Échantillonnage pour l'accélération des méthodes à noyaux et sélection gloutonne pour les représentations parcimonieuses

Farah Cherfaoui

▶ **To cite this version:**

Farah Cherfaoui. Échantillonnage pour l'accélération des méthodes à noyaux et sélection gloutonne pour les représentations parcimonieuses. Informatique [cs]. Aix-Marseille Université, 2022. Français. NNT : . tel-04004296

HAL Id: tel-04004296
https://hal.science/tel-04004296

Submitted on 24 Feb 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Aix*Marseille université
Socialement engagée

NNT/NL : 2020AIXM0001/001ED000

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 11 juillet 2022 par

Farah CHERFAOUI

Échantillonnage pour l'accélération des méthodes à noyaux et sélection gloutonne pour les représentations parcimonieuses.

Discipline
MATHEMATIQUES ET INFORMATIQUE

Spécialité
Informatique

École doctorale
ED 184 MATHEMATIQUES ET INFORMATIQUE

Laboratoires
Laboratoire d'Informatique et des Systèmes (LIS)
Institut de Mathématiques de Marseille (I2M)

Partenaire
Euranova

Composition du jury

Charles SOUSSEN Université Paris saclay	Rapporteur
Massih-Reza AMINI Université Grenoble Alpes	Rapporteur
Elisa FROMONT Université Rennes 1	Examinatrice
Matthieu KOWALSKI Université Paris 11 - Paris sud	Examinateur
Thomas PEEL GSK	Examinateur
Cécile CAPPONI Université D'Aix-Marseille	Présidente du jury
Liva RALAIVOLA Université D'Aix-Marseille	Directeur de thèse
Sandrine ANTHOINE Université D'Aix-Marseille	Co-directrice de thèse

Résumé

Les contributions de cette thèse se divisent en deux parties. Une première partie dédiée à l'accélération des *méthodes à noyaux* et une seconde à l'optimisation sous contrainte de parcimonie.

Les méthodes à noyaux sont largement connues et utilisées en apprentissage automatique. Toutefois, la complexité de leur mise en œuvre est élevée et elles deviennent inutilisables lorsque le nombre de données est grand. Nous proposons dans un premier temps une approximation des *Ridge Leverage Scores*. Nous utilisons ensuite ces scores pour définir une distribution de probabilité pour le processus d'échantillonnage de la méthode de *Nyström* afin d'accélérer les méthodes à noyaux. Nous proposons dans un second temps un nouveau *framework* basé sur les noyaux, permettant de représenter et de comparer les distributions de probabilités discrètes. Nous exploitons ensuite le lien entre notre *framework* et la *Maximum Mean Discrepancy* pour proposer une approximation précise et peu coûteuse de cette dernière.

La deuxième partie de cette thèse est consacrée à l'optimisation avec contrainte de parcimonie pour l'optimisation de signaux et l'élagage de forêts aléatoires. Tout d'abord, nous prouvons sous certaines conditions sur la *cohérence* du dictionnaire, les propriétés de reconstruction et de convergence de l'algorithme Frank-Wolfe. Ensuite, nous utilisons l'algorithme OMP pour réduire la taille de forêts aléatoires et ainsi réduire la taille nécessaire pour son stockage. La forêt élaguée est constituée d'un sous-ensemble d'arbres de la forêt initiale sélectionnés et pondérés par OMP de manière à minimiser son erreur empirique de prédiction.

Mots clés: approximation de Nyström, Ridge Leverage Scores, accélération méthodes à noyaux, Frank-Wolfe, élagage de forêts aléatoires..

Table des matières

1 Introduction **7**
- 1.1 Introduction et contexte 7
- 1.2 Plan du manuscrit 11

2 Pré-requis et notation **15**
- 2.1 Méthodes à noyaux, approximation de Nyström et Ridge Leverage Scores 15
  - 2.1.1 Méthodes à noyaux 15
  - 2.1.2 Approximation de Nyström 19
  - 2.1.3 Ridge Leverage Scores 21
- 2.2 Parcimonie et algorithmes gloutons 23
  - 2.2.1 Signaux parcimonieux 23
  - 2.2.2 Matching Pursuit et Orthogonal Matching Pursuit . . 24
  - 2.2.3 Cohérence et fonction de Babel 26

3 Approximation des Ridge Leverage Scores pour la méthode de Nyström **29**
- 3.1 Introduction et état de l'art 29
  - 3.1.1 Introduction 29
  - 3.1.2 État de l'art 31
- 3.2 DAC-RLS pour l'approximation des RLS 35
- 3.3 Expériences numériques 39
  - 3.3.1 DAC-RLS pour la méthode de Nyström 40
  - 3.3.2 DAC-RLS pour la réduction du coût d'étiquetage . . 45
- 3.4 Conclusion 47

4 RKHSs discrets pour la comparaison de distributions de probabilités et l'approximation de la MMD **49**
- 4.1 Introduction 49
- 4.2 Maximum Mean Discrepancy (MMD) 51
- 4.3 RKHSs discrets 54

4.4 Comparer des distributions dans les RKHSs discrets 56
4.5 Nyström MMD 61
4.6 Expériences numériques 65
4.6.1 Dispositif expérimental 66
4.6.2 Résultats 67
4.7 Conclusion 72

5 Frank-Wolfe pour l'approximation parcimonieuse de signaux 75
5.1 Introduction et état de l'art 75
5.1.1 Introduction 75
5.1.2 État de l'art 77
5.2 Frank-Wolfe pour l'approximation parcimonieuse 81
5.2.1 L'algorithme Frank-Wolfe 82
5.2.2 Frank-Wolfe pour l'approximation parcimonieuse . . 82
5.3 Propriétés de reconstruction et de convergence 85
5.3.1 Condition de reconstruction 85
5.3.2 Vitesse de convergence 89
5.4 Expériences numériques 98
5.5 Conclusion 100

6 Élagage de forêts aléatoires avec OMP 101
6.1 Introduction et état de l'art 101
6.1.1 Introduction 101
6.1.2 État de l'art 103
6.2 Forêts OMP 104
6.2.1 Notation 104
6.2.2 OMP pour l'élagage de forêt 105
6.2.3 OMP et Gradient Boosting 108
6.3 Expériences numériques 111
6.3.1 Cadre expérimental 111
6.3.2 Résultats 112
6.4 Conclusion 120

7 Conclusion et perspectives 125

A Preuves du Chapitre 3 131
A.1 Preuve du Corollaire 1 131
A.2 Preuve du Corollaire 2 132

B Preuves du Chapitre 5 135
B.1 Preuve du Théorème 5 135

Chapitre 1

Introduction

1.1 Introduction et contexte

L'apprentissage automatique, ou le *machine learning* en anglais, est aujourd'hui omniprésent dans nos vies. Les applications dans lesquelles il est utilisé sont diverses et variées et semblaient encore infaisables dix ans en arrière. Certaines villes côtières utilisent par exemple des bouées intelligentes équipées de caméra, capables de détecter un déchet en mer et d'aller le collecter. Nous pouvons aussi citer l'exemple des petits robots intelligents qui ont été utilisés par les pompiers lors de l'incendie de l'Église de Notre Dame de Paris. Ils sont capables de détecter les flammes, à l'aide de plusieurs caméras et détecteurs thermiques, et de les éteindre. Cela avait permis de préserver la vie de plusieurs pompiers et de leur faire gagner de précieuses minutes. Ces applications, plus excitantes les unes que les autres, sont devenues possibles grâce à l'apparition de grandes masses de données ainsi que de puissantes machines de calcul. Ces grandes bases de données sont utilisées pour entraîner des *modèles d'apprentissage automatique* qui seront embarqués sur des objets tel que des téléphones ou des robots. C'est précisément ces *modèles d'apprentissage automatique* qui leur donneront par la suite la capacité de prendre des décisions.

Avant d'aller plus loin, il est utile de préciser un peu plus ce qu'on entend par le terme *modèles d'apprentissage automatique*. Ainsi, nous utiliserons les termes *modèle d'apprentissage automatique* ou *fonction de décision* afin de désigner une fonction capable de prédire un résultat ou un comportement à partir d'une entrée [Hastie et al., 2009]. Si nous reprenons notre exemple de bouée intelligente, alors le modèle est ici la fonction qui permet de prédire, à partir d'une photo d'un objet, s'il correspond à un déchet ou pas. Nous utiliserons le terme donnée afin de désigner l'objet d'étude (ici

l'image), et le terme étiquette afin de désigner la sortie qui lui est associée (ici déchet/non-déchet).

De tels modèles sont entraînés à l'aide d'un ensemble de données d'entraînement qui sont souvent étiquetées i.e. pour chaque donnée, la sortie associée est connue d'avance. Une fonction de décision peut être représentée par un ensemble de paramètres, et le processus d'apprentissage revient donc à apprendre les meilleurs paramètres qui permettent de bien prédire les sorties associées aux données d'entraînement. Il sera ensuite attendu de ces modèles d'apprentissage automatique d'être capables de prédire la sortie de nouvelles données non vues pendant leur entraînement.

Le processus global permettant d'obtenir un tel modèle d'apprentissage peut donc se résumer en ces quatre étapes :

— collecter et stocker les données,

— les étiqueter,

— entraîner un modèle de prédiction,

— stocker ou embarquer le modèle sur l'appareil de notre choix (bouée, robot, téléphone portable,...).

Malheureusement, lorsque nous disposons de grandes bases de données, des difficultés peuvent surgir à chacune de ces quatre étapes. Premièrement, si en plus d'en avoir beaucoup, les données sont en très grande dimension, alors leur stockage peut rapidement devenir problématique. De plus, l'étiquetage des données étant généralement fait par des humains, il est souvent lent et coûteux d'étiqueter de grandes bases de données. L'entraînement et le stockage du modèle n'échappent pas non plus à ces difficultés. En effet, les modèles entraînés à partir de grands ensembles de données sont souvent complexes et contiennent beaucoup de paramètres à régler. L'apprentissage peut ainsi rapidement devenir très coûteux en temps et en énergie. Pour finir, le stockage de tels modèles est quasi impossible sur de petits appareils.

En somme, bien que les grandes bases de données nous permettent d'apprendre des modèles de plus en plus performants, ces derniers sont aussi de plus en plus volumineux, gourmands et complexes. De nombreuses recherches sont menées afin de relever tous ces défis. Nous présentons dans la partie suivante quelques-unes des méthodes proposées dans la littérature pour résoudre les problématiques qui apparaissent dans chacune des quatre étapes citées plus haut.

Stockage des données : Pour réduire le coût de stockage des données, il est possible de considérer une « approximation » moins vo-

lumineuse de cette donnée. Des méthodes de compression de données [Lelewer and Hirschberg, 1987] sont par exemple utilisées pour construire une telle approximation. Cette approximation est construite en ne gardant que les informations « essentielles » dans la donnée d'origine. Des auto-encodeurs basés sur les réseaux de neurones convolutifs ont aussi été utilisés pour construire des représentations condensées des données [Ni et al., 2020]. Une autre manière de procéder est d'approximer la donnée par une représentation *parcimonieuse* dans une base préalablement choisie. Ainsi, plus cette représentation est parcimonieuse dans la base, plus l'espace nécessaire pour la stocker est petit. Toutefois, les bases ne permettant pas de correctement approximer certains signaux réels, il est possible d'utiliser des objets moins « restrictifs » qu'on appelle des *dictionnaires* [Mallat and Zhang, 1993]. Ils peuvent être préalablement choisis en utilisant des connaissances a priori sur les données [Tropp, 2004, Gribonval and Vandergheynst, 2006, Herzet et al., 2013] ou être appris [Mairal et al., 2009]. Enfin, ils peuvent aussi être discrets [Tropp, 2004, Gribonval and Vandergheynst, 2006, Herzet et al., 2013] ou continus [Elvira et al., 2021]. Néanmoins, ces dictionnaires pouvant être redondants, le problème d'approximation d'une donnée par une représentation parcimonieuse est NP-dur [Davis et al., 1997], et des méthodes d'approximations doivent être utilisées [Tropp, 2006, Tropp, 2004, Gribonval and Vandergheynst, 2006, Herzet et al., 2013, Elvira et al., 2021].

Étiquetage des données : Il existe plusieurs méthodes pour réduire l'effort d'étiquetage. Parmi elles, l'apprentissage actif qui vise à apprendre des modèles moins gourmands sur les étiquettes notamment en choisissant précieusement les données les plus pertinentes à étiqueter [Tong and Koller, 2001, Settles, 2009]. L'apprentissage non supervisé [Hastie et al., 2009] propose quant à lui d'apprendre des modèles sans aucune étiquette en se basant uniquement sur les structures et potentiels liens entre les données. Une autre manière de réduire le coût d'étiquetage est l'adaptation de domaine [Germain et al., 2013, Courty et al., 2016, Gretton et al., 2009]. Au lieu d'étiqueter le jeu de données cible, l'adaptation de domaine propose d'utiliser un jeu de données différent mais déjà étiqueté, qu'on appelle les données source. Une manière de s'assurer que le modèle appris sur ces données source ait de bonnes performances sur les données cible, est de pondérer les données source (on dit aussi adapter) pour qu'elles deviennent « proches » du jeu de données cible. Ces données pondérées et étiquetées peuvent ensuite être utilisées pour apprendre une

fonction de décision, qui devrait avoir de bonnes performances de prédiction sur les données cibles. Une manière de construire ces poids est d'utiliser une distance entre distributions de probabilités tel que la *Maximum Mean discrepancy* (MMD) [Gretton et al., 2012, Gretton et al., 2009], ou la distance de Wasserstein [Shen et al., 2018]. Le problème avec ces distances est que leur calcul devient très difficile lorsque le nombre de données augmente. La complexité du calcul de la MMD se fait par exemple en temps quadratique par rapport au nombre de données.

Entraînement des modèles : Par ailleurs, il est aussi possible de réduire le temps d'entraînement de certains modèles d'apprentissage tels que les réseaux de neurones profonds, les méthodes à noyaux, et les algorithmes de *clustering*. Des méthodes de factorisation par des matrices parcimonieuses ont par exemple été utilisées pour réduire le temps de calcul des réseaux de neurones [Chandra and Sharma, 2016] ainsi que pour accélérer l'algorithme *K-means* [Giffon et al., 2021]. Des méthodes d'approximation de matrice de Gram par des matrices de rang faibles tel que *Random Fourier Features* (RFF) [Rahimi and Recht, 2007], la factorisation de Cholesky [Fine and Scheinberg, 2001], l'échantillonnage de colonnes [Kumar et al., 2009a, Frieze et al., 2004], la méthode des puissances [Gittens and Mahoney, 2016], et la méthode de Nyström [Williams and Seeger, 2000] ont aussi été utilisées pour accélérer l'apprentissage de modèles à noyaux. En particulier, les RFF et Nyström ont par exemple été exploitées pour l'apprentissage de modèles de régression à noyaux [Musco and Musco, 2017, Alaoui and Mahoney, 2015, Avron et al., 2017], pour le *clustering* [Pourkamali-Anaraki, 2020], et l'entraînement des machines à vecteurs de supports (SVM) [Fine and Scheinberg, 2001]. Il est aussi possible d'approximer les matrices de Gram par des matrices diagonales par bloc pour accélérer l'entraînement des SVM [Dong et al., 2005]. Pour finir, des représentations parcimonieuses des données, qu'on appelle *vector quantization*, ont été utilisées pour accélérer l'algorithme des k-plus proches voisins [Gray, 1984].

Stockage des modèles : Enfin, plusieurs travaux visent à réduire le coût de stockage de certains modèles [Buciluǎ et al., 2006]. L'objectif ici est de produire des approximations du modèle considéré qui doivent respecter deux conditions majeures : être moins gourmandes en espace de stockage que le modèle initial, tout en gardant des performances assez « proches » de celles du modèle considéré. De telles mé-

thodes d'approximation ont été proposées par exemple pour les réseaux de neurones profonds [Yu et al., 2017, Novikov et al., 2015] et les forêts aléatoires [Yang et al., 2012a, Zhang and Wang, 2009, Caruana et al., 2004, Hu et al., 2007, Fawagreh et al., 2016]. Comme expliqué plus haut, l'entraînement de réseaux de neurones profonds se fait à travers l'ajustement de matrices, souvent de très grandes tailles, qui contiennent les paramètres du réseaux de neurones. Pour réduire le coût de stockage de ces matrices de paramètres, les auteurs de [Yu et al., 2017] proposent par exemple de les approximer en utilisant des représentations parcimonieuses ainsi que des approximations de rang faible. De telles approximations permettent de stocker ces matrices de paramètres dans un plus petit espace. La décomposition des tenseurs *Tensor-train* (TT) [Novikov et al., 2015] a aussi été explorée pour réduire l'espace de stockage des couches denses et des couches de convolution de ces réseaux de neurones. D'un autre côté, des méthodes d'élagage de forêts aléatoires ont été utilisées pour réduire l'espace nécessaire à leur stockage. Plusieurs critères, souvent basés sur l'erreur empirique de la forêt élaguée ou sur la diversité des arbres de décision qui la constituent, ont été proposés [Yang et al., 2012a, Zhang and Wang, 2009, Caruana et al., 2004, Hu et al., 2007, Fawagreh et al., 2016].

1.2 Plan du manuscrit

Nous proposons dans cette thèse de nous intéresser aux problématiques décrites précédemment. Ce manuscrit est constitué de sept chapitres dont quatre dédiés à nos contributions. Chacun répond à une problématique qui intervient dans l'une des quatre étapes discutées plus haut.

Nous nous intéressons à travers ces chapitres au rôle que jouent les méthodes d'échantillonnage ainsi que l'optimisation avec contrainte de parcimonie pour la réduction des différents coûts constatés lorsque le nombre de données est grand. En particulier, nous étudions comment l'échantillonnage dans la méthode de Nyström permet de réduire le temps d'entraînement des méthodes à noyaux et en particulier le calcul de la distance MMD. Nous verrons ensuite comment l'optimisation sous contrainte de parcimonie permet de construire des représentations parcimonieuses des données ainsi que l'élagage de forêts aléatoires pour réduire l'espace nécessaire pour le stockage de ces dernières.

La suite de ce manuscrit est organisée comme suit. Dans un premier temps, nous rappelons dans le chapitre 2 les principales notions qui seront utilisées dans la suite. En particulier, les notions propres aux méthodes à noyaux et leurs approximations seront introduites en section 2.1. Les

notions de parcimonie d'un signal, ainsi que certains algorithmes permettant de construire des approximations parcimonieuses seront introduite en section 2.2.

Nous aborderons ensuite dans le chapitre 3 l'accélération des méthodes basées sur les fonctions noyaux. Lorsque ces méthodes sont utilisées pour construire une fonction de prédiction, il est souvent nécessaire de construire et surtout de manipuler la matrice de Gram. Toutefois, construire et manipuler cette matrice se fait en temps quadratique (voir même cubique pour certaines opérations) par rapport au nombre de données n. Pour accélérer ces calculs, nous nous intéressons à la méthode de Nyström, qui propose d'approximer la matrice de Gram par une matrice de rang faible. Cette approximation est construite à partir d'un échantillon de $s \ll n$ données, qu'on appelle *landmarks*. Évidemment, la qualité d'une telle approximation dépend du choix de ces *landmarks*. Même si ce choix est souvent fait avec une distribution uniforme [Williams and Seeger, 2000] il existe des méthodes d'échantillonnages, comme celles basées sur les *Ridge Leverage Scores* (RLS), qui permettent de meilleures approximations [Alaoui and Mahoney, 2015]. Malheureusement, le calcul de ces scores est lui même très coûteux : $\Theta\left(n^3\right)$. Nous proposons dans ce chapitre une approximations des RLS dont le calcul se fait en $\Theta\left(nm^2\right)$ où m est un paramètre à régler par l'utilisateur. Nous démontrons que notre approximation des RLS est théoriquement justifiée pour la méthode de Nyström.

Dans le chapitre 4, nous proposons deux contributions. Tout d'abord, nous utiliserons les récents travaux sur les espaces de Hilbert à noyaux reproduisant (RKHS) discrets [Jorgensen and Tian, 2015] afin d'introduire un nouveau *framework* pour représenter et comparer des distributions de probabilités discrètes. Nous verrons comment la MMD, une distance entre distribution bien connue, s'inscrit dans ce *framework*. Enfin, nous utiliserons le lien entre notre *framework* et la MMD pour proposer une approximation à cette dernière. Notre méthode permet de réduire le temps de calcul de la MMD d'un temps quadratique à un temps linéaire en n.

Nous nous intéressons ensuite dans le chapitre 5 à l'approximation de signaux par une représentation parcimonieuse dans un dictionnaire donné. Nous proposons d'utiliser l'algorithme d'optimisation convexe Frank-Wolfe (FW) [Frank and Wolfe, 1956] pour construire une telle approximation. Nous verrons ensuite que sous certaines hypothèses sur la *cohérence* du dictionnaire ainsi que sur le signal cible, il est possible de montrer que l'approximation construite par FW est optimale. Ces résultats sont une extension de ceux déjà connus pour les algorithmes *Matching Pursuit* (MP) [Gribonval and Vandergheynst, 2006] et *Orthogonal Matching Pursuit* (OMP) [Tropp, 2004].

Pour finir, nous utiliserons dans le chapitre 6 l'algorithme *Orthogonal Matching Pursuit*, afin de réduire la taille d'une forêt aléatoire déjà entrainée. En particulier, nous utiliserons OMP pour sélectionner et pondérer un petit sous-ensemble d'arbres de la forêt initiale de manière à approximer le mieux possible le vecteur des étiquettes des données considérées. Nous verrons expérimentalement que notre méthode permet de considérablement réduire la taille de la forêt tout en gardant des erreurs de généralisations assez similaires à celles de la forêt initiale.

Nous résumons l'ensemble des travaux présentés dans ce manuscrit et nous proposons certaines perspectives dans le chapitre 7.

Chapitre 2

Pré-requis et notation

Nous introduisons dans ce chapitre les différentes notations, définitions et outils dont nous aurons besoin tout au long de ce manuscrit. En particulier, nous rappelons en section 2.1 les méthodes à noyaux, l'approximation de Nyström et les *Ridge Leverage Scores*. Ces outils nous seront utiles pour les chapitres 3 et 4. Nous aborderons les concepts de parcimonie et d'algorithmes gloutons, qui nous seront utiles pour les chapitres 5 et 6, en section 2.2.

Nous souhaitons attirer l'attention du lecteur sur le fait que, tout au long de ce manuscrit, certains mots en anglais décrivant des méthodes ou algorithmes ne seront pas traduits en français.

Enfin, nous listons dans la Table 2.1 les différentes notations que nous utiliserons ainsi que leur signification.

2.1 Méthodes à noyaux, approximation de Nyström et Ridge Leverage Scores

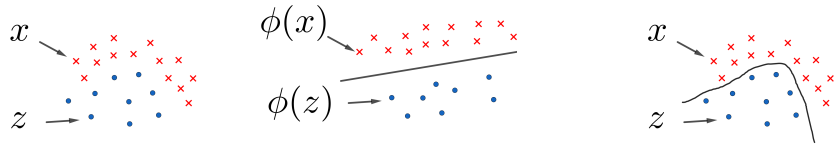
2.1.1 Méthodes à noyaux

En 1992 Boser et al. [Boser et al., 1992a] proposent d'apprendre les premières machines à vecteur de support (SVM) non linéaires en utilisant les méthodes à noyaux.

Nous illustrons en Figure 2.1 leur fonctionnement. Nous prenons pour exemple un problème de classification binaire, dont les données sont non linéairement séparables (Figure 2.1a). Dans ce contexte, des séparateurs linéaires ne sont pas d'une grande utilité puisqu'ils ne permettent pas de

Notations	**Significations**
$\mathbb{R}$	Ensemble des réels.
$\mathbb{R}^+$	Ensemble des réels positifs ou nul.
$\mathbb{X}$	Sous ensemble de $\mathbb{R}^d$.
$\mathbb{Y}$	Sous ensemble de $\mathbb{R}$.
A^T	Transposée de la matrice A.
A^{-1}	Inverse de la matrice A.
A^+	Inverse de Moore Penrose de A.
$\text{trace}(A)$	Trace de la matrice A.
$\sigma(A)$	Valeur singulière de A.
$\varsigma(A)$	Valeur propre de A.
$\sigma_{\min}(A)$, resp. $\sigma_{\max}(A)$	Plus petite, resp. plus grande, valeur singulière de A.
$\varsigma_{\min}(A)$, resp. $\varsigma_{\max}(A)$	Plus petite, resp. plus grande, valeur propre de A.
$A[i,j]$	L'élément de la ligne i et colonne j de A.
$A[i,:]$ (resp. $A[:,i]$)	Ligne (resp. colonne) i de A.
$a[i]$	i^{eme} élément du vecteur $a = (a[1], \ldots, a[d])$.
$\|A\|_F$	Norme de Frobenius de $A : \|A\|_F := \sqrt{\sum_{i,j} A[i,j]^2}$.
$\|A\|_2$	Norme spectrale de $A : \|A\|_2 := \sigma_{\max}(A)$.
$\text{rang}\,(A)$	Rang de la matrice A.
$A \preceq B$	La matrice $B - A$ est semi-définie positive.
I	Matrice identité. Sa taille est précisée si le contexte est ambigu.
$\|a\|_0$	pseudo-norme l_0 du vecteur $a : \|a\|_0 := \|\{i \| a[i] \neq 0\}\|$.
$\mathcal{X} := \{x_1, \ldots, x_n\}$	Ensemble de n points de $\mathbb{X}$.
$X := [x_1^T, \ldots, x_n^T]$	La matrice de taille $n \times d$ des données.
$\mathcal{Y} := \{y_1, \ldots, y_n\}$	Ensemble d'étiquettes de $\mathcal{X}$.
$y := [y_1, \ldots, y_n]$	Vecteur colonne des étiquettes de X.
$[n] := \{1, \ldots, n\}$	L'ensemble des entiers entre 1 et n.
$\Theta(.)$	Mesure de complexité.

TABLE 2.1 – Notations et leur signification.



(a) (b) (c)

FIGURE 2.1 – Illustration des méthodes à noyaux sur un problème de classification binaire. La figure 2.1a représente un ensemble de données dans l'espace d'entrée $\mathbb{X}$ (ici $x, z \in \mathbb{X}$). La Figure 2.1b représente les mêmes données plongées dans *l'espace de représentation* à l'aide d'une fonction ϕ, ainsi que le modèle de classification linéaire appris dans cet espace. La Figure 2.1c illustre le fait que la fonction linéaire apprise dans l'espace de représentation correspond à une fonction non linéaire dans l'espace d'entrée.

correctement séparer nos données (ici les données représentées par des points bleus appartiennent à une classe différente de celle des données représentées par des croix rouges). Pour résoudre ce problème, les données sont plongées dans un *espace de représentation*, de dimension potentiellement infinie, en utilisant une fonction de plongement ϕ. Dans ce nouvel espace, les données sont linéairement séparables, et une fonction de classification linéaire peut être apprise (Figure 2.1b). Cette fonction linéaire construite dans l'espace de représentation, correspond alors à une fonction non linéaire dans l'espace d'entrée, comme illustré dans la Figure 2.1c.

Même si l'espace de représentation est de dimension potentiellement infinie et que l'expression analytique de ces plongements est très souvent inconnue, l'entraînement des modèles d'apprentissage à base de noyau reste faisable. En effet, il existe une famille de fonctions, appelées les noyaux positifs k, faciles à calculer et dont on peut montrer, grâce au théorème de Mercer, qu'elles correspondent à un produit scalaire :

$$k(x, z) := \langle \phi(x), \phi(z) \rangle.$$

Toutefois, commencer par fixer la fonction de plongement ϕ, puis chercher le noyau calculable qui lui est associé, peut s'avérer complexe. En pratique, c'est le chemin inverse qui est fait, en choisissant directement un noyau positif. Nous avons alors une garantie qu'il correspond à un produit scalaire dans un espace de représentation qui restera inconnu et non explicite.

Avant d'aller plus loin, définissons formellement ce qu'est une fonction noyau. Notons que dans ce manuscrit nous nous intéressons uniquement aux fonctions noyaux à valeurs dans $\mathbb{R}$.

Définition 1 (Fonction noyau)**.** *Un noyau k est une fonction à deux variables à valeur dans $\mathbb{R}$, symétrique. C'est-à-dire :*

— $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$

— *pour tout $x, z \in \mathbb{X}$: $k(x, z) = k(z, x)$.*

Notons ici que comme k est une fonction à deux variables, pour un $x \in \mathbb{X}$ fixé, la notation $k(x, .)$, ou encore $k_x(.)$, désigne la fonction à une variable de $\mathbb{X}$ vers $\mathbb{R}$. Un noyau est aussi appelé fonction noyau.

Lors de l'apprentissage de modèles à base de noyaux, nous avons souvent besoin d'évaluer la fonction noyau entre tous les couples d'un jeu de données. Ces mesures sont stockées dans une matrice qu'on appelle matrice de Gram.

Définition 2 (Matrice de Gram)**.** *Soit k un noyau, et $\mathcal{X} := \{x_1, \ldots, x_n\}$ un ensemble de n données. La matrice $K_\mathcal{X}$ de taille $n \times n$ définie comme :*

$$K_\mathcal{X} := (k(x, z))_{(x,z)\in\mathcal{X}\times\mathcal{X}}$$

est appelée la matrice de Gram de k sur $\mathcal{X}$.

Remarque 1. *Pour tout $\mathcal{R}, \mathcal{S} \subseteq \mathcal{X}$, nous noterons $K_{\mathcal{R},\mathcal{S}} := (k(x, z))_{(x,z)\in\mathcal{R}\times\mathcal{S}} \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{S}|}$. Lorsque $\mathcal{R} = \mathcal{S}$ nous utiliserons la notation $K_\mathcal{R}$ (voir Définition 2).*

Définition 3 (Matrice (semi)-définie positive)**.** *Une matrice réelle K de taille $n \times n$ est semi-définie positive si :*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K[i, j] \geq 0,$$

pour tout $c_i \in \mathbb{R}$. La matrice K est définie positive si l'inégalité précédente est stricte dès qu'il existe i tel que $c_i \neq 0$.

Maintenant que nous avons défini les fonctions noyaux ainsi que les matrices semi-définies positives (PSD), nous pouvons nous intéresser au noyaux définis positifs.

Définition 4 (Noyau défini positif)**.** *Soit $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ une fonction noyau. Si pour tout ensemble $\mathcal{X} := \{x_1, \ldots, x_n\} \subseteq \mathbb{X}$, on a que $K_\mathcal{X}$ est une matrice semi-définie positive, alors k est un noyau défini positif.*

Chaque fonction noyau k est le noyau reproduisant d'un unique espace de Hilbert, qui devient ainsi un espace de Hilbert à noyau reproduisant ou en anglais, *Reproducing Kernel Hilbert Space* (RKHS).

Définition 5 (Espace de Hilbert à noyau Reproduisant (RKHS))**.** *Soit* $\mathbb{X}$ *un ensemble non vide et* $\mathcal{H} := \{f \mid f : \mathbb{X} \to \mathbb{R}\}$ *un espace de Hilbert de fonctions muni du produit scalaire* $\langle .,. \rangle_{\mathcal{H}}$*.* $\mathcal{H}$ *est un espace de Hilbert à noyau reproduisant s'il existe un noyau défini positif k tel que :*

— $\forall x \in \mathbb{X}, \forall f \in \mathcal{H} : \langle f, k(x,.) \rangle_{\mathcal{H}} = f(x)$

— $\mathcal{H} = \overline{\text{span}} \left(k_x(.)\right)_{x \in \mathbb{X}}$.

Comme vu plus haut, la taille de la matrice de Gram est quadratique par rapport au nombre de données n. Ainsi, plus n est grand, plus le calcul et le stockage de cette matrice devient difficile. De plus, apprendre des modèles d'apprentissage automatique à base de noyaux requiert souvent le calcul de produits matriciels impliquant la matrice de Gram, ou même le calcul de l'inverse de cette matrice. Ces opérations peuvent rapidement coûter cher. En effet, si l'on considère un ensemble $\mathcal{X}$ de n données, et en supposant que l'évaluation du noyau k se fait en $\Theta(1)$, alors construire la matrice de Gram $K_{\mathcal{X}}$ se fait en temps $\Theta\left(n^2\right)$. Le produit de $K_{\mathcal{X}}$ par une matrice quelconque A de taille $n \times m$ se fait en $\Theta\left(n^2 m\right)$, tandis que l'inversion de $K_{\mathcal{X}}$ se fait en $\Theta\left(n^3\right)$. Tous ces calculs deviennent très longs, voire impossibles, lorsque n est grand.

Plusieurs méthodes de la littérature permettent d'approximer la matrice de Gram ou directement la fonction noyau k afin d'accélérer ces calculs, parmi elles : Random Fourier Features (RFF) [Rahimi and Recht, 2007], méthodes par sélection gloutonnes [Smola and Schölkopf, 2000], décomposition de Cholesky [Fine and Scheinberg, 2001], et la méthode de Nyström [Williams and Seeger, 2000]. Nous présentons plus en détail dans ce qui suit la méthode de Nyström. Cette dernière nous permettra par exemple de réduire la complexité du calcul de $K_{\mathcal{X}} A$ à un temps linéaire en n.

2.1.2 Approximation de Nyström

L'approximation de Nyström repose sur la remarque suivante : si $K_{\mathcal{X}}$ est de rang $s < n$, alors il existe $B \in \mathbb{R}^{n \times s}$ tel que : $K_{\mathcal{X}} = BB^T$. Dans ce cas, calculer le produit $K_{\mathcal{X}} A$ se fait en $\Theta\left(nms\right)$ en calculant d'abord la matrice $B^T A$, puis en calculant son produit avec B. Aussi, stocker $K_{\mathcal{X}}$ revient à stocker B, ce qui se fait en $\Theta\left(ns\right)$.

Malheureusement, la matrice $K_{\mathcal{X}}$ est rarement de rang faible en pratique. Le but de la méthode de Nyström est d'approximer $K_{\mathcal{X}}$ par une matrice

de rang faible $\hat{K}_{\mathcal{X}}$. Cette approximation est construite à partir d'un sous-ensemble $\mathcal{S}$ de s points choisis dans $\mathcal{X}$. La taille de $\mathcal{S}$, qu'on note s, définit ainsi le rang de $\hat{K}_{\mathcal{X}}$. Plus s est petit plus les calculs décrits ci-dessus sont rapides. Toutefois, cette accélération est bien évidemment obtenue au prix d'une perte d'information, qui devient importante au fur et à mesure que s devient petit. Tout l'enjeu est alors de trouver une valeur de s permettant d'accélérer les calculs tout en garantissant une approximation de qualité relativement bonne.

Plus formellement, la méthode de Nyström construit son approximation $\hat{K}_{\mathcal{X}}$ en utilisant un sous ensemble $\mathcal{S} \subseteq \mathcal{X}$ de taille s :

$$\hat{K}_{\mathcal{X}} := K_{\mathcal{X},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{X},\mathcal{S}}^{T}$$

où $K_{\mathcal{S}}$ et $K_{\mathcal{X},\mathcal{S}}$ sont définis comme dans la Remarque 1.

En plus d'accélérer les calculs dans lesquels intervient $\hat{K}_{\mathcal{X}}$, l'approximation de Nyström nous permet de gagner du temps lors de sa construction. En effet, la construction de $\hat{K}_{\mathcal{X}}$ se fait en $\Theta\left(ns^2 + s^3\right)$, contre $\Theta\left(n^2\right)$ pour la construction de $K_{\mathcal{X}}$. Dans la suite de ce manuscrit, nous supposerons que $s \ll n$, et cette complexité devient alors $\Theta\left(ns^2\right)$. De plus, le stockage de $\hat{K}_{\mathcal{X}}$ se fait en $\Theta\left(ns\right)$ en stockant $K_{\mathcal{X},\mathcal{S}}$ et $K_{\mathcal{S}}^{+}$ séparément.

La qualité d'une telle approximation est souvent mesurée par la norme Frobenius ou la norme spectrale de la différence entre $\hat{K}_{\mathcal{X}}$ et $K_{\mathcal{X}}$. Ainsi, pour deux approximations $\hat{K}_{\mathcal{X}}$ et $\hat{K}'_{\mathcal{X}}$, on dira dans ce manuscrit que $\hat{K}_{\mathcal{X}}$ est meilleure que $\hat{K}'_{\mathcal{X}}$ en norme de Frobenius si $\left\|K_{\mathcal{X}} - \hat{K}_{\mathcal{X}}\right\|_F \leq \left\|K_{\mathcal{X}} - \hat{K}'_{\mathcal{X}}\right\|_F$ et on dira que $\hat{K}_{\mathcal{X}}$ est meilleure que $\hat{K}'_{\mathcal{X}}$ en norme spectrale si $\left\|K_{\mathcal{X}} - \hat{K}_{\mathcal{X}}\right\|_2 \leq \left\|K_{\mathcal{X}} - \hat{K}'_{\mathcal{X}}\right\|_2$.

Il est clair que le choix de l'ensemble $\mathcal{S}$ joue un rôle central dans la qualité de l'approximation $\hat{K}_{\mathcal{X}}$. Beaucoup d'efforts ont été fournis afin d'élaborer les meilleures stratégies pour choisir un tel ensemble. On parle dans ce cas de stratégie ou méthode d'échantillonnage. Dans la version originale de Nyström [Williams and Seeger, 2000], les auteurs proposent de choisir $\mathcal{S}$ suivant un tirage uniforme. Bien que cette méthode d'échantillonnage soit simple à implémenter et très peu coûteuse, elle ne prend pas en compte la structure de l'ensemble des données. D'autres méthodes d'échantillonnage telles que les *Determinantal Point Process* (DPP) [Li et al., 2016] et les *Ridge Leverage Scores* (RLS) [Alaoui and Mahoney, 2015] ont été utilisées pour la méthode de Nyström et il a été démontré qu'elles permettent d'obtenir de meilleures approximations de $K_{\mathcal{X}}$. Nous rappelons dans ce qui suit la définition des RLS car cette notion est au cœur des travaux que nous présentons dans le chapitre 3. Nous aborderons aussi quelques stratégies d'échantillonnage basées sur les RLS. Enfin, nous discuterons des garanties

théoriques qu'offrent les RLS et les DPP dans l'approximation de Nyström dans le chapitre 3.

2.1.3 Ridge Leverage Scores

Les *Ridge Leverage Score* (RLS) ont été introduits par Alaoui et Mahoney [Alaoui and Mahoney, 2015], dans le cadre de l'approximation de Nyström pour le problème de régression ridge à noyau. Les RLS sont une méthode permettant de mesurer l'importance d'un point dans un ensemble. Son importance est ici supposée proportionnelle au « niveau de son influence » sur la prédiction de la fonction de régression (nous expliquons plus en détail cette notion dans le paragraphe suivant).

Pour mieux illustrer le concept des RLS, supposons avoir accès à un ensemble de n données étiquetées représenté par la matrice $X \in \mathbb{R}^{n \times d}$ et le vecteur d'étiquettes correspondantes $y \in \mathbb{R}^n$. Soit f la fonction de régression ridge apprise sur X, y avec un noyau k. Il est connu que $f(x_i)$, l'étiquette de x_i prédite par f est définie comme : $f(x_i) := \left[K_{\mathcal{X}} \left(K_{\mathcal{X}} + \lambda I\right)^{-1} y\right][i]$. Soit maintenant f_ϵ la fonction de régression obtenue en modifiant légèrement l'étiquette associée à y_i : par exemple, en y ajoutant $\epsilon > 0$. Ainsi, $f_\epsilon(x_i) := \left[K_{\mathcal{X}} \left(K_{\mathcal{X}} + \lambda I\right)^{-1} \left(y + \epsilon e_i\right)\right][i]$, où e_i contient des zéros partout, sauf à la i^{eme}position où il contient un 1. Lorsque la valeur $f_\epsilon(x_i) - f(x_i)$ est grande, cela signifie que la fonction de régression a beaucoup changé lorsque $y[i]$ a changé. Dans ce cas, le score associé à cette donnée sera grand. En remarquant que :

$$\begin{aligned} f_\epsilon(x_i) - f(x_i) &= \left[K_{\mathcal{X}} \left(K_{\mathcal{X}} + \lambda I\right)^{-1} \left(y + \epsilon e_i\right)\right][i] - \left[K_{\mathcal{X}} \left(K_{\mathcal{X}} + \lambda I\right)^{-1} y\right][i] \\ &= \left[K_{\mathcal{X}} \left(K_{\mathcal{X}} + \lambda I\right)^{-1} \epsilon e_i\right][i] = \epsilon \left[K_{\mathcal{X}} \left(K_{\mathcal{X}} + \lambda I\right)^{-1}\right][i, i], \end{aligned}$$

nous en déduisons le score proposé par Alaoui et Mahoney [Alaoui and Mahoney, 2015] afin de définir l'importance d'un point :

Définition 6 (Ridge Leverage Scores (RLS) [Alaoui and Mahoney, 2015])**.** *Soit $\mathcal{X}$ un ensemble de n données, k une fonction noyau et $K_{\mathcal{X}}$ la matrice de Gram de k sur $\mathcal{X}$. Soit $\lambda > 0$ un paramètre de régularisation. Le Ridge Leverage Score de $x_i \in \mathcal{X}$ est défini comme :*

$$l_i := \left[K_{\mathcal{X}} \left(K_{\mathcal{X}} + \lambda I\right)^{-1}\right][i, i]. \tag{2.1}$$

Remarquons que les RLS dépendent du paramètre λ, mais que nous avons fait le choix de ne pas expliciter cette dépendance dans notre notation afin d'alléger cette dernière.

Nous verrons dans la suite de ce manuscrit qu'il est utile de considérer une autre manière d'écrire les RLS : en utilisant la décomposition en valeurs propres de $K_{\mathcal{X}}$. Supposons donc que $K_{\mathcal{X}} := U\Sigma U^T$, où Σ est une matrice diagonale telle que $\Sigma_{i,i} = \varsigma_i(K_{\mathcal{X}})$ est la i^{eme} valeur propre de $K_{\mathcal{X}}$, et que U est une matrice orthonormée contenant les vecteurs propres associés. On a alors que :

$$l_i := \sum_{j=1}^{n} \frac{\varsigma_j(K_{\mathcal{X}})}{\varsigma_j(K_{\mathcal{X}}) + \lambda} U[i,j]^2. \tag{2.2}$$

Pour vérifier que l'Équation (2.2) est équivalente à l'Équation (2.1), il suffit de se souvenir que : $K_{\mathcal{X}}(K_{\mathcal{X}} + \lambda I)^{-1} = U\Sigma U^T U(\Sigma + \lambda I)^{-1} U^T = U\Sigma(\Sigma + \lambda I)^{-1} U^T$, et ensuite que le i^{eme} élément de la diagonale de cette matrice est : $\sum_{j=1}^{n} \frac{\varsigma_j(K_{\mathcal{X}})}{\varsigma_j(K_{\mathcal{X}})+\lambda} U[i,j]^2$.

Dans ce cas, il est facile de remarquer aussi que :

$$l_i = 1 - \lambda(K_{\mathcal{X}} + \lambda I)^{-1}[i,i]. \tag{2.3}$$

En effet :

$$\begin{aligned} l_i &= \sum_{j=1}^{n} \frac{\varsigma_j(K_{\mathcal{X}})}{\varsigma_j(K_{\mathcal{X}}) + \lambda} U[i,j]^2 = \sum_{j=1}^{n} \frac{\varsigma_j(K_{\mathcal{X}}) + \lambda - \lambda}{\varsigma_j(K_{\mathcal{X}}) + \lambda} U[i,j]^2 \\ &= \sum_{j=1}^{n} U[i,j]^2 - \lambda \sum_{j=1}^{n} \frac{1}{\varsigma_j(K_{\mathcal{X}}) + \lambda} U[i,j]^2 = \|U[i,:]\|_2^2 - \lambda \sum_{j=1}^{n} \frac{1}{\varsigma_j(K_{\mathcal{X}}) + \lambda} U[i,j]^2 \\ &= 1 - \lambda(K_{\mathcal{X}} + \lambda I)^{-1}[i,i]. \end{aligned}$$

Enfin, notons par d_{eff} (pour *effective dimension*) la somme des RLS de $\mathcal{X}$:

$$d_{\text{eff}} := \sum_{i=1}^{n} l_i = Tr\left(K_{\mathcal{X}}(K_{\mathcal{X}} + \lambda I)^{-1}\right). \tag{2.4}$$

Une fois les RLS calculés, ils peuvent être utilisés de différentes manière pour l'échantillonnage. Comme les RLS sont tous positifs ou nuls, ils peuvent par exemple être normalisés (en divisant chaque l_i par d_{eff}), et former ainsi une loi de probabilité. Il est aussi possible de définir des loi de probabilité plus complexes basées sur les RLS, comme nous le verrons dans la Section 3.2.

Nous introduisons dans la section suivante quelques notions et outils d'approximation parcimonieuse qui nous seront utiles pour les chapitres 5 et 6.

2.2 Parcimonie et algorithmes gloutons

2.2.1 Signaux parcimonieux

Nous utiliserons dans cette section, ainsi que dans les chapitres 5 et 6, les termes de signal et dictionnaire pour désigner respectivement un vecteur y de $\mathbb{R}^d$ et une collection de n vecteurs de $\mathbb{R}^d$: $\mathcal{D} := \{v_1, \dots, v_n\}$. Les éléments de $\mathcal{D}$ sont appelés des atomes que nous supposerons normés tout au long de ce manuscrit, i.e. pour tout $i \in [n] : \|v_i\|_2 = 1$.

Certains signaux peuvent être *parcimonieux* dans un dictionnaire donné, c'est-à-dire qu'ils peuvent être exprimés à partir d'un petit sous-ensemble d'éléments du dictionnaire. Nous formalisons cette notion dans la définition suivante.

Définition 7 (Signal m-parcimonieux)**.** *Soit y un signal, $\mathcal{D} := \{v_1, \dots, v_n\}$ un dictionnaire et $m \geq 1$. y est m-parcimonieux dans $\mathcal{D}$ s'il existe $\alpha \in \mathbb{R}^n$ tel que :*

— $y = \sum_{i=1}^{n} \alpha[i] v_i$,

— $\|\alpha\|_0 := |\{i \in [n] \mid \alpha[i] \neq 0\}| \leq m$.

L'écriture d'un signal y comme une combinaison linéaire des éléments de $\mathcal{D}$ est appelée une décomposition de y dans $\mathcal{D}$. Elle est dite m-parcimonieuse si cette décomposition est construite à partir de m éléments de $\mathcal{D}$. Les éléments de $\mathcal{D}$ qui interviennent dans cette décomposition sont indexés par un ensemble qu'on appelle un *support de y*.

Définition 8 (Support d'un signal)**.** *Soit $\mathcal{D} := \{v_1, \dots, v_n\}$ un dictionnaire, et y un signal qui se décompose dans $\mathcal{D}$ de la manière suivante : $y := \sum_{i=1}^{n} \alpha[i] v_i$. L'ensemble*

$$\Lambda(y) := \{i \in [n] \mid \alpha[i] \neq 0\},$$

est appelé un support de y.

Ainsi, un support dépend du signal y, mais aussi d'une décomposition de ce dernier dans un dictionnaire donné.

Malheureusement, trouver une décomposition pour un signal dans un dictionnaire quelconque n'est pas toujours facile. En effet, en notant par $D := [v_1, \dots, v_n] \in \mathbb{R}^{d \times n}$ la matrice dont les colonnes sont les atomes du dictionnaire, alors une décomposition de y dans $\mathcal{D}$ peut être retrouvée en résolvant le problème suivant :

$$\text{Trouver } \alpha \in \mathbb{R}^n \text{ tel que : } y = D\alpha.$$

Lorsque $d < n$, le dictionnaire $\mathcal{D}$ est redondant (i.e. qu'un signal peut avoir plusieurs décomposition différentes dans $\mathcal{D}$) et la matrice D n'est pas inversible. Le problème précédent est mal posé, et il existe dans ce cas plusieurs solutions. Ainsi, pour un signal et un dictionnaire fixés, il existe plusieurs décompositions (m-parcimonieuses ou pas) de y dans $\mathcal{D}$.

Les signaux m-parcimonieux avec $m \ll n$ et $m \ll d$ ont plusieurs avantages. En effet, il est possible de stocker efficacement $y = \sum_{i=1}^{n} \alpha[i] v_i$ en ne gardant en mémoire que les valeurs et positions d'un tel vecteur α qui ne sont pas nulles. De plus, les calculs dans lesquels interviennent les signaux m-parcimonieux sont moins coûteux que ceux dans lesquels interviennent des vecteurs quelconques.

Pour un signal y donné, si y n'est pas m-parcimonieux, nous pouvons souhaiter l'approximer par un signal m-parcimonieux, afin de bénéficier des avantages décrits juste avant. Si y est déjà m-parcimonieux dans $\mathcal{D}$, il y a un intérêt à retrouver l'un de ses supports.

Toutefois, trouver une décomposition m-parcimonieuse de y dans $\mathcal{D}$ est un problème combinatoire qui est NP-dur [Davis et al., 1997]. Il existe cependant plusieurs méthodes permettant de construire des solutions approximées. Nous présentons dans la section suivante deux algorithmes gloutons permettant de faire cela. Le reste des méthodes sera présenté dans le chapitre 5.

2.2.2 Matching Pursuit et Orthogonal Matching Pursuit

Il existe plusieurs méthodes permettant d'approximer un signal y par une décomposition parcimonieuse dans $\mathcal{D}$. Il s'agit dans ce cas de construire un nouveau signal parcimonieux dans $\mathcal{D}$ qui « ressemble » le plus possible à y. Les algorithmes gloutons ont beaucoup été utilisés dans la littérature pour construire de telles approximations [Gribonval and Vandergheynst, 2006, Tropp, 2004]. Ces méthodes construisent de manière itérative une approximation en choisissant à chaque étape l'atome qui permet de « se rapprocher » le plus possible de y. Les algorithmes de Matching Pursuit (MP) [Mallat and Zhang, 1993] et Orthogonal Matching Pursuit (OMP) [Pati et al., 1993] sont deux algorithmes gloutons permettant de construire une telle approximation.

Pour mieux comprendre le fonctionnement de ces méthodes, nous considérons un cas particulier. Supposons que $\mathcal{D}$ soit une base orthonormée et y un signal m-parcimonieux dans $\mathcal{D}$. Dans ce cas, y peut être exprimé comme :

$$y = \sum_{i \in \Lambda(y)} \langle y, v_i \rangle v_i \text{ , avec } |\Lambda(y)| = m.$$

Dans ce cas particulier, cette décomposition de y dans $\mathcal{D}$ est *unique*, puisque $\langle y, v_i \rangle = 0$ pour tout $i \notin \Lambda(y)$.

Ainsi, pour retrouver la décomposition en m termes de y dans $\mathcal{D}$, il suffit de choisir les m atomes qui vérifient : $\langle y, v_i \rangle \neq 0$.

Algorithmiquement, cela peut être fait en construisant la décomposition parcimonieuse de y itérativement. Pour toute étape k, notons par y_k l'approximation construite à cette étape, et par $r_k = y - y_k$ le *résidu* associé (i.e. la partie de y qui reste à reconstruire). Le prochain atome à sélectionner est le plus colinéaire avec le résidu.

Les algorithmes MP et OMP utilisent le processus décrit ci-dessus afin de construire des approximations parcimonieuses lorsque le signal et le dictionnaire sont quelconques. Nous résumons leur fonctionnement dans l'Algorithme 1.

Algorithme 1 : Algorithmes MP et OMP

Entrée : signal y, $m \geq 1$, dictionnaire $\mathcal{D} := \{v_1, \ldots, v_n\}$ tel que $\forall i : \|v_i\|_2 = 1$;
Sortie : une approximation m-parcimonieuse de y dans $\mathcal{D}$;
$k = 0$;
$y_k = 0$;
$r_k = y$;
$\lambda_0 = \emptyset$;
Répéter
    $i_k \in \arg\max_{i \in [n]} |\langle v_i, r_k \rangle|$;
    $\lambda_{k+1} = \lambda_k \cup \{i_k\}$;
    MP : $y_{k+1} = y_k + \langle v_{i_k}, r_k \rangle v_{i_k}$;
    OMP : $y_{k+1} = \arg\min_{a \in \text{span}(v_i)_{i \in \lambda_{k+1}}} \|y - a\|_2$;
    $r_{k+1} = y - y_{k+1}$;
    $k = k + 1$;
Renvoyer y_{k+1} ;

La différence entre MP et OMP réside dans l'étape de mise-à-jour de l'approximation y_{k+1}. L'algorithme de MP projette le résidu uniquement sur le dernier atome sélectionné v_{i_k}, tandis que OMP projette le signal y sur l'ensemble des atomes sélectionnés lors des étapes précédentes. Ainsi, à chaque itération k de OMP, y_k est la meilleure approximation qu'on puisse construire à partir des atomes indexés par λ_k. Ici, la qualité de l'approximation dépend de la distance entre y_k et y. Cette différence permet à OMP de ne pas sélectionner deux fois le même atome et de faire donc moins

d'itérations que MP. Toutefois, comme OMP doit résoudre un problème de moindres carrés à chaque itération, il est plus coûteux que MP.

Nous avons vu plus haut que lorsque $\mathcal{D}$ est une base orthonormée et y est m-parcimonieux dans $\mathcal{D}$, alors les algorithmes MP et OMP sont dans une configuration qui leur permet de reconstruire exactement le signal en m étapes.

Néanmoins, en pratique, $\mathcal{D}$ est rarement une base orthonormée. En effet, en général d est très petit par rapport à n. Le dictionnaire $\mathcal{D}$ est alors redondant, et plusieurs décompositions peuvent exister. Nous avons donc besoin d'outils nous permettant d'étudier les propriétés de MP et OMP pour une catégorie de dictionnaires moins restrictive que les bases orthonormées. Nous introduisons dans la suite la fonction Babel et la cohérence d'un dictionnaire, deux outils qui nous permettront de mesurer à quel point un dictionnaire se « rapproche » d'une base.

2.2.3 Cohérence et fonction de Babel

Pour commencer, nous définissons la cohérence d'un dictionnaire [Mallat and Zhang, 1993, Donoho et al., 2001].

Définition 9 (Cohérence d'un dictionnaire)**.** *La cohérence d'un dictionnaire* $\mathcal{D} := \{v_1, \ldots, v_n\}$ *de n atomes normés est donnée par :*

$$\mu(\mathcal{D}) := \max_{i \neq j} |\langle v_i, v_j \rangle|.$$

Lorsque $\mathcal{D}$ est une base orthonormée, alors pour tout $i \neq j : \langle v_i, v_j \rangle = 0$, et donc $\mu(\mathcal{D}) = 0$. Lorsque tous les atomes sont presque orthogonaux les uns aux autres, $\langle v_i, v_j \rangle \approx 0$, et donc $\mu(\mathcal{D}) \approx 0$. Nous pouvons imaginer que dans ce cas, MP et OMP construisent une approximation de très bonne qualité. Nous verrons dans la section 5.1 que cette intuition est correcte.

Toutefois, la cohérence ne caractérise pas très bien un dictionnaire puisqu'elle ne reflète que les corrélations les plus extrêmes entre les atomes. Lorsque la plupart des produits scalaires sont petits, mais qu'il en existe un seul qui est très proche de 1, alors la cohérence peut être trompeuse car dans ce cas elle est égale à la cohérence d'un dictionnaire dont la valeur de tous les produits scalaires est très proche de 1.

Pour remédier à cela, Tropp [Tropp, 2004] a introduit la fonction Babel, qui mesure la cohérence maximale entre un atome fixé et une collection d'autres atomes.

Définition 10 (Fonction Babel d'un dictionnaire)**.** *Soit* $\mathcal{D} := \{v_1, \dots, v_n\}$ *un dictionnaire de n atomes normés et* $m \in [n]$*. La fonction Babel de* $\mathcal{D}$ *est donnée par :*

$$\mu_1(\mathcal{D}, m) := \max_{\substack{\Lambda \subseteq [n] \\ |\Lambda| = m}} \max_{j \in [n] \setminus \Lambda} \sum_{i \in \Lambda} |\langle v_i, v_j \rangle|.$$

Remarquons que lorsque $m = 1$, alors la fonction Babel devient la cohérence, i.e. $\mu_1(\mathcal{D}, 1) = \mu(\mathcal{D})$.

Nous verrons dans la section 5.1, que tout comme la cohérence, plus la fonction Babel d'un dictionnaire est petite, plus les approximations construites par MP et OMP sont de bonne qualité.

Pour finir ce chapitre, nous résumons l'ensemble des abréviations que nous utiliserons dans ce manuscrit en Table 2.2.

Abréviations	**Significations**	**Lien vers définition formelle**
DPP	Determinantal Point Process	[Kulesza et al., 2012]
RFF	Random Fourier Features	[Rahimi and Recht, 2007]
RLS	Ridge Leverage Scores	Définition 6
SVD	Singular Valued Decomposition	[Petersen et al., 2008]
RKHS	Reproducing Kernel Hilbert Space	Définition 5
Matrice PSD	Matrice semi-définie positive	[Petersen et al., 2008]
MP	Matching Pursuit	Algorithme 1
OMP	Orthogonal Matching Pursuit	Algorithme 1
FW	Algorithme Frank-Wolfe	[Frank and Wolfe, 1956]
MMD	Maximum Mean Discrepancy	[Gretton et al., 2012]

TABLE 2.2 – Abréviations et leurs significations.

Chapitre 3

Approximation des Ridge Leverage Scores pour la méthode de Nyström

3.1 Introduction et état de l'art

Nous présentons dans ce chapitre notre algorithme *Divide And Conquer Ridge Leverage Score* (DAC-RLS) pour l'approximation des *Ridge Leverage Scores* (RLS) pour la méthode de Nyström. La majorité des travaux présentés dans ce chapitre ont été publiés et présentés lors de la conférence ICASSP 2022 [Farah et al., 2022].

Avant d'entamer ce chapitre, rappelons que les notions de méthodes à noyaux, approximation de Nyström et les RLS ont été introduites en section 2.1.

3.1.1 Introduction

Les méthodes à noyaux [Hofmann et al., 2008, Manton and Amblard, 2015] sont un puissant outil permettant de généraliser les méthodes linéaires à des problèmes non linéaires. Leur attrait provient de leurs solides bases théoriques, et de leur efficacité à traiter des données structurées [Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004]. Néanmoins, ces dernières peuvent s'avérer coûteuses lorsque l'ensemble de données est grand. En effet, étant donné $\mathcal{X}$, un ensemble de n points, les méthodes à noyaux requièrent le calcul et la manipulation de la matrice de Gram $K_{\mathcal{X}}$ qui se fait souvent en temps quadratique ou même cubique en n.

Pour cela, un certain nombre de méthodes permettant d'approximer $K_{\mathcal{X}}$ ont été proposées : les *Random Fourier Features* (RFF) [Rahimi and Recht, 2007], la factorisation de Cholesky [Fine and Scheinberg, 2001], l'échantillonnage de colonnes [Kumar et al., 2009a, Frieze et al., 2004], la méthode des puissances [Gittens and Mahoney, 2016], et la méthode de Nyström [Williams and Seeger, 2000].

La méthode de Nyström approxime $K_{\mathcal{X}}$ par une matrice de rang faible $\hat{K}_{\mathcal{X}}$ en utilisant un petit échantillon (notons le $\mathcal{S}$) de $\mathcal{X}$. Traditionnellement, cet échantillon est choisi uniformément. Néanmoins, depuis peu, il a été démontré qu'utiliser des méthodes d'échantillonnage plus dépendantes des données, tel que les *Determinantal Point Process* (DPP) [Li et al., 2016], les RLS [Alaoui and Mahoney, 2015], ou les k-Means [Zhang et al., 2008], permet d'avoir de meilleures approximations.

En particulier, l'utilisation des RLS pour l'approximation de Nyström est théoriquement justifiée, et les expériences numériques démontrent qu'ils permettent de construire de très bonnes approximations [Alaoui and Mahoney, 2015, Musco and Musco, 2017]. Nous proposons donc de nous intéresser dans ce chapitre à la méthode d'échantillonnage par les RLS pour l'approximation de Nyström.

Bien que les RLS constituent une méthode d'échantillonnage intéressante pour l'approximation de Nyström, le calcul de ces scores pour l'ensemble des données se fait en temps cubique par rapport au nombre de données n, ce qui est coûteux pour de grandes valeurs de n.

Nous présentons dans ce chapitre un nouvel algorithme, basé sur la méthode diviser pour régner, permettant d'approximer les RLS en temps $\Theta(nm^2)$, où m est un paramètre réglable par l'utilisateur. Nous verrons que notre méthode permet un bon compromis entre qualité d'approximation de la matrice de Gram et temps de calcul. De plus, notre algorithme est intuitif et simple à implémenter.

Pour commencer, nous présentons en section 3.1.2 l'état de l'art des méthodes d'approximation des RLS. Nous présentons ensuite en section 3.2 notre algorithme DAC-RLS pour l'approximation des RLS, ainsi que ses garanties théoriques. Enfin, la section 3.3 est dédiée à quelques expériences sur des données réelles, ou nous évaluerons notre approximation des RLS sur deux thématiques : l'approximation de la matrice de Gram, ainsi que l'apprentissage actif pour la réduction du coût d'étiquetage.

3.1.2 État de l'art

L'approximation de la matrice de Gram par une matrice de rang faible est un très vieux problème, qui suscite aujourd'hui encore beaucoup d'intérêt. Une telle approximation d'une matrice $K_{\mathcal{X}}$ peut être obtenue en résolvant le problème suivant :

$$K_s := \arg\min_{K} \|K_{\mathcal{X}} - K\|_F, \text{ t.q } \operatorname{rang}(K) = s. \tag{3.1}$$

Ce problème a une solution analytique qui dépend de la décomposition en valeurs propres de $K_{\mathcal{X}}$. Ainsi, si $K_{\mathcal{X}} := U\Sigma U^T$, alors $K_s := U_s\Sigma_s U_s^T$ où $\Sigma_s \in \mathbb{R}^{s\times s}$ contient les s plus grandes valeurs propres de $K_{\mathcal{X}}$, et $U_s \in \mathbb{R}^{n\times s}$ contient les vecteurs propres de $K_{\mathcal{X}}$ correspondants aux valeurs propres dans Σ_s. La matrice K_s est la meilleure approximation de rang s de $K_{\mathcal{X}}$. Néanmoins, calculer la décomposition en valeurs propres de $K_{\mathcal{X}}$ se fait en $\Theta\left(n^2 s\right)$ se qui devient infaisable lorsque le nombre de données n est grand.

Heureusement, il existe d'autres méthodes, moins coûteuses, permettant de construire des approximations de rang faible. Nous proposons dans cette partie de commencer par faire un rapide tour sur les méthodes d'approximation des matrices de Gram. Dans un second temps, nous nous concentrons sur les différentes méthodes d'échantillonnage pour la méthode de Nyström. Enfin, nous présentons les méthodes d'approximation des RLS.

Les RFF [Rahimi and Recht, 2007] sont une méthode d'approximation des fonctions noyaux invariants par translation (un noyau k est dit invariants par translation s'il existe q tel que $k(x,z) = q(x-z)$). L'approximation du noyau k est construite en plongeant les données dans un espace de représentation relativement petit. Cette projection est faite de sorte que le produit scalaire entre deux données ainsi projetées soit presque égal à l'évaluation de k sur ces deux points. Ainsi, les auteurs de [Rahimi and Recht, 2007] proposent d'approximer k par $\hat{k}$ tel que $\hat{k}(x,z) := \langle \hat{\phi}(x), \hat{\phi}(z) \rangle$, où

$$\hat{\phi}(x) := \sqrt{\frac{2}{s}} \left(\cos(\langle x, w_1, \rangle), \ldots, \cos(x, w_s), \sin(\langle x, w_1, \rangle), \ldots, \sin(x, w_s)\right)$$

où $w_1, \ldots, w_s$ sont des vecteurs aléatoires tirés suivant une distribution de probabilité appropriée. Ils montrent ensuite que $\hat{k}$ converge avec grande probabilité vers k lorsque s tend vers l'infini. Cette approximation de la fonction noyau $\hat{k}$ est utilisée pour construire une matrice de Gram approximée $\hat{K}_{\mathcal{X}}$ de rang faible (ici s).

Plusieurs études sur l'impact de l'utilisation de $\hat{k}$ au lieu de k dans les algorithmes d'apprentissage automatique ont étés faites, dont par exemple [Sutherland and Schneider, 2015]. L'avantage des RFF est que la fonction de projection $\hat{\phi}$ explicitement construite est indépendante du jeux de données et ne dépend que du noyau utilisé. Ainsi, elle peut-être appliquée à n'importe quel point $x \in \mathbb{X}$. Néanmoins, les RFF ne sont valables que pour les noyaux invariants par translation. Même s'il existe certains noyaux très utilisés en pratique qui respectent cette condition (comme le noyau *Radial Basis Function* (RBF)), cette restriction peut freiner l'utilisation des RFF.

Frieze et al. [Frieze et al., 2004] proposent de directement approximer la SVD de $K_{\mathcal{X}}$, par celle de $K_{\mathcal{X},\mathcal{S}} = U\Sigma V^T$ où $\mathcal{S} \subseteq \mathcal{X}$ est choisi uniformément sans remise. Les valeurs singulières approximées de $K_{\mathcal{X}}$ sont les éléments de la diagonale de $\sqrt{\frac{n}{|\mathcal{S}|}}\Sigma$, et ses vecteurs singuliers sont les colonnes de U. Il est trivial de voir que lorsque $|\mathcal{S}|$ se rapproche de n, la SVD de $K_{\mathcal{X},\mathcal{S}}$ se rapproche de celle de $K_{\mathcal{X}}$.

Contrairement à la méthode des RFF, cette méthode d'approximation est valable pour toute fonction noyau. Néanmoins, elle est dépendante des données et une nouvelle approximation doit-être calculée lorsque le jeu de données change.

Enfin, l'une des méthodes d'approximation les plus utilisées est celle de Nyström, qui a été proposée pour la première fois par Williams et Seeger en 2001 [Williams and Seeger, 2000, Kumar et al., 2009b, Gittens and Mahoney, 2016]. Dans leur papier, les auteurs démontrent empiriquement l'utilité de la méthode de Nyström sur un certain nombre de jeux de données.

La première étude théorique et rigoureuse de la méthode de Nyström a été proposée en 2005 par Drineas et Mahoney [Drineas et al., 2005]. Même si l'échantillonnage de $\mathcal{S}$ se fait traditionnellement de manière uniforme, les auteurs de [Drineas et al., 2005] proposent un échantillonnage non-uniforme et dépendant des données. Ils démontrent que si $\hat{K}_{\mathcal{X}}$ est une approximation de Nyström de $K_{\mathcal{X}}$ construite à partir d'un sous-ensemble $\mathcal{S}$ échantillonné suivant la loi de probabilité $\left\{\frac{K_{\mathcal{X}}[1,1]}{\sum_{j=1}^{n} K_{\mathcal{X}}[j,j]}, \ldots, \frac{K_{\mathcal{X}}[n,n]}{\sum_{j=1}^{n} K_{\mathcal{X}}[j,j]}\right\}$, alors avec grande probabilité, si $|\mathcal{S}| = \Theta\left(\frac{r}{\rho^4}\right)$, nous avons :

$$\left\|K_{\mathcal{X}} - \hat{K}_{\mathcal{X}}\right\|_2 \leq \|K_{\mathcal{X}} - K_r\|_2 + \rho \sum_{i=1}^{n} K_{\mathcal{X}}[i,i]^2.$$

Ici K_r est la meilleure approximation de rang r de $K_{\mathcal{X}}$ (solution du Pro-

blème (3.1)). Lorsque ρ est très petit, la taille de $\mathcal{S}$ tend vers n, et $\hat{K}_{\mathcal{X}}$ devient une aussi bonne approximation de $K_{\mathcal{X}}$ que l'est K_r. De plus, dans le cas particulier où $K_{\mathcal{X}}$ est de rang r, on a $\|K_{\mathcal{X}} - K_r\|_2 = 0$, et donc lorsque ρ est très petit $\hat{K}_{\mathcal{X}}$ converge vers $K_{\mathcal{X}}$.

Lorsque le noyau est normalisé (i.e. $k(x,x) = 1$ pour tout $x \in \mathbb{X}$), l'échantillonnage de Drineas et Mahoney [Drineas et al., 2005] devient un échantillonnage uniforme. Toutefois, la borne sur $\left\|K_{\mathcal{X}} - \hat{K}_{\mathcal{X}}\right\|_2$ dépend dans ce cas du nombre de données n, puisque $\sum_{i=1}^{n} K_{\mathcal{X}}[i,i]^2 = n$. Lorsque n est très grand, la borne proposée par [Drineas et al., 2005] n'est plus très informative.

L'échantillonnage uniforme ainsi que celui proposé par [Drineas et al., 2005] ne permettent cependant pas d'avoir les meilleures approximations. Pour cela, des méthodes d'échantillonnage plus complexes ont été développées.

Par exemple, l'échantillonnage suivant les DPP [Li et al., 2016]. Il s'agit d'échantillonner $\mathcal{S}$ de taille s proportionnellement à $\det(K_{\mathcal{S}})$. L'approximation $\hat{K}_{\mathcal{X}}$ obtenue dans ce cas vérifie :

$$\mathbb{E}_{\mathcal{S}}\left[\left\|K_{\mathcal{X}} - \hat{K}_{\mathcal{X}}\right\|_2\right] \leq \|K_{\mathcal{X}} - K_s\|_2 (s+1)(n-s).$$

où K_s est la meilleure approximation de rang s de $K_{\mathcal{X}}$. Ainsi, lorsque s tend vers n, l'espérance de $\left\|K_{\mathcal{X}} - \hat{K}_{\mathcal{X}}\right\|_2$ tend vers zéro, ce qui veut dire que l'approximation $\hat{K}_{\mathcal{X}}$ tend vers $K_{\mathcal{X}}$. Malheureusement cette borne dépend du nombre de données n. Si n est très grand, et que s est très petit, alors le résultat de [Li et al., 2016] ne nous permet pas de dire grand chose sur la qualité de $\hat{K}_{\mathcal{X}}$.

Une autre distribution de probabilité dépendante des données est celle basée sur les RLS.

Il a été démontré dans [Alaoui and Mahoney, 2015, Musco and Musco, 2017], que choisir $\mathcal{S}$ avec une probabilité proportionnelle aux RLS (tel que définis dans la Définition 6) permet de construire une approximation de Nyström qui vérifie :

$$\left\|K_{\mathcal{X}} - \hat{K}_{\mathcal{X}}\right\|_2 \leq \lambda,$$

dès lors que $|\mathcal{S}| = \Theta(d_{\text{eff}})$, où $d_{\text{eff}} := \text{trace}\left[K_{\mathcal{X}}\left(K_{\mathcal{X}} + \lambda I\right)^{-1}\right]$ est la somme des RLS. De plus, il est aussi possible de borner l'erreur de prédiction d'un modèle appris avec $\hat{K}_{\mathcal{X}}$ au lieu de $K_{\mathcal{X}}$ [Bach, 2013].

Remarquons que contrairement aux DPP, cette borne ne dépend pas de n. Toutefois, rappelons que la croissance de d_{eff} est inversement proportion-

nelle à celle de λ :

$$d_{\text{eff}} := \text{trace}\left[K_{\mathcal{X}}\left(K_{\mathcal{X}} + \lambda I\right)^{-1}\right] = \sum_{i=1}^{n} \frac{\varsigma_i\left(K_{\mathcal{X}}\right)}{\varsigma_i\left(K_{\mathcal{X}}\right) + \lambda}.$$

Ainsi, si l'on peut être tenté de prendre λ très petit pour réduire la borne sur $\left\|K_{\mathcal{X}} - \hat{K}_{\mathcal{X}}\right\|_2$, alors d_{eff} devient grand, et par conséquent $|\mathcal{S}|$ tend vers n.

Le problème avec les RLS est que leur calcul coûte aussi cher que la décomposition en valeurs propres de $K_{\mathcal{X}}$: $\Theta(n^3)$. Ils ne sont donc pas directement utilisables pour la méthode de Nyström. Par conséquent, des méthodes permettant d'approximer ces techniques d'échantillonnage ont été proposées.

Alaoui et Mahoney [Alaoui and Mahoney, 2015] proposent la première approximation des RLS, en remplaçant la valeur de la Définition 6 par :

$$\hat{l}_i := \left[\widetilde{K}_{\mathcal{X}}\left(\widetilde{K}_{\mathcal{X}} + \lambda I\right)^{-1}\right][i,i], \tag{3.2}$$

où $\widetilde{K}_{\mathcal{X}}$ est une approximation de Nyström de $K_{\mathcal{X}}$ construite avec un échantillon de taille m choisi uniformément. Ainsi, les auteurs proposent de construire une première approximation de Nyström de $K_{\mathcal{X}}$ en utilisant un échantillonnage uniforme, puis d'utiliser cette approximation pour estimer les RLS. Ces scores ainsi approximés serviront ensuite de méthode d'échantillonnage pour choisir $\mathcal{S}$, et construire $\hat{K}_{\mathcal{X}}$. La matrice $\widetilde{K}_{\mathcal{X}}$ étant PSD et de rang m, elle peut être décomposée comme : $\widetilde{K}_{\mathcal{X}} = BB^T$, où $B \in \mathbb{R}^{n \times m}$. En utilisant l'identité de Woodbury [Petersen et al., 2008], nous pouvons démontrer que : $\hat{l}_i = B[i,:]^T\left(B^TB + \lambda I\right)^{-1}B[i,:]$. Ainsi le calcul de l'approximation des RLS se fait en temps $\Theta\left(nm^2\right)$ lorsque m est très petit par rapport à n. De plus, les auteurs montrent que si $m = \Theta\left(\frac{Tr(K_{\mathcal{X}})}{n\lambda\epsilon}\right)$, alors avec grande probabilité :

$$l_i - 2\epsilon \leq \hat{l}_i \leq l_i.$$

En d'autres termes, plus le rang de $\widetilde{K}_{\mathcal{X}}$ (i.e. m) est grand, plus l'approximation des RLS est bonne (car dans ce cas ϵ est petit). Toutefois, plus m est grand, plus le calcul des RLS approximés est lent. Il y a donc un compromis naturel entre qualité d'approximation et temps de calcul.

Alaoui et Mahoney [Alaoui and Mahoney, 2015] ont aussi étudié l'impact de l'utilisation de $\hat{K}_{\mathcal{X}}$ dans des algorithmes d'apprentissage. Ils montrent en particulier, que les modèles appris avec leur approximation $\hat{K}_{\mathcal{X}}$

ont de meilleurs performances que les modèles appris avec l'approximation Nyström avec échantillonnage uniforme [Alaoui and Mahoney, 2015].

Plus élaboré encore, les auteurs de [Musco and Musco, 2017] proposent de récursivement répéter l'algorithme de Alaoui et Mahoney [Alaoui and Mahoney, 2015]. A chaque itération, une approximation $\hat{l}_i$ est calculée comme dans l'Équation (3.2), mais avec $\widetilde{K}_{\mathcal{X}}$ construite à partir de m points choisis proportionnellement aux RLS calculés à l'étape précédente. Les auteurs proposent ensuite de choisir $\mathcal{S}$ proportionnellement aux RLS récursivement approximés et de les utiliser pour construire $\hat{K}_{\mathcal{X}}$. Ils montrent ensuite que si $|\mathcal{S}| = \Theta\left(d_{\text{eff}} \log\left(d_{\text{eff}}\right)\right)$, alors avec grande probabilité, leur approximation vérifie :

$$\hat{K}_{\mathcal{X}} \preceq K_{\mathcal{X}} \preceq \hat{K}_{\mathcal{X}} + \lambda I,$$

ou encore :

$$\left\| K_{\mathcal{X}} - \hat{K}_{\mathcal{X}} \right\|_2 \leq \lambda.$$

Enfin, ils montrent que la complexité de leur algorithme est de $\Theta\left(nm^2\right)$. Notons que les RLS récursifs ont besoin de plus de données que les RLS exacts (un facteur logarithmique en d_{eff}) pour atteindre la même qualité d'approximation de $K_{\mathcal{X}}$.

Nous présentons dans la suite notre algorithme DAC-RLS pour l'approximation des RLS. Notre méthode se positionne entre les deux précédentes : elle permet une meilleure approximation en pratique de la matrice de Gram que celle des RLS approximés par [Alaoui and Mahoney, 2015] et équivalente à celle des RLS récursifs de [Musco and Musco, 2017]. De plus, notre méthode est plus rapide que celle de [Musco and Musco, 2017], mais plus lente que celle de [Alaoui and Mahoney, 2015].

3.2 DAC-RLS pour l'approximation des RLS

Comme dit dans la section précédente, la complexité de calcul des RLS exacts est cubique par rapport au nombre de données. C'est en fait l'inversion de la matrice $K_{\mathcal{X}} + \lambda I$ qui coûte le plus cher. Nous proposons un algorithme d'approximation des RLS basé sur la méthode de diviser pour régner. Ainsi, au lieu de construire entièrement la matrice $K_{\mathcal{X}}$, et d'inverser ensuite une matrice de taille $n \times n$, nous proposons de diviser notre ensemble de données en r petits sous-ensembles disjoints, chacun de taille m_i pour $i \in [r]$. Nous calculons ensuite les RLS sur chacun des sous-ensembles séparément, en n'inversant que des matrices de tailles m_i. Plus la taille des sous-ensembles (m_i) est petite, plus le calcul de la matrice

inverse sera rapide. Ainsi, la complexité du calcul des RLS approximés est réduite de $\Theta(n^3)$ à $\Theta(\sum_{i=1}^{r} m_i^3)$.

Nous illustrons notre méthode dans la Figure 3.1, et résumons ces étapes dans l'Algorithme 2.

Dans la suite de ce chapitre, et sans perte de généralisation, nous supposerons que tous les sous-ensembles ont la même taille : m. Dans ce cas, la complexité de notre méthode est de : $\Theta(nm^2)$.

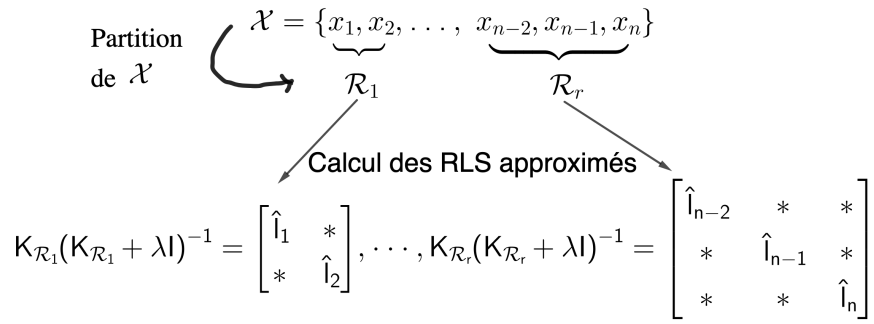


FIGURE 3.1 – Illustration de notre algorithme DAC-RLS. Le jeu de n données est divisé en r sous-ensembles : $\mathcal{R}_1 := \{x_1, x_2\}; \ldots; \mathcal{R}_r := \{x_{n-2}, x_{n-1}, x_n\}$. On construit les matrices noyau correspondant à chaque sous-ensemble : $K_{\mathcal{R}_1}, \ldots, K_{\mathcal{R}_r}$. Chaque matrice est ensuite utilisée pour calculer les RLS des éléments du sous-ensemble correspondant. Notons que les petites étoiles (*) dans les matrices correspondent à des valeurs non exploitées.

Notre algorithme DAC-RLS est simple et intuitif. Une question légitime qu'on peut se poser est alors si nous sommes capables d'avoir des garanties théoriques sur la qualité de ces RLS approximés. Plus particulièrement, nous nous intéressons à l'utilisation des RLS approximés dans l'approximation de la matrice de Gram par la méthode de Nyström.

Pour cela, nous utilisons l'algorithme de [Musco and Musco, 2017], rappelé ici dans l'Algorithme 3. Cet algorithme constitue une trame à suivre pour construire une approximation de Nyström à partir d'une *sur-approximation* des RLS. L'expression *sur-approximation* désigne une approximation $\left(\hat{l}_i\right)_{i\in[n]}$ des RLS qui vérifie $\hat{l}_i \geq l_i$ pour tout $i \in [n]$. Chaque donnée

Algorithme 2 : Approximation des Ridge Leverage Scores par DAC-RLS

Entrée : ensemble de données $\mathcal{X} := \{x_1, \dots, x_n\}$, noyau k, nombre de sous-ensembles $r > 0$, taille des sous-ensembles $\{m_i\}_{i=1}^r$;
Sortie : RLS approximés : $\hat{l}_i$, $\forall i \in [n]$;
Partitionner $\mathcal{X}$: Soit $\mathcal{R}_1, \dots, \mathcal{R}_r$ des sous-ensembles de $\mathcal{X}$ tel que : $|\mathcal{R}_i| = m_i$, $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$, pour tout $i, j \in [n]$ et $\cup_{i=1}^r \mathcal{R}_i = \mathcal{X}$;
Calculer $K_{\mathcal{R}_1}(K_{\mathcal{R}_1} + \lambda I)^{-1}, \dots, K_{\mathcal{R}_r}(K_{\mathcal{R}_r} + \lambda I)^{-1}$ où $K_{\mathcal{R}_i} = (k(x,z))_{(x,z)\in\mathcal{R}_i^2}$
pour $x_i \in \mathcal{X}$ **faire**
    Soit $\mathcal{R} \in \{\mathcal{R}_1, \dots, \mathcal{R}_r\}$ le sous-ensemble qui contient x_i ;
    Soit j_i la position de x_i dans $\mathcal{R}$;
    Soit $\hat{l}_i = [K_{\mathcal{R}}(K_{\mathcal{R}} + \lambda I)^{-1}][j_i, j_i]$;
fin
Renvoyer $\hat{l}_i$ pour tout $i \in [n]$

x_i est tirée suivant une loi de Bernoulli avec probabilité de succès proportionnelle à son RLS approximé $\hat{l}_i$ pour construire $\mathcal{S}$. Contrairement à la méthode de [Alaoui and Mahoney, 2015] où le rang de $\hat{K}_{\mathcal{X}}$ est connu par avance, dans notre cas (et celui de [Musco and Musco, 2017]), la taille de $\mathcal{S}$ est une variable aléatoire. Enfin, remarquons que les probabilités p_i utilisées pour échantillonner $\mathcal{S}$ dans l'Algorithme 3 dépendent d'un paramètre ρ. Nous verrons dans le Théorème 1 que ce paramètre nous permettra de contrôler la qualité de notre approximation.

Algorithme 3 : Nyström avec RLS approximés [Musco and Musco, 2017]

Entrée : $\mathcal{X} := \{x_1, \dots, x_n\}$, $\rho \in]0, 1]$;
Sortie : $\hat{K}_{\mathcal{X}}$, une approximation de $K_{\mathcal{X}}$;
Calculer $\hat{l}_1, \dots, \hat{l}_n$ une sur-approximation des RLS;
Soit $p_i = \min\left\{1, 16\hat{l}_i \log\left(\frac{1}{\rho}\sum_j \hat{l}_j\right)\right\}$;
Tirer $\mathcal{S} \subseteq \mathcal{X}$, où chaque x_i est choisi avec probabilité p_i ;
Calculer l'approximation Nyström : $\hat{K}_{\mathcal{X}} := K_{\mathcal{X},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{X},\mathcal{S}}^T$ où $K_{\mathcal{X},\mathcal{S}}$ et $K_{\mathcal{S}}$ sont définis dans la Remarque 1;
Renvoyer $\hat{K}_{\mathcal{X}}$;

Avant de pouvoir utiliser l'Algorithme 3, nous devons vérifier que

notre approximation des RLS calculée par l'Algorithme 2, est bien une sur-approximation des RLS exacts. Ceci est fait dans le corollaire suivant :

Corollaire 1. *Soit $\mathcal{X} = \{x_1, \dots, x_n\}$ un ensemble de données et k un noyau. Pour tout $i \in [n]$, soit l_i le RLS de x_i tel que défini dans la Définition 6, et soit $\hat{l}_i$ une approximation de l_i calculée avec l'Algorithme 2. On a alors :*

$$\hat{l}_i \geq l_i.$$

Démonstration. Voir Annexe A.1. □

Grâce au Corollaire 1, nous sommes maintenant en mesure d'utiliser les approximations des RLS calculées par DAC-RLS dans l'Algorithme 3 afin de construire $\hat{K}_{\mathcal{X}}$, une approximation de la matrice de Gram. En faisant ainsi, nous sommes capables de mesurer la qualité de l'approximation $\hat{K}_{\mathcal{X}}$.

Théorème 1. *Soit $\mathcal{X}$ un ensemble de données et k un noyau. Soit $\hat{K}_{\mathcal{X}}$ une approximation de $K_{\mathcal{X}}$ construite avec l'Algorithme 3 en utilisant les RLS approximés avec l'Algorithme 2. On a alors, avec probabilité supérieure à $1 - \rho$:*

$$K_{\mathcal{X}} - \lambda I \preceq \hat{K}_{\mathcal{X}} \preceq K_{\mathcal{X}}.$$

De plus, $|\mathcal{S}| \leq 32 \log\left(\frac{1}{\rho} \sum_{i=1}^{n} \hat{l}_i\right) \sum_{i=1}^{n} \hat{l}_i$.

Démonstration. Appliquer le Théorème 3 de [Musco and Musco, 2017] en utilisant le Corollaire 1. □

Notons que ce résultat implique aussi le résultat suivant sur la norme de la différence des deux matrices :

$$\left\| K_{\mathcal{X}} - \hat{K}_{\mathcal{X}} \right\|_2 \leq \lambda.$$

Le Théorème 1 nous donne trois informations importantes. La première, est que la distance entre $\hat{K}_{\mathcal{X}}$ et $K_{\mathcal{X}}$ est bornée. Cette borne dépend du terme de régularisation λ (résultat classique qu'on retrouve dans l'état de l'art [Musco and Musco, 2017]), mais ne dépend pas de la taille des sous-ensembles m. Cela nous laisse donc la liberté de prendre des sous-ensembles de taille relativement petite, afin de réduire le temps de calcul sans pour autant affecter la qualité de l'approximation. Attention quand même : si m est très petit, $\hat{l}_i$ risque d'être grand. Dans ce cas $p_i = 1$ (voir Ligne 4 de l'Algorithme 3), et le rang de $\hat{K}_{\mathcal{X}}$ reste le même que celui de $K_{\mathcal{X}}$, et aucune approximation n'est faite.

La deuxième information importante est que les probabilités p_i dépendent d'un paramètre ρ qui contrôle le niveau de certitude que nous

avons sur la borne proposée dans le Théorème 1. Plus ρ est petit, plus nous seront sûr que notre borne sur la qualité d'approximation de la matrice de Gram est bonne. Toutefois, si ρ est trop petit, p_i risque d'être égale à 1, et dans ce cas aussi, aucune approximation n'est faite.

Enfin, la troisième information importante que nous donne ce théorème est que le rang de la matrice $\hat{K}_{\mathcal{X}}$, i.e. $|\mathcal{S}|$, est borné par la somme des RLS approximés. Il est donc important de borner aussi la somme des RLS approximés avec DAC-RLS. Nous présentons ce résultat dans le Corollaire 2.

Corollaire 2. *Soit $\mathcal{X} = \{x_1, \dots, x_n\}$ un ensemble de données et k un noyau. Pour tout $i \in [n]$, soit l_i le RLS de x_i tel que défini dans la Définition 6, et soit $\hat{l}_i$ une approximation de l_i calculée avec l'Algorithme 2. On a alors :*

$$\sum_{i=1}^{n} \hat{l}_i \leq \sum_{i=1}^{n} l_i + n\left(1 - \frac{1}{\beta}\right),$$

où $\beta := \frac{\varsigma_{max}(K_{\mathcal{X}})+\lambda}{\varsigma_{min}(K_{\mathcal{X}})+\lambda}$.

Démonstration. Voir Annexe A.2. □

β est le conditionnement de la matrice $K_{\mathcal{X}} + \lambda I$. Lorsque la matrice $K_{\mathcal{X}}$ est bien conditionnée, i.e. $\varsigma_{max}(K_{\mathcal{X}}) \approx \varsigma_{min}(K_{\mathcal{X}})$ (c'est le cas de la matrice identité par exemple) alors $\sum_{i=1}^{n} \hat{l}_i \approx \sum_{i=1}^{n} l_i$. Dans ce cas le Corollaire 2 nous assure que le rang de $\hat{K}_{\mathcal{X}}$ construite avec les RLS de l'Algorithme 2 est le même que le rang de $\hat{K}_{\mathcal{X}}$ construite avec les RLS de la méthode récursive [Musco and Musco, 2017]. Toutefois, si la matrice $K_{\mathcal{X}}$ est mal conditionnée, i.e. $\varsigma_{\min}(K_{\mathcal{X}}) \ll \varsigma_{\max}(K_{\mathcal{X}})$, alors $\frac{1}{\beta}$ se rapproche de zéro, et le corollaire précédent ne nous donne aucune information sur le rang de $\hat{K}_{\mathcal{X}}$.

La borne sur la qualité de $\hat{K}_{\mathcal{X}}$ construite avec les RLS calculés par DAC-RLS est donc similaire à celle de [Musco and Musco, 2017] lorsque la matrice $K_{\mathcal{X}}$ est bien conditionnée, mais est plus lâche lorsque $K_{\mathcal{X}}$ n'est pas bien conditionnée. Cette perte peut être vue comme le prix à payer pour avoir la possibilité de contrôler la complexité de notre algorithme (en prenant m petit) sans affecter notre borne.

3.3 Expériences numériques

Dans cette section nous évaluons empiriquement notre algorithme DAC-RLS, que nous notons dans cette section DAC afin d'alléger les notations.

Nous proposons de faire cela sur deux problématiques différentes. Premièrement, nous nous intéressons à l'utilisation des RLS dans la méthode de Nyström pour l'approximation de la matrice de Gram. Dans un second temps, nous nous intéressons à l'utilisation des RLS pour la réduction du coût d'étiquetage.

Une implémentation Python de notre algorithme est disponible ici : https://github.com/DAC-paper/Divide_And_Conquer_leverage_score_approximation.

Dans toute cette section, nous considérons plusieurs jeux de données de référence que nous résumons dans la Table 3.1. Nous utilisons un noyau gaussien de variance σ^2 et un hyper-paramètre de régularisation λ que nous fixons comme proposé par [Brefeld et al., 2006]. Ainsi, en notant notre jeu de données par $\mathcal{X} := \{x_1, \ldots, x_n\}$, nous avons :

$$\sigma^2 := \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \|x_i - x_j\|_2^2,$$

et

$$\lambda := \left(\frac{1}{n} \sum_{i=1}^{n} \|x_i\|_2 \right)^{-1}.$$

Jeu de données	nombre de données	nombre
Covertype [R. Collobert, S. Bengio, and Y. Bengio, 2002]	581012	54
Adult [Kohavi, 1996]	48842	15
KC1 [Sayyad Shirabad and Menzies, 2005]	2109	21
Houses [Ahmed and Moustafa, 2016]	20640	8

TABLE 3.1 – Description des jeux de données

3.3.1 DAC-RLS pour la méthode de Nyström

Dans cette partie nous proposons d'utiliser notre Algorithme 2 afin d'approximer les RLS. Ces scores sont ensuite normalisés (en les divisant par leur somme) pour former des probabilités. Nous choisissons un ensemble $\mathcal{S}$ de s points, où chaque donnée est choisie avec une probabilité proportionnelle aux RLS approximés. Enfin, nous utilisons cet ensemble de points afin de construire l'approximation de Nyström de la matrice $K_{\mathcal{X}}$.

Notons que nous n'avons pas utilisé exactement l'Algorithme 3 pour nos expériences car dans ce cas nous n'avons pas le contrôle sur la taille de

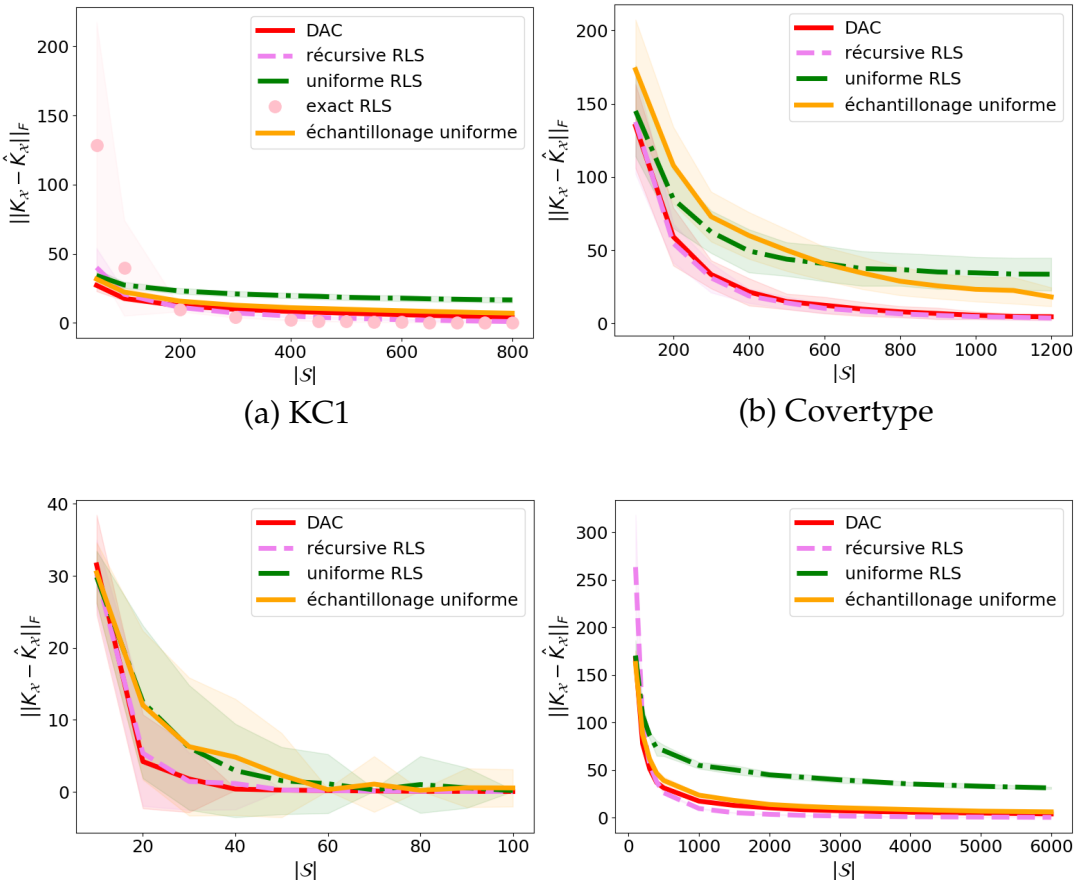
$\mathcal{S}$. Or, nous souhaitons ici étudier la qualité de l'approximation en fonction de cette taille. Ainsi, nous proposons ici de commencer par fixer la taille de l'ensemble des *landmarks*, puis de tirer ces points proportionnellement aux RLS approximés.

Nous comparons notre algorithme DAC à deux autres méthodes de l'état de l'art dont le temps de calcul est aussi de $\Theta\left(nm^2\right)$: l'approximation uniforme [Alaoui and Mahoney, 2015] (notée par uniforme RLS dans nos figures), et l'approximation récursive [Musco and Musco, 2017] (notée par récursive RLS). Pour la dernière, nous utilisons le code fourni par les auteurs et disponible ici : `https://github.com/axelv/recursive-nystrom`. Nous nous comparons aussi à l'échantillonnage uniforme, où chaque donnée est tirée avec probabilité $\frac{1}{n}$. Dans toutes nos expériences, nous fixons $m = \sqrt{n}$ (taille des petites matrices dans l'algorithme DAC).

Pour comparer ces méthodes d'échantillonnage, nous calculons l'erreur comme la norme de Frobenius de la différence entre $K_{\mathcal{X}}$ et son approximation de Nyström $\hat{K}_{\mathcal{X}}$. Comme la plupart des jeux de données considérés sont très grands, nous suivons [Musco and Musco, 2017] et estimons l'erreur sur 10000 données choisies aléatoirement. Notons aussi que lorsque le nombre de données est trop grand, nous ne sommes pas en mesure de calculer les RLS exacts. Le seul jeu de données sur lequel nous pouvons calculer les RLS exacts est KC1.

Erreur d'approximation de la matrice de Gram

Nous traçons dans la Figure 3.2 la moyenne et la variance de la norme de Frobenius en fonction de $|\mathcal{S}|$ (nombre de données utilisées dans l'approximation de Nyström), calculées sur 10 tirages différents. Nous remarquons que pour tous les jeux de données, notre méthode a de meilleurs résultats d'approximation que les méthodes uniforme RLS et échantillonnage uniforme, et des résultats similaires à ceux de la méthode récursive. De plus, nous remarquons sur le jeu de donnée KC1 que lorsque $|\mathcal{S}|$ est très petit, toutes les méthodes d'approximation des RLS ont de meilleures performances que les RLS exacts. Cela n'est plus vrai dès que $|\mathcal{S}| \geq 10\%$ de n. Remarquons aussi que sur l'ensemble des jeux de données toutes les méthodes ont approximativement la même variance, sauf l'échantillonnage uniforme qui a une variance légèrement plus grande que les autres méthodes sur le jeu de données Adult. Une conclusion intermédiaire serait déjà de dire que si nous considérons de très petites valeurs de $|\mathcal{S}|$, alors il est plus judicieux d'utiliser des RLS approximés par des méthodes tel que uniforme RLS, DAC ou récursive plutôt que les RLS exacts.



(a) KC1 (b) Covertype

(c) Adult (d) Houses

FIGURE 3.2 – Erreur d'approximation de la matrice de Gram en fonction de $|\mathcal{S}|$, avec $m = \sqrt{n}$.

Temps de calcul

Afin de départager DAC et récursive, nous proposons d'étudier le temps de calcul nécessaire pour calculer les RLS approximés par chacune des méthodes considérées plus haut. Étant donné que les auteurs de [Musco and Musco, 2017] ont aussi proposé une version accélérée de la méthode récursive RLS, nous nous comparons aussi à cette dernière dans cette expérience (nous la nommerons récursive RLS accélérée dans notre figure). Ici aussi, nous utilisons le code fourni par les auteurs : https://github.com/axelv/recursive-nystrom. Les auteurs ont montré que la méthode récursive RLS accélérée permet d'améliorer le temps de cal-

cul par rapport à récursive RLS, mais pas la qualité d'approximation. C'est pour cela que nous n'avons pas considéré cette méthode dans l'expérience précédente.

Remarquons que nous ne considérons pas le temps de calcul de la méthode échantillonnage uniforme puisqu'elle peut être calculée en un temps constant.

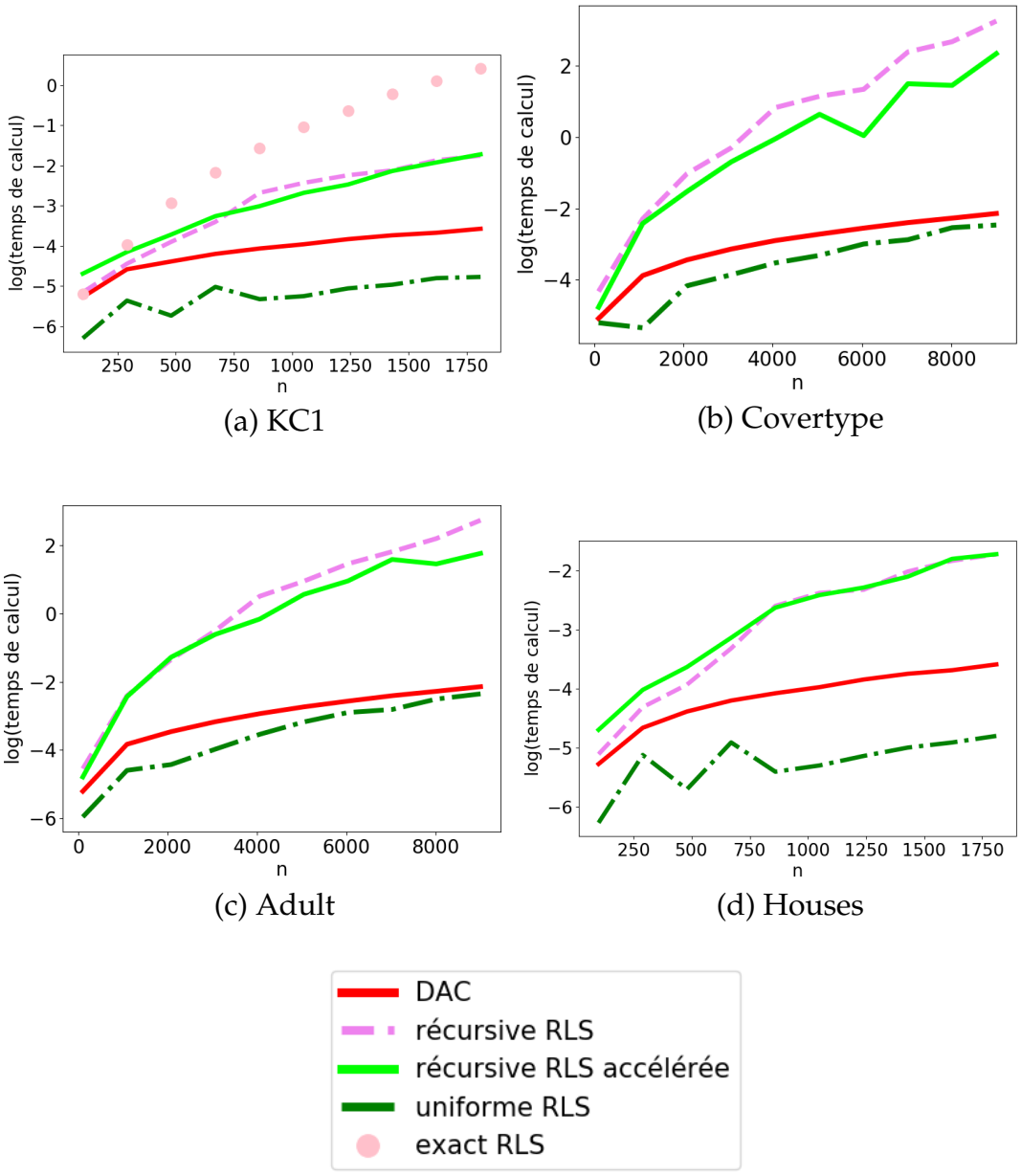
Enfin, notons que les implémentations de récursive RLS et récursive RLS accélérée proposées par les auteurs (https://github.com/axelv/recursive-nystrom) ne permettent pas d'atteindre de grandes valeurs de n. Pour cela, nous considérons deux expériences. Dans la première, nous comparons les méthodes : DAC, uniforme RLS, récursive RLS, récursive RLS accélérée et exacts RLS pour des valeurs de n relativement petites. Nous comparons ensuite uniquement DAC et uniforme RLS sur des valeurs de n plus grandes.

Dans un premier temps, nous commençons par comparer les cinq méthodes en considérant des valeurs de n relativement petites. Ainsi, nous traçons dans la Figure 3.3, le logarithme du temps de calcul des approximations des RLS en fonction de la taille du jeu de données n. Sans surprise, nous remarquons sur le jeu de donnée KC1 que le calcul des RLS exacts est le plus long. La méthode uniforme RLS est la plus rapide, tandis que récursive RLS et sa version accélérée sont plus lentes que DAC.

On en déduit que la méthode uniforme RLS est la plus rapide mais ne bénéficie pas de la meilleure qualité d'approximation. Les méthodes DAC et récursive RLS ont des performances d'approximation similaires mais DAC est plus rapide.

Nous considérons maintenant des valeurs de n plus grandes pour comparer notre méthode DAC et uniforme RLS. Pour cela, nous considérons nos deux plus grands jeux de données : adult et covertype, et nous traçons les résultats dans la Figure 3.4.

Nous remarquons maintenant que pour de petites valeurs de n ($n \leq 50000$ pour covertype et $n \leq 10000$ pour adult) notre méthode DAC est plus lente que uniforme RLS. Toutefois, en considérant de grandes valeurs de n (i.e. plus grandes que 100000, le temps de calcul de notre méthode DAC et celui de uniforme RLS sont les mêmes. Comme nous avons montré dans la Figure 3.2 que l'approximation de Nyström construite avec notre approximation des RLS (avec l'algorithme DAC) est meilleure en norme de Frobenius que celle construite par uniforme RLS, alors il est plus intéressant d'utiliser notre approximation lorsque le nombre de données est grand. Lorsque n est relativement petit (i.e. $n \leq 100000$), alors l'utilisation de uniforme RLS est plus intéressante.



(a) KC1

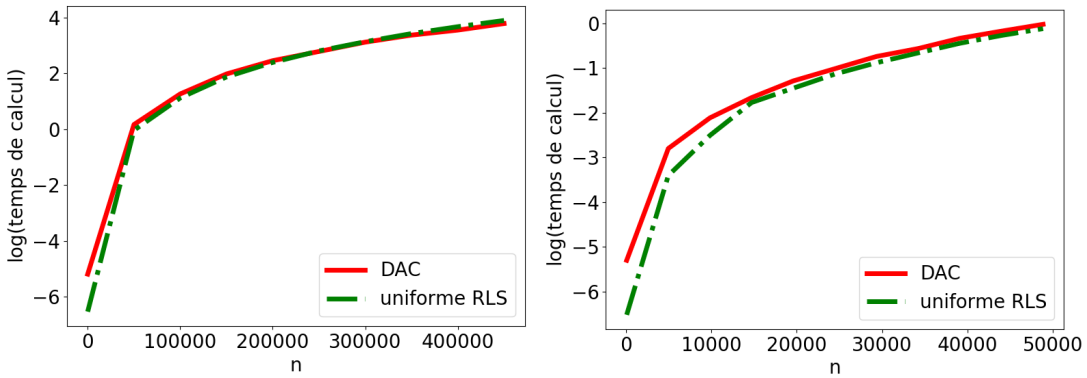
(b) Covertype

(c) Adult

(d) Houses

(e) Légende des méthodes

FIGURE 3.3 – Temps de calcul en fonction de la taille du jeu de données n, avec $m = \sqrt{n}$.



(a) Covertype (b) Adult

FIGURE 3.4 – Temps de calcul en fonction de la taille du jeu de données n, avec $m = \sqrt{n}$.

3.3.2 DAC-RLS pour la réduction du coût d'étiquetage

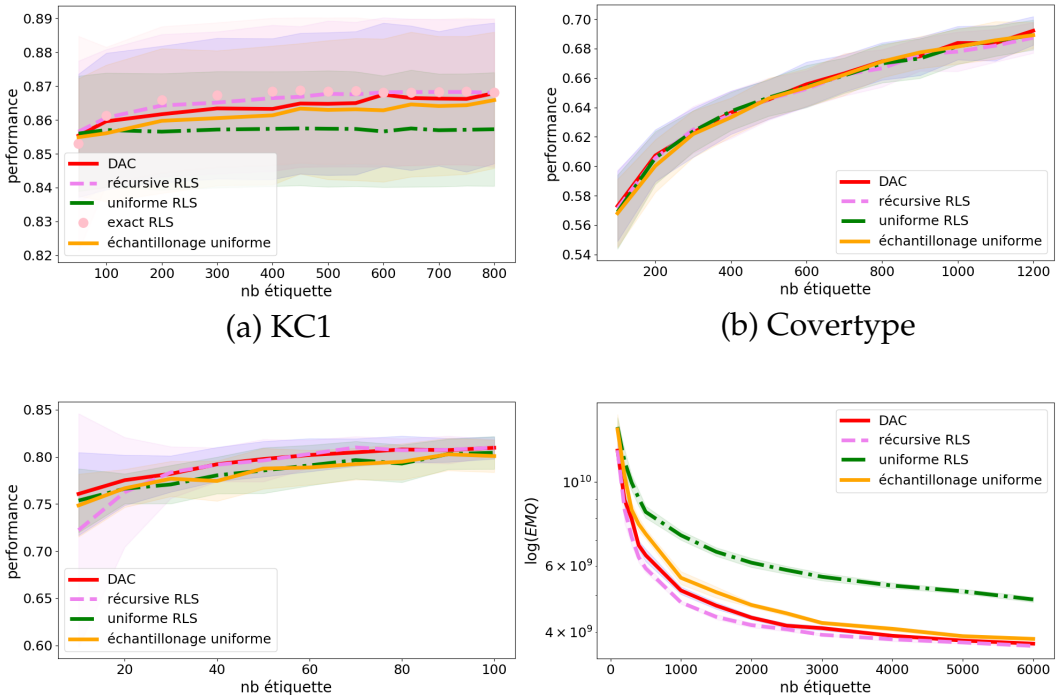
Lorsque nous considérons de grandes masses de données, il est fréquent que nous rencontrions des difficultés à toutes les étiqueter. Pour pallier ce problème, il existe plusieurs solutions, parmi elle, de sélectionner un sous-ensemble de données à étiqueter et à utiliser lors de l'apprentissage. Si le choix du sous-ensemble est bien fait, alors il permettra d'apprendre un modèle presque aussi bon que celui appris si nous avions eu accès à l'ensemble des étiquettes.

Nous proposons dans cette partie d'utiliser les approximations des RLS comme méthode d'échantillonnage des données à étiqueter et à utiliser pendant l'entraînement. Nous comparons notre algorithme DAC avec la méthode récursive et l'uniforme RLS. Nous considérons aussi l'échantillonnage uniforme comme méthode de référence. Comme dans la partie précédente, ces scores sont normalisés, pour former des probabilités. L'ensemble de données à étiqueter est choisi proportionnellement à ces scores, et est utilisé pour apprendre un SVM ou une KRR avec un noyau gaussien.

Nous affichons dans la Figure 3.5 la performance (le taux de bonne classification) en fonction du nombre de données étiquetées. Lorsqu'il s'agit d'un problème de régression (c'est le cas du jeu de données Houses), nous affichons l'Erreur Moyenne Quadratique (EMQ).

Nous remarquons que sur KC1 et Houses, la méthode récursive permet d'avoir les meilleures performances (juste après les RLS exacts dans le cas de KC1). Sur Adult, récursive RLS et DAC sont souvent équivalentes, mais

DAC a de meilleures performances lorsque le nombre d'étiquettes est très petit. Dans ce cas, puisque nous avons montré plus haut que DAC est plus rapide que récursive RLS, il est plus intéressant d'utiliser DAC. Néanmoins, sur Covertype, toutes les méthodes sont équivalentes. Dans ce cas, il est plus judicieux d'utiliser un échantillonnage uniforme étant donné qu'il est très facile à calculer. Remarquons aussi que sur le jeu de données Adult, la méthode DAC est celle qui donne les meilleures performances lorsque le nombre d'étiquettes est très petit.



(a) KC1 (b) Covertype

(c) Adult (d) Houses

FIGURE 3.5 – Performance d'un modèle appris sur un échantillon de données choisi proportionnellement aux RLS approximés.

Nous avons vu dans ces expériences que notre algorithme permet souvent d'atteindre des performances similaires à celles de la méthode récursive tout en étant plus rapide. Il permet d'obtenir de meilleures performances que la méthode uniforme RLS et la méthode d'échantillonnage uniforme mais est légèrement plus lent.

Nous en concluons que notre algorithme permet un bon compromis entre performance et temps de calcul tout en étant intuitif et très simple à implémenter.

3.4 Conclusion

Nous avons présenté dans ce chapitre un algorithme pour l'approximation des *Ridge Leverage Scores* (RLS) basé sur une approche diviser pour régner. Cet algorithme permet de réduire le temps de calcul de ces scores d'un temps cubique par rapport au nombre de données n à un temps $\Theta(nm)$, où m est un paramètre à régler par l'utilisateur. Nous avons théoriquement justifié l'utilité de ces RLS comme méthode d'échantillonnage pour l'approximation des matrices de Gram avec la méthode de Nyström. Enfin, nous avons expérimentalement montré que notre méthode propose le meilleur compromis entre qualité d'approximation de la matrice de Gram et temps de calcul par rapport aux méthodes de l'état de l'art.

Toutefois, ce travail soulève quelques questions que nous n'avons pas eu le temps d'aborder et il sera intéressant de s'y pencher. En particulier, nous avons théoriquement vu que le conditionnement de la matrice de Gram jouait un rôle important dans la qualité de l'approximation et sur le temps de calcul de l'approximation de Nyström. Une étude empirique sur l'effet de ce conditionnement serait intéressante. Par exemple, quel est le conditionnement des matrices de Gram sur les jeux de données de références avec des noyaux largement utilisés comme le noyau gaussien ou polynomial ? La question de comment choisir la bonne taille des sous-ensembles peut aussi se poser. Nous avons dans nos expériences utilisé $m = \sqrt{n}$, mais d'autres valeurs peuvent peut-être donner de meilleurs résultats.

Une prochaine étape pourrait aussi être l'étude de l'impact de l'approximation de la matrice de Gram à l'aide des RLS calculés avec DAC-RLS sur des modèles d'apprentissage.

Enfin, notre approximation des RLS pourrait être utilisée dans d'autres contextes d'échantillonnage comme la réduction de dimension. En effet, ces scores peuvent être utilisés pour définir une distribution de probabilité qui permet de choisir les attributs « importants » lorsque la dimension des données est trop grande.

Chapitre 4

RKHSs discrets pour la comparaison de distributions de probabilités et l'approximation de la MMD

Nous avons présenté dans le chapitre précédent une méthode d'approximation des *Ridge Leverage Scores* (RLS). Nous avons ensuite théoriquement et empiriquement évalué notre approximation pour la méthode de Nyström. Nous présentons maintenant un nouveau *framework* qui permet de représenter et de comparer des distributions de probabilités discrètes. Nous utilisons ensuite un lien entre notre *framework* et la *Maximum Mean Discrepency* (MMD) pour proposer une approximation de cette dernière.

Une grande partie des travaux présentés dans ce chapitre ont été soumis à la conférence ECML 2022.

4.1 Introduction

À travers les sections 2.1.1 et 3.1 nous avons abordé le concept des méthodes à noyaux [Hofmann et al., 2008, Manton and Amblard, 2015]. Nous avons vu que ces méthodes sont un puissant outil permettant de généraliser des méthodes linéaires à des problèmes non linéaires. Cela est rendu possible, en plongeant les données dans un *espace de représentation*, de dimension potentiellement infinie, qu'on appelle un espace de Hilbert à noyau reproduisant (RKHS).

Récemment, les auteurs de [Gretton et al., 2012] ont proposé d'étendre les méthodes à noyaux aux distributions de probabilités, et de construire

ainsi un *framework* pour représenter et comparer les distributions de probabilités dans les RKHS. Ils utilisent ensuite ce *framework* afin de développer un test leur permettant de vérifier si deux ensembles de données sont tirés suivant la même distribution. Ce test est basé sur une distance entre distributions de probabilités appelée la *Maximum Mean Discrepancy (MMD)*.

Toutefois, la majorité des travaux sur les méthodes à noyaux, y compris sur les distributions de probabilités, ne concerne que les RKHSs continus i.e. des RKHS sur des domaines continus. Ainsi, les méthodes à noyaux ont été utilisées pour représenter uniquement les distributions de probabilités continues dans des RKHSs continus.

Récemment, Jorgensen and Tian [Jorgensen and Tian, 2015] ont introduit et étudié les propriétés des RKHSs discrets i.e. des RKHSs sur des domaines discrets. Ils montrent notamment que les RKHSs discrets, contrairement aux RKHSs continus, contiennent les fonctions Diracs.

Les contributions de ce chapitre sont doubles. Dans un premier temps, nous proposons de suivre [Jorgensen and Tian, 2015] et [Gretton et al., 2012] et d'étudier le lien entre distribution de probabilité discrètes et méthodes à noyaux sur des domaines discrets. Comme les distributions de probabilités discrètes sont des combinaisons linéaires de fonction Diracs, et que ces dernières sont dans le RKHS, alors les distributions de probabilités discrètes sont aussi dans le RKHS discret. Nous nous baserons sur cette remarque pour présenter un nouveau *framework* pour représenter et comparer les distributions de probabilités dans les RKHSs discrets. Cela nous permettra de proposer une famille de distances entre distributions de probabilités. Nous verrons par la suite comment la MMD s'inscrit dans notre *framework*.

Dans un second temps, nous proposons une approximation de la MMD, dont il est connu que le calcul se fait en temps quadratique par rapport au nombre de données n (voir [Gretton et al., 2012] et section 4.2). Notre approximation, basée sur la méthode de Nyström et valable pour une large famille de fonctions noyau, permet de réduire cette complexité à $\Theta(ns)$, où $s \ll n$ est un paramètre à régler.

Le reste de ce chapitre est organisé comme suit : nous commençons par rappeler la MMD en section 4.2. Nous ferons aussi un point sur les différentes méthodes d'approximation de cette dernière dans cette même section. Nous introduisons ensuite formellement les RKHSs discrets en section 4.3. En section 4.4, nous présentons notre *framework* pour la représentation et la comparaison des distributions de probabilités dans les RKHSs discrets. Enfin, nous utiliserons notre *framework* pour proposer une approximation de la MMD en section 4.5, que nous évaluerons ensuite empiriquement en section 4.6.

4.2 Maximum Mean Discrepancy (MMD)

La *Maximum Mean Discrepancy (MMD)* est basée sur le plongement de distributions de probabilités dans des RKHSs [Muandet et al., 2017, Gretton et al., 2012]. Elle repose sur l'idée que deux distributions sont différentes si et seulement si leurs plongements dans un RKHS le sont aussi.

Pour formellement définir la MMD, supposons avoir accès à $\mathbb{X}$ un espace d'entrée, $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ un noyau continu et borné, i.e. $\sup_{x \in \mathbb{X}} k(x, x) < \infty$, et $\mathcal{H}$ le RKHS associé au noyau k.

Le plongement d'une distribution de probabilité $\mathbb{P}$ dans $\mathcal{H}$ (appelé aussi *mean embedding*) est défini comme :

$$\mu_{\mathbb{P}} := \int_{\mathbb{X}} k(x, .)\mathbb{P}(x) \in \mathcal{H}.$$

Étant donné un ensemble de points $\mathcal{X} := \{x_1 \dots, x_n\}$ tirés de manière i.i.d. selon $\mathbb{P}$ (i.e. pour tout i, $x_i \sim \mathbb{P}$), une estimation empirique de $\mu_{\mathbb{P}}$ peut être calculée :

$$\hat{\mu}_{\mathbb{P}} := \frac{1}{n} \sum_{x \in \mathcal{X}} k(x, .) \in \mathcal{H}. \tag{4.1}$$

La MMD entre deux distributions $\mathbb{P}$ et $\mathbb{Q}$ est définie comme la norme de la différence entre leurs plongements dans le RKHS $\mathcal{H}$ [Gretton et al., 2012] :

$$\text{MMD}_k(\mathbb{P}, \mathbb{Q}) := \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathcal{H}}.$$

La MMD n'est pas toujours une distance. En effet, les propriétés de positivité, de symétrie et d'inégalité triangulaire sont assurées par les propriétés de la norme dans $\mathcal{H}$. Toutefois, la propriété : $\text{MMD}_k(\mathbb{P}, \mathbb{Q}) = 0$ si seulement si $\mathbb{P} = \mathbb{Q}$ n'est vérifiée que si la fonction $\mu : \mathbb{P} \to \mu_{\mathbb{P}}$ est injective. Lorsque μ est injective, on dira que le noyau k est *caractéristique*. Ainsi, la MMD est une distance lorsque le noyau k est *caractéristique*.

En utilisant l'estimation empirique de $\mu_{\mathbb{P}}$, nous pouvons en déduire une estimation empirique de la MMD. Pour deux ensembles de données $\mathcal{X} := \{x_1 \dots, x_n\}$ et $\mathcal{Z} := \{z_1 \dots, z_m\}$ tirés respectivement suivant $\mathbb{P}$ et $\mathbb{Q}$, une estimation empirique de $\text{MMD}_k(\mathbb{P}, \mathbb{Q})$ est définie comme :

$$\text{MMD}_k(\hat{\mathbb{P}}, \hat{\mathbb{Q}})^2 := \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} k(x_i, x_j) + \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} k(z_i, z_j) - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, z_j), \tag{4.2}$$

où $\hat{\mathbb{P}}$ et $\hat{\mathbb{Q}}$ sont les distributions empiriques de $\mathbb{P}$ et $\mathbb{Q}$ à support dans $\mathcal{X}$ et $\mathcal{Z}$.

La distance MMD a eu beaucoup de succès en apprentissage automatique et a été utilisée dans plusieurs applications comme l'adaptation de domaine [Zhang and Wang, 2020], l'apprentissage actif [Liu et al., 2015], pour résoudre le *two sample problem* [Borgwardt et al., 2006, Gretton et al., 2012], et comme fonction de perte dans l'optimisation des réseaux de neurones génératifs (GAN) [Dziugaite et al., 2015].

Malheureusement, le calcul de la MMD se fait en temps quadratique par rapport au nombre de données, $\Theta\left((n+m)^2\right)$, soit $\Theta\left(n^2\right)$ lorsque m et n sont du même ordre de grandeur. Notons ici que nous avons supposé que l'évaluation de k se fait en $\Theta(1)$.

Plusieurs méthodes ont donc été proposées pour réduire ce temps de calcul [Gretton et al., 2012, Zaremba et al., 2013, Zhao and Meng, 2015].

La première, et la plus intuitive, est celle qu'on appellera *MMD linéaire* [Gretton et al., 2012]. Elle a été proposée afin de résoudre le *two sample problem* dont le but est, étant donné deux ensembles de points, vérifier s'ils ont été tirés suivant la même distribution. Cette méthode commence par aléatoirement choisir un sous ensemble de taille $\sqrt{n}$ dans $\mathcal{X}$, et un sous ensemble de taille $\sqrt{m}$ dans $\mathcal{Z}$, puis approxime la MMD entre $\mathcal{X}$ et $\mathcal{Z}$ par la MMD entre ces deux sous-ensembles. Ainsi, la complexité globale de cette méthode d'approximation est de $\Theta\left(\left(\sqrt{n}+\sqrt{m}\right)^2\right)$, qui est équivalente à $\Theta(n)$ lorsque m et n sont du même ordre de grandeur.

La méthode *MMD bloc*, proposée par [Zaremba et al., 2013], est une amélioration de la méthode *MMD linéaire*. Au lieu de calculer la MMD sur un unique sous-ensemble, *MMD bloc* choisit plusieurs sous-ensembles, chacun de taille s, et approxime la MMD entre $\mathcal{X}$ et $\mathcal{Z}$ par la moyenne des MMD calculées sur ces sous-ensembles de taille s. La méthode de *MMD bloc* est plus coûteuse que *MMD linéaire* mais reste moins coûteuse que la MMD exacte. En effet, en supposant que le nombre de bloc est de $\frac{n}{s}$, la complexité globale de *MMD bloc* est de $\Theta\left(\frac{n}{s}s^2\right) = \Theta(ns)$. Les auteurs de [Zaremba et al., 2013] prouvent que leur approximation converge en distribution vers une gaussienne centrée en la valeur de la MMD exacte :

$$\mathrm{MMD}_k^{\mathrm{bloc}}\left(\hat{\mathbb{P}},\hat{\mathbb{Q}}\right)^2 \to \mathcal{N}\left(\mathrm{MMD}_k\left(\hat{\mathbb{P}},\hat{\mathbb{Q}}\right)^2, \frac{\sigma^2}{n+m}\right),$$

où σ est une constante, et $\mathrm{MMD}_k^{\mathrm{bloc}}\left(\hat{\mathbb{P}},\hat{\mathbb{Q}}\right)$ est une approximation de $\mathrm{MMD}_k\left(\hat{\mathbb{P}},\hat{\mathbb{Q}}\right)$ construite par la méthode *MMD bloc*. Ainsi, en considérant un très grand nombre de tirages de $\hat{\mathbb{P}}$ et $\hat{\mathbb{Q}}$, l'approximation construite par *MMD bloc* converge vers la MMD exacte et cela indépendamment de la taille des blocs s. Les auteurs de [Zaremba et al., 2013] montrent ensuite

expérimentalement que *MMD bloc* permet de résoudre le *two sample problem* plus efficacement (i.e. en faisant moins d'erreurs) qu'avec *MMD linéaire*.

Enfin, les auteurs de [Zhao and Meng, 2015] proposent une méthode différente des deux précédentes, en approximant directement la fonction noyau. Ainsi, ils proposent d'utiliser les *Random Fourier Features* (RFF) [Rahimi and Recht, 2007] pour approximer le noyau k par $\hat{k}$ tel que $\hat{k}(x,z) := \langle \hat{\phi}(x), \hat{\phi}(z) \rangle$, où

$$\hat{\phi}(x) := \sqrt{\frac{2}{s}} \left(\cos(\langle x, w_1, \rangle), \ldots, \cos(x, w_s), \sin(\langle x, w_1, \rangle), \ldots, \sin(x, w_s)\right),$$

et où $w_1, \ldots, w_s$ sont des vecteurs aléatoires tirés suivant une distribution de probabilité appropriée. Les auteurs utilisent ensuite des formules trigonométriques afin d'accélérer le calcul de la MMD approximée, qui se fait désormais en $\Theta(ns)$. De plus, ils montrent que leur approximation $\text{MMD}_{\hat{k}}$ converge vers la MMD exacte :

$$\Pr\left[\sup_{\hat{\mathbb{P}},\hat{\mathbb{Q}}} |\text{MMD}_k(\hat{\mathbb{P}},\hat{\mathbb{Q}})^2 - \text{MMD}_{\hat{k}}(\hat{\mathbb{P}},\hat{\mathbb{Q}})^2| \geq \epsilon\right] \leq \Theta\left(e^{-s\epsilon}\right),$$

où $\Pr[a]$ est la probabilité que l'évènement a se produise et où $\text{MMD}_{\hat{k}}$ est la MMD calculée avec le noyau approximé par la méthode des RFF. Ainsi, lorsque le nombre de *bases* s est grand, la probabilité que l'approximation de la MMD s'éloigne d'une distance ϵ de la MMD exacte devient de plus en plus petite. Malheureusement, l'approximation proposée par [Zhao and Meng, 2015] n'est valable que pour des noyaux invariants par translation.

Nous proposons dans la section 4.5 une méthode d'accélération du calcul de la MMD basée sur une approximation de la fonction noyau construite avec la méthode de Nyström valable pour toute fonction noyau, y compris les noyaux sur graphes.

Il est important de noter que la difficulté majeure dans les accélérations de la MMD basées sur une approximation de la fonction noyau avec les RFF ou Nyström est que la fonction qui à $\mathbb{P}$ associe $\tilde{\mu}_{\mathbb{P}} := \frac{1}{n}\sum_{x\in\mathcal{X}} \hat{k}(x,.)$ n'est plus injective. Dans ce cas, la MMD calculée avec de tel noyaux approximés n'est plus une distance.

Toutefois, nous verrons dans la suite de ce chapitre comment utiliser notre *framework* pour démontrer que notre approximation basée sur la méthode de Nyström reste une distance.

4.3 RKHSs discrets

Nous résumons dans cette section les principaux résultats de [Jorgensen and Tian, 2015] sur les RKHSs discrets. La Définition 12 est la seule notion nouvelle dans cette section qui n'a pas été introduite dans [Jorgensen and Tian, 2015] et qui fait partie de nos contributions.

Définition 11 (Noyau (semi)-défini positif sur un domaine discret [Jorgensen and Tian, 2015]). *Soit $V := \{a_1, a_2, \ldots\}$ un ensemble dénombrable et potentiellement infini, et $\mathcal{P}(V)$ l'ensemble de tous les sous-ensembles finis de V. Une fonction $k : V \times V \to \mathbb{R}$ est un noyau semi-défini positif si pour tout $\mathcal{F} := \{x_1, \ldots, x_n\} \in \mathcal{P}(V)$ et pour tout ensemble de coefficients $\{c_i\}_{i\in[n]}$, on a :*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0. \tag{4.3}$$

Le noyau k est défini positif si :

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j) = 0 \Longrightarrow c_i = 0, \forall i.$$

Dans la définition suivante nous proposons une nouvelle catégorie de fonction noyau, celle de noyau λ-défini positif.

Définition 12 (Noyau λ-défini positif sur un domaine discret). *Soit $V := \{a_1, a_2, \ldots\}$ un ensemble dénombrable et potentiellement infini, et $\mathcal{P}(V)$ l'ensemble de tous les sous-ensembles finis de V. Une fonction $k : V \times V \to \mathbb{R}$ est un noyau λ-défini positif si il existe un $\lambda > 0$ tel que pour tout $\mathcal{F} := \{x_1, \ldots, x_n\} \in \mathcal{P}(V)$, pour tout ensemble de coefficients $\{c_i\}_{i\in[n]}$ on a :*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq \lambda \|c\|_2^2,$$

où $c := (c_1, \ldots, c_n)$ est le vecteur des coefficients $\{c_i\}_{i\in[n]}$.

Dans ce chapitre, nous considérons uniquement le cas où le noyau k est λ-défini positif. Cette condition implique que le spectre de $K_{\mathcal{F}}$ (i.e. l'ensemble de ses valeurs propres) est dans $[\lambda, +\infty]$ pour tout $\mathcal{F} := \{x_1, \ldots, x_n\} \in \mathcal{P}(V)$. Dans ce cas, il est assez trivial de voir que $K_{\mathcal{F}}^{-1}[i,i] \leq \lambda^{-1}$ pour tout $i \in [n]$. En effet :

$$K_{\mathcal{F}}^{-1}[i,i] = e_i^T K_{\mathcal{F}}^{-1} e_i = \langle e_i, K_{\mathcal{F}}^{-1} e_i \rangle \leq \|e_i\|_2^2 \left\|K_{\mathcal{F}}^{-1}\right\|_{2,2}^2 = \sigma_{max}\left(K_{\mathcal{F}}^{-1}\right) \leq \lambda^{-1},$$

où e_i est le i^{eme} élément de la base canonique de $\mathbb{R}^n$.

Nous définissons dans ce qui suit les RKHSs sur des domaines discrets.

Définition 13 (RKHS sur des domaines discrets [Jorgensen and Tian, 2015]). *Soit V un ensemble dénombrable et potentiellement infini. Un espace de Hilbert $\mathcal{H}$ de fonctions de V dans $\mathbb{R}$ est un espace de Hilbert à noyau reproduisant (RKHS) s'il existe un noyau semi-défini positif $k : V \times V \to \mathbb{R}$ tel que :*

— *la fonction $k(x, .) : V \to \mathbb{R}$ est dans $\mathcal{H}$ pour tout $x \in V$,*

— *pour toute fonction $f \in \mathcal{H}$ et $x \in V$: $\langle k(x, .), f \rangle_{\mathcal{H}} = f(x)$.*

Les Définitions 11 et 13 ne sont pas nouvelles et il est connu que la théorie des noyaux est valide autant pour les ensembles continus que pour les ensembles discrets. Ce qui est toutefois nouveau, c'est le résultat suivant qui caractérise les RKHSs discrets, et qui stipule que ces derniers contiennent les fonctions Dirac.

Définition 14 (Propriété des masses discrètes [Jorgensen and Tian, 2015]). *Le RKHS $\mathcal{H}$ défini sur un ensemble infini et discret V est dit avoir la propriété des masses discrètes, si $\delta_a(.) \in \mathcal{H}$ pour tout $a \in V$, où $\delta_a(.)$ est la masse de Dirac au point a i.e. :*

$$\delta_a(x) = \begin{cases} 1 & \text{si } a = x \\ 0 & \text{sinon.} \end{cases}$$

Lorsque $\mathcal{H}$ a la propriété des masses discrètes, il est alors appelé un RKHS discret.

Le théorème suivant énonce une condition nécessaire et suffisante pour caractériser les points de V dont les fonctions Dirac associées sont dans le RKHS.

Théorème 2 (condition nécessaire et suffisante pour que $\delta_a(.) \in \mathcal{H}$ [Jorgensen and Tian, 2015]). *Soit k un noyau semi-défini positif sur un ensemble discret dénombrable et potentiellement infini V, et soit $\mathcal{H}$ le RKHS associé. Soit $i \in \mathbb{N}$ fixé, et $\mathcal{P}(V)$ l'ensemble de tous les sous-ensembles finis de V. Alors, $\delta_{a_i}(.) \in \mathcal{H}$ si et seulement si :*

$$\sup_{\mathcal{F} \in \mathcal{P}(V) :\ a_i \in \mathcal{F}} K_{\mathcal{F}}^{-1}[j_i, j_i] < \infty, \tag{4.4}$$

où $K_{\mathcal{F}} := (k(x, z))_{x,z \in \mathcal{F}}$ est la matrice de Gram de k sur $\mathcal{F}$, et j_i est la position de a_i dans $\mathcal{F}$. Dans ce cas nous avons :

$$\|\delta_{a_i}(.)\|_{\mathcal{H}}^2 = \sup_{\mathcal{F} \in \mathcal{P}(V) :\ a_i \in \mathcal{F}} K_{\mathcal{F}}^{-1}[j_i, j_i].$$

Lorsque ces fonctions Dirac sont dans le RKHS, il est possible de caractériser leur projection orthogonale sur l'espace vectoriel engendré par les fonctions $\{k(a, .)\}$.

Lemme 1 (Projection orthogonale de fonctions Dirac [Jorgensen and Tian, 2015]). *Soit k et $\mathcal{H}$ comme précédemment, $\mathcal{F} := \{x_1, \ldots, x_n\} \in \mathcal{P}(V)$, et $\mathcal{H}_\mathcal{F} := \text{span}\left(\{k(x,.)\}\right)_{x\in\mathcal{F}}$. Soit $\mathrm{P}_\mathcal{F} : \mathcal{H} \to \mathcal{H}_\mathcal{F}$ la projection orthogonale sur $\mathcal{H}_\mathcal{F}$. Pour un $i \in [n]$ tel que $\delta_{x_i}(.) \in \mathcal{H}$, on a :*

$$\mathrm{P}_\mathcal{F}\left(\delta_{x_i}(.)\right) := \sum_{j=1}^{n} K_\mathcal{F}^{-1}[i,j]k(x_j,.).$$

Comme dit plus haut, nous considérons ici uniquement le cas où le noyau k est λ-défini positif. Dans ce cas, l'ensemble des valeurs propres de $K_\mathcal{F}$ est dans $[\lambda, +\infty]$ pour tout $\mathcal{F} := \{x_1, \ldots, x_n\} \in \mathcal{P}(V)$, et l'Hypothèse (4.4) du Théorème 2 est toujours vérifiée, et toutes les fonctions Dirac sont dans le RKHS.

Notons toutefois que cette hypothèse n'est pas très restrictive. En effet, à partir de tout noyau $\hat{k}$ défini positif, il est possible de définir un noyau k λ-défini positif : $k(x,z) = \hat{k}(x,z) + \lambda\delta_x(z), \forall x, z \in V$.

4.4 Comparer des distributions dans les RKHSs discrets

Comme les fonctions Dirac sont des éléments du RKHS, cela permet une manière naturelle de traiter les distributions de probabilités discrètes.

En effet, en considérant une distribution discrète $\hat{\mathbb{P}}$ à support dans un ensemble de données $V := \{a_1, a_2, \ldots\}$, elle peut être exprimée comme :

$$\hat{\mathbb{P}} := \sum_{i\in\mathbb{N}} p_i\delta_{a_i}(.),$$

où $\delta_{a_i}(.)$ est la fonction Dirac associée au point a_i et $p_i \in [0,1]$ est la probabilité associée.

Ainsi, puisque les fonctions Dirac sont dans le RKHS $\mathcal{H}$, et puisque les distributions de probabilités discrètes $\hat{\mathbb{P}}$ sont des combinaisons linéaires de fonctions Dirac, alors ces dernières sont aussi dans le RKHS, i.e. $\hat{\mathbb{P}} \in \mathcal{H}$ (voir Figure 4.1). Nous pouvons donc manipuler ces distributions de probabilités discrètes à l'aide des outils des RKHSs.

Dans la suite de cette section, nous nous basons sur cette remarque pour définir une distance entre distributions discrètes.

Une première manière de comparer les distributions discrètes est d'utiliser le produit scalaire dans $\mathcal{H}$. Plus formellement, soit V un ensemble

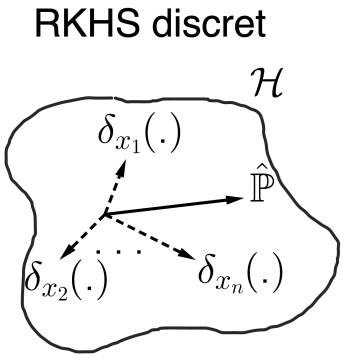


FIGURE 4.1 – Illustration d'une Distribution discrète dans le RKHS : comme les fonctions Dirac (représentées ici par des vecteurs en pointillés), sont des éléments du RKHS discret $\mathcal{H}$, et comme $\hat{\mathbb{P}}$ est une combinaison linéaire de ces Dirac, alors $\hat{\mathbb{P}}$ est aussi dans le RKHS discret $\mathcal{H}$.

dénombrable et potentiellement infini, $\mathcal{X} = \{x_i\}_{i=1}^n$ et $\mathcal{Z} = \{z_i\}_{i=1}^m$ deux sous-ensembles de V, $\hat{\mathbb{P}} = \sum_{i=1}^n p_i \delta_{x_i}(.)$ et $\hat{\mathbb{Q}} = \sum_{i=1}^m q_i \delta_{z_i}(.)$ deux distributions discrètes. Ainsi, comme $\hat{\mathbb{P}} \in \mathcal{H}$ et $\hat{\mathbb{Q}} \in \mathcal{H}$, nous pouvons définir la distance entre elles comme : $\left\|\hat{\mathbb{P}} - \hat{\mathbb{Q}}\right\|_{\mathcal{H}}$. Même si cela semble logique, cette valeur n'est pas explicitement calculable. En effet, comme :

$$\begin{aligned}
\left\|\hat{\mathbb{P}} - \hat{\mathbb{Q}}\right\|_{\mathcal{H}}^2 &= \langle \hat{\mathbb{P}}, \hat{\mathbb{P}} \rangle_{\mathcal{H}} + \langle \hat{\mathbb{Q}}, \hat{\mathbb{Q}} \rangle_{\mathcal{H}} - 2\langle \hat{\mathbb{P}}, \hat{\mathbb{Q}} \rangle_{\mathcal{H}} \\
&= \sum_{i=1}^n \sum_{j=1}^n p_i p_j \langle \delta_{x_i}(.), \delta_{x_j}(.) \rangle_{\mathcal{H}} + \sum_{i=1}^m \sum_{j=1}^m q_i q_j \langle \delta_{z_i}(.), \delta_{z_j}(.) \rangle_{\mathcal{H}} \\
&\quad - 2 \sum_{i=1}^n \sum_{j=1}^m p_i q_j \langle \delta_{x_i}(.), \delta_{z_j}(.) \rangle_{\mathcal{H}},
\end{aligned}$$

le calcul de $\left\|\hat{\mathbb{P}} - \hat{\mathbb{Q}}\right\|_{\mathcal{H}}$ requiert alors l'évaluation du produit scalaire entre les fonctions Dirac, qui n'est pas connu. Toutefois, le produit scalaire dans $\mathcal{H}$ entre les fonctions noyaux est quant à lui connu. L'idée est alors d'introduire un opérateur linéaire O_k qui projette les fonctions Dirac sur l'espace vectoriel généré par les fonctions noyaux : span $(\{k(x,.)\}_{x \in V})$, et de définir ensuite une distance entre $\hat{\mathbb{P}}$ et $\hat{\mathbb{Q}}$ paramétrée par O_k.

Afin de définir un tel opérateur O_k ainsi qu'une distance entre distributions discrètes, définissons d'abord $\mathcal{D}_V :=$ span $(\{\delta_x(.)\}_{x \in V})$ l'espace vectoriel généré par les fonctions Dirac associées aux éléments de V. Notons Null $(O_k) := \{v \in \mathcal{D}_V \mid O_k(v) = 0\}$ le *null space* de l'opérateur O_k (nous préférons éviter l'utilisation du terme français « noyau » afin de ne pas confondre avec les fonctions noyaux k).

Définition 15 (Distance entre distributions discrètes dans le RKHS discret)**.** *Soit $\mathcal{H}$ un RKHS discret de fonctions défini sur un ensemble dénombrable et infini V. Soit $O_k : \mathcal{D}_V \to \text{span}\left(\{k(x,.)\}_{x\in V}\right)$ un opérateur linéaire avec $\text{Null}\left(O_k\right) = \{0\}$. On définit la distance entre les distributions discrètes $\hat{\mathbb{P}}$ et $\hat{\mathbb{Q}}$ dans le RKHS discret comme :*

$$D_{O_k}\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right) := \left\|\hat{\mathbb{P}} - \hat{\mathbb{Q}}\right\|_{O_k} = \left\|O_k\left(\hat{\mathbb{P}} - \hat{\mathbb{Q}}\right)\right\|_{\mathcal{H}} \tag{4.5}$$

L'opérateur D_{O_k} est induit par la forme bilinéaire $\langle .,. \rangle_{O_k} := \langle O_k\left(.\right), O_k\left(.\right)\rangle_{\mathcal{H}}$ définie sur $\mathcal{D}_V \times \mathcal{D}_V$. En réalité, D_{O_k} est une distance car la forme bilinéaire est un produit scalaire : elle est symétrique et semi-définie positive grâce à la linéarité de O_k, elle est aussi définie positive (i.e. $\langle u, u\rangle_{O_k} \geq 0\ \forall u$, et $\langle u, u\rangle_{O_k} = 0 \implies u = 0$) parce que le null space de O_k est réduit à zéro (i.e. $O_k(u) = 0 \implies u = 0$).

Ainsi, l'ensemble des opérateurs linéaires O_k de *null space* réduit à zéro définit une famille de distances. Une question importante se pose alors : comment pouvons-nous choisir un tel opérateur ? Dans la suite de cette section nous présentons une famille d'opérateurs linéaires O_k basés sur la projection orthogonale des fonctions Dirac introduite dans le Lemme 1. Ces projections nous garantiront que les distances associées à O_k sont bien explicitement calculables à l'aide des fonctions noyaux. Nous verrons aussi qu'un cas particulier de cette famille de distance se trouve être la MMD. Nous laisserons toutefois l'étude des autres choix possibles de O_k pour des travaux futurs.

Soit $V := \{a_1, a_2, \ldots\}$ un ensemble dénombrable et potentiellement infini comme défini plus haut. Nous proposons donc de nous intéresser à l'opérateur O_k qui projette les éléments de $\mathcal{D}_V$, qui est un sous espace de $\mathcal{H}$, dans $\text{span}\left(\{k(x,.)\}_{x\in V}\right)$ en utilisant la projection orthogonale définie dans le Lemme 1 :

$$\begin{aligned} O_k : \mathcal{D}_V &\to \text{span}\left(\{k(x,.)\}_{x\in V}\right) \\ u &\mapsto O_k(u) := \sum_{i\in\mathbb{N}} c_i \mathrm{P}_{\{a_i\}}(u) \text{ avec } c_i \in \mathbb{R}\backslash\{0\} \forall i, \end{aligned} \tag{4.6}$$

où $\mathrm{P}_{\{a\}}(u)$ est la projection orthogonale de u sur $k(a,.)$.

Avant d'aller plus loin, montrons que l'opérateur O_k défini dans l'Équation (4.6) est un opérateur bien défini dans $\mathcal{D}_V$, c'est-à-dire que la somme $\sum_{i\in\mathbb{N}} c_i \mathrm{P}_{\{a_i\}}(u)$ est convergente. Pour faire cela, nous allons explicitement calculer la valeur de $O_k(u)$.

Lemme 2. *Soit $V := \{a_1, a_2, \ldots\}$, k et $\mathcal{H}$ comme ci-dessus. On a :*

— *pour tout* $a, b \in V$ *:*

$$\mathrm{P}_{\{b\}}(\delta_a(.)) = \begin{cases} \frac{1}{k(a,a)} k(a,.) & \text{si } a = b \\ 0 & \text{sinon.} \end{cases}$$

— *pour tout* $u := \sum_{i=1}^{n} \alpha_i \delta_{x_i}(.) \in \mathcal{D}_V$*, on a :*

$$O_k(u) = \sum_{i=1}^{n} \frac{\alpha_i c_i}{k(x_i, x_i)} k(x_i, .) \in \operatorname{span}\left(\{k(a,.)\}_{a \in V}\right).$$

Démonstration. Remarquons que pour tout $a, b \in V$, comme $\langle \delta_a(.), k(b,.) \rangle_{\mathcal{H}} = \delta_a(b)$, nous avons :

$$\mathrm{P}_{\{b\}}(\delta_a(.)) = \frac{\langle \delta_a(.), k(b,.) \rangle_{\mathcal{H}}}{\|k(b,.)\|^2_{\mathcal{H}}} k(b,.) = \frac{\delta_a(b)}{k(b,b)} k(b,.).$$

On a alors : si $a \neq b$: $\delta_a(b) = 0$, et dans ce cas $\mathrm{P}_{\{b\}}(\delta_a(.)) = 0$. Lorsque $a = b$: $\delta_a(a) = 1$ et alors $\mathrm{P}_{\{a\}}(\delta_a(.)) = \frac{1}{k(a,a)} k(a,.)$, ce qui prouve le premier point de notre théorème.

Passons maintenant au second point. En utilisant le précédent résultat, nous pouvons conclure que, pour tout $i \in [n]$:

$$\sum_{j \in [n]} c_j \mathrm{P}_{\{a_j\}}(\delta_{a_i}(.)) = c_i \mathrm{P}_{\{a_i\}}(\delta_{a_i}(.)).$$

Ainsi en utilisant le premier point de ce théorème nous avons :

$$O_k(\delta_{a_i}(.)) = \sum_{j \in \mathbb{N}} c_j \mathrm{P}_{\{a_j\}}(\delta_{a_i}(.)) = c_i \mathrm{P}_{\{a_i\}}(\delta_{a_i}(.)) = \frac{c_i}{k(a_i, a_i)} k(a_i, .).$$

De la même manière, pour $u := \sum_{i=1}^{n} \alpha_i \delta_{a_i}(.) \in \mathcal{D}_V$, nous avons :

$$\mathrm{P}_{\{a_j\}}(u) = \sum_{i=1}^{n} \alpha_i c_i \mathrm{P}_{\{a_j\}}\left(\delta_{a_i}(.)\right) = \frac{\alpha_j c_j}{k(a_j, a_j)} k(a_j, .).$$

On en déduit enfin que :

$$O_k(u) = \sum_{j \in \mathbb{N}} c_j \mathrm{P}_{\{a_j\}}(u) = \sum_{j=1}^{n} \frac{\alpha_j c_j}{k(a_j, a_j)} k(a_j, .).$$

□

Pour s'assurer que D_{O_k} est une distance, il suffit maintenant de vérifier deux choses : que $\text{Null}(O_k) = \{0\}$, et que O_k est linéaire. Faisons cela dans le lemme suivant.

Lemme 3. *Soit V, k et $\mathcal{H}$ comme ci-dessus, et soit O_k tel que défini dans l'Équation (4.6). On a alors que O_k est linéaire et* $\text{Null}(O_k) = \{0\}$.

Démonstration. Soit $u := \sum_{i=1}^{n} \alpha_i \delta_{x_i}(.) \in \mathcal{D}_V$. En utilisant le Lemme 2, nous avons :

$$\|O_k(u)\|_{\mathcal{H}}^2 = \left\| \sum_{i\in[n]} \frac{\alpha_i c_i}{k(x_i,x_i)} k(x_i,.) \right\|_{\mathcal{H}}^2 = \sum_{i\in[n]} \sum_{j\in[n]} \frac{\alpha_i c_i}{k(x_i,x_i)} \frac{\alpha_j c_j}{k(z_j,z_j)} k(x_i,z_j),$$

et comme le noyau k est défini positif :

$$O_k(u) = 0 \Longrightarrow \frac{\alpha_i c_i}{k(x_i,x_i)} = 0 \; \forall i.$$

Enfin, comme c_i est supposé différent de zéro pour tout i, cela implique que α_i est égal à zéro pour tout i. On en conclut ainsi que $u = 0$, et donc que $\text{Null}(O_k) = \{0\}$.

Passons maintenant à la linéarité de O_k. Soit $u := \sum_{i=1}^{n} \alpha_i \delta_{a_i}(.)$, $v := \sum_{j=1}^{m} \beta_j \delta_{a_j}(.)$ et $\gamma \in \mathbb{R}$. En utilisant la linéarité de la projection orthogonale, et le fait que la somme infinie $\sum_{j\in[n]} c_j \mathrm{P}_{\{a_j\}}(\delta_{a_i}(.))$, n'est en réalité pas infinie puisqu'elle se réduit à un seul terme, nous avons :

$$\begin{aligned} O_k(u+\gamma v) &= \sum_{i\in\mathbb{N}} c_i \mathrm{P}_{\{a_i\}}(u+\gamma v) = \sum_{i\in\mathbb{N}} c_i \mathrm{P}_{\{a_i\}}(u) + \gamma \sum_{i\in\mathbb{N}} c_i \mathrm{P}_{\{a_i\}}(v) \\ &= \sum_{j=1}^{n} \frac{\alpha_i c_i}{k(a_i,a_i)} k(a_i,.) + \gamma \sum_{j=1}^{m} \frac{\alpha_j c_j}{k(a_j,a_j)} k(a_j,.) \\ &= O_k(u) + \gamma O_k(v). \end{aligned}$$

où la deuxième égalité est vraie par le Lemme 2. □

Ce résultat prouve que la famille d'opérateurs introduite dans L'Équation (4.6) définit une famille de distances D_{O_k} entre distribution de probabilités discrètes. Ainsi, à chaque fois qu'une nouvelle famille de paramètres strictement positifs $\{c_i\}_{i\in\mathbb{N}}$ est définie, cela induit une nouvelle distance.

Un autre résultat important est que la distance MMD, présentée en section 4.2, est une instance de cette famille de distances. En effet, nous verrons dans le théorème suivant que pour une définition particulière des paramètres $\{c_i\}_{i\in\mathbb{N}}$, la distance D_{O_k} associée est la MMD.

Théorème 3. *Soit* $V := \{a_1, a_2, \ldots\}$, k, $\mathcal{H}$ *et* $\mathcal{F} := \{x_1, \ldots, x_n\} \in \mathcal{P}(V)$ *comme ci-dessus,* $\hat{\mathbb{P}} = \sum_{i=1}^{n} p_i \delta_{x_i}(.)$ *et* $\hat{\mathbb{Q}} = \sum_{i=1}^{n} q_i \delta_{x_i}(.)$ *deux distributions de probabilités discrètes, et enfin soit :*

$$O_k : u \mapsto \sum_{a \in V} k(a,a) \mathrm{P}_{\{a\}}(u).$$

On a alors :

$$\mathrm{MMD}_k\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right) = D_{O_k}\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right).$$

Démonstration. Comme pour tout $a \in V : k(a,a) \neq 0$, alors l'opérateur $O_k : u \mapsto \sum_{a \in V} k(a,a) \mathrm{P}_{\{a\}}(u)$ respecte les hypothèses de l'Équation (4.6) et il définit une distance D_{O_k}. Il est aussi facile de voir que $O_k\left(\hat{\mathbb{P}}\right) = \sum_{i=1}^{n} p_i k(x_i, .)$ et $O_k\left(\hat{\mathbb{Q}}\right) = \sum_{i=1}^{n} q_i k(x_i, .)$. Ainsi, O_k fait correspondre une distribution $\hat{\mathbb{P}} \in \mathcal{H}$ avec son *mean embedding* $\hat{\mu}_{\hat{\mathbb{P}}}$ tel que défini dans l'Équation (4.1). On a alors :

$$D_{O_k}\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right) = \left\| O_k\left(\hat{\mathbb{P}} - \hat{\mathbb{Q}}\right) \right\|_{\mathcal{H}} = \left\| \hat{\mu}_{\hat{\mathbb{P}}} - \hat{\mu}_{\hat{\mathbb{Q}}} \right\|_{\mathcal{H}} = \mathrm{MMD}\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right).$$

□

Le Théorème 3 prouve que la distance MMD est un cas particulier de la famille des distances entre probabilités discrètes dans les RKHSs discrets présentée dans la Définition 15. Ce résultat justifie l'utilisation de la MMD avec une large famille de fonction noyau, dès lors qu'ils sont λ définis positifs (voir Définition 12).

Dans la section suivante, nous présentons une approximation de la MMD basée sur la méthode de Nyström. Cette approximation permet de réduire le temps de calcul de cette dernière d'un temps quadratique à un temps linéaire par rapport au nombre de données.

4.5 Nyström MMD

Comme déjà discuté en Section 2.1.2, Nyström est l'une des méthodes les plus utilisées pour l'approximation de matrice de Gram. Rappelons que pour un jeu de données $\mathcal{F} := \{x_1, \ldots, x_n\}$, cette méthode construit $\hat{K}_{\mathcal{F}}$, une approximation de $K_{\mathcal{F}}$, définie comme :

$$\hat{K}_{\mathcal{F}} := K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^{T},$$

où $\mathcal{S} := \{z_1, \ldots, z_s\} \subseteq \mathcal{F}$ est un échantillon de $\mathcal{F}$ tiré uniformément. Rappelons aussi que la matrice $K_{\mathcal{F},\mathcal{S}} := (k(x,z))_{x \in \mathcal{F}; z \in \mathcal{S}}$ et $K_{\mathcal{S}} := (k(x,z))_{x \in \mathcal{S}; z \in \mathcal{S}}$.

Cette approximation de la matrice de Gram est associée à un noyau approximé $\hat{k}_{nys}$ tel que :

$$\hat{k}_{nys}(x, x') := \langle \hat{\phi}_{nys}(x), \hat{\phi}_{nys}(x') \rangle,$$

où :

$$\hat{\phi}_{nys}(x) := [k(z_1, x), \ldots, k(z_s, x)] \left(K_{\mathcal{S}}^{+}\right)^{\frac{1}{2}} \in \mathbb{R}^s.$$

Nous pouvons assez facilement vérifier que pour tout $i, j \in [n]$: $\langle \hat{\phi}_{nys}(x_i), \hat{\phi}_{nys}(x_j) \rangle = \hat{K}_{\mathcal{F}}[i, j]$.

Dans la suite de cette section, nous proposons de construire une approximation de la MMD en utilisant la méthode de Nyström. Nous utiliserons ensuite notre *framework* de distance entre distributions afin de démontrer, dans le Lemme 4, que cette approximation reste encore une distance.

Pour ce faire, nous considérons le noyau $\hat{k}$ obtenu en régularisant le noyau approximé de Nyström $\hat{k}_{nys}$, i.e. pour tout $x, z \in V$:

$$\hat{k}(x, z) = \hat{k}_{nys}(x, z) + \lambda \delta_x(z), \tag{4.7}$$

où $\lambda > 0$ est un paramètre de « régularisation ».

Lemme 4. *Soit V et k comme ci-dessus. Si $\hat{k}$ est défini comme dans l'Équation (4.7), alors $\mathrm{MMD}_{\hat{k}}(.,.)$ est une distance sur $\mathcal{D}_V$.*

Démonstration. La preuve de ce lemme est une application directe du Théorème 3 qui s'applique car $\hat{k}$ est λ-défini positif. □

Rappelons que nous avons expliqué dans la Section 4.2 que la fonction qui à $\mathbb{P}$ associe $\tilde{\mu}_{\mathbb{P}} := \frac{1}{n} \sum_{x \in \mathcal{X}} \hat{k}_{nys}(x, .)$ associée au noyau $\hat{k}_{nys}$ n'est plus injective. Dans ce cas, il n'est pas possible de démontrer avec les outils de [Gretton et al., 2012] que la MMD calculée avec un noyau approximé par Nyström est une distance. Toutefois, ce lemme nous permet de démontrer cela lorsque le noyau approximé est régularisé comme ci-dessus. En particulier, ce lemme justifie l'utilisation de l'approximation de Nyström régularisée avec la distance MMD. En effet, ce lemme prouve que la MMD calculée avec un noyau approximé par la méthode de Nyström et régularisé comme ci-dessus reste une distance. Son utilisation est alors théoriquement justifiée pour comparer des distributions de probabilités.

Nous montrons dans le lemme suivant qu'en plus de rester une distance, l'approximation de la MMD avec la méthode de Nyström régularisée permet un gain en temps de calcul considérable. En effet, nous démontrons que calculer la distance MMD entre deux distributions de n données avec

une approximation de Nyström régularisée se fait en temps linéaire en n ce qui est moins coûteux que le calcul de la MMD avec le noyau exact qui se fait en $\Theta\left(n^2\right)$.

Lemme 5. *Soit V et k comme ci-dessus. Soit $\mathcal{F} := \{x_1, \ldots, x_n\} \subseteq V$ de taille n, et $\hat{k}$ défini comme dans l'Équation (4.7) où $\hat{k}_{nys}$ est construit avec la méthode de Nyström en utilisant un échantillon $\mathcal{S}$ de s données. Enfin, soit $\hat{\mathbb{P}} := \sum_{i=1}^{n} p_i \delta_{x_i}(.)$ et $\hat{\mathbb{Q}} := \sum_{i=1}^{n} q_i \delta_{x_i}(.)$ deux distributions de probabilités discrètes. Le calcul de $\mathrm{MMD}_{\hat{k}}\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right)$ se fait en $\Theta\left(ns\right)$.*

Démonstration. Pour commencer, remarquons que :

$$
\begin{aligned}
\mathrm{MMD}_{\hat{k}}^2\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right) &= \sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j \hat{k}(x_i, x_j) + \sum_{i=1}^{n}\sum_{j=1}^{n} q_i q_j \hat{k}(x_i, x_j) - 2\sum_{i=1}^{n}\sum_{j=1}^{n} p_i q_j \hat{k}(x_i, x_j), \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j \hat{k}(x_i, x_j) + q_i q_j \hat{k}(x_i, x_j) - 2 p_i q_j \hat{k}(x_i, x_j), \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \left(p_i - q_i\right) \hat{k}(x_i, x_j) \left(p_j - q_j\right). \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \left(p_i - q_i\right) \left(\hat{k}_{nys}(x_i, x_j) + \lambda \delta_{x_i}(x_j)\right) \left(p_j - q_j\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \left(p_i - q_i\right) \left(K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^{T} + \lambda I\right)[i, j] \left(p_j - q_j\right) \\
&= v^T \left(K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^{T} + \lambda I\right) v \\
&= v^T K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^{T} v + \lambda \left\|v\right\|_2^2.
\end{aligned}
$$

où $v := (p_1 - q_1, \ldots, p_n - q_n) \in \mathbb{R}^n$.

La création de $K_{\mathcal{F},\mathcal{S}}$ et le calcul de $vK_{\mathcal{F},\mathcal{S}}$ se font en $\Theta\left(ns\right)$, tandis que la création de $K_{\mathcal{S}}^{+}$ se fait en $\Theta\left(s^3\right)$, et enfin le calcul de $\left\|v\right\|_2^2$ se fait en $\Theta\left(n\right)$. La complexité globale est alors en $\Theta\left(ns + s^3\right)$, ce qui se réduit à $\Theta\left(ns\right)$ lorsque $s \ll n$, ou lorsque les matrices $K_{\mathcal{F},\mathcal{S}}$ et $K_{\mathcal{S}}^{+}$ sont préalablement construites. □

Maintenant que nous savons que le calcul de la MMD avec un noyau approximé par Nyström est rapide, et que son utilisation est théoriquement fondée, une question naturelle se pose : à quel point cette approximation est loin de la MMD exacte ? Pour répondre à cette question, nous utiliserons des résultats récents sur la qualité des approximations de matrices de Gram

construites avec des algorithmes aléatoires [Gittens and Mahoney, 2016]. Lorsque l'approximation de la matrice de Gram est construite avec un échantillonnage uniforme, des hypothèses sur la *cohérence* de la matrice de Gram sont nécessaires. Pour définir cette notion, soit $K_{\mathcal{F}} := U\Sigma U^T$ la diagonalisation de $K_{\mathcal{F}}$. La cohérence de la matrice $K_{\mathcal{F}}$ d'ordre $r \in [n]$ est calculée comme :

$$c_r\left(K_{\mathcal{F}}\right) = \frac{n}{r} \max_{i \in [n]} \|U_r[i,:]\|_2^2,$$

où $U_r \in \mathbb{R}^{n \times r}$ contient les r vecteurs propres associés aux r plus grandes valeur propres de $K_{\mathcal{F}}$ (en d'autres termes, U_r contient les r premières colonnes de U).

Avant d'énoncer ce résultat, nous attirons l'attention du lecteur que la *cohérence d'une matrice* définie ci-dessus est une notion différente de la *cohérence d'un dictionnaire* définie en Section 2.2.3.

Lemme 6. *Soit V, k, et $c_r\left(K_{\mathcal{F}}\right)$ comme ci-dessus. Soit $\mathcal{F} := \{x_1, \ldots, x_n\} \subseteq V$, $\hat{k}$ défini comme dans l'Équation (4.7) où $\hat{k}_{nys}$ est construit avec la méthode de Nyström en utilisant un échantillon $\mathcal{S}$ de s données. Soit $\rho \in]0,1[$ une probabilité d'échec, et $\epsilon \in]0,1[$ un paramètre de précision. Enfin, soit $\hat{\mathbb{P}} := \sum_{i=1}^n p_i \delta_{x_i}(.)$ et $\hat{\mathbb{Q}} := \sum_{i=1}^n q_i \delta_{x_i}(.)$ deux distributions de probabilités discrètes. Si $s \geq 2c_r\left(K_{\mathcal{F}}\right)\epsilon^{-2} r \ln\left(\frac{r}{\rho}\right)$, alors avec probabilité supérieure à $1 - 3\rho$, nous avons :*

$$\left|\mathrm{MMD}_k^2\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right) - \mathrm{MMD}_{\hat{k}}^2\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right)\right| \leq \left(2\sqrt{2} + \frac{2\sqrt{2}}{\rho^2(1-\epsilon)}\right) \|K_{\mathcal{F}} - K_r\|_* + 2\lambda,$$

où K_r est la meilleure approximation de rang r de $K_{\mathcal{F}}$, et où $\|A\|_* :=$ trace $\left(\sqrt{A^T A}\right)$ est la norme nucléaire de la matrice A.

Démonstration. Notons $v := (p_1 - q_1, \ldots, p_n - q_n) \in \mathbb{R}^n$, et rappelons que $\mathrm{MMD}_k^2\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right) = v K_{\mathcal{F}} v^T$ (voir la preuve du Lemme 5). Ainsi, en utilisant les propriétés de la trace : $\mathrm{MMD}_k^2\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right) =$ trace $\left(v K_{\mathcal{F}} v^T\right) =$ trace $\left(K_{\mathcal{F}} v^T v\right)$.

En faisant de même avec $\mathrm{MMD}^2_{\hat{k}}\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right)$ nous obtenons :

$$
\begin{aligned}
&|\mathrm{MMD}^2_k\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right) - \mathrm{MMD}^2_{\hat{k}}\left(\hat{\mathbb{P}}, \hat{\mathbb{Q}}\right)| = |\mathrm{trace}\left(K_{\mathcal{F}} v v^T\right) - \mathrm{trace}\left(\left(\hat{K}_{\mathcal{F}} + \lambda I\right) v v^T\right)| \\
&= |\mathrm{trace}\left(K_{\mathcal{F}} v v^T\right) - \mathrm{trace}\left(\left(K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^T + \lambda I\right) v v^T\right)| \\
&= |\mathrm{trace}\left(K_{\mathcal{F}} v v^T\right) - \mathrm{trace}\left(K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^T v v^T + \lambda v v^T\right)| \\
&= |\mathrm{trace}\left(K_{\mathcal{F}} v v^T\right) - \mathrm{trace}\left(K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^T v v^T\right) + \mathrm{trace}\left(\lambda v v^T\right)| \\
&= |\mathrm{trace}\left(\left(K_{\mathcal{F}} - K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^T\right) v v^T\right) - \lambda\, \mathrm{trace}\left(v v^T\right)| \\
&\leq |\mathrm{trace}\left(\left(K_{\mathcal{F}} - K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^T\right) v v^T\right)| + \lambda |\mathrm{trace}\left(v v^T\right)| \\
&= |\mathrm{trace}\left(\left(K_{\mathcal{F}} - K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^T\right) v v^T\right)| + \lambda \left\|v\right\|_2^2 . \\
&\overset{(a)}{\leq} \left\|K_{\mathcal{F}} - K_{\mathcal{F},\mathcal{S}} K_{\mathcal{S}}^{+} K_{\mathcal{F},\mathcal{S}}^T\right\|_* \left\|v v^T\right\|_\infty + \lambda \left\|v\right\|_2^2 . \\
&\overset{(b)}{\leq} \left(2 + \frac{2}{\rho^2(1-\epsilon)}\right) \left\|K_{\mathcal{F}} - K_r\right\|_* \left\|v\right\|_2 + \lambda \left\|v\right\|_2^2 .
\end{aligned}
$$

où l'inégalité (a) est vraie grâce à l'inégalité de Hölder [Bhatia, 1997], et l'inégalité (b) grâce au Lemme 8 de [Gittens and Mahoney, 2016]. Enfin, comme $\sum_{i=1}^n p_i = \sum_{i=1}^n q_i = 1$, on a $\left\|v\right\|_2^2 \leq 2$, ce qui termine notre preuve. □

4.6 Expériences numériques

Nous proposons maintenant d'évaluer empiriquement notre approximation de la MMD basée sur la méthode de Nyström, que nous notons ici : *MMD Nyström*. Pour cela, nous considérons le *three sample problem* sur des données réelles et synthétiques.

Avant de présenter nos résultats, rappelons en quoi consiste ce problème : étant données trois ensembles de points $\mathcal{X} := \{x_1, \ldots, x_{n_x}\}$, $\mathcal{Z} := \{z_1, \ldots, z_{n_z}\}$ et $\mathcal{W} := \{w_1, \ldots, w_{n_w}\}$ tel que $\mathcal{X}$ et $\mathcal{Z}$ sont générés suivant deux distributions de probabilités différentes $\mathbb{P}_{\mathcal{X}}$ et $\mathbb{P}_{\mathcal{Z}}$, décider si $\mathcal{W}$ est généré suivant $\mathbb{P}_{\mathcal{X}}$ ou $\mathbb{P}_{\mathcal{Z}}$. Nous supposerons évidemment que $\mathcal{W}$ est généré suivant l'une des deux distributions de probabilités. Ce problème peut être résolu en utilisant une distance entre distributions discrètes D. En effet, si $D\left(\mathcal{X}, \mathcal{W}\right) \leq D\left(\mathcal{Z}, \mathcal{W}\right)$, alors nous décidons que $\mathcal{W}$ est généré suivant $\mathbb{P}_{\mathcal{X}}$, sinon nous décidons que $\mathcal{W}$ est généré suivant $\mathbb{P}_{\mathcal{Z}}$. Contrairement au *two sample problem*, dont le but est de décider si deux jeux de

Jeu de données	Nombre de données	Structur
Covertype[R. Collobert, S. Bengio, and Y. Bengio, 2002]	581012	vectoriel
MNIST [Yann et al., 1998]	70000	vectoriel
$\mathcal{N}(0,1) - \mathcal{N}(0,1.001)$	2×10^5	vectoriel
MUTAG [Debnath et al., 1991]	188	graphe

TABLE 4.1 – Description des jeux de données

données sont tirés suivant la même distribution, dans ce problème nous n'avons pas besoin de définir un seuil pour prendre la décision.

4.6.1 Dispositif expérimental

Nous considérons deux structures de données : vectorielles et graphes, que nous résumons dans le Tableau 4.1. Pour les données vectorielles, nous utilisons le noyau *RBF (Radial basis function)* avec un paramètre $\sigma := \left(\frac{1}{n}\sum_{x_i,x_j\in\mathcal{F}} \left\|x_i - x_j\right\|_2^2\right)^{-1}$ tel que proposé dans [Brefeld et al., 2006], tandis que nous utiliserons le noyau *ShortestPath* [Borgwardt and Kriegel, 2005] pour les données graphes.

Nous considérons dans la suite uniquement le cas où $\mathcal{X}$, $\mathcal{Z}$ et $\mathcal{W}$ ont le même nombre de données que nous notons par n, mais nos résultats restent valides lorsqu'ils ont des tailles différentes.

Nous comparons notre approximation basée sur la méthode de Nyström avec trois autres méthodes d'approximation de la MMD que nous avons introduits en section 4.2 : *MMD linéaire* [Gretton et al., 2012], *MMD bloc* [Zaremba et al., 2013], et enfin l'approximation basée sur les Random Fourier Features (*MMD RFF*) [Zhao and Meng, 2015]. Lorsque le nombre de données n'est pas très grand, nous calculons aussi la valeur de la MMD exacte telle que définie dans l'Équation (4.2).

Rappelons que la complexité de la méthode *MMD linéaire* est de $\Theta(n)$, que celle de *MMD bloc*, *MMD Nyström* et *MMD RFF* sont de $\Theta(ns)$, et enfin que le calcul de la valeur exacte de la MMD se fait en $\Theta(n^2)$.

Afin de comparer des méthodes de complexités équivalentes, nous fixons $s = \log(n)$ pour les méthodes *MMD bloc*, *MMD Nyström* et *MMD RFF*.

Nous nous intéressons à l'erreur commise par chacune des méthodes dans la résolution du *three sample problem*. L'erreur est définie comme le ratio entre le nombre de jeux de données mal classées et le nombre total de

jeux de données :

$$erreur := \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\left(D\left(\mathcal{X}_i, \mathcal{W}_i\right) > D\left(\mathcal{Z}_i, \mathcal{W}_i\right)\right),$$

tel que $\forall i : \mathcal{W}_i := \{w_{i_1}, \ldots, w_{i_n}\} \sim \mathbb{P}_{\mathcal{X}}, \mathcal{X}_i := \{x_{i_1}, \ldots, x_{i_n}\} \sim \mathbb{P}_{\mathcal{X}}, \mathcal{Z}_i := \{z_{i_1}, \ldots, z_{i_n}\} \sim \mathbb{P}_{\mathcal{Z}}$ sont des ensembles de points, et où m est le nombre de tirages qui sera spécifié plus tard. Nous calculons aussi le temps de calcul ainsi que la distance entre les valeurs de MMD approximées et la valeur exacte de la MMD :

$$\Delta_{\mathcal{X},\mathcal{Z}} := |\mathrm{MMD}_{\hat{k}}\left(\mathcal{X}, \mathcal{Z}\right) - \mathrm{MMD}_k\left(\mathcal{X}, \mathcal{Z}\right)|,$$

où $\mathrm{MMD}_{\hat{k}}$ est calculée en utilisant l'une des méthodes d'approximation : *MMD linéaire*, *MMD bloc*, *MMD RFF* ou *MMD Nyström*. Nous appellerons aussi $\Delta_{\mathcal{X},\mathcal{Z}}$ la vitesse de convergence de la méthode d'approximation utilisée vers la MMD exacte.

Toutes les expériences sont répétées sur plusieurs tirages, et nous affichons la moyennes sur ces différents tirages.

Nous considérons quatre expériences différentes : une sur des données vectorielles synthétiques, deux sur des données vectorielles réelles, et une sur des données graphes. La première expérience nous permet de valider les complexités de calculs théoriques ainsi que l'utilité des méthodes d'approximation de la MMD pour le *three sample problem*. Les données vectorielles réelles serviront à illustrer la robustesse de ces méthodes d'approximation sur des jeux de données réels. Enfin, la dernière expérience permet d'illustrer la possibilité d'utiliser notre méthode avec des noyaux sur graphes.

4.6.2 Résultats

Données vectorielles

Nous commençons par un jeu de données synthétique constitué de 10^5 données avec $\mathbb{P}_{\mathcal{X}} = \mathcal{N}(0, 1)$ et $\mathbb{P}_{\mathcal{Z}} = \mathcal{N}(0, 1.001)$. Ces deux distributions sont très proches, et il faudra considérer un nombre de données n très grand pour arriver à les discriminer. Cette expérience nous permet donc de voir la limite de la MMD qui ne sera pas calculable au delà d'une certaine valeur de n. Cette expérience nous permet aussi de valider les complexités de temps de calcul annoncées plus haut.

Pour ce jeu de données, nous considérons 200 tirages d'ensembles $\mathcal{X}$, $\mathcal{Z}$ et $\mathcal{W}$ différents. Comme nous pouvons le voir dans la Figure 4.2a, les méthodes *MMD Nyström* et *MMD RFF* ont une erreur équivalente, tandis que

MMD linéaire et *MMD bloc* semblent incapables de discriminer ces deux gaussiennes. Pour de petite valeur de n, nous remarquons que la MMD exacte fait autant d'erreur que *MMD Nyström* et *MMD RFF*. Cette figure montre l'intérêt de telles approximations : ces deux gaussiennes sont très similaires, et il faut donc tirer un très grand nombre de données avant de pouvoir les discriminer. La MMD exacte n'est pas calculable pour des valeurs de n aussi grandes, tandis que les méthodes d'approximation le sont ce qui leur permet au final de discriminer ces deux gaussiennes. En comparant la distance entre l'approximation et la MMD exacte (Figure 4.2b), nous pouvons faire le même constat : *MMD Nyström* et *MMD RFF* convergent vers la valeur exacte de la MMD à la même vitesse, tandis que *MMD linéaire* et *MMD bloc* prennent plus de temps. Toutefois, la méthode *MMD linéaire* est la plus rapide comme prévu et comme nous pouvons le constater dans la Figure 4.2c. Les méthodes *MMD Nyström* et *MMD RFF* ont un temps de calcul similaire lorsque n est grand et sont toutes deux plus rapides que *MMD bloc*. Enfin, la MMD exacte a un temps de calcul qui explose et ne peux même plus être calculée lorsque n est très grand. Notons que la plus grande valeur de n considérée dans la Figure 4.2b est de 10^4 car le calcul de $\Delta_{\mathcal{X},\mathcal{Z}}$ nécessite le calcul de la MMD exacte qui devient trop lent au-delà de cette valeur.

Nous avons vu dans la Figure 4.2 que sur un jeu de données synthétique, les méthodes d'approximation de la MMD permettaient de relativement bien (selon la méthode considérée) résoudre le *three sample problem*. Nous proposons maintenant de regarder comment ces méthodes se comportent sur des jeux de données réels. Commençons par le jeu de données MNIST, dont les données sont divisées en 10 classes. Sur ce jeu de données, la distribution $\mathbb{P}_{\mathcal{X}}$ est représentée par les données étiquetées par les classes entre 0 et 4, alors que $\mathbb{P}_{\mathcal{Z}}$ est représentée par les données étiquetées par les classes entre 5 et 9. Nous faisons les mêmes expériences que précédemment, que nous répétons cette fois-ci 500 fois, et affichons les résultats dans la Figure 4.3. Cette fois ci, nous constatons que lorsque n est petit (i.e. $n \leq 5000$), alors *MMD RFF* fait moins d'erreur que toutes les autres méthodes d'approximation. Toutefois, lorsque $n \geq 5000$, notre approximation *MMD Nyström* est la méthode d'approximation qui fait le moins d'erreurs. De plus, à partir de $n \approx 10000$, notre méthode ne fait aucune erreur (tout comme la MMD exacte), tout en étant plus rapide que la MMD exacte. Notons que sur ce jeu de données, la MMD exacte ne fait aucune erreur même pour de très petites valeurs de n. Sur la Figure 4.3b, nous constatons que *MMD RFF* converge plus vite vers la valeur exacte de la MMD que les autres méthodes d'approximation. MMD Nyström et *MMD bloc* convergent à la même vitesse vers la valeur exacte de la MMD, tandis que *MMD linéaire*

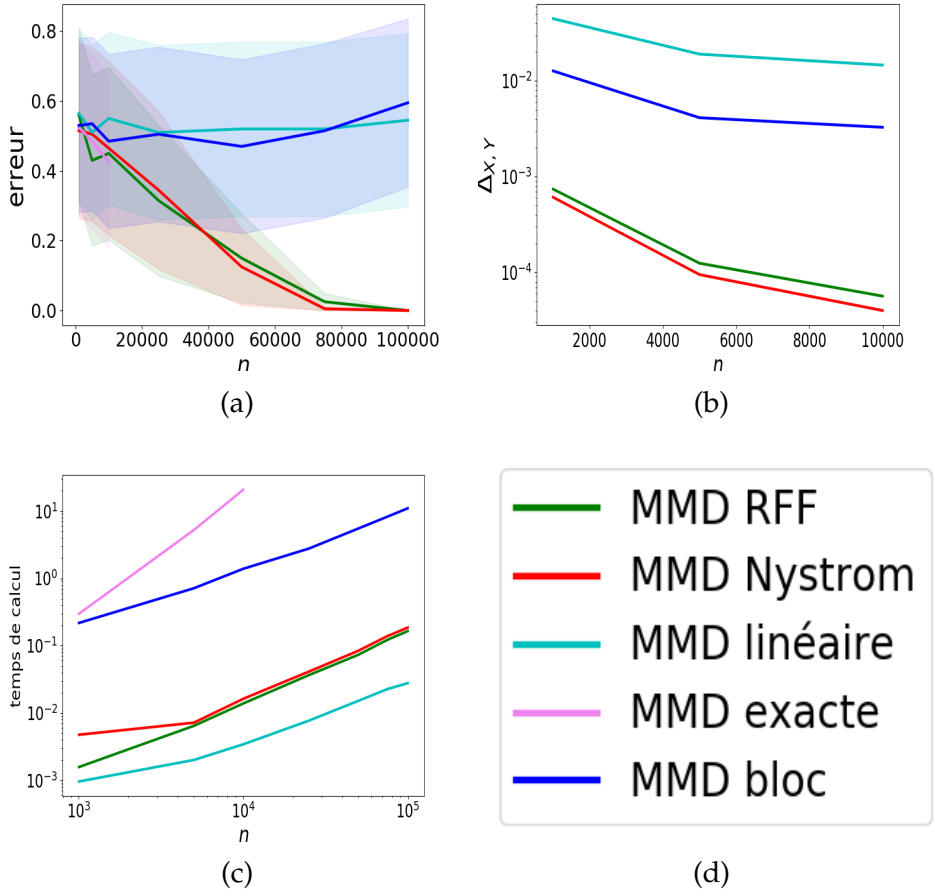


FIGURE 4.2 – Erreur dans le *three sample problem*, convergence vers la MMD exacte et temps de calcul avec $\mathbb{P}_{\mathcal{X}} = \mathcal{N}(0,1)$ et $\mathbb{P}_{\mathcal{Z}} = \mathcal{N}(0,1.001)$.

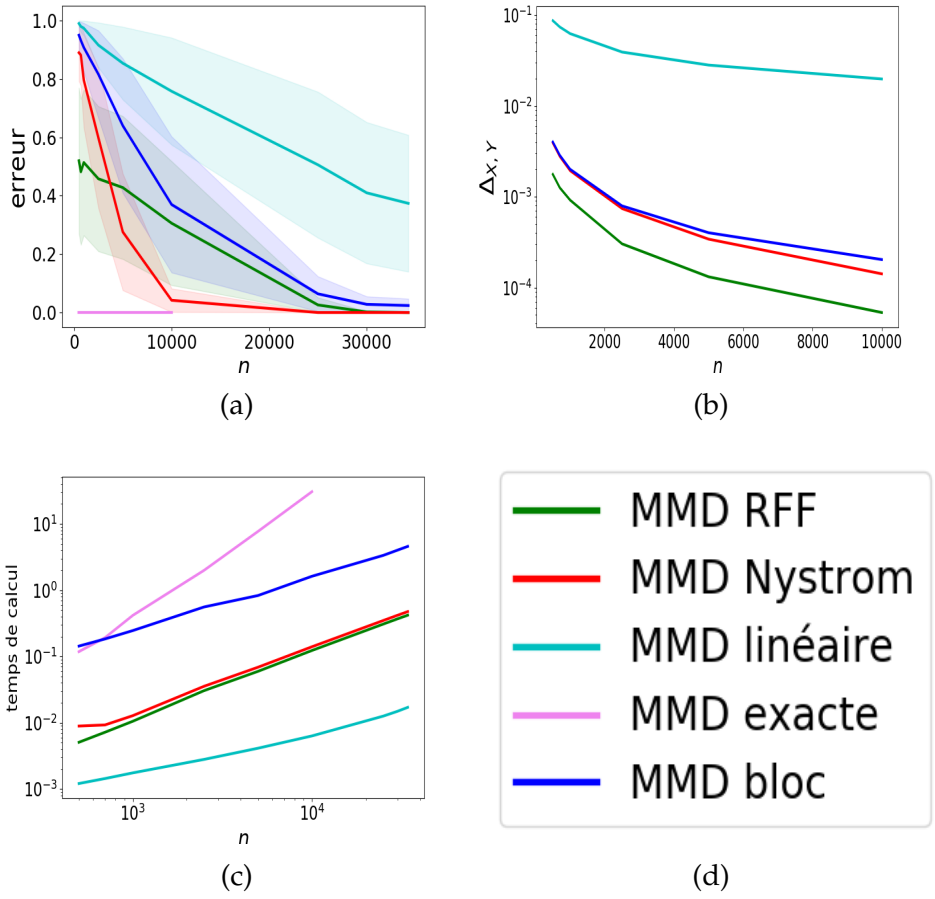


FIGURE 4.3 – Erreur dans le *three sample problem*, convergence vers la MMD exacte et temps de calcul sur le jeu de données MNIST.

est plus lente. Enfin, sur la Figure 4.3c nous constatons les mêmes résultats que précédemment, la *MMD linéaire* est la plus rapide, suivi par *MMD RFF* et *MMD Nyström* qui ont un temps de calcul similaire, suivi par *MMD bloc* et enfin par MMD exacte.

Enfin, considérons notre dernier jeu de données vectoriel réel de classification binaire : Covertype, qui contient encore plus de données que MNIST. Ce jeu de données nous permet de comparer les temps de calcul des différentes méthodes d'approximation pour de plus grandes valeurs de n que celles considérées dans MNIST. Ici, la distribution $\mathbb{P}_{\mathcal{X}}$ est représentée par les données étiquetées par la classe 0, alors que $\mathbb{P}_{\mathcal{Z}}$ est représentée par les données étiquetées par la classe 1. Nous répétons les expériences 500 fois, et affichons les résultats dans la Figure 4.4. Cette fois ci, nous

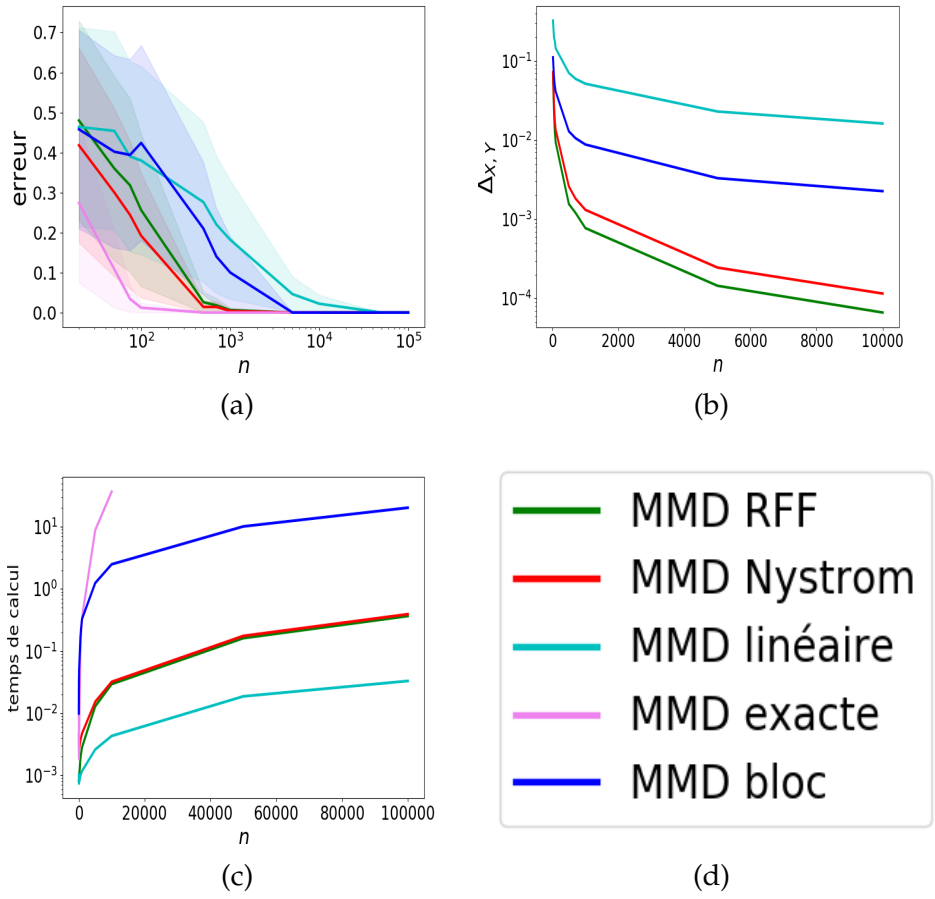


FIGURE 4.4 – Erreur dans le *three sample problem*, convergence vers la MMD exacte et temps de calcul sur le jeu de données Covertype.

constatons que la MMD exacte est celle qui fait le moins d'erreurs, suivie juste après par *MMD Nyström*. Sur la Figure 4.4b, nous constatons que notre approximation *MMD Nyström* et *MMD RFF* convergent presque à la même vitesse vers la valeur exacte de la MMD.

Données graphes

Passons maintenant aux données graphes. Nous traçons les résultats obtenus sur le jeu de données MUTAG dans la Figure 4.5. Notons que la méthode *MMD RFF* ne peut pas être utilisée pour ce jeu de données de graphe. Comme attendu, la MMD exacte est celle qui fait le moins d'erreur, suivi juste après par notre méthode *MMD Nyström*, puis par *MMD bloc* et

linéaire. En terme de convergence vers la valeur exacte de la MMD, nous remarquons que *MMD Nyström* est la méthode qui converge le plus vite vers la valeur optimale. Enfin, et sans surprise aussi, MMD exacte mise de côté, la méthode *MMD linéaire* est la plus rapide, tandis que *MMD bloc* est la plus lente.

Nous concluons de ces expériences que notre approximation *MMD Nyström* permet un bon compromis entre erreurs et temps de calcul dans la résolution du *three sample problem*. En effet, nous avons vu que notre méthode d'approximation de la MMD basée sur la méthode de Nyström fait souvent moins d'erreurs que *MMD linéaire* et que *MMD bloc*. En terme de temps de calcul, notre méthode est plus rapide à calculer que *MMD bloc* mais plus lente que *MMD linéaire*. Ce gain en temps de calcul nous permet de discriminer des distributions de probabilités très similaire (comme nous l'avons vu avec les deux gaussiennes) en considérant des ensembles de données de très grande taille. De plus, contrairement à *MMD RFF* notre méthode d'approximation est valide pour une large famille de noyaux, parmi lesquels les noyaux sur graphes.

4.7 Conclusion

Nous avons dans ce chapitre présenté une nouvelle connexion entre les distributions de probabilités et les RKHSs sur des domaines discrets. Nous avons utilisé cette connexion afin de proposer un framework pour représenter et comparer les distributions de probabilités.

Dans un second temps, nous avons montré comment la MMD, une célèbre distance entre distributions de probabilités, s'intègre dans notre framework. Nous avons ensuite utilisé cette connexion pour proposer une nouvelle approximation de la MMD, basée sur la méthode de Nyström. Notre approximation est théoriquement justifiée pour une large gamme de fonctions noyaux y compris les noyaux sur graphes.

Enfin, nous avons empiriquement évalué notre méthode, et nous avons vu qu'elle présentait un bon compromis entre erreur et temps de calcul dans le cadre du *three sample problem*.

Il serait intéressant d'étudier dans les travaux futurs le comportement de notre approximation de la MMD dans d'autres problèmes tel que le *two sample problem*, l'apprentissage actif ou l'adaptation de domaine. Il serait aussi intéressant d'étudier empiriquement l'utilisation d'autres noyaux avec notre méthode d'approximation. Enfin, comment d'autres distances telles que la distance de Wasserstein s'inscrivent dans notre framework est aussi une question intéressante.

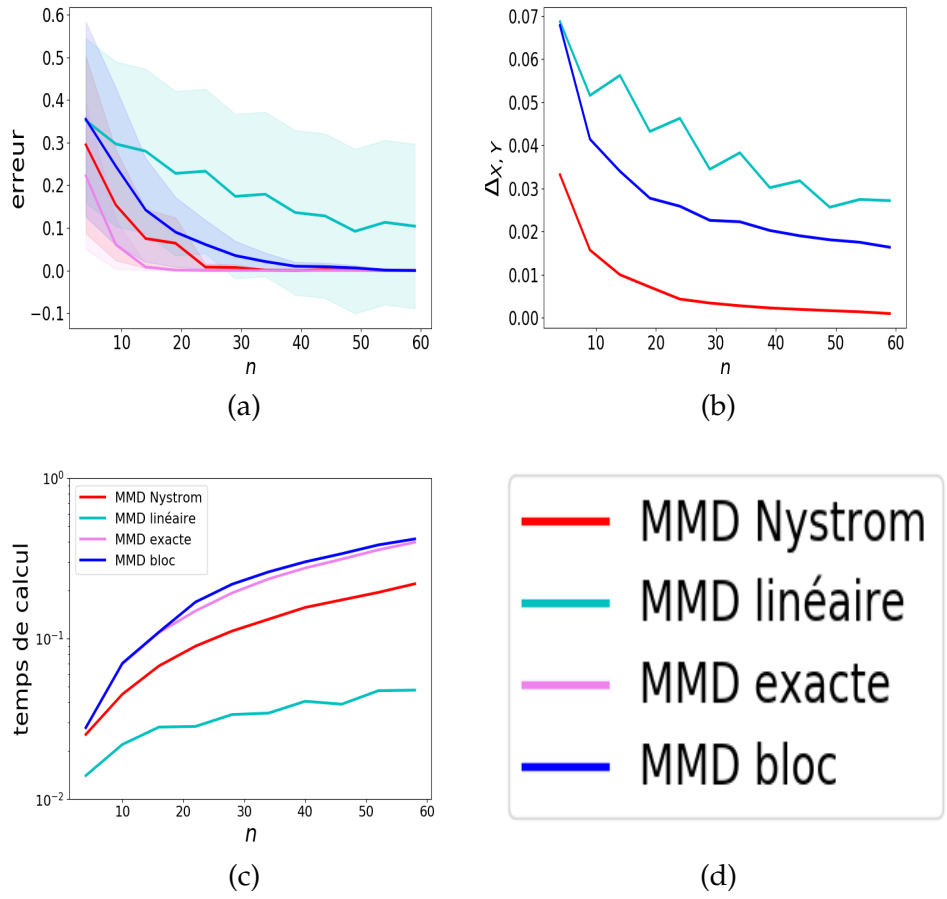


FIGURE 4.5 – Erreur dans le *three sample problem*, convergence vers la MMD exacte et temps de calcul sur le jeu de données MUTAG.

Chapitre 5

Frank-Wolfe pour l'approximation parcimonieuse de signaux

Nous nous sommes consacrés dans les chapitres 3 et 4 à l'accélération de méthodes basées sur les noyaux.

Nous proposons dans les deux chapitres suivants de nous intéresser à l'optimisation avec contrainte de parcimonie. Nous étudions dans le chapitre 5 les propriétés de reconstruction et de convergence de l'algorithme Frank-Wolfe (FW) dans l'approximation de signaux par des représentations parcimonieuses. Nous étudions ensuite dans le chapitre 6 l'élagage de forêts aléatoires avec l'algorithme Orthogonal Matching Pursuit (OMP).

5.1 Introduction et état de l'art

Nous présentons dans ce chapitre nos résultats sur les propriétés de l'algorithme de Frank-Wolfe dans la reconstruction de signaux parcimonieux. Ces travaux ont été publiés dans le journal IEEE Transaction and Information Theory (TIT) 2019 [Cherfaoui et al., 2019], et présentés dans les conférences GRETSI 2019 et iTWIST 2018.

5.1.1 Introduction

Pour un signal $y \in \mathbb{R}^d$ et un dictionnaire $\mathcal{D} := \{v_1, \ldots, v_n\}$, le problème d'approximation parcimonieuse a pour but de trouver une approximation m-parcimonieuse de y dans $\mathcal{D}$. Cela revient à construire une combinaison linéaire de m éléments de $\mathcal{D}$ qui se « rapproche » le plus possible de y. La qualité de cette approximation peut être par exemple mesurée par la norme euclidienne de la différence entre y et son approximation. Plus

formellement, ce problème peut être posé comme :

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}^n} \frac{1}{2} \left\| y - \sum_{i=1}^{n} \alpha[i] v_i \right\|_2^2 \text{ tel que } \|\alpha\|_0 \leq m. \tag{5.1}$$

Minimiser $\|y - \sum_{i=1}^{n} \alpha[i] v_i\|_2^2$ permet de trouver le vecteur des poids α associé à la meilleure approximation de y, tandis que la condition $\|\alpha\|_0 \leq m$ permet de s'assurer qu'au plus m atomes sont utilisés dans la construction de cette approximation.

Malheureusement, le Problème (5.1) est NP-dur [Davis et al., 1997]. Plusieurs méthodes permettant de construire des approximations de α^* ont été alors proposées.

La majorité des méthodes d'approximation peuvent être classées dans deux catégories : les méthodes gloutonnes [Mallat and Zhang, 1993, Pati et al., 1993, Adler, 2015], et les relaxations convexes [Chen et al., 2001, Tibshirani, 1996]. Les méthodes gloutonnes construisent de manière itérative une approximation de y en sélectionnant à chaque itération l'élément de $\mathcal{D}$ qui leur permet de mieux approximer y. Comme un seul élément est sélectionné à chaque itération, ces méthodes permettent un total contrôle sur la parcimonie de leur approximation. En effet, au bout de k mises-à-jour, l'approximation de α^* construite par les méthodes gloutonnes contient au plus k éléments non nuls. Toutefois, comme la quantité $\|y - \sum_{i=1}^{n} \alpha[i] v_i\|_2^2$ n'est pas explicitement minimisée, les méthodes gloutonnes risquent de construire des approximations, certes m-parcimonieuses, mais pas forcément très « proches » de y. Néanmoins, lorsque y et $\mathcal{D}$ ont des formes particulières, la distance entre y et l'approximation construite par les méthodes gloutonnes peut être bornée [Gribonval and Vandergheynst, 2006, Tropp, 2004, Gilbert et al., 2003, Candes and Tao, 2005].

Les méthodes de relaxation convexe quant à elles partent de la remarque que la difficulté majeure du Problème (5.1) est la contrainte sur la quasi-norme l_0. Le Problème (5.1) est alors relaxé en remplaçant la contrainte l_0 par une contrainte sur une norme convexe comme la norme l_1 :

$$\alpha_1 = \arg\min_{\alpha \in \mathbb{R}^n} \left\| y - \sum_{i=1}^{n} \alpha[i] v_i \right\|_2^2 \text{ tel que } \|\alpha\|_1 \leq \beta. \tag{5.2}$$

Ce problème d'optimisation relaxé est convexe et peut être résolu en temps polynomial. Toutefois, contrairement aux méthodes gloutonnes, les méthodes de relaxation ne permettent pas toujours de connaître à chaque itération le nombre d'atomes qui interviennent dans la décomposition de

leur approximation. Il existe cependant certaines conditions sur y et $\mathcal{D}$ sous lesquelles cela est possible [Tropp, 2006, Chen et al., 2001, Tibshirani, 1996].

Nous nous intéressons dans ce chapitre à la relaxation l_1 décrite dans l'Équation (5.2). Nous proposons de résoudre ce problème avec un algorithme d'optimisation convexe : Frank-Wolfe (FW) [Frank and Wolfe, 1956]. Plus particulièrement, nous étudions les conditions sous lesquelles la solution du Problème (5.1) et celle du Problème (5.2) construite par FW coïncident.

5.1.2 État de l'art

Il sera question dans cette partie de convergence, et plus préciseraient de vitesse de convergence, de certains algorithmes. Nous commencerons donc par définir ces notions. Soit $\{f_k\}$ une suite. On dira que cette suite converge exponentiellement vite vers f^* s'il existe $\theta \in]0,1[$ tel que pour tout k :

$$\|f_{k+1} - f^*\| \leq \theta \|f_k - f^*\| .$$

On dira qu'elle converge linéairement vers f^* si pour tout k :

$$\|f_k - f^*\| \leq \Theta\left(\frac{1}{k}\right) .$$

Revenons maintenant à notre état de l'art.

Comme discuté en section 2.2.2, lorsque $\mathcal{D}$ est une base orthonormée, et que y est m-parcimonieux dans $\mathcal{D}$, la solution du Problème (5.1) peut être trouvée en temps linéaire par rapport au nombre d'atomes. En effet, il suffit dans ce cas de projeter indépendamment le signal sur chacun des atomes, et de sélectionner ensuite les m atomes qui sont le plus corrélés avec le signal. Toutefois, lorsque $\mathcal{D}$ n'est pas une base orthonormée, le Problème (5.1) est NP-dur [Davis et al., 1997], et des méthodes d'approximation doivent être utilisées.

Matching Pursuit (MP) [Mallat and Zhang, 1993] et Orthogonal Matching Pursuit (OMP) [Pati et al., 1993] sont deux algorithmes gloutons permettant de construire une solution approximée pour le Problème (5.1). Ils sont largement connus et utilisés dans les communautés de traitement du signal et d'apprentissage automatique. En plus d'être simples à comprendre et faciles à implémenter, ils sont théoriquement justifiés. Nous avons déjà vu que, lorsque $\mathcal{D}$ est une base orthonormée et que y est m-parcimonieux dans $\mathcal{D}$, alors OMP et MP retrouvent la solution optimale en exactement m itérations.

Malheureusement, beaucoup de signaux ne peuvent pas être correctement exprimés, ou même approximés, dans une base orthonormée. Il nous faut donc considérer des dictionnaires moins restrictifs que les bases orthonormées, mais avec suffisamment de structure afin de pouvoir espérer des garanties théoriques sur nos approximations. Pour cela, les notions de cohérence et de fonction Babel d'un dictionnaire $\mathcal{D}$ ont étés introduites. Rappelons que la cohérence de $\mathcal{D}$ est définie comme :

$$\mu(\mathcal{D}) := \max_{i \neq j} |\langle v_i, v_j \rangle|,$$

et que sa fonction Babel est donnée par :

$$\mu_1(\mathcal{D}, m) := \max_{|\Lambda|=m} \max_{j \in [n] \setminus \Lambda} \sum_{i \in \Lambda} |\langle v_i, v_j \rangle|.$$

Ces outils permettent de mesurer à quel point un dictionnaire est proche d'une base orthonormée. En effet, lorsque $\mathcal{D}$ est une base orthonormée, alors $\mu(\mathcal{D}) = \mu_1(\mathcal{D}, m) = 0$ pour tout m. Lorsque tous les atomes sont presque orthogonaux les uns aux autres, i.e. $\langle v_i, v_j \rangle \approx 0$, alors $\mu(\mathcal{D}) \approx 0$ et $\mu_1(\mathcal{D}, m) \approx 0$. Intuitivement, en considérant des dictionnaires de petite cohérence ou de petite fonction Babel, MP et OMP seront plus capables de reconstruire la solution optimale, ou du moins s'en approcher en un temps raisonnable. La cohérence d'un dictionnaire a été proposée par Mallat et Zhang en 1993 [Mallat and Zhang, 1993], et la fonction Babel par Tropp en 2004 [Tropp, 2004]. Ces outils permettent l'établissement de garanties théoriques pour différentes méthodes d'approximation du Problème (5.1). Nous présentons dans la suite certains de ces travaux.

Avant de commencer, nous introduisons quelques notations. Nous utiliserons l'expression de signal m-parcimonieux bruité dans $\mathcal{D}$ pour désigner les signaux qui s'écrivent comme : $y = y_{opt} + \nu$, où ν est un vecteur de bruit inconnu, et $y_{opt} := \sum_{i=1}^{n} \alpha^*[i] v_i$ avec α^* une solution du Problème (5.1).

Les auteurs de [Gilbert et al., 2003] montrent que pour tout signal y m-parcimonieux bruité dans $\mathcal{D}$, si $m \leq \Theta\left(\mu(\mathcal{D})^{-1}\right)$, alors y_{OMP}, l'approximation construite par OMP, vérifie :

$$\|y - y_{OMP}\|_2 \leq 8\sqrt{m} \|y - y_{opt}\|_2, \tag{5.3}$$

Ainsi, plus la cohérence est petite, plus les valeurs de parcimonie m que nous pouvons considérer sont grandes. Si $\mu(\mathcal{D}) \approx 0$, alors m tend vers l'infini, et dans ce cas les garanties théoriques présentées par [Gilbert et al., 2003] sont vérifiées pour tout signal y. Inversement, si $\mu(\mathcal{D})$ est très grande, alors m tend vers zéro, et nous ne serons capables d'approximer aucun signal.

Tropp [Tropp, 2004] généralise le résultat de [Gilbert et al., 2003] en introduisant des conditions impliquant la fonction de Babel. Il énonce une condition suffisante pour que OMP reconstruise exactement tout signal m-parcimonieux. Cette condition, connue sous le nom d'ERC (pour Exact Recovery Condition), est vérifiée lorsque $\mu_1(\mathcal{D}, m) \leq \frac{1}{2}$. Ainsi, lorsque $\mu_1(\mathcal{D}, m) \leq \frac{1}{2}$, OMP reconstruit exactement tout signal m-parcimonieux au bout de m itérations. Il démontre ensuite que pour tout signal y m-parcimonieux bruité, si $m \leq \Theta\left(\mu(\mathcal{D})^{-1}\right)$, alors :

$$\|y - y_{OMP}\|_2 \leq \sqrt{1+6m}\,\|y - y_{opt}\|_2 . \tag{5.4}$$

Remarquons que, pour des valeurs de cohérence et de parcimonie fixées, la borne de [Tropp, 2004] (Équation (5.4)) est plus serrée que celle de [Gilbert et al., 2003] (Équation (5.3)), puisque $\sqrt{1+6m} \leq 8\sqrt{m}$.

En 2006, Gribonval et Vandergheynst [Gribonval and Vandergheynst, 2006] ont étendu les résultats de Tropp à l'algorithme MP. Ils montrent que si y est m-parcimonieux et si la condition ERC est vérifiée, alors l'approximation construite par MP converge exponentiellement vite vers la solution optimale. Ils étudient aussi le cas où y est m-parcimonieux bruité, et montrent que si l'ERC est vérifiée, il existe une itération à partir de laquelle :

$$\|y - y_{MP}\|_2 \leq \sqrt{1+4m}\,\|y - y_{opt}\|_2 .$$

Notons que Gribonval et Vandergheynst ont décidé de renommer la condition ERC par « la condition de stabilité », puisqu'il n'est pas correct de parler de reconstruction exacte avec MP contrairement à OMP.

L'algorithme OMP a aussi été étudié sous une autre condition : RIP (Restricted Isometry Property) [Candes and Tao, 2005]. Cette condition permet de caractériser les dictionnaires qui se comportent « presque » comme une base, mais uniquement sur les signaux parcimonieux.

Les auteurs de [Davenport and Wakin, 2010] montrent que sous certaines conditions sur la constante RIP, OMP reconstruit exactement tout signal m-parcimonieux. Ils montrent aussi expérimentalement qu'il existe des dictionnaires qui satisfont les conditions sur la cohérence mais ne satisfont pas celles sur RIP, et vice versa. Il faut donc voir les conditions sur la constante RIP et celles sur la cohérence comme des conditions complémentaires. La reconstruction de signaux m-parcimonieux bruité sous condition RIP a aussi été étudiée [Zhang, 2011].

Plus récemment, une revisite de OMP, appelée CAMP (pour Covariance Assisted Matching Pursuit), a été proposée dans [Adler, 2015]. Dans cet algorithme les auteurs supposent que les coefficients du support de y sont tirés suivant une distribution de probabilité gaussienne de moyenne et de

variance connue. Ils utilisent ensuite ces deux informations lors de l'étape de mise-à-jour du signal approximé, ce qui leur permet de prendre en compte les potentielles dépendances statistiques entre les coefficients. Il a été démontré que CAMP a de meilleurs résultats d'approximation que OMP en pratique [Adler, 2015]. De plus, des garanties théoriques sur la reconstruction exacte ont été démontrées sous certaines conditions dérivées des conditions RIP [Ge et al., 2019].

Tournons nous maintenant vers les méthodes de relaxation convexe. Le principe de Basis Pursuit (BP) [Chen et al., 2001] par exemple, propose de remplacer le Problème (5.1) par :

$$\alpha_{\text{BP}} = \underset{\alpha \in \mathbb{R}^n}{\arg\min} \|\alpha\|_1 \text{ tel que } \sum_{i=1}^{n} \alpha[i] v_i = y.$$

Les auteurs utilisent ensuite un algorithme basé sur la méthode des points intérieurs pour le résoudre. Sous certaines conditions sur la cohérence, ou lorsque l'ERC est vérifiée, BP reconstruit exactement tout signal m-parcimonieux [Donoho et al., 2001, Tropp, 2004].

D'un autre côté, Tibshirani propose la relaxation LASSO [Tibshirani, 1996] pour la régression :

$$\alpha_{\text{LASSO}} = \underset{\alpha \in \mathbb{R}^n}{\arg\min} \left\| y - \sum_{i=1}^{n} \alpha[i] v_i \right\|_2^2 \text{ tel que } \|\alpha\|_1 \leq \beta.$$

La relaxation LASSO est moins restrictive que celle de BP, puisqu'elle ne contient pas de contrainte sur une reconstruction exacte. Elle a ensuite été légèrement modifiée pour être étudiée pour l'approximation parcimonieuse :

$$\widetilde{\alpha}_{\text{LASSO}} = \underset{\alpha \in \mathbb{R}^n}{\arg\min} \left\| y - \sum_{i=1}^{n} \alpha[i] v_i \right\|_2^2 + \gamma \|\alpha\|_1 .$$

L'auteur de [Tropp, 2006] montre que si y est m-parcimonieux bruité et si $m \leq \Theta\left(\mu(\mathcal{D})^{-1}\right)$, alors pour un choix approprié de γ, le support de $y_{\text{LASSO}} = \sum_{i=1}^{n} \alpha_{\text{LASSO}}[i] v_i$ est inclus dans le support de y [Tropp, 2006]. De plus, sous ces mêmes conditions, la distance entre α_{LASSO} et α^* peut être contrôlée [Tropp, 2006].

D'un autre côté, Frank-Wolfe (FW) [Frank and Wolfe, 1956] est un algorithme itératif, proposé en 1956 par Marguerite Frank et Philip Wolfe pour la résolution de problèmes d'optimisation convexes contraints. FW converge exponentiellement vite lorsque la fonction objective est fortement convexe [Guélat and Marcotte, 1986], mais linéairement lorsqu'elle est simplement convexe [Frank and Wolfe, 1956]. De plus, la convergence devient

encore plus lente lorsque la solution se situe sur les bords du polytope des contraintes. En effet, dans ce cas les itérés commencent un zig-zag entre les différents sommets du polytope [Lacoste-Julien and Jaggi, 2015]. Des variantes de l'algorithme de FW ont été proposées afin d'accélérer sa convergence en cassant ces zig-zag [Wolfe, 1970, Guélat and Marcotte, 1986, Lacoste-Julien and Jaggi, 2015]. L'essentiel de l'idée proposée par ces méthodes est de faire un « pas » en plus pour s'éloigner de l'approximation actuelle qui cause ces zig-zag.

Des propriétés de convergence de FW dans le cas de fonctions objectives non convexes existent aussi [Lacoste-Julien, 2016].

Au premier abord, on ne soupçonne pas forcément de liens entre FW et MP. Pourtant, nous verrons dans la section 5.3, que FW et MP ont beaucoup de similarités et de points communs. Ces similarités ont inspirés Jaggi et al. [Locatello et al., 2017] pour exploiter les outils traditionnellement utilisés pour l'analyse de FW afin d'analyser MP dans le cas de dictionnaires génériques.

Nous proposons dans ce chapitre de faire le chemin inverse. Nous utilisons les outils traditionnellement utilisés pour l'analyse de MP et OMP afin d'analyser l'algorithme de FW. Nous étendons les résultats de Tropp [Tropp, 2004], Gribonval et Vandergheynst [Gribonval and Vandergheynst, 2006] à l'algorithme de Frank-Wolfe (FW) pour la résolution du Problème (5.2), lorsque le signal y est m-parcimonieux. Nous étudions ensuite ses propriétés de reconstruction et de convergence sous la condition ERC.

En particulier, nous montrons que tout comme MP et OMP, l'algorithme FW choisit à chaque itération un élément du support de y. De plus, il converge exponentiellement vite vers la solution optimale.

La suite de ce chapitre est organisée comme suit : en section 5.2 nous rappelons l'algorithme Frank-Wolfe, que nous instancierons ensuite pour le problème d'approximation parcimonieuse. Nous étudions ses propriétés lorsque y est m-parcimonieux en section 5.3. Enfin, nous évaluerons empiriquement FW sur des données simulées en section 5.4.

5.2 Frank-Wolfe pour l'approximation parcimonieuse

Avant de commencer cette section, fixons quelques notations que nous utiliserons dans la suite. Pour tout dictionnaire $\mathcal{D}$, nous notons par $D := [v_1, \dots, v_n] \in \mathbb{R}^{d \times n}$ la matrice dont les colonnes sont les éléments de $\mathcal{D}$.

Pour tout ensemble $\Lambda \subseteq [n]$, la matrice $D_\Lambda \in \mathbb{R}^{d \times |\Lambda|}$ contient les colonnes de D indexées par Λ, et pour tout vecteur $\alpha \in \mathbb{R}^n$, le vecteur $\alpha_\Lambda \in \mathbb{R}^{|\Lambda|}$ contient les éléments de α aux positions indexées par Λ.

5.2.1 L'algorithme Frank-Wolfe

Frank-Wolfe (FW) [Frank and Wolfe, 1956] est un algorithme itératif proposé par Marguerite Frank et Philip Wolfe pour résoudre des problèmes d'optimisation convexes de la forme :

$$\min_{\alpha \in \mathcal{C}} f(\alpha) \tag{5.5}$$

où f est une fonction convexe et continuement dérivable, et où $\mathcal{C}$ est un ensemble compact et convexe.

À chaque itération, l'algorithme Frank-Wolfe considère une approximation linéaire de f, puis minimise cette approximation dans $\mathcal{C}$ pour trouver la direction de descente. Ces étapes sont résumées dans l'Algorithme 4. Ainsi, à chaque itération k, nous cherchons s_k le minimiseur de l'approximation de Taylor d'ordre 1 de f au tour de α_k. La prochaine approximation α_{k+1} est ensuite construite comme une combinaison convexe de α_k et s_k. Cette combinaison convexe dépend du paramètre γ_k qui définit la proportion de contribution de chacun. Dans le papier original de Frank et Wolfe [Frank and Wolfe, 1956], le paramètre γ_k est calculé comme : $\gamma := \frac{2}{k+2}$. Néanmoins, des méthodes plus avancées peuvent être utilisés sans affecter les propriétés de convergence de Frank-Wolfe. Il est par exemple possible de choisir γ_k tel que $f(\alpha_{k+1})$ soit la plus petite possible :

$$\gamma_k = \arg\min_{\gamma \in [0,1]} f(\alpha_{k+1}) = \arg\min_{\gamma \in [0,1]} f\left(\gamma s_k + (1-\gamma)\alpha_k\right).$$

Pour plus de détails sur les possibles définitions de γ_k, nous invitons le lecteur à voir les travaux de [Locatello et al., 2017] et [Jaggi, 2013].

5.2.2 Frank-Wolfe pour l'approximation parcimonieuse

Nous instancions dans cette section l'Équation (5.5), pour le problème de reconstruction de signaux parcimonieux relaxé. Rappelons que cette relaxation s'écrit comme :

$$\alpha_1 = \arg\min_{\alpha \in \mathbb{R}^n} \left\| y - \sum_{i=1}^{n} \alpha[i] v_i \right\|_2^2 \text{ tel que } \|\alpha\|_1 \leq \beta, \tag{5.2}$$

Algorithme 4 : Algorithme Frank-Wolfe (FW)

Entrée : une fonction convexe f, un ensemble $\mathcal{C}$ convexe.;
Sortie : $\alpha \in \mathcal{C}$, solution du Problème (5.5) ;
choisir $\alpha_0 \in \mathcal{C}$;
$k = 0$;
Répéter {
    $s_k = \arg\min_{s \in \mathcal{C}} \langle s, \nabla f(\alpha_k) \rangle$;
    choisir $\gamma_k \in [0, 1]$:
        variante 1 : $\gamma_k := \frac{2}{k+2}$;
        variante 2 : $\gamma_k = \arg\min_{\gamma \in [0,1]} f\left(\gamma s_k + (1-\gamma)\alpha_k\right)$;
    $\alpha_{k+1} = \gamma_k s_k + (1 - \gamma_k)\alpha_k$;
    $k = k + 1$;
}
Renvoyer α_{k+1} ;

avec β un paramètre à régler. Le Problème (5.2) est alors une instance du Problème (5.5), avec :

$$f(\alpha) = \tfrac{1}{2}\|y - \sum_{i=1}^{n} \alpha[i] v_i\|_2^2 = \tfrac{1}{2}\|y - D\alpha\|_2^2,$$

et

$$\mathcal{C} = \mathcal{B}_1(\beta) := \{\alpha \in \mathbb{R}^n;\ \|\alpha\|_1 \leq \beta\}.$$

Ici $\mathcal{B}_1(\beta)$ est la boule l_1 de rayon β. Elle peut être écrite comme l'ensemble des combinaisons convexes de $\pm\beta e_i$ où e_i est le i^{eme} élément de la base canonique de $\mathbb{R}^n$:

$$\mathcal{B}_1(\beta) = \text{conv}\{\pm\beta e_i \mid i \in [n]\}.$$

La fonction f et l'ensemble $\mathcal{C}$ ainsi définis sont tous deux convexes. Nous pouvons dans ce cas instancier l'Algorithme 4 pour notre problème.

Comme $\nabla f(\alpha_k) = D^T(D\alpha_k - y)$, l'étape de sélection de s de l'algorithme de FW devient :

$$s_k = \underset{s \in \text{conv}\{\pm\beta e_i \mid i \in [n]\}}{\arg\min} \langle s, D^T(D\alpha_k - y) \rangle.$$

Comme ce problème d'optimisation est linéaire et que $\mathcal{B}_1(\beta)$ est fermé et borné, il existe toujours une solution qui est de plus un extremum

de $\mathcal{B}_1(\beta)$ [Boyd and Vandenberghe, 2004]. Ainsi, l'équation précédente devient :

$$s_k = \underset{s\in\{\pm\beta e_i \mid i\in[n]\}}{\arg\min} \langle s, D^T(D\alpha_k - y)\rangle.$$

En utilisant la propriété de $\langle u, Av\rangle = \langle A^T u, v\rangle$ pour tout vecteur u, v et matrice A nous obtenons :

$$s_k = \underset{s\in\{\pm\beta e_i \mid i\in[n]\}}{\arg\min} \langle Ds, (D\alpha_k - y)\rangle.$$

Comme $s = \pm\beta e_i$, alors $Ds = \pm\beta v_i$. Nous en concluons que l'étape de sélection de s_k dans l'algorithme de FW pour le Problème (5.2) est :

$$\begin{cases} i_k &= \arg\max_{i\in[n]} |\langle v_i, y - D\alpha_k, |\rangle \\ s_k &= \text{signe}(\langle v_{i_k}, y - D\alpha_k\rangle)\beta e_{i_k}. \end{cases}$$

En notant $r_k = y - D\alpha_k$ le résidu, nous remarquons que l'étape de sélection de FW est exactement la même que celle de MP et OMP décrite dans l'Algorithme 1.

Pour finir, il ne nous reste plus qu'à définir l'initialisation $\alpha_0 = 0 \in \mathcal{B}_1(\beta)$. Nous résumons ces étapes dans l'Algorithme 5.

Algorithme 5 : Algorithme Frank-Wolfe pour le Problème (5.2)

Entrée : un dictionnaire $\mathcal{D} = \{v_1, \ldots, v_n\}$, un signal y, $\beta \in \mathbb{R}^{+*}$;
Sortie : $\alpha \in \mathcal{C}$, solution du Problème (5.2) ;
$\alpha_0 = 0$;
$k = 0$;
Répéter {
    $i_k = \arg\max_{i\in[n]} |\langle v_i, y - D\alpha_k\rangle|$;
    $s_k = \text{signe}\left(\langle v_{i_k}, y - D\alpha_k\rangle\right) \beta e_{i_k}$;
    choisir $\gamma_k \in [0, 1]$:
        variante 1 : $\gamma_k := \frac{2}{k+2}$;
        variante 2 : $\gamma_k = \arg\min_{\gamma\in[0,1]} \frac{1}{2} \|y - D\left(\alpha_k + \gamma(s_k - \alpha_k)\right)\|_2^2$;
    $\alpha_{k+1} = \alpha_k + \gamma_k \left(s_k - \alpha_k\right)$;
    $k = k + 1$;
}
Renvoyer α_{k+1} ;

Bien que nous ayons dit dans l'introduction de ce chapitre que les méthodes de relaxation ne permettent pas toujours de connaître à chaque

itération le nombre d'atomes qui interviennent dans la décomposition de leur approximation, cela n'est pas vrai pour l'algorithme FW. En effet, à chaque itération de l'Algorithme 5, au plus un seul nouvel atome est choisi. Ainsi, à chaque itération k, l'approximation $D\alpha_k$ est k-parcimonieuse dans $\mathcal{D}$.

Nous présentons dans la section suivante les propriétés de reconstruction et de convergence de l'Algorithme 5 sous la condition $m < \frac{1}{2}(\mu^{-1} + 1)$. Cette condition implique que tout sous-ensemble de m atomes est linéairement indépendant. Aussi, sous cette condition, pour tout signal m-parcimonieux, le vecteur de coefficients α^* qui vérifie $y = D\alpha^*$ et $\|\alpha^*\|_0 \leq m$ est unique [Tropp, 2004]. Dans ce cas, α^* est l'unique solution du Problème (5.1) mais pas toujours solution du problème relaxé (5.2). En effet, nous avons toujours $f(\alpha^*) = 0$, mais si $\|\alpha^*\|_1 > \beta$, alors $\alpha^* \notin \mathcal{B}_1(\beta)$, et donc α^* n'est pas solution de (5.2). Lorsque $\|\alpha^*\|_1 \leq \beta$, alors $\alpha^* \in \mathcal{B}_1(\beta)$, et α^* est bien solution du Problème (5.2).

5.3 Propriétés de reconstruction et de convergence

Nous présentons dans cette section nos principaux résultats. Nous démontrons dans le Théorème 4 que l'algorithme FW choisit uniquement des atomes du support de y. Nous verrons ensuite dans les Théorèmes 5 et 6 qu'il converge exponentiellement vite vers la solution optimale.

5.3.1 Condition de reconstruction

Afin d'étudier les hypothèses sous lesquelles FW reconstruit un signal parcimonieux, nous utiliserons les outils de cohérence [Donoho et al., 2001] et fonction Babel [Tropp, 2004] introduites par Donoho, Huo et Tropp et rappelés ici en section 2.2.3. Nous utiliserons aussi la condition ERC [Tropp, 2004]. Pour tout signal y de support $\Lambda(y)$, l'ERC est vérifiée si :

$$\max_{i \in [n];\, i \notin \Lambda(y)} \left\| D^+_{\Lambda(y)} v_i \right\|_1 < 1 \tag{5.6}$$

où $D_{\Lambda(y)} \in \mathbb{R}^{d \times |\Lambda(y)|}$ est la matrice qui contient les atomes indexés par $\Lambda(y)$ dans ses colonnes.

L'ERC est une condition suffisante pour que OMP reconstruise exactement tout signal m-parcimonieux en exactement m étapes [Tropp, 2004],

et pour que MP converge exponentiellement vite vers la solution optimale [Gribonval and Vandergheynst, 2006]. Nous verrons dans les prochains théorèmes que c'est aussi une condition nécessaire pour que FW converge exponentiellement vite vers la solution optimale.

Cependant, cette condition dépend du support du signal, qui est inconnu. Elle ne peut donc pas être vérifiée en pratique. Néanmoins, il a été démontré [Tropp, 2004] qu'une condition suffisante pour que l'ERC soit vérifiée pour tous les signaux m-parcimonieux est que : $m < \frac{1}{2}(\mu(\mathcal{D})^{-1} + 1)$. Cette dernière condition est quant à elle plus facile à vérifier puisqu'elle ne dépend que de quantités connues et calculables.

La preuve des résultats de convergence de MP et OMP réside dans le fait qu'à chaque itération, l'atome sélectionné appartient au support du signal. Nous étendons ce résultat à l'algorithme de FW dans le théorème suivant.

Théorème 4. *Soit $\mathcal{D} := \{v_1, \dots, v_n\}$ un dictionnaire, $\mu(\mathcal{D})$ sa cohérence, et $y = D\alpha^*$ un signal m-parcimonieux de support $\Lambda(y)$ tel que $\|\alpha^*\|_1 < \beta$. Si*

$$\max_{i \notin \Lambda(y)} \left\| D^+_{\Lambda(y)} v_i \right\|_1 < 1,$$

ou plus généralement si :

$$m < \tfrac{1}{2}(\mu(\mathcal{D})^{-1} + 1),$$

alors à chaque itération k, l'Algorithme 5 choisit un atome correct, i.e. un atome qui appartient au support de y :

$$\forall\, k,\ i_k \in \Lambda(y).$$

Démonstration. La preuve de ce théorème est exactement la même que celle du Théorème 3.1 de [Tropp, 2004] en appliquant le Corollaire 3.6 de [Tropp, 2004].

Nous présentons toutefois quelques éléments de la preuve, pour permettre au lecteur d'avoir une idée de son fonctionnement.

Afin d'alléger les notation, notons par $\Lambda = \Lambda(y)$. Notons aussi par $\bar{\Lambda} = [n] \backslash \Lambda$ le complémentaire de Λ dans $[n]$.

La preuve de ce théorème est une démonstration par récurrence. Nous supposons qu'à l'itération k de l'Algorithme 5 : $i_k \in \Lambda$. Nous souhaitons trouver une condition suffisante pour que le prochain atome sélectionné soit aussi dans le support, i.e. $i_{k+1} \in \Lambda$. Observons que le vecteur $D_\Lambda r_k$ liste l'ensemble des produits scalaires entre r_k et les atomes indexés par Λ. Ainsi, $\|D_\Lambda r_k\|_\infty$ donne la plus grande valeur de ces produits scalaires, où

ici la norme infinie d'un vecteur $a \in \mathbb{R}^d$ est définie comme le maximum de la valeur absolue de ses éléments : $\|a\|_\infty := \max_{i\in[d]} |a[i]|$. De manière similaire, $\|D_{\bar{\Lambda}} r_k\|_\infty$ donne la plus grande valeur des produits scalaires entre r_k et les atomes de $\mathcal{D}$ non indexés par Λ. Ainsi, $i_{k+1} \in \Lambda$, si et seulement si :

$$\frac{\|D_{\bar{\Lambda}} r_k\|_\infty}{\|D_\Lambda r_k\|_\infty} < 1.$$

En utilisant ensuite des bornes usuelles sur les normes vectorielles, et en remarquant que $r_k := y - D\alpha_k \in \text{span}\,(v_i)_{i\in\Lambda}$, car par hypothèse $y_k = D\alpha_k \in \text{span}\,(v_i)_{i\in\Lambda}$, on peut montrer que :

$$\frac{\|D_{\bar{\Lambda}} r_k\|_\infty}{\|D_\Lambda r_k\|_\infty} \leq \max_{i\notin\Lambda(y)} \left\|D^+_{\Lambda(y)} v_i\right\|_1.$$

On en déduit qu'il suffit que $\max_{i\notin\Lambda(y)} \left\|D^+_{\Lambda(y)} v_i\right\|_1 < 1$, pour que $i_{k+1} \in \Lambda$.

On conclut cette preuve en utilisant le Théorème 3.5 et le Corollaire 3.6 de [Tropp, 2004] qui démontrent que si $m < \frac{1}{2}(\mu(\mathcal{D})^{-1} + 1)$ alors $\max_{i\notin\Lambda(y)} \left\|D^+_{\Lambda(y)} v_i\right\|_1 < 1$. □

Ce théorème nous dit que l'algorithme de FW choisit uniquement des éléments de $\mathcal{D}$ qui sont dans le support de y. Cependant, il ne nous dit rien sur la convergence vers la solution optimale. Ce résultat est énoncé dans le corollaire suivant.

Corollaire 3. *Soit $\mathcal{D} := \{v_1, \ldots, v_n\}$ un dictionnaire, $\mu(\mathcal{D})$ sa cohérence, $\beta \in \mathbb{R}^+$, et $y := \sum_{i=1}^n \alpha^*[i] v_i$ un signal m-parcimonieux de support $\Lambda(y)$ tel que $\|\alpha^*\|_1 \leq \beta$. Si*

$$\max_{i\notin\Lambda(y)} \left\|D^+_{\Lambda(y)} v_i\right\|_1 < 1,$$

ou plus généralement si :

$$m < \tfrac{1}{2}(\mu(\mathcal{D})^{-1} + 1),$$

alors la séquence de α_k produite par l'Algorithme 5 converge vers α^*.

Avant de pouvoir démontrer ce corollaire, nous avons besoin d'introduire une remarque.

Remarque 2. *Soit $A \in \mathbb{R}^{d\times n}$ une matrice, $\alpha \in \mathbb{R}^n$ un vecteur tel que $\|\alpha\|_0 = m$, et $\Lambda := \{i \mid \alpha[i] \neq 0\}$. Soit la décomposition en valeurs singulières (SVD) de*

$A_\Lambda \in \mathbb{R}^{d\times|\Lambda|}$: $A_\Lambda = U\Sigma V^T$, *où* $U \in \mathbb{R}^{d\times d}, V \in \mathbb{R}^{|\Lambda|\times|\Lambda|}, \Sigma \in \mathbb{R}^{d\times|\Lambda|}$ *et tel que* $\Sigma[i,i] = \sigma_i(A_\Lambda)$ *est la* i^{eme} *valeur singulière de* A_Λ. *On a alors :*

$$\|A\alpha\|_2 \geq \sigma_{\min}(A_\Lambda)^2 \|\alpha\|_2 \tag{5.7}$$

En effet :

$$\begin{aligned}
\|A\alpha\|_2^2 = \|A_\Lambda\alpha_\Lambda\|_2^2 = \left\|U\Sigma V^T\alpha_\Lambda\right\|_2^2 &\overset{(a)}{=} \left\|\Sigma V^T\alpha_\Lambda\right\|_2^2 \\
&= \sum_{i=1}^{n}\langle V_{:,i},\alpha_\Lambda\rangle^2\varsigma_i(A_\Lambda)^2 \geq \sigma_{\min}(A_\Lambda)^2\sum_{i=1}^{n}\langle V_{:,i},\alpha_\Lambda\rangle^2 \\
&= \sigma_{\min}(A_\Lambda)^2\|V\alpha_\Lambda\|_2^2 \overset{(b)}{=} \sigma_{\min}(A_\Lambda)^2\|\alpha_\Lambda\|_2^2 \\
&= \sigma_{\min}(A_\Lambda)^2\|\alpha\|_2^2
\end{aligned} \tag{5.8}$$

où les égalités (a) *et* (b) *sont vraies car* U *et* V *sont des matrices orthogonales, et préservent donc les normes.*

Passons maintenant à la preuve du Corollaire 3.

Démonstration. Puisque $\|\alpha^*\|_1 \leq \beta$, alors α^* est aussi solution du Problème (5.2). D'un autre côté, il est connu que les valeurs de la fonction objective construites par l'algorithme FW, $f(\alpha_k)$, convergent vers la valeur optimale $f(\alpha^*)$. Nous démontrons ici que les itérés α_k convergent eux aussi vers α^*.

Avant de commencer, introduisons quelques notations. Tout d'abord, afin d'alléger les notations, notons le support de y par Λ au lieu de $\Lambda(y)$. Le vecteur $(\alpha_k)_\Lambda = (\alpha_k[j])_{j\in\Lambda} \in \mathbb{R}^{|\Lambda|}$ (respectivement $\alpha^*_\Lambda = (\alpha^*[j])_{j\in\Lambda} \in \mathbb{R}^{|\Lambda|}$) est le vecteur qui contient les valeurs de α_k (respectivement de α^*) uniquement aux positions indexées par Λ. Enfin, notons par $D_\Lambda \in \mathbb{R}^{d\times|\Lambda|}$ la matrice contenant dans ses colonnes les éléments de $\mathcal{D}$ indexés par Λ.

Ainsi, nous avons :

$$\begin{aligned}
|f(\alpha_k) - f(\alpha^*)| = |f(\alpha_k) - 0| &= \tfrac{1}{2}\|y - D\alpha_k\|_2^2 \\
&= \tfrac{1}{2}\|D\alpha^* - D\alpha_k\|_2^2 \\
&\overset{(a)}{=} \tfrac{1}{2}\|D_\Lambda\alpha^*_\Lambda - D_\Lambda(\alpha_k)_\Lambda\|_2^2 = \tfrac{1}{2}\|D_\Lambda(\alpha^*_\Lambda - (\alpha_k)_\Lambda)\|_2^2 \\
&\geq \tfrac{1}{2}(\varsigma_{min}(D_\Lambda))^2\|\alpha^* - \alpha_k\|_2^2,
\end{aligned}$$

où l'égalité (a) est vraie pour deux raisons. La première est que, comme y est m-parcimonieux alors, $D\alpha^* = D_\Lambda\alpha^*_\Lambda$. La seconde raison est de au

Théorème 4, qui nous dit que pour toute itération k, $i_k \in \Lambda$. Ainsi, α_k ne contient que des zéros dans les positions non indexées par Λ, et nous avons donc que : $D\alpha_k = D_\Lambda (\alpha_k)_\Lambda$. Enfin, la dernière inégalité est vraie pour deux raisons aussi. Premièrement, on a grâce à la Remarque 2 que $\|D_\Lambda\|_{2,2} \geq \varsigma_{min}(D_\Lambda)$. Ensuite, comme α^* (respectivement α_k) contient les mêmes valeurs que α^*_Λ (respectivement que $(\alpha_k)_\Lambda$), avec des zéros en plus, qui n'ont pas d'influence sur la valeur de la norme. Ainsi : $\|\alpha^* - \alpha_k\|_2^2 = \|\alpha^*_\Lambda - (\alpha_k)_\Lambda\|_2^2$.

Enfin, comme $|f(\alpha_k) - f(\alpha^*)|$ converge vers 0, nous avons que $\|\alpha^* - \alpha_k\|_2^2$ aussi qui converge vers 0. □

Ainsi, lorsque la solution optimale α^* est dans la boule $\mathcal{B}_1(\beta)$, et sous condition ERC, alors l'approximation produite par FW converge vers α^*. Notons ici qu'on ne parle pas de reconstruction exacte du signal comme pour OMP, mais plutôt de *convergence* vers la solution optimale comme pour MP (i.e. $\|y - D\alpha_k\|_2$ tend vers zéro lorsque k est grand). Cette différence entre FW et OMP est due au fait qu'à chaque itération k de OMP, le résidu r_k est orthogonal à tous les atomes précédemment sélectionnés. Ainsi, chaque atome n'est sélectionné qu'une seule fois. Si la condition ERC est vérifiée, OMP ne sélectionne que des atomes du support [Tropp, 2004]. Le support étant de taille m, OMP converge alors au bout de m itérations.

Le résidu de FW n'est quant à lui orthogonal qu'au dernier atome sélectionné, et non pas à tous ceux sélectionnés aux itérations précédentes. FW a donc probablement besoin de sélectionner un même atome plusieurs fois. Même si la taille du support est finie, et que FW ne sélectionne que des atomes du support, il risque de faire plusieurs itérations avant de retrouver la solution optimale.

Nous verrons toutefois dans la section suivante, que si la condition ERC est vérifiée, FW converge exponentiellement vite vers la solution optimale.

5.3.2 Vitesse de convergence

Lorsque l'algorithme FW est utilisé pour des problèmes d'optimisation génériques comme le Problème (5.5), ses propriétés de convergence sont connues. Il converge exponentiellement vite vers la solution optimale lorsque la fonction objective f est fortement convexe [Guélat and Marcotte, 1986] (en particulier elle admet un unique minimum), et linéairement dans les autres cas [Frank and Wolfe, 1956].

Dans notre cas, comme $f(\alpha) = \frac{1}{2}\|y - D\alpha_k\|_2^2$, et comme $\mathcal{D}$ est redondant, alors f admet plusieurs minimas. La fonction objective n'est donc

pas fortement convexe, et la convergence de FW devrait être linéaire. Toutefois, nous démontrons dans le théorème ci-dessous que si la condition ERC est vérifiée, alors FW converge exponentiellement vite vers la solution optimale.

En réalité, la condition ERC nous permet de nous assurer qu'à chaque itération $k : \Lambda(y_k) \subseteq \Lambda(y)$, et donc $D\alpha_k = D_{\Lambda(y)}(\alpha_k)_{\Lambda(y)}$. Rappelons que les notations $D_{\Lambda(y)}$ et $(\alpha_k)_{\Lambda(y)}$ sont introduites au début de la section 5.2, et désignent respectivement la matrice dont les colonnes sont les éléments de D indexés par $\Lambda(y)$ et le vecteur contenant les éléments de α_k indexés par $\Lambda(y)$. On a alors :

$$f(\alpha_k) = \frac{1}{2}\|y - D\alpha_k\|_2^2 = \frac{1}{2}\left\|y - D_{\Lambda(y)}(\alpha_k)_{\Lambda(y)}\right\|_2^2.$$

Ainsi, grâce à l'ERC la fonction f agit comme une fonction fortement convexe sur l'ensemble des vecteurs α dont les positions non nulles sont indexées par $\Lambda(y)$. Cela nous permettra de démontrer la convergence exponentielle. Nous formalisons ce résultat dans le théorème suivant.

Théorème 5. *Soit $\mathcal{D} := \{v_1, \ldots, v_n\}$ un dictionnaire, $\mu(\mathcal{D})$ sa cohérence, $\beta \in \mathbb{R}^+$, $y := \sum_{i=1}^n \alpha^*[i]v_i$ un signal m-parcimonieux tel que $\|\alpha^*\|_1 < \beta$, et $\mu_1(\mathcal{D}, m)$ la fonction de Babel de $\mathcal{D}$. Si*

$$m < \tfrac{1}{2}(\mu(\mathcal{D})^{-1} + 1),$$

et que pour chaque itération k, le paramètre γ_k est défini suivant la variante 2, alors il existe $j \geq 1$ tel que pour toute itération $k \geq j$, l'Algorithme 5 vérifie :

$$\|y - y_{k+1}\|_2^2 \leq \|y - y_k\|_2^2(1 - \theta)$$

où $y_k := \sum_{i=1}^n \alpha_k[i]v_i$ et $\theta = \frac{1}{16}\left(\frac{1-\mu_1(\mathcal{D},m-1)}{m}\right)\left(1 - \frac{\|\alpha^*\|_1}{\beta}\right)^2$.

Afin de démontrer ce théorème, nous avons besoin d'introduire un résultat intermédiaire qui nous donne la solution analytique de γ_k.

Lemme 7. *Soit $\mathcal{D} := \{v_1, \ldots, v_n\}$ un dictionnaire, $\beta \geq 0$, et y un signal m-parcimonieux tel que $\|y\|_2 < \beta$. Pour toute itération k de l'Algorithme 5, si le paramètre γ_k est défini suivant la variante 2, et si $\gamma_k \neq 0$, alors :*

$$\gamma_k = \frac{\langle D(s_k - \alpha_k), r_k\rangle}{\|D(s_k - \alpha_k)\|_2^2}.$$

Démonstration. Rappelons d'abord que :

$$\gamma_k = \underset{\gamma \in [0,1]}{\arg\min} \|y - D(\alpha_k + \gamma(s_k - \alpha_k))\|_2^2 .$$

et définissons :

$$\gamma_k^* = \underset{\gamma \in \mathbb{R}}{\arg\min} \|y - D(\alpha_k + \gamma(s_k - \alpha_k)\|_2^2 .$$

Ce problème d'optimisation étant quadratique et non contraint, sa solution peut facilement être calculée :

$$\gamma_k^* = \frac{\langle D(s_k - \alpha_k), r_k \rangle}{\|D(s_k - \alpha_k)\|_2^2}.$$

Nous souhaitons ici montrer que $\gamma_k = \gamma_k^*$.

Comme γ_k est la solution du même problème que γ_k^*, mais restreinte à $[0, 1]$, nous avons trois cas de figures possibles :

— $\gamma_k^* \geq 1$, et dans ce cas $\gamma_k = 1$.

— $0 \leq \gamma_k^* \leq 1$, et dans ce cas $\gamma_k = \gamma_k^*$.

— $\gamma_k^* \leq 0$, et dans ce cas $\gamma_k = 0$.

Comme nous supposons que $\gamma_k \neq 0$, alors la troisième possibilité est éliminée. Il suffit maintenant de montrer que la première possibilité n'est pas possible aussi, afin de terminer notre preuve. Pour cela, considérons ces deux cas :

— $k = 0$: comme $\alpha_0 = 0$ et $r_0 = y$, et comme pour tout $j \in [n] : \left\|v_j\right\|_2 = 1$, nous avons :

$$\gamma_0^* = \frac{\langle Ds_0, y \rangle}{\|Ds_0\|_2^2} = \frac{\langle \pm\beta v_{i_k}, y \rangle}{\left\|\pm\beta v_{i_k}\right\|_2^2} \leq \frac{\|y\|_2}{\beta}.$$

Or, comme $\|y\|_2 < \beta$, alors $\gamma_0^* < 1$.

— $k \neq 0$: Démontrons par l'absurde que $\gamma_k \neq 1$. Supposons que $\gamma_k = 1$. Nous avons ainsi que $\alpha_{k+1} = s_k$. Par construction de l'algorithme de FW, nous obtenons :

$$f(\alpha_{k+1}) = f(s_k) \leq f(\alpha_k) \leq \cdots \leq f(\alpha_1) = f(\gamma_0 s_0) .$$

Comme nous avons déjà démontré que $\gamma_0 \neq 1$, alors nous avons $f(\alpha_1) = f(\alpha_0 + \gamma_0(s_0 - \alpha_0)) = f(\gamma_0 s_0) < f(s_0)$.

Cela implique que $f(s_k) < f(s_0)$, c'est-à-dire :

$$\|y - Ds_k\|_2^2 < \|y - Ds_0\|_2^2,$$

ou encore :

$$\langle y, Ds_0\rangle_2^2 < \langle y, Ds_k\rangle_2^2,$$

cela est clairement une contradiction puisque $s_0 = \arg\max_{s\in\mathcal{B}_1(\beta)} |\langle Ds, y\rangle|$. Ainsi, $\gamma_k \neq 1$ et donc $0 < \gamma_k < 1$ pour toute itération k. On en conclut que $\gamma_k = \gamma^* = \frac{\langle D(s_k-\alpha_k), r_k\rangle}{\|D(s_k-\alpha_k)\|_2^2}$

□

Démonstration du Théorème 5. Nous donnons ici des éléments de la preuve. La démonstration complète est disponible dans l'Annexe B.1.

Soit k une itération de l'Algorithme 5. Comme $\gamma_k \in [0, 1]$, nous pouvons découper cette plage de valeurs en deux parties :

— $\gamma_k = 0$: dans ce cas, $\alpha_{k+1} = \alpha_k$, et ainsi pour toute itération $l \geq k$ nous aurons : $f(\alpha_k) = f(\alpha_l)$. Comme les valeurs de la fonction objective convergent vers $f(\alpha^*) = 0$, nous en déduisons que $f(\alpha_l) = f(\alpha^*) = 0$. Par définition de f, nous avons donc : $\|r_{k+1}\|_2^2 = \|r_k\|_2^2 = 0$. Ainsi, le Théorème 5 est vrai.

— $0 < \gamma_k \leq 1$: en utilisant la définition du résidu, ainsi que le Lemme 7, nous avons :

$$\|r_{k+1}\|_2^2 = \|r_k\|_2^2 - \frac{\langle D(s_k - \alpha_k), r_k\rangle^2}{\|D(s_k - \alpha_k)\|_2^2}. \tag{5.9}$$

Nous devons maintenant trouver une borne inférieure à $\langle D(s_k - \alpha_k), r_k\rangle^2$, et une borne supérieure à $\|D(s_k - \alpha_k)\|_2^2$. Pour borner $\|D(s_k - \alpha_k)\|_2^2$, nous utilisons le fait que pour tout vecteur a : $\|Da\|_2 \leq \|a\|_1$ et le fait que s_k et α_k sont dans la boule $\mathcal{B}_1(\beta)$:

$$\|D(s_k - \alpha_k)\|_2^2 \leq \|s_k - \alpha_k\|_1^2 \leq 4\beta^2. \tag{5.10}$$

Trouvons maintenant une borne inférieure à $\langle D(s_k - \alpha_k), r_k\rangle^2$. D'après le Corrolaire 3, les itérés α_k convergent vers α^*. Il existe alors une itération l à partir de laquelle pour tout $k \geq l$: $\|\alpha_k - \alpha^*\|_1 \leq \epsilon := \frac{\beta - \|\alpha^*\|_1}{2}$.

En utilisant le Théorème 4, la Remarque 2, et le Lemme 2.3 de [Tropp, 2004], nous pouvons montrer que :

$$\langle D(s_k - \alpha_k), r_k\rangle \geq \epsilon\sqrt{\frac{1 - \mu_1(\mathcal{D}, m-1)}{m}}\,\|r_k\|_2\,. \tag{5.11}$$

Enfin, il suffit d'insérer les Équations (5.10) et (5.11) dans l'Équation (5.9), pour obtenir :

$$\begin{aligned}\|r_{k+1}\|^2 &= \|r_k\|^2 - \frac{\langle D(s_k - \alpha_k), r_k\rangle^2}{\|D(s_k - \alpha_k)\|^2} \\ &\leq \|r_k\|^2 \left(1 - \frac{\epsilon^2(1 - \mu_1(\mathcal{D}, m-1))}{4\beta^2 m}\right) \\ &\leq \|r_k\|^2 \left(1 - \frac{1}{16}\left(\frac{1-\mu_1(\mathcal{D},m-1)}{m}\right)\left(1 - \frac{\|\alpha^*\|_1}{\beta}\right)^2\right),\end{aligned}$$

où la dernière égalité est vraie car : $\frac{\epsilon^2}{4\beta^2} = \frac{1}{16}\left(1 - \frac{\|\alpha^*\|_1}{\beta}\right)$. ce qui termine notre preuve.

□

Afin de démontrer la convergence exponentielle, il suffit maintenant de montrer que $0 < \theta \leq 1$. Nous faisons cela dans le lemme suivant :

Lemme 8. *Soit $\mathcal{D} := \{v_1, \ldots, v_n\}$ un dictionnaire, $\mu(\mathcal{D})$ sa cohérence, $\beta \in \mathbb{R}^+$, $y := \sum_{i=1}^n \alpha^*[i]v_i$ un signal m-parcimonieux tel que $\|\alpha^*\|_1 < \beta$, et $\mu_1(\mathcal{D}, m)$ la fonction de Babel de $\mathcal{D}$. On a alors :*

$$0 < \frac{1}{16}\left(\frac{1-\mu_1(\mathcal{D},m-1)}{m}\right)\left(1 - \frac{\|\alpha^*\|_1}{\beta}\right)^2 \leq 1$$

Démonstration. comme $\mu_1(\mathcal{D}, m-1) \leq \mu(\mathcal{D})(m-1)$ [Tropp, 2004], et que $m < \frac{1}{2}(\mu(\mathcal{D})^{-1} + 1)$ cela implique que $0 < \mu_1(\mathcal{D}, m-1) < 1$. Enfin, comme $\|\alpha^*\|_1 < \beta$ on a donc $0 < \frac{1}{16m}\left((1 - \mu_1(\mathcal{D}, m-1)\right)\left(1 - \frac{\|\alpha^*\|_1}{\beta}\right)^2 \leq 1$. □

Ainsi, Théorème 5 prouve que FW converge exponentiellement vite vers la solution optimale à partir d'une certaine itération.

Contrairement au Théorème 4 et au Corollaire 3, qui ne dépendent pas de comment le paramètre γ_k de l'Algorithme 5 est défini, le Théorème 5 n'est valide que pour une seule variante de calcul de γ_k. Il n'est toutefois pas dit que d'autres variantes de γ_k ne permettent pas un tel résultat. Cette question pourrait d'ailleurs faire l'objet d'une étude dans un prochain travail.

Au vu de ces résultats de convergence, une question naturelle peut surgir : est-il possible de garantir une convergence exponentielle dès la première itération ? Nous montrons dans le théorème suivant que c'est le cas, lorsque β est assez grand.

En effet, la preuve du Théorème 5 repose sur le fait que lorsque les approximations α_k construites par l'Algorithme 5 sont à une distance $\epsilon := \frac{\beta - \|\alpha^*\|_1}{2}$ de α^*, alors FW converge exponentiellement vite vers la solution optimale. Comme les itérés α_k convergent vers α^*, on sait qu'il existe une itération l à partir de laquelle α_k sera à une distance ϵ de α^* pour tout $k \geq l$. Ainsi, plus ϵ est grand, plus cela arrivera tôt (i.e. plus l sera petit). En particulier, pour une valeur de ϵ assez grande, α_k sera à une distance ϵ de α^* dès la première itération. Il y aura dans ce cas, convergence exponentielle de l'Algorithme FW dès la première itération. Pour que ϵ soit assez grand, il suffit de prendre β assez grand.

Nous formalisons cette idée dans le Théorème 6.

Théorème 6. *Soit $\mathcal{D} := \{v_1, \ldots, v_n\}$ un dictionnaire, $\mu(\mathcal{D})$ sa cohérence, $\beta \in \mathbb{R}^+$, $y := \sum_{i=1}^{n} \alpha^*[i] v_i$ un signal m-parcimonieux, et $\mu_1(\mathcal{D}, m)$ la fonction de Babel de $\mathcal{D}$. Si $m < \frac{1}{2}(\mu(\mathcal{D})^{-1} + 1)$, et que*

$$\beta > 2\|y\|_2 \sqrt{\frac{m}{1-\mu_1(\mathcal{D}, m-1)}},$$

alors l'Algorithme 5 converge exponentiellement vite dès la première itération.

Avant de présenter la preuve du Théorème 6, nous avons besoin d'introduire un nouveau résultat qui nous permet de borner la norme des itérés α_k.

Lemme 9. *Soit $\mathcal{D}$ un dictionnaire de cohérence $\mu(\mathcal{D})$ et $\mu_1(\mathcal{D}, m)$ sa fonction Babel, y un signal m-parcimonieux dans $\mathcal{D}$. Si $m < \frac{1}{2}(\mu(\mathcal{D})^{-1} + 1)$, alors à chaque itération k de l'Algorithme 5 :*

$$\|\alpha_k\|_1 \leq 2\|y\|_2 \sqrt{\frac{m}{1-\mu_1(\mathcal{D}, m-1)}}.$$

Démonstration. Par le Lemme 2.3 de [Tropp, 2004], nous avons que : $\lambda_{min}\left(D_{\Lambda(y)}\right)^2 \geq 1 - \mu_1(\mathcal{D}, m-1)$. Ainsi, en utilisant la Remarque 2, nous avons :

$$\|D\alpha_k\|_2 \geq \lambda_{min}\left(D_{\Lambda(y)}\right)\|\alpha_k\|_2 \geq \sqrt{1 - \mu_1(\mathcal{D}, m-1)}\,\|\alpha_k\|_2 .$$

Or, d'après le Théorème 4, α_k contient au plus m valeurs non nulles, nous avons donc : $\|\alpha_k\|_1 \leq \sqrt{m}\,\|\alpha_k\|_2$, et alors :

$$\|\alpha_k\|_1 \leq \sqrt{\frac{m}{1-\mu_1(\mathcal{D}, m-1)}}\,\|D\alpha_k\|_2 .$$

Il ne nous reste maintenant plus qu'à borner $\|D\alpha_k\|_2$:

$$\begin{aligned}\|D\alpha_k\|_2 &\leq \|D\alpha_k - y\|_2 + \|y\|_2 \\ &\leq \sqrt{2f(\alpha_k)} + \|y\|_2 \\ &\leq \sqrt{2f(\alpha_0)} + \|y\|_2 \\ &\leq \|y\|_2 + \|y\|_2 \leq 2\|y\|_2,\end{aligned}$$

où la troisième inégalité est vraie car par construction de l'algorithme de FW, la suite $\{f(\alpha_k)\}_k$ est décroissante. On conclut alors que $\|\alpha_k\|_1 \leq 2\|y\|_2 \sqrt{\frac{m}{1-\mu_1(\mathcal{D},m-1)}}$ pour tout k. □

Nous pouvons maintenant nous consacrer à la preuve du Théorème 6.

Démonstration du Théorème 6. Par la Remarque 3, nous savons que si $\beta > 2\|y\|_2 \sqrt{\frac{m}{1-\mu_1(\mathcal{D},m-1)}}$, alors $\beta \geq \|\alpha^*\|_1$. Nous somme donc sous les hypothèses du Théorème 5, et nous pouvons alors ré-utiliser une partie de sa preuve. Ainsi, nous avions démontré que :

$$\|r_{k+1}\|_2^2 = \|r_k\|_2^2 - \frac{\langle D(s_k - \alpha_k), r_k\rangle^2}{\|D(s_k - \alpha_k)\|_2^2}, \tag{5.9}$$

et que :

$$\|D(s_k - \alpha_k)\|_2^2 \leq \|s_k - \alpha_k\|_1^2 \leq 4\beta^2. \tag{5.10}$$

Nous proposons ici de borner différemment $\langle D(s_k - \alpha_k), r_k\rangle^2$. Par définition de s_k, nous avons :

$$\langle Ds_k, r_k\rangle = \beta \max_i |\langle v_i, r_k\rangle|,$$

et

$$\begin{aligned}\langle D\alpha_k, r_k\rangle &= \langle \alpha_k, D^T r_k\rangle = \sum_i (\alpha_k)[i](D^T r_k)[i] \\ &\leq \|\alpha_k\|_1 \max_i |(D^T r_k)[i]| \\ &= \|\alpha_k\|_1 \max_i |\langle v_i, r_k\rangle|.\end{aligned}$$

On en conclut que :

$$\begin{aligned}\langle D(s_k - \alpha_k), r_k\rangle &= \langle Ds_k, r_k\rangle - \langle D\alpha_k, r_k\rangle \\ &\geq \beta \max_i |\langle v_i, r_k\rangle| \left(1 - \frac{\|\alpha_k\|_1}{\beta}\right).\end{aligned}$$

Par le Lemme 2 de [Gribonval and Vandergheynst, 2006] :

$$\max_i |\langle v_i, r_k \rangle| \geq \|r_k\|_2 \sqrt{\frac{1-\mu_1(\mathcal{D},m-1)}{m}}.$$

Ainsi :

$$\langle D(s_k - \alpha_k), r_k \rangle \geq \beta \|r_k\|_2 \sqrt{\frac{1-\mu_1(\mathcal{D},m-1)}{m}} \left(1 - \frac{\|\alpha_k\|_1}{\beta}\right). \tag{5.12}$$

Il nous reste à démontrer que $1 - \frac{\|\alpha_k\|_1}{\beta}$ est inférieurement borné par 0. En utilisant le Lemme 9, nous obtenons :

$$1 - \frac{\|\alpha_k\|_1}{\beta} \geq 1 - \frac{2\|y\|_2}{\beta} \sqrt{\frac{m}{1-\mu_1(\mathcal{D},m-1)}}. \tag{5.13}$$

Par hypothèse, nous avons $1 - \frac{2\|y\|_2}{\beta} \sqrt{\frac{m}{1-\mu_1(\mathcal{D},m-1)}} > 0$, et nous en déduisons que :

$$\langle D(s_k - \alpha_k), r_k \rangle \geq \beta \|r_k\|_2 \sqrt{\frac{1-\mu_1(\mathcal{D},m-1)}{m}} (1 - \tau) > 0,$$

avec :

$$\tau = \frac{2\|y\|_2}{\beta} \sqrt{\frac{m}{1-\mu_1(\mathcal{D},m-1)}}.$$

En utilisant ce résultat ainsi que l'Équation (5.10) dans l'Équation (5.9) nous obtenons :

$$\begin{aligned} \|r_{k+1}\|_2^2 &= \|r_k\|_2^2 - \frac{\langle D(s_k - \alpha_k), r_k \rangle^2}{\|D(s_k - \alpha_k)\|^2} \\ &\leq \|r_k\|_2^2 - \|r_k\|_2^2 \frac{1 - \mu_1(\mathcal{D}, m-1)}{4m} (1 - \tau)^2. \end{aligned}$$

On en conclut que pour tout k :

$$\|r_{k+1}\|_2^2 \leq \|r_k\|_2^2 \left(1 - \frac{(1 - \mu_1(\mathcal{D}, m-1))}{4m} (1 - \tau)^2\right),$$

avec $0 < \left(1 - \frac{(1-\mu_1(\mathcal{D},m-1))}{4m} (1 - \tau)^2\right) < 1$ ce qui prouve la convergence exponentielle dès la première itération. □

Remarquons que le Théorème 6 ne requiert pas explicitement que $\alpha^* \in \mathcal{B}_1(\beta)$. En réalité, nous pouvons démontrer que lorsque $\beta > 2\|y\|_2 \sqrt{\frac{m}{1-\mu_1(\mathcal{D},m-1)}}$, alors : $\|\alpha^*\|_1 < \beta$, et donc $\alpha^* \in \mathcal{B}_1(\beta)$.

Remarque 3. *En utilisant la Remarque 2, nous avons :*

$$\begin{aligned}\|y\|_2 = \|D\alpha^*\|_2 = \left\|D_{\Lambda(y)}\alpha^*_{\Lambda(y)}\right\|_2 &\geq \lambda_{\min}\left(D_{\Lambda(y)}\right)\|\alpha^*\|_2 \\ &\overset{(b)}{\geq} \sqrt{1-\mu_1\left(\mathcal{D}, m-1\right)}\,\|\alpha^*\|_2\,,\end{aligned}$$

où l'inégalité (b) est vraie par le Lemme 2.3 de [Tropp, 2004]. Or, comme y est m-parcimonieux, on en conclut que :

$$\begin{aligned}\|\alpha^*\|_1 = \left\|(\alpha^*)_{\Lambda(y)}\right\|_1 &\leq \sqrt{m}\left\|(\alpha^*)_{\Lambda(y)}\right\|_1 = \sqrt{m}\,\|\alpha^*\|_2 \\ &\leq \|y\|_2\sqrt{\frac{m}{1-\mu_1\left(\mathcal{D}, m-1\right)}} \\ &< \frac{1}{2}\beta \leq \beta.\end{aligned}$$

Par ailleurs, notons que la condition $\beta > 2\|y\|_2\sqrt{\frac{m}{1-\mu_1(\mathcal{D},m-1)}}$ est plus facile à vérifier en pratique que $\|\alpha^*\|_1 \leq \beta$, puisqu'elle ne dépend pas du vecteur inconnu α^*.

Enfin, nous terminons ce chapitre par montrer que les itérées $\{\alpha_k\}_k$ convergent aussi exponentiellement vite vers α^*.

Théorème 7. *Soit $\mathcal{D}$ un dictionnaire, $\mu(\mathcal{D})$ sa cohérence, $\beta \in \mathbb{R}^+$, $y := \mathcal{D}\alpha^*$ un signal m-parcimonieux dans $\mathcal{D}$ tel que $\|\alpha^*\|_1 < \beta$. Si*

$$m < \tfrac{1}{2}(\mu(\mathcal{D})^{-1} + 1),$$

alors il existe une itération k à partir de laquelle FW vérifie :

$$\|\alpha_k - \alpha^*\|_2^2 \leq \frac{\|y\|_2^2}{\sigma_{min}\left(D_\Lambda\right)}(1-\theta)^k$$

où $\theta = \frac{1}{16}\left(\frac{1-\mu_1(\mathcal{D},m-1)}{m}\right)\left(1 - \frac{\|\alpha^*\|_1}{\beta}\right)^2 \in]0,1]$.

Démonstration. Commençons par utiliser la définition de la fonctionnelle $f(\alpha) = \frac{1}{2}\|y - D\alpha_k\|_2^2$ ainsi que le fait que y est m-parcimonieux dans $\mathcal{D}$:

$$\begin{aligned}|f(\alpha_k) - f(\alpha^*)| = |f(\alpha_k) - 0| &= \tfrac{1}{2}\|y - D\alpha_k\|_2^2 \\ &= \tfrac{1}{2}\left\|D\alpha^* - D\alpha_k\right\|_2^2 \\ &= \tfrac{1}{2}\left\|D_\Lambda\alpha^*_\Lambda - D_\Lambda\left(\alpha_k\right)_\Lambda\right\|_2^2 \\ &\geq \tfrac{1}{2}\left(\sigma_{min}\left(D_\Lambda\right)\right)^2\left\|\alpha^* - \alpha_k\right\|_2^2,\end{aligned}$$

Comme $\|\alpha^*\|_1 < \beta$, alors α^* est solution possible de notre problème. Enfin, en utilisant le Théorème 5, nous obtenons le résultat souhaité. □

5.4 Expériences numériques

Le Théorème 5 montre que l'Algorithme 5 converge exponentiellement vite lorsque m est relativement petit ($m \leq \frac{1}{2}(\mu(\mathcal{D})^{-1}+1)$) et que β est assez grand (i.e. $\beta \geq \|\alpha^*\|_1$).

Le but de cette section est de vérifier empiriquement si ces conditions ainsi que la borne du Théorème 5 sont serrées i.e. si elles sont optimales ou si elles peuvent encore être améliorées.

Nous proposons pour cela deux expériences. Une première que nous réaliserons sous les hypothèses du Théorème 5. Cette expérience nous permettra de vérifier si la borne sur la vitesse de convergence est optimale ou si elle peut encore être améliorée. La deuxième expérience sera réalisée en dehors des hypothèses du Théorème 5, afin de voir si la convergence exponentielle est encore constatée.

Nous proposons d'effectuer ces expériences sur des données synthétiques. Nous générons des signaux de dimension $d = 10000$ qui sont parcimonieux dans des dictionnaires de $n = 20000$ atomes. Les dictionnaires sont générés aléatoirement suivant une loi normale. La cohérence moyenne de ces dictionnaires est de $\mu(\mathcal{D}) = 5.8 \times 10^{-2}$, et $m^* := \lceil \frac{1}{2}\left(\mu(\mathcal{D})^{-1}-1\right) \rceil = 8$ est la plus grande valeur permettant de vérifier les conditions du Théorème 5.

Les signaux sont aussi créés aléatoirement. Le support de taille m est choisi uniformément tandis que les coefficients correspondants sont tirés suivant une loi normale.

Pour toutes nos expériences, l'Algorithme 5 est lancé sur 2000 tirages différents de dictionnaires et de signaux.

La convergence exponentielle est constatée dans le Théorème 5 par la relation :

$$\|r_k\|_2^2 \leq \|r_{k-1}\|_2^2 (1-\theta),$$

où $r_k := y - D\alpha_k$ pour tout k. Cette relation est équivalente à :

$$\|r_k\|_2^2 \leq \|y\|_2^2 (1-\theta)^k,$$

puisque $r_0 := y - D\alpha_0 = y$.

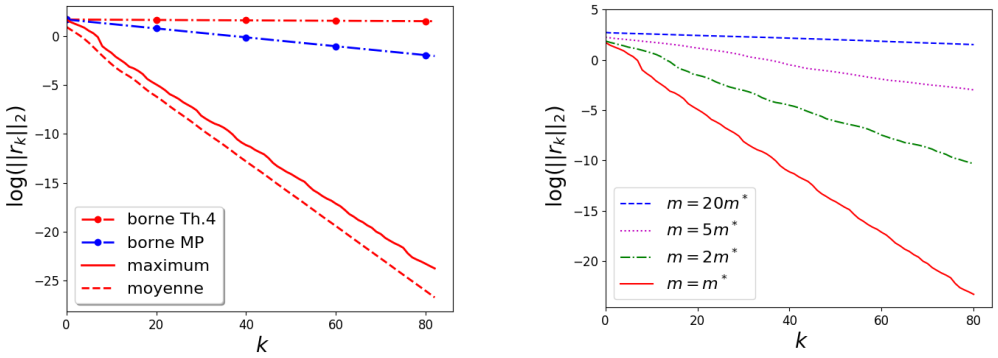
En appliquant maintenant la fonction logarithmique sur les deux côtés de l'inégalité nous obtenons :

$$\log\left(\|r_k\|_2\right) \leq \log\left(\|y\|_2\right) + \frac{k}{2}\log\left(1-\theta\right).$$

Pour nos deux expériences, nous traçons la quantité $\log\left(\|r_k\|_2\right)$ en fonction des itérations k. Lorsque cette courbe peut être supérieurement bornée

par une droite de pente négative, on dira que l'Algorithme 5 converge exponentiellement vite. Plus cette pente est raide, plus la convergence est rapide.

Nous commençons par nous placer dans les hypothèses du Théorème 5. Nous prenons $\beta = 8\|\alpha^*\|_1$, $m = m^*$, et nous traçons dans la Figure 5.1a le maximum et la moyenne de $\log(\|r_k\|_2)$ en fonction de k. Nous comparons ces résultats à la borne du Théorème 5. Nous traçons aussi la borne théorique de l'algorithme MP proposée dans [Gribonval and Vandergheynst, 2006].



(a) $m = m^*$ (b) Différentes valeurs de m

FIGURE 5.1 – Norme du résidu en fonction des itérations k, avec $\beta = 8\|\alpha^*\|_1$.

Sans surprise, nous remarquons que la fonction $\log(\|r_k\|_2)$ peut facilement être bornée par une droite de pente négative, ce qui atteste de la convergence exponentielle. Nous remarquons aussi, qu'en pratique, le maximum et la moyenne de $\log(\|r_k\|_2)$ sont tous deux inférieurs à la borne théorique. Cela suggère que la borne du Théorème 5 est sujette à amélioration. Notons que la droite de la borne du Théorème 5 semble plate. Toutefois, ce n'est pas le cas, elle a juste une pente de -0.002, qui est donc très proche de 0 par rapport aux autres.

Pour notre deuxième expérience, nous souhaitons voir si la convergence exponentielle est toujours possible avec des valeurs de parcimonie m plus grandes que celles prédites par le Théorème 5. Ainsi, nous considérons les valeurs de m suivantes : $m = 2m^*$, $m = 5m^*$, et $m = 20m^*$.

Nous observons dans la Figure 5.1b que la convergence demeure exponentielle pour $m = 2m^*$ et $m = 5m^*$. Cependant, c'est moins le cas pour $m = 20m^*$. Cette expérience suggère qu'en pratique FW peut reconstruire très rapidement un ensemble de signaux plus grand que ceux prédits par

le Théorème 5. Il y a donc peut-être une possibilité d'améliorer l'hypothèse sur m pour que le Théorème 5 soit aussi valide pour des valeurs de $m \geq m^*$.

5.5 Conclusion

Dans ce chapitre, nous avons étudié les propriétés de l'algorithme Frank-Wolfe dans le cas de la reconstruction de signaux parcimonieux. Nous avons démontré que tout comme les algorithmes MP et OMP, lorsque le signal est de parcimonie assez faible par rapport à la cohérence du dictionnaire, alors l'algorithme de Frank-Wolfe sélectionne uniquement des éléments du support. Nous avons aussi démontré que sous les mêmes conditions, il converge exponentiellement vite. Enfin, nous avons empiriquement évalué l'algorithme Frank-Wolfe dans le cas de la reconstruction de signaux parcimonieux. Nous avons observé que nos hypothèses sur la parcimonie des signaux est sujette à amélioration, et que la borne sur la vitesse de convergence n'est en pratique pas serrée.

Des travaux futurs pourraient inclure l'étude de la vitesse de convergence de Frank-Wolfe lorsque γ_k est calculé suivant la variante 1. En effet, cette variante étant moins coûteuse que la variante 2, elle pourrait permettre une baisse du coût de calcul. Étendre ces travaux au cas de signaux non parcimonieux est aussi une prochaine étape naturelle. Enfin, il peut aussi être intéressant d'étudier les propriétés de l'algorithme de Frank-Wolfe dans la reconstruction de signaux parcimonieux, sous des hypothèses RIP, ou encore dans le cas de dictionnaire cohérent. En d'autres termes, est-il possible d'avoir des résultats de convergence et de reconstructions similaires à ceux obtenus dans ce chapitre en s'affranchissant de l'hypothèse sur la cohérence du dictionnaire ?

Sur un plan expérimental, une étude impliquant des dictionnaires et signaux générés différemment que suivant une loi normale permettrait de nous conforter sur l'optimalité de nos hypothèses et bornes. Aussi, il serait intéressant de comparer la qualité des signaux approximés construits par Frank-Wolfe avec ceux construits par MP et OMP.

Chapitre 6

Élagage de forêts aléatoires avec OMP

Nous avons étudié dans le chapitre 5 les propriétés de reconstruction et de convergence de l'algorithme Frank-Wolfe (FW) dans l'approximation de signaux par des représentations parcimonieuses. Nous étudions dans ce qui suit l'élagage de forêts aléatoires avec l'algorithme *Orthogonal Matching Pursuit* (OMP).

6.1 Introduction et état de l'art

Les travaux que nous présentons dans ce chapitre sont légèrement différents des autres. Même si nous utilisons des outils et des méthodes vus dans les chapitres précédents, les résultats présentés ici sont purement expérimentaux. Il nous paraît pertinent de souligner que ces travaux s'inscrivent dans un projet qui a été initié et mené par un groupe constitué uniquement de doctorants et post doctorants de l'équipe Qarma du LIS, et qu'ils ont été publiés et présentés à la conférence CAP 2020 [Giffon et al., 2020].

6.1.1 Introduction

Une forêt aléatoire [Breiman, 2001] est un modèle d'apprentissage automatique pour la classification supervisée ou la régression. Elle est constituée de plusieurs arbres de décision, dont les sorties sont « combinées » pour former la décision de la forêt aléatoire. Dans son papier fondateur, Breiman [Breiman, 2001] explique que l'augmentation de la taille de la forêt, donc du nombre d'arbres qui la constitue, ne cause pas de sur-apprentissage.

Mieux encore, il montre que l'erreur de généralisation de la forêt aléatoire décroit à mesure que sa taille augmente.

Une conséquence de ce résultat est l'apparition de forêts aléatoires de plus en plus grandes, et de fait, de moins en moins interprétables. De plus, ces forêts de grandes tailles sont volumineuses et donc difficiles à stocker sur des appareils de petites mémoires tel que les téléphones.

Une approche permettant de réduire la taille des forêts aléatoires est l'utilisation de méthodes d'élagage [Kulkarni and Sinha, 2012]. Ces méthodes construisent une forêt aléatoire en choisissant m arbres à partir d'une forêt initiale de taille $l \gg m$.

Les auteurs de [Kulkarni and Sinha, 2012] proposent de classer les méthodes d'élagage en deux catégories : les méthodes dynamiques et les méthodes statiques.

Les méthodes dynamiques construisent les forêts aléatoires élaguées de manière itérative. A chaque itération, un nouvel arbre est entraîné et ajouté à la forêt s'il satisfait un certain critère [Kulkarni and Sinha, 2012, Bernard et al., 2012]. Ainsi, la forêt aléatoire initiale de taille l n'a pas besoin d'être créée et stockée au préalable. Même si les méthodes d'élagage dynamique atteignent l'objectif final de création de forêts aléatoires de tailles réduites, elles ne sont pas basées sur un sous échantillonnage de la forêt initiale. De ce fait, nous estimons qu'elles s'éloignent légèrement du sujet de ce chapitre, et nous n'aborderons donc pas plus ces méthodes dans la suite.

D'un autre côté, les méthodes d'élagage statique sont basées sur le paradigme de *surproduction et choix*. Elles commencent par créer une forêt aléatoire initiale de l arbres, puis construisent itérativement la forêt élaguée de taille $m \ll l$, en optimisant un certain critère. Ces méthodes sont dites *forward* si la forêt élaguée est construite en partant de l'ensemble vide, et en ajoutant un nouvel arbre à chaque itération. Elles sont dites *backward* si la forêt élaguée est construite en supprimant des arbres en partant de la forêt initiale.

Nous présentons dans ce chapitre une méthode d'élagage de forêts aléatoires *statique et forward*, basée sur l'algorithme Orthogonal Matching Pursuit (OMP). Notre méthode est *statique* car la forêt élaguée est construite en échantillonnant un sous-ensemble d'arbres de la forêt initiale. Elle est *forward* car nous utilisons OMP pour choisir, à chaque itération, un nouvel arbre à ajouter à la forêt élaguée.

Avant de présenter notre méthode, nous faisons un état de l'art des méthodes d'élagage statique dans la section suivante.

6.1.2 État de l'art

Nous présentons dans cette partie quelques méthodes d'élagage existantes. Pour un rappel de l'algorithme d'OMP ainsi qu'un état de l'art de ce dernier, nous invitons le lecteur à voir les sections 2.2 et 5.1.

Comme décrit ci-dessus, les méthodes statiques sont des algorithmes itératifs qui se basent sur certains critères pour choisir l'arbre à ajouter ou à supprimer de la forêt élaguée. La différence entre toutes ces méthodes réside dans le critère de sélection ou de suppression des arbres.

Les auteurs de [Yang et al., 2012a] et [Zhang and Wang, 2009] proposent des méthodes *backward*. Dans [Yang et al., 2012a] le critère de suppression d'arbres est basé sur la confiance de la forêt dans sa prédiction, tandis que dans [Zhang and Wang, 2009] le critère est basé sur son erreur empirique.

Les auteurs de [Caruana et al., 2004], proposent une version *forward* de la méthode de [Zhang and Wang, 2009], en sélectionnant à chaque itération l'arbre de décision qui permet la plus grande réduction de l'erreur empirique de la forêt élaguée.

Le critère proposé par les auteurs de [Zhang and Wang, 2009] est quant à lui basé sur la diversité de la forêt. Ainsi, à chaque itération, l'arbre qui maximise la corrélation moyenne entre lui et les autres arbres de la forêt est retiré. Cette méthode supprime donc les arbres qui semblent redondants, et favorise ainsi la diversité dans la forêt élaguée.

Toutefois, on pourrait souhaiter avoir des critères qui combinent diversité et bonnes prédictions, pour avoir le meilleur des deux mondes. C'est l'approche proposée par [Hu et al., 2007] et [Fawagreh et al., 2016]. Les auteurs de [Hu et al., 2007] proposent une méthode en deux phases. Une première étape *forward* construit une forêt intermédiaire en sélectionnant un ensemble de plus de m arbres qui minimisent l'erreur de cette forêt. Une deuxième étape *backward* permet de supprimer les arbres redondants n'ayant pas d'impact sur les performances de la forêt intermédiaire. Dans [Fawagreh et al., 2016], la diversité est assurée grâce à une étape de clustering des arbres. Ainsi, les arbres redondants se retrouvent dans les mêmes clusters, et il suffit ensuite de sélectionner l'arbre avec les meilleures performances dans chaque cluster.

Nous proposons dans ce chapitre de suivre [Hu et al., 2007] et [Fawagreh et al., 2016] en proposant une méthode permettant de construire une forêt élaguée avec de bonnes performances de prédiction et constituée d'arbres décorrélés les uns des autres. Notre méthode d'élagage est basée sur l'algorithme d'Orthogonal Matching Pursuit (OMP) [Pati et al., 1993], et se résume en deux étapes : premièrement, un

dictionnaire est construit à partir des prédictions de chacun des arbres de la forêt initiale (nous expliquerons plus tard comment cette forêt a été apprise). L'algorithme d'OMP est ensuite utilisé afin de sélectionner un sous-ensemble de m arbres qui permettent de mieux approximer le vecteur des vraies étiquettes de l'ensemble de données considéré.

En plus de sélectionner m arbres, notre algorithme associe aussi un poids à chacun d'entre eux. Nous verrons en section 6.3 que ces poids ont un rôle très important sur les performances de généralisation de la forêt aléatoire. De plus, par construction de OMP, les arbres sélectionnés sont naturellement décorrélés les uns avec les autres.

La suite de ce chapitre est organisée comme suit. Nous présentons notre algorithme de OMP pour l'élagage de forêts aléatoires en section 6.2. Nous évaluerons ensuite empiriquement ses performances dans la section 6.3.

6.2 Forêts OMP

6.2.1 Notation

Soit $\mathcal{X} := \{x_1, \dots, x_n\}$ un ensemble de n données de $\mathbb{X}$, et $y \in \mathbb{Y}^n$ les étiquettes associées.

Un arbre de décision $t \in \mathcal{T} := \{t \mid t : \mathbb{X} \to \mathbb{Y}\}$ est une fonction qui associe à toute donnée $x \in \mathbb{X}$ une étiquette $t(x)$:

$$\begin{array}{rccc} t(\cdot) & : & \mathbb{X} & \to & \mathbb{Y} \\ & & x & \mapsto & t(x). \end{array}$$

Une forêt aléatoire est une collection de l arbres : $\mathbf{t_l} := \{t_1(\cdot), \dots, t_l(\cdot)\}$. L'étiquette prédite d'une donnée est calculée en combinant les prédictions de chacun de ces arbres. Dans le cas d'un problème de classification binaire (i.e. $\mathbb{Y} = \{-1, +1\}$), un vote de majorité entre les prédictions des arbres est appliqué : signe $\left(\sum_{i=1}^{l} t_i(x)\right)$, alors que dans le cas d'un problème de régression (i.e. $\mathbb{Y} = \mathbb{R}$) c'est une moyenne de ces prédictions qui est calculée : $\frac{1}{l}\sum_{i=1}^{l} t_i(x)$. Il est aussi possible d'associer un poids à chacun des arbres de la forêt afin de lui donner plus ou moins d'importance. Ainsi, une forêt aléatoire est aussi une fonction que l'on peut noter comme :

$$\begin{array}{rccc} f_l(\cdot, \mathbf{t_l}, \alpha) & : & \mathbb{X} & \to & \mathbb{Y} \\ & & (x, \mathbf{t_l}, \alpha) & \mapsto & g\left(\sum_{i=1}^{l} \alpha[i] t_i(x)\right), \end{array}$$

où $g : \mathbb{R} \to \mathbb{Y}$ est égale à la fonction identité dans le cas de la régression, et à la fonction signe dans le cas de la classification binaire.

Dans ce chapitre nous ferons un abus des notations, et représenterons un arbre et une forêt par leurs prédictions sur l'ensemble $\mathcal{X}$. Ainsi, un arbre t sera représenté par le vecteur :

$$t(\mathcal{X}) := (t(x_1), \ldots, t(x_n)) \in \mathcal{Y}^n.$$

tandis qu'une forêt sera représentée par le vecteur $T_{\mathcal{X}}\alpha$, où

$$T_{\mathcal{X}} := [t_1(\mathcal{X}), \ldots, t_l(\mathcal{X})] \in \mathcal{Y}^{n \times l},$$

est la matrice dont les colonnes sont les prédictions des l arbres sur l'ensemble des données $\mathcal{X}$.

Nous attirons l'attention du lecteur sur le fait que la matrice $T_{\mathcal{X}}$ dépend aussi de $\mathbf{t_l}$. Toutefois, nous avons fait le choix de ne pas le faire apparaître dans notre notation afin d'alléger cette dernière.

6.2.2 OMP pour l'élagage de forêt

Étant donné un ensemble de l arbres, le problème d'élagage de cette forêt peut être exprimé comme : trouver un sous-ensemble de m arbres pondérés tel que les prédictions de cette forêt de m arbres soient « bonnes ». La qualité des prédictions de cette forêt élaguée peut être mesurée en les comparant aux prédictions de la forêt initiale, ou même aux vraies étiquettes.

En utilisant les notations introduites plus haut, nous pouvons écrire le problème d'élagage d'une forêt comme :

$$\alpha^* = \underset{\alpha \in \mathbb{R}^l}{\arg\min} \frac{1}{2} \|T_{\mathcal{X}}\alpha - y\|_2^2 \text{ t.q. } \|\alpha\|_0 \leq m. \tag{6.1}$$

Minimiser la quantité $\|T_{\mathcal{X}}\alpha - y\|_2^2$ permet de construire un vecteur de poids α tel que la forêt $T_{\mathcal{X}}\alpha$ ait la plus petite erreur de prédiction sur l'ensemble $\mathcal{X}$. De plus, la contrainte sur la quasi-norme l_0 permet de choisir au plus m arbres, puisque $\|\alpha\|_0$ compte le nombre de valeur non nulles dans α. Ainsi, la solution du Problème (6.1) permet de construire une forêt élaguée $T_{\mathcal{X}}\alpha^*$ d'au plus m arbres qui fait le moins d'erreur sur les prédictions de l'ensemble $\mathcal{X}$.

Comme vu dans le chapitre 5, le Problème (6.1) est NP-dur [Davis et al., 1997]. Malgré cela, il a suscité beaucoup d'intérêt en traitement du signal, et plusieurs algorithmes permettant de construire des approximations de α^* ont été proposés. Parmi eux, les algorithmes de

Matching Pursuit (MP) [Davis et al., 1997] et Orthogonal Matching Pursuit (OMP) [Pati et al., 1993].

Nous proposons dans ce chapitre d'utiliser OMP [Pati et al., 1993] pour construire une approximation de la solution du Problème (6.1). Nous verrons qu'en plus de construire des forêts avec une faible erreur de prédiction, notre méthode hérite de la propriété d'orthogonalité d'OMP, et sélectionne donc naturellement des arbres décorrélés.

Comme dans la section 2.2, nous utiliserons le terme *dictionnaire* pour désigner l'ensemble des colonnes de $T_{\mathcal{X}}$:

$$\mathcal{D} := \{t_i(\mathcal{X})\}_{i=1}^{l},$$

et le terme *atome* pour désigner les éléments du dictionnaire. Nous supposerons aussi que les atomes sont normalisés i.e. $\|t_i(\mathcal{X})\|_2 = 1$. Remarquons que cette hypothèse n'est pas très restrictive, puisqu'il suffit de diviser chacun des vecteurs $t_i(\mathcal{X})$ par sa norme.

OMP est un algorithme itératif, qui construit à chaque itération k une forêt élaguée approximée $T_{\mathcal{X}}\alpha_k$. Notons par $r_k = y - T_{\mathcal{X}}\alpha_k$ le résidu associé, c'est-à-dire la partie des étiquettes qui reste à approximer. À chaque itération k, OMP sélectionne l'arbre le plus corrélé avec le résidu, et l'ajoute à la forêt approximée. Une mise-à-jour de la forêt élaguée est alors construite en cherchant les poids à associer à chaque arbre sélectionné qui permettent la plus petite erreur de prédiction. Nous résumons ces étapes dans l'Algorithme 6.

Remarquons que par construction de OMP, au plus un seul nouvel arbre est ajouté à chaque itération de l'Algorithme 6. Ainsi, à chaque itération k, la forêt élaguée $T_{\mathcal{X}}\alpha_k$ contient au plus k arbres pondérés par le vecteur α_k.

Pour résumer, notre méthode d'élagage de forêt aléatoire présentée dans l'Algorithme 6 répond à deux attentes : taille réduite, et diversité des arbres sélectionnés.

Toutefois, il reste une question primordiale à laquelle nous n'avons pas répondu : qu'en est-il des performances de prédiction de la forêt élaguée ?

Cette question n'est pas si simple, et nous n'allons pas y répondre intégralement dans ce manuscrit. Toutefois, nous présentons dans ce qui suit des pistes qui permettront dans de futurs travaux d'y répondre. Nous utiliserons dans la Remarque 4 ci-dessous les garanties théoriques de OMP afin d'étudier les performances de prédiction de la forêt élaguée sur l'ensemble d'entraînement $\mathcal{X}$. Nous utiliserons ensuite en section 6.2.3 un lien entre OMP et Gradient Boosting afin d'étudier ses performances de prédiction en généralisation.

Algorithme 6 : OMP pour l'élagage de forêts aléatoires

Entrée : ensemble de données $\mathcal{X}$, étiquettes y, dictionnaire $\mathcal{D} := \{t_i(\mathcal{X})\}_{i=1}^{l}$, $m \geq 1$;
Sortie : α_k une approximation de la solution du Problème (6.1);
$k = 0$;
$\alpha_k = 0$;
$r_k = y$;
$\lambda_0 = \emptyset$;
tant que *critère d'arrêt non vérifié* **faire**
  $i_k = \arg\max_{i \in [l]} |\langle t_i(\mathcal{X}), r_k \rangle|$;
  $\lambda_{k+1} = \lambda_k \cup \{i_k\}$;
  $\alpha_{k+1} = \arg\min_{\alpha \in \mathcal{C}_{\lambda_{k+1}}} \|y - T_{\mathcal{X}}\alpha\|_2$ où $\mathcal{C}_{\lambda_{k+1}} := \{\alpha \in \mathbb{R}^l \mid \forall i \notin \lambda_{k+1} : \alpha[i] = 0\}$;
  $r_{k+1} = y - T_{\mathcal{X}}\alpha_{k+1}$;
  $k = k + 1$;
fin
Renvoyer α_k ;

Remarque 4. *Nous avons vu en chapitre 5 que sous certaines conditions sur le dictionnaire $\mathcal{D} := \{t_i(\mathcal{X})\}_{i=1}^{l}$ et sur y, il est possible d'avoir des garanties théoriques sur la qualité de l'approximation du Problème (6.1). La condition sur le dictionnaire implique sa cohérence $\mu(\mathcal{D})$ qui est définie comme :*

$$\mu(\mathcal{D}) := \max_{i \neq j} |\langle t_i(\mathcal{X}), t_j(\mathcal{X}) \rangle|.$$

Rappelons que la cohérence mesure à quel point un dictionnaire se rapproche d'une base orthonormée. Lorsque $\mathcal{D}$ est presque une base orthonormée, alors pour tout i, j : $\langle t_i(\mathcal{X}), t_j(\mathcal{X}) \rangle \approx 0$, et donc $\mu(\mathcal{D}) \approx 0$.

Nous pouvons maintenant énoncer l'un des résultats théoriques sur la qualité de l'approximation de OMP [Tropp, 2004] : si y est m-parcimonieux dans $\mathcal{D}$, i.e. s'il existe $\alpha^* \in \mathbb{R}^l$, tel que $\|\alpha\|_0 \leq m$ et $y = T_{\mathcal{X}}\alpha^*$, et si $m \leq \Theta\left(\mu(\mathcal{D})^{-1}\right)$, alors $T_{\mathcal{X}}\alpha_m = y$ où α_m est la solution construite par OMP au bout de m itérations.

Regardons ce résultat de plus prés. La condition y est m-parcimonieux dans $\mathcal{D}$ se traduit par : il existe un sous-ensemble de m arbres de la forêt initiale, tel qu'il est possible de les pondérer de manière à ce que leur prédictions sur $\mathcal{X}$ soient égales aux vraies étiquettes y. La condition $m \leq \Theta\left(\mu(\mathcal{D})^{-1}\right)$ implique que : si $\mathcal{D}$ se rapproche d'une base orthonormée (i.e. $\mu(\mathcal{D}) \approx 0$), alors OMP peut reconstruire toute forêt m-parcimonieuse pour une large plage de valeur de m.

Malheureusement, ces résultats ne portent que sur l'erreur de prédiction pour les données d'entraînement, et ne permettent pas d'en déduire des garanties sur l'erreur de généralisation.

Nous proposons dans ce qui suit une piste, qui nous permettrait dans le futur de proposer des garanties théoriques sur l'erreur de généralisation de la forêt élaguée par OMP.

Il est aussi possible d'avoir des garanties dans le cas où y n'est pas exactement m-parcimonieux dans $\mathcal{D}$ [Tropp, 2004]. En effet, si $y = T_{\mathcal{X}}\alpha^* + \epsilon$, et si $m \leq \Theta\left(\mu\left(\mathcal{D}\right)^{-1}\right)$ alors :

$$\|y - T_{\mathcal{X}}\alpha_k\|_2 \leq \sqrt{1+m}\,\|y - T_{\mathcal{X}}\alpha^*\|_2\,,$$

où α_k est l'approximation construite par l'Algorithme 6 à l'étape k, et α^* est la solution du Problème (6.1).

6.2.3 OMP et Gradient Boosting

Le Boosting [Schapire, 1999] est un ensemble de méthodes permettant de combiner plusieurs *classifieurs faibles* pour construire un classifieur final performant. Ces classifieurs faibles sont des modèles prédictifs qu'on suppose au moins aussi performants que le hasard.

Il existe plusieurs méthodes permettant de construire de telles combinaisons de classifieurs faibles, comme par exemple la méthode de Gradient Boosting (GB) [Natekin and Knoll, 2013]. GB est un algorithme itératif, qui effectue à chaque itération une descente de gradient sur un espace de fonctions préalablement défini.

Comme un arbre de décision peut être vu comme un classifieur faible, le lien entre forêts aléatoires et Boosting est très clair. De même, le lien entre GB et MP est connu depuis longtemps. En effet, comme nous le verrons ci dessous, les auteurs de [Friedman, 2001] démontrent que GB revient à appliquer MP sur un dictionnaire soigneusement choisi. Le lien entre GB et OMP est quant à lui moins clair, et doit encore être étudié. Toutefois, dans certains cas, les algorithmes MP et OMP sont équivalents (i.e. qu'ils construisent la même solution) [Tropp, 2004, Gribonval and Vandergheynst, 2006]. Ainsi sous ces mêmes conditions, appliquer OMP revient à appliquer la procédure de GB. Cette équivalence nous permet de définir des conditions sous lesquelles notre méthode d'élagage basée sur OMP hérite des nombreuses garanties théoriques sur les performances en généralisation de GB [Natekin and Knoll, 2013].

Nous présentons dans ce qui suit un bref rappel de GB afin de montrer son équivalence avec MP, et ensuite avec OMP.

Soit $\mathcal{T} := \{t \mid t : \mathbb{X} \to \mathbb{Y}\}$ un ensemble de classifieurs faibles, et soit $lin(\mathcal{T}) := \{f \in \mathcal{T} \mid \text{tel que } f = \sum_{t\in\mathcal{T}} \alpha_t t \ ; \ \alpha_t \in \mathbb{R}\}$ l'ensemble des combinaisons linéaires des éléments de $\mathcal{T}$. Enfin, soit $loss : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}$ une fonction de perte. Le but de GB est de résoudre le problème suivant :

$$f^* = \underset{f\in lin(\mathcal{T})}{\arg\min}\, \mathbb{E}_{(x,y)\sim P_X \times P_Y} \left[loss\left(y, f(x)\right)\right],$$

où P_X, P_Y sont deux distributions de probabilité. Ces deux distributions sont souvent inconnues, mais peuvent être observées à travers un échantillon de points $\mathcal{X} := \{x_1, \ldots, x_n\}$ et $y := \{y_1, \ldots, y_n\}$ où $x_i \sim P_X$ et $y_i \sim P_Y$. Ainsi, f^* peut-être approximée en résolvant le problème suivant :

$$f = \underset{f\in lin(\mathcal{T})}{\arg\min} \sum_{i=1}^{n} loss\left(y_i, f(x_i)\right).$$

Nous supposerons dans ce chapitre que la fonction de perte est l'erreur quadratique, ainsi :

$$f = \underset{f\in lin(\mathcal{T})}{\arg\min} \frac{1}{2} \left\|y - f(\mathcal{X})\right\|_2^2,$$

où par abus de notation $f(\mathcal{X}) := (f(x_1), \ldots, f(x_n))$ désigne le vecteur des prédictions de f sur l'ensemble $\mathcal{X}$.

A chaque itération k, l'approximation f_k est mise à jour comme :

$$f_{k+1}(\mathcal{X}) = f_k(\mathcal{X}) + \alpha_k t_k(\mathcal{X}),$$

où t_k est l'élément de $\mathcal{T}$ qui est le plus corrélé avec la direction de descente donnée par le gradient de la fonction de perte :

$$t_k := \underset{t\in\mathcal{T}}{\arg\max} \left| \left\langle t(\mathcal{X}), \frac{\partial \sum_{i=1}^{n} loss\left(y_i, f_k(x_i)\right)}{\partial f_k(\mathcal{X})} \right\rangle \right|.$$

et où

$$\alpha_k := \underset{\alpha\in\mathbb{R}}{\arg\min} \frac{1}{2} \left\|y - \left(f_k\left(\mathcal{X}\right) + \alpha t_k\left(\mathcal{X}\right)\right)\right\|_2^2$$

En développant les calculs des deux équations précédentes nous obtenons :

$$\begin{aligned} \alpha_k &:= \underset{\alpha\in\mathbb{R}}{\arg\min} \frac{1}{2} \left\|\left(y - f_k\left(\mathcal{X}\right)\right) + \alpha t_k\left(\mathcal{X}\right)\right\|_2^2 = \underset{\alpha\in\mathbb{R}}{\arg\min} \frac{1}{2} \left\|r_k - \alpha t_k\left(\mathcal{X}\right)\right\|_2^2 \\ &= \underset{\alpha\in\mathbb{R}}{\arg\min} \frac{1}{2} \left\|r_k\right\|_2^2 - \alpha\left\langle r_k, t_k\left(\mathcal{X}\right)\right\rangle + \frac{1}{2}\alpha^2 = \left\langle r_k, t_k\left(\mathcal{X}\right)\right\rangle \end{aligned} \tag{6.2}$$

où la troisième égalité est vraie car $\|t_k(\mathcal{X})\|_2 = 1$. Enfin, en remarquant que :

$$\frac{\partial \sum_{i=1}^{n} loss\left(y_i, f_k(x_i)\right)}{\partial f_k(\mathcal{X})} = \frac{\partial \frac{1}{2}\|y - f_k(\mathcal{X})\|_2^2}{\partial f_k(\mathcal{X})} = -y + f_k(\mathcal{X}) = -r_k.$$

nous obtenons enfin :

$$t_k := \arg\max_{t \in \mathcal{T}} |\langle t(\mathcal{X}), r_k \rangle|.$$

En résumé, GB répète à chaque itération k ces deux instructions :

— sélection d'atome :

$$t_k(.) := \arg\max_{t(.) \in \mathcal{T}} |\langle t(\mathcal{X}), r_k \rangle|. \tag{6.3}$$

— mise-à-jour de l'approximation :

$$f_{k+1}(\mathcal{X}) = f_k(\mathcal{X}) + \langle r_k, t_k(\mathcal{X}) \rangle t_k(\mathcal{X}). \tag{6.4}$$

L'étape de sélection des atomes de GB (Équations (6.3)), ainsi que l'étape de mise à jour de l'approximation (Équations (6.4)), sont exactement les mêmes que celles de MP (Algorithme 1 en section 2.2).

Cette équivalence entre GB et MP est très importante pour notre méthode d'élagage. En définissant l'espace fonctionnel $\mathcal{T}$ comme l'ensemble des arbres de la forêt aléatoire initiale, et la fonction d'erreur comme l'erreur quadratique moyenne, alors appliquer GB est équivalent à appliquer MP avec comme dictionnaire l'ensemble des arbres de la forêt initiale.

Dans le cas général, il n'y a pas d'équivalence entre MP et OMP qui nous permettrait d'avoir une équivalence entre OMP et GB. Toutefois, il existe un cas particulier dans lequel cette équivalence est vérifiée. En effet, comme discuté en section 5.1, lorsque l'Hypothèse 1 est vérifiée, alors les solutions construites par MP et OMP coïncident. Ainsi, lorsque l'Hypothèse 1 est vérifiée, il y a une équivalence entre la solution construite par notre méthode (Algorithme 6) est celle construite par GB.

Hypothese 1. *Soit $\mathcal{D} := \{t_i(\mathcal{X})\}_{i=1}^{l}$ un dictionnaire de l arbres de cohérence $\mu(\mathcal{D})$, et soit $T_{\mathcal{X}} := [t_1(\mathcal{X}), \ldots, t_l(\mathcal{X})] \in \mathbb{Y}^{n \times l}$, les prédictions des l arbres sur $\mathcal{X}$. Alors, il existe $\alpha \in \mathbb{R}^l$ tel que :*

— $\|\alpha\|_0 \leq m$

— $y = T_{\mathcal{X}}\alpha$

— $m \leq \Theta\left(\mu(\mathcal{D})^{-1}\right)$

Remarquons que l'Hypothèse 1 est la même que celle discutée dans la Remarque 4.

6.3 Expériences numériques

Nous présentons dans cette section quelques expériences afin d'évaluer notre méthode d'élagage de forêts aléatoires, présentée dans l'Algorithme 6. Nous commençons par décrire en section 6.3.1 le cadre expérimental ainsi que les jeux de données considérés, puis nous discuterons en section 6.3.2 des performances de notre méthode.

6.3.1 Cadre expérimental

Nos expériences ont été faites sur 11 jeux de données de régression et de classification binaire. Nous résumons les caractéristiques de ces jeux de données dans la Table 6.1. Nous indiquons aussi la taille de la forêt initiale de chaque jeu de données (nous expliquerons comment cette forêt a été construite et comment sa taille a été fixée dans le paragraphe suivant).

Nom	Acronyme	n	d	Tâche	Taille forê
California [Pace and Barry, 1997]	C.H.	20 640	8	Régression	1000
Kin8nm [Corke, 1996]	Kin.	8 192	8	Régression	1000
Diamonds [Dia,]	Diam.	53 940	9	Régression	429
Diabetes [Efron et al., 2004]	Diab.	442	10	Régression	108
Boston house [Belsley et al., 1980]	Bos.	506	13	Régression	100
Gamma [Bock et al., 2004]	Gam.	19 020	11	Classification	100
Breast Cancer [Mangasarian et al., 1995]	B.C.	569	30	Classification	1000
Steel Plates [Buscema, 1998]	St. P.	1941	33	Classification	1000
King-Rook [Shapiro, 1987]	KR-KP	3 196	36	Classification	1000
Spambase [Cranor and LaMacchia, 1998]	Sp. B.	4 601	57	Classification	1000
LFW pairs [Huang et al., 2008]	LFW	13 233	5 828	Classification	1000

TABLE 6.1 – Description des jeux de données, et de la taille de la forêt initiale pour chaque ensemble de données. Ici n est le nombre de données, d la dimension de chacune des données, et « tâche » est la tâche d'apprentissage (i.e. régression ou classification binaire) .

Toutes nos expériences ont été faites en utilisant la bibliothèque scikit-learn [Buitinck et al., 2013] pour les algorithmes OMP et forêt aléatoire. Nous avons utilisé un découpage des données de 60-20-20% pour les ensembles d'entraînement, de test et de validation. Les forêts initiales sont

entraînées sur l'ensemble de données d'entraînement. Les meilleurs paramètres de chacune (comme par exemple leur taille indiquées dans la Table 6.1) ont été définis en utilisant le jeu de données de validation. L'ensemble de données d'entraînement et de validation sont ensuite fusionnés pour l'élagage de la forêt et l'apprentissage des poids de chaque arbre. Cela peut sembler contre-intuitif de ne pas utiliser uniquement l'ensemble de validation, mais nos expériences ont démontré que fusionner ces deux ensembles permet d'avoir les meilleures performances de généralisation, et cela pour toutes les méthodes d'élagage que nous considérons ici.

Une fois la forêt élaguée construite, ses performances en généralisation sont mesurées sur l'ensemble test en calculant la MSE (*Mean Square Error*) pour les problèmes de régression, et en calculant le pourcentage normalisé de précision (i.e. pourcentage de bonne classification divisé par 100) dans le cas de classification binaire. Toutes nos expériences ont été répétées 10 fois, et nous affichons la moyenne et la variance des performances pour chaque méthode d'élagage considérée.

Nous comparons notre méthode à plusieurs autres méthodes de l'état de l'art pour l'élagage de forêt aléatoire : la méthode *Ensemble* [Caruana et al., 2004], *Zhang Predictions* [Wang and Zhang, 2009], *Zhang Similarity* [Wang and Zhang, 2009] et enfin *Kmeans* [Fawagreh et al., 2016]. Nous comparons aussi notre méthode à un échantillonnage uniforme, qui consiste à choisir les m arbres de manière uniforme (nous noterons cette méthode par Random).

6.3.2 Résultats

Nous décrivons dans cette section nos résultats expérimentaux. Dans un premier temps, nous proposons d'utiliser la forêt élaguée construite par OMP comme décrit dans l'Algorithme 6. Nous appellerons cette méthode *OMP*. Nous considérons aussi la forêt élaguée qui contient les arbres sélectionnés par OMP, mais sans leurs poids associés (i.e. que tous les arbres ont un poids égale à 1). Nous appellerons cette méthode *OMP w/o weights* (pour *OMP without weights*).

Nous verrons que la méthode *OMP* permet d'excellents résultats dans certains cas, mais souffre de sur-apprentissage dans d'autres. Ces résultats nous permettront de mettre en lumière un lien entre les poids négatifs de OMP et les mauvaises performances de la forêt élaguée. Cela a motivé l'introduction d'une contrainte de non négativité sur les poids de OMP, et d'introduire deux nouvelles méthodes : *NN-OMP* et *NN-OMP w/o weights*. Nous verrons dans un second temps, qu'en plus de permettre de meilleures

performances de généralisation par rapport à la version standard de OMP, cette heuristique fournit un critère d'arrêt précoce sur la taille de la forêt élaguée.

OMP pour l'élagage

Nous commençons par comparer les performances de généralisation sur l'ensemble test de notre méthode avec les méthodes de l'état de l'art. Comme nous considérons plusieurs jeux de données, pour des soucis de place nous proposons de les séparer en trois figures : Figure 6.1, Figure 6.2 et Figure 6.3.

Nous remarquons dans la Figure 6.1 que sur les jeux de données *Kin8nm*, *California*, *KR-VS-KP* et *Steel Plates* la méthode d'élagage avec OMP a de meilleures performances en généralisation que les autres méthodes d'élagage. Notre méthode atteint même sur certains points de meilleures performances que la forêt initiale. Sur ces jeux de données, nous remarquons aussi l'intérêt des poids appris par OMP puisque la méthode *OMP* a de meilleures performances de généralisation que sa version non pondérée (méthode *OMP w/o weights*). La méthode *Ensemble* montre aussi d'excellents résultats sur la régression, comme sur le jeu de données *California*. Nous remarquons que cette méthode est autorisée à prendre plusieurs fois le même arbre dans la forêt, ce qui est équivalent à lui donner un poids plus important que les autres. Cela souligne à nouveau l'intérêt d'utiliser des poids pour les arbres dans la forêt élaguée. Toutefois, la méthode *Ensemble* a de très mauvaises performances sur des problèmes de classification binaire par rapport à *OMP*. Bien que les méthodes *Zhang* et *Kmeans* soient stables dans l'ensemble, elles ont même du mal à être aussi performantes que la sélection aléatoire des arbres (méthode *Random*). Elles sont aussi moins performantes que la méthode d'élagage *OMP*.

Sur les jeux de données *Diamonds*, *LFW pairs*, *Gamma*, *Spambase* de la Figure 6.2, la version non pondérée de notre méthode donne toujours des résultats au moins aussi bons que les méthodes de l'état de l'art.

Toutefois sur les jeux de données *Diabetes*, *Boston* et *Breast Cancer* de la Figure 6.3, les poids semblent avoir un effet néfaste sur les performances. Nous remarquons d'abord que dans ces cas, aucune méthode d'élagage n'arrive à battre la sélection uniforme des arbres. Nous remarquons aussi que ces trois jeux de données sont vraiment petits ce qui les rend très vulnérables au sur-apprentissage des poids. Pour vérifier cette théorie, nous affichons en Figure 6.4, les performances sur le jeu de données de validation qui a servi lors de l'apprentissage des poids. Nous remarquons sur cette figure que les performances de notre méthode sont très bonnes. Ainsi, nous

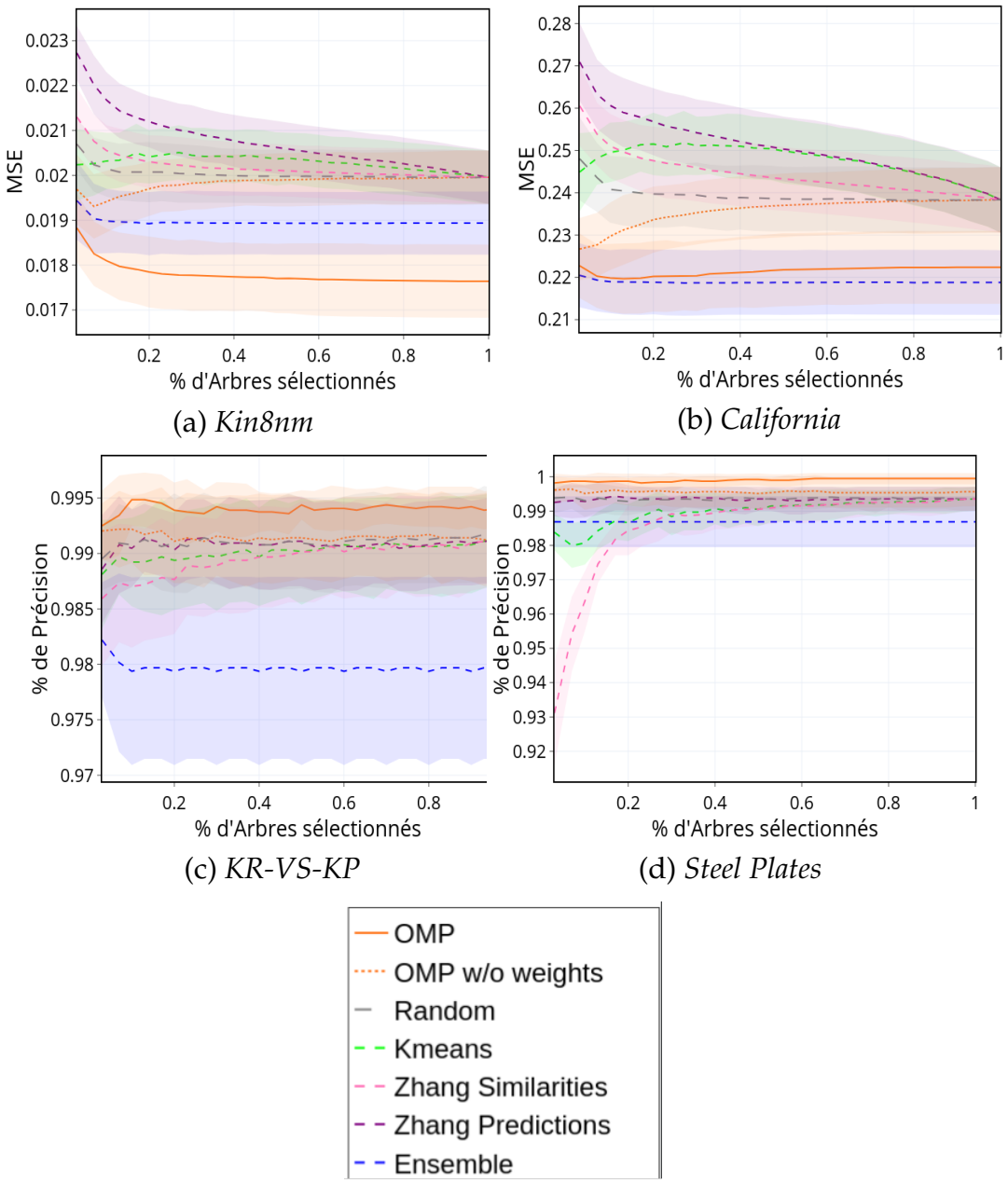
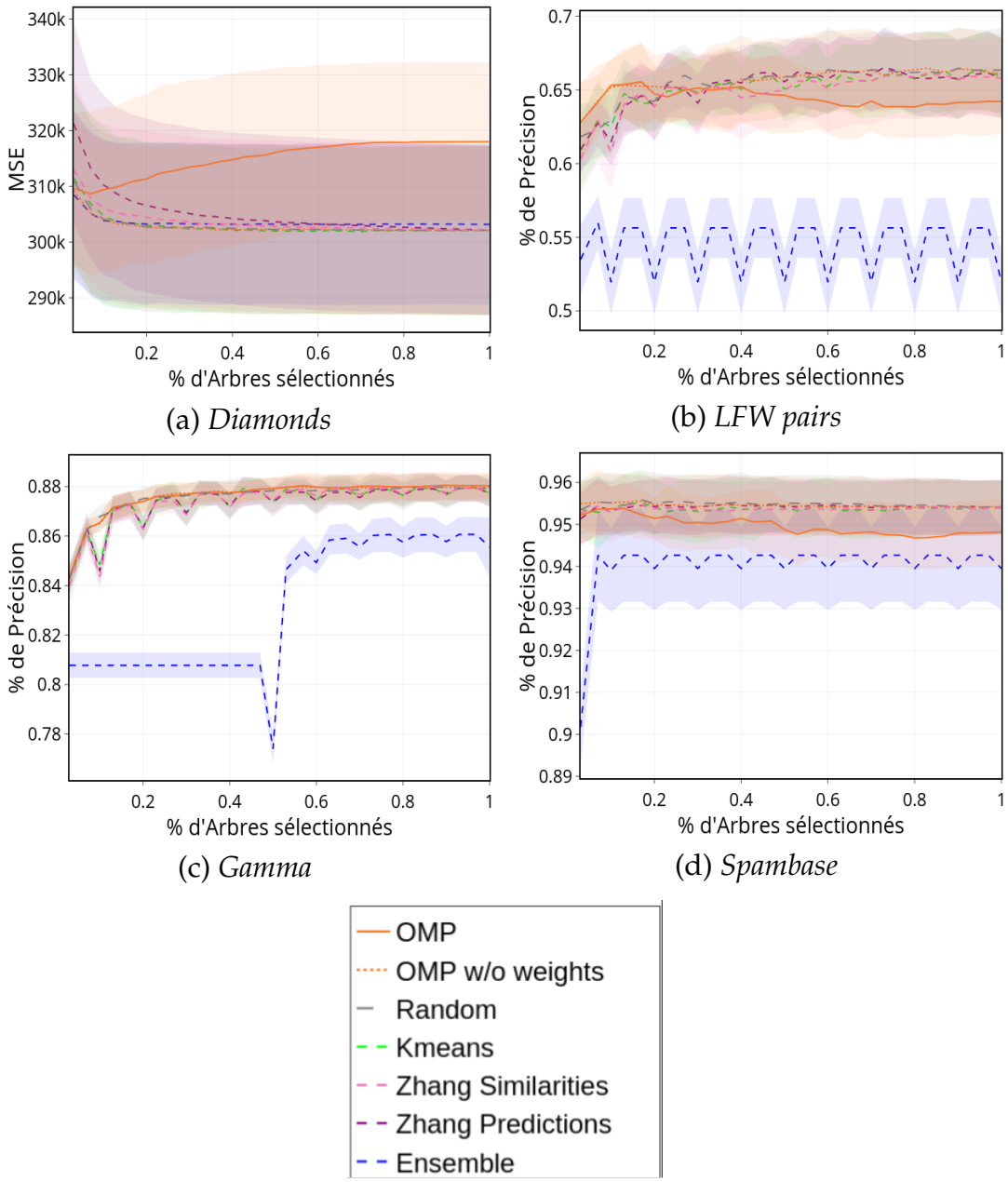


FIGURE 6.1 – Performance sur les données test en fonction de la taille de la forêt élaguée sur les jeux de données : *Kin8nm*, *California*, *KR-VS-KP* et *Steel Plates*.

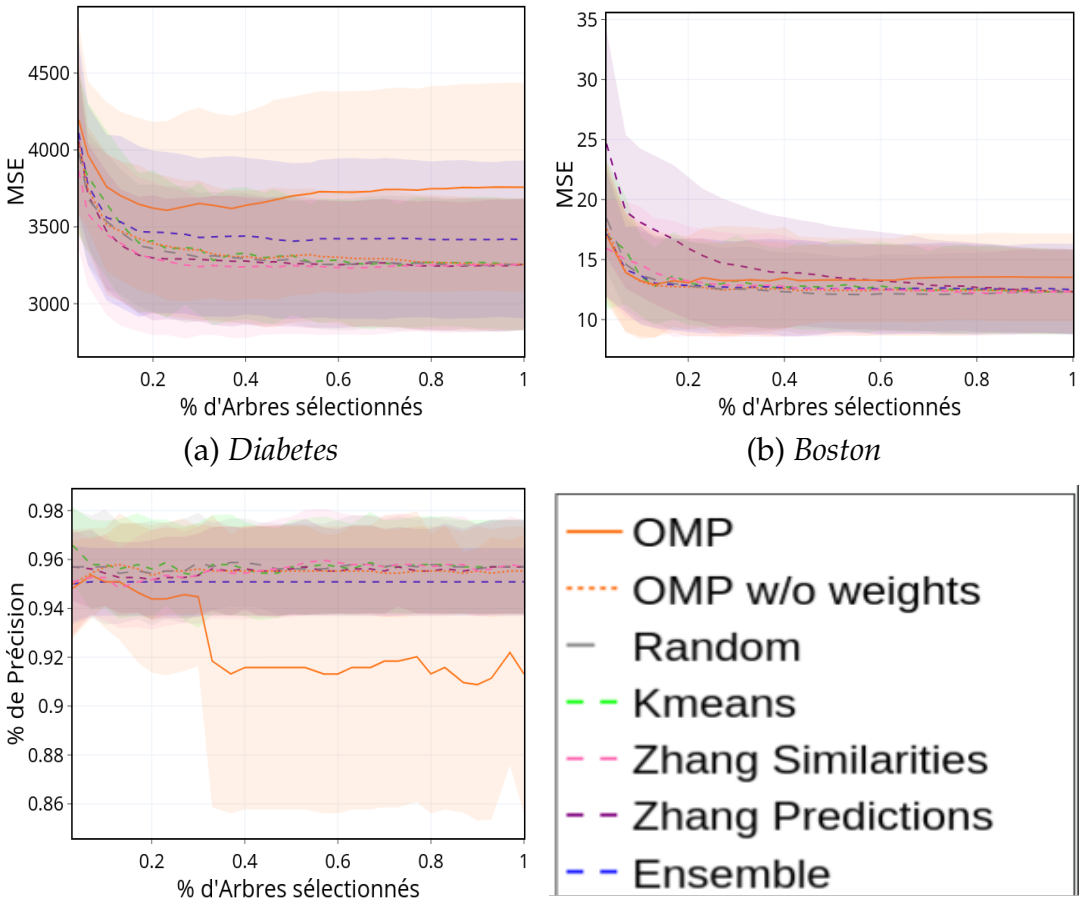


(a) *Diamonds* (b) *LFW pairs*

(c) *Gamma* (d) *Spambase*

(e) Légende

FIGURE 6.2 – Performance sur les données test en fonction de la taille de la forêt élaguée sur les jeux de données : *Diamonds*, *LFW pairs*, *Gamma*, *Spambase*.



(a) *Diabetes* (b) *Boston*

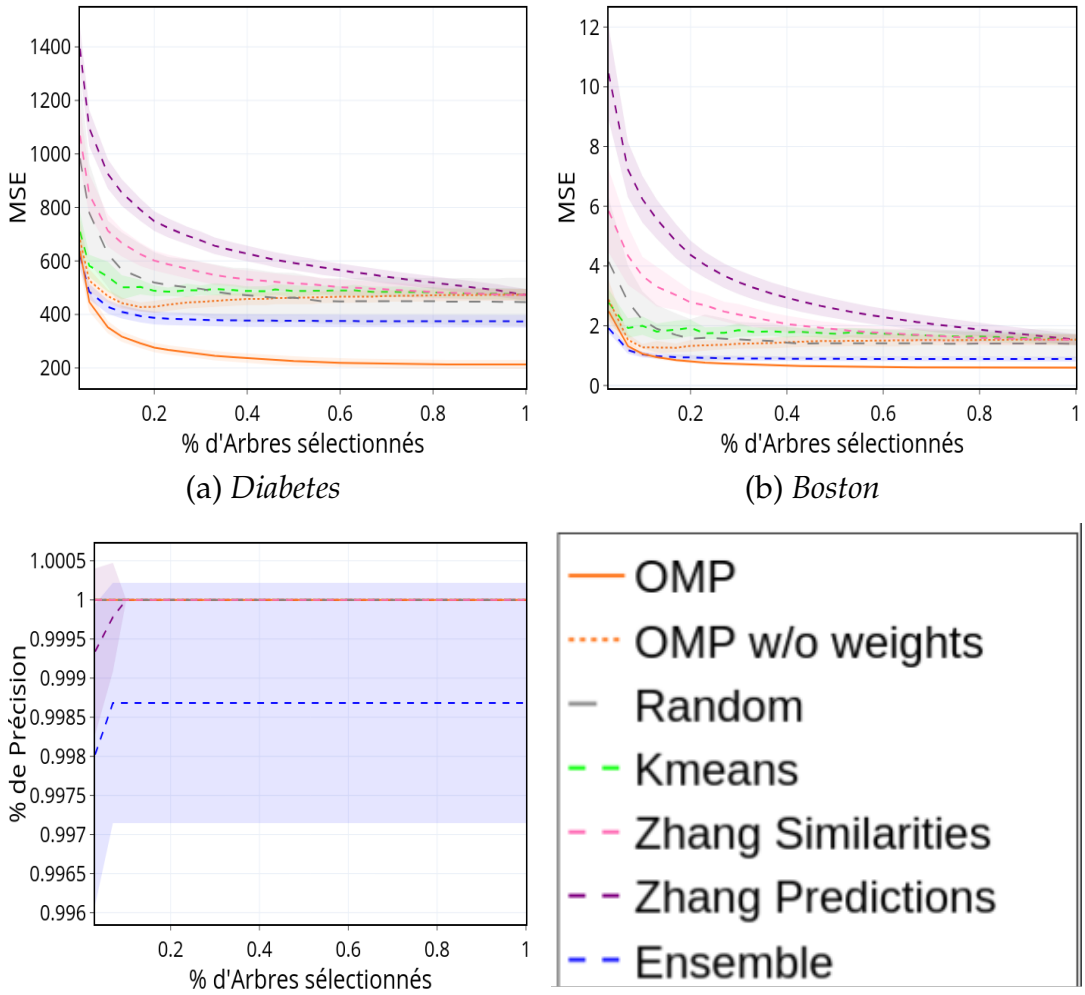
(c) *Breast Cancer* (d) Légende

FIGURE 6.3 – Performance sur les données test en fonction de la taille de la forêt élaguée sur les jeux de données : *Diabetes*, *Boston* et *Breast Cancer*.

pouvons affirmer que dans le cas des jeux de données *Boston*, *Diabetes* et *Breast Cancer*, il y a eu un sur-apprentissage des poids puisque la méthode OMP a de très bonnes performances sur l'ensemble de données qui a servi à l'apprentissage des poids, mais de très mauvaises performances sur les données de test.

OMP avec contrainte de non négativité

Bien que la technique d'élagage basée sur l'OMP ait souvent d'excellentes performances lors de la sélection d'un petit nombre d'arbres, ces performances se dégradent lorsque le nombre d'arbres sélectionnés augmente. Ceci est extrêmement problématique car aucune indication particulière



(a) *Diabetes* (b) *Boston*

(c) *Breast Cancer* (d) Légende

FIGURE 6.4 – Performance sur les données de validation en fonction de la taille de la forêt élaguée sur les jeux de données : *Diabetes*, *Boston* et *Breast Cancer*.

n'est donnée pendant la phase de sélection des arbres qui permettrait de concevoir un critère d'arrêt précoce pour fixer la taille maximale de la forêt élaguée. En effet, nous avons constaté en Figure 6.4, que les performances sur les données de validation de la méthode OMP ne semblent pas atteindre un plateau.

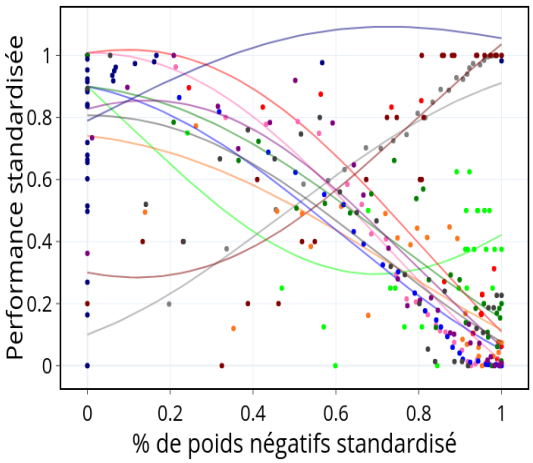
Encouragé par les performances assez bonnes de la version non pondérée de OMP, c'est tout naturellement que nous nous sommes intéressés aux poids associés aux arbres sélectionnés par OMP, afin d'expliquer les

mauvais comportements de notre méthode. Nous avons remarqué que la proportion d'arbres avec un poids négatif augmente avec le nombre d'arbres sélectionnés, comme illustré dans la Figure 6.5b. Dans cette figure, les performances mesurées sont l'inverse de la MSE dans le cas de la régression. Ainsi, une grande valeur signifie de bonnes performances. Remarquons aussi que nous n'affichons pas les résultats sur le jeu de données *Gamma*, car il ne contient aucun poids négatif.

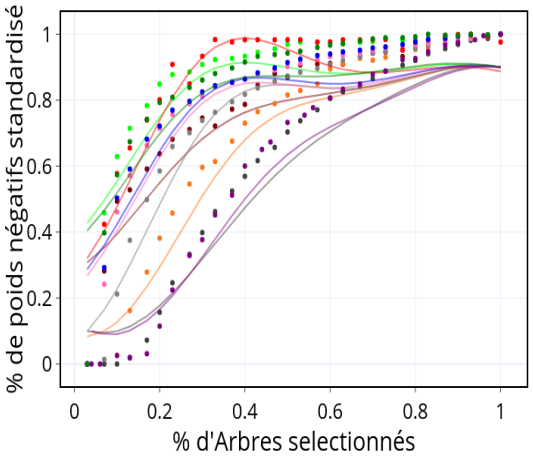
Le comportement mis en évidence par la Figure 6.5b a certainement un lien avec les mauvaises performances de la forêt élaguée avec OMP. Nous expliquons ce lien dans le cadre de la classification binaire, mais il peut être facilement transférable au cas de la régression : il est tout à fait raisonnable de penser que chacun des arbres de la forêt initiale devrait avoir des performances au moins aussi bonnes que celle d'un classifieur qui répond aléatoirement. Cela veut dire que la précision de chacun des arbres devrait être au moins supérieure à 0.5 (i.e. 50% d'erreur). Cependant, affecter un poids négatif a un arbre, revient a inverser sa prédiction dans le cas de la classification binaire, et de l'utiliser ensuite dans la prédiction de la forêt élaguée. Ce comportement qui semble contre intuitif peut s'expliquer par le fait que OMP affecte des poids négatifs à certains arbres pour parfaitement reconstruire le vecteur des étiquettes des données d'entraînement. Cette analyse est validée par la Figure 6.5a, où nous pouvons clairement voir que les performances en généralisation ont tendances à décroître lorsque la proportion de poids négatifs augmente. On remarque toutefois que cette tendance n'est pas vérifiée pour les jeux de données *Kin8nm*, *Diamonds* et *Steel Plates*. Notez que seules les forêts de taille supérieure à 10% de la taille de la forêt initiale sont considérées. Nous avons décidé de considérer uniquement ces forêts là, puisque celles avec des tailles plus petites ont naturellement de mauvaises performances.

Cette analyse nous a conduit à la conclusion suivante : l'utilisation d'une version d'OMP avec contrainte de non négativité (NN-OMP) [Bruckstein et al., 2008] permettrait d'éviter le sur-apprentissage et donc de donner de meilleures performances. La méthode d'élagage avec le NN-OMP n'est pas très différente de l'Algorithme 6. Seules deux lignes changent : premièrement, uniquement les arbres positivement corrélés avec le résidu sont sélectionnés, et deuxièmement, seules les poids positifs sont admis pour la construction de la forêt élaguée. Nous résumons ces étapes dans l'Algorithme 7

Les résultats de la méthode *NN-OMP* ainsi que sa version non pondérée *NN-OMP w/o weights* sont tracés dans la Figure 6.6. Notons que la Figure 6.6 reprend les courbes des Figures 6.1, 6.2 et 6.3 auxquelles s'ajoutent les courbent correspondantes aux méthodes *NN-OMP* et *NN-*



(a) Effet des poids négatifs sur les performances.



(b) Proportion de poids négatifs par rapport au nombre d'arbre.

(c) Légende jeux de données.

FIGURE 6.5 – Relation entre la taille de la forêt élaguée, performances de cette forêt, et le nombre de poids négatifs affectés par OMP. Dans la Figure 6.5a seules les forêts élaguées d'une taille supérieure à 10% de la taille de la forêt initiale sont représentées. Les performances mesurées sont l'inverse de la MSE dans le cas de la régression. Les valeurs de performances et le pourcentage de poids négatifs sont normalisés afin d'être comprises entre 0 et 1 et simplifier la comparaison entre les différents jeux de données. Les courbes ont été ajoutées afin de faciliter la lecture et ont été créée en entraînant un SVM avec noyau RBF à partir des observations originales (représentées ici par les points).

OMP w/o weights. Nous remarquons clairement qu'en plus de permettre de très bonnes performances de généralisation, la méthode *NN-OMP* permet

de naturellement définir la taille optimale de la forêt aléatoire élaguée. En effet, une fois qu'il n'est plus possible pour *NN-OMP* d'améliorer son approximation du vecteur des étiquettes sans utiliser de poids négatifs, l'algorithme s'arrête. Ce point d'arrêt est marqué par une étoile rouge dans la Figure 6.6.

Algorithme 7 : NN-OMP pour l'élagage de forêts aléatoires

Entrée : ensemble de données $\mathcal{X}$, étiquettes y, ensemble de l arbres $\{t_i(\mathcal{X})\}_{i=1}^{l}$, $m \geq 1$;
Sortie : α_k une approximation de la solution du Problème 6.1;
$k = 0$;
$\alpha_k = 0$;
$r_k = y$;
$\lambda_0 = \emptyset$;
tant que *critère d'arrêt non vérifié* **faire**
  $i_k = \arg\max_{i \in [l]} \langle t_i(\mathcal{X}), r_k \rangle$;
  si $\langle t_{i_k}(\mathcal{X}), r_k \rangle < 0$ **alors**
    Renvoyer α_k ;
  fin
  sinon
    $\lambda_{k+1} = \lambda_k \cup \{i_k\}$;
    $\alpha_{k+1} = \arg\min_{\alpha \in \mathcal{C}^+_{\lambda_{k+1}}} \|y - T_{\mathcal{X}}\alpha\|_2$, où
    $\mathcal{C}^+_{\lambda_{k+1}} := \{\alpha \in (\mathbb{R}^+)^l \mid \forall i \notin \lambda_{k+1} : \alpha[i] = 0\}$;
    $r_{k+1} = y - T_{\mathcal{X}}\alpha_k$;
    $k = k + 1$;
  fin
fin
Renvoyer α_{k+1} ;

Nous concluons de ces expériences que notre méthode d'élagage de forêts aléatoires basée sur *OMP* et *NN-OMP* permet de créer des forêts avec peu d'arbres, tout en gardant des performances de généralisation « assez proches » de celles de la forêt initiale.

6.4 Conclusion

Nous avons présenté dans ce chapitre une méthode originale pour la réduction de la taille de forêts aléatoires. Notre méthode utilise l'algorithme

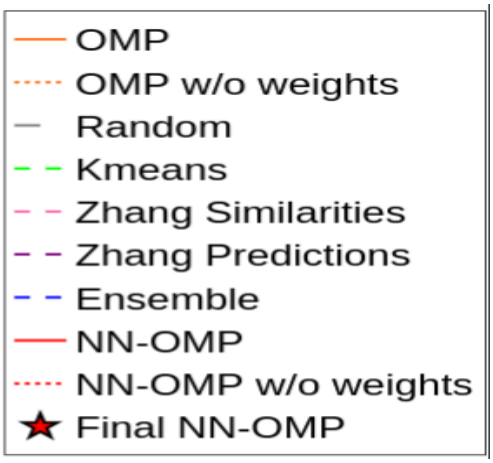
glouton Orthogonal Matching Pursuit (OMP) afin de construire une forêt élaguée dont les prédictions « ressemblent » le plus possible aux vraies étiquettes.

En plus de sélectionner un sous-ensemble d'arbres de la forêt initiale, notre méthode associe à chacun un coefficient. Les prédictions de la forêt élaguée sont alors construites comme une combinaison linéaire des prédictions de ces arbres. Les coefficients sont choisis de sorte que les prédictions de la forêt élaguée approximent le mieux les vraies étiquettes.

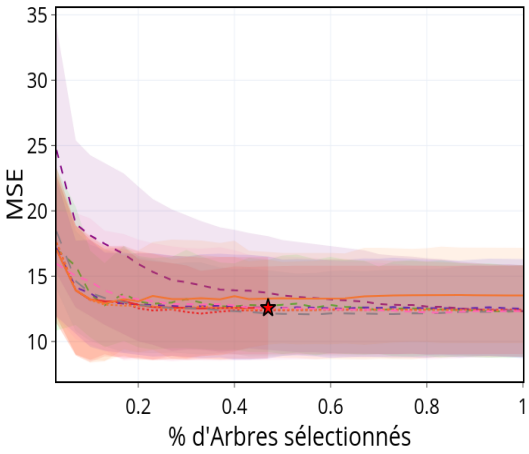
Nous avons expérimentalement vu que l'utilisation de OMP pour l'élagage de forêts aléatoires donne de très bons résultats dans certains cas. Toutefois, elle peut également conduire à un sur-apprentissage dans certains scénarios, en particulier, sur des jeux de données de petite taille. Pour pallier ce problème, nous avons proposé l'utilisation d'une contrainte de non-négativité sur les coefficients construits par OMP. Nous avons expérimentalement observé que cette méthode permettait de construire des forêts élaguées avec d'excellents résultats sur l'ensemble de données de test par rapport à l'état de l'art.

Enfin, nous avons exploité un lien très fin entre OMP et Gradient Boosting pour donner une piste qui permettrait, dans des travaux futurs, de théoriquement justifier notre méthode.

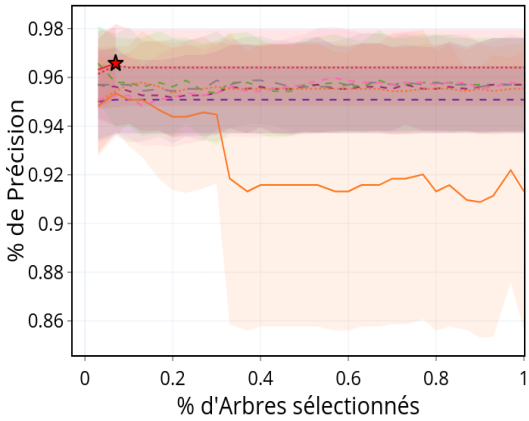
Ce travail ouvre la porte à plusieurs directions de recherche intéressantes. Sur l'aspect théorique, il serait intéressant d'exploiter le lien entre OMP et Gradient Boosting afin de dériver des garanties sur les performances en généralisation de la forêt élaguée avec notre méthode. Il serait utile d'étendre notre méthode d'élagage avec OMP aux problèmes de classification multi-classes. Enfin, il serait aussi intéressant de quantifier le gain en interprétabilité de la forêt élaguée construite par OMP et sa version non négative.



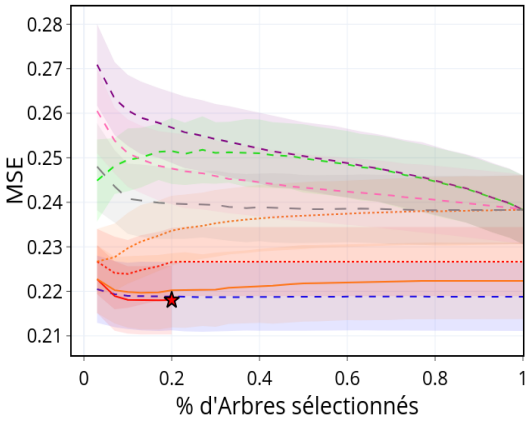
(a) Légende



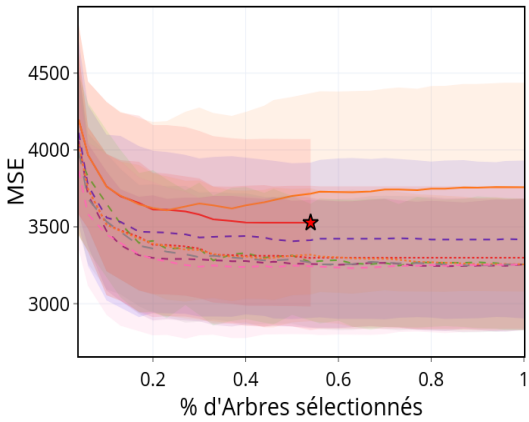
(b) *Boston*



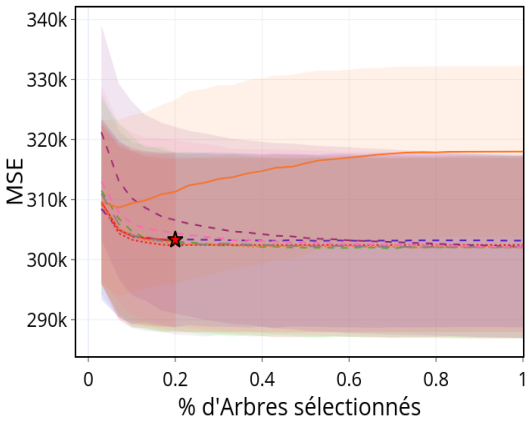
(c) *Breast Cancer*



(d) *California*

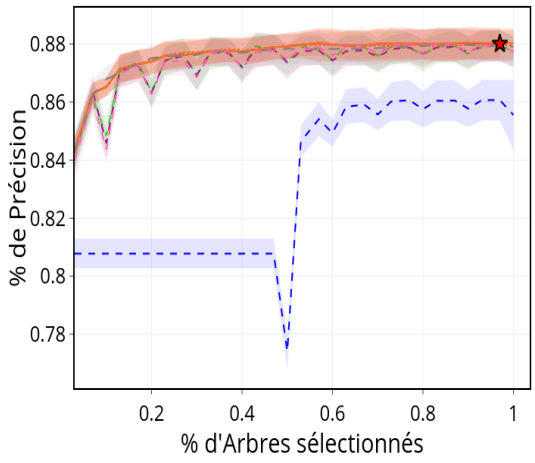


(e) *Diabetes*

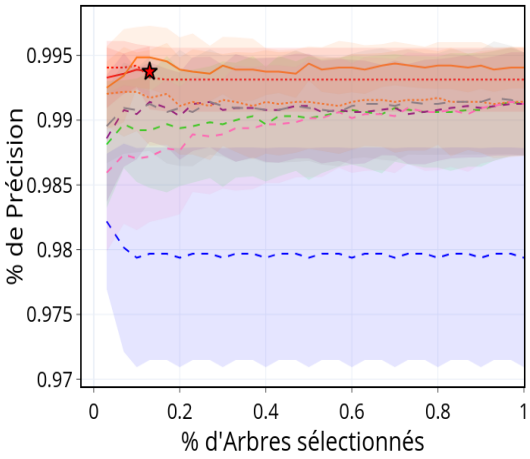


(f) *Diamonds*

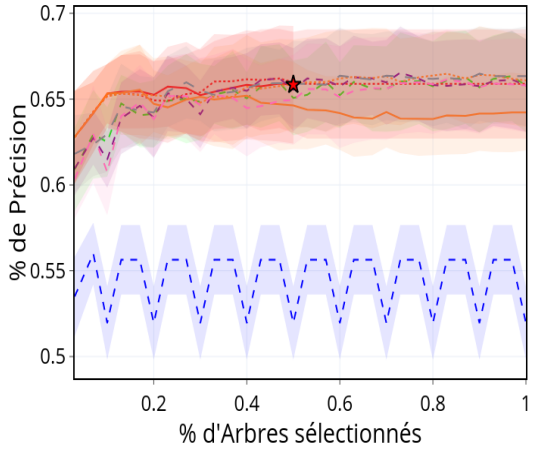
FIGURE 6.6 – Performance sur les données test en fonction de la taille de la forêt élaguée.



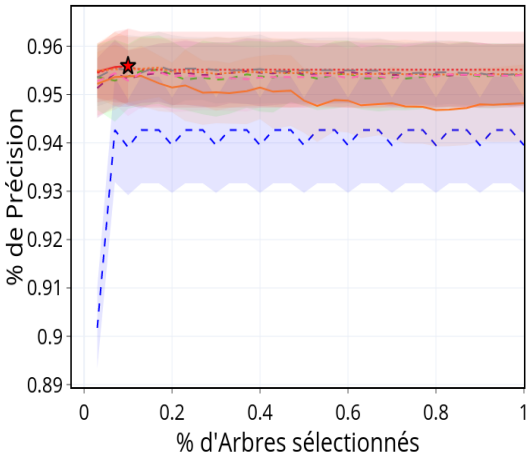
(g) *Gamma*



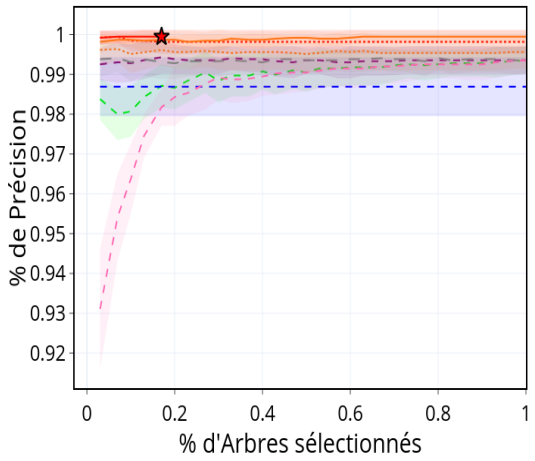
(h) *KR-VS-KP*



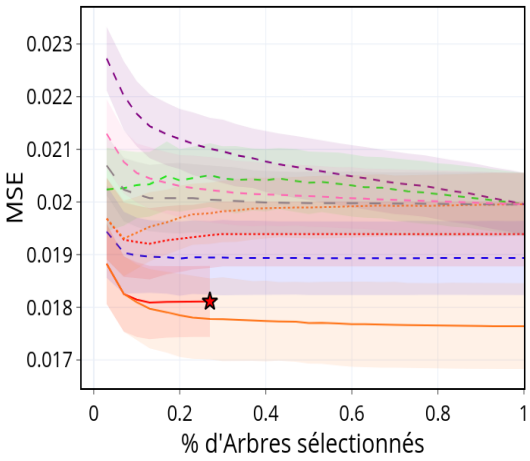
(i) *LFW pairs*



(j) *Spambase*



(k) *Steel Plates*



(l) *Kin8nm*

FIGURE 6.6 – Performance sur les données test en fonction de la taille de la forêt élaguée.

Chapitre 7

Conclusion et perspectives

Nous avons dans ce manuscrit présenté des contributions sur l'accélération des *méthodes à noyaux* et l'optimisation sous contrainte de parcimonie.

Dans ce cadre, nous avons présenté dans le chapitre 3 notre algorithme DAC pour l'approximation des Ridge Leverage Scores (RLS), basé sur la méthode « diviser pour régner ». Nous avons théoriquement justifié l'utilisation de notre algorithme pour l'approximation de matrices de Gram par la méthode de Nyström. Enfin, nous avons empiriquement évalué notre méthode sur deux applications : la méthode de Nyström et l'apprentissage actif. Nous avons montré qu'elle présente un bon compromis entre erreur d'approximation et temps de calcul par rapport aux méthodes de l'état de l'art.

Ce travail a ouvert la porte à un ensemble de questions qui pourront faire l'objet de travaux futurs. Premièrement, il est essentiel d'étudier l'impact de l'utilisation de matrices de Gram approximées comme proposé dans le chapitre 3 sur l'erreur de généralisation des modèles entraînés avec de telles matrices. Une manière d'atteindre cet objectif est d'utiliser les résultats de [Cortes et al., 2010], où les auteurs proposent une borne sur l'erreur d'un SVM et d'une régression ridge à noyau appris avec une matrice de Gram approximée. Cette borne dépend de la norme de la différence entre la matrice de Gram et son approximation. L'idée est alors de combiner ce résultat avec le Théorème 1 du chapitre 3, qui propose justement de borner cette norme. Dans la même veine de recherche, des bornes sur l'erreur des modèles entraînés uniquement sur les données sélectionnées suivant nos RLS approximés (apprentissage actif) peuvent être proposées. Les auteurs de [Imberg et al., 2020] ont déjà théoriquement étudié l'utilisation des RLS exacts dans l'apprentissage actif. Nous pourrions nous en inspirer afin d'étudier l'utilisation de nos RLS approximés avec DAC dans l'apprentissage actif.

Un objectif plus optimiste serait de s'intéresser aux propriétés de l'ensemble des données tirés suivant notre approximation des RLS. En effet, même s'il est clair que les RLS approximés ne sont pas égaux aux RLS exacts, nous pouvons espérer que l'ensemble $\mathcal{S}_1$ des données tirées suivant les RLS approximés par DAC soit « proche » de l'ensemble $\mathcal{S}_2$ des données échantillonnées suivant les RLS exacts. Pouvons-nous par exemple dire que l'intersection entre $\mathcal{S}_1$ et $\mathcal{S}_2$ contient une proportion importante de l'ensemble de toutes les données considérées ? Est-il possible de borner cette proportion ? Un tel résultat nous permettrait de justifier l'utilisation de notre méthode DAC dans l'échantillonnage *online*. Ce type d'échantillonnage est utilisé lorsque le nombre de données est infini et qu'elles arrivent par petit blocs. À chaque nouveau bloc, il faut alors décider quelles données garder et quelles données rejeter, avec la contrainte que l'échantillon final soit tiré suivant une distribution de probabilité basée sur les RLS. Comme notre algorithme DAC ne nécessite pas l'accès à l'ensemble des données, il pourrait être utilisé dans ce contexte d'échantillonnage *online*. Ainsi, si l'ensemble des données tiré suivant les RLS approximés par DAC est « proche » de l'ensemble des données échantillonné suivant les RLS exacts, alors nous pourrons justifier l'utilisation de DAC dans l'échantillonnage *online*.

Dans le chapitre 4, nous avons proposé un nouveau framework pour représenter les distributions de probabilités dans les RKHSs discrets. Nous avons ensuite proposé une famille de distances, paramétrées par un opérateur O_k permettant de comparer les distributions de probabilités discrètes. Nous avons vu que la distance Maximum Mean Discrepancy (MMD) était un cas particulier de cette famille de distances. Cela nous a permis de proposer une approximation de la MMD basée sur la méthode de Nyström. L'utilisation de cette approximation est théoriquement justifiée avec une large famille de fonctions noyaux parmi lesquels les noyaux sur graphes. Cette théorie repose sur l'hypothèse que les noyaux considérés sont *λ-positifs définis* (une notion que nous avons introduite dans ce même chapitre). Dans le cas de la MMD, cette hypothèse vient alors remplacer celle de Gretton and al. [Gretton et al., 2012] qui nécessite que le noyau soit *caractéristique* pour que la théorie de la MMD soit valide. Toutefois, nous avons abordé ces deux familles de fonctions noyaux sans forcément faire le lien entre elles. Ainsi, une perspective très importante à ces travaux serait d'étudier les possibles liens entre ces deux hypothèses. En particulier, un noyau *λ-positif défini* n'est-il pas automatiquement *caractéristique* ? Peut-on affirmer que l'une de ces deux hypothèses soit vérifiée par un nombre plus grand de fonctions noyaux ? Dans ce cas, ce travail permettrait peut-être de justifier l'utilisation de la MMD dans le cas discret avec de nouvelles fonctions noyaux.

D'un point de vue expérimental, nous avons évalué notre méthode sur le *three sample problem* sur des jeux de données vectorielles et de graphes, pour lesquels nous avons utilisé respectivement les noyaux RBF et *ShortestPath*. Une perspective naturelle serait d'utiliser d'autres fonctions noyaux tel que le noyau polynomial pour les données vectorielles et le *RandomWalk* pour les graphes. Notons aussi qu'il serait intéressant de considérer plus de jeux de données de graphes, et en particulier plus grands que celui considéré ici.

Une autre perspective de ce travail est d'étudier les propriétés de nouveaux opérateurs O_k. Rappelons que dans le chapitre 4, l'opérateur O_k était défini comme :

$$\begin{aligned} O_k : \text{span}\left(\{\delta_x(.)\}_{x\in V}\right) &\to \text{span}\left(\{k(x,.)\}_{x\in V}\right) \\ u &\mapsto O_k(u) := \sum_{x_i\in V} c_i \mathrm{P}_{\{x_i\}}(u) \end{aligned} \tag{4.6}$$

où $\mathrm{P}_{\{x_i\}}(u)$ est la projection orthogonale de u sur $k(x_i,.)$. Quelle distance est associée à un opérateur qui projette u, non plus sur une seule fonction noyau, mais sur l'espace vectoriel engendré par un nombre fini de fonctions noyaux indexées par $\mathcal{S}$: $\mathrm{P}_{\{\mathcal{S}\}}(u)$? Est-il par exemple possible de faire le lien entre une telle distance et la MMD calculée sur un noyau approximé par Nyström où $\mathcal{S}$ est l'ensemble de ses *landmarks* ?

Une autre perspective intéressante est l'utilisation de notre MMD approximée pour l'adaptation de domaine. Dans ce scénario, les données d'entraînement $\mathcal{X} := \{x_i\}_{i=1}^n$ et les données de test $\mathcal{Z} := \{z_i\}_{i=1}^m$ ne sont pas tirées suivant la même distribution. Dans ce cas, un modèle naïvement appris sur $\mathcal{X}$, risque d'avoir de très mauvaises performances sur $\mathcal{Z}$. Une méthode connue pour résoudre ce problème est le Kernel Mean Matching (KMM) [Gretton et al., 2009]. Le but de cette méthode est d'apprendre un vecteur de poids β^* tel que la distribution empirique pondérée $\frac{1}{n}\sum_{i=1}^n \beta^*[i]\delta_{x_i}(.)$ « ressemble » le plus possible à la distribution empirique $\frac{1}{m}\sum_{i=1}^m \delta_{z_i}(.)$. Les données d'entraînement ainsi pondérés serviront à l'apprentissage d'un modèle qui devrait maintenant avoir de bonnes performances sur les données de test. Dans la méthode KMM, le vecteur des poids est construit en minimisant la distance MMD entre les deux distributions empiriques. Cela donne lieu à un problème d'optimisation quadratique qui peut être résolu avec la méthode des points intérieurs en $\Theta\left(n^3\right)$. L'idée est alors de remplacer l'utilisation de la MMD exacte par notre approximation de la MMD proposée dans le chapitre 4. La structure de rang faible de la matrice de Gram approximée par la méthode de Nyström nous permet de réduire cette complexité à un temps $\Theta\left(ns^2\right)$ [Fine and Scheinberg, 2001].

Enfin, une dernière perspective serait d'utiliser les RLS approximés par notre algorithme DAC dans la méthode de Nyström pour l'approximation de la MMD. Comme nous avons démontré dans le chapitre 3 que l'utilisation des RLS permet une meilleure approximation de la matrice de Gram que l'échantillonnage uniforme, cette méthode permettrait donc une vitesse de convergence vers la valeur de la MMD exacte meilleure que celle présentée dans le Lemme 6 du chapitre 4. De plus, cela permettrait de borner la distance entre la MMD approximée et la MMD exacte sans hypothèse sur la *cohérence* de la matrice de Gram.

Le chapitre 5 présentait, quant à lui, des travaux légèrement différents des deux précédents. En effet, nous nous sommes intéressés à un problème largement connu dans la communauté de traitement du signal : l'approximation de signaux par des représentations parcimonieuses. Ce problème a souvent été résolu à l'aide d'algorithmes gloutons tel que *Matching Pursuit* (MP) et *Orthogonal Matching Pursuit* (OMP). L'utilisation massive de ces deux algorithmes a été motivée par une large littérature qui justifie théoriquement leur utilisation pour la résolution de ce problème. Nous avons dans ce chapitre étendu une partie de ces résultats à l'algorithme Frank-Wolfe (FW). En particulier, nous avons démontré que si le signal cible est m-parcimonieux dans un dictionnaire de *cohérence* relativement petite par rapport à m, alors tout comme MP et OMP, l'algorithme FW reconstruit exactement le signal cible. De plus, la solution produite par FW converge exponentiellement vite vers la solution optimale.

Une perspective naturelle serait d'étendre ces travaux aux signaux cibles bruités (et donc non m-parcimonieux dans un dictionnaire donné). Pour cela, il est possible de s'inspirer des travaux qui ont été faits pour les algorithmes MP et OMP [Tropp, 2004, Gribonval and Vandergheynst, 2006]. Sur le plan expérimental, nous avons démontré que nos hypothèses, notamment sur le degré de parcimonie du signal et sur la *cohérence* du dictionnaire, ne semblaient pas optimales sur les données synthétiques considérées. Des travaux futurs devraient soit permettre l'exhibition de signaux pour lesquels nos hypothèses sont optimales, soit des conditions moins restrictives sous lesquelles nos résultats restent valides.

Enfin, dans le chapitre 6 nous avons travaillé à l'intersection entre l'apprentissage automatique et le traitement du signal. Nous avons ainsi proposé d'utiliser l'algorithme Orthogonal Matching Pursuit (OMP) ainsi que OMP avec coefficients positifs (NN-OMP), pour la réduction de la taille de forêts aléatoires. Nous avons ensuite empiriquement évalué notre méthode sur plusieurs jeux de données réels de référence. Nous avons vu que notre méthode permettait de réduire la taille des forêts aléatoires tout en gardant de bonnes performances en généralisation. Nous avons enfin

esquissé une piste pour proposer des garanties théoriques en se basant sur les liens entre OMP et Gradient Boosting. Il serait aussi intéressant d'étudier l'utilisation de l'algorithme FW pour l'élagage de forêts aléatoires. Comme nous avons vu dans le chapitre 5 que FW avait de bonnes propriétés pour la reconstruction parcimonieuse, il constituerait peut-être un solide outil pour l'élagage de forêts aléatoires.

Pour conclure, nous avons présenté à travers les différents chapitres de ce manuscrit un ensemble de méthodes permettant de réduire les temps de calculs de certains algorithmes, ainsi que le coût de stockage de certains modèles et certaines données. Nous avons vu que ces approximations permettent très souvent un grand gain en temps de calcul, tout en conservant des performances compétitives.

Le but de cette dernière partie est de donner quelques exemples de grands projets motivants que je souhaite voir émerger dans les prochaines années et dans lesquels j'aimerais m'impliquer.

Le premier concerne la santé. Il existe déjà un certains nombre de travaux à l'intersection entre l'intelligence artificielle (IA) et la santé, mais des freins subsistent encore. Il est par exemple crucial que les modèles utilisés soient interprétables afin de pouvoir justifier à un patient ou au médecin la décision prise par l'IA. De plus, il est important que l'IA fournisse sa réponse dans un temps relativement court tout en offrant de solides garanties sur sa prédiction. En effet, nous pouvons imaginer que des médecins décident d'utiliser une IA afin de calibrer la durée optimale d'un traitement lourd. L'enjeu dans ce cas est de trouver la durée idéale qui permet le rétablissement du patient sans trop le fatiguer en lui infligeant le traitement sur une trop longue période. Les médecins pourraient dans ce cas souhaiter prédire l'état du patient après différentes durées de traitement et choisir l'option optimale. Pour cela, des méthodes d'accélération théoriquement justifiées telles que présentées dans ce manuscrit pourraient être exploitées pour réduire le temps de réponse des modèles d'IA. Cette perspective représente pour moi un axe de recherche intéressant dans lequel je souhaite m'engager. Il est toutefois primordial pour moi de souligner l'importance de garder les médecins au centre des pratiques médicales. L'intérêt de l'IA est avant tout d'assister le praticien, de conforter son premier avis ou au contraire le faire douter et le pousser à effectuer des examens supplémentaires et non de le remplacer.

Le deuxième volet qui me paraît important est l'environnement. Il devient urgent de réfléchir à comment cette riche littérature d'apprentissage automatique peut être utilisée pour considérablement ralentir le réchauffement climatique. Cela pourrait prendre, par exemple, la forme de réseaux de transports intelligents dont la fréquence de passage serait adaptative

à la demande. L'IA pourrait aussi être utilisée pour une meilleure prédiction de la météo, permettant ainsi aux agriculteurs de mieux gérer leurs ressources en eau. Enfin, l'IA pourrait aussi être utilisée pour prédire les émissions de carbone des entreprises afin d'orienter les investissements vers celles qui polluent le moins, et pousser ainsi les autres à réduire leur empreinte écologique. Toutes ces méthodes doivent évidemment rester peu coûteuses pour que leur utilisation ne devienne pas tout aussi polluante que la tâche qu'elle essaient de « dépolluer ». Ainsi, les pratiques d'accélération et d'approximation de modèles doivent continuer à être encouragées.

Annexe A

Preuves du Chapitre 3

A.1 Preuve du Corollaire 1

Corollaire. *1 Soit $\mathcal{X} = \{x_1, \ldots, x_n\}$ un ensemble de données et k un noyau. Pour tout $i \in [n]$, soit l_i le RLS de x_i tel que défini dans la Définition 6, et soit $\hat{l}_i$ une approximation de l_i calculée avec l'Algorithme 2. On a alors :*

$$\hat{l}_i \geq l_i.$$

Démonstration. Soit $\mathcal{R}_1, \ldots, \mathcal{R}_r$ une partition de $\mathcal{X}$ tel que définie dans l'Algorithme 2. Pour tout $\mathcal{R} \in \{\mathcal{R}_1, \ldots, \mathcal{R}_r\}$, soit $\overline{\mathcal{R}}$ le complement de $\mathcal{R}$ dans $\mathcal{X}$, c'est-à-dire : $\overline{\mathcal{R}} := \mathcal{X} \setminus \mathcal{R}$, et soit $K_{\mathcal{R}}$ la matrice de Gram sur les données de $\mathcal{R}$: $K_{\mathcal{R}} = (k(x,z))_{x \in \mathcal{R}, z \in \mathcal{R}}$.

Dans cette preuve nous allons utiliser la définition des RLS décrite dans l'Équation 2.3 : $l_i := 1 - \lambda \left(K_{\mathcal{X}} + \lambda I\right)^{-1}[i,i]$.

Considérons maintenant un x_i fixé. Soit $\mathcal{R}$ le sous-ensemble qui contient x_i, et soit j_i la position de x_i dans $\mathcal{R}$. Ainsi, nous avons : $\hat{l}_i := 1 - \lambda \left(K_{\mathcal{R}} + \lambda I\right)^{-1}[j_i, j_i]$ Sans perte de généralité, supposons que $K_{\mathcal{X}}$ est triée de sorte que les éléments de $\mathcal{R}$ apparaissent dans le premier block :

$$K_{\mathcal{X}} = \begin{bmatrix} K_{\mathcal{R}} & K_{\mathcal{R},\overline{\mathcal{R}}} \\ K_{\mathcal{R},\overline{\mathcal{R}}}^T & K_{\overline{\mathcal{R}}} \end{bmatrix}$$

où $K_{\mathcal{R},\overline{\mathcal{R}}} := (k(x,z))_{x \in \mathcal{R}, z \in \overline{\mathcal{R}}}$ et $K_{\mathcal{R}} := (k(x,z))_{x \in \mathcal{R}, z \in \mathcal{R}}$.

En utilisant l'une des définitions du complément de Shur d'une matrice par block [Petersen et al., 2008], nous avons :

$$\left(K_{\mathcal{X}} + \lambda I\right)^{-1} = \begin{bmatrix} \left(K_{\mathcal{R}} + \lambda I\right)^{-1} + CZC^T & * \\ * & * \end{bmatrix}$$

où

$$C = (K_{\mathcal{R}} + \lambda I)^{-1} K_{\mathcal{R},\overline{\mathcal{R}}}$$

et

$$Z = \left(K_{\overline{\mathcal{R}}} + \lambda I - K_{\mathcal{R},\overline{\mathcal{R}}}^T (K_{\mathcal{R}} + \lambda I)^{-1} K_{\mathcal{R},\overline{\mathcal{R}}}\right)^{-1}.$$

Ainsi, pour tout $x_i \in \mathcal{R}$ nous avons :

$$(K_{\mathcal{X}} + \lambda I)^{-1} [j_i, j_i] = (K_{\mathcal{R}} + \lambda I)^{-1} [j_i, j_i] + [CZC^T][j_i, j_i].$$

et donc :

$$(K_{\mathcal{X}} + \lambda I)^{-1} [j_i, j_i] - (K_{\mathcal{R}} + \lambda I)^{-1} [j_i, j_i] = [CZC^T][j_i, j_i].$$

Puisque la matrice CZC^T est PSD, et que les éléments de la diagonale d'une matrice PSD sont tous positifs ou nuls, nous avons : $[CZC^T][j_i, j_i] \geq 0$. Ainsi :

$$(K_{\mathcal{X}} + \lambda I)^{-1} [j_i, j_i] \geq (K_{\mathcal{R}} + \lambda I)^{-1} [j_i, j_i].$$

Enfin, comme $\lambda > 0$, on en déduit que :

$$1 - \lambda (K_{\mathcal{R}} + \lambda I)^{-1} [j_i, j_i] \geq 1 - \lambda (K_{\mathcal{X}} + \lambda I)^{-1} [j_i, j_i].$$

On en conclut que : $\hat{l}_{j_i} \geq l_{j_i}$. □

A.2 Preuve du Corollaire 2

Corollaire. 2 *Soit $\mathcal{X} = \{x_1, \ldots, x_n\}$ un ensemble de données et k un noyau. Pour tout $i \in [n]$, soit l_i le RLS de x_i tel que défini dans la Définition 6, et soit $\hat{l}_i$ une approximation de l_i calculée avec l'Algorithme 2. On a alors :*

$$\sum_{i=1}^{n} \hat{l}_i \leq \sum_{i=1}^{n} l_i + n\left(1 - \frac{1}{\beta}\right),$$

où $\beta := \frac{\varsigma_{max}(K_{\mathcal{X}}) + \lambda}{\varsigma_{min}(K_{\mathcal{X}}) + \lambda}$.

Démonstration. Comme dans la preuve du Corollaire 1, nous allons utiliser la définition des RLS décrite dans l'Équation 2.3 : $l_i := 1 - \lambda (K_{\mathcal{X}} + \lambda I)^{-1} [i, i]$. Aussi, pour tout $\mathcal{R} \in \{\mathcal{R}_1, \ldots, \mathcal{R}_r\}$, soit $\overline{\mathcal{R}}$ le complement de $\mathcal{R}$ dans $\mathcal{X}$, c'est-à-dire : $\overline{\mathcal{R}} := \mathcal{X} \setminus \mathcal{R}$, et soit $K_{\mathcal{R}}$ la matrice de Gram sur les données de $\mathcal{R}$: $K_{\mathcal{R}} = (k(x, z))_{x \in \mathcal{R}, z \in \mathcal{R}}$. Enfin, pour tout x_i tel

que $x_i \in \mathcal{R}$, soit j_i sa position dans $\mathcal{R}$ tel que défini dans l'Algorithme 2. On a alors :

$$\sum_{i=1}^{n} \hat{l}_i = n - \lambda \sum_{\mathcal{R} \in \{\mathcal{R}_1, \ldots, \mathcal{R}_r\}} \sum_{x_i \in \mathcal{R}} \left(K_{\mathcal{R}} + \lambda I\right)^{-1} [j_i, j_i]. \tag{A.1}$$

Pour commencer, fixons $\mathcal{R}$ et bornons : $\sum_{x_i \in \mathcal{R}} \left(K_{\mathcal{R}} + \lambda I\right)^{-1} [j_i, j_i]$.

Sans perte de généralité, supposons que $K_{\mathcal{X}}$ est triée de sorte que les éléments de $\mathcal{R}$ apparaissent dans le premier block :

$$K_{\mathcal{X}} = \begin{bmatrix} K_{\mathcal{R}} & K_{\mathcal{R},\overline{\mathcal{R}}} \\ K_{\mathcal{R},\overline{\mathcal{R}}}^T & K_{\overline{\mathcal{R}}} \end{bmatrix}$$

où $K_{\mathcal{R},\overline{\mathcal{R}}} := (k(x,z))_{x \in \mathcal{R}, z \in \overline{\mathcal{R}}}$. En utilisant l'une des définitions du complement de Shur [Petersen et al., 2008], on a :

$$\left(K_{\mathcal{X}} + \lambda I\right)^{-1} = \begin{bmatrix} A & * \\ * & * \end{bmatrix},$$

où $A := \left(K_{\mathcal{R}} + \lambda I - K_{\mathcal{R},\overline{\mathcal{R}}} (K_{\overline{\mathcal{R}}} + \lambda I)^{-1} K_{\mathcal{R},\overline{\mathcal{R}}}^T\right)^{-1}$.

L'ingrédient principal de notre preuve est le Théorème 3.6.1 de [Bhatia, 2009] :

$$A \preceq \alpha \left(K_{\mathcal{R}} + \lambda I\right)^{-1},$$

où $\alpha := \frac{(\varsigma_{min} + \varsigma_{max} + 2\lambda)^2}{4(\varsigma_{min} + \lambda)(\varsigma_{max} + \lambda)}$, et où ς_{min} (respectivement ς_{max}) est la plus petite (respectivement la plus grande) valeur propre de $K_{\mathcal{X}}$. Appliquons maintenant la trace à l'inégalité précédente :

$$\text{trace}\left(\left(K_{\mathcal{R}} + \lambda I\right)^{-1}\right) \geq \alpha^{-1} \,\text{trace}\,(A)\,.$$

En utilisant le fait que la trace d'une matrice est la somme des éléments de sa diagonale, et le fait que A soit un block de $\left(K_{\mathcal{X}} + \lambda I\right)^{-1}$, nous obtenons :

$$\sum_{x_i \in \mathcal{R}} (K_{\mathcal{R}} + \lambda I)^{-1} [j_i, j_i] \geq \alpha^{-1} \sum_{x_i \in \mathcal{R}} (K_{\mathcal{X}} + \lambda I)^{-1} [j_i, j_i].$$

On prend maintenant la somme sur tous les sous-ensembles $\mathcal{R}$ pour arriver à borner l'Équation A.1 :

$$\sum_{\mathcal{R}} \sum_{x_i \in \mathcal{R}} (K_{\mathcal{R}} + \lambda I)^{-1} [j_i, j_i] \geq \alpha^{-1} \sum_{\mathcal{R}} \sum_{x_i \in \mathcal{R}} (K_{\mathcal{X}} + \lambda I)^{-1} [j_i, j_i].$$

Nous utilisons ce résultat pour borner la partie droite de l'Équation A.1, et nous obtenons :

$$\sum_{i=1}^{n} \hat{l}_i \leq n - \lambda\alpha^{-1} \sum_{\mathcal{R}} \sum_{x_i \in \mathcal{R}} (K_{\mathcal{X}} + \lambda I)^{-1}[j_i, j_i].$$

Rappelons maintenant que les sous-ensembles $\mathcal{R}$ sont disjoints et que leur union forme $\mathcal{X}$. Ainsi, nous obtenons :

$$\sum_{i=1}^{n} \hat{l}_i \leq n - \lambda\alpha^{-1} \sum_{i=1}^{n} (K_{\mathcal{X}} + \lambda I)^{-1}[i, i].$$

Puisque $(K_{\mathcal{X}} + \lambda I)^{-1}[i, i] = \frac{1-l_i}{\lambda}$, on a :

$$\begin{aligned} \sum_{i=1}^{n} \hat{l}_i &\leq n - \lambda\alpha^{-1} \sum_{i=1}^{n} \frac{1 - l_i}{\lambda} \\ &\leq n - \alpha^{-1} \sum_{i=1}^{n} 1 + \alpha^{-1} \sum_{i=1}^{n} l_i \\ &\leq n \left(1 - \alpha^{-1}\right) + \alpha^{-1} \sum_{i=1}^{n} l_i \end{aligned} \tag{A.2}$$

Notons par $\beta := \frac{\varsigma_{max}+\lambda}{\varsigma_{min}+\lambda}$. Afin d'arriver à notre résultat final, il reste à démontrer que : $\frac{1}{\beta} \leq \alpha^{-1} \leq 1$. Tout d'abord, nous avons :

$$\begin{aligned} \alpha^{-1} - 1 &= \frac{4(\varsigma_{min} + \lambda)(\varsigma_{max} + \lambda) - (\varsigma_{min} + \varsigma_{max} + 2\lambda)^2}{(\varsigma_{min} + \varsigma_{max} + 2\lambda)^2} \\ &= \frac{4\left(\varsigma_{min}\varsigma_{max} + \lambda\varsigma_{min} + \lambda\varsigma_{max}\right) - \varsigma_{min}^2 - \varsigma_{max}^2 - 4\lambda\varsigma_{max} - 2\varsigma_{min}\varsigma_{max} - 4\lambda\varsigma_{min}}{(\varsigma_{min} + \varsigma_{max} + 2\lambda)^2} \\ &= \frac{-\varsigma_{min}^2 - \varsigma_{max}^2 + 2\varsigma_{min}\varsigma_{max}}{(\varsigma_{min} + \varsigma_{max} + 2\lambda)^2} = \frac{-\left(\varsigma_{min} - \varsigma_{max}\right)^2}{(\varsigma_{min} + \varsigma_{max} + 2\lambda)^2} \leq 0. \end{aligned}$$

mais aussi :

$$\begin{aligned} \alpha^{-1} &= \frac{4(\varsigma_{min} + \lambda)(\varsigma_{max} + \lambda)}{(\varsigma_{min} + \varsigma_{max} + 2\lambda)^2} \geq \frac{4(\varsigma_{min} + \lambda)(\varsigma_{max} + \lambda)}{(\varsigma_{max} + \varsigma_{max} + 2\lambda)^2} \\ &\geq \frac{\varsigma_{min} + \lambda}{\varsigma_{max} + \lambda} := \frac{1}{\beta}. \end{aligned}$$

Pour finir, nous utilisons le fait que $\frac{1}{\beta} \leq \alpha^{-1} \leq 1$ dans l'Équation A.2 pour obtenir :

$$\sum_{i=1}^{n} \hat{l}_i \leq \sum_{i=1}^{n} l_i + n \left(1 - \frac{1}{\beta}\right).$$

□

Annexe B

Preuves du Chapitre 5

Dans cette annexe, afin d'alléger les notations, nous notons le support de y par Λ au lieu de $\Lambda(y)$. Nous notons aussi par D la matrice dont les colonnes sont les atomes de $\mathcal{D}$, et par $D_\Lambda \in \mathbb{R}^{n,|\Lambda|}$ la matrice dont les colonnes sont les atomes indexés par Λ.

B.1 Preuve du Théorème 5

Théorème. *5 Soit $\mathcal{D} := \{v_1, \ldots, v_n\}$ un dictionnaire, $\mu(\mathcal{D})$ sa cohérence, $y := \sum_{i=1}^{n} \alpha^*[i] v_i$ un signal m-parcimonieux tel que $\|\alpha^*\|_1 \leq \beta$, et $\mu_1(\mathcal{D}, m)$ la fonction de Babel de $\mathcal{D}$. Si*

$$m < \tfrac{1}{2}(\mu(\mathcal{D})^{-1} + 1),$$

et que pour chaque itération k, le paramètre γ_k est défini suivant la variante 2, alors il existe $j \geq 1$ tel que pour toute itération $k \geq j$ de l'Algorithme 5 vérifie :

$$\|y - y_{k+1}\|_2^2 \leq \|y - y_k\|_2^2 (1 - \theta)$$

où $y_k := \sum_{i=1}^{n} \alpha_k[i] v_i$ et

$$\theta = \tfrac{1}{16} \left(\tfrac{1 - \mu_1(m-1)}{m} \right) \left(1 - \tfrac{\|\alpha^*\|_1}{\beta} \right)^2.$$

Démonstration du Théorème 5. Soit k une itération de l'Algorithme 5. Comme $\gamma_k \in [0, 1]$, nous pouvons découper cette plage de valeur en deux parties :

— $\gamma_k = 0$: dans ce cas, $\alpha_{k+1} = \alpha_k$, et ainsi pour toute itération $l \geq k$ nous aurons : $f(\alpha_k) = f(\alpha_l)$. Comme les valeurs de la fonction objective

convergent vers $f(\alpha^*) = 0$, nous en déduisons que $f(\alpha_l) = f(\alpha^*) = 0$. Par définition de f, nous avons donc : $\|r_{k+1}\|_2^2 = \|r_k\|_2^2 = 0$. Ainsi, le Théorème 5 est vrai.

— $0 < \gamma_k \leq 1$: Par définition du résidu, nous avons :

$$\begin{aligned}\|r_{k+1}\|_2^2 &= \|y - D\alpha_{k+1}\|_2^2 = \|r_k + D\alpha_k - D\alpha_{k+1}\|_2^2 \\ &= \|r_k - D\gamma_k(s_k - \alpha_k)\|_2^2 \\ &= \|r_k\|_2^2 - 2\gamma_k\langle D(s_k - \alpha_k), r_k\rangle + \gamma_k^2 \|D(s_k - \alpha_k)\|_2^2 .\end{aligned} \tag{B.1}$$

Comme pour tout vecteur $a \in \mathbb{R}^n$: $\|Da\|_2 \leq \|a\|_1$ puisque les v_i sont normés ($\|Da\|_2^2 = \sum_{i,j=1}^n a[i]a[j]\langle v_i, v_j\rangle \leq \sum_{i,j=1}^n |a[i]||a[j]| = \|a\|_1^2$). Alors nous avons :

$$\|y\|_2 = \|D\alpha^*\|_2 \leq \|\alpha^*\|_1 < \beta.$$

Nous pouvons donc utiliser le résultat du Lemme 7, qui nous dit :

$$\gamma_k = \frac{\langle D(s_k - \alpha_k), r_k\rangle}{\|D(s_k - \alpha_k)\|_2^2}.$$

En remplaçant la valeur de γ_k dans l'Équation (B.1), nous obtenons :

$$\|r_{k+1}\|_2^2 = \|r_k\|_2^2 - \frac{\langle D(s_k - \alpha_k), r_k\rangle^2}{\|D(s_k - \alpha_k)\|_2^2}. \tag{B.2}$$

Nous devons maintenant trouver une borne inférieure à $\langle D(s_k - \alpha_k), r_k\rangle^2$, et une borne supérieure à $\|D(s_k - \alpha_k)\|_2^2$. Pour borner $\|D(s_k - \alpha_k)\|_2^2$, nous utilisons le fait que $\|Da\|_2 \leq \|a\|_1$ et le fait que s_k et α_k sont dans la boule $\mathcal{B}_1(\beta)$:

$$\|D(s_k - \alpha_k)\|_2^2 \leq \|s_k - \alpha_k\|_1^2 \leq 4\beta^2. \tag{B.3}$$

Pour borner $\langle D(s_k - \alpha_k), r_k\rangle^2$, nous fixons $\epsilon = \frac{\beta - \|\alpha^*\|_1}{2} > 0$. Comme prouver dans le Corollaire 3, les itérés α_k convergent vers α^*. Ainsi, il existe une itération $l \geq 0$, tel que pour tout itération $k \geq l$: $\|\alpha_k - \alpha^*\|_1 \leq \epsilon$. Définissons $u \in \mathbb{R}^n$, tel que :

$$u[i] = \begin{cases} \frac{\epsilon}{\sqrt{m}\|D_\Lambda^T r_k\|_2}(D_\Lambda^T r_k)[i] & \text{si } i \in \Lambda \\ 0 & \text{sinon.} \end{cases}$$

On peut montrer que $\alpha_k + u$ est dans la boule $\mathcal{B}_1(\beta)$. En effet :

$$\begin{aligned}\|\alpha_k + u\|_1 &\leq \|\alpha_k - \alpha^*\|_1 + \|\alpha^*\|_1 + \|u\|_1 \\ &\leq \epsilon + \|\alpha^*\|_1 + \frac{\epsilon}{\sqrt{m}\left\|D_\Lambda^T r_k\right\|_2}\left\|D_\Lambda^T r_k\right\|_1 .\end{aligned}$$

En remarquant que $\left\|D_\lambda^T r_k\right\|_1 \leq \sqrt{m}\left\|D_\Lambda^T r_k\right\|_2$ (parce que $D_\Lambda^T r_k \in \mathbb{R}^{|\Lambda|}$ et $|\Lambda| \leq m$), nous obtenons :

$$\|\alpha_k + u\|_1 \leq 2\epsilon + \|\alpha^*\|_1 = \beta.$$

On conclut que : $\alpha_k + u$ est dans la boule $\mathcal{B}_1(\beta)$. Comme $s_k = \arg\min_{s\in\mathcal{B}_1(\beta)}\langle s, \nabla f(\alpha_k)\rangle$, alors :

$$\langle s_k, \nabla f(\alpha_k)\rangle \leq \langle \alpha_k + u, \nabla f(\alpha_k)\rangle,$$

et donc :

$$\langle s_k - \alpha_k, \nabla f(\alpha_k)\rangle \leq \langle u, \nabla f(x_k)\rangle.$$

Par définition de f : $\nabla f(\alpha_k) = -D^T r_k$, ainsi :

$$\langle s_k - \alpha_k, D^T r_k\rangle \geq \langle u, D^T r_k\rangle,$$

et donc :

$$\langle D(s_k - \alpha_k), r_k\rangle \geq \langle u, D^T r_k\rangle.$$

Nous allons maintenant borner $\langle u, D^T r_k\rangle$. Or, comme :

$$\langle u, D^T r_k\rangle = \sum_{i\in\Lambda}\frac{\epsilon}{\sqrt{m}\left\|D_\Lambda^T r_k\right\|_2}(D_\Lambda^T r_k)^2[i] = \frac{\epsilon\left\|D_\Lambda^T r_k\right\|_2}{\sqrt{m}},$$

on a alors :

$$\langle D(s_k - \alpha_k), r_k\rangle \geq \frac{\epsilon}{\sqrt{m}}\left\|D_\Lambda^T r_k\right\|_2 . \tag{B.4}$$

Par le Théorème 4, nous savons que r_k est dans le span(D_Λ), et comme les atomes indexés par Λ sont linéairement indépendants, nous avons que $\lambda_{min}(D_\Lambda) > 0$. Aussi, par la remarque 2, nous avons :

$$\left\|D_\Lambda^T r_k\right\|_2 \geq \lambda_{min}(D_\Lambda)\|r_k\|_2$$

Par le Lemme 2.3 de [Tropp, 2004], $\lambda_{min}(D_\Lambda)^2 \geq 1 - \mu_1(\mathcal{D}, m-1)$. Ainsi, l'Équation (B.4) devient :

$$\langle D(s_k - \alpha_k), r_k\rangle \geq \epsilon\sqrt{\frac{1-\mu_1(m-1)}{m}}\|r_k\|_2 . \tag{B.5}$$

En utilisant les Équations (B.3) et (B.5) dans l'Équation (B.2), on obtient :

$$\begin{aligned}
\|r_{k+1}\|^2 &= \|r_k\|^2 - \frac{\langle D(s_k - \alpha_k), r_k\rangle^2}{\|D(s_k - \alpha_k)\|^2} \\
\|r_{k+1}\|^2 &\leq \|r_k\|^2 \left(1 - \frac{\epsilon^2(1-\mu_1(m-1))}{4\beta^2 m}\right) \\
\|r_{k+1}\|^2 &\leq \|r_k\|^2 \left(1 - \frac{1}{16}\left(\frac{1-\mu_1(m-1)}{m}\right)\left(1 - \frac{\|\alpha^*\|_1}{\beta}\right)^2\right),
\end{aligned}$$

où la dernière égalité est vraie car : $\frac{\epsilon^2}{4\beta^2} = \frac{1}{16}\left(1 - \frac{\|\alpha^*\|_1}{\beta}\right)$. En effet, par la définition de $\epsilon = \frac{\beta - \|\alpha^*\|_1}{2}$, on a :

$$\begin{aligned}
\frac{\epsilon^2}{4\beta^2} - \frac{1}{16}\left(1 - \frac{\|\alpha^*\|_1}{\beta}\right)^2 &= \frac{4\epsilon^2 - \|\alpha^*\|_1^2 - \beta^2 + 2\|\alpha^*\|_1 \beta}{16\beta^2} \\
&= \frac{(\beta - \|\alpha\|_1)^2 - \|\alpha^*\|_1^2 - \beta^2 + 2\|\alpha^*\|_1 \beta}{16\beta^2} = 0,
\end{aligned}$$

ce qui termine notre preuve. □

Bibliographie

[Dia,] Diamonds dataset. https://www.kaggle.com/shivam2503/diamonds.

[Adler, 2015] Adler, A. (2015). Covariance-assisted matching pursuit. *IEEE Signal Processing Letters*, 23(1) :149–153.

[Ahmed and Moustafa, 2016] Ahmed, E. H. and Moustafa, M. (2016). House price estimation from visual and textual features. In *International Conference on Neural Computation Theory and Applications*, volume 4, pages 62–68. SCITEPRESS.

[Aizerman, 1964] Aizerman, M. A. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25 :821–837.

[Aizerman et al., 1964] Aizerman, M. A., Braverman, E. M., and Rozonoèr, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25 :821–837.

[Alaoui and Mahoney, 2015] Alaoui, A. and Mahoney, M. W. (2015). Fast randomized kernel ridge regression with statistical guarantees. *Advances in neural information processing systems (NeurIPS)*, 28.

[Arcolano and Wolfe, 2010] Arcolano, N. and Wolfe, P. J. (2010). Nyström approximation of Wishart matrices. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3606–3609.

[Arcolano and Wolfe, 2011] Arcolano, N. and Wolfe, P. J. (2011). Estimating principal components of large covariance matrices using the nyström method. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3784–3787.

[Aronszajn, 1950] Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3) :337–404.

[Avron et al., 2017] Avron, H., Kapralov, M., Musco, C., Musco, C., Velingker, A., and Zandieh, A. (2017). Random fourier features for kernel ridge regression : Approximation bounds and statistical guarantees. In *International Conference on Machine Learning*, pages 253–262. PMLR.

[Bach, 2013] Bach, F. (2013). Sharp analysis of low-rank kernel matrix approximations. In *Conference on Learning Theory*, pages 185–209. PMLR.

[Belabbas and Wolfe, 2009] Belabbas, M.-A. and Wolfe, P. J. (2009). Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences*, 106(2) :369–374.

[Belsley et al., 1980] Belsley, D. A., Kuh, E., and Welsch, R. E. (1980). *Regression diagnostics : Identifying influential data and sources of collinearity*, volume 571. John Wiley & Sons.

[Bernard et al., 2012] Bernard, S., Adam, S., and Heutte, L. (2012). Dynamic random forests. *Pattern Recognition Letters*, 33(12) :1580–1586.

[Bhatia, 1997] Bhatia, R. (1997). *Matrix analysis*. Springer.

[Bhatia, 2009] Bhatia, R. (2009). *Positive definite matrices*. Princeton university press.

[Blumensath and Davies, 2008] Blumensath, T. and Davies, M. E. (2008). Gradient pursuits. *IEEE Transactions on Signal Processing*, 56(6) :2370–2382.

[Bock et al., 2004] Bock, R., Chilingarian, A., Gaug, M., Hakl, F., Hengstebeck, T., Jiřina, M., Klaschka, J., Kotrč, E., Savickỳ, P., Towers, S., et al. (2004). Methods for multidimensional event classification : a case study using images from a cherenkov gamma-ray telescope. *Nuclear Instruments and Methods in Physics Research Section A : Accelerators, Spectrometers, Detectors and Associated Equipment*, 516(2-3) :511–528.

[Borgwardt et al., 2006] Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14) :e49–e57.

[Borgwardt and Kriegel, 2005] Borgwardt, K. M. and Kriegel, H.-P. (2005). Shortest-path kernels on graphs. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM 2005)*, pages 74–81, Washington, DC, USA. IEEE Computer Society.

[Boser et al., 1992a] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992a). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

[Boser et al., 1992b] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992b). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Brefeld et al., 2006] Brefeld, U., Gärtner, T., Scheffer, T., and Wrobel, S. (2006). Efficient co-regularised least squares regression. In *International conference on Machine learning (ICML)*, pages 137–144.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1) :5–32.

[Brown and Kuncheva, 2010] Brown, G. and Kuncheva, L. I. (2010). "good" and "bad" diversity in majority vote ensembles. In El Gayar, N., Kittler, J., and Roli, F., editors, *Multiple Classifier Systems*, pages 124–133, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Bruckstein et al., 2008] Bruckstein, A. M., Elad, M., and Zibulevsky, M. (2008). Sparse non-negative solution of a linear system of equations is unique. In *2008 3rd International Symposium on Communications, Control and Signal Processing*, pages 762–767. IEEE.

[Bubeck et al., 2015] Bubeck, S. et al. (2015). Convex optimization : Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4) :231–357.

[Buciluǎ et al., 2006] Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

[Buitinck et al., 2013] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Olivier Grisel, V. N., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software : experiences from the scikit-learn project. In *ECML PKDD Workshop : Languages for Data Mining and Machine Learning*, pages 108–122.

[Buscema, 1998] Buscema, M. (1998). Metanet* : The theory of independent judges. *Substance use & misuse*, 33(2) :439–461.

[Calandriello et al., 2016] Calandriello, D., Lazaric, A., and Valko, M. (2016). Analysis of nyström method with sequential ridge leverage score sampling. In *International conference on Uncertainty in Artificial Intelligence (UAI)*.

[Campbell and Gear, 1995] Campbell, S. L. and Gear, C. W. (1995). The index of general nonlinear DAES. *Numer. Math.*, 72(2) :173–196.

[Candes and Tao, 2005] Candes, E. J. and Tao, T. (2005). Decoding by linear programming. *IEEE transactions on information theory*, 51(12) :4203–4215.

[Caruana et al., 2004] Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004). Ensemble selection from libraries of models. In *Procee-*

dings of the twenty-first international conference on Machine learning, page 18. ACM.

[Chandra and Sharma, 2016] Chandra, B. and Sharma, R. K. (2016). Fast learning in deep neural networks. *Neurocomputing*, 171 :1205–1215.

[Chattopadhyay et al., 2013a] Chattopadhyay, R., Fan, W., Davidson, I., Panchanathan, S., and Ye, J. (2013a). Joint transfer and batch-mode active learning. In *International conference on machine learning*, pages 253–261. PMLR.

[Chattopadhyay et al., 2013b] Chattopadhyay, R., Wang, Z., Fan, W., Davidson, I., Panchanathan, S., and Ye, J. (2013b). Batch mode active sampling based on marginal probability distribution matching. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3) :1–25.

[Chen et al., 2001] Chen, S. S., Donoho, D. L., and Saunders, M. A. (2001). Atomic decomposition by basis pursuit. *SIAM review*, 43(1) :129–159.

[Chen and Yang, 2021] Chen, Y. and Yang, Y. (2021). Fast statistical leverage score approximation in kernel ridge regression. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2935–2943.

[Cherfaoui et al., 2019] Cherfaoui, F., Emiya, V., Ralaivola, L., and Anthoine, S. (2019). Recovery and convergence rate of the frank–wolfe algorithm for the m-exact-sparse problem. *IEEE Transactions on Information Theory*, 65(11) :7407–7414.

[Chwialkowski et al., 2015] Chwialkowski, K. P., Ramdas, A., Sejdinovic, D., and Gretton, A. (2015). Fast two-sample testing with analytic representations of probability measures. *Advances in Neural Information Processing Systems*, 28 :1981–1989.

[Cohen et al., 2015a] Cohen, M. B., Lee, Y. T., Musco, C., Musco, C., Peng, R., and Sidford, A. (2015a). Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190.

[Cohen et al., 2015b] Cohen, M. B., Lee, Y. T., Musco, C., Musco, C., Peng, R., and Sidford, A. (2015b). Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190.

[Cohen et al., 2016] Cohen, M. B., Musco, C., and Pachocki, J. (2016). Online row sampling. *arXiv preprint arXiv :1604.05448*.

[Corke, 1996] Corke, P. I. (1996). A robotics toolbox for matlab. *IEEE Robotics & Automation Magazine*, 3(1) :24–32.

[Cortes et al., 2010] Cortes, C., Mohri, M., and Talwalkar, A. (2010). On the impact of kernel approximation on learning accuracy. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 113–120. JMLR Workshop and Conference Proceedings.

[Courty et al., 2016] Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2016). Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9) :1853–1865.

[Cranor and LaMacchia, 1998] Cranor, L. F. and LaMacchia, B. A. (1998). Spam ! *Communications of the ACM*, 41(8) :74–83.

[Davenport and Wakin, 2010] Davenport, M. A. and Wakin, M. B. (2010). Analysis of orthogonal matching pursuit using the restricted isometry property. *IEEE Transactions on Information Theory*, 56(9) :4395–4401.

[Davis et al., 1997] Davis, G., Mallat, S., and Avellaneda, M. (1997). Adaptive greedy approximations. *Constructive approximation*, 13(1) :57–98.

[Debnath et al., 1991] Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C. (1991). Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2) :786–797.

[Dong et al., 2005] Dong, J.-x., Krzyzak, A., and Suen, C. Y. (2005). Fast svm training algorithm with decomposition on very large data sets. *IEEE transactions on pattern analysis and machine intelligence*, 27(4) :603–618.

[Donoho et al., 2001] Donoho, D. L., Huo, X., et al. (2001). Uncertainty principles and ideal atomic decomposition. *IEEE transactions on information theory*, 47(7) :2845–2862.

[Drineas et al., 2005] Drineas, P., Mahoney, M. W., and Cristianini, N. (2005). On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(12).

[Dziugaite et al., 2015] Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015). Training generative neural networks via maximum mean discrepancy optimization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 258–267.

[Efron et al., 2004] Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. *The Annals of statistics*, 32(2) :407–499.

[Elvira et al., 2021] Elvira, C., Gribonval, R., Soussen, C., and Herzet, C. (2021). When does omp achieve exact recovery with continuous dictionaries ? *Applied and Computational Harmonic Analysis*, 51 :374–413.

[Fahrbach et al., 2021] Fahrbach, M., Ghadiri, M., and Fu, T. (2021). Fast low-rank tensor decomposition by ridge leverage score sampling. *arXiv preprint arXiv :2107.10654*.

[Farah et al., 2022] Farah, C., Hachem, K., and Liva, R. (2022). Scalable ridge leverage score sampling for the nyström method. *IEEE International Conference on Acoustics, Speech and Signal Processing*.

[Fawagreh et al., 2016] Fawagreh, K., Gaber, M. M., and Elyan, E. (2016). An outlier ranking tree selection approach to extreme pruning of random forests. In *International Conference on Engineering Applications of Neural Networks*, pages 267–282. Springer.

[Fine and Scheinberg, 2001] Fine, S. and Scheinberg, K. (2001). Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2 :243–264.

[Frank and Wolfe, 1956] Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 3(1-2) :95–110.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation : a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

[Frieze et al., 2004] Frieze, A., Kannan, R., and Vempala, S. (2004). Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6) :1025–1041.

[Fukumizu et al., 2004] Fukumizu, K., Bach, F. R., and Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5(Jan) :73–99.

[Gartner, 2008] Gartner, T. (2008). *Kernels for structured data*, volume 72. World Scientific.

[Ge et al., 2019] Ge, H., Wang, L., Wen, J., and Xian, J. (2019). An rip condition for exact support recovery with covariance-assisted matching pursuit. *IEEE Signal Processing Letters*, 26(3) :520–524.

[Germain et al., 2013] Germain, P., Habrard, A., Laviolette, F., and Morvant, E. (2013). A pac-bayesian approach for domain adaptation with specialization to linear classifiers. In *International conference on machine learning*, pages 738–746. PMLR.

[Giffon et al., 2021] Giffon, L., Emiya, V., Kadri, H., and Ralaivola, L. (2021). Quick-means : accelerating inference for k-means by learning fast transforms. *Machine Learning*, 110(5) :881–905.

[Giffon et al., 2020] Giffon, L., Lamothe, C., Bouscarrat, L., Milanesi, P., Cherfaoui, F., and Koço, S. (2020). Pruning random forest with orthogonal matching trees.

[Gilbert et al., 2003] Gilbert, A. C., Muthukrishnan, S., and Strauss, M. J. (2003). Approximation of functions over redundant dictionaries using coherence. In *SODA*, pages 243–252. Citeseer.

[Gittens and Mahoney, 2013] Gittens, A. and Mahoney, M. (2013). Revisiting the nyström method for improved large-scale machine learning. In *International Conference on Machine Learning*, pages 567–575.

[Gittens and Mahoney, 2016] Gittens, A. and Mahoney, M. W. (2016). Revisiting the nyström method for improved large-scale machine learning. *The Journal of Machine Learning Research (JMLR)*, 17(1) :3977–4041.

[Gray, 1984] Gray, R. (1984). Vector quantization. *IEEE Assp Magazine*, 1(2) :4–29.

[Gretton et al., 2012] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1) :723–773.

[Gretton et al., 2009] Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. (2009). Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4) :5.

[Gribonval and Vandergheynst, 2006] Gribonval, R. and Vandergheynst, P. (2006). On the exponential convergence of matching pursuits in quasi-incoherent dictionaries. *IEEE Transactions on Information Theory*, 52(1) :255–261.

[Guélat and Marcotte, 1986] Guélat, J. and Marcotte, P. (1986). Some comments on wolfe's away step. *Mathematical Programming*, 35(1) :110–119.

[Gutman, 1989] Gutman, M. (1989). Asymptotically optimal classification for multiple tests with empirically observed statistics. *IEEE Transactions on Information Theory*, 35(2) :401–408.

[Hagberg and Conway,] Hagberg, A. and Conway, D. Networkx : Network analysis with python.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning : data mining, inference, and prediction*, volume 2. Springer.

[He et al., 2017] He, Y., Zhang, X., and Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397.

[Head et al., 2018] Head, T., MechCoder, L., Shcherbatyi, I., et al. (2018). scikit-optimize/scikit-optimize : v0. 5.2.

[Herzet et al., 2013] Herzet, C., Soussen, C., Idier, J., and Gribonval, R. (2013). Exact recovery conditions for sparse representations with partial support information. *IEEE transactions on information theory*, 59(11) :7509–7524.

[Hofmann et al., 2008] Hofmann, T., Schölkopf, B., and Smola, A. J. (2008). Kernel methods in machine learning. *The annals of statistics*, 36(3) :1171–1220.

[Hu et al., 2007] Hu, Q., Yu, D., Xie, Z., and Li, X. (2007). Eros : Ensemble rough subspaces. *Pattern recognition*, 40(12) :3728–3739.

[Huang et al., 2008] Huang, G. B., Mattar, M., Berg, T., and Learned-Miller, E. (2008). Labeled faces in the wild : A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images : detection, alignment, and recognition*.

[Imberg et al., 2020] Imberg, H., Jonasson, J., and Axelson-Fisk, M. (2020). Optimal sampling in unbiased active learning. In *International Conference on Artificial Intelligence and Statistics*, pages 559–569. PMLR.

[Jaggi, 2013] Jaggi, M. (2013). Revisiting frank-wolfe : Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435. PMLR.

[Jorgensen and Tian, 2015] Jorgensen, P. and Tian, F. (2015). Discrete reproducing kernel hilbert spaces : sampling and distribution of dirac-masses. *The Journal of Machine Learning Research*, 16(1) :3079–3114.

[Kohavi, 1996] Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers : a decision-tree hybrid. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*.

[Kolouri et al., 2019] Kolouri, S., Nadjahi, K., Simsekli, U., Badeau, R., and Rohde, G. (2019). Generalized sliced wasserstein distances. *Advances in Neural Information Processing Systems*, 32.

[Kulesza et al., 2012] Kulesza, A., Taskar, B., et al. (2012). Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3) :123–286.

[Kulkarni and Sinha, 2012] Kulkarni, V. Y. and Sinha, P. K. (2012). Pruning of random forest classifiers : A survey and future directions. In *2012 International Conference on Data Science & Engineering (ICDSE)*, pages 64–68. IEEE.

[Kumar et al., 2009a] Kumar, S., Mohri, M., and Talwalkar, A. (2009a). On sampling-based approximate spectral decomposition. In *Proceedings of the 26th annual international conference on machine learning*, pages 553–560.

[Kumar et al., 2009b] Kumar, S., Mohri, M., and Talwalkar, A. (2009b). Sampling techniques for the nyström method. In *Artificial intelligence and statistics*, pages 304–311. PMLR.

[Lacoste-Julien, 2016] Lacoste-Julien, S. (2016). Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv :1607.00345*.

[Lacoste-Julien and Jaggi, 2015] Lacoste-Julien, S. and Jaggi, M. (2015). On the global linear convergence of frank-wolfe optimization variants. *Advances in neural information processing systems*, 28.

[Lelewer and Hirschberg, 1987] Lelewer, D. A. and Hirschberg, D. S. (1987). Data compression. *ACM Computing Surveys (CSUR)*, 19(3) :261–296.

[Li et al., 2016] Li, C., Jegelka, S., and Sra, S. (2016). Fast DPP sampling for nyström with application to kernel methods. In *International Conference on Machine Learning (ICML)*, pages 2061–2070.

[Li et al., 2017] Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. (2017). Mmd gan : Towards deeper understanding of moment matching network. *Advances in neural information processing systems*, 30.

[Li et al., 2019] Li, Z., Ton, J.-F., Oglic, D., and Sejdinovic, D. (2019). Towards a unified analysis of random fourier features. In *International conference on machine learning*, pages 3905–3914. PMLR.

[Liu et al., 2016] Liu, G., Chen, H., Sun, X., and Qiu, R. C. (2016). Modified MUSIC algorithm for DOA estimation with nyström approximation. *IEEE Sensors Journal*, 16(12) :4673–4674.

[Liu et al., 2015] Liu, Y., Wang, Y., and Sowmya, A. (2015). Batch mode active learning for object detection based on maximum mean discrepancy. In *2015 International Conference on Digital Image Computing : Techniques and Applications (DICTA)*, pages 1–7. IEEE.

[Locatello et al., 2017] Locatello, F., Khanna, R., Tschannen, M., and Jaggi, M. (2017). A unified optimization view on generalized matching pursuit and frank-wolfe. In *Artificial Intelligence and Statistics*, pages 860–868.

[Locatello et al., 2018] Locatello, F., Raj, A., Karimireddy, S. P., Rätsch, G., Schölkopf, B., Stich, S., and Jaggi, M. (2018). On matching pursuit and coordinate descent. In *International Conference on Machine Learning*, pages 3198–3207. PMLR.

[Lopez-Paz et al., 2014] Lopez-Paz, D., Sra, S., Smola, A., Ghahramani, Z., and Schölkopf, B. (2014). Randomized nonlinear component analysis. In *International conference on machine learning*, pages 1359–1367. PMLR.

[Mairal et al., 2009] Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696.

[Mallat and Zhang, 1993] Mallat, S. G. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12) :3397–3415.

[Mangasarian et al., 1995] Mangasarian, O. L., Street, W. N., and Wolberg, W. H. (1995). Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4) :570–577.

[Manton and Amblard, 2015] Manton, J. H. and Amblard, P.-O. (2015). A primer on reproducing kernel hilbert spaces. *Foundations and Trends® in Signal Processing*, 8(1–2) :1–126.

[McCurdy, 2018] McCurdy, S. (2018). Ridge regression and provable deterministic ridge leverage score sampling. *Advances in Neural Information Processing Systems*, 31.

[Miller, 2002] Miller, A. (2002). *Subset selection in regression*. Chapman and Hall/CRC.

[Mitchell et al., 1974] Mitchell, B., Dem'yanov, V. F., and Malozemov, V. (1974). Finding the point of a polyhedron closest to the origin. *SIAM Journal on Control*, 12(1) :19–26.

[Muandet et al., 2017] Muandet, K., Fukumizu, K., Sriperumbudur, B., and Schölkopf, B. (2017). Kernel mean embedding of distributions : A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2) :1–141.

[Musco and Musco, 2017] Musco, C. and Musco, C. (2017). Recursive sampling for the nyström method. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

[Natekin and Knoll, 2013] Natekin, A. and Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7 :21.

[Ni et al., 2020] Ni, F., Zhang, J., and Noori, M. N. (2020). Deep learning for data anomaly detection and data compression of a long-span suspension bridge. *Computer-Aided Civil and Infrastructure Engineering*, 35(7) :685–700.

[Novikov et al., 2015] Novikov, A., Podoprikhin, D., Osokin, A., and Vetrov, D. P. (2015). Tensorizing neural networks. *Advances in neural information processing systems*, 28.

[Pace and Barry, 1997] Pace, R. K. and Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3) :291–297.

[Pati et al., 1993] Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S. (1993). Orthogonal matching pursuit : Recursive function approximation with applications to wavelet decomposition. *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44.

[Petersen et al., 2008] Petersen, K. B., Pedersen, M. S., et al. (2008). The matrix cookbook. *Technical University of Denmark*, 7(15) :510.

[Pourkamali-Anaraki, 2020] Pourkamali-Anaraki, F. (2020). Scalable spectral clustering with nyström approximation : Practical and theoretical aspects. *IEEE Open Journal of Signal Processing*, 1 :242–256.

[R. Collobert, S. Bengio, and Y. Bengio, 2002] R. Collobert, S. Bengio, and Y. Bengio (2002). A parallel mixture of SVMs for very large scale problems. *Neural Computation, 14(05) :1105-1114*.

[Rahimi and Recht, 2007] Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20.

[Rao and Kreutz-Delgado, 1999] Rao, B. D. and Kreutz-Delgado, K. (1999). An affine scaling methodology for best basis selection. *IEEE Transactions on signal processing*, 47(1) :187–200.

[Rudi et al., 2018] Rudi, A., Calandriello, D., Carratino, L., and Rosasco, L. (2018). On fast leverage score sampling and optimal learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 31.

[Ryabko and Mary, 2013] Ryabko, D. and Mary, J. (2013). A binary-classification-based metric between time-series distributions and its use in statistical and learning problems. *The Journal of Machine Learning Research*, 14(1) :2837–2856.

[Sayyad Shirabad and Menzies, 2005] Sayyad Shirabad, J. and Menzies, T. (2005). The promise repository of software engineering databases.

[Schapire, 1999] Schapire, R. E. (1999). A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406. Citeseer.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT press.

[Settles, 2009] Settles, B. (2009). Active learning literature survey.

[Shapiro, 1987] Shapiro, A. D. (1987). *Structured induction in expert systems*. Addison-Wesley Longman Publishing Co., Inc.

[Shawe-Taylor and Cristianini, 2004] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

[Shen et al., 2018] Shen, J., Qu, Y., Zhang, W., and Yu, Y. (2018). Wasserstein distance guided representation learning for domain adaptation. In *Thirty-second AAAI conference on artificial intelligence*.

[Simon-Gabriel and Schölkopf, 2018] Simon-Gabriel, C.-J. and Schölkopf, B. (2018). Kernel distribution embeddings : Universal kernels, characteristic kernels and kernel metrics on distributions. *The Journal of Machine Learning Research*, 19(1) :1708–1736.

[Smola and Schölkopf, 2000] Smola, A. J. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning.

[Sriperumbudur et al., 2011] Sriperumbudur, B. K., Fukumizu, K., and Lanckriet, G. R. (2011). Universality, characteristic kernels and rkhs embedding of measures. *Journal of Machine Learning Research*, 12(7).

[Sutherland and Schneider, 2015] Sutherland, D. J. and Schneider, J. (2015). On the error of random fourier features. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 862–871.

[Temlyakov, 2003] Temlyakov, V. N. (2003). Nonlinear methods of approximation. *Foundations of Computational Mathematics*, 3(1).

[Temlyakov, 2011] Temlyakov, V. N. (2011). *Greedy approximation*, volume 20. Cambridge University Press.

[Tibshirani, 1996] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society : Series B (Methodological)*, 58(1) :267–288.

[Tong and Koller, 2001] Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov) :45–66.

[Tropp, 2004] Tropp, J. A. (2004). Greed is good : Algorithmic results for sparse approximation. *IEEE Transactions on Information theory*, 50(10) :2231–2242.

[Tropp, 2006] Tropp, J. A. (2006). Just relax : Convex programming methods for identifying sparse signals in noise. *IEEE transactions on information theory*, 52(3) :1030–1051.

[Wang and Zhang, 2009] Wang, M. and Zhang, H. (2009). Search for the smallest random forest. *Statistics and its Interface*, 2(3) :381–388.

[Williams and Seeger, 2000] Williams, C. and Seeger, M. (2000). Using the nyström method to speed up kernel machines. *Advances in neural information processing systems (NeurIPS)*, 13.

[Wolfe, 1970] Wolfe, P. (1970). Convergence theory in nonlinear programming. *Integer and nonlinear programming*, pages 1–36.

[Yang et al., 2012a] Yang, F., Lu, W.-h., Luo, L.-k., and Li, T. (2012a). Margin optimization based pruning for random forest. *Neurocomputing*, 94 :54–63.

[Yang et al., 2012b] Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. (2012b). Nyström method vs random fourier features : A theoretical and empirical comparison. *Advances in neural information processing systems*, 25 :476–484.

[Yann et al., 1998] Yann, L., Corinna, C., and Christopher, J. B. (1998). Mnist handwritten digit database.

[Yu et al., 2017] Yu, X., Liu, T., Wang, X., and Tao, D. (2017). On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379.

[Zaremba et al., 2013] Zaremba, W., Gretton, A., and Blaschko, M. (2013). B-test : A non-parametric, low variance kernel two-sample test. *Advances in neural information processing systems*, 26.

[Zhang and Wang, 2009] Zhang, H. and Wang, M. (2009). Search for the smallest random forest. *Statistics and its Interface*, 2(3) :381.

[Zhang et al., 2008] Zhang, K., Tsang, I. W., and Kwok, J. T. (2008). Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239.

[Zhang, 2011] Zhang, T. (2011). Sparse recovery with orthogonal matching pursuit under rip. *IEEE transactions on information theory*, 57(9) :6215–6221.

[Zhang and Wang, 2020] Zhang, T. and Wang, S. (2020). Nyström kernel algorithm under generalized maximum correntropy criterion. *IEEE Signal Processing Letters*, 27 :1535–1539.

[Zhao and Meng, 2015] Zhao, J. and Meng, D. (2015). Fastmmd : Ensemble of circular discrepancy for efficient two-sample test. *Neural computation*, 27(6) :1345–1372.