie 4.4.1 présente une utilisation des inégalités empiriques pour l'établissement de bornes d'erreur de généralisation en fonction de la taille de l'échantillon sur lequel une fonction d'ordonnancement $f$ a été testée. La deuxième 4.4.2 présente quant à elle une application pour la sélection de modèle dans le cadre de l'ordonnancement en ligne (algorithmes de *course*) ainsi qu'une partie plus algorithmique permettant d'optimiser le temps de calcul des U-statistiques mises en jeu dans ce problème.

## 4.1 Contexte

### 4.1.1 Quelques notations

Nous commençons par introduire quelques notations que nous allons utiliser dans la suite de ce chapitre. La variable $Z$ est une variable aléatoire distribuée selon une distribution de probabilité $\mathcal{D}$ qui prend ses valeurs dans l'espace $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$. Pour un entier $n > 1$ donné, les variables $Z', Z_1, \ldots, Z_n$ sont des copies indépendantes de $Z$ et nous considérerons les ensembles

$$\underline{Z}_n := \{Z_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^n \text{ et } \underline{Z}_{p:q} := \{Z_i\}_{i=p}^q .$$

Pour un entier $m$ tel que $0 \leq m \leq n$, nous notons $\mathbf{P}_{m,n}$ les ensemble de $m$ éléments pris parmi $n$ défini par

$$\mathbf{P}_{m,n} := \{(i_1, \ldots, i_m) : 1 \leq i_1 \neq \ldots \neq i_m \leq n\} .$$

Il est évident que la taille de l'ensemble $\mathbf{P}_{m,n}$ est $|\mathbf{P}_{m,n}| = m!\binom{n}{m}$. Nous introduisons également l'ensemble $\mathbf{C}_{m,n}$ des combinaisons de $m$ éléments pris parmi $n$ tel que

$$\mathbf{C}_{m,n} := \{(i_1, \ldots, i_m) : 1 \leq i_1 < \ldots < i_m \leq n\} .$$

On peut aisément remarquer que la taille de $\mathbf{C}_{m,n}$ est $|\mathbf{C}_{m,n}| = \frac{1}{m!}|\mathbf{P}_{m,n}| = \binom{n}{m}$. Enfin, une fonction $q : \mathcal{Z}^m \to \mathbb{R}$ est dite symétrique si la valeur de $q(\underline{Z}_m) = q(Z_1, \ldots, Z_m)$ est indépendante de l'ordre des $Z_i$ dans $\underline{Z}_m$. Avant d'évoquer quelques notions sur les U-statistiques, on rappelle un résultat de base en théorie des probabilités connu sous le nom de « borne de l'union » :

**Proposition 4.1** **(Borne de l'union)** *Soit $A_1, \ldots, A_n$ des évènements, alors*

$$\mathbb{P}\left[A_1 \cup \ldots \cup A_n\right] \leq \mathbb{P}\left[A_1\right] + \ldots + \mathbb{P}\left[A_n\right] .$$

### 4.1.2 U-statistique

**Définition 4.1** **(U-statistique [Hoeffding 1948])** *Soit $q : \mathcal{Z}^m \to \mathbb{R}$ une fonction mesurable sur l'espace $\mathcal{Z}^m$. La variable aléatoire $\hat{U}_q(\underline{Z}_n)$ définie par*

$$\hat{U}_q(\underline{Z}_n) := \frac{m!}{\binom{n}{m}} \sum_{\mathbf{i} \in \mathbf{P}_{m,n}} q(Z_{i_1}, \ldots, Z_{i_m}) \tag{4.4}$$

*est une U-statistique d'ordre $m$ et de noyau $q$. De plus, si le noyau $q$ est symétrique, $\hat{U}_q(\underline{Z}_n)$ s'écrit alors de façon équivalente à l'équation (4.4) sous la forme*

$$\hat{U}_q(\underline{Z}_n) := \frac{1}{\binom{n}{m}} \sum_{\mathbf{i} \in \mathbf{C}_{m,n}} q(Z_{i_1}, \ldots, Z_{i_m}). \tag{4.5}$$

**Exemple 4.1** *Le risque empirique pour l'ordonnancement dont on rappelle l'expression ci-dessous*

$$\hat{R}_\ell(f, \underline{Z}_n) := \frac{1}{n(n-1)} \sum_{i \neq j} \ell(f, (\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)),$$

*est une U-statistique d'ordre 2. En effet, c'est une estimation qui repose sur les valeurs de $\ell(f, Z, Z')$ pour tous les couples $(Z, Z')$ que l'on peut former avec les éléments de $\underline{Z}_n$.*

**Remarque 4.1** *Il est facile de voir que*

$$\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right] = \mathbb{E}_{\underline{Z}_n}\left[\hat{U}_q(\underline{Z}_n)\right];$$

*de plus* $\mathbb{E}_{\underline{Z}_n}\left[\hat{U}_q(\underline{Z}_n)\right]$ *est un estimateur sans biais de* $\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right]$ *basé sur* $\underline{Z}_n$ *possédant la plus petite variance [Hoeffding 1948].*

**Remarque 4.2** *Les deux spécificités des U-statistiques nécessitant une attention toute particulière sont les suivantes : (i) elles sont des sommes de variables aléatoires identiquement distribuées mais* dépendantes *entre elles ce qui implique l'utilisation d'outils spécifiques permettant de traiter ces dépendances afin de pouvoir caractériser la déviation de* $\hat{U}_q(\underline{Z}_n)$ *par rapport à* $\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right]$*, et (ii) d'un point de vue algorithmique la complexité de calcul de ces sommes peut être très élevée puisqu'elle varie en* $\mathcal{O}(n^m)$*. A ce sujet, en Section 4.3, nous proposons une approche permettant de réduire de façon significative la complexité de calcul de la U-statistique dans le cas particulier de l'ordonnancement bipartite.*

### 4.1.3 Inégalités de concentration pour les U-statistiques

Nous pouvons maintenant rappeler trois inégalités de concentration (Equations (4.6), (4.7), (4.8)), concernant les U-statistiques possédant un noyau $q$ symétrique et borné. Ces inégalités utilisent en général de façon explicite le fait que $q \in [q_{\min}; q_{\max}]$. Cependant et afin de simplifier la lecture, nous considérerons sans perte de généralité que $q$ prend ses valeurs dans l'intervalle $[0, 1]$. Une façon simple de retrouver les résultats proposés à partir d'une fonction $q$ dont on sait seulement qu'elle est bornée est de considérer $(q/\left\|q\right\|_\infty + 1)/2$.

Une quantité importante qui apparaît dans les inégalités (4.6) and (4.7) ci-dessous est $\lfloor n/m \rfloor$. Cette quantité, la partie entière du ratio $n/m$, peut être vue comme le nombre *effectif* de données disponibles une fois traitées les dépendances entre $m$-uplets. Toujours dans l'optique de rendre la lecture de ces inégalités plus aisée, nous allons émettre l'hypothèse que $n$ est un multiple de $m$, ce qui nous permet alors d'avoir $\lfloor n/m \rfloor = (n/m)$.

Le premier théorème que nous rappelons est dû à Hoeffding. Il a prouvé les résultats suivants concernant la déviation de $\hat{U}_q(\underline{Z}_n)$ par rapport à $\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right]$,

**Théorème 4.1** **(Inégalité de concentration du premier ordre pour $\hat{\mathbf{U}}_\mathbf{q}$, [Hoeffding 1963])** *Soit* $\underline{Z}_n$ *un échantillon de variables aléatoires indépendantes et* $q \in [0, 1]$ *un noyau symétrique d'ordre* $m$*. Alors,* $\forall \varepsilon > 0$

$$\mathbb{P}_{\underline{Z}_n}\left[\left|\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right] - \hat{U}_q(\underline{Z}_n)\right| \geq \varepsilon\right] \leq 2\exp\left(-(n/m)\varepsilon^2\right).$$

*Il s'ensuit que* $\forall \delta \in (0, 1]$*, avec une probabilité au moins* $1 - \delta$ *sur la réalisation de* $\underline{Z}_n$

$$\left|\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right] - \hat{U}_q(\underline{Z}_n)\right| \leq \sqrt{\frac{1}{(n/m)}\ln\left(\frac{2}{\delta}\right)}. \tag{4.6}$$

Afin de passer de la version exponentielle de l'inégalité de concentration à la version bornée présentée dans l'équation (4.6), il suffit d'utiliser le très utile *lemme de renversement d'inégalité* (Lemme 1.1) présenté dans le premier chapitre de ce manuscrit et rappelé à la section 4.2. Ce lemme est de nouveau utilisé dans les deux bornes données ci-dessous.

Hoeffding [1963] et, plus tard, [Arcones 1995] ont raffiné le résultat précédent sous la forme d'inégalités de type Bernstein (faisant apparaître un moment du second ordre). Ces deux nouvelles inégalités sont détaillées dans le théorème qui suit.

**Théorème 4.2** **(Inégalité de Bernstein pour $\hat{\mathbf{U}}_\mathbf{q}$ - Arcones [1995], Hoeffding [1963])** *Soit $\underline{Z}_n$ un échantillon de variables aléatoires indépendantes et $q \in [0,1]$ un noyau symétrique d'ordre $m$, alors pour $a, \vartheta_{q,m}$ et $b_m$ définis ci-après :*

$$\forall \varepsilon > 0,\ \mathbb{P}_{\underline{Z}_n}\left[\left|\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right] - \hat{U}_q(\underline{Z}_n)\right| \geq \varepsilon\right] \leq a \exp\left\{-\frac{(n/m)\varepsilon^2}{2\vartheta_{q,m} + b_m\varepsilon}\right\}.$$

*Pour Hoeffding, $a = 2$, $\vartheta_{q,m} = \Sigma_q^2$, où $\Sigma_q^2$ est la variance de $q(\underline{Z}_m)$ et $b_m = 2/3$. Par la suite, $\forall \delta \in ]0,1]$, avec une probabilité au moins $1-\delta$ :*

$$\left|\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right] - \hat{U}_q(\underline{Z}_n)\right| \leq \sqrt{\frac{2\Sigma_q^2}{(n/m)}\ln\left(\frac{2}{\delta}\right)} + \frac{2}{3(n/m)}\ln\left(\frac{2}{\delta}\right). \quad (4.7)$$

*Pour Arcones, $a = 4$, $\vartheta_{q,m} = m\sigma_q^2$, où $\sigma_q^2$ est la variance de $\mathbb{E}_{Z_2,\ldots,Z_m}\left[q(\underline{Z}_m)|Z_1\right]$ (qui est une fonction de la variable $Z_1$ uniquement) et $b_m = 2^{m+3}m^{m-1} + (2/3)m^{-2}$. Ainsi, $\forall \delta \in ]0,1]$ avec une probabilité au moins $1-\delta$ :*

$$\left|\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right] - \hat{U}_q(\underline{Z}_n)\right| \leq \sqrt{\frac{2m\sigma_q^2}{(n/m)}\ln\left(\frac{4}{\delta}\right)} + \frac{b_m}{(n/m)}\ln\left(\frac{4}{\delta}\right). \quad (4.8)$$

Nous évoquerons à présent, et ce de manière légèrement abusive, les équations (4.6), (4.7) et (4.8) en utilisant les termes « inégalités de concentration ». En réalité, elles sont plus précisément des intervalles de confiance pour $\mathbb{E}_{\underline{Z}_n}\left[q(\underline{Z}_n)\right]$.

**Remarque 4.3** *L'équation (4.8) est basée sur la célèbre décomposition de Hoeffding des U-statistiques [Hoeffding 1963]. Elle fournit une inégalité de type Bernstein plus précise que celle de l'équation (4.7), puisque $m\sigma_q^2$ est connu pour être plus petit que $\Sigma_q^2$ (voir Serfling [1980]). Cependant, pour des valeurs $n/m$ modérées (de l'ordre de $n/m < 10^5$) et des valeurs raisonnables de $\delta$ (par exemple $\delta = 0.05$), l'influence du terme logarithmique peut être telle que l'avantage de l'équation (4.8) sur (4.7) (donné par $m\sigma_q^2$) devienne insignifiant (voire s'inverse). Ainsi, nous détaillons nos résultats en nous concentrant sur une version empirique de (4.7).*

**Exemple 4.2** *Pour illustrer la façon dont l'utilisation de l'information donnée par la variance fournit des intervalles de confiance plus petits, nous considérons un noyau spécifique $q_m$ et deux distributions $\mathcal{D}_U$ et $\mathcal{D}_{B_p}$. Le noyau $q_m$ est tel que $q_m(\underline{Z}_m) := \prod_{i=1}^m Z_i$. $\mathcal{D}_U$ est la distribution uniforme sur l'intervalle $[0,1]$ :*

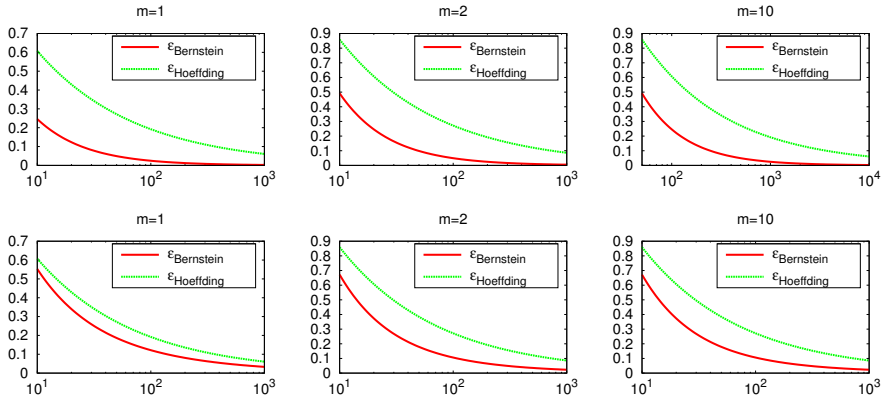



FIGURE 4.1 – *Première ligne : valeurs du membre de droite des équations (4.6) et (4.7), pour la distribution $\mathcal{D}_U$ et le noyau $q_m$ pour $m = 1$, $m = 2$ et $m = 10$ (voir les détails dans l'exemple 4.2) comme des fonctions de $n$ (axe des abscisses). Seconde ligne : la même chose avec la distribution $\mathcal{D}_{B_{0.15}}$.*

*il est facile de voir que pour cette distribution, $\Sigma^2 = \frac{1}{3^m} - \frac{1}{4^m}$. $\mathcal{D}_{B_p}$ est la distribution de Bernoulli de paramètre $p \in [0,1]$ pour laquelle $\Sigma^2 = p^m(1-p^m)$. La Figure 4.1 montre le comportement de (4.6) et (4.7) en fonction de la taille $n$ de l'échantillon pour plusieurs valeurs $m$. On observe que l'information sur la variance apportée par les inégalités de type Bernstein rend celles-ci plus précises. Cette amélioration est particulièrement notable pour des échantillons de petite taille.*

## 4.2 Inégalités de Bernstein empiriques pour les U-statistiques

Cette section contient les principaux résultats théoriques de ce chapitre. Elle expose les nouvelles inégalités empiriques de type Bernstein pour les U-statistiques que nous proposons. Avant d'aller plus loin, nous rappelons ici le *lemme de renversement d'inégalité* qui nous permet de transformer de façon immédiate une inégalité de concentration sous la forme d'une borne supérieure (ou d'un intervalle de confiance) comme cela a été fait dans les équations (4.6), (4.7) et (4.8).

**Lemme 4.1** **(Lemme de renversement d'inégalité)** *Soit $X$ une variable aléatoire. S'il existe $a, b > 0$ d'une part, $c, d \geq 0$ non nuls ensemble simultanément d'autre part, tels que*

$$\forall \varepsilon > 0,\ \mathbb{P}_X\left[|X| \geq \varepsilon\right] \leq a \exp\left\{-\frac{b\varepsilon^2}{c + d\varepsilon}\right\}, \tag{4.9}$$

*alors,*

$$\mathbb{P}\left[|X| \leq \sqrt{\frac{c}{b}\ln\frac{a}{\delta}} + \frac{d}{b}\ln\frac{a}{\delta}\right] \geq 1 - \delta. \tag{4.10}$$

**Définition 4.2** *Soit $\hat{\Sigma}_q^2$ la U-statistique d'ordre $2m$ définie par :*

$$\hat{\Sigma}_q^2(\underline{Z}_n) := \frac{1}{|\mathbf{P}_{2m,n}|} \sum_{\mathbf{i} \in \mathbf{P}_{2m,n}} \left(q(Z_{i_1}, \ldots, Z_{i_m}) - q(Z_{i_{m+1}}, \ldots, Z_{i_{2m}})\right)^2 \tag{4.11}$$

*et* $\hat{\sigma}_q^2$ *la U-statistique d'ordre* $2m-1$ *définie par :*

$$\hat{\sigma}_q^2(\underline{Z}_n) := \frac{1}{|\mathbf{P}_{2m-1,n}|} \sum_{\mathbf{i}\in\mathbf{P}_{2m-1,n}} q(Z_{i_1}, Z_{i_2}, \ldots, Z_{i_m}) q(Z_{i_1}, Z_{i_{m+1}} \ldots, Z_{i_{2m-1}}). \tag{4.12}$$

Ces deux U-statistiques sont des estimateurs basés sur l'échantillon $\underline{Z}_n$ de la variance et de la variance conditionnelle de $q(\underline{Z}_m)$. On voit de façon évidente (cf. les définitions de $\Sigma_q^2$ dans (4.7) et de $\sigma_q^2$ dans (4.8)) que :

$$\mathbb{E}_{\underline{Z}_n}\left[\hat{\Sigma}_q^2(\underline{Z}_n)\right] = \Sigma_q^2, \quad \text{et} \quad \mathbb{E}_{\underline{Z}_n}\left[\hat{\sigma}_q^2(\underline{Z}_n)\right] = \sigma_q^2 + \mathbb{E}_{\underline{Z}_m}\left[q(Z_1, \ldots, Z_m)\right]^2.$$

Nous sommes maintenant en mesure de montrer le théorème suivant qui est le principal résultat de ce chapitre.

**Théorème 4.3** **(Bornes empiriques de type Bernstein)** *Avec une probabilité au moins* $1-\delta$ *sur la réalisation de* $\underline{Z}_n$*,*

$$\left|\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right] - \hat{U}_q(\underline{Z}_n)\right| \leq \sqrt{\frac{2\hat{\Sigma}_q^2}{(n/m)} \ln\left(\frac{4}{\delta}\right)} + \frac{5}{(n/m)} \ln\left(\frac{4}{\delta}\right). \tag{4.13}$$

*Et, avec une probabilité* $1-\delta$ *également,*

$$\left|\mathbb{E}_{\underline{Z}_m}\left[q(\underline{Z}_m)\right] - \hat{U}_q(\underline{Z}_n)\right| \leq \sqrt{\frac{2m\hat{\sigma}_q^2}{(n/m)} \ln\left(\frac{8}{\delta}\right)} + \frac{5\sqrt{m} + b_m}{(n/m)} \ln\left(\frac{8}{\delta}\right). \tag{4.14}$$

*Démonstration.* La démonstration suivante prouve la borne supérieure de l'intervalle de confiance pour (4.13) ; le même raisonnement peut être appliqué pour montrer la borne inférieure. La démonstration de l'équation (4.14) est très similaire, les détails spécifiques à celle-ci sont donnés en commentaire. Dans un premier temps, appelons $Q$ le noyau de $\hat{\Sigma}_q^2$ :

$$Q(Z_1, \ldots, Z_{2m}) := \left(q(Z_1, \ldots, Z_m) - q(Z_{m+1}, \ldots, Z_{2m})\right)^2$$

$Q$ est d'ordre $2m$, prend ses valeurs dans l'intervalle $[0;1]$ mais n'est pas nécessairement symétrique. Une version *symétrique* et équivalente pour $\hat{\Sigma}_q^2$ du noyau $Q$ est le noyau $Q_{\text{sym}}$ défini par :

$$Q_{\text{sym}}(Z_1, \ldots, Z_{2m}) := \frac{1}{(2m)!} \sum_{\omega\in\mathcal{P}_{2m}} \left(q(Z_{\omega(1)}, \ldots, Z_{\omega(m)}) - q(Z_{\omega(m+1)}, \ldots, Z_{\omega(2m)})\right)^2$$

où $\mathcal{P}_{2m}$ est l'ensemble de toutes les permutations sur l'ensemble $\{1, \ldots, 2m\}$. En remplaçant le noyau $Q$ par le noyau $Q_{\text{sym}}$ à présent symétrique (et prenant toujours ses valeurs dans l'intervalle $[0,1]$) et le théorème 4.2 peut dès lors être utilisé pour borner $\Sigma^2$. En remarquant ensuite que $\Sigma^2 = \mathbb{E}_{\underline{Z}_{2m}}\left[Q_{\text{sym}}(\underline{Z}_{2m})\right] = \mathbb{E}_{\underline{Z}_n}\left[\hat{\Sigma}_q^2(\underline{Z}_n)\right]$, avec une probabilité au moins $1-\delta$

$$\Sigma^2 \leq \hat{\Sigma}_q^2(\underline{Z}_n) + \sqrt{\frac{2\mathbb{V}\left[Q_{\text{sym}}\right]}{(n/2m)} \ln\left(\frac{2}{\delta}\right)} + \frac{2}{3(n/2m)} \ln\left(\frac{2}{\delta}\right),$$

où $\mathbb{V}\left[Q_{\text{sym}}\right]$ est la variance de $Q_{\text{sym}}$. En utilisant alors le fait que $Q_{\text{sym}}$ appartient à l'intervalle $[0,1]$, on a

$$\mathbb{V}\left[Q_{\text{sym}}\right] = \mathbb{E}\left[Q_{\text{sym}}^2\right] - \mathbb{E}\left[Q_{\text{sym}}\right]^2 \leq \mathbb{E}\left[Q_{\text{sym}}^2\right] \leq \mathbb{E}\left[Q_{\text{sym}}\right] = \Sigma^2 \quad (4.15)$$

(pour établir (4.14) nous utilisons de plus $\hat{\sigma}_q^2(\underline{Z}_n) \geq \sigma_q^2$) et par la suite

$$\Sigma^2 \leq \hat{\Sigma}_q^2(\underline{Z}_n) + \sqrt{\frac{4\Sigma^2}{(n/m)} \ln\left(\frac{2}{\delta}\right)} + \frac{4}{3(n/m)} \ln\left(\frac{2}{\delta}\right).$$

En suivant une approche identique à celle donnée dans Maurer et Pontil [2009], on introduit la quantité $\left(\sqrt{\Sigma^2} - \sqrt{(m/n)\ln(2/\delta)}\right)^2$ et on obtient

$$\left(\sqrt{\Sigma^2} - \sqrt{(m/n)\ln(2/\delta)}\right)^2 \leq \hat{\Sigma}_q^2(\underline{Z}_n) + \frac{7}{3(n/m)} \ln\left(\frac{2}{\delta}\right).$$

En prenant la racine carrée à la fois du membre de droite et de celui de gauche de l'inégalité puis en utilisant de nouveau $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ pour $a, b \geq 0$ on arrive à

$$\begin{aligned}
\sqrt{\Sigma^2} &\leq \sqrt{\hat{\Sigma}_q^2(\underline{Z}_n)} + \sqrt{\frac{7}{3(n/m)} \ln\left(\frac{2}{\delta}\right)} + \sqrt{\frac{1}{(n/m)} \ln\left(\frac{2}{\delta}\right)} \\
&\leq \sqrt{\hat{\Sigma}_q^2(\underline{Z}_n)} + (1+\sqrt{7/3}) \sqrt{\frac{1}{(n/m)} \ln\left(\frac{2}{\delta}\right)} \\
&\leq \sqrt{\hat{\Sigma}_q^2(\underline{Z}_n)} + 3 \sqrt{\frac{1}{(n/m)} \ln\left(\frac{2}{\delta}\right)},
\end{aligned}$$

toujours avec une probabilité au moins $1-\delta$. Pour terminer, on utilise une borne de l'union (Proposition 4.1) en remplaçant $\delta$ par $\delta/2$ dans le résultat précédent ainsi que dans (4.7). Quelques étapes simples de calcul et une majoration appropriée de certaines constantes terminent la preuve et donnent le résultat désiré. □

**Remarque 4.4** *En plus de fournir une version empirique de la borne de type Bernstein pour les U-statistiques, notre résultat diffère de celui de Maurer et Pontil [2009] par la façon dont nous l'obtenons. Ici, nous appliquons deux fois la même inégalité de concentration, en tirant parti du fait que les estimations pour les variances qui nous intéressent sont également des U-statistiques. Maurer et Pontil utilisent une inégalité de concentration sur les fonctions aléatoires auto-bornées (*self bounding random functions*) et n'utilisent pas explicitement à leur avantage que les estimations qu'ils considèrent sont des U-statistiques.*

## 4.3 Optimisation des calculs d'une U-statistique

Dans la section précédente, nous avons décrit de nouvelles inégalités de concentration empiriques de type Bernstein pour les U-statistiques. Les estimations rentrant en jeu sont dans les résultats présentés précédemment sont des U-statistiques dont le noyau est d'ordre $m$ pour $\hat{U}_q$, d'ordre $2m$ pour $\hat{\Sigma}_q^2$ et d'ordre $2m-1$ pour $\hat{\sigma}_q^2$. Cela signifie qu'une approche directe pour le

calcul de ces deux quantités se soldera par une complexité calculatoire de l'ordre de $\mathcal{O}(n^m)$ opérations pour $\hat{U}_q$, $\mathcal{O}(n^{2m})$ opérations pour $\hat{\Sigma}^2_q$ ou encore $\mathcal{O}(n^{2m-1})$ pour $\hat{\sigma}^2_q$. Cette complexité est au coeur de plusieurs travaux (dont ceux de Lin et Xi [2010]) et il est évident qu'elle peut vite devenir problématique dès lors que $n$ devient grand. On propose donc dans cette section une manière beaucoup plus rapide afin d'évaluer les estimateurs $\hat{\Sigma}^2_q$ et $\hat{\sigma}^2_q$ lorsque la fonction de perte $\ell_f$ pour une fonction d'ordonnancement $f$ est de la forme

$$\ell_f(Z, Z') = \mathbb{1}_{[(y-y')(f(\boldsymbol{x})-f(\boldsymbol{x}'))<0]}\,. \tag{4.16}$$

Ce noyau est un noyau symétrique d'ordre $m = 2$ tel que $\ell \in [0,1]^{\mathcal{Y}\times\mathcal{Y}}$. On introduit tout d'abord les versions spécifiques des U-statistiques évoquées ci-dessus pour le cas d'un problème d'ordonnancement utilisant la perte $\ell_f$. Tout d'abord, le risque empirique d'une fonction d'ordonnancement $f$ mesuré sur un échantillon $\underline{Z}_n$ est défini par la U-statistique

$$\hat{U}_{\ell_f}(\underline{Z}_n) := \frac{1}{|\mathbf{P}_{2,n}|}\sum_{\mathbf{i}\in\mathbf{P}_{2,n}} \ell_f(Z_{i_1}, Z_{i_2}) = \frac{2}{|\mathbf{C}_{2,n}|}\sum_{\mathbf{i}\in\mathbf{C}_{2,n}} \ell_f(Z_{i_1}, Z_{i_2}). \tag{4.17}$$

La variance empirique de cette même fonction d'ordonnancement mesurée elle-aussi sur l'échantillon $\underline{Z}_n$ est quant à elle donnée par la U-statistique

$$\hat{\Sigma}^2_{\ell_f}(\underline{Z}_n) := \frac{1}{|\mathbf{P}_{4,n}|}\sum_{\mathbf{i}\in\mathbf{P}_{4,n}} \left(\ell_f(Z_{i_1}, Z_{i_2}) - \ell_f(Z_{i_3}, Z_{i_4})\right)^2. \tag{4.18}$$

Enfin, la variance empirique de l'espérance conditionnelle pour la fonction $f$ mesurée sur $\underline{Z}_n$ est la U-statistique

$$\hat{\sigma}^2_{\ell_f}(\underline{Z}_n) := \frac{1}{|\mathbf{P}_{3,n}|}\sum_{\mathbf{i}\in\mathbf{P}_{3,n}} \ell_f(Z_{i_1}, Z_{i_2})\ell_f(Z_{i_1}, Z_{i_3}). \tag{4.19}$$

**Remarque 4.5** *Une approche naïve pour le calcul de ces U-statistiques entraine dans ce cas $\mathcal{O}(n^2)$ opérations pour $\hat{U}_{\ell_f}$, $\mathcal{O}(n^4)$ opérations pour $\hat{\Sigma}^2_{\ell_f}$ et $\mathcal{O}(n^3)$ opérations pour $\hat{\sigma}^2_{\ell_f}$.*

### 4.3.1 Ordonnancement bipartite

Nous proposons ici une méthode efficace (de l'ordre de $\mathcal{O}(n \log n)$ opérations) pour calculer les quantités exprimées ci-dessus dans le cadre d'un problème d'ordonnancement bipartite. Par « ordonnancement bipartite » nous entendons un problème d'ordonnancement pour lequel la valeur d'un exemple $\boldsymbol{x}$ appartient à l'espace $\mathcal{Y} = \{-1, +1\}$. Dans ce cas, nous avons le résultat suivant

**Proposition 4.2** *Pour tout entier $n > 0$, le calcul des U-statistiques $\hat{U}_{\ell_f}$, $\hat{\sigma}^2_{\ell_f}$ et $\hat{\Sigma}^2_{\ell_f}$ définies ci-dessus (équations (4.17), (4.18) et (4.19)) dans le cadre de l'ordonnancement bipartite peut être effectué en $\mathcal{O}(n \log n)$.*

*Démonstration.* Nous donnons un algorithme de calcul de $\hat{\Sigma}^2_{\ell_f}(\underline{Z}_n)$ basé sur une décomposition de cette U-statistique faisant intervenir des termes proportionnels à $\hat{U}_{\ell_f}$ et $\hat{\sigma}^2_{\ell_f}$. Afin de faciliter la lecture, on remplace les termes $i_1, i_2, i_3, i_4$ par $i, j, k, l$ respectivement. En remarquant que $\ell^2_f(Z, Z') =$

$\ell_f(Z,Z')$ et $\ell_f(Z,Z) = 0$ puis en notant pour plus de clarté $S(\underline{Z}_n) := \sum_{i,j} \ell_f(Z_i,Z_j)$, nous avons la décomposition suivante

$$\begin{aligned}
\hat{\Sigma}^2_{\ell_f}(\underline{Z}_n) &= \frac{1}{|\mathbf{P}_{4,n}|} \sum_{\substack{i,j,k,l \\ i\neq j\neq k\neq l}} \left(\ell_f(Z_i,Z_j) - \ell_f(Z_k,Z_l)\right)^2 \\
&= \frac{1}{|\mathbf{P}_{4,n}|} \sum_{\substack{i,j,k,l \\ i\neq j\neq k\neq l}} \left(\ell_f^2(Z_i,Z_j) - 2\ell_f(Z_i,Z_j)\ell_f(Z_k,Z_l) + \ell_f^2(Z_k,Z_l)\right) \\
&= \frac{2}{|\mathbf{P}_{4,n}|} \left( (n-2)(n-3) \sum_{i,j} \ell_f(Z_i,Z_j) - \sum_{\substack{i,j,k,l \\ i\neq j\neq k\neq l}} \ell_f(Z_i,Z_j)\ell_f(Z_k,Z_l) \right) \\
&= \frac{2}{|\mathbf{P}_{4,n}|} \left( (n-2)(n-3) S(\underline{Z}_n) - \sum_{\substack{i,j,k,l \\ i\neq j\neq k\neq l}} \ell_f(Z_i,Z_j)\ell_f(Z_k,Z_l) \right).
\end{aligned}$$

**Remarque 4.6** *Le premier terme de la dernière ligne (en $S(\underline{Z}_n)$) est proportionnel à la statistique bien connue Wilcoxon-Mann-Whitney [Hanley et McNeil 1982]. Il existe des manières efficaces, basées sur un tri des valeurs de $\ell_f$, permettant de calculer cette statistique en n'effectuant que de l'ordre de $\mathcal{O}(n \log n)$ opérations. De plus, étant donné que $\ell_f(Z,Z) = 0$, les termes de la somme $\sum_{i,j} \ell_f(Z_i,Z_j)$ pour lesquels $i = j$ ont une contribution nulle et cette somme est donc également proportionnelle à $\hat{U}_{\ell_f}$.*

Nous allons maintenant voir comment s'occuper de la deuxièmre somme, en utilisant également une approche basée sur un tri des valeurs de $\ell_f$. Nous remarquons dans un premier temps la décomposition suivante

$$\begin{aligned}
\left(\sum_{i,j} \ell_f(Z_i,Z_j)\right)^2 &= \sum_{i,j} \ell_f(Z_i,Z_j) \sum_{k,l} \ell_f(Z_k,Z_l) \\
&= \sum_{\substack{i,j,k,l \\ i\neq j\neq k\neq l}} \ell_f(Z_i,Z_j)\ell_f(Z_k,Z_l) + 4 \sum_{i\neq j\neq k} \ell_f(Z_i,Z_j)\ell_f(Z_i,Z_k) \\
&\qquad - 2\sum_{i,j} \ell_f^2(Z_i,Z_j) \\
&= \sum_{\substack{i,j,k,l \\ i\neq j\neq k\neq l}} \ell_f(Z_i,Z_j)\ell_f(Z_k,Z_l) + 4 \sum_{i\neq j\neq k} \ell_f(Z_i,Z_j)\ell_f(Z_i,Z_k) \\
&\qquad - 2\sum_{i,j} \ell_f(Z_i,Z_j) \\
&= \sum_{\substack{i,j,k,l \\ i\neq j\neq k\neq l}} \ell_f(Z_i,Z_j)\ell_f(Z_k,Z_l) + 4 \sum_{i\neq j\neq k} \ell_f(Z_i,Z_j)\ell_f(Z_i,Z_k) \\
&\qquad - 2S(\underline{Z}_n).
\end{aligned}$$

En effet, la multiplication de la somme $\sum_{i,j} \ell_f(Z_i,Z_j)$ par elle-même se décompose en trois termes.

Le premier terme est constitué par tous les produits $\ell_f(Z_i,Z_j)\ell_f(Z_k,Z_l)$ pour lesquels toutes les variables sont distinctes.

Ensuite, il faut ajouter tous les produits tels qu'une variable (et une seule) apparaît à la fois dans $\ell_f(Z_i, Z_j)$ et dans $\ell_f(Z_k, Z_l)$, ce qui se produit quand $i = k$, $i = l$, $j = k$ ou encore $j = l$. En utilisant ensuite la symétrie de $\ell_f$ on obtient le second terme de la décomposition (y compris le facteur 4).

Enfin, il nous reste la somme des produits $\ell_f(Z_i, Z_j)\ell_f(Z_k, Z_l)$ pour lesquels $i = k$ et $j = l$ ou alors $i = l$ et $j = k$. Dans ce cas, cette somme se réduit à deux fois la somme des produits $\ell_f^2(Z_i, Z_j)$, ce qui nous donne le dernier terme de la décomposition.

La dernière ligne s'obtient en utilisant une nouvelle fois $\ell_f^2(Z, Z') = \ell_f(Z, Z')$. Quelques étapes simples de calcul nous permettent d'obtenir la décomposition

$$\begin{aligned}\hat{\Sigma}_{\ell_f}(\underline{Z}_n) = \frac{2}{|\mathbf{P}_{4,n}|}\Big[ &- S^2(\underline{Z}_n) + (n^2 - 5n + 8)\, S(\underline{Z}_n) \\ &+ 4 \sum_{i \neq j \neq k} \ell_f(Z_i, Z_j)\ell_f(Z_i, Z_k)\Big].\end{aligned} \tag{4.20}$$

Nous avons vu plus haut (remarque 4.6) que le terme $S(\underline{Z}_n)$ pouvait être évalué en $\mathcal{O}(n \log n)$. Le seul terme qui demande donc une attention particulière est le dernier terme de la décomposition ci-dessus (qui est proportionnel à $\hat{\sigma}^2_{\ell_f}(\underline{Z}_n)$). Notons maintenant $W(\underline{Z}_n) := \sum_{i \neq j \neq k} \ell_f(Z_i, Z_j)\ell_f(Z_i, Z_k)$ ce terme qui nous intéresse. Rappelons dans un premier temps que

$$\ell_f(Z_i, Z_j) = \mathbb{1}_{[(y_i - y_j)(f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)) < 0]}$$

et observons alors que

$$\ell_f(Z_i, Z_j)\ell_f(Z_i, Z_k) = 1 \Leftrightarrow \left\{ \begin{array}{l} y_i = -1,\ y_j = y_k = +1 \text{ et } f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j), f(\boldsymbol{x}_k) \\ \text{ou} \\ y_i = +1,\ y_j = y_k = -1 \text{ et } f(\boldsymbol{x}_i) < f(\boldsymbol{x}_j), f(\boldsymbol{x}_k) \end{array} \right. . \tag{4.21}$$

En effet, il est facile de remarquer que tous les cas de figure où $y_i = y_j$ ou $y_i = y_k$ entrainent $\ell_f(Z_i, Z_j)\ell_f(Z_i, Z_k) = 0$. Définissons à présent les ensembles $\mathcal{E}^+(i)$ et $\mathcal{E}^-(i)$ comme

$$\begin{aligned}\mathcal{E}^+(i) &:= \{j : y_j = -1,\ f(\boldsymbol{x}_j) > f(\boldsymbol{x}_i)\} \\ \mathcal{E}^-(i) &:= \{j : y_j = +1,\ f(\boldsymbol{x}_j) < f(\boldsymbol{x}_i)\}\end{aligned}$$

et leurs cardinalités $\kappa_i^+ := |\mathcal{E}^+(i)|$, et $\kappa_i^- := |\mathcal{E}^-(i)|$.

Pour $i$ tel que $y_i = +1$, $\kappa_i^+$ est le nombre d'instances négatives auxquelles la fonction $f$ a attribué un score plus élevé que celui de l'instance $\boldsymbol{x}_i$. La contribution de l'instance $Z_i$ à $W(\underline{Z}_n)$ correspond au nombre $\kappa_i^+(\kappa_i^+ - 1)$ de paires ordonnées dont les indices sont dans $\mathcal{E}^+(i)$ (cf. équation (4.21)). On applique un raisonnement similaire pour $\kappa_i^-$, avec $y_i = -1$ afin d'obtenir

$$W(\underline{Z}_n) = \sum_{i:Y_i=+1} \kappa_i^+(\kappa_i^+ - 1) + \sum_{i:Y_i=-1} \kappa_i^-(\kappa_i^- - 1).$$

Un moyen simple de calculer la première somme (la somme sur $i$, pour laquelle $y_i = +1$) est de trier et de parcourir les listes données par ordre décroissant de scores (attribués par la fonction $f$) et de calculer les $\kappa_i^+$ de façon

incrémentale, tout comme la somme à laquelle ils contribuent : à chaque fois qu'un exemple négatif est rencontré, on incrémente $\kappa_i^+$ et lorsqu'un exemple positif est visité, $\kappa_i^+(\kappa_i^+ - 1)$ est ajouté à la somme courante. La seconde somme est calculée de manière similaire en parcourant les exemples dans l'ordre inverse puis en incrémentant $\kappa_i^-$ dans le cas d'un exemple positif rencontré et en ajoutant $\kappa_i^-(\kappa_i^- - 1)$ dans le cas d'un exemple négatif visité. L'Algorithme 7 détaille la procédure de calcul. Le coût du calcul de $\hat{\Sigma}_{\ell_f}$ est donc celui du tri des scores, qui s'effectue en temps $\mathcal{O}(n \log n)$. □

**Algorithme 7 :** Calcul efficace de $W(\underline{Z}_n)$, $\mathcal{Y} = \{-1, +1\}$

**Entrée :** $\underline{Z}_n = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ et une fonction d'ordonnancement $f$

**Sortie :** $W(\underline{Z}_n) = \sum_{i:y_i=+1} \kappa_i^+(\kappa_i^+ - 1) + \sum_{i:y_i=-1} \kappa_i^-(\kappa_i^- - 1)$

```
Trier les exemples dans l'ordre croissant selon f
W ← 0
κ⁺, κ⁻ ← 0
pour i = 0 à n faire
  j ← n − i
  si y_i = +1 alors
    W ← W + κ⁺(κ⁺ − 1)
  sinon
    κ⁺ ← κ⁺ + 1
  fin si
  si y_j = −1 alors
    W ← W + κ⁻(κ⁻ − 1)
  sinon
    κ⁻ ← κ⁻ + 1
  fin si
fin pour
```

### 4.3.2 Au-delà de l'ordonnancement bipartite

Nous venons de voir que dans le cas de l'ordonnacement bipartite ($\mathcal{Y} = \{-1, +1\}$), il existait un moyen efficace de calculer les estimations empiriques associées à la fonction $\ell_f$. Nous allons maintenant brièvement présenter une extension des précédents résultats pour le cas où les données peuvent être rangées suivant une liste discrète de valeurs $\mathcal{Y} = \{\gamma_1, \ldots, \gamma_k\}$.

**Proposition 4.3** *Pour tout entier $n > 0$, le calcul des U-statistiques $\hat{U}_{\ell_f}$, $\hat{\sigma}^2_{\ell_f}$ et $\hat{\Sigma}^2_{\ell_f}$ définies ci-dessus (equations (4.17), (4.18) et (4.19)) dans le cadre de l'ordonnancement multipartite peut être effectué en $\mathcal{O}(kn + n \log n)$.*

On remarque que la complexité calculatoire acquise dans la sous-section précédente ($\mathcal{O}(n \log n)$) ne peut être maintenue pour cette problématique. La version modifiée (Algorithme 8) de l'algorithme précédent, adaptée à ce problème plus général, effectue de l'ordre de $\mathcal{O}(kn + n \log n)$ opérations où $k$ représente le nombre de rangs différents pris par les exemples appartenant à $\underline{Z}_n$. Ainsi, tant que $k \leq log(n)$, c'est le deuxième terme qui prime et donc la complexité calculatoire reste en $\mathcal{O}(n \log n)$. On retrouve bien ici la même

**Algorithme 8 :** Calcul efficace de $W_n$, $\mathcal{Y} = \mathbb{R}$

**Entrée :** $\mathbf{Z}_n = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ et une fonction d'ordonnancement $f$
**Sortie :** $W'(\underline{Z}_n)$

```
Trier les exemples dans l'ordre croissant selon f
W' ← 0
pour γ ∈ {γ_1, ..., γ_n} faire
  κ^+, κ^- ← 0
  ∀i, κ_i ← 0
  pour i = 0 à n faire
    si y_i = γ alors
      κ_i ← κ^+
      W' ← W' + κ^+(κ^+ − 1)
    sinon
      si y_i < γ alors
        κ^+ ← κ^+ + 1
      fin si
    fin si
  fin pour
  pour i = n à 0 faire
    si y_i = γ alors
      W' ← W' + κ^-(κ^- − 1) + 2κ^-κ_i
    sinon
      si y_i > γ alors
        κ^- ← κ^- + 1
      fin si
    fin si
  fin pour
fin pour
```

complexité que précédemment pour le cas de l'ordonnacement bipartite où $k = 2$. Dans le cas contraire, c'est le premier terme qui prend le dessus est la complexité est alors de l'ordre de $\mathcal{O}(kn)$ opérations. Dans le cas extrême pour lequel $\mathcal{Y} = \mathbb{R}$, étant donné que l'ensemble $\{Y_1, \ldots, Y_n\}$ est de taille finie, $y \in \{Y_1, \ldots, Y_n\}$ prend au plus $n$ valeurs et donc la complexité de notre algorithme est bornée par $\mathcal{O}(n^2)$. Même dans ce cas de figure, on reste toujours bien en-dessous du nombre d'opérations que nécessite une approche naïve du calcul de $\hat{\sigma}^2_{\ell_f}(\underline{Z}_n)$ et $\hat{\Sigma}^2_{\ell_f}(\underline{Z}_n)$, qui sont respectivement de l'ordre de $\mathcal{O}(n^3)$ et $\mathcal{O}(n^4)$ opérations. Nous ne fournissons ici que les intuitions permettant de démontrer la complexité de l'Algorithme 8.

*Démonstration.* Il est facile de voir que pour une fonction de perte $\ell_f$ comme exprimée en (4.16) on a maintenant (à mettre en parallèle avec l'équation (4.21))

$$\ell_f(X_i,X_j)\ell_f(X_i,X_k)=1 \Leftrightarrow \begin{cases} Y_i \leq Y_j, Y_k \text{ et } f(X_i) \geq f(X_j), f(X_k) \\ \text{ou} \\ Y_j \leq Y_i \leq Y_k \text{ et } f(X_j) \geq f(X_i) \geq f(X_k) \\ \text{ou} \\ Y_i \geq Y_j, Y_k \text{ et } f(X_i) \leq f(X_j), f(X_k). \end{cases} \tag{4.22}$$

On propose donc une version modifiée de l'algorithme présenté dans la sous-section précédente, intégrant la contribution nouvelle de la deuxième ligne de l'équation (4.22). Soit $\{\gamma_1, \ldots, \gamma_k\}$ l'ensemble (de taille $k$) des valeurs distinctes prises par les exemples de $\underline{Z}_n$ et $W'(\underline{Z}_n) := \sum_{i \neq j \neq l} \ell_f(Z_i, Z_j)\ell_f(Z_i, Z_l)$ pour le cas de l'ordonnancement multi-partite. Pour $i$ tel que $y_i = \gamma$, $\gamma \in \{\gamma_1, \ldots, \gamma_k\}$, $\kappa_i^+$ est le nombre d'instances dont la valeur est plus petite que $\gamma$ auxquelles la fonction $f$ a attribué un score plus élevé que celui de l'instance $\boldsymbol{x}_i$. $\kappa_i^-$ désigne le nombre d'instances dont la valeur est plus grande que $\gamma$ auxquelles la fonction $f$ a attribué un score plus faible que celui de l'instance $\boldsymbol{x}_i$. La contribution de l'instance $Z_i$ à $W'(\underline{Z}_n)$ correspond au nombre $\kappa_i^+(\kappa_i^+ - 1)$ de paires ordonnées dont les indices sont dans $\mathcal{E}^+(i)$ plus le nombre $\kappa_i^-(\kappa_i^- - 1)$ de paires ordonnées dont les indices sont dans $\mathcal{E}^-(i)$ plus enfin le nombre $2\kappa_i^+\kappa_i^-$ (cf. l'équation (4.22)). On obtient donc

$$W'(\underline{Z}_n) = \sum_{\gamma} \left( \sum_{i:y_i=\gamma} \kappa_i^+(\kappa_i^+ - 1) + \sum_{i:y_i=\gamma} \kappa_i^-(\kappa_i^- - 1) + \sum_{i:y_i=\gamma} 2\kappa_i^+\kappa_i^- \right).$$

□

## 4.4 Application au ranking bipartite

### 4.4.1 Bornes sur la taille de l'échantillon de test

Une utilisation évidente des inégalités de Bernstein empiriques est l'établissement de bornes impliquant la taille de l'échantillon de test, par exemple dans un but de sélection de modèle. Cela signifie que nous considérons le scénario où nous avons à notre disposition un échantillon $\underline{Z}_n = \{Z_i := (\boldsymbol{x}_i, y_i)\}_{i=1}^n$ que nous partageons entre un ensemble d'apprentissage $\underline{Z}_{\text{train}} := \underline{Z}_{1:n_{\text{train}}}$ de $n_{\text{train}}$ données et un ensemble de test $\underline{Z}_{\text{test}} := \underline{Z}_{n_{\text{train}}+1:n}$ de taille $n_{\text{test}} := n - n_{\text{train}}$. L'ensemble $\underline{Z}_{\text{train}}$ est utilisé pour apprendre un modèle $f$ qui minimise un risque empirique basé sur une U-statistique induisant une perte (telle la perte de mauvais ordonnancement (4.1) ou encore (4.2)). Les données de l'ensemble $\underline{Z}_{\text{test}}$ sont quant à elles utilisées pour donner un intervalle de confiance sur l'erreur de généralisation du modèle $f$ appris. Par exemple, si nous considérons le problème de l'ordonnancement bipartite, alors la perte est $\ell^{\text{rank}}$ et le noyau correspondant est $q_f(Z, Z') = \ell^{\text{rank}}(f, Z, Z')$. Nous avons alors, avec une probabilité au moins $1 - \delta$

$$R_{\ell^{\text{rank}}}(f) \leq \hat{R}_{\ell^{\text{rank}}}(f, \underline{Z}_{\text{test}}) + \sqrt{\frac{4\hat{\Sigma}^2_{q_f}(\underline{Z}_{\text{test}})}{n_{\text{test}}} \ln\left(\frac{4}{\delta}\right)} + \frac{10}{n_{\text{test}}} \ln\left(\frac{4}{\delta}\right),$$

où $\hat{\Sigma}^2_{q_f}(\underline{Z}_{\text{test}})$ est bien entendu la variance empirique de $q_f$ calculée sur $\underline{Z}_{\text{test}}$.

La Figure 4.2 expose le comportement de telles bornes sur la taille de l'échantillon de test lorsque $n_{\text{test}}$ grandit pour l'ensemble de données « banana » de l'UCI (présenté sur la gauche). Pour réaliser le graphique, nous avons procédé de la manière suivante. Nous avons appris une fonction de

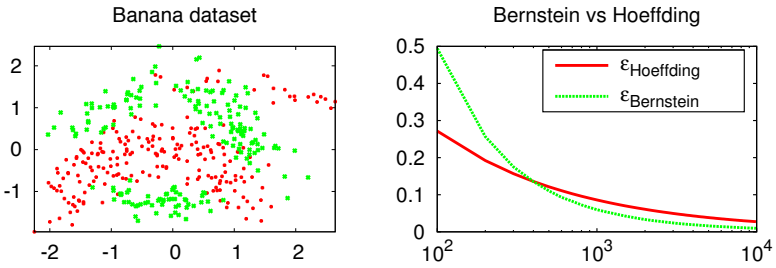



FIGURE 4.2 – *Gauche : ensemble de données « banana » de l'UCI. Droite : moitié de la taille de l'intervalle de confiance de la borne de Hoeffding et de la borne empirique de type Bernstein en fonction de* $n_{test}$.

score linéaire $f(\cdot) = \langle \mathbf{w}, \cdot \rangle$ minimisant

$$\lambda \|\mathbf{w}\|_2^2 + \sum_{i \neq j} \left(1 - (y_i - y_j)\langle \mathbf{w}, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle\right)^2 ,$$

pour $\lambda = 1.0$. Bien sûr, une fonction de score purement linéaire n'aurait pas pu obtenir une bonne précision d'ordonnancement (puisque, comme on peut le voir sur le graphique de gauche, l'ensemble n'est pas linéairement séparable). Nous avons donc travaillé avec l'espace de Hilbert à noyau reproduisant associé au noyau Gaussien $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 / 2)$. Nous avons appris notre fonction de score sur $n_{\text{train}} = 1000$ points et évalué les bornes sur les ensembles de test composés de $n_{\text{test}} = 100, 500, 1000, 5000, 10000$ points. La Figure 4.2 (graphique de droite) rapporte la taille de la moitié de l'intervalle de confiance pour la borne de Hoeffding et pour la borne empirique de type Bernstein. De la même manière que dans la situation décrite dans l'Exemple 4.2, l'utilisation de l'information empirique obtenue sur la variance donne des intervalles de confiance plus petits, même pour des échantillons de test de taille modérée.

### 4.4.2 Ordonnancement en ligne et algorithmes empiriques de course

Une autre application que nous aimerions décrire est l'ordonnancement bipartite « en ligne ». Nous ne présenterons ici que les idées principales concernant la manière dont nos inégalités de concentration empiriques peuvent être utiles dans ce scénario ainsi qu'une nouvelle amélioration concernant le calcul des estimateurs de variance dans ce cas.

Dans un premier temps, nous allons préciser ce que nous appelons ordonnancement bipartite en ligne. Evidemment, cela signifie que $\mathcal{Y} = \{-1, +1\}$ et que la perte qui nous intéresse est $\ell^{ord}$. De plus, cela implique qu'étant donné un ensemble d'apprentissage $\underline{Z}_n = \{Z_i := (\boldsymbol{x}_i, y_i)\}_{i=1}^n$, la procédure d'apprentissage traitera les données de $\underline{Z}_n$ de façon incrémentale en donnant lieu aux hypothèses $f_1, f_2, \ldots, f_T$. Comme la perte $\ell^{ord}$ implique l'utilisation d'un noyau d'ordre $m = 2$, nous supposons que $n = 2T$ et que nous traitons les données de $\underline{Z}_n$ paire par paire. En d'autres termes, le couple $(Z_1, Z_2)$ est utilisé pour apprendre $f_1$, puis le couple $(Z_3, Z_4)$ et $f_1$ sont utilisés pour apprendre $f_2$ et, plus généralement, $(Z_{2t-1}, Z_{2t})$ et $f_{t-1}$ sont utilisés pour produire $f_t$ (il existe des manières plus élaborées pour manipuler les données

mais cela sort légèrement du cadre de cette application). Nous ne spécifions pas d'algorithme d'apprentissage particulier mais on peut imaginer que nous essayons de minimiser un risque empirique basé sur la perte surrogée $\ell^{sur}$ : si des fonctions linéaires $f(\cdot) = \langle \mathbf{w}, \cdot \rangle$ sont considérées et qu'un terme de pénalisation du type $\|\mathbf{w}\|_2^2$ est utilisé alors le problème d'optimisation à résoudre est de la forme

$$\min_{\mathbf{w}} \lambda \|\mathbf{w}\|_2^2 + \sum_{i \neq j} \left(1 - (y_i - y_j)\langle \mathbf{w}, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle\right)^2 ,$$

et des formules de mise à jour simple pour l'inversion incrémentale de matrices fournissent aisément un moyen de résoudre ce problème quand une nouvelle donnée est prête à être traitée (c'est la raison pour laquelle nous avons mentionné cette fonction surrogée).

Comme évoqué par Cesa-Bianchi et al. [2004], une particularité plaisante de l'apprentissage en ligne est le fait que le risque de l'hypothèse $f_t$ peut être estimé sur les $n - 2t$ exemples de $\underline{Z}_n$ sur lesquels elle n'a pas été apprise. A savoir, quand $2\tau$ données ont été traitées, il existe $\tau$ hypothèses $f_1, \ldots, f_\tau$ et, pour $t < \tau$, avec une probabilité au moins $1 - \delta$ :

$$\left| R_{\ell^{ord}}(f_t) - \hat{R}_{\ell^{ord}}(f_t, \underline{Z}_{2t:2\tau}) \right| \leq \sqrt{\frac{2\hat{\Sigma}^2_{q_f}(\underline{Z}_{2t:2\tau})}{\tau - t} \ln\left(\frac{4}{\delta}\right) + \frac{5}{\tau - t} \ln\left(\frac{4}{\delta}\right)} .$$

Si l'on souhaite que ces intervalles de confiance existent simultanément pour tout $t$ et pour tout $\tau$ avec une probabilité $1 - \delta$, des manoeuvres simples afin de calculer le nombre de paires $(t, \tau)$, avec $1 \leq t < \tau \leq T$ montrent qu'il suffit de modifier $\delta$ en $4\delta/(T+1)^2$ et d'avoir recours à la borne de l'union (Proposition 4.1). On obtient alors que, avec une probabilité au moins $1 - \delta$, $\forall 1 \leq t < \tau \leq T$ :

$$\left| R_{\ell^{ord}}(f_t) - \hat{R}_{\ell^{ord}}(f_t, \underline{Z}_{2t:2\tau}) \right| \leq \sqrt{\frac{4\hat{\Sigma}^2_{q_f}(\underline{Z}_{2t:2\tau})}{\tau - t} \ln\left(\frac{T+1}{\delta}\right) + \frac{10}{\tau - t} \ln\left(\frac{T+1}{\delta}\right)} .$$

Au-delà de ces intervalles de confiance uniformes, nous aimerions attirer l'attention du lecteur sur deux points : le premier a à voir avec des considérations statistiques alors que le second est lié à des considérations plus algorithmiques.
Premièrement, pendant que le processus d'apprentissage suit son cours, il est possible de rejeter certaines hypothèses $f_t$ pour lesquelles la borne inférieure sur $R_{\ell^{ord}}(f_t)$, exposée à partir des intervalles de confiance, est plus grande que la borne supérieure donnée par un intervalle de confiance sur $R_{\ell^{ord}}(f_{t'})$ pour une autre hypothèse $f_{t'}$. Cette application correspond à un *algorithme de course*, dont les détails sont décrits dans Maron et Moore [1993]. Une analyse théorique de la pertinence de telles « courses » peut aisément être effectuée grâce aux résultats de Mnih et al. [2008], lesquels traitent des algorithmes de course empiriques de type Bernstein mais pas pour des U-statistiques.
Deuxièmement, il est possible de préserver l'efficacité du calcul de plusieurs estimations de la variance à travers le processus d'apprentissage en ligne : vu que la rapidité du calcul est basée sur des notions de tri, il est possible de

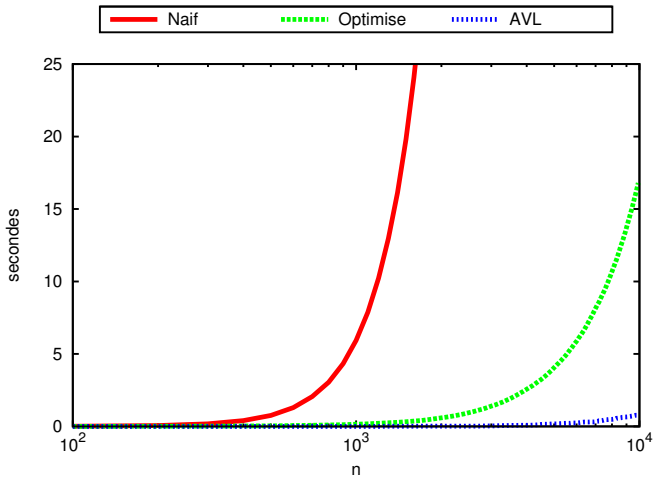



FIGURE 4.3 – *Comparaison du temps de calcul de différentes approches pour le calcul des U-statistiques.*

tirer parti des avantages fournis par structures comme les arbres binaires de recherche et plus particulièrement des arbres AVL [Adelson-Velskii et Landis 1962]. Ces arbres sont précisément conçus pour permettre de conserver et de mettre à jour des listes triées de nombres. Pour chacune des hypothèses apprises au cours du processus en ligne, nous aimerions pouvoir calculer les intervalles de confiance et ainsi pouvoir écarter au fur et à mesure les hypothèses les plus faibles. Il est facile de voir que pour $2\tau$ données traitées, l'hypothèse $f_t$ telle que $t \leq \tau$ utilisera $\tau - t$ exemples afin d'obtenir les intervalles de confiance sur $R_{\ell ord}(f_t)$. La Figure 4.3 montre l'avantage de l'utilisation d'un AVL pour le calcul des estimateurs de variance associés à une hypothèse $f_t$ particulière dans le cas de l'ordonnancement bipartite en ligne. Pour ce faire, nous utilisons un AVL dans lequel chaque noeud contient un exemple $Z_i$ et pour celui-ci les valeurs suivantes : le nombre d'exemples positifs moins bien classés par $f_t$ et le nombre d'exemples négatifs mieux classés par $f_t$. Les trois méthodes comparées dans la Figure 4.3 sont (i) le calcul complet et naïf des U-statistiques, (ii) le calcul complet utilisant les algorithmes optimisés proposés dans la section précédente et (iii) le calcul en ligne des U-statistiques en utilisant un AVL. Les résultats sont clairement à l'avantage du calcul en ligne des intervalles de confiance pour une hypothèse $f_t$ en utilisant des AVL. La question qui reste en suspens est de savoir s'il est possible d'avoir une structure *partagée* permettant de résumer les listes triées des scores de toutes les hypothèses déjà apprises au temps $t$ (en se rappelant que les scores sont calculés sur les mêmes données) plutôt que d'avoir un arbre AVL par hypothèse apprise.

## 4.5 Conclusion

Dans ce chapitre, nous avons proposé de nouvelles inégalités empiriques de type Bernstein pour les U-statistiques. Elles généralisent les inégalités empiriques de Maurer et Pontil [2009] et Audibert et al. [2007] tout en résultant simplement de l'application à deux reprises de la même inégalité de concentration non-empirique pour les U-statistiques.

Nous avons montré de quelle façon, dans la situation de l'ordonnancement bipartite, la variance empirique peut être calculée bien plus efficacement que par une approche naïve. L'extension de ce résultat à l'ordonnancement n'impliquant qu'une légère augmentation de la complexité de calcul a ensuite été exposée.

Par la suite, nous avons proposé des applications pratiques potentielles, avec des résultats illustratifs pour le cas de bornes d'erreur évaluées sur un échantillon de test dans le domaine de l'ordonnancement bipartite. Nous avons également montré une application de nos résultats pour la sélection de modèle dans le cas de l'ordonnancement bipartite ainsi qu'une solution algorithmique efficace afin de calculer en ligne les U-statistiques mises en jeu.

En plus des extensions dont nous avons parlé dans la section précédente, il serait intéressant d'établir des inégalités empiriques similaires pour d'autres types de statistiques riches comme, par exemple, les statistiques de rang linéaires [Hájek et Sidák 1967]. Enfin, l'utilisation de ces nouvelles inégalités pour l'établissement de nouveaux algorithmes d'apprentissage pour l'ordonnancement (tentant de minimiser non seulement le risque empirique mais aussi l'information empirique sur la variance) est une piste très intéressante.

Dans le chapitre suivant, nous proposons une nouvelle inégalité de concentration empirique du second ordre pour les martingales que nous appliquons à l'apprentissage en ligne. A la manière de ce qui a été effectué dans ce chapitre, celle-ci s'obtient à l'aide d'arguments simples et permet d'envisager de nombreuses applications.

# 5 Inégalités de Bernstein empiriques pour l'apprentissage en ligne

Sommaire


5.1 Préliminaires . . . 93
5.1.1 Martingales . . . 93
5.1.2 Inégalité d'Azuma-Hoeffding . . . 93
5.1.3 Inégalité de Bernstein pour les martingales . . . 94
5.2 Inégalités de Bernstein empiriques pour les martingales . . . 95
5.3 Inégalités de Bernstein empiriques pour l'apprentissage en ligne . . . 98
5.3.1 Estimateur instantané de la variance conditionnelle pour l'apprentissage en ligne . . . 98
5.3.2 Bornes théoriques pour l'apprentissage en ligne . . . 98
5.3.3 Cas d'une fonction de perte convexe . . . 100
5.4 Elaboration d'un algorithme d'apprentissage en ligne . . . 101
5.4.1 Pegasos . . . 102
5.4.2 Une nouvelle règle de mise à jour . . . 103
5.4.3 Utilisation d'un noyau de Mercer . . . 105
5.5 Résultats expérimentaux . . . 107
5.6 Conclusion . . . 110


Dans ce chapitre, nous nous plaçons dans le paradigme de l'apprentissage en ligne, qui se caractérise par un traitement séquentiel des données. Un processus d'apprentissage en ligne maintient à jour un modèle qui est raffiné à chaque nouvelle donnée (ou petit ensemble de données) observée. Les deux grands domaines d'application de ce type de processus sont l'apprentissage à partir d'un flux de données et le traitement de très grands ensembles de données (plusieurs millions de données) dont la taille met à mal les algorithmes « classiques » (et pour lesquels une solution possible est de considérer les données une à une ou par petits groupes). Dans le premier cas, les données sont produites séquentiellement et parviennent de cette

manière à l'algorithme d'apprentissage responsable de leur traitement. On peut par exemple penser au problème de la détection de spam dans des e-mails. Les e-mails arrivent les uns après les autres et pour chacun, un algorithme de classification va prédire si celui-ci est désirable. Si l'algorithme se trompe sur un message, l'utilisateur va le lui indiquer et le modèle va être mis à jour en conséquence de manière à mieux classer les prochains e-mails. Dans le second cas, le très grand nombre de données peut mettre à mal la puissance de calcul des machines servant à leur traitement, notamment lorsqu'il s'agit d'obtenir un classifieur en un temps raisonnable. Afin de pallier ce problème, on peut mettre en œuvre une approche itérative se concentrant à chaque étape sur un échantillon de données, dans un schéma similaire à celui décrit juste avant. On peut par exemple penser ici aux données commerciales d'un site de vente en ligne qui dispose d'informations provenant de millions d'utilisateurs. Pour pouvoir traiter efficacement les profils de ces utilisateurs et ainsi réagir à temps aux tendances, une approche séquentielle sur de petits ensembles peut s'avérer profitable.

Il n'existe pas de définition formelle et unanimement reconnue d'un processus d'apprentissage en ligne, même dans des travaux de référence comme ceux de Littlestone et al. [1995] ou Shalev-Shwartz [2007]. On le définit en général de la manière suivante. Soit un ensemble de données étiquetées $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$. Nous supposons que ces données sont identiquement et indépendamment distribuées suivant une distribution de probabilité $\mathcal{D}$ inconnue sur $\mathcal{X} \times \mathcal{Y}$. Un algorithme d'apprentissage en ligne travaillant sur l'ensemble $\underline{Z}_n$ produit à partir d'une hypothèse initiale $h_0$ [1] et de la première donnée $(\boldsymbol{x}_1, y_1)$ une nouvelle hypothèse $h_1$. Cette nouvelle hypothèse est une fonction de la variable aléatoire $z_1 = (\boldsymbol{x}_1, y_1)$ (et de l'hypothèse $h_0$). L'algorithme utilise ensuite l'exemple $(\boldsymbol{x}_2, y_2)$ suivant ainsi que l'hypothèse $h_1$ afin de générer la deuxième hypothèse $h_2$ et ainsi de suite. A la fin du processus d'apprentissage, l'algorithme produit l'ensemble $\{h_0, \ldots, h_n\}$ où chaque hypothèse $h_t$ est construite en utilisant l'hypothèse $h_{t-1}$ la précédant et l'exemple $(\boldsymbol{x}_t, y_t)$. Ainsi, chaque hypothèse $h_t$ dépend de la suite de variables aléatoires $\{z_1, \ldots, z_t\}$ et son risque, noté $R(h_t) = \mathbb{E}\left[\ell(h_t(X), Y) \,|\, z_1, \ldots, z_t\right]$ est l'espérance de la fonction de perte [2] $\ell$ conditionnellement à ces variables. Cette quantité est bien évidemment inconnue. Dans la suite de ce chapitre, nous considérons que la fonction de perte $\ell$ est telle que $\ell \in [0,1]^{\tilde{\mathcal{Y}} \times \mathcal{Y}}$. Il est important de noter que cela ne limite en rien l'applicabilité des résultats présentés ici.

Nous souhaitons caractériser le risque moyen

$$\frac{1}{n}\sum_{t=0}^{n-1} R(h_t) = \frac{1}{n}\sum_{t=0}^{n-1} \mathbb{E}\left[\ell(h_t(X), Y) \,|\, z_1, \ldots, z_t\right]$$

associé aux hypothèses produites par un processus d'apprentissage en ligne

1. Nous abandonnons la notation $f$ utilisée dans le chapitre précédent pour faire référence aux fonctions de score au profit de la notation $h$ plus courante dans la littérature pour faire référence à la notion d'hypothèse.

2. Nous rappelons que l'on mesure les performances d'une hypothèse $h_t : \mathcal{X} \to \tilde{\mathcal{Y}}$ sur un exemple $(\boldsymbol{x}, y)$ par la quantité $\ell(h_t(\boldsymbol{x}), y)$ où $\ell$ est une fonction de perte $\ell : \tilde{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}^+$ positive et bornée.

en estimant chaque terme de cette somme de manière instantanée. Les dépendances entre les hypothèses $h_t$ qui sont inhérentes à ce type de processus ne permettent pas l'utilisation des inégalités de concentrations pour des sommes de variables aléatoires indépendantes vues jusqu'à présent. Il nous faut donc introduire de nouvelles versions de celles-ci, applicables à une martingale possédant des différences bornées. Ces inégalités ont été largement utilisées par le passé dans des travaux (théoriques) mettant en avant les propriétés des hypothèses apprises en ligne. Les travaux de Cesa-Bianchi et al. [2004], Cesa-Bianchi et Gentile [2008], Kakade et Tewari [2009] sont particulièrement liés aux résultats que nous présentons dans ce chapitre. Ainsi, selon Cesa-Bianchi et Gentile [2008] :

**Proposition 5.1** *Soit $h_0, \ldots, h_{n-1}$ l'ensemble des hypothèses générées par un algorithme d'apprentissage en ligne utilisant la fonction de perte bornée $\ell \in [0,1]^{\hat{\mathcal{Y}} \times \mathcal{Y}}$. Alors, pour tout $0 < \delta \leq 1$,*

$$\mathbb{P}\left[\frac{1}{n}\sum_{t=0}^{n-1} R(h_t) \geq \hat{R}_n + 2\sqrt{\frac{\hat{R}_n}{n}\ln\left(\frac{n\hat{R}_n+3}{\delta}\right)} + \frac{36}{n}\ln\left(\frac{n\hat{R}_n+3}{\delta}\right)\right] \leq \delta, \tag{5.1}$$

*où*

$$\hat{R}_n = \hat{R}_n(\underline{Z}_n) = \frac{1}{n}\sum_{t=0}^{n-1} \ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}). \tag{5.2}$$

Cette dernière quantité introduite, $\hat{R}_n$, sera souvent appelée par la suite *risque instantané moyen*. Elle est au cœur de nombreux travaux sur l'apprentissage en ligne (voir par exemple Cesa-Bianchi et al. [2004]). $\hat{R}_n$ est en effet un estimateur de la moyenne des erreurs réelles de chaque hypothèse $h_t$ produite par l'algorithme d'apprentissage en ligne à l'itération $t$ :

$$\frac{1}{n}\sum_{t=0}^{n-1} \mathbb{E}\left[\ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}) \,|\, z_1, \ldots, z_t\right] = \frac{1}{n}\sum_{t=0}^{n-1} R(h_t).$$

L'adjectif *instantané* provient du fait que $\hat{R}_n$ ne se base que sur l'exemple $(\boldsymbol{x}_{t+1}, y_{t+1})$ se présentant à l'itération suivante pour évaluer le risque de $h_t$. La clé du résultat exposé dans la proposition précédente réside dans l'utilisation d'une inégalité de concentration du second ordre pour les martingales (proposée par Freedman [1975]) appliquée à la somme $V_n$ des variances conditionnelles des pertes associées à chaque hypothèse :

$$V_n = \sum_{t=0}^{n-1} \mathbb{V}\left[\ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}) \,|\, z_1, \ldots, z_t\right].$$

Cette quantité n'étant pas disponible, Cesa-Bianchi et Gentile [2008] proposent une majoration de celle-ci afin d'obtenir leur inégalité. Dans ce chapitre, nous améliorons cette borne en employant un estimateur instantané $\hat{V}_n$ de la somme $V_n$, qui permet un contrôle plus fin de cette dernière. Sous les mêmes hypothèses que celles formulées à la Proposition 5.1 nous montrons que

$$\mathbb{P}\left[\frac{1}{n}\sum_{t=0}^{n-1} R(h_t) \geq \hat{R}_n + \frac{1}{n}\sqrt{\beta_n \ln\left(\frac{2}{\delta}\right)} + \frac{2}{3n}\ln\left(\frac{2}{\delta}\right)\right] \leq \delta, \tag{5.3}$$

où $\beta_n$ est une fonction de $\hat{V}_n$ que nous expliciterons par la suite. Ce résultat provient d'une nouvelle inégalité empirique de type Bernstein pour les martingales que nous proposons, et dont le domaine d'application s'étend au-delà de l'apprentissage en ligne.

Nous détaillons ce résultat et ses conséquences dans le reste de ce chapitre qui est organisé comme suit. Dans la Section 5.1 nous rappelons quelques notions essentielles sur les martingales et nous présentons les inégalités de concentration classiques associées à ce type de processus aléatoire. La Section 5.2 présente le premier résultat de ce chapitre, une inégalité tirant parti d'une information empirique de second ordre pour les martingales. Celle-ci est ensuite appliquée dans la Section 5.3 pour obtenir une borne sur l'erreur en généralisation moyenne des hypothèses apprises au cours d'un processus d'apprentissage en ligne. Cette borne améliore de façon substantielle la qualité des résultats cités précédemment. Ensuite, nous présentons en Section 5.4 un nouvel algorithme d'apprentissage en ligne découlant directement de ce résultat théorique. Nous clôturons ce chapitre par quelques résultats expérimentaux issus de l'application de notre inégalité de concentration et de l'algorithme qui en découle sur un problème d'apprentissage en ligne « jouet ».

## 5.1 Préliminaires

### 5.1.1 Martingales

La théorie des martingales fit son apparition dans le domaine des jeux de hasard. Le terme martingale désigne pour beaucoup une technique mise en œuvre afin de gagner de manière « sûre » à un jeu de hasard. Pour les mathématiciens, il désigne un type de processus aléatoire dont la théorie s'étend bien au-delà de ses origines. Nous rappelons dans un premier temps quelques notions élémentaires sur les martingales en nous basant sur l'introduction faite par Steele [2001].

**Définition 5.1** **(Martingales)** *On dit qu'une séquence de variables aléatoires* $\{M_n : 0 \leq n < \infty\}$ *est une* martingale *par rapport à la séquence de variables aléatoires* $\{X_n : 1 \leq n < \infty\}$ *si la séquence* $\{M_0, \ldots, M_n\}$ *répond à deux conditions élémentaires. La première condition est que pour tout entier* $n \geq 1$ *il existe une fonction* $f_n : \mathbb{R}^n \to \mathbb{R}$ *telle que* $M_n = f_n(X_1, X_2, \ldots, X_n)$*. La seconde condition est que la séquence* $\{M_n\}$ *doit satisfaire les propriétés suivantes pour tout* $n \geq 1$

$$\mathbb{E}\left[|M_n|\right] < \infty \tag{5.4}$$
$$\mathbb{E}\left[M_n \,|\, X_1, \ldots, X_{n-1}\right] = M_{n-1}. \tag{5.5}$$

En d'autres termes, une martingale désigne un processus stochastique tel que l'espérance de la prochaine observation, conditionnellement à toutes les observations antérieures, est égale à la dernière observation.

**Remarque 5.1** *En utilisant la linéarité de l'espérance, la condition* (5.5) *peut aussi s'écrire*

$$\mathbb{E}\left[M_{n+1} - M_n \,|\, X_1, \ldots, X_n\right] = \mathbb{E}\left[M_{n+1} \,|\, X_1, \ldots, X_n\right] - M_n = 0.$$

Cette remarque permet de mettre en avant l'absence de « gain » espéré entre les observations $n$ et $n+1$ et donne une première intuition de la définition suivante.

**Définition 5.2** *On dit qu'une séquence de variables aléatoires* $\{Y_n : 0 \leq n < \infty\}$ *est une* différence d'incréments de martingale *si la séquence* $\{Y_n\}$ *satisfait les propriétés suivantes pour tout* $n \geq 1$

$$\mathbb{E}\left[|Y_n|\right] < \infty \tag{5.6}$$
$$\mathbb{E}\left[Y_n \,|\, Y_1, \ldots, Y_{n-1}\right] = 0. \tag{5.7}$$

Par construction, cela implique que si la suite $\{M_n\}$ est une martingale alors la suite $\{Y_n = M_n - M_{n-1}\}$ est une différence d'incréments de martingale.

### 5.1.2 Inégalité d'Azuma-Hoeffding

L'inégalité d'Azuma-Hoeffding [Hoeffding 1963, Azuma 1967] donne un résultat sur la concentration des valeurs d'une martingale, possédant des différences bornées, autour de sa valeur initiale $M_0$. Nous rappelons ici l'inégalité d'Azuma-Hoeffding appliquée à la somme des différences d'une martingale.

**Théorème 5.1** ***(Inégalité d'Azuma-Hoeffding)*** *Soit $\{M_n\}$ une martingale et $\{Y_n = M_n - M_{n-1}\}$ la différence d'incréments de martingale associée telle que $|Y_i| \leq c_i$ pour $1 \leq i \leq n$. Alors, pour tout $\epsilon > 0$*

$$\mathbb{P}\left[\sum_{i=1}^{n} Y_i \geq \epsilon\right] = \mathbb{P}\left[M_n - M_0 \geq \epsilon\right] \leq \exp\left(-\frac{\epsilon^2}{2\sum_{i=1}^{n} {c_i}^2}\right). \tag{5.8}$$

Ce résultat permet une extension de l'inégalité de Hoeffding (Théorème 1.1) au cas où les variables aléatoires étudiées ne sont pas indépendantes. En effet, on obtient facilement le corollaire suivant :

**Corollaire 5.1** *Soit $X_1, \ldots, X_n$ une suite de variables aléatoires telles que pour $1 \leq i \leq n$ on a $|\mathbb{E}\left[X_i \,|\, X_1, \ldots, X_{i-1}\right] - X_i| \leq c_i$. Posons $S_n = \sum_{i=1}^{n} X_i$, alors pour tout $\epsilon > 0$*

$$\mathbb{P}\left[\sum_{i=1}^{n} \mathbb{E}\left[X_i \,|\, X_1, \ldots, X_{i-1}\right] - S_n \geq \epsilon\right] \leq \exp\left(-\frac{\epsilon^2}{2\sum_{i=1}^{n} {c_i}^2}\right). \tag{5.9}$$

*Démonstration.* Une application directe du Théorème 5.1 sur la différence d'incréments de la martingale $\{Y_n\}$ telle que $Y_i = \mathbb{E}\left[X_i \,|\, X_1, \ldots, X_{i-1}\right] - X_i$ donne le résultat. □

Lorsque les variables aléatoires considérées sont indépendantes, le Corollaire 5.1 correspond alors à l'inégalité de Hoeffding classique. Il existe d'autres versions de cette inégalité, le lecteur intéressé appréciera par exemple le travail de McDiarmid [1989] sur le sujet.

### 5.1.3 Inégalité de Bernstein pour les martingales

L'inégalité que nous reportons dans le lemme suivant est une conséquence de l'inégalité de Bernstein pour les martingales présentée dans [Freedman 1975]. Ce lemme étend l'inégalité classique de Bernstein présentée dans le Théorème 1.2 qui suppose l'indépendance entre les variables aléatoires $X_i$. On s'affranchit de cette condition en considérant la différence d'incréments de martingale $\{Y_n = \mathbb{E}\left[X_n | X_1, \ldots, X_{n-1}\right] - X_n\}$. Ce lemme est essentiel dans notre analyse, tout comme il l'était dans le travail réalisé par Cesa-Bianchi et Gentile [2008].

**Lemme 5.1** ***(Inégalité de Bernstein pour les martingales)*** *Soit $X_1, \ldots, X_n$ une séquence de variables aléatoires telles que $0 \leq X_i \leq 1$. Définissons la différence d'incréments de la martingale $\{Y_n = \mathbb{E}\left[X_n | X_1, \ldots, X_{n-1}\right] - X_n\}$ et notons $V_n$ la somme des variances conditionnelles*

$$V_n = \sum_{t=1}^{n} \mathbb{V}\left[X_n | X_1, \ldots, X_{n-1}\right]. \tag{5.10}$$

*Posons $S_n = \sum_{i=1}^{n} X_i$, alors pour tout $\epsilon, v \geq 0$,*

$$\mathbb{P}\left[\sum_{i=1}^{n} \mathbb{E}\left[X_n | X_1, \ldots, X_{n-1}\right] - S_n \geq \epsilon, V_n \leq v\right] \leq \exp\left(-\frac{\epsilon^2}{2v + 2\epsilon/3}\right). \tag{5.11}$$

## 5.2 Inégalités de Bernstein empiriques pour les martingales

Nous avons vu dans le chapitre précédent l'apport des inégalités de type Bernstein (second ordre) par rapport à leur équivalent du premier ordre. Cependant, la variance conditionnelle étant inconnue en général, il est d'usage de majorer celle-ci pour pouvoir évaluer la borne. De la même manière qu'au chapitre précédent, nous proposons d'utiliser un estimateur de variance conditionnelle en lieu et place de cette majoration. Le résultat escompté est bien entendu un contrôle plus fin de cette propriété du processus aléatoire menant à une inégalité de concentration plus précise. Cette section présente le principal résultat du chapitre. Il s'agit d'une version raffinée de l'inégalité de Bernstein pour les martingales rappelée ci-dessus où la somme des variances conditionnelles est majorée empiriquement en utilisant un estimateur instantané.

Nous utilisons la notation $f_{\{Z_t\}}$ pour désigner une fonction dont l'expression est fixée par la suite de variables aléatoires $\{Z_t\} = \{Z_1, \ldots, Z_t\}$.

**Théorème 5.2** ***(Inégalité de Bernstein empirique pour les martingales)*** *Soit $Z_1, \ldots, Z_n$ une séquence de variables aléatoires issues de la même distribution $\mathcal{D}$ telles que $Z_{t+1}, Z_{t+2}$ sont conditionnellement indépendantes par rapport à $\{Z_t\}$, pour tout $1 \leq t \leq n$. Soit $\{f_{\{Z_t\}}\}_{t=1}^n$ une famille de fonctions à valeurs dans $[0,1]$, notons*

$$\beta_n = n\hat{V}_n + \sqrt{\frac{n}{2}\ln\left(\frac{2}{\delta}\right)}, \tag{5.12}$$

*où*

$$\hat{V}_n = \frac{1}{2n}\sum_{t=1}^{n}\left(f_{\{Z_t\}}(Z_{t+1}) - f_{\{Z_t\}}(Z_{t+2})\right)^2. \tag{5.13}$$

*Alors, pour tout $0 < \delta \leq 1$,*

$$\mathbb{P}\left[\frac{1}{n}\sum_{t=1}^{n}\mathbb{E}\left[f_{\{Z_t\}}(Z) \,|\, Z_1, \ldots, Z_t\right] \geq \frac{1}{n}\sum_{i=1}^{n} f_{\{Z_t\}}(Z_{t+1}) + \frac{1}{n}\sqrt{\beta_n \ln\left(\frac{2}{\delta}\right)} + \frac{2}{3n}\ln\left(\frac{2}{\delta}\right)\right] \leq \delta. \tag{5.14}$$

Pour démontrer ce théorème, le premier résultat dont nous avons besoin est lié à l'estimateur instantané de variance conditionnelle présenté à l'Equation (5.13). Il s'agit d'une inégalité de concentration permettant de quantifier la déviation de celui-ci. Ce résultat s'inscrit dans le paradigme de la *méthode des différences bornées* [McDiarmid 1989].

**Lemme 5.2** ***(Déviation d'un estimateur instantané de variance conditionnelle)*** *Soit $Z_1, \ldots, Z_n$ une séquence de variables aléatoires issues de la même distribution $\mathcal{D}$ telles que $Z_{t+1}, Z_{t+2}$ sont conditionnellement indépendantes par rapport à $\{Z_t\}$, pour tout $1 \leq t \leq n$. Soit $\{f_{\{Z_t\}}\}_{t=1}^n$ une famille*

*de fonctions à valeurs dans* $[0,1]$*, posons*

$$V_n = \sum_{t=1}^{n} \mathbb{V}\left[f_{\{Z_t\}}(Z) \,|\, Z_1, \ldots, Z_t\right], \tag{5.15}$$

$$\hat{V}_n = \frac{1}{2n} \sum_{t=1}^{n} \Big(f_{\{Z_t\}}(Z_{t+1}) - f_{\{Z_t\}}(Z_{t+2})\Big)^2. \tag{5.16}$$

*Alors, pour tout* $0 < \delta \leq 1$*,*

$$\mathbb{P}\left[V_n \geq n\hat{V}_n + \sqrt{\frac{n}{2}\ln\left(\frac{1}{\delta}\right)}\right] \leq \delta. \tag{5.17}$$

*Démonstration. (Lemme 5.2)* Commençons par définir la séquence de variables aléatoires $\{M_n\}$ telle que

$$M_t = \frac{1}{2}\Big(f_{\{Z_t\}}(Z_{t+1}) - f_{\{Z_t\}}(Z_{t+2})\Big)^2,$$

et la différence d'incréments de martingale associée

$$\{A_n = \mathbb{E}\left[M_n \,|\, Z_1, \ldots, Z_n\right] - M_n\}.$$

Remarquons ensuite que

$$\begin{aligned}
\mathbb{E}\left[M_t \,|\, Z_1, \ldots, Z_t\right] &= \frac{1}{2}\,\mathbb{E}\left[\Big(f_{\{Z_t\}}(Z_{t+1}) - f_{\{Z_t\}}(Z_{t+2})\Big)^2 \,|\, Z_1, \ldots, Z_t\right] \\
&= \frac{1}{2}\left(\mathbb{E}\left[f_{\{Z_t\}}(Z_{t+1})^2 \,|\, Z_1, \ldots, Z_t\right] + \mathbb{E}\left[f_{\{Z_t\}}(Z_{t+2})^2 \,|\, Z_1, \ldots, Z_t\right]\right) \\
&\quad - \mathbb{E}\left[f_{\{Z_t\}}(Z_{t+1}) f_{\{Z_t\}}(Z_{t+2}) \,|\, Z_1, \ldots, Z_t\right] \\
&= \mathbb{E}\left[f_{\{Z_t\}}(Z_{t+1})^2 \,|\, Z_1, \ldots, Z_t\right] - \mathbb{E}\left[f_{\{Z_t\}}(Z_{t+1}) \,|\, Z_1, \ldots, Z_t\right]^2 \\
&= \mathbb{V}\left[f_{\{Z_t\}}(Z) \,|\, Z_1, \ldots, Z_t\right],
\end{aligned}$$

où la troisième ligne provient du fait que les $Z_i$ suivent la même distribution et la dernière ligne de l'indépendance conditionnelle des variables $Z_{t+1}$ et $Z_{t+2}$. Il est facile de voir que

$$\frac{1}{n}\sum_{t=1}^{n} A_t = \frac{1}{n}\sum_{t=1}^{n} \mathbb{V}\left[f_{\{Z_t\}}(Z) \,|\, Z_1, \ldots, Z_t\right] - \hat{V}_n.$$

En remarquant que $M_t \in [0, \frac{1}{2}]$ car $f$ prend ses valeurs dans $[0,1]$, nous en déduisons que $\mathbb{E}\left[M_t \,|\, Z_1, \ldots, Z_t\right] \in [0, \frac{1}{2}]$ et donc que chaque terme de la suite $\{A_n\}$ est borné :

$$-\frac{1}{2} \leq A_t \leq \frac{1}{2}.$$

$\{A_n\}$ est donc une différence d'incréments de martingale bornée. Nous lui appliquons l'inégalité d'Azuma-Hoeffding (Th. 5.1) et nous obtenons

$$\mathbb{P}\left[\frac{1}{n}\sum_{t=1}^{n} \mathbb{V}\left[f_{\{Z_t\}}(Z) \,|\, Z_1, \ldots, Z_t\right] - \hat{V}_n \geq \epsilon\right] \leq \exp\left(-\frac{2\epsilon^2}{n}\right).$$

Nous utilisons enfin le Lemme 1.1 de renversement de l'inégalité pour terminer la preuve. □

Grâce à ce premier résultat, nous sommes maintenant en mesure de prouver le Théorème 5.2.

*Démonstration. (Théorème 5.2)* Définissons la séquence de variables aléatoires $\{M_n\}$ telle que

$$M_i = f_{\{Z_i\}}(Z),$$

et la différence d'incréments de martingale associée

$$\{A_n = \mathbb{E}\left[M_n \,|\, Z_1, \ldots, Z_n\right] - M_n\}.$$

Nous remarquons que pour $\beta_n$ tel qu'à l'Equation (5.12) et $s$ fixés

$$\mathbb{P}\left[\sum_{t=1}^{n} A_t \geq s\right] = \mathbb{P}\left[\sum_{t=1}^{n} A_t \geq s, V_n \geq \beta_n\right] + \mathbb{P}\left[\sum_{t=1}^{n} A_t \geq s, V_n < \beta_n\right].$$

Le travail est alors de borner supérieurement les deux parties du membre de droite de l'égalité afin d'obtenir la borne désirée sur le membre de gauche. Remarquons que $\mathbb{P}\left[\sum_{t=1}^{n} A_t \geq s, V_n \geq \beta_n\right] \leq \mathbb{P}\left[V_n \geq \beta_n\right]$. On utilise alors le Lemme 5.2 pour borner $\mathbb{P}\left[V_n \geq \beta_n\right]$ et obtenir

$$\mathbb{P}\left[\sum_{t=1}^{n} A_t \geq s, V_n \geq \beta_n\right] \leq \frac{\delta}{2}. \tag{5.18}$$

Ensuite, en utilisant l'inégalité de Bernstein pour les martingales du Lemme 5.1 sur la différence d'incréments de la martingale $\{A_n\}$ on obtient

$$\mathbb{P}\left[\sum_{t=1}^{n} A_t \geq s, V_n < b\right] \leq \exp\left(-\frac{s^2}{2b + 2s/3}\right), \tag{5.19}$$

que nous pouvons écrire en utilisant le Lemme de renversement d'inégalité (Lemme 1.1)

$$\mathbb{P}\left[\sum_{t=1}^{n} A_t \geq \sqrt{b \ln\left(\frac{2}{\delta}\right)} + \frac{2}{3}\ln\left(\frac{2}{\delta}\right), V_n < b\right] \leq \frac{\delta}{2}. \tag{5.20}$$

On termine la preuve en posant $b = \beta_n$ dans (5.20) et

$$s = \sqrt{\beta_n \ln\left(\frac{2}{\delta}\right)} + \frac{2}{3}\ln\left(\frac{2}{\delta}\right),$$

dans l'Equation (5.18). □

**Remarque 5.2** *Le Théorème5.2 admet une formulation plus générale. En effet, il est possible de considérer deux séquences de variables aléatoires $\{a_n\}$ et $\{b_n\}$ issues de la même distribution de probabilité $\mathcal{D}$ que les variables $Z_1, \ldots, Z_n$ telles que $a_t$ et $b_t$ sont conditionnellement indépendantes par rapport à $\{Z_t\}$ pour tout $1 \leq t \leq n$ et d'identifier dans l'énoncé $Z_{t+1}$ à $a_t$ et $Z_{t+2}$ à $b_t$.*

Dans la section qui suit, nous appliquons le Théorème 5.2 au cadre de l'apprentissage en ligne. Plus précisément, nous l'utilisons pour caractériser la moyenne des risques

$$\frac{1}{n}\sum_{t=0}^{n-1} R(h_t)$$

associés aux hypothèses apprises au cours d'un tel processus d'apprentissage.

## 5.3 Inégalités de Bernstein empiriques pour l'apprentissage en ligne

Avant d'énoncer le théorème principal de cette section, nous rappelons l'expression de l'estimateur instantané du risque introduit dans le travail de Cesa-Bianchi et al. [2004] avant de définir un estimateur instantané de variance conditionnelle adapté à l'apprentissage en ligne.

### 5.3.1 Estimateur instantané de la variance conditionnelle pour l'apprentissage en ligne

Un algorithme d'apprentissage en ligne travaillant avec l'échantillon $Z_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ de variables aléatoires i.i.d. produit à partir d'une hypothèse initiale $h_0$ l'ensemble d'hypothèses $\{h_0, \ldots, h_n\}$. Pour une fonction de perte $\ell$, on peut définir le *risque instantané moyen* $\hat{R}_n$ comme

$$\hat{R}_n = \frac{1}{n} \sum_{t=0}^{n-1} \ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}). \tag{5.21}$$

L'hypothèse $h_n$ est écartée de cette quantité pour des raisons purement techniques. Chaque terme de cette somme est un estimateur du risque $R(h_t)$ associé à l'hypothèse $h_t$ (conditionnellement aux exemples $z_1, \ldots, z_t$) :

$$\mathbb{E}\left[\ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}) \,|\, z_1, \ldots, z_t\right] = R(h_t).$$

Nous définissons à présent la *variance instantanée moyenne* $\hat{V}_n$ comme

$$\hat{V}_n = \frac{1}{2(n-1)} \sum_{t=0}^{n-2} \Big(\ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}) - \ell(h_t(\boldsymbol{x}_{t+2}), y_{t+2})\Big)^2. \tag{5.22}$$

Nous écartons de cette quantité, à nouveau pour des raisons techniques, les hypothèses $h_{n-1}$ et $h_n$. Chaque terme de cette somme est un estimateur de la variance conditionnelle de $\ell(h_t(\boldsymbol{x}), y)$ :

$$\begin{aligned} &\mathbb{E}\left[\Big(\ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}) - \ell(h_t(\boldsymbol{x}_{t+2}), y_{t+2})\Big)^2 \,|\, z_1, \ldots, z_t\right] \\ &= 2\mathbb{V}\left[\ell(h_t(\boldsymbol{x}), y) \,|\, z_1, \ldots, z_t\right]. \end{aligned} \tag{5.23}$$

$\hat{V}_n$ peut être évaluée facilement à chaque itération d'un processus d'apprentissage en ligne et joue un rôle essentiel dans le théorème que nous présentons ici. Nous la retrouvons non seulement dans les résultats théoriques que nous montrons mais aussi dans l'algorithme que nous élaborons à partir de ceux-ci.

### 5.3.2 Bornes théoriques pour l'apprentissage en ligne

Dans le théorème suivant, nous utilisons le Théorème 5.2 et les estimateur instantanés $\hat{R}_n$ et $\hat{V}_n$ pour borner $\frac{1}{n}\sum_{t=0}^{n-1} R(h_t)$, la moyenne des risques des hypothèses apprises au cours d'un processus d'apprentissage.

**Théorème 5.3** ***(Inégalité de Bernstein empirique pour l'apprentissage en ligne)*** *Soit $h_0, \dots, h_{n-1}$ l'ensemble des hypothèses générées à partir de l'échantillon $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ de variables aléatoires i.i.d. par un algorithme d'apprentissage en ligne utilisant la fonction de perte bornée $\ell \in [0,1]^{\hat{\mathcal{Y}} \times \mathcal{Y}}$. Alors, pour tout $0 < \delta \leq 1$*

$$\mathbb{P}\left[\frac{1}{n}\sum_{t=0}^{n-1} R(h_t) \geq \hat{R}_n + \frac{1}{n}\sqrt{\beta_n \ln\left(\frac{2}{\delta}\right)} + \frac{2}{3n}\ln\left(\frac{2}{\delta}\right)\right] \leq \delta, \tag{5.24}$$

*où*

$$\beta_n = (n-1)\hat{V}_n + \sqrt{\frac{n-1}{2}\ln\left(\frac{2}{\delta}\right)}. \tag{5.25}$$

Avant de démontrer ce théorème, il est important de comparer cette borne à celle de Cesa-Bianchi et Gentile [2008] rappelée ci-dessous :

$$\mathbb{P}\left[\frac{1}{n}\sum_{t=0}^{n-1} R(h_t) \geq \hat{R}_n + 2\sqrt{\frac{\hat{R}_n}{n}\ln\left(\frac{n\hat{R}_n + 3}{\delta}\right)} + \frac{36}{n}\ln\left(\frac{n\hat{R}_n + 3}{\delta}\right)\right] \leq \delta. \tag{5.26}$$

Une première amélioration portée par notre résultat réside dans la diminution des constantes. Cela est particulièrement appréciable lorsque la borne est calculée pour un petit nombre d'hypothèses (lorsque $n$ est petit, le dernier terme de la borne n'est pas négligeable par rapport aux deux autre). Pour analyser le comportement de notre borne lorsque nous disposons d'un nombre suffisant d'hypothèses pour que ce terme soit négligeable, il nous faut prêter attention au terme

$$\begin{aligned}
\frac{1}{n}\sqrt{\beta_n \ln\left(\frac{2}{\delta}\right)} &= \frac{1}{n}\sqrt{\ln\left(\frac{2}{\delta}\right)\left(\sqrt{\frac{n-1}{2}\ln\left(\frac{2}{\delta}\right)} + (n-1)\hat{V}_n\right)} \\
&\leq \frac{1}{n}\left(\sqrt{\ln\left(\frac{2}{\delta}\right)(n-1)\hat{V}_n} + \sqrt{\ln\left(\frac{2}{\delta}\right)\sqrt{\frac{n-1}{2}\ln\left(\frac{2}{\delta}\right)}}\right) \\
&\leq \frac{1}{n}\left(\sqrt{\ln\left(\frac{2}{\delta}\right)n\hat{V}_n} + \sqrt{\ln\left(\frac{2}{\delta}\right)\sqrt{n\ln\left(\frac{2}{\delta}\right)}}\right) \\
&= \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)\hat{V}_n}{n}} + \left(\frac{\ln\left(\frac{2}{\delta}\right)}{n}\right)^{3/4}.
\end{aligned}$$

La majoration effectuée ici n'est pas la plus précise mais elle permet de pouvoir comparer les bornes plus aisément. Ainsi, en négligeant les termes constants en $\ln\left(\frac{2}{\delta}\right)$, notre borne tend vers $\hat{R}_n$ au moins en

$$\mathcal{O}\left(\sqrt{\frac{\hat{V}_n}{n}} + \frac{1}{n^{3/4}} + \frac{1}{n}\right),$$

quand la borne de Cesa-Bianchi et Gentile [2008] tend vers $\hat{R}_n$ en

$$\mathcal{O}\left(\sqrt{\hat{R}_n\frac{\ln(n\hat{R}_n)}{n}} + \frac{\ln(n\hat{R}_n)}{n}\right).$$

Nous supposons que l'erreur commise par chaque hypothèse $h_t$ sur la donnée $z_{t+2}$ n'est pas très différente de l'erreur commise par cette même hypothèse sur la donnée $z_{t+1}$ :

$$\ell(h_t(\boldsymbol{x}_{t+2}), y_{t+2}) \approx \ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}).$$

Dans ce cas,

$$\begin{aligned}
\hat{V}_n &= \frac{1}{2(n-1)} \sum_{t=0}^{n-2} \Big(\ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}) - \ell(h_t(\boldsymbol{x}_{t+2}), y_{t+2})\Big)^2 \\
&\leq \frac{1}{2(n-1)} \left(\sum_{t=0}^{n-2} \ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1})^2 + \sum_{t=0}^{n-2} \ell(h_t(\boldsymbol{x}_{t+2}), y_{t+2})^2\right) \\
&\leq \frac{1}{2(n-1)} \left(\sum_{t=0}^{n-2} \ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}) + \sum_{t=0}^{n-2} \ell(h_t(\boldsymbol{x}_{t+2}), y_{t+2})\right) \\
&\approx \frac{1}{(n-1)} \sum_{t=0}^{n-2} \ell(h_t(\boldsymbol{x}_{t+1}), y_{t+1}) \\
&\approx \hat{R}_n \,.
\end{aligned}$$

La troisième ligne est obtenue en utilisant $\ell \in [0,1]^{\tilde{\mathcal{Y}} \times \mathcal{Y}}$. Un cas de figure évoqué par Cesa-Bianchi et Gentile [2008] est celui où le risque empirique cumulé $n\hat{R}_n$ est en $\mathcal{O}(1)$, c'est-à-dire borné. Leur borne atteint dans ce cas un comportement asymptotique en $\mathcal{O}(\frac{1}{n})$ (les termes en $\ln(n\hat{R}_n)$ peuvent être assimilés à des constantes). Dans l'hypothèse où $\hat{V}_n$ est aussi en $\mathcal{O}(1)$, notre borne affiche un taux de convergence légèrement inférieur en $\mathcal{O}(\frac{1}{n^{3/4}})$. Cependant, dès lors que le risque cumulé $n\hat{R}_n$ croît avec $n$, la borne de Cesa-Bianchi et Gentile [2008] converge avec un taux en $\mathcal{O}(\sqrt{\ln n / n})$ pendant que la nôtre se comporte en $\mathcal{O}(\sqrt{1/n})$. Nous démontrons maintenant le Théorème 5.3.

*Démonstration. (Théorème 5.3)* La démonstration de ce théorème est directe. Considérons l'échantillon de variables aléatoires i.i.d. $\underline{Z}_n = \{z_i\}_{i=1}^n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ et la famille de fonctions $\{\ell(h_i(\cdot), \cdot)\}_{i=0}^n$ où chaque fonction $\ell(h_t(\cdot), \cdot)$ ne dépend que des variables $z_1, \ldots, z_t$ par définition de $h_t$. En remarquant que les variables $z_{t+1}, z_{t+2}$ sont indépendantes vis-à-vis des variables $z_1, \ldots, z_t$ (par définition de $\underline{Z}_n$), il suffit d'appliquer le Théorème 5.2 et d'ajuster les indices pour obtenir le résultat. □

### 5.3.3 Cas d'une fonction de perte convexe

On peut à présent utiliser le Théorème 5.3 afin de montrer un résultat sur le risque associé à l'hypothèse moyenne construite en appliquant un algorithme d'apprentissage en ligne utilisant une fonction de perte convexe $\ell$ sur un échantillon $\underline{Z}_n$. Nous commençons par définir l'hypothèse moyenne :

$$\bar{h} = \frac{1}{n} \sum_{t=0}^{n-1} h_t.$$

Dans le cas où l'espace de décision $\tilde{\mathcal{Y}}$ (pour mémoire, $h_t : \mathcal{X} \to \tilde{\mathcal{Y}}$) associé aux classifieurs considérés est convexe alors l'hypothèse $\bar{h}$ appartient à la

même classe de fonction que chacun des $h_t$, $\bar{h} : \mathcal{X} \to \tilde{\mathcal{Y}}$. La construction de l'hypothèse moyenne permet ainsi l'obtention d'un classifieur dont on peut évaluer le risque. Dans le cas contraire, le résultat suivant s'applique au risque du classifieur de Gibbs [McAllester 1999]. Ce classifieur stochastique est obtenu en sélectionnant une hypothèse parmi $h_0, \ldots, h_{n-1}$ pour chaque donnée à classer, selon une distribution de probabilité sur les hypothèses. $R(\bar{h})$ est alors le risque du classifieur de Gibbs pour une distribution uniforme.

**Corollaire 5.2** *Soit $h_0, \ldots, h_{n-1}$ l'ensemble des hypothèses générées par un algorithme d'apprentissage en ligne utilisant la fonction de perte convexe $\ell$ telle que $\ell \in [0,1]^{\tilde{\mathcal{Y}} \times \mathcal{Y}}$. Alors, pour tout $0 < \delta \leq 1$*

$$\mathbb{P}\left[R(\bar{h}) \geq \hat{R}_n + \frac{1}{n}\sqrt{\beta_n \ln\left(\frac{2}{\delta}\right)} + \frac{2}{3n}\ln\left(\frac{2}{\delta}\right)\right] \leq \delta, \tag{5.27}$$

*où*

$$\beta_n = (n-1)\hat{V}_n + \sqrt{\frac{n-1}{2}\ln\left(\frac{2}{\delta}\right)}.$$

*Démonstration.* En utilisant l'inégalité de Jensen et la linéarité de l'espérance, il est facile de montrer que

$$\begin{aligned} R(\bar{h}) &= \mathbb{E}\left[\ell(\frac{1}{n}\sum_{t=0}^{n-1} h_t(X), Y)\right] \\ &\leq \frac{1}{n}\sum_{t=0}^{n-1}\mathbb{E}\left[\ell(h_t(X), Y)\right] \\ &= \frac{1}{n}\sum_{t=0}^{n-1} R(h_t). \end{aligned}$$

Pour terminer la preuve, il suffit de combiner ce résultat avec le Théorème 5.3. □

Le Corollaire 5.2 montre que le contrôle du risque associé à l'hypothèse moyenne passe non seulement par le contrôle du risque instantané empirique mais aussi par le contrôle de la variance instantanée de la fonction de perte considérée. Il fait écho aux travaux de Seldin et al. [2012] sur de nouvelles inégalités de concentration (non empiriques) de type Bernstein pour les martingales dans le cadre PAC-Bayes.

## 5.4 Elaboration d'un algorithme d'apprentissage en ligne

Nous proposons un nouvel algorithme d'apprentissage en ligne motivé par le Théorème 5.2, tout comme les premières inégalités empiriques de type Bernstein [Maurer et Pontil 2009] usant d'un estimateur de variance calculé sur l'ensemble de l'échantillon d'apprentissage avaient motivé de nombreux algorithmes basés sur la minimisation d'une fonction de coût incorporant

celui-ci [Shivaswamy et Jebara 2010; 2011]. Notre travail se base spécifiquement sur l'algorithme Pegasos [Shalev-Shwartz et al. 2011] mais il est important de noter que la méthode appliquée ici s'étend facilement à d'autres algorithmes d'apprentissage en ligne.

### 5.4.1 Pegasos

Pegasos n'est pas présenté par ses auteurs comme un algorithme d'apprentissage en ligne, pourtant il en possède certaines caractéristiques. Il agit comme une descente stochastique de sous-gradient sur le problème d'optimisation associé aux Machines à Vecteur de Support (SVM, Cristianini et Shawe-Taylor [2000] par exemple) en utilisant un ou plusieurs exemples sélectionnés aléatoirement à chaque itération. Le lien entre apprentissage en ligne et optimisation stochastique a déjà été exhibé dans les travaux de Hu et al. [2009]. Nous rappelons ici l'algorithme Pegasos et présentons ses connexions avec l'apprentissage en ligne.

Etant donné un échantillon d'apprentissage $\underline{Z}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, le problème d'optimisation associé aux SVM est un problème d'optimisation quadratique sous contraintes. Cependant, dans sa version naïve, il s'écrit comme un problème de minimisation du risque empirique avec un terme de régularisation sur la norme du classifieur linéaire $h(\cdot) = \langle \mathbf{w}, \cdot \rangle$ recherché :

$$\min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|_2^2 + \frac{1}{n} \sum_{i=1}^{n} \ell_{hinge}(\mathbf{w}, \boldsymbol{x}_i, y_i), \tag{5.28}$$

avec

$$\ell_{hinge}(\mathbf{w}, \boldsymbol{x}, y) = \max\{0, 1 - y\langle \mathbf{w}, \boldsymbol{x} \rangle\}. \tag{5.29}$$

Dorénavant, puisque la fonction $h$ est entièrement paramétrée par le vecteur $\mathbf{w}$, nous identifions celle-ci à $\mathbf{w}$ pour désigner les classifieurs appris. Pegasos, dans sa version de base, sélectionne aléatoirement à chaque itération un exemple $Z_{i_t} = (\boldsymbol{x}_{i_t}, y_{i_t})$ et cherche à minimiser une approximation de la fonction objectif de l'Equation (5.28) basée sur $Z_{i_t}$ :

$$f(\mathbf{w}^t, Z_{i_t}) = \frac{\lambda}{2} \left\|\mathbf{w}^t\right\|_2^2 + \ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{i_t}, y_{i_t}). \tag{5.30}$$

L'algorithme considère alors le sous-gradient suivant, pris en $\mathbf{w}_t$ de la fonction objectif précédente donné par

$$\nabla_t = \nabla_{\mathbf{w}^t} f(\mathbf{w}^t, Z_{i_t}) = \lambda \mathbf{w}^t - \mathbb{1}_{\left[y_{i_t}\langle \mathbf{w}^t, \boldsymbol{x}_{i_t} \rangle < 1\right]} y_{i_t} \boldsymbol{x}_{i_t}, \tag{5.31}$$

et met à jour $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta_t \nabla_t$ en utilisant un pas $\eta_t = 1/(\lambda t)$. On obtient alors le vecteur

$$\mathbf{w}^{t+1} \leftarrow \left(1 - \frac{1}{t}\right) \mathbf{w}^t + \eta_t \mathbb{1}_{\left[y_{i_t}\langle \mathbf{w}^t, \boldsymbol{x}_{i_t} \rangle < 1\right]} y_{i_t} \boldsymbol{x}_{i_t}. \tag{5.32}$$

Une étape de projection (facultative) dont nous donnons le détail par la suite vient terminer l'itération courante. L'algorithme s'arrête lorsque $t = T$, où $T$ est un nombre d'itérations donné en paramètre. Ainsi, Pegasos peut être vu comme un algorithme d'apprentissage en ligne utilisant la suite d'exemples $Z_{i_1}, \ldots, Z_{i_T}$ construite en effectuant un tirage aléatoire avec remise sur l'échantillon $\underline{Z}_n$. Une même instance $Z$ peut donc apparaître plusieurs fois dans la séquence. L'Algorithme 9 en résume les différentes étapes.

**Algorithme 9 :** Pegasos

**Entrée :** $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n, \lambda \geq 0$ et $T \geq 0$
**Sortie :** $\mathbf{w}_{T+1}$
  $\mathbf{w}_1 \leftarrow \mathbf{0}$
  **pour** $t \leftarrow 1$ à $T$ **faire**
    Choisir aléatoirement $i_t \in \{1, \ldots, n\}$
    Définir $\eta_t = \frac{1}{\lambda t}$
    **si** $y_{i_t} \langle \mathbf{w}^t, \boldsymbol{x}_{i_t} \rangle < 1$ **alors**
      $\mathbf{w}^{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}^t + \eta_t y_{i_t} \boldsymbol{x}_{i_t}$
    **sinon**
      $\mathbf{w}^{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}^t$
    **fin si**
    $\mathbf{w}^{t+1} = \min\left[1, \frac{\sqrt{2/\lambda}}{\|\mathbf{w}^{t+1}\|_2}\right] \mathbf{w}^{t+1}$
  **fin pour**

### 5.4.2 Une nouvelle règle de mise à jour

Le Théorème 5.3 nous incite à suivre une nouvelle voie : minimiser au cours des itérations d'un algorithme d'apprentissage en ligne un compromis entre les deux estimateurs instantanés définis précédemment. Cela nous amène alors naturellement à considérer à chaque itération la quantité instantanée, avec $\nu \geq 0$ :

$$\frac{\lambda}{2}\left\|\mathbf{w}^t\right\|_2^2 + \ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+1}, y_{t+1}) + \frac{\nu}{2}\Big[\ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+1}, y_{t+1}) - \ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+2}, y_{t+2})\Big]^2.$$

Pour simplifier les écritures, nous choisissons de consommer les données deux par deux. En nous basant sur l'Equation (5.30), nous cherchons donc à minimiser pour chaque hypothèse $h_t(\cdot) = \langle \mathbf{w}^t, \cdot \rangle$ la fonction $g : \mathcal{H} \times \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ définie par

$$\begin{aligned} g(\mathbf{w}^t, Z_{t+1}, Z_{t+2}) = {} & \frac{1}{2}\Big[\ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+1}, y_{t+1}) + \ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+2}, y_{t+2})\Big] \\ & + \frac{\nu}{2}\Big[\ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+1}, y_{t+1}) - \ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+2}, y_{t+2})\Big]^2 \\ & + \frac{\lambda}{2}\left\|\mathbf{w}^t\right\|_2^2. \end{aligned} \tag{5.33}$$

Nous procédons de la même manière que pour l'algorithme Pegasos et nous différencions cette fonction. Nous commençons par introduire les quantités $\nabla_{1_t}$ et $\nabla_{2_t}$ définies comme

$$\nabla_{1_t} = \frac{\left(\mathbb{1}_{[y_{t+1}\langle \mathbf{w}^t, \boldsymbol{x}_{t+1}\rangle < 1]} y_{t+1}\boldsymbol{x}_{t+1} + \mathbb{1}_{[y_{t+2}\langle \mathbf{w}^t, \boldsymbol{x}_{t+2}\rangle < 1]} y_{t+2}\boldsymbol{x}_{t+2}\right)}{2}$$

et

$$\nabla_{2_t} = \nu\left(\mathbb{1}_{[y_{t+1}\langle \mathbf{w}^t, \boldsymbol{x}_{t+1}\rangle < 1]} y_{t+1}\boldsymbol{x}_{t+1} - \mathbb{1}_{[y_{t+2}\langle \mathbf{w}^t, \boldsymbol{x}_{t+2}\rangle < 1]} y_{t+2}\boldsymbol{x}_{t+2}\right) \cdot \Big[\ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+1}, y_{t+1}) - \ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+2}, y_{t+2})\Big].$$

**Algorithme 10 :** Pegasos-V

**Entrée :** $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, $\lambda, \nu \geq 0$ et $T \geq 0$

$\mathbf{w}_1 \leftarrow \mathbf{0}$

**pour** $t \leftarrow 1$ à $T$ **faire**

Définir $\eta_t = \frac{1}{\lambda t}$

**si** $y_{t+1}\langle \mathbf{w}^t, \boldsymbol{x}_{t+1}\rangle < 1$ **alors**

**si** $y_{t+2}\langle \mathbf{w}^t, \boldsymbol{x}_{t+2}\rangle < 1$ **alors**

$\mathbf{w}^{t+1} \leftarrow (1-\eta_t\lambda)\,\mathbf{w}^t + \eta_t\left(\frac{y_{t+1}\boldsymbol{x}_{t+1}+y_{t+2}\boldsymbol{x}_{t+2}}{2}\right)$
$+\,\eta_t\nu(y_{t+1}\boldsymbol{x}_{t+1} - y_{t+2}\boldsymbol{x}_{t+2})\,(y_{t+2}\langle \mathbf{w}^t, \boldsymbol{x}_{t+2}\rangle - y_{t+1}\langle \mathbf{w}^t, \boldsymbol{x}_{t+1}\rangle)$

**sinon**

$\mathbf{w}^{t+1} \leftarrow (1-\eta_t\lambda)\,\mathbf{w}^t + \eta_t y_{t+1}\boldsymbol{x}_{t+1}\left(\frac{1}{2} + \nu\ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+1}, y_{t+1})\right)$

**fin si**

**sinon**

**si** $y_{t+2}\langle \mathbf{w}^t, \boldsymbol{x}_{t+2}\rangle < 1$ **alors**

$\mathbf{w}^{t+1} \leftarrow (1-\eta_t\lambda)\,\mathbf{w}^t + \eta_t y_{t+2}\boldsymbol{x}_{t+2}\left(\frac{1}{2} + \nu\ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+2}, y_{t+2})\right)$

**sinon**

$\mathbf{w}^{t+1} \leftarrow (1-\eta_t\lambda)\,\mathbf{w}^t$

**fin si**

**fin si**

$\mathbf{w}^{t+1} = \min\left[1, \frac{\sqrt{2/\lambda}}{\|\mathbf{w}^{t+1}\|_2}\right]\,\mathbf{w}^{t+1}$

**fin pour**

Le sous-gradient que nous considérons s'écrit alors dans ce cas

$$\nabla_t = \nabla_{\mathbf{w}^t}\, g(\mathbf{w}^t, Z_{t+1}, Z_{t+2}) = \lambda\mathbf{w}^t - (\nabla_{1_t} + \nabla_{2_t})\,. \tag{5.34}$$

En reprenant la mise à jour $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta_t\nabla_t$ avec le pas $\eta_t = 1/(\lambda t)$, nous obtenons alors le vecteur

$$\mathbf{w}^{t+1} \leftarrow \left(1 - \frac{1}{t}\right)\mathbf{w}^t + \eta_t\,(\nabla_{1_t} + \nabla_{2_t})\,. \tag{5.35}$$

Une étape de projection vient terminer la mise à jour que nous réitérons jusqu'à ce que $t = T$. Nous nommons cette procédure Pegasos-V (abréviation de Pegasos-Variance) et nous récapitulons dans l'Algorithme 10 les étapes énoncées ici.

**Remarque 5.3** ***(Borne sur la fonction de perte)*** *A première vue, rien n'empêche la fonction de perte $\ell_{hinge}$ considérée par Pegasos-V de prendre des valeurs supérieures à 1. Est-il cependant possible de la borner ? En notant $\mathbf{w}^*$ le vecteur optimal, on obtient que*

$$\frac{\lambda}{2}\,\|\mathbf{w}^*\|_2^2 \leq g(\mathbf{w}^*) \leq g(\mathbf{w}_0 = \mathbf{0}) \quad \Rightarrow \quad \|\mathbf{w}^*\|_2 \leq \sqrt{2/\lambda}.$$

*On peut donc restreindre la recherche du minimiseur de la fonction $g$ à la boule de rayon $\sqrt{2/\lambda}$. Pour ce faire, nous ajoutons l'étape de projection suivante à la fin de chaque itération*

$$\mathbf{w}^{t+1} = \min\left[1, \frac{\sqrt{2/\lambda}}{\|\mathbf{w}^{t+1}\|_2}\right]\,\mathbf{w}^{t+1}.$$

*En supposant que les données sont normalisées ($\|\boldsymbol{x}\|_2 \leq 1$), la fonction de perte est bornée par :*

$$\ell_{hinge}(\mathbf{w}, \boldsymbol{x}, y) \leq 1 + \|\mathbf{x}\|_2 \|\mathbf{w}\|_2 \leq 1 + \sqrt{2/\lambda}.$$

*Elle respecte alors l'hypothèse du Théorème 5.3.*

**Remarque 5.4** ***(Choix du paramètre $\nu$)*** *Le paramètre $\nu$ permet de contrôler l'impact de l'estimateur instantané de variance sur la fonction objectif. Dans les expériences qui suivent, nous avons choisi de fixer $\nu = 1/M$ où $M$ est une borne supérieure sur la valeur de la fonction de perte. Ce choix est motivé par le constat suivant : lorsque la fonction de perte est telle que $\ell \in [0, M]$, les mises à jour induites par le terme $\nabla_{1_t}$ peuvent être de l'ordre de $M$ quand celles induites par $\nabla_{2_t}$ sont de l'ordre de $M^2$. Le choix $\nu = 1/M$ permet alors d'équilibrer la contribution des deux termes.*

### 5.4.3 Utilisation d'un noyau de Mercer

Il est facile d'introduire une fonction noyau dans le nouvel algorithme que nous présentons afin de le rendre utilisable sur des jeux de données non linéairement séparables. L'astuce utilisée est toujours la même, nous plongeons les données par le biais d'une application $\Phi$ dans un espace de Hilbert $\mathcal{H}_k$ de dimension supérieure muni d'un produit scalaire tel que $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle = k(\boldsymbol{x}, \boldsymbol{x}')$ où $k$ est appelé noyau. Nous cherchons toujours un classifieur $h$ qui est alors de la forme $h(\cdot) = \langle \mathbf{w}, \Phi(\cdot) \rangle$. Le vecteur $\mathbf{w}$ possède la propriété de pouvoir s'écrire comme une combinaison linéaire des images $\Phi(\boldsymbol{x}_i)$ de nos données dans le nouvel espace considéré (Théorème du représentant, Kimeldorf et Wahba [1971] puis généralisé par Schölkopf et al. [2001])

$$\mathbf{w} = \sum_i \alpha_i \Phi(\boldsymbol{x}_i). \tag{5.36}$$

De cette manière, le produit scalaire dans $\mathcal{H}_k$ entre $\mathbf{w}$ et $\Phi(\boldsymbol{x})$ s'écrit alors

$$\begin{aligned} \langle \mathbf{w}, \Phi(\boldsymbol{x}) \rangle &= \langle \sum_i \alpha_i \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}) \rangle \\ &= \sum_i \alpha_i \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}) \rangle \\ &= \sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}). \end{aligned} \tag{5.37}$$

Le vecteur que nous cherchons maintenant est donc le vecteur $\boldsymbol{\alpha}$ qui caractérise totalement la solution du problème d'optimisation (5.28). Nous adoptons la même stratégie que les auteurs de l'algorithme Pegasos pour ajouter la fonction noyau. Au lieu d'introduire la formulation (5.36) directement dans la fonction objectif et de dériver par rapport à $\boldsymbol{\alpha}$, nous dérivons toujours par rapport à $\mathbf{w}$ et n'utilisons son expression en fonction de $\boldsymbol{\alpha}$ qu'au niveau des mises à jour. On obtient alors les nouvelles équations

$$\boldsymbol{\alpha}^{t+1} \leftarrow \left(1 - \frac{1}{t}\right) \boldsymbol{\alpha}^t \tag{5.38}$$

**Algorithme 11 :** Pegasos-VK

**Entrée :** $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{T}, k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ et $\lambda, \nu \geq 0$

$\boldsymbol{\alpha}_1 \leftarrow \mathbf{0}$

**pour** $t \leftarrow 1$ à $T$ **faire**

Définir $\eta_t = \frac{1}{\lambda t}$

$\boldsymbol{\alpha}_{t+1} \leftarrow (1 - \eta_t \lambda)\, \boldsymbol{\alpha}_t$

**si** $y_{t+1}\langle \mathbf{w}^t, \Phi(\boldsymbol{x}_{t+1})\rangle < 1$ **alors**

**si** $y_{t+2}\langle \mathbf{w}^t, \Phi(\boldsymbol{x}_{t+2})\rangle < 1$ **alors**

$\boldsymbol{\alpha}_{t+1}[\boldsymbol{x}_{t+1}] \leftarrow \eta_t \left(\frac{y_{t+1}}{2} + \nu(y_{t+1}y_{t+2}\langle \mathbf{w}^t, \Phi(\boldsymbol{x}_{t+2})\rangle - \langle \mathbf{w}^t, \Phi(\boldsymbol{x}_{t+1})\rangle)\right)$

$\boldsymbol{\alpha}_{t+1}[\boldsymbol{x}_{t+2}] \leftarrow \eta_t \left(\frac{y_{t+2}}{2} + \nu(y_{t+1}y_{t+2}\langle \mathbf{w}^t, \Phi(\boldsymbol{x}_{t+1})\rangle - \langle \mathbf{w}^t, \Phi(\boldsymbol{x}_{t+2})\rangle)\right)$

**sinon**

$\boldsymbol{\alpha}_{t+1}[\boldsymbol{x}_{t+1}] \leftarrow \eta_t y_{t+1} \left(\frac{1}{2} + \nu(1 - y_{t+1}\langle \mathbf{w}^t, \Phi(\boldsymbol{x}_{t+1})\rangle)\right)$

**fin si**

**sinon**

**si** $y_{t+2}\langle \mathbf{w}^t, \Phi(\boldsymbol{x}_{t+2})\rangle < 1$ **alors**

$\boldsymbol{\alpha}_{t+1}[\boldsymbol{x}_{t+2}] \leftarrow \eta_t y_{t+2} \left(\frac{1}{2} + \nu(1 - y_{t+2}\langle \mathbf{w}^t, \Phi(\boldsymbol{x}_{t+2})\rangle)\right)$

**fin si**

**fin si**

$\boldsymbol{\alpha}_{t+1} = \min\left[1, \frac{\sqrt{2/\lambda}}{\sum_i |\alpha_i|}\right] \boldsymbol{\alpha}_{t+1}$

**fin pour**

et

$$\boldsymbol{\alpha}^{t+1}[\boldsymbol{x}_{t+1}] \leftarrow \mathbb{1}_{[y_{t+1}\langle \mathbf{w}^t, \boldsymbol{x}_{t+1}\rangle < 1]}\ \eta_t \left[\frac{1}{2} + \nu\left(\ell_{t+1} - \ell_{t+2}\right)\right] y_{t+1} \tag{5.39}$$

$$\boldsymbol{\alpha}^{t+1}[\boldsymbol{x}_{t+2}] \leftarrow \mathbb{1}_{[y_{t+2}\langle \mathbf{w}^t, \boldsymbol{x}_{t+2}\rangle < 1]}\ \eta_t \left[\frac{1}{2} + \nu\left(\ell_{t+2} - \ell_{t+1}\right)\right] y_{t+2} \tag{5.40}$$

où

$$\ell_{t+1} = \ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+1}, y_{t+1}) \text{ et } \ell_{t+2} = \ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+2}, y_{t+2}).$$

La notation $\boldsymbol{\alpha}_{t+1}[\boldsymbol{x}_{t+1}]$ désigne le coefficient du vecteur $\boldsymbol{\alpha}^{t+1}$ qui correspond à l'exemple $\boldsymbol{x}_{t+1}$. A ces opérations de mise à jour s'ajoute une étape de projection détaillée dans la remarque qui suit. L'Algorithme 11 présente cette nouvelle version que nous appelons Pegasos-VK (pour Pegasos-Variance Kernel).

**Remarque 5.5** ***(Borne sur la fonction de perte)*** *On suppose ici que $k$ est le noyau Gaussien : $k(\boldsymbol{x}, \boldsymbol{x}') = k_G(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 / 2\sigma^2)$. Nous remarquons que $k_G(\boldsymbol{x}, \boldsymbol{x}') \leq 1$. D'une façon plus générale, il suffit de considérer un noyau borné pour obtenir de manière similaire un majorant de la valeur de la fonction de perte. En suivant le même raisonnement que dans la sous-section précédente, on a $\|\mathbf{w}^*\|_2 \leq \sqrt{2/\lambda}$ et une utilisation de l'inégalité ci-dessus*

*sur la valeur de* $k_G(\boldsymbol{x}, \boldsymbol{x}')$ *donne*

$$\begin{aligned}\|\mathbf{w}\|_2 &= \sqrt{\sum_i \sum_j \alpha_i \alpha_j k_G(\boldsymbol{x}_i, \boldsymbol{x}_j)} \\ &\leq \sqrt{\sum_i \sum_j \alpha_i \alpha_j} \\ &= \sqrt{\left(\sum_i \alpha_i\right)^2} \\ &\leq \sum_i |\alpha_i| = \|\boldsymbol{\alpha}\|_1\end{aligned}$$

*A chaque itération, en projetant le vecteur* $\boldsymbol{\alpha}$ *de la façon suivante*

$$\boldsymbol{\alpha}_{t+1} = \min\left[1, \frac{\sqrt{2/\lambda}}{\sum_i |\alpha_i|}\right] \boldsymbol{\alpha}_{t+1}, \tag{5.41}$$

*on obtient la borne suivante sur la fonction de perte* hinge *:*

$$\begin{aligned}\ell_{hinge}(\mathbf{w}, \boldsymbol{x}, y) &= \max\left[0, 1 - y\langle \mathbf{w}, \Phi(\boldsymbol{x})\rangle\right] \\ &= \max\left[0, 1 - y\langle \sum_i \alpha_i \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x})\rangle\right] \\ &= \max\left[0, 1 - y \sum_i \alpha_i \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x})\rangle\right] \\ &= \max\left[0, 1 - y \sum_i \alpha_i k_G(\boldsymbol{x}_i, \boldsymbol{x})\right] \\ &\leq 1 + \sum_i |\alpha_i|\end{aligned}$$

*La fonction de perte respecte alors les hypothèses du Théorème 5.2 et l'on peut ainsi évaluer le risque des hypothèses apprises au cours des itérations de l'algorithme Pegasos-VK.*

## 5.5 Résultats expérimentaux

Dans cette section, nous souhaitons mettre en avant de manière expérimentale l'apport de l'inégalité de concentration empirique de type Bernstein pour l'apprentissage en ligne ainsi que la pertinence de l'algorithme d'apprentissage qui en découle. Pour cela, nous nous mesurons à la borne de la Proposition 5.1 et nous comparons les performances de Pegasos-V (et Pegasos-VK) par rapport à une version légèrement différente de l'algorithme Pegasos présenté dans la section précédente. Celle-ci minimise à chaque itération une approximation $f'$ du problème d'optimisation SVM basée sur deux exemples, au lieu de n'en considérer qu'un seul, telle que

$$f'(\mathbf{w}^t, (\boldsymbol{x}_{t+1}, y_{t+1}), (\boldsymbol{x}_{t+2}, y_{t+2})) = \frac{\lambda}{2}\left\|\mathbf{w}^t\right\|_2^2 + \frac{1}{2}\sum_{i=1}^{2} \ell_{hinge}(\mathbf{w}^t, \boldsymbol{x}_{t+i}, y_{t+i}).$$

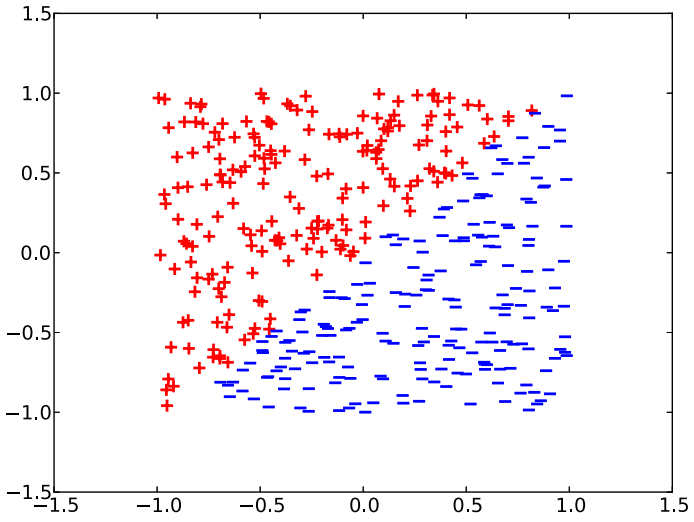



FIGURE 5.1 – *Jeux de données linéairement séparables,* $\boldsymbol{x}_i \in [-1,1]^2$.

**Jeux de données linéairement séparables**

Dans un premier temps, nous mettons en œuvre la version de l'algorithme Pegasos présentée ci-dessus sur un problème jouet et comparons la convergence de l'erreur moyenne des hypothèses dans le cas de données linéairement séparables. Nous générons aléatoirement des vecteurs $x_i \in [-1,1]^2$ auxquels nous assignons la classe $y_i = signe(\langle \mathbf{w}^*, \boldsymbol{x}_i \rangle) \in \{+1,-1\}$ pour un vecteur $\mathbf{w}^* \in [-1,1]^2$ lui aussi généré aléatoirement. La Figure 5.1 illustre un tel jeu de données. Nous travaillons sur un échantillon d'apprentissage composé de 200000 points et reportons dans la Figure 5.2 les valeurs des bornes apparaissant dans la Proposition 5.1 [Cesa-Bianchi et Gentile 2008] et dans le Théorème 5.3 (notre travail) sur l'erreur moyenne des hypothèses calculées pour une confiance de 95% ($\delta = 0.05$). Ces valeurs sont le résultat d'une moyenne sur 20 expériences différentes menées pour plusieurs valeurs du paramètre $\lambda$. On peut voir que notre inégalité surpasse très nettement l'état de l'art au cours des premières itérations, comme cela avait été pressenti lors de la comparaison effectuée en Section 5.2. Cet écart s'amenuise au fil de la procédure mais demeure en notre faveur.

Nous comparons maintenant le comportement de notre borne calculée sur le même échantillon avec l'algorithme Pegasos et avec notre algorithme Pegasos-V cherchant à minimiser l'estimateur instantané de variance. Les valeurs obtenues sont reportées dans la Figure 5.3. On ne note pas de différence notable entre les deux procédures, les deux courbes semblent confondues. En réalité pour $\lambda = 0.01$ et $\lambda = 0.1$, la courbe correspondant à Pegasos-V est très légèrement sous la courbe correspondant à Pegasos. Les autres valeurs de $\lambda$ conduisent au constat inverse. Dans cette configuration, l'ajout d'un estimateur de variance dans la fonction objectif semble avoir un effet limité sur la valeur de l'intervalle de confiance. Cela peut être dû à plusieurs raisons. On peut par exemple penser qu'il n'existe que très peu de différence entre $\hat{V}_n$ et $\hat{R}_n$ pour la fonction de perte considérée ou encore que les termes

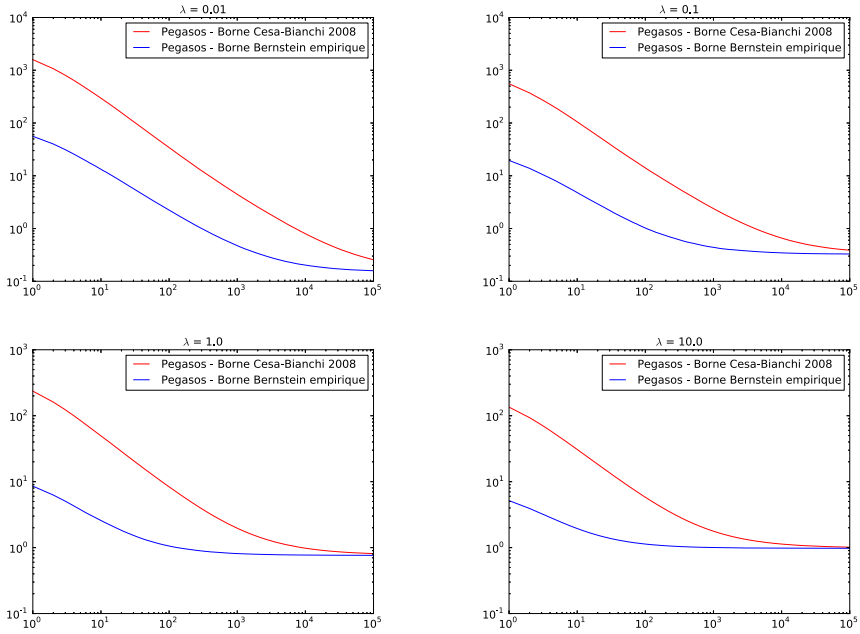



FIGURE 5.2 – *Comparaison des bornes de la Proposition 5.1 et du Théorème 5.2 calculées pour l'algorithme Pegasos appliqué à un jeu de données linéairement séparables.*

constants jouent encore un rôle important et masquent l'apport de l'estimateur de variance. La considération d'une autre fonction de perte amènerait sans doute quelques réponses à ces questions.

### Jeux de données demi-lunes

Pour évaluer les performances de notre algorithme dans sa version avec un noyau (Pegasos-VK), nous constituons un nouveau jeu de données qui ne sont pas linéairement séparables. Ce jeu de données est formé de deux ensembles de vecteurs $x_i \in \mathbb{R}^2$ générés autour de deux demi-cercles en ajoutant un bruit Gaussien. La classe $y_i \in \{+1, -1\}$ est assignée en fonction du demi-cercle utilisé pour générer le vecteur $\boldsymbol{x}_i$. La Figure 5.4 illustre un tel jeu de données. Nous comparons Pegasos-VK à la version de Pegasos utilisant un noyau de Mercer présentée dans Shalev-Shwartz et al. [2011]. Nous appelons cette dernière Pegasos-K. Nous travaillons sur un échantillon d'apprentissage composé de 20000 points et utilisons un noyau Gaussien (voir Eq. (1.10)) dont le paramètre $\sigma$ est fixé par validation-croisée à 0.01. Nous reportons dans la Figure 5.5 les valeurs des bornes considérées dans l'expérience précédente, toujours calculées pour une confiance de 95% ($\delta = 0.05$). Ces valeurs sont le résultat d'une moyenne sur 20 expériences différentes. Ici encore, l'amélioration de la précision de l'inégalité de concentration apportée par notre travail est indéniable. Cependant, l'algorithme incorporant un terme lié à l'estimateur $\hat{V}_n$ que nous avons élaboré peine encore à se détacher de son homologue se focalisant uniquement sur $\hat{R}_n$.

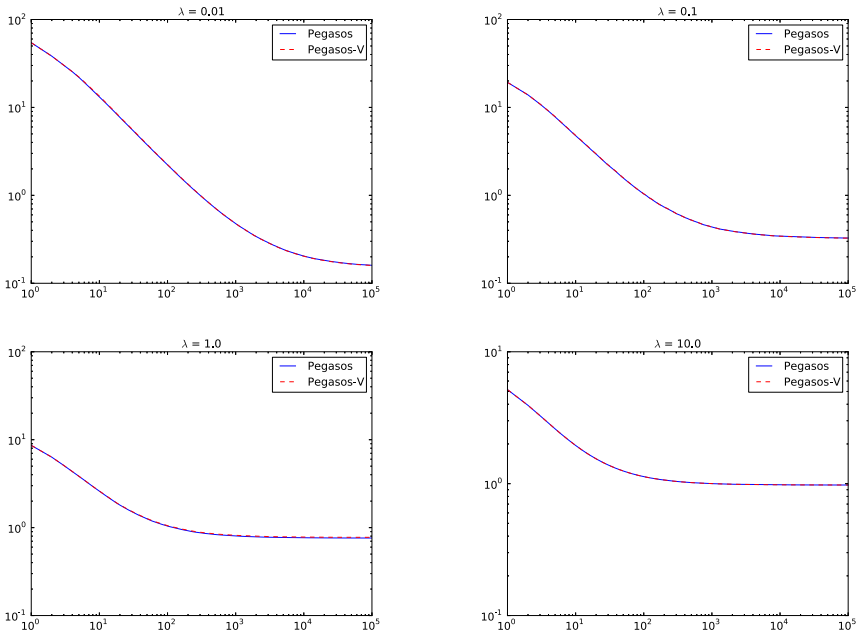



FIGURE 5.3 – *Comparaison de la borne du Théorème 5.2 calculée pour les algorithmes Pegasos et Pegasos-V appliqués à un jeu de données linéairement séparables.*

## 5.6 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle inégalité de concentration empirique de type Bernstein pour les martingales. Nous l'avons utilisée dans le cadre de l'apprentissage en ligne afin de borner le risque moyen des hypothèses apprises au cours d'un tel processus. En introduisant un nouvel estimateur instantané de variance, cette inégalité permet grâce à des arguments simples d'améliorer les résultats existants. La présence dans la borne de cet estimateur nous a amené à considérer un nouvel algorithme d'apprentissage en ligne cherchant à minimiser ce dernier, conjointement à la minimisation du risque empirique. Ce résultat donne ainsi naissance à une nouvelle famille d'algorithmes dont nous avons présenté une instance, Pegasos-V, inspiré de l'algorithme Pegasos [Shalev-Shwartz et al. 2011]. Les expériences menées à la fin de ce chapitre viennent asseoir la qualité de notre inégalité de concentration. Cependant, l'algorithme Pegasos-V (ainsi que son équivalent à noyau Pegasos-VK) ne conduit pas, sur les exemples testés, à une réelle amélioration vis à vis de la version *classique* de Pegasos.

Nous restons pourtant convaincus que notre approche n'est pas vaine, confortés par les résultats encourageants obtenus par des algorithmes minimisant un estimateur de variance dans le cas de l'apprentissage « batch » [par exemple *Sample Variance Penalization* de Maurer et Pontil 2009] même si en pratique le gain apporté par ces algorithmes est parfois limité [*Variance Penalizing AdaBoost*, Shivaswamy et Jebara 2011]. Parmi les pistes envisagées pour tirer parti de notre borne, on peut penser tout d'abord à l'utilisation d'un estimateur de variance plus précis, par exemple en considérant plus de deux données à chaque itération. Au contraire, il pourrait aussi être intéres-

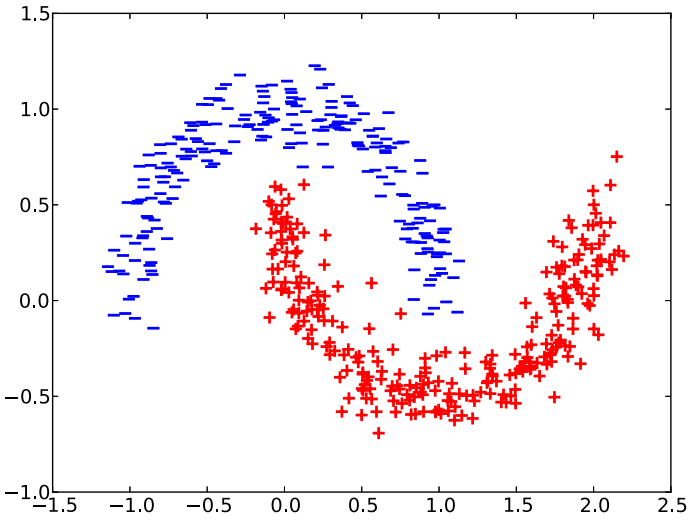



FIGURE 5.4 – *Jeux de données demi-lunes,* $\boldsymbol{x}_i \in \mathbb{R}^2$.

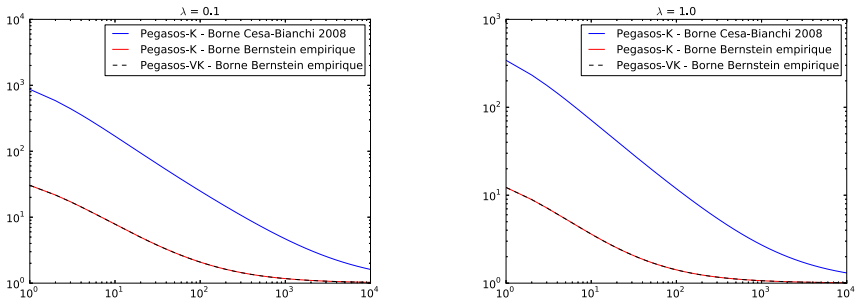



FIGURE 5.5 – *Comparaison des algorithmes Pegasos-K et Pegasos-VK sur le jeu de données demi-lunes.*

sant de réfléchir à une méthode permettant de traiter les données une à une (et non deux par deux comme c'est le cas à présent). Au lieu de considérer l'hypothèse moyenne, une application potentielle de notre inégalité serait de procéder à la sélection, au cours du processus d'apprentissage, de la meilleure hypothèse [Cesa-Bianchi et al. 2004, Cesa-Bianchi et Gentile 2008]. Concernant l'algorithme Pegasos-VK, nous pointons du doigt la décroissance du vecteur de poids $\alpha$ au cours des itérations menant à un nombre grandissant de coefficients non-nuls mais dont la contribution est très petite. L'élagage des coefficients devenus négligeables [Kivinen et al. 2004] mériterait ainsi d'être étudié afin d'apporter une solution à ce constat. Enfin, il pourrait être intéressant de généraliser le chemin que nous avons suivi pour établir l'algorithme Pegasos-V et de modifier d'autres algorithmes d'apprentissage en ligne afin qu'ils prennent en compte l'estimateur instantané de variance.

# Conclusion

Dans ce manuscrit, nous avons abordé deux problématiques distinctes que sont la représentation parcimonieuse de signaux à l'aide d'algorithmes gloutons et l'élaboration de bornes d'erreur en généralisation empiriques du second ordre. Elles possèdent cependant un point commun dans ce document, celui de faire intervenir des inégalités de concentrations statistiques dans leur analyse. Nous revenons de façon synthétique sur les principales contributions de ce document avant d'énoncer quelques-unes des perspectives qui en découlent.

Dans le Chapitre 2, nous avons présenté une nouvelle procédure, *Matching Pursuit avec Sélection Aléatoire (MP-SA)*, permettant de diminuer le temps de calcul nécessaire à la sélection d'un atome dans l'algorithme Matching Pursuit (MP). MP-SA repose sur un double échantillonnage aléatoire du dictionnaire, réalisé à chaque itération, conduisant à la sélection d'un atome potentiellement sous-optimal. Dans la pratique, cette sous-optimalité ne diminue en rien la qualité de l'approximation fournie par MP-SA. Au contraire, le gain de temps réalisé à chaque itération permet, pour un temps de calcul total fixé, d'en effectuer un nombre plus important. Cela aboutit souvent à l'obtention d'une approximation parcimonieuse de meilleure qualité que celle obtenue par MP dans le même temps. Nous avons analysé les propriétés de notre algorithme, notamment en montrant les conditions pour que celui-ci puisse être interprété comme un algorithme de poursuite faible. De plus, notre approche s'étend facilement à la plupart des algorithmes de poursuite basés sur le calcul de corrélations et permet également d'en accélérer l'étape de sélection.

Nos principales contributions issues du Chapitre 3 concernent la représentation parcimonieuse simultanée de plusieurs signaux et son application à la classification multi-classes. Nous avons proposé un algorithme, *Greedy Block Coordinate Descent for Regularized Least Squares (GBCD-RLS)*, ainsi qu'une implémentation efficace de celui-ci pour résoudre ce problème. Ici encore, nous nous sommes attelés à montrer la pertinence de notre approche en en fournissant une analyse théorique. Le point de vue que nous avons adopté, en exhibant le fait que cet algorithme peut être interprété comme une descente de gradient par bloc de coordonnées, nous a permis de proposer un éventail d'extensions possibles de notre procédure suivant la structure de la matrice de coefficients recherchée. En utilisant des codes correcteurs d'erreur pour reformuler un problème d'apprentissage multi-classes, nous avons pu mettre en œuvre notre algorithme sur ce type de problématique pour laquelle nous avons obtenu des performances de classification au niveau de l'état de l'art.

Le Chapitre 4 est construit autour des nouvelles inégalités de concentration empiriques de type Bernstein que nous proposons pour les U-Statistiques. Celles-ci ont été appliquées au cadre de l'ordonnancement et ont permis d'établir une borne sur l'écart entre le risque empirique et le risque réel d'une fonction de score apprise pour résoudre ce type de problèmes. C'est bien évidemment la présence d'un estimateur de variance, calculable empiriquement, qui confère à ces inégalités tout leur attrait. Nous avons ensuite exposé un résultat algorithmique permettant de réduire la complexité du calcul de cet estimateur avant de proposer quelques applications directes de notre résultat.

Le dernier chapitre, le Chapitre 5, a été l'occasion pous nous d'introduire une nouvelle inégalité de concentration empirique de type Bernstein pour les martingales. Celle-ci met à nouveau en jeu un estimateur de variance et donne un résultat sur la concentration de la somme des incréments d'une martingale. Nous avons ensuite utilisé cette inégalité pour borner le risque moyen des hypothèses issues d'un processus d'apprentissage en ligne grâce à l'utilisation d'un estimateur instantané de variance facilement calculable au cours d'un tel processus. Notre borne se révèle plus précise que l'inégalité de référence de Cesa-Bianchi et Gentile [2008], à la fois dans la théorie et en pratique. La présence clé de cet estimateur instantané de variance dans la borne nous a conduit à proposer un nouvel algorithme d'apprentissage en ligne basé sur Pegasos, minimisant un compromis entre le risque empirique instantané et l'estimateur sus-cité. Cet algorithme, pour les données jouets considérées, n'améliore cependant que très peu la qualité moyenne des hypothèses qu'il produit par rapport à son ainé. Cependant, nous espérons que d'autres algorithmes en ligne puissent tirer parti de cet estimateur instantané de variance.

## Perspectives

Avant toute chose, une perspective commune à tous les travaux menés dans cette thèse réside dans la validation de ceux-ci sur de nouvelles tâches, potentiellement de grande(s) dimension(s), afin de conforter leur apport dans la pratique. Nous ne revenons pas ici en détail sur les nombreuses perspectives exposées à la fin de chaque chapitre de ce document. Nous proposons plutôt, pour chacun, une piste de recherche qui nous semble revêtir un intérêt particulier.

### Sélections aléatoires dynamiques

Dans sa thèse, Moussallam [2012] propose d'adapter le processus de sélection aléatoire du sous-dictionnaire au cours des itérations de l'algorithme. Nous pensons également que le fait de guider ce processus de sélection (par exemple en considérant une distribution de probabilité non uniforme sur les atomes) serait profitable. En effet, nous avons pu voir que la qualité de l'approximation fournie par la procédure MP-SA était sensiblement liée à la proportion d'atomes et de coordonnées sélectionnés au cours des itérations. Ainsi, pouvoir ajuster au mieux ces proportions au cours de l'algorithme afin de réduire au maximum le temps de calcul tout en préservant la perti-

nence des atomes sélectionnés semble être une piste intéressante. Cette tâche nécessiterait sans doute pour être menée à bien d'obtenir des estimateurs plus précis que celui que nous proposons (basé sur l'inégalité de Serfling [1974]) concernant les produits scalaires partiels et d'étudier plus en détail l'impact de l'atome sélectionné à chaque itération sur le sous-dictionnaire à considérer à l'itération suivante. Dériver un outil semblable aux *screening tests* [Xiang et Ramadge 2012] éliminant presque sûrement des atomes non-pertinents et pouvant être calculé rapidement à chaque itération est aussi une piste que nous souhaitons explorer.

### Sélection dépendant de la distance

Dans leur travail sur la régression aux moindres carrés régularisée, Naula et al. [2011] proposent de sélectionner un atome en se basant sur un estimateur leave-one-out calculable à faible coût à chaque itération. Cette méthode de sélection vise alors à identifier un atome spécifiquement pertinent pour la tâche de régression visée. Nous pensons que cette approche est digne d'intérêt, la méthode de sélection que nous proposons, basée sur le gradient, est d'ailleurs directement liée à la distance que nous utilisons pour assigner une classe à un code estimé. Appliquer cette idée à de nouvelles distances, c'est-à-dire spécialiser la sélection d'une variable singulière en fonction de la distance utilisée pour l'étape de classification finale, étendrait les possibilités de notre algorithme de classification. La difficulté est ici de construire un estimateur, facile à calculer, permettant de réaliser cette sélection.

### Meta-algorithmes minimisant un compromis risque / variance empirique

Les deux derniers chapitres de ce manuscrit, consacrés aux inégalités empiriques du second ordre, partagent le même message : contrôler la variance de façon empirique au cours du processus d'apprentissage semble être une bonne chose. Nous avons présenté un exemple d'algorithme en ligne qui vise cet objectif mais il serait très intéressant de pouvoir généraliser cette approche en établissant par exemple un meta-algorithme capable d'être instancié aisément à partir d'algorithmes de référence en apprentissage statistique. Il s'agirait ensuite de pouvoir étudier les propriétés de cette famille d'algorithmes, notamment afin de mieux cerner de quelle façon l'estimateur de variance viendrait jouer le rôle d'un terme de régularisation et quelles seraient les propriétés de généralisation associées à ces algorithmes.

Peut-être ce dernier point constituerait-il alors un bémol à l'utilisation excessive d'inégalités de concentration en apprentissage statistique. En effet, l'apport des inégalités de concentration empiriques du second ordre est indéniable pour quantifier le lien entre risque réel et risque empirique mais de là à leur permettre de guider entièrement le processus d'apprentissage il est des questions auxquelles il faut encore apporter des réponses. Nous restons en tous cas convaincus de leur potentiel et sommes curieux d'étudier les nouvelles applications ouvertes par leurs extensions matricielles [Tropp 2011], par exemple pour améliorer l'analyse théorique de l'algorithme MP-SA.

# Bibliographie


G. Adelson-Velskii et E.M. Landis. An algorithm for the organization of information. *Doklady Akademii Nauk SSSR*, 146 :263–266, 1962. (Cité page 87.)

Amir Adler, Valentin Emiya, Maria G. Jafari, Michael Elad, Rémi Gribonval, et Mark D. Plumbley. Audio Inpainting. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3) :922–932, Mars 2012. ISSN 1558-7916. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6020748. (Cité page 44.)

Michal Aharon. *Overcomplete Dictionaries for Sparse Representation of Signals*. PhD thesis, Israel Institute of Technology, 2006. (Cité page 18.)

Erin L. Allwein, Robert Elias Schapire, et Yoram Singer. Reducing Multiclass to Binary : A Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1 :113–141, 2000. URL http://jmlr.org/papers/volume1/allwein00a/allwein00a.pdf. (Cité pages 50 et 53.)

Miguel A. Arcones. A Bernstein-type inequality for U-statistics and U-processes. *Statistics & Probability Letters*, 22(3) :239–247, Février 1995. ISSN 0167-7152. URL http://linkinghub.elsevier.com/retrieve/pii/016771529400072G. (Cité pages 6 et 75.)

Jean-yves Audibert, Rémi Munos, et Csaba Szepesvári. Tuning Bandit Algorithms in Stochastic Environments. Dans *Proceedings of the 18th International Conference on Algorithmic Learning Theory - ALT '07*, pages 150–165, Sendai, Japan, 2007. ISBN 978-3-540-75224-0. URL http://www.springerlink.com/index/10.1007/978-3-540-75225-7_15. (Cité pages 71 et 87.)

Kazuoki Azuma. Weighted Sums of Certain Dependent Random Variables. *Tohoku Mathematical Journal*, 19(3) :357–367, 1967. ISSN 0040-8735. URL http://projecteuclid.org/euclid.tmj/1178243286. (Cité page 93.)

George Bennett. Probability Inequalities for the Sum of Independent Random Variables. *Journal of the American Statistical Association*, 57(297) : 33–45, Mars 1962. ISSN 0162-1459. URL http://www.jstor.org/stable/2282438. (Cité page 14.)

T. Blumensath et Mike E. Davies. Stagewise Weak Gradient Pursuits. *IEEE Transactions on Signal Processing*, 57(11) :4333–4346, Novembre 2009. ISSN 1053-587X. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5109633. (Cité page 25.)

Thomas Blumensath et Mike E. Davies. Gradient Pursuits. *IEEE Transactions on Signal Processing*, 56(6) :2370–2382, Juin 2008. ISSN 1053-587X. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4480155. (Cité pages 23, 25 et 29.)

Bernhard E Boser, Isabelle M Guyon, et Vladimir N Vapnik. A Training Algorithm for Optimal Margin Classifiers. Dans *Proceedings of the 5th Annual Workshop on Computational learning Theory - COLT '92*, pages 144–152, New York, New York, USA, 1992. ACM Press. ISBN 089791497X. URL http://portal.acm.org/citation.cfm?doid=130385.130401. (Cité page 15.)

Stéphane Boucheron, Olivier Bousquet, et Gábor Lugosi. Theory of classification : a survey of some recent advances. *ESAIM : Probability and Statistics*, 9 :323–375, 11 2005. ISSN 1262-3318. URL http://www.esaim-ps.org/article_S1292810005000182. (Cité pages 6, 14 et 70.)

Emmanuel J. Candès, Justin Romberg, et Terence Tao. Robust Uncertainty Principles : Exact Signal Reconstruction From Highly Incomplete Frequency Information. *IEEE Transactions on Information Theory*, 52 (2) :489–509, 2006. ISSN 0018-9448. URL http://authors.library.caltech.edu/4792/1/CANieeetit06.pdf. (Cité pages 24 et 29.)

Nicolò Cesa-Bianchi, Alex Conconi, et Claudio Gentile. On the Generalization Ability of On-Line Learning Algorithms. *IEEE Transactions on Information Theory*, 50(9) :2050–2057, Septembre 2004. ISSN 0018-9448. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1327806. (Cité pages 7, 86, 91, 98 et 111.)

Nicolò Cesa-Bianchi et Claudio Gentile. Improved Risk Tail Bounds for On-Line Algorithms. *IEEE Transactions on Information Theory*, 54(1) :386–390, Janvier 2008. ISSN 0018-9448. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4418464. (Cité pages 7, 91, 94, 99, 100, 108, 111 et 114.)

Jie Chen et Xiaoming Huo. Theoretical Results on Sparse Representations of. *IEEE Transactions on Signal Processing*, 54(12) :4634–4643, 2006. (Cité page 56.)

Stéphan Clémençon, Gábor Lugosi, et Nicolas Vayatis. Ranking and Empirical Minimization of U-Statistics. *The Annals of Statistics*, 36(2) :844–874, Avril 2008. ISSN 0090-5364. URL http://projecteuclid.org/euclid.aos/1205420521. (Cité pages 6, 69 et 70.)

Thomas H. Cormen, Charles E. Leiderson, Ronald L. Rivest, et Stein-Clifford. *Introduction to Algorithms*. The MIT Press, 2009. ISBN 9780262033848. (Cité page 24.)

Corinna Cortes et Vladimir N Vapnik. Support-Vector Networks. *Machine learning*, 20(3) :273–297, 1995. URL http://link.springer.com/article/10.1007/BF00994018. (Cité page 15.)

Shane F. Cotter, Bhaskar D. Rao, Kjersti Engan, et Kenneth Kreutz-Delgado. Sparse Solutions to Linear Inverse Problems With Multiple

Measurement Vectors. *IEEE Transactions on Signal Processing*, 53(7) : 2477–2488, 2005. ISSN 1053-587X. (Cité pages 5 et 56.)

Courtenay Cotton et Daniel P. W. Ellis. Finding Similar Acoustic Events Using Matching Pursuit And Locality-Sensitive Hashing. Dans *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 125–128, New Paltz, 2009. (Cité page 25.)

T. Cover et P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1) :21–27, 1967. ISSN 0018-9448. (Cité page 4.)

Koby Crammer et Yoram Singer. On the Learnability and Design of Output Codes for Multiclass Problems. *Machine Learning*, 47 :201–233, 2002. (Cité page 54.)

Nello Cristianini et John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000. ISBN 9780521780193. URL http://books.google.fr/books?id=B-Y88GdO1yYC. (Cité page 102.)

Geoffrey Davis, Stéphane G. Mallat, et Marco Avellaneda. Adaptive Greedy Approximations. *Constructive Approximation*, 13(1) :57–98, Mars 1997. ISSN 0176-4276. URL http://link.springer.com/10.1007/BF02678430. (Cité page 23.)

Thomas G. Dietterich et Bakiri Ghulum. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Intelligence Research*, 2 :263–286, 1995. (Cité pages 5, 50 et 53.)

David L. Donoho. Compressed Sensing. *IEEE Transactions on Information Theory*, 52(4) :1289–1306, Avril 2006. ISSN 0018-9448. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1614066. (Cité pages 24 et 29.)

David L. Donoho, Yaakov Tsaig, Iddo Drori, et Jean-Luc Starck. Sparse Solution of Underdetermined Systems of Linear Equations by Stagewise Orthogonal Matching Pursuit. *IEEE Transactions on Information Theory*, 58(2) :1094–1121, Février 2012. ISSN 0018-9448. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6145475. (Cité page 25.)

Petros Drineas, Ravi Kannan, et Michael W. Mahoney. Fast Monte Carlo Algorithms for Matrices I : Approximating Matrix Multiplication. *SIAM Journal on Computing*, 36(1) :132–157, Janvier 2006. ISSN 0097-5397. URL http://epubs.siam.org/doi/abs/10.1137/S0097539704442684. (Cité pages 35 et 47.)

Michael Elad et Michal Aharon. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing*, 15(12) :3736–3745, Décembre 2006. (Cité page 24.)

David A. Freedman. On Tail Probabilities for Martingales. *The Annals of Probability*, 3(1) :100 – 118, 1975. URL http://www.jstor.org/stable/10.2307/2959268. (Cité pages 91 et 94.)

Jerome Friedman, Trevor Hastie, et Rob Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1) :1–22, 2010. (Cité page 57.)

Pascal Frossard et Pierre Vandergheynst. Redundancy in non-orthogonal transforms. Dans *Proceedings. 2001 IEEE International Symposium on Information Theory*, page 196. IEEE, 2001. ISBN 0-7803-7123-2. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=936059. (Cité page 32.)

Rémi Gribonval et Morten Nielsen. Approximate Weak Greedy Algorithms. *Advances in Computational Mathematics*, 14(4) :361–378, 2001. ISSN 1019-7168. URL http://link.springer.com/content/pdf/10.1023%2FA%3A1012255021470.pdf. (Cité pages 5, 24, 29 et 31.)

Rémi Gribonval et Pierre Vandergheynst. On the Exponential Convergence of Matching Pursuits in Quasi-Incoherent Dictionaries. *IEEE Transactions on Information Theory*, 52(1) :255–261, Janvier 2006. ISSN 0018-9448. URL http://hal.archives-ouvertes.fr/docs/00/54/49/45/PDF/2006_IEEE_TIT_GribVandergheynst_ExponMP.pdf. (Cité pages 24, 29 et 31.)

Isabelle Guyon et André Elisseeff. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3 :1157–1182, 2003. URL http://machinelearning.wustl.edu/mlpapers/paper_files/GuyonE03.pdf. (Cité page 50.)

J. Hájek et Z. Sidák. *Theory of Rank Tests*. Academic Press, 1967. (Cité page 88.)

James A. Hanley et Barbara J. McNeil. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143 : 29–36, 1982. (Cité page 80.)

W Hoeffding. A Class of Statistics with Asymptotically Normal Distribution. *The Annals of Mathematical Statistics*, 19(3) :293–325, 1948. URL http://projecteuclid.org/euclid.aoms/1177730196. (Cité pages 6, 70, 73 et 74.)

W Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301) :13–30, 1963. URL http://www.jstor.org/stable/2282952. (Cité pages 6, 14, 74, 75 et 93.)

Arthur E Hoerl et Robert W Kennard. Ridge Regression : Applications to Nonorthogonal Problems. *Technometrics*, 12(1) :69–82, 1970. URL http://www.jstor.org/stable/1267352. (Cité page 55.)

Thomas Hofmann, Bernhard Schölkopf, et Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3) :1171–1220, Juin 2008. ISSN 0090-5364. URL http://projecteuclid.org/euclid.aos/1211819561. (Cité page 16.)

Daniel Hsu, Sham M. Kakade, John Langford, et Tong Zhang. Multi-Label Prediction via Compressed Sensing. Dans *Advances in Neural Information Processing Systems 22, NIPS '09*, pages 772–780, 2009. (Cité page 51.)

Chonghai Hu, James T. Kwok, et Weike Pan. Accelerated Gradient Methods for Stochastic Optimization and Online Learning. Dans *Advances in Neural Information Processing Systems 22 - NIPS '09*, pages 781–789, 2009. (Cité page 102.)

Philippe Jost, Pierre Vandergheynst, et Pascal Frossard. Tree-Based Pursuit : Algorithm and Properties. *IEEE Transactions on Signal Processing*, 54(12) :4685–4697, Décembre 2006. ISSN 1053-587X. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4014380. (Cité pages 24 et 29.)

Sham M. Kakade et Ambuj Tewari. On the Generalization Ability of Online Strongly Convex Programming Algorithms. Dans *Advances in Neural Information Processing Systems 21 - NIPS '08*, pages 801–808, 2009. URL http://books.nips.cc/papers/files/nips21/NIPS2008_0290.pdf. (Cité page 91.)

George Kimeldorf et Grace Wahba. Some Results on Tchebycheffian Spline Functions. *Journal of Mathematical Analysis and Applications*, 33(1) : 82–95, 1971. (Cité pages 15 et 105.)

Jyrki Kivinen, Alexander J Smola, et Robert C Williamson. Online Learning with Kernels. *IEEE Transactions on Signal Processing*, 52(8) :2165 – 2176, 2004. ISSN 1053-587X. URL http://eprints.pascal-network.org/archive/00002055/01/KivSmoWil04.pdf. (Cité page 111.)

Ken Lang. NewsWeeder : Learning to Filter Netnews. Dans *Proceedings of the 12th International Conference on Machine Learning Theory - ICML '95*, pages 331–339, 1995. (Cité page 63.)

Dany Leviatan et Vladimir N. Temlyakov. Simultaneous Approximation by Greedy Algorithms. *Advances in Computational Mathematics*, 25(1-3) : 73–90, Juillet 2006. ISSN 1019-7168. URL http://link.springer.com/10.1007/s10444-004-7613-4. (Cité pages 5 et 56.)

N. Lin et R. Xi. Fast Surrogates of U-statistics. *Computational Statistics & Data Analysis*, 54(1) :16–24, Janvier 2010. ISSN 0167-9473. URL http://linkinghub.elsevier.com/retrieve/pii/S0167947309002801. (Cité page 79.)

Nicholas Littlestone, Philip Long, et Manfred Warmuth. On-line Learning of Linear Functions. *Computational Complexity*, 5(1) :1–23, 1995. URL http://users.soe.ucsc.edu/~manfred/pubs/J29.pdf. (Cité page 90.)

Pierre Machart, Thomas Peel, Sandrine Anthoine, Liva Ralaivola, et Hervé Glotin. Stochastic low-rank kernel learning for regression. Dans *Proceedings of the 28th International Conference on Machine Learning - ICML '11*, pages 969–976, 2011a. ISBN 978-1-4503-0619-5. (Cité page 7.)

Pierre Machart, Thomas Peel, Liva Ralaivola, Sandrine Anthoine, et Hervé Glotin. Apprentissage Stochastique de Noyau de Rang Faible pour la Régression. Dans *Actes de la 7ème Plateforme de l'Association Française pour l'Intelligence Artificielle, AFIA*, pages 263–278, 2011b. (Cité page 7.)

Julien Mairal, Michael Elad, et Guillermo Sapiro. Sparse Representation for Color Image Restoration. *IEEE Transactions on Image Processing*, 17 (1) :53–69, Janvier 2008. (Cité page 24.)

Stéphane G. Mallat et Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12) : 3397–3415, 1993. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=258082. (Cité pages 4, 23, 24, 25, 27, 31 et 36.)

Oded Maron et Andrew W. Moore. Hoeffding Races : Accelerating Model Selection Search for Classification and Function Approximation. Dans *Advances in Neural Information Processing Systems 6 - NIPS'93*, pages 59–66, 1993. URL http://books.nips.cc/papers/files/nips06/0059.pdf. (Cité page 86.)

Andreas Maurer et Massimiliano Pontil. Empirical Bernstein Bounds and Sample Variance Penalization. Dans *Proceedings of the 22nd Annual Conference on Learning Theory - COLT '09*, 2009. URL http://www.cs.mcgill.ca/~colt2009/papers/012.pdf. (Cité pages 6, 71, 78, 87, 101 et 110.)

David A. McAllester. PAC-Bayesian Model Averaging. Dans *Proceedings of the 12th Annual Conference on Computational Learning Theory - COLT '99*, pages 164–170, New York, New York, USA, 1999. ACM Press. ISBN 1581131674. URL http://portal.acm.org/citation.cfm?doid=307400.307435. (Cité page 101.)

Colin McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148–188, 1989. URL http://www.stats.ox.ac.uk/people/academic_staff/colin_mcdiarmid/?a=4113. (Cité pages 94 et 95.)

Volodymyr Mnih, Csaba Szepesvári, et Jean-Yves Audibert. Empirical Bernstein stopping. Dans *Proceedings of the 25th International Conference on Machine learning - ICML '08*, pages 672–679, 2008. URL http://icml2008.cs.helsinki.fi/papers/523.pdf. (Cité page 86.)

Manuel Moussallam. *Représentations Redondantes et Hiérarchiques pour l'Archivage et la Compression de Scènes Sonores*. PhD thesis, TELECOM ParisTech, 2012. (Cité pages 32, 47 et 114.)

Manuel Moussallam, Laurent Daudet, et Gaël Richard. Matching pursuits with random sequential subdictionaries. *Signal Processing*, 92(10) :2532–2544, Octobre 2012. ISSN 0165-1684. URL http://dx.doi.org/10.1016/j.sigpro.2012.03.019. (Cité pages 5, 25, 29, 30, 41 et 46.)

Pekka Naula, Tapio Pahikkala, Antti Airola, et Tapio Salakoski. Greedy Regularized Least-Squares for Multi-task Learning. Dans *Proceedings of the 11th IEEE International Conference on Data Mining Workshops,*

*ICDMW '11*, pages 527–533, Vancouver, Décembre 2011. Ieee. ISBN 978-1-4673-0005-6. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6137424. (Cité pages 57, 66 et 115.)

Deanna Needell et Joel A. Tropp. CoSaMP. *Communications of the ACM*, 53(12) :93–100, Décembre 2010. ISSN 00010782. URL http://portal.acm.org/citation.cfm?doid=1859204.1859229. (Cité pages 25 et 47.)

Tapio Pahikkala, Antti Airola, et Tapio Salakoski. Speeding Up Greedy Forward Selection for Regularized Least-Squares. Dans *Proceedings of the 9th International Conference on Machine Learning and Applications, ICMLA '10*, pages 325–330. Ieee, Décembre 2010. ISBN 978-1-4244-9211-4. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5708852. (Cité page 57.)

Y.C. Pati, R. Rezaiifar, et P.S. Krishnaprasad. Orthogonal Matching Pursuit : Recursive Function Approximation with Applications to Wavelet Decomposition. Dans *Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993. ISBN 0-8186-4120-7. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.1254&rep=rep1&type=pdf. (Cité pages 5, 23 et 25.)

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, et Edouard Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011. URL http://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf. (Cité page 64.)

Thomas Peel, Sandrine Anthoine, et Liva Ralaivola. Empirical Bernstein Inequalities for U-Statistics. Dans *Advances in Neural Information Processing Systems 23, NIPS '10*, pages 1903–1911, 2010. URL http://books.nips.cc/papers/files/nips23/NIPS2010_1114.pdf. (Cité pages 7 et 71.)

Thomas Peel, Valentin Emiya, Liva Ralaivola, et Sandrine Anthoine. Matching Pursuit with Stochastic Selection. Dans *Proceedings of the 20th European Signal Processing Conference, EUSIPCO 2012*, pages 879–883, 2012. (Cité pages 7 et 25.)

Vlad Popovici, Samy Bengio, et Jean-Philippe Thiran. Kernel Matching Pursuit for Large Datasets. *Pattern Recognition*, 38(12) :2385–2390, Décembre 2005. ISSN 00313203. URL http://linkinghub.elsevier.com/retrieve/pii/S0031320305000932. (Cité pages 5 et 25.)

John Ross Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kauffman, 1993. ISBN 1-55860-238-0. (Cité page 53.)

Frank Rosenblatt. The Perceptron : A probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6) :386–408, 1958. ISSN 0033-295X. URL http://content.apa.org/journals/rev/65/6/386. (Cité page 15.)

Ron Rubinstein, Michael Zibulevsky, et Michael Elad. Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit. Rapport technique, Technion - Israel Institute of Technology, 2008. (Cité page 60.)

Bernhard Schölkopf, Ralf Herbrich, et Alex J. Smola. A Generalized Representer Theorem. Dans *Proceedings of the 14th Annual Conference on Computational Learning Theory, COLT '01 and 5th European Conference on Computational Learning Theory, EuroCOLT '01*, pages 416–426, Amsterdam, 2001. (Cité pages 16 et 105.)

Bernhard Schölkopf et Alexander J. Smola. *Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001. ISBN 0262194759. (Cité page 16.)

Terrence J. Sejnowski et Charles R. Rosenberg. Parallel Networks that Learn to Pronounce English Text. *Complex Systems*, 1 :145–168, 1987. URL http://cnl.salk.edu/Research/ParallelNetsPronounce/. (Cité page 50.)

Yevgeny Seldin, François Laviolette, Nicolò Cesa-Bianchi, John Shawe-Taylor, et Peter Auer. PAC-Bayesian Inequalities for Martingales. *IEEE Transactions on Information Theory*, 58(12) :7086–7093, Décembre 2012. ISSN 0018-9448. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6257492. (Cité page 101.)

Robert J. Serfling. Probability Inequalities for The Sum in Sampling Without Replacement. *The Annals of Statistics*, 1974. URL http://projecteuclid.org/euclid.aos/1176342611. (Cité pages 35, 46 et 115.)

Robert J. Serfling. *Approximation Theorems of Mathematical Statistics*. John Wiley & Sons, 1980. (Cité page 75.)

Shai Shalev-Shwartz. *Online learning : Theory, algorithms, and applications*. PhD thesis, Hebrew University, 2007. URL http://eprints.pascal-network.org/archive/00004161/. (Cité page 90.)

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, et Andrew Cotter. Pegasos : Primal Estimated Sub-Gradient Solver for SVM. *Mathematical Programming*, 127(1) :3–30, Octobre 2011. ISSN 0025-5610. URL http://link.springer.com/content/pdf/10.1007%2Fs10107-010-0420-4.pdf. (Cité pages 102, 109 et 110.)

Pannagadatta K. Shivaswamy et Tony Jebara. Empirical Bernstein Boosting. Dans *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics - AISTATS '10*, pages 733–740, 2010. URL http://jmlr.csail.mit.edu/proceedings/papers/v9/shivaswamy10a/shivaswamy10a.pdf. (Cité page 102.)

Pannagadatta K. Shivaswamy et Tony Jebara. Variance Penalizing AdaBoost. Dans *Advances in Neural Information Processing Systems 24, NIPS '11*, pages 1908–1916, 2011. URL http://books.nips.cc/papers/files/nips24/NIPS2011_1075.pdf. (Cité pages 102 et 110.)

J. Michael Steele. *Stochastic Calculus and Financial Applications*. Springer New York, New York, NY, 2001. ISBN 978-1-4419-2862-7. URL http://www.springerlink.com/index/10.1007/978-1-4684-9305-4. (Cité page 93.)

M Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society*, 36(2) :111–147, 1974. URL http://www.jstor.org/stable/2984809. (Cité page 57.)

Vladimir N. Temlyakov. Weak Greedy Algorithms. *Advances in Computational Mathematics*, 12 :213–227, 2000. ISSN 1019-7168. URL http://link.springer.com/content/pdf/10.1023%2FA%3A1018917218956.pdf. (Cité pages 24, 29 et 31.)

Vladimir N. Temlyakov. A Criterion for Convergence of Weak Greedy Algorithms. *Advances in Computational Mathematics*, 17(3) :269–280, 2002. URL http://link.springer.com/article/10.1023/A:1016061804993. (Cité page 31.)

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1) :267–288, 1996. URL http://www.jstor.org/stable/10.2307/2346178. (Cité pages 18 et 55.)

Steven K. Tjoa et K. J. Ray Liu. Factorization of Overlapping Harmonic Sounds Using Approximate Matching Pursuit. *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 257–262, 2011. URL http://ismir2011.ismir.net/papers/PS2-15.pdf. (Cité pages 5, 25 et 29.)

Joel Tropp. User-Friendly Tail Bounds for Sums of Random Matrices. *Foundations of Computational Mathematics*, 12(4) :389–434, Août 2011. ISSN 1615-3375. URL http://link.springer.com/10.1007/s10208-011-9099-z. (Cité page 115.)

Joel A. Tropp. Greed is Good : Algorithmic Results for Sparse Approximation. *IEEE Transactions on Information Theory*, 50(10) :2231–2242, 2004. URL http://authors.library.caltech.edu/9035/1/TROieeetit04a.pdf. (Cité pages 24 et 56.)

Joel a. Tropp, Anna C. Gilbert, et Martin J. Strauss. Algorithms for simultaneous sparse approximation. Part I : Greedy pursuit. *Signal Processing*, 86(3) :572–588, Mars 2006. ISSN 01651684. URL http://linkinghub.elsevier.com/retrieve/pii/S0165168405002227. (Cité pages 5 et 56.)

Vladimir N. Vapnik. An Overview of Statistical Learning Theory. *IEEE Transactions on Neural Networks*, 10(5) :988–999, Janvier 1999. ISSN 1045-9227. URL http://www.ncbi.nlm.nih.gov/pubmed/18252602. (Cité page 11.)

Jason Weston et Chris Watkins. Multiclass Support Vector Machines. Rapport technique, Royal Holloway - University of London, 1998. (Cité pages 50 et 53.)

Zhen James Xiang et Peter J. Ramadge. Fast Lasso Screening Tests Based on Correlations. Dans *IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '12*, pages 2137–2140, Kyoto, 2012. ISBN 978-1-4673-0045-2. (Cité pages 47 et 115.)

Zhen James Xiang, Hao Xu, et Peter J. Ramadge. Learning Sparse Representations of High Dimensional Data on Large Scale Dictionaries. Dans *Advances in Neural Information Processing Systems 24 - NIPS '11*, pages 900–908, 2011. (Cité page 47.)

Ming Yuan et Yi Lin. Model Selection and Estimation in Regression with Grouped Variables. *Journal of the Royal Statistical Society*, 68(1) :49–67, 2006. URL http://www.stat.ucla.edu/~sczhu/courses/UCLA/Stat_232B/chapters/Group_lasso.pdf. (Cité page 61.)

Tong Zhang. On the Consistency of Feature Selection using Greedy Least Squares Regression. *Journal of Machine Learning Research*, 10 :555–568, 2009. URL http://jmlr.org/papers/volume10/zhang09a/zhang09a.pdf. (Cité page 56.)

Hui Zou et Trevor Hastie. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society*, 67(2) :301–320, Avril 2005. ISSN 1369-7412. URL http://doi.wiley.com/10.1111/j.1467-9868.2005.00503.x. (Cité page 55.)

**Titre** Algorithmes de Poursuite Stochastiques et Inégalités de Concentration Empiriques pour l'Apprentissage Statistique

**Résumé** La première partie de cette thèse introduit de nouveaux algorithmes de décomposition parcimonieuse de signaux. Basés sur *Matching Pursuit (MP)* ils répondent au problème suivant : comment réduire le temps de calcul de l'étape de sélection de MP, souvent très coûteuse. En réponse, nous sous-échantillonnons le dictionnaire à chaque itération, en lignes et en colonnes. Nous montrons que cette approche fondée théoriquement affiche de bons résultats en pratique. Nous proposons ensuite un algorithme itératif de descente de gradient par blocs de coordonnées pour sélectionner des caractéristiques en classification multi-classes. Celui-ci s'appuie sur l'utilisation de codes correcteurs d'erreurs transformant le problème en un problème de représentation parcimonieuse simultanée de signaux.
La deuxième partie expose de nouvelles inégalités de concentration empiriques de type Bernstein. En premier, elles concernent la théorie des U-statistiques et sont utilisées pour élaborer des bornes en généralisation dans le cadre d'algorithmes de ranking. Ces bornes tirent parti d'un estimateur de variance pour lequel nous proposons un algorithme de calcul efficace. Ensuite, nous présentons une version empirique de l'inégalité de type Bernstein proposée par Freedman [1975] pour les martingales. Ici encore, la force de notre borne réside dans l'introduction d'un estimateur de variance calculable à partir des données. Cela nous permet de proposer des bornes en généralisation pour l'ensemble des algorithmes d'apprentissage en ligne améliorant l'état de l'art et ouvrant la porte à une nouvelle famille d'algorithmes d'apprentissage tirant parti de cette information empirique.

**Mots-clés** Matching Pursuit ; Algorithmes Stochastiques ; Sélection de Caractéristiques ; Classification Multi-Classes ; Inégalités de Bernstein Empiriques ; U-Statistiques ; Martingales ; Ranking ; Apprentissage en Ligne ; Bornes d'Erreur en Généralisation

**Title** Stochastic Pursuit Algorithms and Empirical Concentration Inequalities for Machine Learning

**Abstract** The first part of this thesis introduces new algorithms for the sparse encoding of signals. Based on *Matching Pursuit (MP)* they focus on the following problem : how to reduce the computation time of the selection step of MP. As an answer, we sub-sample the dictionary in line and column at each iteration. We show that this theoretically grounded approach has good empirical performances. We then propose a bloc coordinate gradient descent algorithm for feature selection problems in the multiclass classification setting. Thanks to the use of error-correcting output codes, this task can be seen as a simultaneous sparse encoding of signals problem. The second part exposes new empirical Bernstein inequalities. Firstly, they concern the theory of the U-Statistics and are applied in order to design generalization bounds for ranking algorithms. These bounds take advantage of a variance estimator and we propose an efficient algorithm to compute it. Then, we present an empirical version of the Bernstein type inequality for martingales by Freedman [1975]. Again, the strength of our result lies in the variance estimator computable from the data. This allows us to propose generalization bounds for online learning algorithms which improve the state of the art and pave the way to a new family of learning algorithms taking advantage of this empirical information.

**Keywords** Matching Pursuit ; Stochastic Algorithms ; Feature Selection ; Multiclass Classification ; Empirical Bernstein Inequalities ; U-Statistics ; Martingales ; Ranking ; Online Learning ; Generalization Bounds