Apprentissage multi-label extrême : Comparaisons d'approches et nouvelles propositions

Wissam Siblini

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE NANTES

THESE DE DOCTORAT DE

L'UNIVERSITE DE NANTES

COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par **Wissam SIBLINI**

Apprentissage multi-label extrême

Comparaisons d'approches et nouvelles propositions

Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N – équipe Duke)
Thèse N° : 150522

Soutenue le 23-11-2018

Encadrants :

Dir. de thèse :	Pascale Kuntz	Professeure des universités, Université de Nantes
Encadrant CIFRE :	Frank Meyer	Docteur Ingénieur R&D, Orange Labs Lannion

Composition du Jury :

Présidente :	Elisa Fromont	Professeure des universités, Université de Rennes 1

Rapporteurs avant soutenance :

Stéphane Canu	Professeur des universités, INSA de Rouen
Amaury Habrard	Professeur des universités, Université Jean Monnet de Saint-Etienne

Examinateur :

Jean-Michel Poggi	Professeur des universités, Université Paris Descartes

Table des matières

Table des figures

Liste des tableaux

Notations mathématiques

Notation	Définition
$\mathcal{T}$	ensemble d'apprentissage
n	nombre d'instances d'apprentissage
$\mathcal{S}$	ensemble de test
$n_{\mathcal{S}}$	nombre d'instances de test
$\mathcal{F}$	ensemble des attributs
x	vecteur d'attributs
X	matrice d'attributs
x_{ij} ou X_{ij}	composante (i, j) de la matrice d'attributs
d_x	nombre d'attributs
d'_x	nombre d'attributs réduits
s_x	nombre moyen d'attributs non nuls
$\mathcal{L}$	ensemble des labels
y	vecteur de labels
Y	matrice de labels
y_{ij} ou Y_{ij}	composante (i, j) de la matrice de labels
d_y	nombre de labels
d'_y	nombre de labels réduits
s_y	nombre moyen de labels non nuls

Tableau 1 – Résumé des notations mathématiques majeures. Ces notations sont majoritairement introduites dans le premier chapitre.

Chapitre 1

Introduction

Sommaire

A la fin du XXème siècle, une part importante des données des entreprises et des autres institutions était difficilement accessible via des bases numérisées. La digitalisation au cours de la dernière décennie a fortement accéléré l'augmentation des volumes des systèmes d'information, et en 2013, des études ont montré que 90% des données disponibles sur terre avaient été produites numériquement pendant les 24 derniers mois [4]. Ces dernières abritent une grande diversité d'ingrédients exploitables pour le développement d'applications variées : traduction automatique [5], recommandation [6], analyse automatique d'image [7], recherche d'information dans des corpus de textes [8], prédiction de maladie [9], etc... Leur traitement, qui contribue à l'essor de la discipline de la Science des Données, est devenu un enjeu majeur pour la société [10][11][12]. Ce domaine de recherche très attractif s'intéresse à une très grande diversité de problèmes qui se structurent autour de plusieurs spécialités dont notamment la fouille de données, la visualisation de données et l'apprentissage. Cette thèse s'inscrit en apprentissage automatique ("Machine Learning") où il s'agit d'apprendre des modèles sur des données pour réaliser une tâche de prédiction (prédire les besoins du client [13], déterminer automatiquement des mots clés pour qualifier un texte [14], évaluer des risques de fraudes [15], etc...). Et, plus précisément, elle porte sur la classification multi-label à très large échelle (ou "Apprentissage Multi-Label Extrême") qui vise à associer des objets comme des musiques, des textes ou des images à quelques labels, parfois appelés étiquettes ou classes, parmi des millions de possibilités.

1 Apprentissage multi-label et données à grande échelle

En apprentissage, les problèmes rencontrés s'inscrivent dans différents paradigmes. Le plus étudié est certainement l'apprentissage supervisé monolabel [16]. Son objectif est d'associer un objet, décrit par un vecteur de variables attributs, à un unique concept d'intérêt (variable cible) que l'on qualifie ici de label. Par exemple, on peut souhaiter concevoir un système apprenant qui permet de déterminer si un mail décrit par les mots qu'il contient (attributs) est un spam (unique label, qui peut être "vrai" ou "faux") [17]. Pour couvrir une famille plus large de problèmes, le paradigme de la classification multiclasse a été developpé : il permet de traiter des cas où la variable de sortie a plus de 2 modalités. Dans ce cas, chaque objet est associé à une classe parmi un ensemble de $d_y > 2$ classes possibles. Par exemple, il permet de déterminer, pour des images de textes manuscrits décrites par des pixels (attributs), les chiffres qui y sont écrits et la variable de sortie prend donc une unique valeur parmi "1","2",...,"10" [18]. Cependant, pour certaines applications, les objets doivent intrinsèquement être décrits par plusieurs labels. Par exemple, en annotation de textes, on peut souhaiter construire un système qui qualifie un texte à la fois de *drôle*, *traitant de sport* et *en anglais*. On peut également, dans une bibliothèque musicale, souhaiter annoter une musique comme étant à la fois *triste* et *classique*. Au cours des dernières décennies, de nombreuses applications (analyse de sons/musique [19] [20], vision par ordinateur [1][21], analyse de textes [22][23], biologie et santé [24][25], systèmes de recommandation [26][27]) ont contribué à orienter la recherche vers la classification multi-label qui permet d'associer un objet à un ou plusieurs labels parmi un ensemble de possibilités prédéfinies (Figure 1.1).

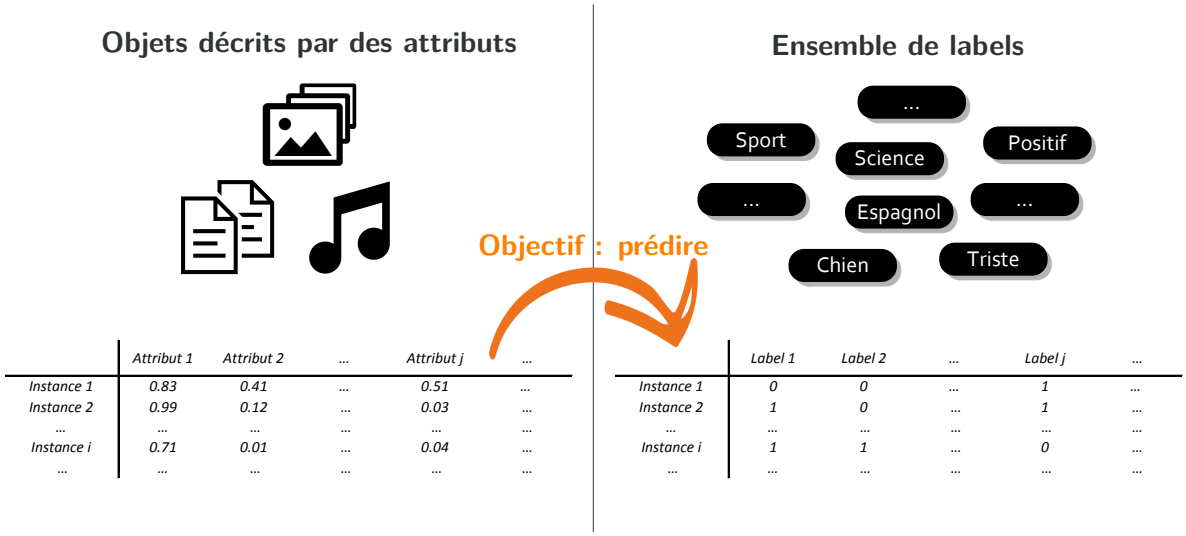

Figure 1.1 – Principe de la classification multi-label.

Depuis les premiers travaux de Boutell and al. [28], Zhang et al. [29] et Tsoumakas et al. [30], de nombreux algorithmes ont été proposés dans ce cadre et ceux-ci ont depuis été répertoriés dans plusieurs états de l'art [30][31][32][33][34]. Ils y sont généralement catégorisés en trois grandes familles : (i) les approches d'apprentissage par transformation qui divisent le problème multi-label en plusieurs problèmes monolabel, (ii) les approches d'apprentissage par adaptation qui adaptent des algorithmes monolabel pour qu'ils puissent traiter des données multi-label et (iii) les approches d'apprentissage ensemble qui effectuent des prédictions multi-label à partir d'une agrégation ("bagging" ou "boosting" [35]) des prédictions d'une collection de modèles simples. Cette effervescence dans la recherche a permis d'améliorer significativement la qualité des résultats prédictifs sur des données de bancs d'expériences couramment utilisés dans la littérature. Mais depuis les premiers travaux la dimension des données a fortement augmenté et la plupart des algorithmes pionniers ne résistent pas à ces nouvelles échelles. Cette évolution a donc récemment conduit au problème d'apprentissage multi-label extrême, régulièrement noté XML (eXtreme Multi-label Learning) dans la suite du manuscrit, qui considère des problèmes où le nombre de variables est extrêmement grand [36][37][38]. Dans ce cadre, les données analysées sont caractérisées par un nombre très important d'objets et requièrent la capacité de traiter un très grand nombre de variables attributs en entrée et de variables labels en sortie (de l'ordre de 10^4 à 10^7). Par exemple, dans le récent challenge Kaggle portant sur la catégorisation automatique d'articles de Wikipédia (Figure 1.2), le corpus était composé d'environ 2.4 millions d'articles. A partir du contenu des articles, encodés comme des sacs de n-grams avec 1.6 millions de variables attributs, le but était de déterminer automatiquement la ou les catégories Wikipédia associées parmi environ 350 000 possibilités.

En sus des données en très grande dimension dès l'origine de la collecte, il existe des problèmes multi-label structurés (traitement de séquences, problème avec contexte) sur lesquels les pré-traitements standards (utilisation d'un vocabulaire avec des n-grams, concaténation) conduisent à l'explosion des dimensions à traiter.

2 Caractéristiques du problème : des limites des approches standards aux nouvelles voies de recherche

Dans la suite, on considère un ensemble de labels $\mathcal{L}$ (resp. d'attributs $\mathcal{F}$) de cardinal d_y (resp. d_x). On note $\mathcal{T}$ l'ensemble d'apprentissage composé des n exemples/items annotés décrits par des variables attributs et associés à quelques labels de $\mathcal{L}$. Plus précisément, chaque exemple est décrit par un vecteur d'attributs $x_i \in \mathbb{R}^{d_x}$ dont chacune des composantes contient la valeur d'une des variables attributs de $\mathcal{F}$ et est associé à un vecteur binaire de labels $y_i \in \{0,1\}^{d_y}$ où chaque composante correspond à un label de

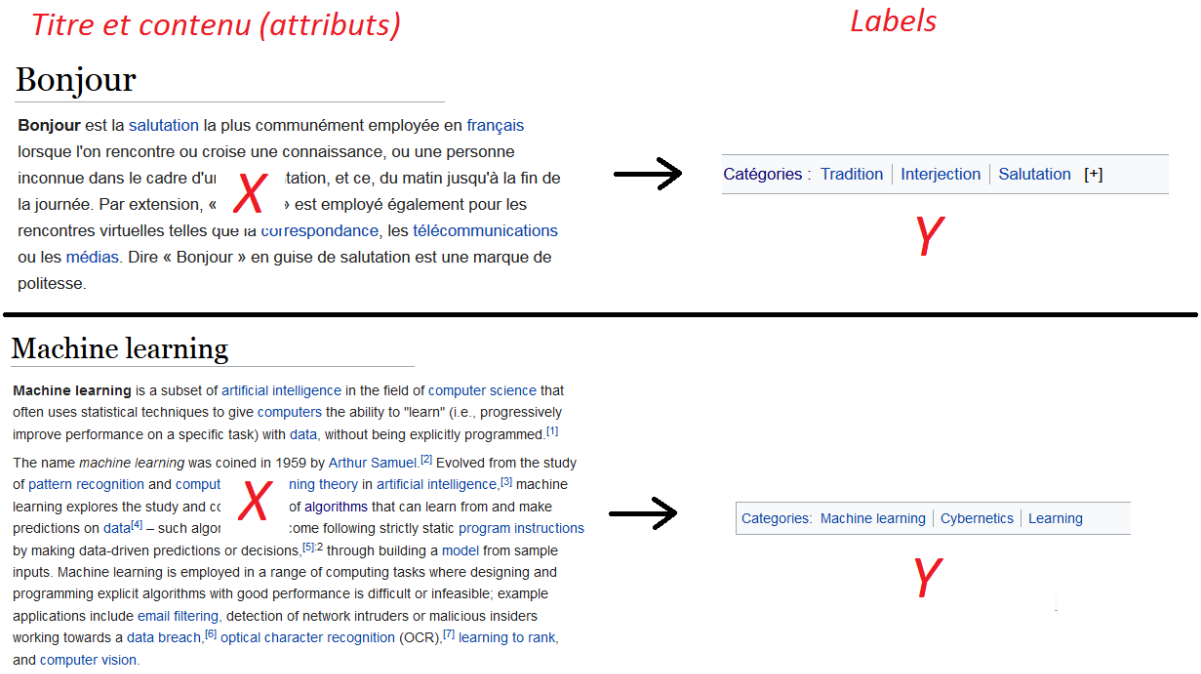

Figure 1.2 – Illustration du jeu de données WikiLSHTC.

$\mathcal{L}$ et vaut 1 si le label est associé à l'item. On note $X \in \mathbb{R}^{n \times d_x}$ (resp. $Y \in \{0,1\}^{n \times d_y}$) la matrice des attributs (resp. labels) de l'ensemble d'apprentissage dont chaque ligne correspond au vecteur d'attributs (resp. de labels) d'une instance. Le but de l'apprentissage multi-label est d'apprendre, à partir de l'ensemble d'apprentissage, un modèle qui permet de prédire les labels y d'une instance à partir de ses attributs x ; l'apprentissage multi-label est qualifié d'extrême lorsque d_x et d_y sont grands (de 10^4 à 10^7). Les données sont souvent creuses et on note s_x (resp. s_y) le nombre moyen d'attributs (resp. labels) non nuls par instance.

Pour le problème d'apprentissage multi-label extrême, un banc d'expériences standard avec des données publiques est disponible pour évaluer les algorithmes [1]. Les données proviennent de domaines variés : la catégorisation de produits pour le commerce en ligne, le ciblage publicitaire sur un moteur de recherche, la catégorisation d'articles Wikipedia, la classification d'images, la recommandation d'objets à très grande échelle. Ces données ont des centaines de milliers voire des millions de labels (Tableau 1.1). Les jeux Mediamill, Bibtex et Delicious font partie de la litterature multi-label classique et leurs dimensions sont petites par rapport aux autres jeux de données mais ils sont néanmoins souvent utilisés en apprentissage extrême car ils permettent de vérifier que les algorithmes proposés restent également performants en petite dimension.

Il est difficile d'apprendre sur les données extrêmes pour deux raisons principales :

1. Voir repository XML : `http://manikvarma.org/downloads/XC/XMLRepository.html`

Tableau 1.1 – Caractéristiques des jeux de données de l'apprentissage multi-label extrême - tableau extrait du repository XML.

Dataset	d_x	d_y	n	$n_{\mathcal{S}}$	Avg. Points per Label	s_y
Mediamill	120	101	30993	12914	1902.15	4.38
Bibtex	1836	159	4880	2515	111.71	2.40
Delicious	500	983	12920	3185	311.61	19.03
RCV1-2K	47236	2456	623847	155962	1218.56	4.79
EURLex-4K	5000	3993	15539	3809	25.73	5.31
AmazonCat-13K	203882	13330	1186239	306782	448.57	5.04
AmazonCat-14K	597540	14588	4398050	1099725	1330.1	3.53
Wiki10-31K	101938	30938	14146	6616	8.52	18.64
Delicious-200K	782585	205443	196606	100095	72.29	75.54
WikiLSHTC-325K	1617899	325056	1778351	587084	17.46	3.19
Wikipedia-500K	2381304	501070	1813391	783743	24.75	4.77
Amazon-670K	135909	670091	490449	153025	3.99	5.45
Ads-1M	164592	1082898	3917928	1563137	7.07	1.95
Amazon-3M	337067	2812281	1717899	742507	31.64	36.17
Ads-9M	2082698	8838461	70455530	22629136	14.32	1.79

leurs dimensions et les propriétés de leurs distributions.

Un problème de dimension Sur les jeux de données XML, les trois dimensions d_x, N et d_y atteignent le million. Par conséquent, un algorithme avec une complexité temporelle/spatiale quadratique sur les dimensions (par exemple : d_x^2, $n \times d_x$, ...) ne peut pas être entraîné avec un ordinateur standard. Comme le problème de classification multi-label classique n'impose que de faibles contraintes sur la complexité, la majeure partie des modèles proposés dans le cadre initial sont incapables de passer à l'échelle. Prenons, à titre d'exemple, une régression logistique one-vs-rest (un modèle par label) entraînée avec une descente de gradient stochastique. Cette approche a une complexité mémoire en $O(d_x \times d_y)$ et une complexité temporelle en $O(n \times d_y \times s_x)$. Bien que ces complexités soient relativement faibles par rapport à une grande partie des algorithmes multi-label, sur WikiLSHTC-325K, un tel modèle stocké en mémoire avec le type "float" occuperait 1.89 Teraoctets de RAM et nécessiterait plusieurs années pour l'apprentissage. Les modèles les plus performants [34][39] de la littérature multi-label classique, RF-PCT [40], HOMER [41] et ML-kNN [29], sont également pleinement concernés par le problème de passage à l'échelle.

Un problème de propriétés Malgré la dimension élevée des données, il reste possible d'apprendre directement quelques classifieurs simples comme le bayésien naïf ou la baseline "labels fréquents" mais leurs performances sur les données XML sont mauvaises (voir [36] et Tableau 1.2).

Tableau 1.2 – Comparaison des performances (P@1, P@3, P@5) entre la baseline "label fréquents" et l'algorithme multi-label extrême le plus populaire FastXML.

	AmazonCat-13K			Wiki10-31K			Delicious-200K		
	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
FastXML	0.9310	0.7818	0.6338	0.8295	0.6756	0.5770	0.4320	0.3868	0.3621
Labels fréquents	0.2988	0.1878	0.1486	0.8079	0.5050	0.3675	0.3873	0.3675	0.3552
	WikiLSHTC-325K			Wikipedia-500K			Amazon-670K		
	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
FastXML	0.4975	0.3310	0.2445	0.4934	0.3351	0.2586	0.3697	0.3332	0.3053
Labels fréquents	0.1588	0.0603	0.0380	0.1529	0.0583	0.0368	0.0028	0.0027	0.0023

La difficulté d'apprentissage provient de caractéristiques communes aux données considérées : elles sont redondantes, bruitées, partiellement pertinentes (certaines ne sont pas corrélées aux labels), incomplètes puisque certains labels pertinents ne sont pas renseignés [38]. De plus, la distribution des labels est à traîne épaisse (Figure 1.3) et les algorithmes qui négligent les labels de la longue traîne ont des performances très limitées. Enfin, de nombreux modèles peuvent être exposés à des risques de sur-apprentissage car beaucoup de variables sont peu observées et il y a autant, voire plus, de variables que d'exemples dans les jeux de données.

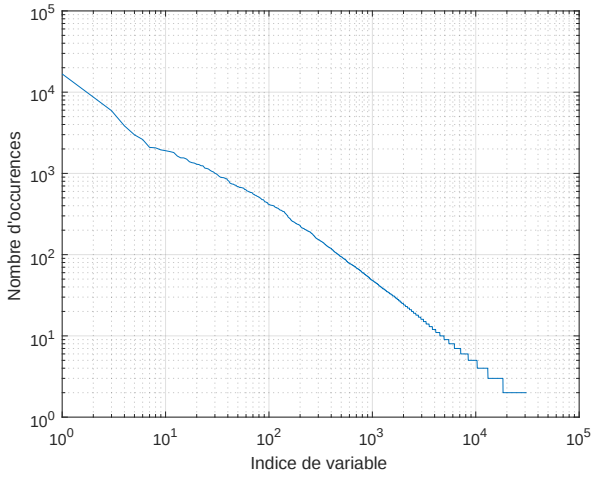

Figure 1.3 – Exemple de distribution ("zipfienne") des labels dans un jeu de données extrême (Wiki10-31K) représentée sur une échelle logarithmique.

Une lueur d'espoir L'objectif général de la thèse est d'obtenir une bonne qualité prédictive pour des problèmes XML sans avoir recours à des super calculateurs (challenge de temps et de mémoire). Malgré les difficultés que l'on vient d'évoquer sur les données XML, certaines caractéristiques du problème ouvrent des possibilités intéressantes. En effet, les données sont très creuses (Tableau 1.1) et les corrélations entre variables sont parcimonieuses puisque chaque label n'est corrélé qu'à quelques attributs. La prise

en compte conjointe de ces deux propriétés structurelles majeures ouvre une voie prometteuse pour le développement d'astuces algorithmiques permettant de contourner la contrainte de dimension.

3 Contenu du manuscrit et contributions

Complété par cette introduction, une conclusion, et des annexes qui présentent des résultats complémentaires et les données multi-label utilisées, ce manuscrit est structuré en 5 chapitres.

Le **chapitre 2** dresse un état de l'art des solutions existantes pour l'apprentissage multi-label extrême. Il est structuré selon les trois types d'approches qui apparaissent naturellement dans la littérature : les approches exploitant la réduction de dimension, les approches exploitant des structures arborescentes, et les approches exploitant des astuces d'optimisation et d'implémentation. Une attention particulière a été portée à la structuration de la présentation des approches basées sur la réduction de dimension. De par leur historique, bien antérieur à l'essor du multi-label, de nombreuses adaptations variées ont été exploitées. Pour faciliter une vision macroscopique de l'existant, nous proposons une typologie basée sur quatre facteurs : l'espace réduit, les fonctions objectifs utilisées, les types de transformation mis en oeuvre pour la réduction, et les contraintes ajoutées pour guider la résolution.

Les chapitres 3 à 5 décrivent ensuite nos contributions méthodologiques et algorithmiques.

Le **chapitre 3** prolonge la partie de l'état de l'art consacrée à la réduction de dimension. Il propose deux formulations génériques qui recouvrent une grande partie des problèmes de réduction explorés en apprentissage multi-label. Ces formulations permettent de mettre en lumière les différents ingrédients associés aux problèmes et facilitent l'identification des similitudes/différences entre les approches. En complément, nous conduisons une méta-analyse originale sur les résultats expérimentaux publiés dans plus de 25 articles de la littérature. Les comparaisons par paires significatives (l'algorithme A_i est meilleur que l'algorithme A_j dans l'article X avec une significativité statistique α) sont représentées par un multigraphe orienté qui permet d'extraire, par une démarche inspirée des travaux sur le consensus en agrégation de préférences, les algorithmes dominants eu égard aux résultats existants. Cette analyse approfondie de l'état de l'art nous a conduits à proposer en dernière partie du chapitre une nouvelle approche d'apprentissage multi-label basée sur un couplage entre le processus de réduction de dimension et le processus de classification. Les performances expérimentales obtenues sur les jeux de données standards sont très compétitives avec celles de l'état de l'art mais le passage à l'échelle (au-delà de 10^4 variables) reste délicat. Une petite étude complémentaire avec un perceptron multicouche nous permet de confirmer deux limites inhérentes aux approches basées sur la

réduction explicite de dimension pour le XML : (i) la complexité du problème initial est transférée sur le problème de réduction et (ii) l'approximation de rang faible entraîne une perte d'informations importante qui dégrade les performances du classifieur. Ces verrous peuvent, par exemple, être levés en combinant la réduction de dimension avec un modèle de type régression one-vs-rest ou en exploitant une arborescence. Ces deux solutions qui font elles-mêmes face à des problématiques de complexité en temps et en mémoire, sont explorées dans les chapitres 4 et 5.

Le **chapitre 4** est consacré à la proposition d'une stratégie permettant de réduire la complexité en mémoire qui est un enjeu important pour l'application du XML sur des calculateurs standards. En s'inspirant de travaux sur la gestion des flux de données, nous proposons une approche pour estimer les plus grands paramètres utiles d'un classifieur pendant son apprentissage. La stratégie consiste à stocker approximativement les paramètres du modèle dans une structure de type "count-sketch" au lieu de les stocker intégralement dans la mémoire. Nos expérimentations sur un problème de régression one-vs-rest appris avec une descente de gradient stochastique montre que cette stratégie obtient les mêmes performances que lorsque le modèle stocke tous les paramètres tout en réduisant très significativement la consommation mémoire (de 80% à 99.9% sur neuf jeux tests). L'analyse expérimentale est complétée par l'établissement d'une borne théorique sur l'erreur d'approximation des paramètres et par l'étude du cas particulier de la distribution des paramètres selon une loi de Zipf qui est fréquemment observée.

Le **chapitre 5** vise à considérer simultanément les contraintes de complexité mémoire et temporelle. Quelques propriétés identifiées sur les travaux pionniers des méthodes hiérarchiques en multi-label complétées par des résultats expérimentaux récents en XML nous ont incités à explorer plus en profondeur les forêts aléatoires d'arbres de décision. Nous proposons une nouvelle approche CRAFTML (Clustering-based RAndom Forest of predictive Trees for extreme Multi-label Learning) basée sur une forêt d'arbres de décision entraînés avec la supervision des labels où les conditions de séparations des instances à chaque noeud sont multivariées. L'originalité principale de CRAFTML eu égard à l'état de l'art est double : (i) les sélections aléatoires des attributs sont remplacées par des projections aléatoires appliquées sur l'espace des attributs et sur l'espace des labels ; et (ii) la stratégie de séparation, différente de celle des autres arbres, a une très faible complexité. Les résultats expérimentaux sur les jeux de données XML sont très prometteurs. Les performances de CRAFTML sont meilleures que celles des autres approches arborescentes avec des temps d'apprentissage et une consommation mémoire plus faible. De plus, il reste compétitif avec les meilleurs méthodes actuelles (DISMEC [42] et PPDSparse [2]) qui fonctionnent sur des super-ordinateurs (100 coeurs). Une implémentation parallélisée sur une machine à 5 coeurs montre que le temps d'apprentissage de CRAFTML devient inférieur à celui de DISMEC et proche de PPDSparse pour le plus grand jeu de données Amazon670k. Au-delà de ses performances expérimentales, CRAFTML présente un

avantage structurel : la taille de son modèle est plus petite et ses complexités temporelles d'apprentissage et de prédiction sont plus faibles. Il présente donc un bon compromis entre précision, ressources de calcul nécessaires et vitesse d'exécution.

Le **chapitre 6** est consacré aux applications auxquelles nous avons contribué dans le cadre industriel de cette thèse CIFRE effectuée avec l'entreprise Orange. Nous présentons un outil interactif de classification multi-label (VIPE) utilisé pour faire de l'analyse d'opinions. Cet outil permet à un utilisateur d'importer des textes courts (tweets, mails, enquêtes, ...), de définir des labels d'intérêts (« client globalement satisfait », « évoque la rapidité du débit »,...) et de proposer pour chaque texte des recommandations de labels et pour chaque label des recommandations de textes. La version actuelle est basée sur une factorisation de matrices mais l'intégration de CRAFTML est en cours d'étude. Le prototype expérimental de CRAFTML a aussi donné lieu à l'implémentation d'une API qui est actuellement utilisée pour des expérimentations exploratoires sur des tâches de compréhension de textes et pour une adaptation au cadre multi-classe.

La figure 1.4 résume la structure de l'état de l'art, l'organisation de ce manuscrit et les contributions de la thèse.

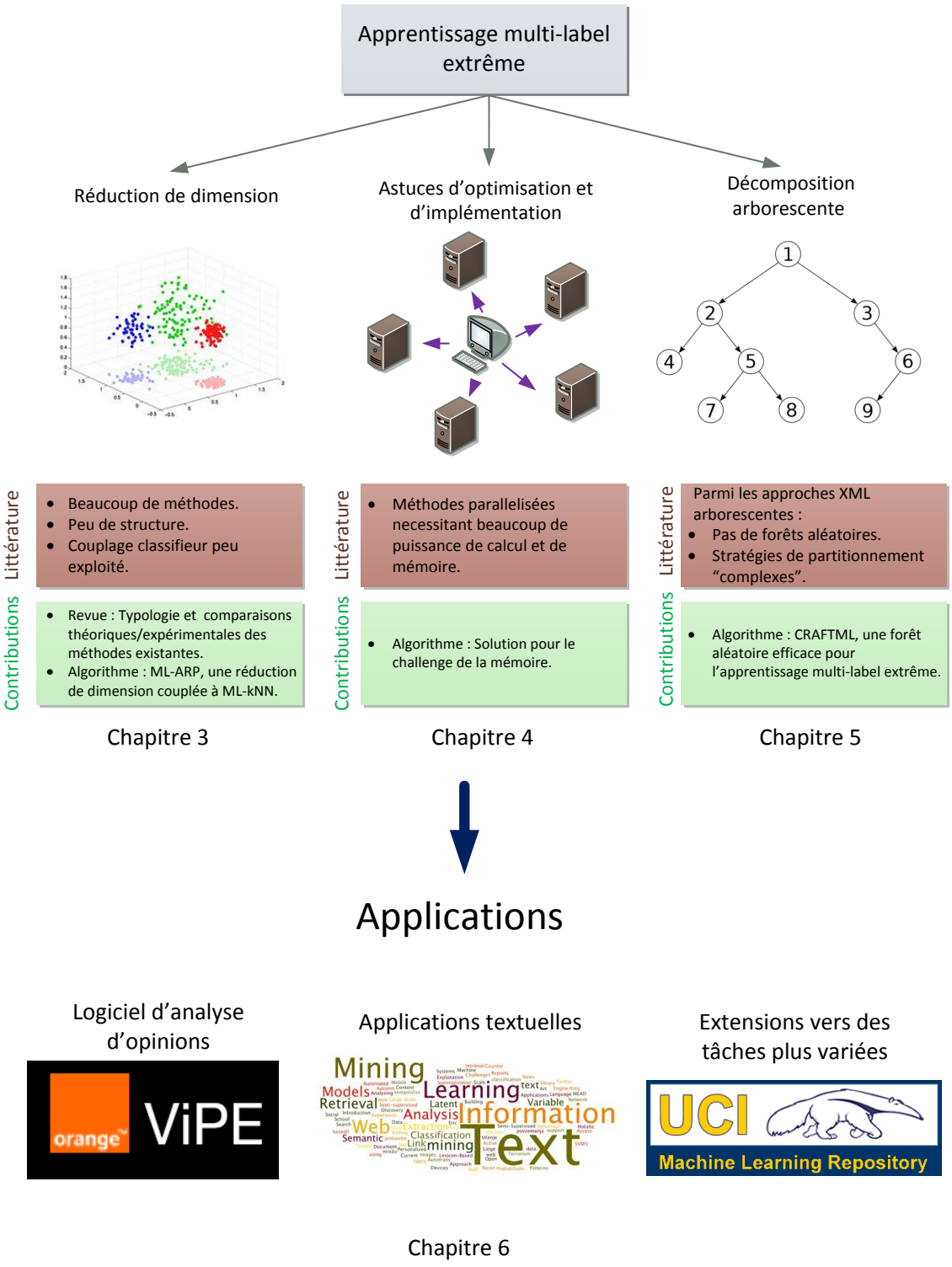

Figure 1.4 – Arbre des solutions en apprentissage multi-label extrême et contributions de la thèse.

Chapitre 2

État de l'art : apprentissage multi-label extrême

Sommaire

1 Introduction

Trois directions majeures ont été explorées dans la littérature pour passer à l'échelle et prédire avec précision dans un cadre multi-label extrême :

1. réduire la dimension des données (en particulier le nombre de variables) pour obtenir un problème latent plus "petit" et donc soluble avec les algorithmes multi-label standards [43, 44].
2. partitionner hiérarchiquement le problème en sous problèmes de petite dimension résolus avec des classifieurs locaux [45, 38]. Cette stratégie concerne les méthodes arborescentes qui construisent récursivement des sous-ensembles d'instances ou de labels permettant de simplifier et d'accélérer la résolution du problème.
3. permettre à des méthodes trop complexes pour le cadre extrême (par exemple la regréssion one-vs-rest) d'apprendre dans un délai raisonnable ou avec une mémoire limitée grâce à des astuces d'optimisation (par exemple la conversion primal/dual

du problème [46]) ou d'implémentation (par exemple l'exploitation de la parcimonie ou la parallélisation sur des serveurs de calculs [42, 2]).

Dans ce chapitre, nous explorons l'état de l'art pour ces différentes stratégies. La **section 2** est consacrée à la première mais elle ne se restreint pas aux méthodes de réduction de dimension multi-label qui ont été spécifiquement développées pour un cadre extrême ; elle va au-delà d'un état de l'art classique, souvent basé sur une liste organisée des méthodes existantes, en structurant la présentation selon une typologie des différentes approches. La typologie est construite à partir des composants principaux qui définissent la nature du problème de réduction et la façon de le résoudre : (i) le choix de l'espace réduit (espace des attributs, espace des labels ou les deux), (ii) l'indépendance/dépendance entre la réduction d'un des espaces et l'information contenue dans l'autre (iii) les types de transformation de réduction, et (iv) les fonctions de régularisation ainsi que l'ensemble des contraintes appliquées au problème de réduction. On présente ensuite avec plus de détails les méthodes adaptées à l'apprentissage multi-label extrême. La **section 3** présente les principes généraux de la stratégie arborescente (construction des partitions, exploitation des partitions pour la résolution du problème initial), puis les méthodes arborescentes développées dans le cadre de l'apprentissage multi-label extrême. Enfin, la **section 4** présente les méthodes qui s'inscrivent dans le troisième axe de solutions : les astuces d'optimisation et d'implémentation.

Notons que pour des raisons d'homogénéité du fait de la variabilité des traductions, et aussi d'identification dans la littérature, nous avons choisi de conserver les noms anglo-saxons des algorithmes.

2 Solution 1 : méthodes de réduction de dimension

La réduction de dimension a une longue histoire en science des données [47] [48] qui a été notamment motivée par des applications variées : la visualisation de données et l'interprétation [49], la compression [50] et le débruitage des données [51]. D'une manière générale, appliquer la réduction de dimension sur des données brutes produit une représentation synthétique qui fait ressortir des liens et des structures cachés dans la masse qui permettent de guider les algorithmes d'apprentissage [52][53]. Le très grand nombre de variables en apprentissage multi-label extrême a renouvelé l'intérêt de la réduction de dimension. Se présentant comme une solution prometteuse pour traiter des données massives et bruitées, elle a fait l'objet d'un grand nombre de publications au cours des deux dernières décennies, entraînant diverses propositions de méthodes pour réduire le cardinal de l'espace des attributs, des labels ou des deux afin de permettre le passage à l'échelle des classifieurs et d'améliorer leur performances prédictives. Il existe plus de cinquante méthodes dans le cadre multi-label mais toutes ne sont pas nécessairement adaptées à l'apprentissage multi-label extrême. Les étudier dans leur globalité permet néanmoins de

se rendre compte de toute la variété des stratégies possibles pour réduire la dimension. A notre connaissance, seul un état de l'art a déjà été publié il y a cinq ans, mais il n'explore pas la vaste gamme d'approches existantes car il met l'accent sur PCA, CCA et PLS essentiellement. Il n'offre pas non plus un cadre global permettant de les comparer. Pour palier à ces limitations, nous présentons un état de l'art global des approches de réduction de dimension multi-label en insistant sur la typologie des méthodes, qui est ensuite plus détaillé pour les quelques méthodes adaptées à l'apprentissage multi-label extrême.

2.1 Typologie des méthodes de réduction de dimension multi-label

La grande majorité des approches d'apprentissage multi-label basées sur la réduction de dimension suivent un processus en deux étapes : (1) réduction de X ou Y ou les deux, (2) prédiction des labels à partir des espaces réduits avec un classifieur. Dans la plupart des cas, la réduction de dimension est appliquée en tant que pré-traitement sur les données indépendamment de l'apprentissage qui suit, mais des recherches récentes stimulent l'exploration du couplage entre réduction et apprentissage [54]. Quelle que soit la stratégie, l'impact de la réduction sur les performances du classifieur est *in fine* évalué par la qualité de la prédiction des labels qui se traduit sous la forme de nombreuses mesures qui ont été proposées dans la littérature [31][34]. Par conséquent, trois éléments sont considérés dans le problème de réduction de dimension : la fonction objectif f_d pour la réduction de dimension indépendante ou dépendante du classifieur, la fonction objectif f_c associée au classifieur et la mesure de qualité de prédiction finale m_q.

Enfin, le choix de l'espace à réduire dépend de la nature du problème et détermine la manière de le résoudre. Notons la réduction de X (resp. Y) par la matrice X' (resp. Y') de dimension $n \times d'_x$ (resp. $n \times d'_y$) où d'_x (resp. d'_y) est la dimension de l'espace réduit. Dans la suite, par abus de langage, on confondra régulièrement espace des attributs (resp. labels) avec matrice des attributs (resp. labels). En pratique, les valeurs de d'_x et d'_y sont souvent fixées *a priori* (100 et 500 sont des valeurs couramment utilisées [55][44]) mais différentes stratégies classiques peuvent être utilisées pour guider leur choix en particulier lorsque la méthode de réduction passe par une décomposition en valeurs propres. Par exemple d'_x et/ou d'_y dépendent du nombre de valeurs propres dont les valeurs dépassent un seuil fixé, ou dont la somme dépasse un pourcentage fixé de la somme totale des valeurs propres.

Il existe trois stratégies différentes pour aborder le problème de réduction de dimension pour l'apprentissage multi-label (figure 2.1) : (i) réduire la matrice des attributs X en X' et prédire la matrice des labels Y à partir de la matrice des attributs réduits X', (ii) réduire la matrice des labels Y en Y' et prédire la matrice des labels réduits Y' à partir de la matrice des attributs X, (iii) réduire à la fois les labels et les attributs en X' et Y' et prédire la matrice des labels réduits Y' à partir de la matrice des attributs réduits X'.

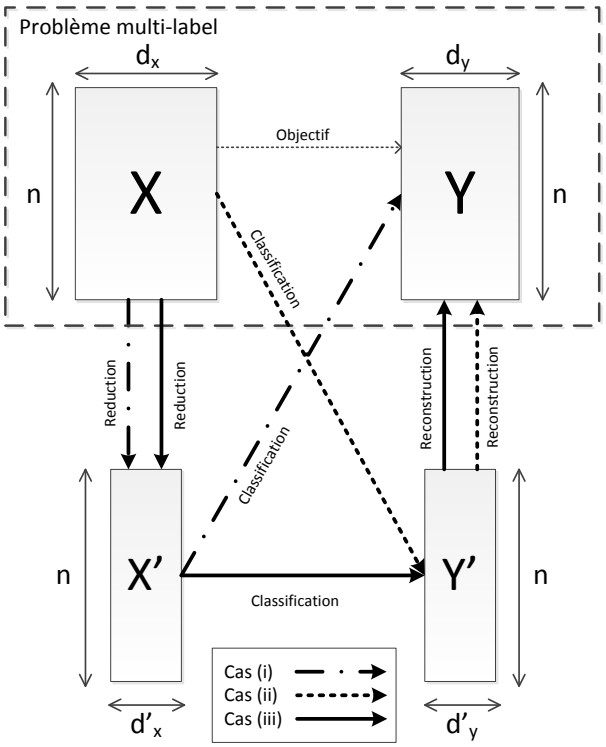

Figure 2.1 – Vue d'ensemble de l'intégration de la stratégie de réduction de dimension dans le processus d'apprentissage multi-label.

Pour chacun de ces cas, le problème de réduction de dimension peut être défini comme un problème d'optimisation :

$$\begin{aligned} &\underset{U,V}{\text{optimise}} && f_d(U, V, X, Y) + r(U, V) \\ &\text{avec} && c(U, V, X, Y) \end{aligned} \tag{2.1}$$

où :

— U et V sont les paramètres d'une fonction de transformation/réduction appliquée sur X et Y (par exemple des matrices de projection P_x et P_y dans le cas d'une transformation linéaire) ou directement les matrices réduites X' et Y'. Quand une méthode réduit seulement un espace (X ou Y), le problème est défini avec un seul paramètre (U ou V).

— f_d est la fonction objectif de réduction qui est indépendante ou dépendante du classifieur.

— r est une fonction de régularisation souvent associée à une norme (L_1, L_2, L_1L_2) appliquée sur les paramètres pour limiter les phénomènes de sur-apprentissage et pour simplifier le modèle.

— c est un ensemble de contraintes sur l'espace de solutions. Certaines approches n'introduisent pas de contraintes mais la plupart d'entre elles le font pour réduire

les degrés de liberté du problème et en faciliter la résolution.

Dans les sous-sections qui suivent nous détaillons les différents ingrédients du problème (2.1). Nous présentons d'abord les fonctions objectif f_d qui sont indépendantes des classifieurs (cas le plus fréquent) et nous les ordonnons en fonction de l'espace ciblé pour la réduction de dimension. Ensuite, nous discutons des différents cas où l'objectif de réduction de dimension est couplé à l'objectif d'apprentissage. Pour chaque cas, un exemple de la littérature est donné à titre d'illustration et nous renvoyons au tableau 2.2 pour montrer à quelles stratégies sont rattachées les autres méthodes de réduction. De plus, les tableaux 1a et 1b résument l'idée principale, en une phrase, de toutes les méthodes analysées. Nous terminons par une présentation synthétique des fonctions de régularisation ainsi que des contraintes usuellement appliquées par les méthodes de réduction.

Notons que nous ne considérons pas, dans cet état de l'art, les méthodes qui réduisent la dimension avec la sélection de variables (voir [56] pour une revue récente). Appliquée à des problèmes réels, la sélection de variable permet l'amélioration des performances de nombreux classifieurs [57][58][59]. Néanmoins, cette approche est souvent insuffisante car elle considère uniquement l'importance relative des variables. Or, il est parfois nécessaire d'utiliser des transformations plus expressives sur les données pour pouvoir retranscrire des équivalences et de la redondance entre des ensembles d'attributs. Par exemple, pour des applications textuelles (voir [60][61][62] et notes du tutoriel EMNLP 2014 de J. Weston et A. Bordes), on souhaite non seulement réduire la dimension pour retranscrire l'importance des différents mots mais également traduire des redondances/similarités sémantiques comme les synonymes. On ne peut pas réaliser une telle tâche avec la sélection de variables uniquement car elle ne permet pas de rapprocher deux instances semantiquement similaires si elles utilisent des mots complètement différents.

Index	Item
1	A feline rests in my garden
2	The cat sleeps on the grass

Bag-of-words

	Do	...	cat	on	grass	rests	feline	a	sleeps	...	in	garden	the	my
1	0	...	0	0	0	1	1	1	0	...	1	1	0	1
2	0	...	1	1	1	0	0	0	1	...	0	0	1	0

Figure 2.2 – Représentation en sac-de-mots de deux phrases sémantiquement proches

Dans l'exemple de la figure 2.2, les deux phrases sont sémantiquement proches mais leur codage en sac-de-mots est complètement disjoint (aucune similarité, utilisation de variables différentes). La sélection de certaines colonnes/variables/mots ne pourrait donc pas améliorer leur similarité. Dans ce cas il semble nécessaire d'effectuer des transforma-

tions plus complexes [55]. Toutefois, si les méthodes décrites ci-après sont plus générales, elles n'écartent évidemment pas la possibilité d'effectuer de la sélection de variables car cela correspond à l'apprentissage du cas particulier d'une projection diagonale.

2.1.1 Fonctions objectif indépendantes du classifieur

Nous présentons ici les méthodes de réduction de dimension dont la fonction objectif est indépendante du classifieur. Elles sont groupées selon l'espace qu'elles réduisent (X, Y, à la fois X et Y).

Réduction de l'espace des attributs (X) Les méthodes de réduction de l'espace des attributs transforment l'espace initial de grande dimension X en un espace réduit X' dans le but d'obtenir des données compressées qui présentent les informations essentielles à la tâche souhaitée. Comme les attributs initiaux sont partiellement bruités, redondants et/ou non pertinents, cela permet entre autre de corriger ces défauts d'origine [63]. Selon les méthodes, la fonction objectif f_d pour la réduction des attributs est soit indépendante soit dépendante de l'information portée par les labels.

Les objectifs indépendants des labels peuvent être organisés en trois familles d'objectifs [1] :

1. *Objectif FI1* : maximiser la conservation de la covariance ou des co-occurences entre les attributs (e.g. Principal Component Analysis (PCA) [64]) ;
2. *Objectif FI2* : minimiser une erreur de reconstruction formulée comme une distance entre X et sa reconstruction à partir de X' (e.g. Auto-encoders (AE) [65]) ;
3. *Objectif FI3* : maximiser la conservation des distances entre les instances en passant de l'espace X à l'espace X' (e.g. Locality Preserving Projection (LPP) [66]). Cette conservation peut être globale si on souhaite préserver toutes les distances entre les paires d'instances où locale si, par exemple, on ne souhaite conserver les distances qu'entre les plus proches voisins.

Remarquons que ces objectifs peuvent être étroitement liés ; par exemple, PCA, classé dans FI1, minimise également implicitement l'erreur de reconstruction quadratique entre une projection de X' et X et peut donc également être classé dans FI2. Notons aussi que les méthodes de réduction à base de projections aléatoires [67][68] ne font partie d'aucune de ces familles. Leur objectif portera le code R comme "Random".

Les objectifs dépendants visent à guider la réduction des attributs avec les informations des labels [69][70]. Cela permet de renforcer le lien entre l'espace des attributs réduit X' et l'espace des labels Y. Ils couvrent trois stratégies principales :

1. Chaque famille est codée pour être identifiée facilement dans le tableau 2.2. Par exemple, **FI1** fait référence à la **1**ère famille d'objectifs pour les méthodes de réduction des attributs (en anglais "**F**eature") qui sont **I**ndépendantes des labels)

1. *Objectif FD1* : maximiser le lien X-Y à travers un critère standard (covariance, dépendance de Hilbert-Schmidt) (e.g. Multi-label Dimensionality Reduction via Dependence Maximization MDDM [54]) ;
2. *Objectif FD2* : obtenir la même isométrie entre les instances dans l'espace des attributs réduits X' que dans l'espaces des labels Y (e.g. Hypergraph Spectral Learning HSL [71])
3. *Objectif FD3* : maximiser le lien entre les attributs et les labels en apprenant un sous-espace X' qui peut être utilisé pour reconstruire à la fois X et Y (e.g. Multi-label Latent Semantic Indexing MLSI [72]).

En outre, plusieurs approches hybrides optimisent un compromis paramétré (par exemple θ_1 *objectif FD1* + θ_2 *objectif FD2*) entre les objectifs ci-dessus. C'est par exemple le cas de Maximizing feature Variance and feature-label Dependence simultaneously MVMD [73]).

Réduction de l'espace des labels (Y) Comme certains labels sont corrélés, il semble intuitif de prendre en compte ces corrélations pour améliorer à la fois la qualité et le passage à l'échelle de l'apprentissage [34]. Cela peut être réalisé en réduisant et en transformant l'espace des labels. L'une des premières méthodes de réduction de l'espace des labels était basée sur l'acquisition compressé ("compressed sensing") [50]. Depuis lors, diverses stratégies ont été proposées. Analogiquement à la réduction des attributs, la réduction des labels peut être indépendante ou dépendante de l'information portée par les attributs.

Les objectifs indépendants des attributs peuvent être organisés en trois familles similaires à celles des méthodes de réduction d'attributs indépendantes des labels :

1. *Objectif LI1* : maximiser la conservation de la covariance des labels (par exemple PLST [74], qui est l'équivalent de PCA appliqué à l'espace des labels) ;
2. *Objectif LI2* : minimiser l'erreur de reconstruction formulée par une distance entre Y et Y' (par exemple, Multi-label prediction via compressed sensing CS [50]).
3. *Objectif LI3* : maximiser la conservation des distances entre les instances en passant de Y à Y' (par exemple Cost-sensitive Label Embedding with Multidimensional Scaling (CLEMS) [75])

Les objectifs dépendants des attributs peuvent être organisés en deux familles :

1. *Objectif LD1* : maximiser les corrélations entre X et Y' (par exemple Rembrandt [76]) ;
2. *Objectif LD2* : maximiser la qualité de prédiction de la matrice des attributs X à partir de la matrice des labels réduits Y' (par exemple Multi-Label Subspace Ensemble (MSE) [77])

Plusieurs approches hybrides résolvent un compromis paramétré entre un objectif indépendant et un objectif dépendant (par exemple Dependence Maximization based Label space dimensionality Reduction (DMLR) [78]).

Terminons ce paragraphe avec une note importante. Lorsque la réduction des labels est appliquée, le modèle d'apprentissage est entraîné avec X et Y'. Ainsi, ce dernier n'est ensuite capable que de prédire des vecteurs de labels réduits y' et il est donc nécessaire d'en déduire les vecteurs de labels y correspondants. Les trois approches les plus courantes pour répondre à ce besoin sont les suivantes :

— Un modèle de reconstruction $\Psi_{inv} : y' \mapsto y$ est construit en même temps ou après le modèle de réduction [79]. Il est ensuite utilisé pour reconstruire y à partir de y' dans la phase de test. Lorsque la réduction est basée sur une projection orthogonale P_y, la reconstruction de y est souvent déduite de la matrice transposée ($y = y' P_y^T$).

— Si la méthode de réduction de dimension fournit explicitement une fonction de réduction $\Psi : y \mapsto y'$, alors, étant donné un vecteur de labels réduit y', le vecteur de labels original y peut être déduit en résolvant le problème d'optimisation suivant ("structured output" [80]) :

$$\min_{y} l(y', \Psi(y)) \tag{2.2}$$

où l est une fonction de perte choisie. L'optimisation est souvent effectuée avec un algorithme de poursuite adapté ("matching pursuit" [81] ou "basis pursuit" [82].

— Les plus proches voisins de y' sont calculés dans l'ensemble d'apprentissage réduit et y est déduit de l'agrégation des labels d'origine de ses voisins [44].

Réduction de l'espace des attributs et des labels Lorsque les deux espaces sont réduits, la réduction de chaque espace dépend généralement de l'autre et deux stratégies principales ont été étudiées :

— *Objectif LFD1* : rechercher les directions principales à la fois dans l'espace des labels et dans l'espace des attributs qui maximisent les corrélations linéaires entre elles. Développée à l'origine avec la méthode populaire CCA (Canonical Correlation Analysis) [48], cette stratégie a conduit à des dizaines d'extensions dans l'apprentissage multi-label : par exemple l'extension avec une résolution moindre carrée LS-CCA [83], l'extension avec une technique de « sketching » [84], l'extension avec « output-code » [85], l'extension avec du deep learning [86]. De plus, certaines méthodes ont étendu la méthode CCA en la combinant avec d'autres approches (par exemple The Two-Stage Dual Space Reduction Framework (2SDSR)[87]).

— *Objectif LFD2* : minimiser une fonction de distance/similarité entre X' et Y' (par exemple Supervised Semantic Indexing SSI [60]).

Notons, pour compléter, qu'il existe un cas particulier (Independant Dual Space Reduction (IDSR) [88]) où l'espace des attributs et l'espace des labels sont traités indépen-

damment (*objectif LFI*) : une méthode de réduction des attributs indépendante des labels est appliquée sur X et une méthode de réduction des labels indépendante des attributs est appliquée sur Y.

2.1.2 Coupler la reduction de dimension avec l'objectif du classifieur

Comme indiqué précédemment, une grande majorité des approches de réduction sont appliquées en tant que pré-traitement des données indépendant de l'apprentissage qui suit. Mais cette procédure peut manquer de souplesse dans certains cas : les performances peuvent être élevées pour certains types de problèmes et très mauvaises sur d'autres. En effet, il a été observé sur de nombreux bancs d'expériences que l'impact d'une méthode de réduction sur les performances d'apprentissage varie avec le classifieur et les jeux de données [73]. Pour surmonter cette limitation, certains travaux ont commencé à étudier le couplage entre la réduction de dimension et l'apprentissage. La stratégie standard consiste à implémenter le couplage comme un problème d'optimisation multi-objectif qui tente d'optimiser simultanément les objectifs de réduction et du classifieur (respectivement f_d et f_c). Un problème multi-objectif et multi-paramètre est difficile à résoudre [89][90][91] et, en pratique, f_d et f_c sont alternativement ou conjointement maximisées via une combinaison linéaire (*Objectif C1* - Exemple : Simultaneous Large-margin and Subspace Learning Approach (TRANS) [92]). Le couplage peut également être mis en place selon deux autres scénarios :

1. *Objectif C2* : la réduction de dimension est intégrée au modèle d'apprentissage en remplaçant X et Y par X' et Y' dans f_c et l'unique objectif est de maximiser de f_c (voir par exemple Linear Dimensionality Reduction for Multi-label Classification (MLSVM) [93]).
2. *Objectif C3* : l'objectif de réduction de dimension f_d est implicitement conçu pour optimiser le classifieur. Cela se produit lorsque le classifieur est k-NN (k plus proches voisins). Par exemple, Supervised Orthonormal Locality Preserving Projection (SOLPP) [94] apprend une projection P_x sur l'espace des attributs X qui réduit la distance entre les instances qui partagent de nombreux labels. Cela optimise implicitement k-NN. Des stratégies similaires sont employées dans d'autres méthodes comme Hypergraph Spectral Learning (HSL) [71].

2.1.3 Transformation explicite ou implicite

Lorsque l'algorithme réduit les données via une fonction de transformation, la réduction est dite explicite et permet de calculer la transformation de n'importe quelle instance en ligne. Sinon, la transformation est dite implicite : elle fournit directement les données réduites mais pas la fonction de transformation.

Tableau 1a – Brève description des méthodes de réduction de dimension multi-label considérées dans cet état de l'art.

#	Method	Description
1	Principal Component Analysis (PCA)	Eigendecomposition of the feature covariance matrix to derive orthogonal directions of maximal variance (principal components).
2	Locality Preserving Projection (LPP)	Spectral decomposition of the instance adjacency graph to compute a reduced feature space that maximally preserves it.
3	Constrained Non-negative Matrix Factorization (CNMF)	Constrained non negative matrix factorization on X. The first factor is considered as the reduced feature space X'.
4	Random Principal Component Analysis (RPCA)	Randomized algebra technique (much faster than PCA when d_x is large) to approximate the principal components of X.
5	Auto-Encoder (AE)	Non linear reduction (projection + activation) and decoding to efficiently reduce and reconstruct the original feature space.
6	Model-Shared Subspace Boosting (MSSBoost)	Multiple random reductions of the feature space to create and combine a set of weak classifiers.
7	Orthonormal Locality Preserving Projection (OLPP)	Extension of LPP with an orthonormality constraint on the reduced feature space.
8	Orthonormal Neighborhood Preserving Projection (ONPP)	Orthonormal feature space projection which preserves each item location with respect to its l nearest neighbors.
9	Shared subspace for multi-Label (ML_{LS})	Embedding resulting from a trade-off between the label space and the feature space reconstructions.
10	Partial Least Square (PLS)	Construction of a dimensionality reducing projection that maximizes the correlations between the projected feature space and the label space.
11	Multi-label Latent Semantic Indexing (MLSI)	Linear feature space projection to optimize both the reconstruction of the original feature space and the correlations between the projected feature space and the label space.
12	Orthonormal Partial Least Square (OPLS)	Extension of PLS with an orthonormality constraint on the reduced feature space.
13	Hypergraph Spectral Learning (HSL)	Spectral decomposition of an hypergraph which links instances with many common labels to obtain a reduced feature space that favors locality between them.
14	Joint Dimensionality Reduction and Multi-label Classification (MLSVM)	Simultaneous learning of a feature space reducing projection and an SVM classifier applied on the obtained reduced space.
15	Multi-Label Dimensionality reduction via Dependence Maximization (MDDM)	Linear projection of the feature space that produces a reduced space with a minimal Hilbert Schmidt Independence with the label space.
16	Semi-Supervised Dimension Reduction for Multi-Label Classification (SSDR-MC)	Construction of a feature space projection which reproduces the neighborhood of the instances in the label space in the projected feature space.
17	Supervised Orthonormal Locality Preserving Projection (SOLPP)	Spectral decomposition of a feature/label adjacency trade-off graph to obtain a reduced feature space where neighbors have common features and labels.
18	Multi-label Linear Discriminant Analysis (MLDA)	Proposition of a definition for multi-label interclass/intraclass variances and computation of the linearly reduced feature space that maximizes their ratio.
19	Direct Multi-label Linear Discriminant Analysis (DMLDA)	Redefinition of MLDA's interclass variance matrix to overcome a limit that MLDA has on the dimensionality of the reduced space.
20	Variable Pairwise Constraint projection for Multi-label Ensemble (VPCME)	Proposition of "must/cannot link" constraints between instances (based on their labels) and computation of the feature space projection that maximally respects them.
21	Hypergraph Orthonormal Partial Least Square (HOPLS)	Trade-off between OPLS and HSL.
22	Shared Subspace Multi-Label Dim reduction via Dependence Maximization (SSMDDM)	Trade-off between ML_{LS} and MDDM.
23	Maximizing feature Variance and feature-label Dependence simultaneously (MVMD)	Trade-off between PCA and MDDM.
24	Multi-label prediction via Compressed Sensing (CS)	Reduction of the label space with a random projection and reconstruction of it with a sparse signal identification technique.
25	Principal Label Space Transformation (PLST)	Dually to PCA, computation of the orthogonal directions of maximum variance (principal components) in the label space.
26	Bayesian Multi-Label Compressed Sensing (BML-CS)	Simultaneous learning, with EM, of probabilistic models for (i) labels reduction, (ii) reduced labels prediction and (iii) labels decoding.
27	Multi-Label Classification via Boolean Matrix Decomposition (MLC-BMaD)	Boolean Matrix Decomposition to construct the binary reduced label space that can optimally reconstruct the original label space.

Tableau 1b – Brève description des méthodes de réduction de dimension multi-label considérées dans cet état de l'art.

#	Method	Description
28	Landmark Selection Method for Multiple Output Prediction (MOPLMS)	Resolution of a strongly regularized label space encoding/decoding problem and selection of the non-zero labels in the solution as the reduced label space.
29	Multi-Label Column Subset Selection Problem (ML-CSSP)	Derivation of label weights from the spectrum of the label covariance matrix and label sampling with the weighted probability to produce the reduced space Y'.
30	Cost-sensitive Label Embedding via Multidimensional Scaling (CLEMS)	Multidimensional scaling of the label space to embed instances according to a chosen instance pairwise cost which reflects the similarity of their label vector.
31	Conditional Principal Label Space Transformation (CPLST)	Combination of PLST and CCA to guide label space reduction with feature information.
32	Multi-label Subspace Ensemble (MSE)	Dimensionality reduction of the label space to improve its linear correlations with the feature space.
33	Feature-aware Implicit label space Encoding (FaIE)	Implicit reduction of the labels to maximize (i) their ability to reconstruct the original labels and (ii) their correlation with the feature space.
34	Response EMBedding via RANDomized Techniques (Rembrandt)	Eigenvalue decomposition of the feature-label covariance matrix with a sketching technique to reduce the label space and improve its link with the feature space.
35	Dependence Maximization based Label space dimension Reduction (DMLR)	Trade-off between PLST and MDDM.
36	Multi-Label Adapatative Random Projection (ML-ARP)	Computation of the feature space projection with an RVNS heuristic that optimizes the performances of the multi-label classifier ML-kNN on the reduced feature space.
37	Independent Dual Space Reduction (IDSR)	Independent application of PCA on the feature space and PLST on the label space.
38	Canonical Correlation Analysis (CCA)	Computation of the principal directions in both label and feature spaces that maximizes their linear correlations with each other.
39	Supervised Semantic Indexing (SSI)	Reduction of the feature space and the label space to increase (resp. decrease) the similarity between relevant (resp. irrelevant) pair x'-y' of feature and label vectors.
40	Least-Square Canonical Correlation Analysis (LS-CCA)	Approximate solution of CCA using an equivalent least square expression and an efficient resolution.
41	Regularized Canonical Correlation Analysis (rCCA)	Regularized version of CCA with an improved behavior when the label-feature covariance matrix is close to singular.
42	Web Scale Annotation By Image Embedding (WSABIE)	Construction of feature and label embeddings such as the instances' features average representation is similar to their labels' representation.
43	Simultaneous Large-margin and Subspace Learning Approach (TRANS)	Combination and simultaneous training of an unsupervised feature reduction method and a large margin multi-label classifier.
44	Deep Canonical Correlation Analysis (DCCA)	Application of CCA on $f_1(X)$ and $f_2(Y)$ where f_1 and f_2 are two deep neural networks. CCA and the networks are trained simultaneously.
45	Supervised Dual Space Reduction (2SDSR)	Family of methods that apply an existing dependent feature space reduction method on X (e.g MDDM) and an existing dependent label space reduction method on Y.
46	Convex Co Embedding (ILA)	Projection of the label and feature spaces which optimize a similarity, for each instance, between the reduced feature vector and reduced label vector.
47	Low rank Empirical risk minimization for Multi-Label Learning (LEML)	Simultaneous training of a classifier and a linear feature space reduction with the low rank Empirical Risk Minimization problem.
48	Bi-Directionnal Representation Learning (Bi-Dir)	Simultaneous predictions of labels from features and features from labels with an intermediary dimensionality reduction based on a bi-directional neural network.
49	Bayesian Multi-label Learning via Positive Labels (BMLPL)	EM-based construction of a subset of reduced labels (called topics) that can (i) reconstruct all the labels (Poisson law) and (ii) be predicted from features (Gamma law).
50	Sparse Local Embedding for Extreme Classification (SLEEC)	Clustering of the instances and construction of local embeddings, on each cluster, of the feature space to obtain the same closest neighborhood as in the label space.
51	Multi-label classification with feature-aware non-linear label space transformation (COMB)	Union of a reduced features space obtained with CCA and a reduced feature space obtained with KCCA.
52	Robust Extreme Multi-label Learning (REML)	Prediction of labels using both a linear model applied on a linearly reduced feature space and a sparse linear model applied on the original feature space.
53	Goal Inductive Matrix Completion (GIMC)	Multi-label matrix completion technique to reduce the instance features and labels.
54	Canonical-Correlated Auto-Encoder (C2AE)	Combination of a label space auto-encoder with CCA to reduce the features and the labels and to decode the predicted reduced labels.

Tableau 2.2 – Les méthodes de réduction de dimension, leur famille typologique et leurs critères. Nous reportons "oui" dans la colonne "Passe à l'éch. % à n (resp. d_x d_y) si la complexité est strictement inférieure à quadratique par rapport à n (resp. d_x d_y). Les codes objectifs, qui font référence au type d'objectif, sont détaillés dans la section 2.1. Les types de contraintes sont détaillées en section 2.1.4.

Famille de méthode	Dépendance	Méthode	Ref.	Année	Contrainte/ Régularisation	Code Objectif	Type de transformation	Passe à l'ech. % à n	Passe à l'ech. % à d_x d_y	Coupl. classif
Reduction de l'espace des attributs	Indépendant des labels	PCA	[64]	1901	Transfo. Orth.	FI1	Explicite Lin	Yes	No	No
		LSI	[95]	1990	Espace non corr.	FI2	Implicit	Yes	No	No
		LPP	[66]	2004	Espace non corr.	FI3	Explicite Lin	No	No	No
		CNMF	[96]	2006	Espace non corr.	FI1	Implicit	Yes	Yes	No
		RPCA	[97]	2006	Transfo. Orth.	FI1	Explicite Lin	Yes	Yes	No
		AE	[65]	2006		FI2	Explicite Non Lin	Yes	No	No
		MSSBoost	[98]	2007		R	Explicite Lin	No	No	No
		OLPP	**[99]**	2007	Espace non corr.	FI3	Explicite Lin	No	No	No
		ONPP	**[99]**	2007	Transfo. Orth.	FI3	Explicite Lin	No	No	No
	Dépendant des labels	PLS	[100]	1983	Transfo. Orth.	FD1	Explicite Lin	Yes	No	No
		MLSI	[72]	2005	Espace non corr.	FD3	Explicite Lin	Yes	No	No
		OPLS	[101]	2006	Espace non corr.	FD1	Explicite Lin	Yes	No	No
		HSL	[71]	2008	Espace non corr.	FD2/C3	Explicite Lin	No	No	Yes
		ML_{LS}	**[102]**	2008	Transfo. Orth.	FD3	Explicite Lin	Yes	No	No
		MLSVM	**[93]**	2009	Transfo. Orth.	C2	Explicite Lin	Yes	No	Yes
		MDDM	**[54]**	2010	Transfo. Orth.	FD1	Explicite Lin	Yes	No	No
		SSDR-MC	**[103]**	2010	Espace non corr.	FD2/C3	Implicit	No	No	Yes
		SOLPP	**[94]**	2010	Transfo. Orth.	FD2/C3	Explicite Lin	No	No	Yes
		MLDA	**[104]**	2010	Transfo. Orth.	FD1	Explicite Lin	Yes	No	No
		DMLDA	**[105]**	2013	Transfo. Orth.	FD1	Explicite Lin	Yes	No	No
		VPCME	**[106]**	2013	Transfo. Orth.	FD2/C3	Explicite Lin	No	No	Yes
		HOPLS	**[107]**	2014	Espace non corr.	FD1/C3	Explicite Lin	No	No	Yes
		SSMDDM	**[108]**	2015	Régularisation	FD1/FD3	Explicite Lin	Yes	Yes	No
		LM-kNN	[109]	2015		C3	Explicite Lin	Yes	No	Yes
		MVMD	**[73]**	2016	Transfo. Orth.	FI1/FD1	Explicite Lin	Yes	No	No
		ML-ARP	**[110]**	2017		C2	Explicite Lin	No	Yes	Yes
Réduction de l'espace des labels	Indépendant des attributs	CS	[50]	2009		R	Explicite Lin	Yes	Yes	No
		PLST	**[74]**	2012	Transfo. Orth.	LI1	Explicite Lin	Yes	No	No
		MLC-BMaD	**[111]**	2012	Espace Binaire	LI2	Implicit	No	No	No
		MOPLMS	**[112]**	2012	Selection	LI2	Explicite	Yes	Yes	No
		ML-CSSP	**[113]**	2013	Selection	LI1	Explicite	Yes	No	No
		CLEMS	**[75]**	2016		LI3	Implicit	No	Yes	No
	Dépendant des attributs	CPLST	**[114]**	2012	Transfo. Orth.	LD1/LD2	Explicite Lin	Yes	No	No
		MSE	**[77]**	2012		C2	Implicit	No	No	Yes
		BML-CS	**[115]**	2012		C1	Explicite Non Lin	Yes	No	Yes
		FaIE	**[116]**	2014	Espace non corr.	LD1/LD2	Implicit	Yes	No	No
		Rembrandt	**[76]**	2015		LD1	Explicite Lin	Yes	Yes	No
		DMLR	**[78]**	2015	Transfo. Orth.	LD1/LD2	Explicite Lin	Yes	No	No
Réduction de l'espace des attributs et des labels	Indépendant	IDSR	**[117]**	2013	Transfo. Orth.	LFI	Explicite Lin	Yes	No	No
	Dépendant	CCA	**[118]**	1936	Espace non corr.	LFD1	Explicite Lin	Yes	No	No
		SSI	[60]	2009		LFD2	Explicite Lin	Yes	Yes	No
		LS-CCA	**[118]**	2011	Espace non corr.	LFD1	Explicite Lin	Yes	Yes	No
		rCCA	**[118]**	2011	Espace non corr.	LFD1	Explicite Lin	Yes	No	No
		WSABIE	[119]	2011		LFD2	Explicite Lin	Yes	Yes	No
		TRANS	[92]	2012	Régularisation	C1	Implicite	Yes	Yes	Yes
		DCCA	**[86]**	2013	Espace non corr.	LFD1	Explicite Non Lin	Yes	Yes	No
		2SDSR	**[87]**	2013	Several	LFD1	Explicite Lin	Yes	No	No
		ILA	[120]	2014		LFD2	Explicite Lin	Yes	Yes	No
		LEML	**[43]**	2014	Régularisation	C2	Explicite	Yes	Yes	Yes
		Bi-Dir	**[79]**	2014		C1	Both	Yes	Yes	Yes
		BMLPL	**[121]**	2015		C1	Implicit	Yes	Yes	Yes
		SLEEC	**[44]**	2015	Régularisation	C3	Both	Yes	Yes	Yes
		COMB	[122]	2015	Espace non corr.	LFD1	Explicite Non Lin	Yes	No	No
		REML	**[123]**	2016	Régularisation	C2	Both	Yes	Yes	Yes
		GIMC	**[124]**	2016	Régularisation	LFD2	Explicite Non Lin	Yes	Yes	No
		C2AE	[125]	2017	Transfo. Orth.	C1	Explicite Lin	Yes	Yes	Yes

Transformations explicites La grande majorité des méthodes de réduction multi-label existantes réduisent la dimension avec des projections ($X' = XP_x$ ou $Y' = YP_y$). Elles sont par conséquent explicites et linéaires. Ces transformations linéaires peuvent être étendues à une transformation non linéaire avec l'astuce classique du noyau et la plupart des méthodes linéaires ont une extension de noyau (par exemple kPCA [126] pour PCA, kCCA [127] pour CCA).

Il existe également des approches explicites nativement non linéaires dans le cadre multi-label. Elles peuvent être classées en trois catégories :

1. Locally Linear Embeddings (LLE) [44][128] : cette technique produit une transformation non linéaire, correspondant à une transformation linéaire par morceaux, en partitionnant l'espace des labels et/ou l'espace des attributs et en calculant une transformation linéaire spécifique par région du partitionnement.
2. Les méthodes d'apprentissage de représentation. Ces méthodes effectuent la réduction via des réseaux de neurones. En général, les variables initiales sont introduites dans la couche d'entrée du réseau, les données réduites correspondent généralement à une couche cachée où le nombre de neurones est inférieur à celui de l'entrée, et la sortie dépend du type de réseau. Pour les auto-encodeurs [65], la sortie est une reconstruction de la couche d'entrée. Pour les réseaux de neurones multi-label [129][130][131], la sortie est une prédiction de Y (respectivement X) et l'entrée est X (resp. Y). Des architectures plus complexes, combinant des auto-encodeurs et des perceptrons multicouches, ont été récemment explorées [125][79]. Pour plus de détails, nous renvoyons à la revue complète [52] sur l'apprentissage de la représentation qui inclut plusieurs méthodes qui ont été adaptées à l'apprentissage multi-label [132].
3. Les méthodes probabilistes [133][115][121]. La transformation de l'espace initial (X ou Y) vers l'espace réduit (X' ou Y') est une combinaison de lois de probabilité paramétrées (souvent des distributions normales, de Dirichlet et Gamma). Dans ce cas, la construction de l'espace réduit est réalisée par inférence.

Transformations implicites Les transformations implicites fournissent directement l'espace réduit sans calculer explicitement l'opérateur de transformation (par exemple en utilisant la strategie de Multi-Dimensional Scaling (MDS) [134] ou la factorisation matricielle [6]). Elles ne sont donc pas contraintes à être linéaires. L'apprentissage direct des espaces réduits X' ou Y' offre plus de degrés de liberté dans le problème d'optimisation, mais il est plus fréquemment confronté à un sur-apprentissage [135]. De plus, il n'est pas adapté aux processus incrémentaux : lorsqu'un nouvel élément est ajouté, la réduction doit être entièrement relancée. Néanmoins, le récent intérêt de la recherche pour l'apprentissage multi-label extrême [36], où le nombre de label est extrêmement grand, stimule le

développement de transformations implicites [119][44][43][116][75]. Elles y sont adaptées car la réduction en ligne d'un vecteur de labels n'y est généralement pas requise. Au contraire, les transformations explicites sont plus appropriées pour réduire l'espace des attributs car, en phase de prédiction, il est nécessaire de transformer en ligne des vecteurs d'attributs jamais rencontrés pendant l'apprentissage (Figure 2.1).

2.1.4 Régularisation et contraintes

Ajouter une fonction de régularisation r à la fonction objectif f_d ou un ensemble de contraintes au problème d'optimisation (2.1) vise (i) à réduire les degrés de liberté du problème, (ii) à simplifier les transformations (des espaces initiaux vers les espaces réduits) en limitant leur paramètres, (iii) à améliorer la généralisation des modèles en limitant le sur-apprentissage et (iv) à construire des données plus favorables à l'apprentissage. Ces objectifs sont intégrés dans le problème d'optimisation de plusieurs façons :

- Transformations parcimonieuses. Certaines méthodes imposent la parcimonie sur les variables de l'espace réduit ou sur les paramètres de la fonction de réduction [44][43]. Formellement, la parcimonie d'une matrice est calculée à partir de sa norme L_0, mais en raison de sa non-continuité et de sa non-dérivabilité, les auteurs recourent généralement au L_1-trick en relaxant la norme L_0 en une norme L_1 [136]. En pratique, cette approche limite le sur-apprentissage, optimise le stockage et accélère l'apprentissage et les prédictions ;
- Espace de solution limité. Une grande partie des algorithmes impose la minimisation de la norme L_2 des paramètres. Cela favorise les solutions dont les paramètres ont de faibles valeurs [79][83].
- Paramètres parcimonieux et petits. Ce compromis entre les deux derniers objectifs correspond à la régularisation "elastic net" [137] qui est une combinaison linéaire des régularisations L_1 et L_2 [44].
- "Clipping" de paramètre. Cette régularisation restreint le domaine de définition des paramètres à un intervalle fixe avec des techniques de seuillage [138].
- Régularisation de type "drop-out". Certaines approches basées sur les réseaux de neurones régularisent leurs paramètres en utilisant la stratégie de "drop-out" [139] qui consiste à ne considérer qu'un sous-ensemble de paramètres différents sélectionnés aléatoirement à chaque étape d'apprentissage.

De plus, des contraintes sont également introduites pour limiter le bruit et les corrélations entre variables qui sont des entraves pour la plupart des classifieurs [140]. Deux contraintes habituelles visent à faciliter la tâche d'apprentissage :

- Espace non corrélé. L'apprentissage est plus facile lorsque les corrélations dans l'espace des variables sont limitées. Une telle contrainte peut être exprimée sous forme matricielle $X'^T X' = I$ (ou $Y'^T Y' = I$). Remarquons que cette contrainte

conduit à une régularisation L_2 ($|X'|_2 = tr(X'^T X) = tr(I)$).

— Projection orthonormale. Cette contrainte ne concerne que le cas le plus populaire des transformations linéaires explicites et est exprimée par $P^T P = I$.

Certains auteurs ont également proposé le compromis suivant $P^T((1-\mu)X^T X + \mu I)P = I$ [73][54].

2.2 Méthodes de réduction en XML

Après réduction des espaces de labels et d'attributs d'un problème multi-label extrême, on obtient un nouveau problème sur lequel la majeure partie des classifieurs multi-label standards sont capables d'apprendre. Cependant, la tâche de réduction n'est pas sans coût. Bien au contraire, la majeure partie des algorithmes de réduction de dimension sont complexes. Par exemple, l'analyse en composantes principales PCA a une complexité qui peut être cubique et il n'est donc pas envisageable de l'employer sur des données extrêmes. Dans cette section, nous présentons les algorithmes qui ont été proposés spécifiquement pour des problèmes en très grande dimension : SSI, WSABIE, LEML, SLEEC, REML, GIMC et AnnexML.

2.2.1 Longue traîne et approximation de rang faible

Avant de détailler les méthodes de réduction de dimension XML, nous ouvrons une parenthèse concernant la distribution des labels et les risques liés à la réduction de dimension. En particulier, l'approximation de rang faible sur les labels faite par une grande partie des méthodes de réduction de dimension peut dégrader fortement les performances dans le cadre extrême. L'approximation de rang faible consiste à modéliser une matrice M par une matrice M' de rang faible (relativement à M). Le plus souvent, celle-ci résulte d'une réduction de dimension effectuée avec une projection globale $P_y \in \mathbb{R}^{d_y \times d'_y}$ de l'espace des labels (figure 2.3) : la matrice des labels réduits Y' est obtenue par la multiplication matricielle suivante YP_y. Dans un second temps, les labels d'origine sont reconstruits par la projection transposée ou une autre projection $P_{inv_y} \in \mathbb{R}^{d'_y \times d_y}$. Pour qu'un tel modèle soit efficace, il faut alors que Y soit proche de $YP_yP_{inv_y}$ où $P_yP_{inv_y}$ est une matrice de dimension d_y mais de rang d'_y. On peut alors dire que ce modèle fait une approximation de rang faible.

Dans certaines circonstances où les d_y labels contiennent des informations encodables sur d'_y variables (par exemple parce qu'ils ont été générés par d'_y thématiques), un tel modèle peut être très efficace. En revanche, en apprentissage multi-label extrême, un tel modèle permet de reconstruire un grand pourcentage de la matrice Y mais se concentre naturellement sur les labels les plus fréquents. La longue traîne est alors très mal encodée dans l'espace réduit et les performances des classifieurs sont fortement affectées [44].

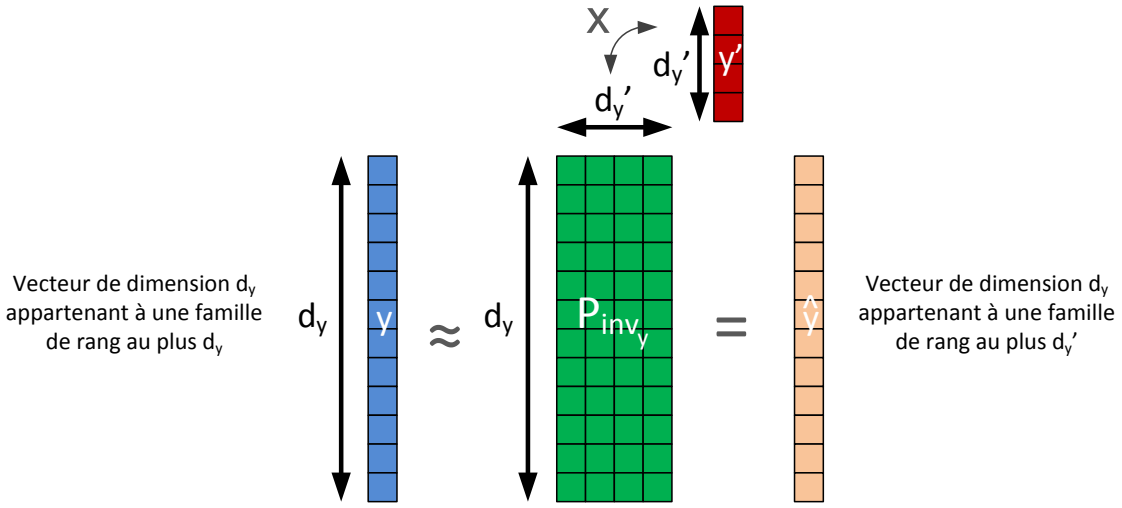

Figure 2.3 – Illustration de l'approximation de rang faible sur les labels lors d'une réduction de dimension avec projection globale.

2.2.2 Méthodes

Supervised Semantic Indexing (SSI) [60] Cette approche a initialement été développée pour apprendre à évaluer et à ordonner la qualité de l'adéquation entre des requêtes textuelles et des documents, avec comme illustration pragmatique la recommandation d'articles sur Wikipedia. Elle repose sur l'apprentissage d'une projection des attributs (P_x) et d'une projection des labels (P_y) telles que la similarité entre un vecteur d'attributs x et un vecteur de label qui lui est pertinent y^+ (resp. non pertinent y^-) soit maximale (resp. minimale) après projection, i.e. telles qu'il y ait une forte (resp. faible) similarité entre xP_x et y^+P_y (resp. y^-P_y).

On suppose qu'au départ, on a construit un ensemble $\mathcal{R}$ de contraintes de triplet (x_i, y_i^+, y_i^-) ou contraintes relatives et que les projections recherchées minimisent.

$$\min_{P_x \in \mathbb{R}^{d_x \times d_x'}, P_y \in \mathbb{R}^{d_y \times d_y'}} \sum_{(x_i, y_i^+, y_i^-) \in \mathcal{R}} max(0, m - x_i P_x P_y^T y_i^{+T} + x_i P_x P_y^T y_i^{-T}) \tag{2.3}$$

où m est un paramètre de marge et $d_x' = d_y'$.

Les matrices de projection P_x et P_y sont initialisées aléatoirement puis ajustées par descente de gradient stochastique. Cette méthode d'optimisation parvient à tirer profit du caractère creux des données x et y et est rapide.

Web Scale Annotation by Image Embedding (WSABIE) [119] Le principe de WSABIE est très similaire à celui de SSI car il apprend à établir un classement des similarités entre les labels (projetés par P_y) et les attributs (projetés par P_x). La différence majeure est que WSABIE ne définit pas son problème sur des triplets d'instances mais sur les différents labels d'une même instance. Plus précisément, l'objectif est que la similarité entre les attributs et les labels associés à l'instance soit plus grande que la similarité entre les attributs et les autres labels. La fonction de perte est définie avec le WARP loss et le problème est résolu par une méthode itérative.

Low rank Empirical risk minimization for Multi-Label Learning (LEML) [43] LEML est un cadre générique couplant apprentissage et réduction dans un même problème d'optimisation (minimisation du risque empirique). Si l'on note $f(x; Z) : x \mapsto \widehat{y}$ le modèle d'apprentissage, $l(y, f(x; Z))$ la perte entre les labels prédits et les labels réels, et $r(Z)$ une régularisation sur les paramètres Z, alors une expression générique de l'apprentissage multi-label pour la minimisation du risque empirique (ERM) est :

$$\begin{aligned} \widehat{Z} = \underset{Z}{\operatorname{argmin}} \quad & \sum_{i=1}^{n} l(y_i, f(x_i; Z)) + \lambda r(Z) \\ \text{Avec} \quad & rang(Z) \leq k \end{aligned} \tag{2.4}$$

La contrainte sur le rang de Z impose, en sus, implicitement une réduction de dimension pendant l'apprentissage (le passage de X à Y a pour intermédiaire un espace réduit). Dans ce cadre, les pertes "squared loss", "logistic loss", "squared hinge loss" et TRON [141] ont été testées par les auteurs. Le problème est résolu avec une méthode de Gradient conjugué et tire profit du caractère creux des données.

Sparse Local Embeddings for Extreme multi-label Classification (SLEEC) [44] SLEEC a été conçu pour pallier les limitations observées sur une grande partie des algorithmes de plongement/réduction de dimension (notamment SSI, WSABIE et LEML) qui considèrent, par construction de leur modèle, que l'espace des labels est de rang faible. En pratique cette approximation est très limitante en XML à cause de la longue traîne non négligeable de labels (voir Section 2.2.1). Pour éviter l'approximation de rang faible, SLEEC propose d'apprendre un ensemble de projections locales. Il construit k ensembles de données avec l'algorithme des k-moyennes (k-means) sur X et apprend, pour chaque ensemble, un plongement local des attributs dont l'objectif est de rapprocher les instances dont les labels sont proches. Pour cela, un plongement des labels qui préserve les similarités entre voisins dans l'espace des labels est d'abord appris :

$$\min_{Y' \in \mathbb{R}^{n \times d'_y}} \|S_\Omega(YY^T) - S_\Omega(Y'Y'^T)\|_F^2 \tag{2.5}$$

où Y^TY (resp. $Y'Y'^T$) est la matrice de similarité entre les instances dans la matrice des labels initiaux (resp. des labels transformés) et S_Ω un opérateur d'échantillonnage ne préservant que les composantes de la matrice dont les indices appartiennent à l'ensemble Ω défini comme suit :

$$\Omega = \{(i,j) \in \{1,...,n\}^2 | \text{l'instance j est voisine de l'instance i dans Y}\} \tag{2.6}$$

Une fois Y' déterminé, on cherche le plongement des attributs qui s'en rapproche au mieux :

$$\min_{P_x \in \mathbb{R}^{d_x \times k}} \|Y' - XP_x\|_F^2 + \lambda \|P_x\|_F^2 + \|XP_x\|_1 \tag{2.7}$$

Les régularisations sont introduites de sorte que la projection ainsi que la matrice des attributs projetés soit parcimonieuses. Le problème (2.5) est une version du problème de complétion de matrice de rang faible que les auteurs résolvent par la méthode SVP (Singular Value Projection) [142]. La norme 1 dans (2.7) conduit les auteurs à résoudre ce problème avec ADMM [143].

L'ensemble des plongements locaux appris de rang faible donne un plongement global de rang fort qui permet de très bien représenter les données XML ; ce qui permet à SLEEC de dépasser largement les anciennes méthodes de plongement [50][74][114][43][119]. Les prédictions de SLEEC sont effectuées au sein des clusters avec l'algorithme des plus proches voisins sur l'espace des attributs réduits.

Robust Extreme Multi-label Learning (REML) [123] Comme SLEEC, REML souhaite éviter l'approximation de rang faible mais avec une stratégie différente. Il apprend donc d'abord une représentation réduite de rang faible $\widehat{Y_L}$ de Y et la complète avec une matrice très creuse dans l'idée d'y ajouter l'information apportée par la longue traîne : $Y \approx \widehat{Y_L} + \widehat{Y_S}$ où $\widehat{Y_L}$ est la matrice de rang faible et $\widehat{Y_S}$ est la matrice creuse. Le problème se formalise alors de la façon suivante :

$$\begin{aligned} \min_{\widehat{Y_L}, \widehat{Y_S}} & \|Y - \widehat{Y_L} - \widehat{Y_S}\|_F^2 \\ & rank(\widehat{Y_L}) \leq d'_x \\ & card(\widehat{Y_S}) \leq s \end{aligned} \tag{2.8}$$

où $card(\widehat{Y_S})$ désigne le nombre de composantes non nuls dans la matrice Y_S.

L'idée finale étant de pouvoir prédire les labels à partir des attributs, l'étape suivante est d'apprendre un classifieur qui prédit $\widehat{Y_L}$ et $\widehat{Y_S}$ à partir de X. En réalité, l'apprentissage du classifieur est directement intégré dans le problème 2.8. Plus précisément, on y intègre directement une régression linéaire de X vers $\widehat{Y_L}$ ($\approx XW$) et de X vers $\widehat{Y_S}$ ($\approx XH$). Pour assurer l'approximation de rang faible on pose $W = P_x P_{inv_y}$ avec $P_x \in \mathbb{R}^{d_x \times d'_x}$ et

$P_{inv_y} \in \mathbb{R}^{d'_y \times d_y}$ (avec $d'_x = d'_y$). Et la contrainte de parcimonie sur $\widehat{Y_S}$ ($card(\widehat{Y_S}) \leq s$) est remplacée par la minimisation de sa norme L_1. Finalement le problème régularisé s'écrit :

$$\min_{U,V,H} \|Y - XP_xP_{inv_y} - XH\|_F^2 + \lambda_1\|H\|_F^2 + \lambda_2(\|P_x\|_F^2 + \|P_{inv_y}\|_F^2) + \lambda_3\|XH\|_1 \quad (2.9)$$

Les paramètres (P_x, P_{inv_y}, H) sont ajustés alternativement. La matrice Hest stockée en creux pour entrer en mémoire et est apprise en parallèle. Les auteurs proposent une borne pour l'erreur d'estimation et l'erreur de généralisation de REML.

Notons que du fait du lien direct entre Y et X via H qui n'est pas de rang faible, REML est en fait une méthode hybride qui ne dépend que partiellement de la réduction de dimension.

Goal-Directed Inductive Matrix Completion (GIMC) [124] GIMC réduit la dimension des données à l'aide d'une méthode de factorisation de matrice exploitant des informations additionnelles (variables supplémentaires décrivant les éléments des colonnes et des lignes de la matrice à factoriser) et introduisant des non linéarités avec des noyaux. Les auteurs proposent en particulier une application à l'apprentissage multi-label extrême.

Pour une matrice A, le problème de complétion de matrice se formule par :

$$\min_{W,H} \|A - WH^T\|_F^2 + \lambda(\|W\|_F^2 + \|H\|_F^2) \quad (2.10)$$

Lorsque A est très creuse, la minimisation de l'erreur n'est calculée que sur les valeurs non nulles de A. Lorsqu'on dispose d'informations (variables) additionnelle(s) X pour les lignes et L pour les colonnes permettant potentiellement de renforcer le lien entre la factorisation (H,W) et la matrice initiale (A), on remplace W par XW et H par YH et le problème résultant est appelé "inductive matrix completion" (IMC). Pour améliorer les performances, GIMC propose non pas d'utiliser X et L mais une transformation de ces matrices avec des noyaux appris ($\phi_U(X)$ et $\phi_V(L)$) comme information additionnelle. La transformation ϕ est une transformation en attributs de Fourrier ou une transformation en attributs de Nystrom et $U \in \mathbb{R}^{d_x \times k}$ et $V \in \mathbb{R}^{d'_y \times k}$ sont des paramètres sur ces noyaux.

La formulation de GIMC est alors :

$$\min_{W,U,V,H} \|A - \phi_U(X)WH^T\phi_V(L)^T)\|_F^2 + \lambda(\|W\|_F^2 + \|H\|_F^2) \quad (2.11)$$

Pour appliquer GIMC à l'apprentissage multi-label, on considère le cas $A = Y$ et $\phi_V(L) = I$ et on pose $Z = WH^T$. Le problème devient alors :

$$\min_{Z,U,V} \|Y - \phi_U(X)Z\|_F^2 + \lambda\|Z\|_F^2 \quad (2.12)$$

Pour U fixé, on résout un système linéaire donc $Z = (\phi_U(X)^T\phi_U(X)+\lambda I)^{-1}\phi_U(X)^T Y$.

Pour Z fixé, le sous-problème sur U depend du type de noyau : de Fourier (GIMC-LFF) ou de Nystrom (GIMC-LNYS). Dans les deux cas, il est résolu par "line search". Finalement, après avoir déterminé Z et U, on effectue des prédictions des labels $\widehat{y}$ à partir d'un vecteur d'attribut x de la façon suivante : $\widehat{y} = \phi_U(x)Z$.

AnnexML [144] AnnexML a une stratégie qui est très proche de SLEEC avec deux améliorations intéressantes : un découpage des instances supervisé (plus pertinent qu'un k-means non supervisé) avant d'apprendre les plongements locaux, et l'apprentissage d'un graphe pour remplacer les plus proches voisins par une version approchée pour accélérer les prédictions de plusieurs ordres de grandeurs.

3 Solution 2 : méthodes hiérarchiques arborescentes

Dans de nombreuses applications d'annotation multi-label, on peut faire l'hypothèse raisonnable de l'existence de structurations, cachées dans la masse ou disponibles pour l'apprentissage dans les labels ou dans les instances. Il s'agit d'un partitionnement, parfois hiérarchique, dans lequel sont regroupés des labels/instances qui ont un sens "proche", la notion de proximité étant dépendante de la définition de l'utilisateur. Par exemple, dans la célèbre base d'image ImageNet [1], tous les labels qui correspondent à des espèces animales sont regroupés dans des ensembles de labels "mammifères", "reptiles", eux même regroupés dans un ensemble de labels correspondant aux "animaux" (figure 2.4). Un autre exemple célèbre est la hiérarchie sur l'ensemble des catégories Wikipédia [145] (figure 2.5).

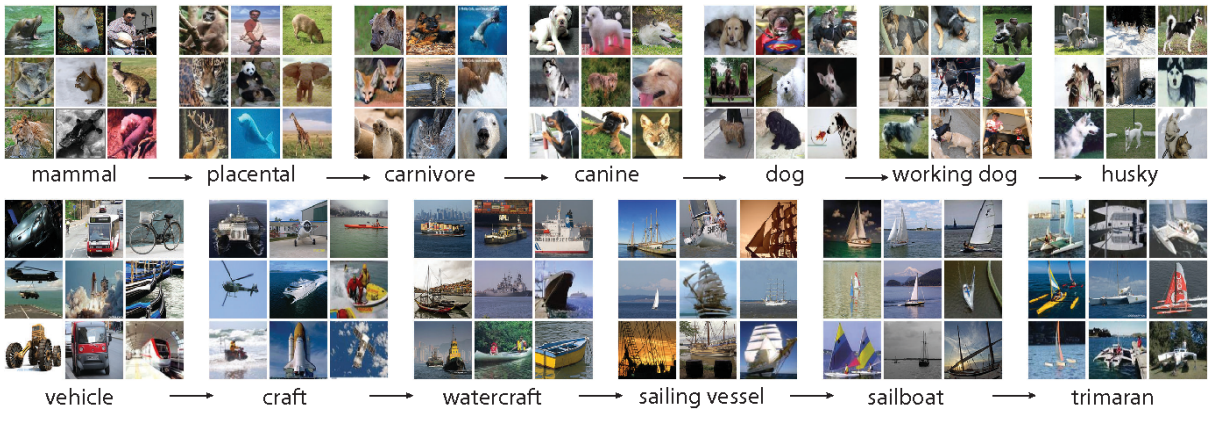

Figure 2.4 – Exemple de deux chemins racine-feuille dans la hierarchie des labels de ImageNet (image extraite de [1])

Selon les problèmes, le partitionnement respecte certaines propriétés comme, par exemple, l'exclusivité : un label peut appartenir à un unique ensemble. La connaissance

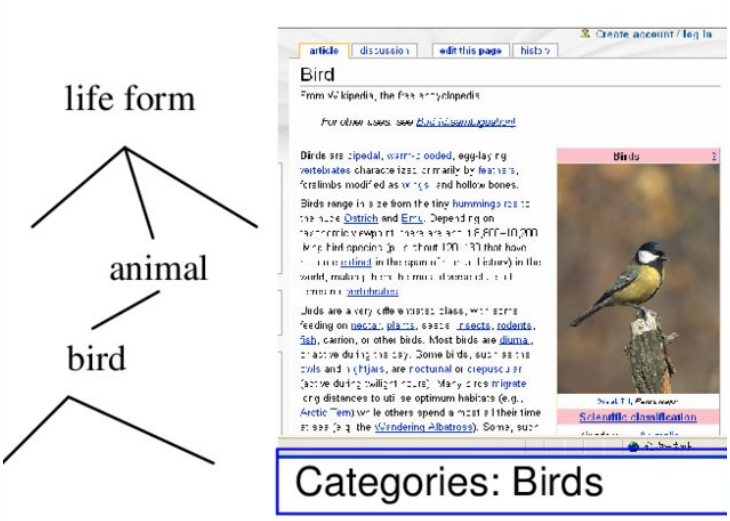

Figure 2.5 – Exemple du chemin dans l'arbre des labels de Wikipedia pour atteindre le label "Bird".

d'un partitionnement hiérarchique sous-jacent permet de scinder le problème pour en accélérer la résolution et obtenir ainsi des solutions plus pertinentes. Plus précisément, cela permet d'exclure ou de conserver successivement des labels/instances et de ne se focaliser à chaque étape que sur un sous-ensemble. Cependant, dans la plupart des problèmes d'apprentissage multi-label, le partitionnement n'est pas connu de l'algorithme apprenant. Mais les algorithmes d'apprentissage appelés méthodes hiérarchiques arborescentes ont une stratégie qui vise à découvrir un partitionnement sur les ensembles bruts d'instances ou de labels.

Les méthodes arborescentes transforment le problème initial à grande échelle en une série de sous-problèmes à petite échelle en partitionnant de manière hiérarchique l'ensemble des instances (arbre de décision classique ou arbre d'instances) ou l'ensemble des labels (arbre de labels). Ces différents sous-ensembles sont associés aux noeuds d'un arbre. L'ensemble initial associé à la racine est partitionné en un nombre fixe k de sous-ensembles associés aux k noeuds enfants de la racine. Le processus de partitionnement est répété jusqu'à ce qu'une condition d'arrêt soit vérifiée sur les sous-ensembles. Dans chaque noeud, deux problèmes d'optimisation sont posés : (i) calculer une partition/couverture pour un critère donné (Point 1 sur la Figure 2.6), et (ii) définir une condition ou construire un classifieur qui, à partir des attributs, détermine à quel(s) sous-ensemble(s) (noeud enfant) une instance est attribuée (Point 2 sur Figure 2.6). Lorsque l'arbre est construit, en phase de prédiction, une instance de test suit un chemin de la racine jusqu'à une feuille (arbre d'instance) ou plusieurs feuilles (arbre de labels) déterminées par les décisions locales successives des classifieurs. Un processus permet ensuite d'effectuer des prédictions à partir des feuilles atteintes.

Dans cette section, nous commençons par répondre successivement aux trois questions suivantes : (i) quelles sont les stratégies générales pour construire les partitions (points 1 sur la Figure 2.6) ? (ii) Connaissant les attributs d'une (nouvelle) instance, comment lui associe-t-on un ensemble dans le partitionnement (Point 2 sur la Figure 2.6) ? (iii) Comment sont effectuées les prédictions avec les arbres construits (Point 3 sur la Figure 2.6) ? Selon les méthodes, ces étapes clés sont effectuées successivement [37][146] où simultanément [36][38]. Ensuite, nous présenterons les méthodes arborescentes de la littérature multi-label extrême : MLRF, LPSR, PLT, FastXML et PFastReXML.

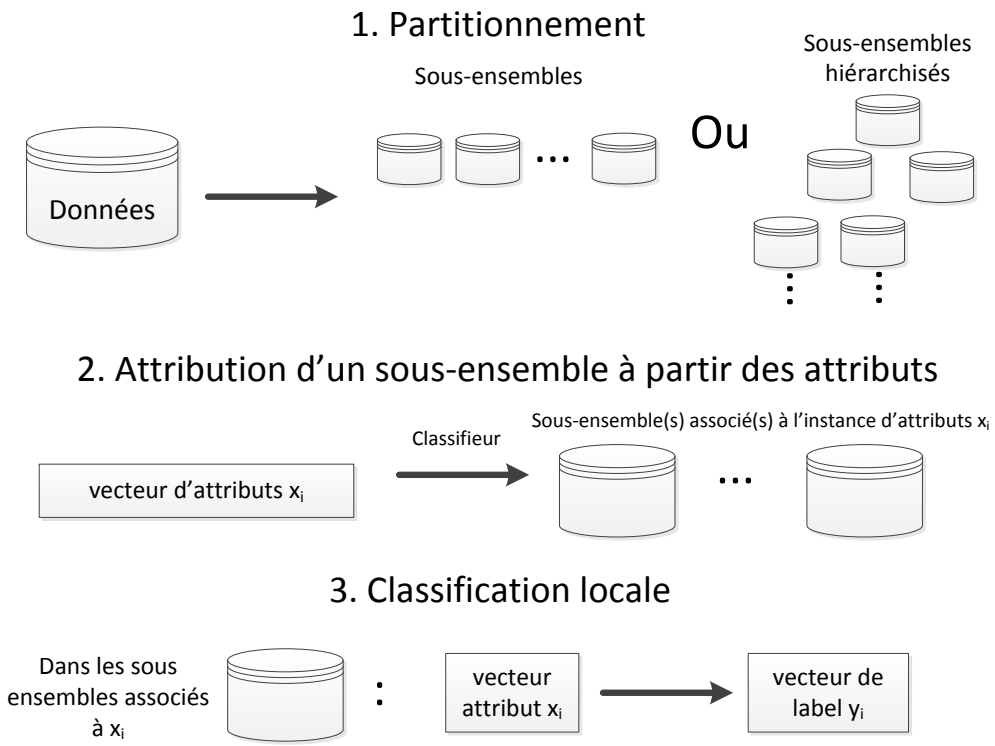

Figure 2.6 – Éléments clés dans l'apprentissage avec partitionnement.

3.1 Construction des partitions

La stratégie de partitionnement est le point essentiel qui différencie les méthodes arborescentes entres elles. Celle-ci consiste à définir sur l'ensemble à partitionner un critère de partitionnement et à construire les sous-ensembles qui l'optimisent. Quatre démarches assez proches ont été explorées :

1. Objectif de proximité : On souhaite que le partitionnement soit tel que deux éléments dans un même sous-ensemble aient une forte proximité. Si les éléments des ensembles sont des instances, la proximité peut être définie par une distance. Si ce

sont des variables, on peut s'intéresser à des mesures comme la co-occurence ou la covariance. Les critères couramment utilisés sont par exemple le critère Gini [40] ou la co-occurrence [41].

2. Objectif d'exclusion : A l'opposé de la stratégie précédente, on souhaite que les éléments de deux sous-ensembles différents aient une faible proximité. Cet objectif maximise donc une forte "variance inter ensembles". On peut également imposer que les sous-ensembles soient mutuellement disjoints [147].
3. Objectif de construction de liens logiques entre les sous-ensembles : En général, la présence de certains labels augmente fortement la probabilité de présence d'autres labels. Par exemple, sur une image, si on prédit les labels "étoiles" et "lune", il est fortement probable de prédire le label "nuit" [146]. A l'inverse, certains labels permettent naturellement d'en exclure d'autres (exclusivité mutuelle) comme la présence d'un label "savane" qui diminuerait fortement les chances de présence de labels relatifs aux animaux polaires. Les couvertures de Markov et les critères d'entropie conditionnelle [146] sont bien adaptés pour ce type de propriétés.
4. Objectif d'amélioration de l'apprentissage : Cet objectif est implicitement celui de tous les auteurs qui intègrent la construction de sous-ensembles dans l'objectif global d'apprentissage. En revanche, seules quelques méthodes l'implémentent explicitement pour améliorer spécifiquement les mesures de performance en apprentissage qui les intéressent [36][38]. LPSR vise par exemple à construire des sous-ensembles où les mesures de performance de "ranking" d'un classifieur sont améliorées [37].

Le partitionnement peut s'effectuer avec des stratégies standards comme le clustering ou avec la résolution d'un problème d'optimisation sous contraintes dont les paramètres à optimiser sont les sous-ensembles et où l'objectif et la résolution sont définis sur mesure. C'est le cas dans FastXML [36] par exemple.

3.2 Attribution des partitions

Après ou pendant le partitionnement, on souhaite pouvoir déterminer à quel(s) sous-ensemble(s) une instance est attribuée en fonction de ses attributs x. Ce problème d'association constitue en lui même un problème d'apprentissage multi-label ou multiclasse (voire monolabel) que l'on peut résoudre comme tel. Si le partitionnement est hiérarchique tel que chaque noeud n'a que k ($<< d_y$) enfants, le problème d'association n'a que k labels ou classes et est soluble avec de nombreux classifieurs standards. Dans le cas le plus général, on utilise un "one-vs-rest" SVM [148] régularisé ou une régression logistique. Dans certains cas, il n'est pas nécessaire d'apprendre un classifieur car il est déduit automatiquement du partitionnement. Par exemple, si les sous-ensembles ont été

construits par une méthode de clustering sur les attributs, on peut considérer qu'une nouvelle instance est classifiée dans le sous-ensemble dont le centroïde est le plus proche [37].

3.3 Prédictions à partir de l'arbre construit

Après avoir réalisé les deux précédentes étapes, on est capable de déterminer, pour une nouvelle instance donnée, le(s) sous-ensemble(s) (feuille(s) de l'arbre) qui lui sont associé(s). Pour répondre finalement au problème d'apprentissage multi-label, il reste nécessaire d'avoir un/des modèle(s) capable(s) d'effectuer un apprentissage sur les sous-problèmes définis localement dans les feuilles et de proposer une prédiction globale.

On distinguera ici les arbres d'instances et les arbres de labels :

— Pour un arbre d'instance, effectuer l'apprentissage local à l'intérieur d'une feuille (constituée d'instances) est un problème d'apprentissage multi-label dont l'avantage par rapport au problème d'apprentissage global est sa faible dimension (peu de labels et d'instances). Dans une majorité des cas, les feuilles sont bien construites (les instances y ont des vecteurs de labels très proches) et sont peu peuplées : on y utilise alors des classifieurs locaux très simples comme la moyenne.

— Pour un arbre de label, les feuilles correspondent à des labels. Donc, les feuilles associées à la nouvelle instance constituent directement les prédictions.

3.4 Notion de forêt aléatoire

Lorsqu'un arbre est peu stable, c'est-à-dire pouvant produire des résultats différents lors de répétition d'entraînement sur le même ensemble d'apprentissage, il est souvent intéressant d'en apprendre plusieurs et d'effectuer des prédictions globales par agrégation des prédictions de chaque arbre. Un tel "ensemble" sur des classifieurs arborescents, appelé forêt, est généralement plus stable et plus performant [149][150].

A contrario, si il y a peu de diversité entre les arbres produits, l'intérêt de construire une forêt est limité. Mais il existe une stratégie très répandue qui permet de générer de façon artificielle de la diversité entre les arbres. L'idée clé des forêts aléatoires consiste à sélectionner aléatoirement pour la construction de chaque arbre des sous-ensembles d'instances et d'attributs et à n'apprendre que sur ces échantillons [151].

Parmi les pionniers des méthodes de forêts aléatoires, on peut citer Tin Kam Ho [152], Yali Amit et Donald Geman [153]. Puis, Leo Breiman et Adele Cutler ont proposé des extensions, et de nombreuses contributions théoriques qui ont permis d'améliorer la compréhension de ce modèle et d'autres contributions pratiques comme, par exemple, l'utilisation d'une forêt aléatoire pour mesurer l'importance des variables [151][154]. Un tel modèle a de nombreux avantages. Avec de nombreux arbres profonds, la forêt est généralement beaucoup plus stable et performante qu'un arbre et évite le sur-apprentissage.

En revanche, elle n'est pas interprétable comme un arbre.

Ajoutons que la démarche de randomisation a été poursuivie avec les forêts extrêmement aléatoires [155] qui, non seulement sélectionnent aléatoirement des attributs et des instances pour apprendre chaque arbre, mais génèrent également aléatoirement les conditions de séparation des instances dans les noeuds des arbres.

3.5 Rappels sur HOMER et RF-PCT

Dans la littérature multi-label non extrême, quelques méthodes de partitionnement ont été proposées comme MSSBoost [98], Rakel [156], HOMER [41], RF-PCT [40]. On présente ici le principe des deux dernières qui font partie des méthodes les plus populaires et qui obtiennent des performances parmi les plus compétitives. Les deux approches pionnières RF-PCT et HOMER ont montré l'intérêt des méthodes arborescentes pour l'apprentissage multi-label.

Random Forest of Predictive Clustering Trees (*RF-PCT*) Cette méthode [157] entraîne une forêt aléatoire de m_F arbres de décisions multi-label PCT. Chaque arbre PCT est construit comme un arbre d'instances/de décision standard. La condition de séparation à chaque noeud est monovariée (un attribut, un seuil) et vise à minimiser la variance des labels dans les deux sous-ensembles obtenus. Le critère de coupure correspond à la maximisation de la moyenne des critères de Gini pour chaque label. L'entropie ou le gain d'information peuvent aussi être utilisés.

Dans les feuilles, un vecteur contenant le vote majoritaire pour chaque label est stocké et utilisé dans la phase de prédiction. La diversité est introduite de façon standard comme dans les forêts aléatoires classiques. La prédiction de la forêt est la moyenne des prédictions de chaque arbre.

Hierarchy Of Multi-label classifiERs (HOMER) HOMER transforme le problème d'apprentissage multi-label en plusieurs problèmes successifs d'apprentissage. L'approche construit d'abord un arbre de labels par un k-clustering récursif équilibré jusqu'à ce que chaque feuille ne contienne qu'un unique label. Un classifieur multi-label est ensuite entraîné dans chaque noeud : son rôle est de prédire, à partir des attributs, le ou les enfants du noeud susceptibles de contenir dans leur descendance les labels pertinents de l'instance.

Pour les prédictions sur un nouveau vecteur d'attributs, le classifieur à la racine prédit les noeuds enfants pertinents. Puis, dans chacun de ces noeuds, les classifieurs associés prédisent à nouveau des enfants. Finalement les labels prédits sont ceux associés aux feuilles prédites. En apprentissage multi-label extrême, le nombre de labels est si important que l'arbre de labels devient très grand. Ainsi, le parcours intégral des arbres en

phase de prédiction est trop coûteux.

3.6 Les méthodes adaptées pour l'apprentissage multi-label extrême

HOMER et RF-PCT ne sont pas adaptés aux dimensions (10^4 à 10^7) de l'apprentissage multi-label extrême (XML). Dans cette section, nous nous restreignons donc aux approches arborescentes développées dans ce cadre.

Probabilistic Label Trees (PLT) - Extreme F-Measure Maximization using Sparse Probability Estimates [45] Cette approche propose un calcul rapide de probabilités parcimonieuses des labels à l'aide d'un arbre. Puis, par une stratégie de seuillage, elle optimise le F-Score dans le cadre d'un problème d'apprentissage multi-label extrême. Elle suit une stratégie très proche de celle du classifieur HOMER.

Pour construire un arbre, PLT construit récursivement des sous-ensembles de labels qui regroupent ceux qui co-occurrent dans les mêmes instances. Le décomposition est arrêtée lorsque les sous-ensembles contiennent un unique label. Puis, dans chaque noeud, un classifieur multi-label est entraîné pour pouvoir estimer les probabilités de suivre les différents chemins racine-feuille dans l'arbre conditionnellement aux attributs d'une instance donnée. Les chemins ayant pour extrémités les feuilles associées aux labels pertinents doivent avoir des probabilités élevées.

En phase de test, pour une instance donnée, le classifieur à la racine de l'arbre estime des probabilités de progression dans les noeuds enfants. De même, dans chacun de ces derniers (excepté ceux dont la probabilité est inférieure à un certain seuil), le classifieur associé estime la probabilité de progression dans les noeuds enfants. Le processus est répété jusqu'à atteinte des feuilles (labels) et les probabilités associées constituent les prédictions. Les prédictions sont parcimonieuses car le seuillage a empêché l'instance d'atteindre toutes les feuilles et de nombreux labels ont donc une probabilité nulle. Cette stratégie, qui limite le parcours de l'arbre, contribue à rendre PLT plus efficace que HOMER pour l'apprentissage extrême.

En fonction des SPE ("sparse probability estimation") fournies par l'arbre probabiliste PLT, trois stratégies ont été testées pour binariser les prédictions et maximiser la F-mesure. Deux optimisent le problème d'expected Utility Maximization et une est "online" :

— STO : Pour chaque label, on ordonne les instances en fonction des probabilités estimées par l'arbre. Cela est rapide car parcimonieux. On évalue le F-score obtenu en seuillant uniquement à la première instance, puis aussi la deuxième et ainsi de suite. On retient finalement le seuil optimal.

— FTA : Puisque la stratégie STO est exposée à un fort risque de sur-apprentissage, on limite le nombre de valeur testées à un ensemble prédéfini de valeurs de seuil.
— OFO "Online F measure Optimization" : La stratégie consiste à modifier le seuil à chaque nouvelle instance. De façon générale, on choisit comme seuil la moitié du dernier F-score calculé.

Label Partitioning For Sublinear Ranking (LPSR) [37] Cette méthode suppose qu'un algorithme multi-label ("scorer") a déjà été appris sur les données et que celui la permet donc de prédire un score pour chacun des labels étant donné les attributs. Dans ce cadre, LPSR a pour but d'accélérer et d'améliorer ces prédictions à l'aide d'un arbre. LPSR créé des partitions hiérarchiques d'instances et associe chaque sous-ensemble à un nombre réduit de labels. En prédiction, un modèle ("input partitionner") affecte l'instance à un sous-ensemble de la partition et le "scorer" est évalué uniquement pour les labels associés au sous-ensemble. LPSR construit d'abord le partitionnement basé sur les attributs g(x) ("input partitionner") avec un des algorithmes suivants : Hierachical k means, Hierarchical weighted k means, Embedded Hierarchical weighted k means où LSH. Ensuite, il sépare les instances d'apprentissage. Puis, il cherche à associer les sous-ensembles à un nombre limité de labels. Par résolution d'un problème d'optimisation, chaque sous-ensemble est associé aux labels "vérité terrain" des instances (x_i, y_i) qu'il contient et est dissocié des labels non pertinents dont le score donné par le "scorer" est plus important que celui des labels vérité terrain. La formulation du problème permet de maximiser la précision à k par sous-ensemble. Le problème, parallélisable par sous-ensemble dans la plupart des cas, est résolu par montée de gradient stochastique.

Multi-Label Random Forest (MLRF) [158] MLRF construit une forêt aléatoire d'arbres d'instances pour d'obtenir des feuilles où le nombre total de labels par union des instances est le logarithme du nombre de labels initial. Puis, dans chaque feuille, un classifieur avec une très faible complexité basé sur la randomisation est appris. Même si les performances d'un arbre sont limitées, la forêt permet de les améliorer.

Le principe de partitionnement est proche de celui de RF-PCT : au niveau d'un noeud, il consiste à trouver l'attribut et le seuil qui permet de séparer les instances en deux sous-ensembles qui optimise le critère de Gini pour tous les labels. A cause des informations manquantes (par exemple les labels pertinents mais non renseignés), le critère de Gini n'est pas idéal sur l'espace des labels d'origine. Pour obtenir de meilleures performances, les auteurs transforment l'espace des labels par complétion de matrice avec l'algorithme Graph-based SSL [159] (eq (2.13)).

$$\min_{Y'} = \frac{1}{2} Tr(Y'^T (I - L_n) Y') + \frac{\beta}{2} \|Y' - Y\|_F^2 \quad t.q. \|Y'\|_0 \leq nz \tag{2.13}$$

où L_n est le Laplacien normalisé du graphe de similarité des instances par rapport aux labels et où nz est la parcimonie souhaitée.

A la fin du processus, les instances dans Y' ont des scores différents pour les labels : score élevé pour les labels qu'ils avaient déjà dans Y, score élevé pour les labels qui co-occurrent souvent avec ceux qu'ils avaient dans Y, et score faible/nul pour les autres. Les auteurs utilisent les labels transformés Y' pour construire les arbres.

Fast eXtreme Multi-label Learning (FastXML) [36] FastXML est une forêt d'arbres d'instances multi-label avec un partitionnement basé sur une condition multivariée adapté au XML.

Le critère de partitionnement à chaque niveau est basé sur le $nDCG$ (normalized Discounted Cumulative Gain) qui est un critère sensible au classement des scores prédits pour les labels et qui conduit à la prédiction des labels vérité terrain avec un taux de confiance élevé. La séparation s'effectue sur l'ensemble des attributs selon un hyperplan w (w est le vecteur normal à l'hyperplan) : si le produit $x^T w$ est positif (resp. négatif), l'instance progresse à droite (resp gauche) dans l'arbre.

Pendant l'apprentissage du noeud, l'objectif est que les instances à gauche (resp. droite) soient bien prédites, au sens du $nDCG$, par un vecteur r^- (resp r^+). La construction de l'arbre pourrait être posée comme un problème sur w uniquement, mais FastXML pose le problème souple sur (w, r^+, r^-, δ) où w est le séparateur, r^- et r^+ sont les labels prédits à gauche et droite, et δ est le partitionnement de taille n (δ_i vaut $+1$ si l'instance est à droite et -1 sinon). L'objectif dans un noeud est donc le suivant :

$$\begin{aligned} \min \|w\|_1 + \sum_{i=1}^{n} C_\delta(\delta_i) log(1 + e^{-\delta_i w^T x_i}) \\ -C_r \sum_{i=1}^{n} \frac{1}{2}(1+\delta_i)\mathcal{L}_{nDCG@d_y}(r^+, y_i) \\ -C_r \sum_{i=1}^{n} \frac{1}{2}(1-\delta_i)\mathcal{L}_{nDCG@d_y}(r^-, y_i) \end{aligned} \tag{2.14}$$

— Le premier terme rend w parcimonieux.
— Le second terme tente de prédire le partitionnement δ avec l'hyperplan w.
— Le troisième terme minimise le critère $nDCG$ entre les prédictions r^+ et les instances dans la feuille droite.
— Le quatrième terme minimise le critère $nDCG$ entre les prédictions r^- et les instances dans la feuille gauche.

Le problème est résolu alternativement sur $r\pm$ puis δ puis w. Les coefficients C permettent de donner une importance différente à chacun des termes mais en pratique, ils sont tous fixés à 1.

Après apprentissage, le modèle sauvegarde les hyperplans w dans les noeuds et les

prédictions $r\pm$ dans les feuilles. En phase de prédiction, les hyperplans sont utilisés récursivement jusqu'à ce que l'instance de test atteigne une feuille de l'arbre. Chaque arbre prédit les k (=20 dans les expériences) meilleurs labels du vecteur de prédiction $r\pm$ de la feuille atteinte. La prédiction de la forêt est la moyenne des prédictions des arbres.

PFastreXML [38] Cet algorithme est une extension de FastXML qui vise à pallier deux difficultés majeures de l'apprentissage multi-label extrême :

— La forte quantité d'information manquante dans les données : il y a certitude sur la pertinence des labels présents mais non certitude de la non pertinence des labels absents.

— La longue traîne de la distribution des labels.

Dans l'article de PFastReXML [38], les auteurs proposent d'abord une nouvelle famille de fonctions de perte (Propensity-based nDCGk ou P@k) qui prennent mieux en compte les labels de la longue traîne et de l'information manquante et, par rapport à FastXML, ils proposent d'optimiser ces nouvelles fonctions de perte à chaque noeud. De plus la prédiction dans les feuilles n'est pas un vecteur de label prédéfini ; elle dépend des attributs de l'instance de test et elle est calculée avec un "reranker" de Rocchio [160]. Ce "reranker" a également tendance à donner un meilleur poids aux labels plus rares de la longue traîne.

Les fonctions de pertes proposées sont des extensions de fonctions de perte classiques (précision, nDCG, ...) normalisées par la propension. Les propensions sont classiquement utilisées dans la littérature pour plusieurs tâches : éviter le sur-apprentissage, gérer un "drift" entre le test et l'apprentissage, gérer un manque d'informations, donner de l'importance aux labels rares (normalisation de la perte par l'inverse de la fréquence du label comme TF ou TF-IDF).

La propension utilisée ici rend compte de la fraction de labels observés sur la totalité des labels pertinents pour une instance donnée. Dans l'exemple du jeu de données Wikipedia, les labels correspondent aux quelques catégories associées à un article (5 catégories environ sur 1 million de possibilités). Il est donc évident que l'annotateur n'a pas passé en revue toutes les catégories possibles avant de proposer ses labels. Ainsi, il y a beaucoup plus de labels pertinents que de labels observés pour une instance. Les auteurs dans [38] proposent d'évaluer cette propension sur Wikipedia et Amazon en parcourant la hiérarchie des labels qui est disponible sur ces sites (note : ces informations externes au jeux de données initiaux). Pour un label observé, les labels parents sont automatiquement considérés comme pertinents. A l'aide de ces informations, la propension p_j du $j^{\text{ème}}$ label est calculée pour tout j. Après avoir calculé les propensions de tous les labels, les auteurs ont observé empiriquement un lien direct avec le nombre d'occurrence des labels dans le jeu de données. Cette observation les a menés à modéliser la propension d'un label comme une fonction du nombre de fois n_j où le label (d'indice j) est observé dans le jeu de données :

$$p_j = \frac{1}{1 + ce^{-alog(n_j+b)}} \tag{2.15}$$

où $c = log(n-1) \times (b+1)^a$ et où a et b sont estimés empiriquement. Dans les cas où on ne peut pas estimer a et b car la hiérarchie n'est pas disponible, les auteurs préconisent les valeurs par défaut $a = 0.55$ et $b = 1.5$.

La fonction de propension est ensuite utilisée pour normaliser la perte associée à chaque instance. Certains modèles normalisent par $n_j^{-\beta}$ où $log(\frac{d_y}{n_j})$.

Le "reranker" de PFastReXML utilisé dans les feuilles prédit la probabilité suivante pour chaque label :

$$p(y_{ij}|x_i) = \frac{1}{1 + v_{ij}^{2y_{ij}-1}} \tag{2.16}$$

où $v_{ij} = e^{\frac{\gamma}{2}\|x_i - \mu\|^2}$

Le paramètre μ_j est estimé pour chaque label j pendant l'apprentissage. Il pourrait être obtenu par descente de gradient mais la complexité est trop important lorsqu'il y a des millions de labels. La solution approchée suivante a donc été privilégiée :

$\mu_j = \frac{\sum y_{ij} x_i}{\sum y_{ij}}$

La probabilité finale prédites pour les labels est le compromis entre celle calculée par la stratégie de FastXML et celle calculée par (2.16).

4 Solution 3 : méthodes complexes rendues efficaces avec des astuces d'optimisation et d'implémentation

Sur les données extrêmes, certains chercheurs ont tenté d'apprendre sans réduire la dimension ni partitionner récursivement le problème. Les modèles considérés associent donc directement un très grand nombre d'attributs à un très grand nombre de labels. Comme expliqué dans le chapitre précédent, peu de modèles ("baseline labels fréquents", "bayesien naif") sont capables de le faire sur des ordinateurs standards mais ils n'obtiennent pas de bonnes performances prédictives. Au lieu d'appliquer ces modèles simples, certains choisissent plutôt d'apprendre des modèles très complexes (mémoire et temps) pour les dimensions XML en recourant à des stratégies d'accélération des calculs comme la parallélisation sur des supercalculateurs ou à des astuces d'optimisation. Par exemple, les approches de type one-vs-rest (aussi appelé one-vs-all) ne passent pas à l'échelle en XML car elles consistent à apprendre un modèle par label. Mais, en parallélisant l'apprentissage sur un millier de machines, des chercheurs sont parvenus à évaluer un one-vs-rest SVM regularisé avec la norme L1 sur le jeu de données Wiki-LSHTC [42]. Les résultats sont bons mais l'approche est difficilement exploitable dans de nombreuses applications car elle nécessite des moyens de calculs importants.

Pour des raisons de moyens de calculs nécessaires, peu d'approches de la littérature ont exploré à ce jour des solutions de ce type pour atteindre des performances compétitives. A notre connaissance, il en existe trois seulement : PD-Sparse, PPD-Sparse et DISMEC.

PD-Sparse - Primal-Dual Sparse PD-Sparse apprend un modèle linéaire ($\widehat{y} = W^T x$) parcimonieux (W creux) dont l'objectif est, pour chaque instance (x_i, y_i), que le score dans $\widehat{y_i}$ associé aux labels présents dans y_i soit plus grand avec une marge que le score des labels absents ("separation Ranking Loss" [161]) :

$$\min_W L(\widehat{y_i}, y_i) = \min_W \max_{j \in \mathcal{N}(y_i), k \in \mathcal{P}(y_i)} (1 + \widehat{y_{ij}} - \widehat{y_{ik}})_+ \tag{2.17}$$

où $\mathcal{P}(y_i)$ (resp. $\mathcal{N}(y_i)$) désigne l'ensemble des indices des labels présents (resp. absents) dans y_i. PD-Sparse résoud le problème régularisé avec Elastic Net [137].

Résolu avec une approche standard (gradient stochastique), un tel modèle est trop complexe (mémoire/temps) pour passer à l'échelle. Pour accélérer les calculs, le problème est converti dans une version duale où les variables duales sont les prédictions, les variables primales étant les paramètres de la projection. De plus, une stratégie de sélection de labels "actifs" permet d'éviter de calculer le gradient complet et de faire de lourdes mises à jour à chaque itération. Enfin, pour optimiser la mémoire et les calculs, des régularisations permettent de rendre les paramètres parcimonieux dans l'espace dual et dans l'espace primal. Puisqu'ils restent parcimonieux tout au long de l'apprentissage, seuls les paramètres non nuls sont stockés en mémoire grâce à des tables de hashage efficaces. La résolution s'effectue une approche FBCFW (Fully Corrective Block Coordinate Frank Wolfe) [162].

DiSMEC - Distributed Sparse Machines for Extreme Multi-label Classification DisMEC apprend un modèle séparateur linéaire avec marge w_j pour chaque label $j \in \{1, ..., d_y\}$ dont l'objectif est d'avoir $w_j^T x_i \geq 1$ si $y_{ij} = 1$ (label présent) et $w_j^T x_i \leq -1$ si $y_{ij} = 0$ (label absent). Avec une régularisation en norme L_2, le problème se formule comme suit :

$$\min_{w_j} \|w_j\|_2^2 + C \sum_{i=1}^{n} max(0, 1 - y_{ij} \times w_j^T x_i)^2 \tag{2.18}$$

Ce problème est résolu par une méthode de Newton à région de confiance [141]. L'implémentation des auteurs est doublement parallélisée : le problème est découpé par paquets de 1000 labels et les modèles associés à ces 1000 labels sont appris en parallèle sur des machines de 32/64 coeurs. Les modèles w_j sont finalement rendus parcimonieux en appliquant un seuillage dur (à 0.01). Cela permet d'avoir un modèle beaucoup moins coûteux en mémoire, plus rapide pour effectuer des prédictions, et plus robuste.

PPDSparse - Parallel Primal-Dual Sparse PPDSparse a été proposé par les mêmes auteurs que PDSparse suite au constat que la fonction de coût du précédent classifieur est non séparable par rapport aux labels. Effectivement, elle dépend du score de tous les labels à la fois et cela a deux inconvénients majeurs : pas de possibilité de parallélisation et coût en mémoire du modèle trop élevé.

Une fonction de coût plus simple suivant la stratégie "1-vs-all" comme dans DISMEC permet non seulement d'avoir de très bonnes performances mais est également parallélisable. Dans PPDSparse, les auteurs choisissent donc d'optimiser cette fonction de coût (voir (2.18)) et de démontrer les mêmes propriétés de parcimonie duale et primale qu'ils avaient dans PD-Sparse. Les sous-modèles associés à chaque label peuvent être appris en parallèle et similairement à PDSparse, une sélection des instances actives permet d'accélérer encore la résolution. Parallélisé sur 100 coeurs, ce modèle est nettement plus rapide que PD-Sparse et DISMEC.

5 Conclusion

Dans un contexte où l'apprentissage multi-label suscite une attention croissante, les chercheurs tentent de répondre aux besoins actuels en matière de traitement de données de grande dimension. Nous avons présenté et structuré les trois solutions de la littérature : réduction de dimension, méthodes arborescentes, astuces d'optimisation/d'implémentation.

Réduction de dimension Pour faire face à la complexité des problèmes, un grand nombre de méthodes de réduction de dimension multi-label ont été publiées au cours des dernières décennies. Ces publications enrichissent grandement la littérature mais il reste difficile de les comparer, de choisir la plus pertinente pour le problème à résoudre et de déterminer le travail qu'il reste à faire dans le domaine. Notre analyse tente de fournir ces éléments. Nous avons proposé une vue d'ensemble des méthodes à travers une typologie unificatrice pour démêler les liens entre les différentes méthodes. Elle repose sur trois critères majeurs. Le premier critère est l'espace que les méthodes réduisent (espace des attributs, espace des labels ou les deux). Les approches de réduction de l'espace des attributs sont répandues pour le moment mais avec l'intérêt croissant pour l'apprentissage multi-label extrême, les méthodes qui réduisent également la dimension de l'espace des labels sont en plein essor. Le deuxième critère distingue les méthodes qui réduisent un espace en tenant compte des informations portées par l'autre de celles qui effectuent la réduction de manière indépendante. Contrairement à il y a quelques années, les méthodes dépendantes prédominent aujourd'hui : en préservant les liens entre les attributs et les labels, elles sont plus efficaces pour la tâche d'apprentissage. Le troisième critère est la présence/absence d'un couplage entre le classifieur et la stratégie de réduction de dimension. Aujourd'hui, les deux scénarios sont très déséquilibrés et la grande majorité des approches

ne sont pas couplées au classifieur. Outre ces aspects structurants majeurs, les méthodes diffèrent pour deux composants supplémentaires : le type de transformation (implicite, explicite, linéaire, non linéaire) qu'elles effectuent et les contraintes/régularisations qu'elles imposent à la résolution du problème. Bien que ces différences ne distinguent pas les approches par leur nature même, elles peuvent avoir un impact important sur leur efficacité et leur capacité de déploiement dans des applications réelles.

Méthodes arborescentes Nous avons présenté deux types de méthodes arborescentes. Celles qui construisent des arbres d'instances et celles qui construisent des arbres de labels. De façon générale, ces méthodes découpent récursivement l'ensemble des labels (resp. des instances) pour maximiser un objectif donné. Les méthodes hiérarchiques ont de nombreux avantages pour la tâche d'apprentissage multi-label extrême : opérations décomposées en sous-tâches simples et rapides, prédiction en temps logarithmique, parallélisation facile, décisions récursives permettant une grande expressivité. Mais elles peuvent néanmoins présenter quelques défauts. Le partitionnement limite de façon définitive la qualité d'un classifieur local au sein d'un sous-ensemble. Et, puisque le partitionnement est hiérarchique, une erreur dans les premières étapes peut fortement pénaliser la qualité de l'algorithme. En revanche, des solutions existent pour limiter ce problème comme la déclinaison de l'arbre en forêt (aléatoire ou non) ou l'application de pondérations dans les noeuds selon leur qualité prédictive sur un ensemble de validation.

Astuces d'optimisation/d'implémentation Cette troisième stratégie est la moins répandue. Plutôt que de réduire la dimension ou de découper récursivement le problème, ces méthodes utilisent d'autres astuces pour passer à l'échelle. Les astuces sont la parallélisation sur des supercalculateurs, la conversion primal/dual du problème d'optimisation ou les régularisations permettant d'assurer un modèle creux. Même si ces approches sont difficilement exploitables dans de nombreuses applications car elles nécessitent des moyens de calculs importants, elle permettent d'obtenir des performances prédictives intéressantes et de mettre au défi les autres types de solutions.

Chapitre 3

Réduction de dimension : comparaison des approches et nouvelle proposition

Sommaire

1 Introduction

L'état de l'art en réduction de dimension présenté dans le chapitre 2 souligne la grande variabilité dans la manière d'aborder la problématique sur les données d'apprentissage multi-label. Dans la littérature, où chaque auteur a recourt à sa propre formulation, cette variabilité est un obstacle à une bonne compréhension des similitudes et des différences entre les approches. Dans ce chapitre, nous proposons une comparaison globale des approches de réduction de dimension selon deux angles complémentaires (théorique et expérimental) :

— La première direction, présentée dans la **section 2**, vise à établir des liens formels entre les problèmes de réduction de dimension proposés dans la littérature. Pour cela, nous explicitons deux formulations génériques du problème de réduction (2.1). La première, étroitement liée à un problème spectral, permet de décrire plus de la moitié des approches existantes. La seconde, plus générale, couvre une grande

majorité des cas rencontrés. Ces formulations aident à identifier les invariants entre les méthodes et à mieux cerner les composantes des problèmes associés qui les distinguent.

— La deuxième direction, présentée dans la **section 3**, vise à rendre compte des performances expérimentales relatives de l'ensemble de ces méthodes. Cette étude n'est pas basée sur nos propres implémentations et un banc d'expériences mais sur une méta-analyse originale des études de la littérature présentant chacune des résultats de comparaisons entre un sous-ensemble des méthodes. L'analyse effectue d'abord un état des lieux des résultats disponibles puis, en s'inspirant des travaux sur le consensus en agrégation de préférences, elle extrait des relations de domination statistiquement significatives entre des paires d'algorithmes. Les résultats permettent d'identifier, pour différentes familles d'algorithmes, les méthodes les plus performantes au regard des résultats publiés dans la littérature.

En complément de ces deux analyses, nous proposons dans la **section 4** une nouvelle approche qui explore une stratégie qui n'apparait pas, ou peu, dans notre panorama de la combinatoire des différentes stratégies publiées. L'état de l'art a souligné l'intérêt du couplage entre la méthode de réduction de dimension et le classifieur auquel elle est combinée. Trois cas existent :

(i) dans la majorité des cas le couplage est inexploité : réduction indépendante du classifieur ;

(ii) le couplage est indirect : la stratégie de réduction est conçue pour intuitivement améliorer les performances du classifieur mais sans garantie ;

(iii) le couplage est direct : l'apprentissage de la méthode de réduction est intégré dans l'apprentissage du classifieur pour optimiser spécifiquement ses performances mais avec des classifieurs linéaires peu représentés dans l'état de l'art des classifieurs multi-label.

Nous avons donc cherché à évaluer un couplage direct avec le classifieur ML-kNN qui fait partie des méthodes multi-label les plus efficaces en ciblant une mesure de performance à optimiser. Le problème est ici que le lien entre les performances de ML-kNN et la transformation qui réduit la dimension est difficile à dériver. La méthode proposée, ML-ARP (Multi-Label Adaptative Random Projection), optimise donc la transformation avec une recherche à voisinage variable. Les performances obtenues sont comparées avec les méthodes standards de réduction de dimension multi-label dont MDDM qui compte parmi les plus efficaces dans le cadre multi-label non extrême.

2 Comparaison théorique : une formulation générique

La vue d'ensemble des stratégies de réduction dans les tableaux 1a et 1b du chapitre 2 permet de rendre compte de l'originalité des différentes approches et donne quelques indications sur les intuitions sous-jacentes des objectifs qu'elles visent à atteindre. Ces

objectifs sont formulés par des problèmes d'optimisation sous contraintes. On peut alors se poser la question suivante : malgré la forte variabilité apparente dans les stratégies ciblées par les auteurs, quelles similarités et différences peut-on retrouver dans la formulation mathématique finale du problème ? Pour contribuer à cette réflexion, nous proposons deux formulations génériques.

2.1 Le cadre basique

Comme nous le montrons dans le tableau 3.1, un grand nombre de problèmes peuvent être décrits comme suit :

$$\begin{array}{ll} \underset{U}{\text{optimiser}} & tr(U^T A_{XY} U) \\ \text{sous} & U^T B_{XY} U = I \end{array} \tag{3.1}$$

où :

1. A_{XY} et B_{XY} sont des matrices qui dépendent de X et Y.
2. en fonction de l'espace réduit et du type de transformation, le paramètre U est l'une des matrices suivantes : X', Y', P_x, ou P_y.
3. l'objectif d'optimisation est un objectif de minimisation ou de maximisation.

Les problèmes exprimés par (3.1) peuvent être résolus avec une décomposition en valeurs propres. En effet, il est bien connu que, avec la méthode du Lagrangien [164], le problème (3.1) équivaut à optimiser λ dans le problème de valeurs propres généralisé suivant :

$$A_{XY} u = \lambda B_{XY} u \tag{3.2}$$

La solution U de (3.1) dans le cas de la maximisation (respectivement de la minimisation) est donc la matrice des vecteurs propres associés aux plus grandes (respectivement aux plus petites) valeurs propres de (3.2). Dans le cas fréquent où la matrice A_{XY} est symétrique positive, les vecteurs propres de A_{XY} peuvent aussi être calculés par une décomposition en valeurs singulières [165] de la "racine carrée" R_{XY} de A_{XY} définie par $R_{XY}^T R_{XY} = A_{XY}$.

Malgré sa solution élégante, la décomposition en valeurs propres (3.2) a une forte complexité : de l'ordre de n^2 pour la complexité spatiale et n^3 pour la complexité temporelle sur une matrice $n \times n$ [166]. Pour passer à l'échelle sur des matrices de grande dimension, différentes approches sont utilisées : les techniques de décomposition rapide (par exemple Jacobi [167] et décomposition QR [168]), l'approximation des plus grandes valeurs propres (algorithme de la puissance itérée [169], méthode de Lanczos [170]), les

Tableau 3.1 – Lien entre les méthodes de réduction de dimension de l'état de l'art et le cadre basique de formulation (3.1)

Méthode	U	A_{XY}	Dépendance	B_{XY}	Contrainte	Année	Réf.
PCA	P_x	X^TX	Non	I	Transfo. Ortho.	1901	[64]
CCAx	P_x	$X^TY(Y^TY)^{-1}Y^TX$	Oui	X^TX	Espace non corr.	1936	[48][118]
PLS	P_x	X^TYY^TX	Oui	I	Transfo. Ortho.	1983	[100]
KPCA	P_x	$\phi(X)^T\phi(X)$	Non	I	Transfo. Ortho.	1997	[126]
LPP	P_x	X^TLX	Non	XDX^T	Espace non corr.	2004	[66]
MLSI	P_x	$X^T((1-\theta)X^TX+\theta Y^TY)X$	Oui	X^TX	Espace non corr.	2005	[72]
OPLS	P_x	X^TYY^TX	Oui	X^TX	Espace non corr.	2006	[101]
RPCA	P_x	X^TX	Non	I	Transfo. Ortho.	2006	[97]
KDA	P_x	$S_w^-1S_b$	Oui	I	Transfo. Ortho.	2007	[163]
OLPP	P_x	X^TLX	Non	I	Transfo. Ortho.	2007	[99]
ONPP	P_x	$X^T(I-W)(I-W^T)X$	Non	I	Transfo. Ortho.	2007	[99]
HSL	P_x	XL_nX^T	Non	X^TX	Espace non corr.	2008	[71]
MLLS	P_x	$S_2^{-1}S_1$	Oui	I	Transfo. Ortho.	2008	[102]
MLSVM	P_x	$(X^TX)^\dagger X^TYY^TX$	Oui	I	Transfo. Ortho.	2009	[93]
MDDM	P_x	X^THYY^THX	Oui	I	Transfo. Ortho.	2010	[54]
MLDA	P_x	$S_w^-1S_b$	Oui	I	Transfo. Ortho.	2010	[104]
SOLPP	P_x	$X^T(I-W)(I-W^T)X$	Oui	I	Transfo. Ortho.	2010	[94]
SSDR-MC	X'	$X^T(I-W)(I-W)^TX$	Oui	X^TX	Espace non corr.	2010	[103]
rCCAx	P_x	$X^TY(Y^TY)^\dagger Y^TX$	Oui	X^TX	Espace non corr.	2011	[118]
CPLST	P_y	$Y^TH(XX^T)^\dagger HY$	Oui	I	Transfo. Ortho.	2012	[114]
PLST	P_y	Y^TY	Non	I	Transfo. Ortho.	2012	[74]
DMLDA	P_x	$X^THYW^{-1}Y^THX$	Oui	I	Transfo. Ortho.	2013	[105]
IDSRx	P_x	X^TX	Non	I	Transfo. Ortho.	2013	[88]
IDSRy	P_y	Y^TY	Non	I	Transfo. Ortho.	2013	[88]
VPCME	P_x	$S_C-\theta S_M$	Oui	I	Transfo. Ortho.	2013	[106]
FaIE	Y'	$YY^T+\theta X(X^TX)^{-1}X^T$	Oui	I	Transfo. Ortho.	2014	[116]
FaIE Linear	P_y	$Y^T(Y^TY+\theta X(X^TX)^{-1}X^T)Y$	Oui	Y^TY	Espace non corr.	2014	[116]
HOPLS	P_x	$X^T(YY^T+\theta S)X$	Oui	X^TX	Espace non corr.	2014	[107]
DMLR	P_y	$Y^T(I+\theta HXX^TH)Y$	Oui	I	Transfo. Ortho.	2015	[78]
SSMDDM	P_x	X^THYY^THX	Oui	I	Transfo. Ortho.	2015	[108]
MVMD	P_x	$(1-\theta)X^TX+\theta X^THYY^THX$	Oui	I	Transfo. Ortho.	2016	[73]

Notations :
$M^\dagger$: pseudo-inverse de la matrice M
L (resp. L_n) : Laplacien (resp. normalisé) d'un graphe
ϕ : noyau
W, S_C, S_M,S_b, S_w, S : matrice de similarité ou de poids de paires
θ, α , β : paramètres de compromis
$H=(\delta_{ij}-\frac{1}{N})_{ij}$ où δ est le delta de Kronecker
$S_1=I-\alpha T^{-1}$ et $S_2=T^{-1}X^TYY^TXT^{-1}$ où $T=\frac{1}{N}X^TX+(\alpha+\beta)I$

méthodes d'algèbre randomisée comme le sketching [171] (par exemple dans PCA randomisé **??** ou Rembrandt [76]). En sus, une reformulation du problème (3.1) sous une forme de moindres carrés est également très populaire pour pouvoir utiliser des méthodes d'optimisation numérique variées. C'est le cas, par exemple, dans la version moindres carrés de CCA [118] ou de LDA [172]. En effet, la forme initiale des moindres carrés $\min_U \min_M \|R_{XY}-MU^T\|_F^2$, où R_{XY} est la racine carrée de A_{XY}, est équivalente à $\max_U \mathrm{tr}(U^TA_{XY}U)$ avec la contrainte $U^TU=I$.

Par exemple, considérons la formulation classique de PCA : $\max_{P_x} \mathrm{tr}(P_x^T(X^TX)P_x)$. Soumis à la contrainte $P_x^TP_x=I$, elle peut être reformulée en un problème de minimisation d'erreur quadratique de reconstruction $\min_{X',P_x}\|X-X'P_x^T\|_F^2$ avec de simples manipulations algébriques. La contrainte forte $U^TU=I$ est parfois remplacée par une plus simple régularisation L_2 sur U.

Notons qu'une partie des méthodes exprimées avec le cadre de base (3.1) nécessite

des décompositions spectrales de graphes [173][174]. Elles suivent une procédure en deux étapes : (i) construire un graphe qui lie les instances avec une mesure de proximité (par exemple une distance sur les labels) et (ii) projeter les instances dans un espace réduit en préservant la structure du graphe. La transformation est calculée par une décomposition spectrale du graphe (cela équivaut au problème (3.1) où A_{XY} est le laplacien normalisé du graphe et B_{XY} la matrice identité).

2.2 Vers un cadre généralisé...

L'équivalence entre le cadre de base (3.1) et la formulation des moindres carrés met en évidence à la fois sa flexibilité et ses limites. Les régularisations L_1 [175], les fonctions de perte multi-label autres que l'erreur quadratique moyenne [32] et beaucoup d'autres éléments ne peuvent pas être exprimés comme une trace de matrice. Une tentative de généralisation a été proposée dans LEML [43]. Le problème est posé comme un problème de minimisation du risque empirique (ERM) [176] qui ne nécessite pas de fonction de perte spécifique ni de régularisation spécifiée. Notons $h(x; Z) : x \mapsto \widehat{y}$ le modèle d'apprentissage du paramètre Z, par $l(y, \widehat{y}) = l(y, h(x; Z))$ la fonction de perte entre le vecteur de labels prédit $\widehat{y}$ et le vrai vecteur de labels y, et $r(Z)$ la régularisation des paramètres du modèle. Le problème de minimisation du risque empirique avec contrainte de rang faible est exprimé comme suit :

$$\begin{aligned} \widehat{Z} = \underset{Z}{\operatorname{argmin}} \quad & \sum_{i=1}^{N} \sum_{j=1}^{d_y} l(Y_{ij}, h^j(x_i; Z)) + \lambda r(Z) \\ \text{subject to} \quad & rank(Z) \leq k \end{aligned} \tag{3.3}$$

Remarquons que cette formulation diffère du problème ERM classique : la contrainte de rang ajoutée sur $Z \in \mathbb{R}^{d_x \times d_y}$ entraîne une réduction de dimension [177].

La formulation (3.3) couvre une grande partie des méthodes de la littérature mais pour inclure les cas restants non couverts, nous proposons une formulation générique de la fonction objectif qui est une combinaison additive des ingrédients essentiels rencontrés dans la typologie de réduction de dimension multi-label :

$$\begin{aligned} J(X', Y', Z_x, Z_y, Z_{xy}) = & \alpha_x e_x(X', X, Z_x) \\ & + \alpha_y e_y(Y', Y, Z_y) \\ & + \alpha_{xy} e_{xy}(X', X, Y, Y', Z_{xy}) \\ & + \alpha_p p(X', Y') \\ & + \alpha_r r(X', Y', Z_x, Z_y, Z_{xy}) \end{aligned} \tag{3.4}$$

où :

— e_x est une erreur de reconstruction entre X et sa version réduite X'.

— e_y est une erreur de reconstruction entre Y et sa version réduite Y'.
— e_{xy} est une erreur conjointe entre X, Y, X' et Y' qui peut, sans s'y limiter, exprimer l'erreur de prédiction ou la corrélation croisée entre labels et attributs.
— r est une régularisation des paramètres.
— Z_x, Z_y, Z_{xy} sont les paramètres des fonctions de réduction et des classifieurs.
— p exprime des propriétés supplémentaires imposées aux deux espaces réduits.

L'erreur de reconstruction e_x peut être exprimée avec une erreur d'encodage l_{x1} (reconstruction de X' à partir de X) et une erreur décodage l_{x2} (reconstruction de X à partir de X') :

$$\begin{aligned} \alpha_x e_x(X', X, Z_x) = \alpha_{x1} l_{x1}(X', f_{x1}(X, Z_x)) \\ + \alpha_{x2} l_{x2}(X, f_{x2}(X', Z_x)) \end{aligned} \tag{3.5}$$

où les fonctions f sont des modèles à paramètres. Cela est également valable pour e_y et e_{xy}.

Dans la plupart des cas, la régularisation r peut être décomposée additivement :

$$\begin{aligned} \alpha_r r(X', Y', Z_x, Z_y, Z_{xy}) = \alpha_{r_1} r_1(X') + \alpha_{r_2} r_2(Y') \\ + \alpha_{r_3} r_3(Z_x) + \alpha_{r_4} r_4(Z_y) \\ Y + \alpha_{r_5} r_5(Z_{xy}) \end{aligned} \tag{3.6}$$

Dans (3.4), (3.5) et (3.6), les constantes α sont des poids qui permettent des compromis entre les différents composants du problème.

Toutes les formes de (3.4) peuvent être traitées avec des méthodes d'optimisation numérique adaptées [178][179]. Considérant la convexité, la régularité, l'ordre, la différentiabilité et le conditionnement de la formulation, le problème peut être reformulé (relaxation convexe [180], conversion primal/dual [181], préconditionnement [182]) et la résolution peuvent être effectuée avec une variante adaptée de la descente de gradient [183][184], une descente par coordonnées [185] ou des algorithmes d'ordre supérieur tels que la méthode de Newton [186] ou l'algorithme de Frank Wolfe [187]. Les problèmes contraints sont généralement résolus avec une méthode Lagrangienne [188], avec l'une de ses diverses extensions (par exemple les Lagrangiens Augmentés comme ADMM [189][44]) ou avec une descente de gradient projetée. Le choix du couple formulation/résolution est essentiel : il affecte les complexités spatiales et temporelles des calculs et la qualité de la convergence vers la solution.

Pour être complet, signalons que deux familles de méthodes de réduction de dimension explorées pour l'apprentissage multi-label atteignent les limites de la formulation générique. La première comprend des approches basées sur des modèles de mélange [121] et résolues avec l'algorithme EM où l'une de ses variantes [190][191]. Le second comprend les stratégies de type ensemble (bagging [106][77] et boosting [98]) où plusieurs transfor-

mations de réduction sont apprises sur des bootstraps et agrégées selon deux stratégies principales : (i) chaque transformation produit son propre espace réduit et les espaces réduits sont agrégés en un espace réduit global et le classifieur est entraîné sur cet espace ou (ii) un classifieur est entraîné sur chaque espace réduit et les prédictions de tous les classifieurs sont agrégées.

3 Comparaison expérimentale : méta-analyse

Les formulations génériques précédentes permettent d'identifier explicitement les différents ingrédients impliqués dans les différents algorithmes proposés dans la littérature et d'aider à comprendre leurs points communs et leurs différences. Cependant, en pratique, une question persiste : quelles sont les approches les plus efficaces ? Il est difficile de répondre car seules des comparaisons partielles sont généralement rapportées dans les articles et, à notre connaissance, il n'existe aucune étude expérimentale comparant toutes les approches présentées dans le tableau 2.2 du chapitre 2. De plus, les implémentations sont très diverses et pour certaines approches, les codes source ou les paramètres ne sont même pas disponibles. Par conséquent, une comparaison normalisée impliquerait un recodage de tous les algorithmes et une nouvelle batterie de tests sur un cadre unifié qui reste à définir dans la communauté de recherche. Compte tenu du grand nombre d'algorithmes à prendre en compte, cela nécessiterait un effort considérable et, par conséquent, l'exploitation des résultats des travaux de recherche publiés existants apparaît comme une alternative plus réaliste. Les protocoles expérimentaux (jeux de données, classifieurs, mesures de performance, etc ...) variant d'une publication à l'autre, nous proposons ici une nouvelle méthodologie de méta-analyse.

Souvent définie comme "l'analyse statistique d'une grande collection de résultats provenant d'études individuelles dans le but d'intégrer les découvertes", la méta-analyse a connu un développement croissant depuis ses travaux pionniers dans les années 30 [192][193]. L'un de ses domaines privilégiés est la médecine où l'agrégation des informations disponibles est nécessaire pour rendre les décisions les plus rationnelles possibles. En informatique, cette approche est encore inhabituelle car la grande majorité des chercheurs préfèrent comparer leurs propres approches avec un sous-ensemble restreint de celles existantes sur un ensemble de données qu'ils utilisent habituellement mais les premières tentatives (par exemple [194][195][196]) semblent prometteuses.

Dans cette section, nous cherchons à identifier les approches de réduction de dimension utilisées dans l'apprentissage multi-label pour lesquelles plusieurs éléments de preuve montrent leur domination sur les autres : ces approches obtiennent statistiquement les meilleures performances dans les résultats publiés dans des conférences et revues internationales. Comme il est bien connu dans l'apprentissage multi-label que les performances peuvent être évaluées avec un large éventail de mesures, nous extrayons les méthodes

pertinentes pour chacune des mesures de qualité les plus fréquemment utilisées et indépendantes. Nous présentons dans un premier temps une analyse descriptive des occurrences et co-occurrences observées dans la littérature pour les algorithmes et les mesures, puis nous détaillons le processus d'extraction des approches dominantes, et enfin nous discutons des résultats obtenus.

3.1 Méthodologie

De tous les articles référencés dans le tableau 2.2, nous avons retenu un corpus $\mathcal{C}$ de 27 articles - marqués en gras - qui présentent des résultats pertinents et exploitables pour une méta-analyse. Plus précisément, nous avons d'abord extrait les 32 articles qui comparent au moins deux méthodes, puis nous avons supprimé les 5 articles dont les résultats ne sont donnés que sur des graphiques car ils sont difficiles à exploiter.

3.1.1 L'ensemble d'algorithmes considéré

Notons $\mathcal{A}$ l'ensemble des 42 algorithmes qui apparaissent dans les papiers sélectionnés du corpus $\mathcal{C}$. Les comparaisons par paires publiées peuvent être décrites par un multigraphe G_c : les sommets représentent les algorithmes de $\mathcal{A}$ et une arête relie deux algorithmes lorsqu'ils sont comparés dans un article. Dans la représentation du multigraphe (Figure 3.1), le diamètre d'un sommet est proportionnel à la fréquence d'apparition de l'algorithme associé dans les comparaisons et l'ensemble des arêtes entre une paire de sommets est représenté par une seule arête dont la largeur est proportionnelle à la cardinalité.

La figure obtenue est très différente de celle d'un graphe complet qui serait le cas idéal, mais loin de la réalité, où chaque algorithme est comparé à tous les autres dans de nombreuses expériences. Cependant, il met en évidence deux communautés qui correspondent à deux familles d'algorithmes qui ont été étudiés principalement séparément. Cela confirme que les comparaisons publiées ont été faites sur des sous-ensembles d'algorithmes qui partagent des propriétés communes. La première communauté C_1 regroupe les approches qui réduisent la dimension de l'espace des attributs et la seconde C_2 regroupe les algorithmes de réduction de l'espace de labels ou des deux espaces, y compris ceux développés dans le contexte d'apprentissage multi-label extrême. De plus, deux sommets (CCA et MDDM) apparaissent à l'intersection de C_1 et C_2 : ils sont considérés comme des baselines depuis longtemps ou sont des cas particuliers comme CCA qui réduit à la fois l'espace des labels et des attributs et qui appartient donc naturellement aux deux communautés. Trois algorithmes (BiDir, MLSVM, MSE) sont liés à CCA ou MDDM seulement et par conséquent, en plus de C_1 et C_2, nous considérons un sous-ensemble C_{1-2} qui inclut ces cinq algorithmes. Les arêtes entre C_1, C_2 et C_{1-2} proviennent principalement de la référence [117] qui est une comparaison récente de différentes approches de réduction de

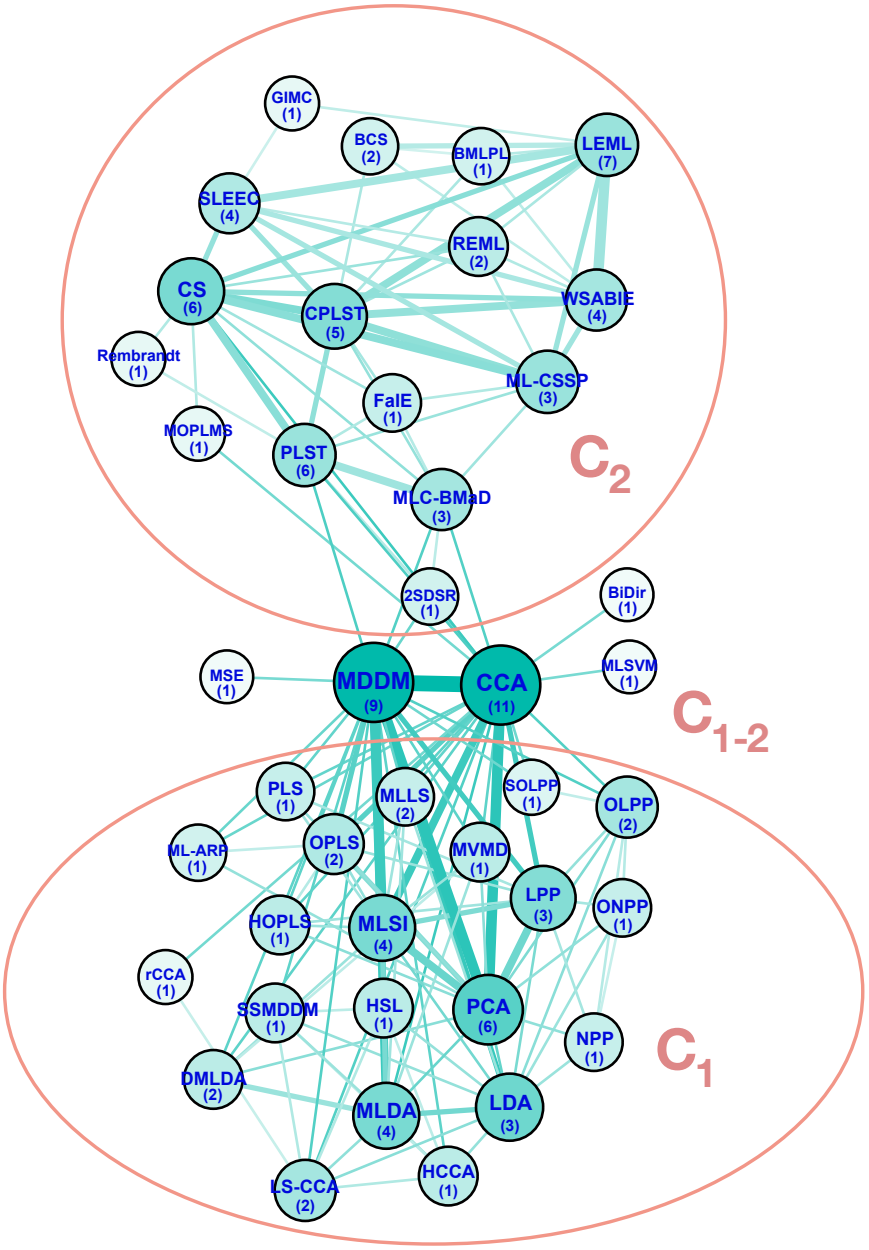

Figure 3.1 – Multigraphe G_c des occurrences/co-occurrences des algorithmes pour les 27 articles sélectionnés. Plus les arêtes (resp. les sommets) sont épaisses et foncées, plus leurs poids (resp. degrés) sont élevés. Le nombre d'articles dans lesquel chaque algorithme apparaît est entre parenthèses.

dimension pour l'apprentissage multi-label et qui souhaite spécifiquement comparer une large gamme de méthodes de typologie variée. Les diamètres des sommets permettent de mettre en évidence les méthodes les plus fréquentes qui sont souvent mentionnées parmi les pionnières de leur communauté : CCA, MDDM, LEML, PCA, CS, PLST et CPLST. Dans la suite, nous visons à identifier les dominations significatives à partir du multigraphe G_c.

3.1.2 Les mesures d'évaluation

Le tableau 3.2 montre les occurrences et les co-occurrences des différentes mesures utilisées dans les articles du corpus $\mathcal{C}$.

Tableau 3.2 – Matrice des co-occurrences des mesures de qualité calculée sur l'ensemble des 27 articles. L'occurrence de chaque mesure est sur la diagonale.

	01Loss	AUC	Accuracy	AveragePrecision	Coverage	ErrorRate	F1	HammingLoss	MacroF1	MicroF1	OneError	P@3	Precision	RankingLoss	Recall	SubsetAccuracy	MacroPrecision	MicroPrecision
01Loss	1																	
AUC	0	9																
Accuracy	1	0	3															
AveragePrecision	0	1	0	1														
Coverage	0	1	0	1	1													
ErrorRate	0	0	0	0	0	1												
F1	1	0	3	0	0	0	7											
HammingLoss	1	3	2	1	1	0	3	12										
MacroF1	0	3	0	1	1	0	0	5	7									
MicroF1	0	3	0	1	1	0	0	5	7	7								
OneError	0	3	0	1	1	0	0	2	2	2	5							
P@3	0	2	0	0	0	0	0	1	0	0	4	4						
Precision	0	0	1	0	0	0	2	1	0	0	0	0	2					
RankingLoss	0	1	0	1	1	0	0	1	1	1	1	0	0	1				
Recall	0	0	1	0	0	0	2	1	0	0	0	0	2	0	2			
SubsetAccuracy	0	0	1	0	0	0	1	1	0	0	0	0	1	0	1	1		
MacroPrecision	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	
MicroPrecision	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

Il souligne la grande variabilité des critères considérés, et la distribution des fréquences permet de distinguer les mesures les plus populaires : Hamming Loss (44 %), AUC (33 %), F1 (26 %), Macro-F1 (26 %), et Micro-F1 (26 %). En plus de ces observations, notre sélection des mesures appropriées pour la méta-analyse est guidée par une analyse récente [197] qui a prouvé expérimentalement que certaines mesures sont fortement corrélées alors que d'autres sont indépendantes. Plus précisément, les auteurs ont testé un ensemble de 16 mesures (celles présentes dans le tableau 3.2 plus quelques variantes) et les ont comparées à l'aide des critères de corrélation de Pearson et de Spearman sur 100000 simulations. Les résultats montrent que le Hamming Loss, le Coverage and le Ranking Loss sont indépendants de toutes les autres mesures, mais dans notre étude seul le Hamming Loss est pris en compte car la fréquence des deux autres est très faible sur $\mathcal{C}$. Les résultats de [197] détectent également une forte corrélation entre les mesures d'un grand ensemble $\mathcal{M}$ = {Subset Accuracy or 01Loss, Accuracy, Precision, Recall, F1, One Error, Average Precision, Micro Precision, Macro Precision, Micro F1, Macro F1, Micro Recall, Macro Recall}. Par conséquent, lorsque plusieurs mesures de $\mathcal{M}$ sont utilisées pour comparer deux algorithmes dans un même article, nous ne retenons que celle dont la fréquence est la plus élevée dans le tableau 3.2. Précisons que AUC et P@3 n'ont pas été considérés dans [197]. Mais ils ont été ajoutés ici à $\mathcal{M}$ car le calcul sur nos données de leurs coefficients de

corrélation de Pearson avec les autres mesures de $\mathcal{M}$ confirme la corrélation : la valeur du coefficient est comprise entre 0.829 (avec Macro-F1) et 0.576 (avec One-Error) pour l'AUC et est proche de 1 (avec One-Error) pour P@3. Dans la suite, nos effectuons donc deux études comparatives des algorithmes de réduction basée respectivement sur le Hamming Loss et sur le sous-ensemble de mesures sélectionnées $\mathcal{M}$ (désignées respectivement par $\mathcal{H}$ et $\mathcal{M}$). Chaque étude est effectuée séparément sur le sous-ensemble d'articles de $\mathcal{C}$ qui s'intéresse à la/les mesures en questions (12 articles pour $\mathcal{H}$ et 24 pour $\mathcal{M}$).

3.1.3 L'approche basée sur le consensus

Notre méta-analyse inspirée de la théorie du consensus [198][199] est décomposée en deux étapes successives : (i) filtrer les relations de domination statistiquement significatives pour les mesures $\mathcal{H}$ et $\mathcal{M}$ et (ii) extraire les algorithmes dominants pour chaque mesure. Enfin, nous identifions les algorithmes qui dominent statistiquement dans les deux cas. Avec un processus similaire, nous complétons l'analyse en distinguant les algorithmes dominés.

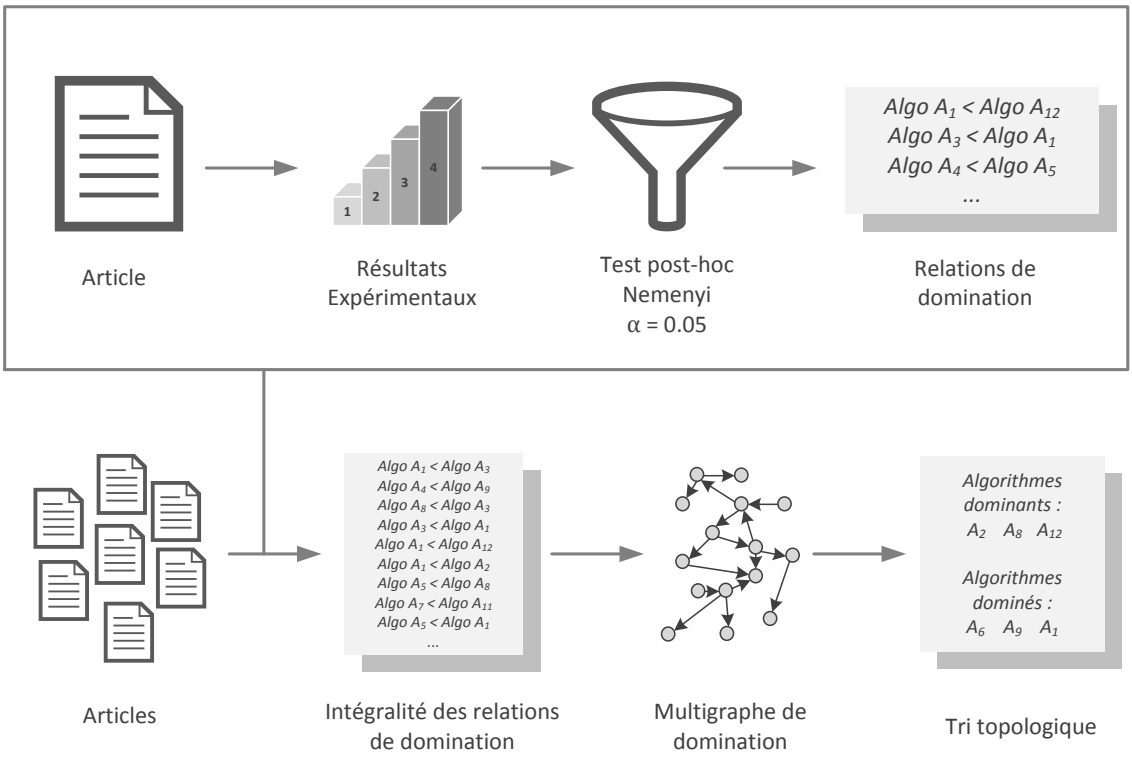

Figure 3.2 – Schéma de la méthodologie employée pour la méta-analyse.

Plus précisément, les relations de domination significatives sont extraites avec un test de Friedman et post-hoc de Nemeyi [200] avec un niveau de confiance standard $\alpha = 0.05$.

Et, pour chaque mesure, nous construisons un multigraphe orienté de domination -noté respectivement $G_D(\mathcal{H})$ et $G_D(\mathcal{M})$- à partir de G_c en conservant les arêtes significatives et en les orientant selon la direction de la domination : une arête orientée de A_i vers A_j signifie que l'algorithme A_i est significativement meilleur que l'algorithme A_j dans un article du corpus $\mathcal{C}$. La première étape d'un tri topologique [201] sur chaque multigraphe permet d'identifier les sous-ensembles $\mathcal{D}(\mathcal{H})$ et $\mathcal{D}(\mathcal{M})$ de $\mathcal{A}$ qui contiennent les algorithmes dominants : A_i est dominant quand son degré entrant est nul et son degré sortant est strictement positif. De même, les algorithmes dominés sont ceux avec un degré sortant nul et un degré entrant strictement positif. Le principe de la méthodologie est résumé dans la figure 3.2.

3.2 Résultats

Les deux multigraphes orientés $G_D(\mathcal{H})$ et $G_D(\mathcal{M})$ sont représentés dans la figure 3.3. Ils ont tous deux beaucoup moins de relations que le graphe de co-occurrence G_c. Avec des seuils α plus élevés, des arêtes orientées supplémentaires apparaissent, mais la confiance qui peut être placée dans celles-ci est plus faible. Dans notre cas, pour le seuil standard $\alpha = 0.05$, les multigraphes orientés sont devenus des graphes orientés avec au plus une arête orientée entre chaque paire de sommets. Les algorithmes de $\mathcal{A}$ avec un degré nul ne sont plus représentés. En effet, dans certains articles, le nombre d'expériences est trop faible pour détecter une domination statistiquement significative. Le tableau 3.3 indique, pour chaque article de $\mathcal{C}$ et quelle que soit la mesure de qualité considérée, le rapport $\mathrm{CD}/r_{\max}$ entre la différence critique du test post-hoc de Neymeni pour $\alpha = 0.05$ et la différence de rang maximale théorique $r_{\max}$ entre les algorithmes comparés. Si q algorithmes sont comparés, alors $r_{\max} = q - 1$. Puisque une relation de domination n'est extraite que lorsque la différence de rang entre les deux algorithmes est plus grande que CD, plus le rapport $\mathrm{CD}/r_{\max}$ est élevé, moins on s'attend à extraire des relations significatives, et lorsqu'il est supérieur à 1, aucune relation ne peut être extraite.

Dans les multigraphes $G_D(\mathcal{H})$ et $G_D(\mathcal{M})$, les communautés identifiées dans la sous-section 3.1.1 sont associées à des graphes non connexes : les algorithmes de différentes communautés ont été rarement comparés et ne sont pas liés par des relations significatives. Par conséquent, nous présentons les méthodes dominantes pour chaque communauté. Les résultats sont résumés dans le tableau 3.4. En raison de l'effet bibliographique qui favorise la présence des meilleures approches à chaque période, les approches les plus récentes (ou les plus anciennes) sont plus susceptibles d'être dominantes (resp. dominées) mais il existe des exceptions notables telles que MDDM et MLLS.

Les trois méthodes (MVMD, SSMDDM et MDDM) qui dominent pour les deux mesures appartiennent à la communauté C_1 (méthodes de réduction de l'espace des attributs dépendant des labels) ou à C_{1-2} et ont des stratégies proches. Rappelons que MDDM mi-

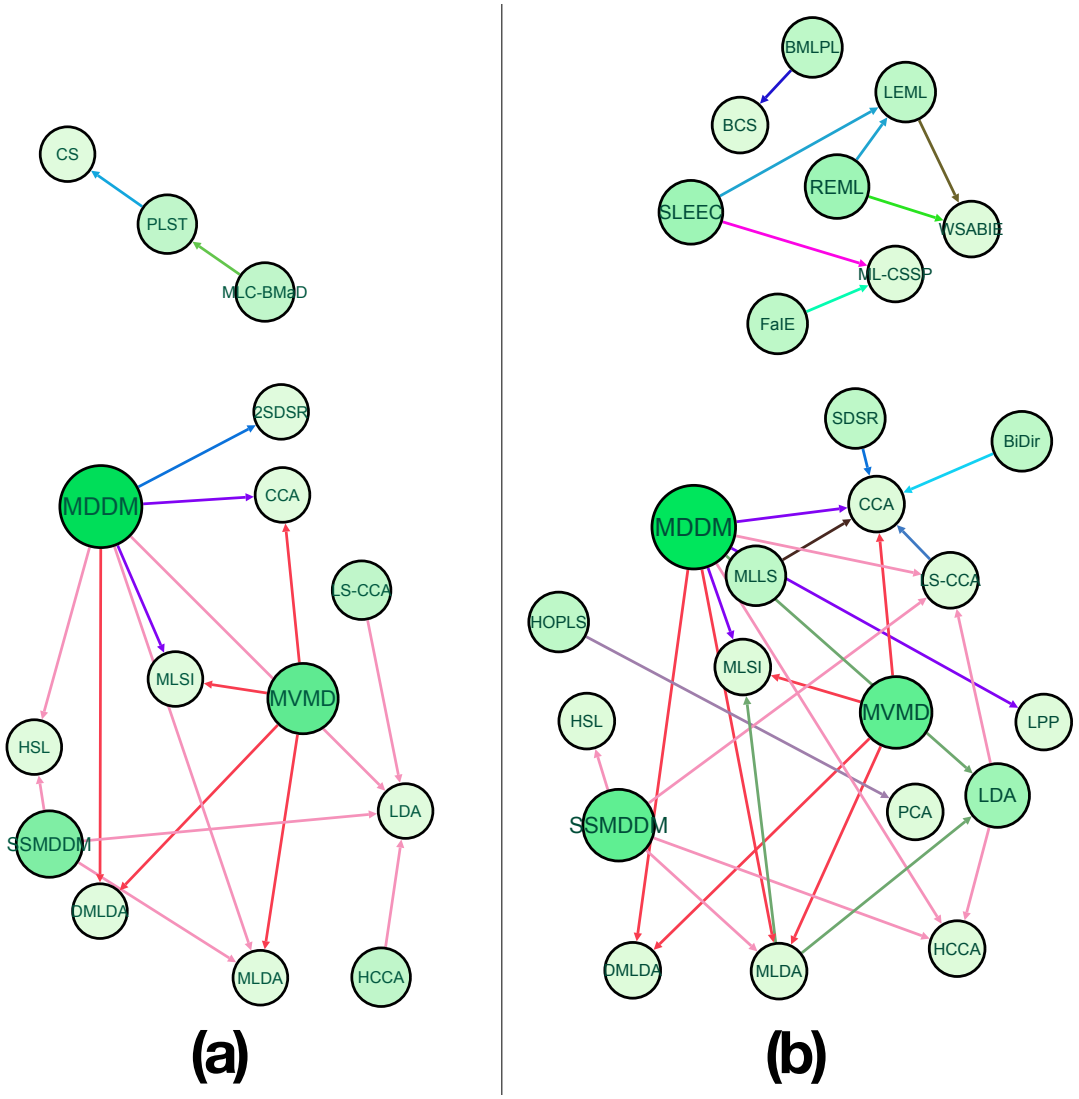

Figure 3.3 – Multigraphes de domination pour le Hamming Loss (a) et pour l'ensemble des mesures corrélées $\mathcal{M}$ (b). Différentes couleurs d'arêtes sont associées à différents articles.

Tableau 3.3 – Rapport, dans chaque article, entre la différence critique CD du test post-hoc de Neymeni pour $\alpha = 0.05$ et la différence maximale théorique de rang $r_{\max}$ entre les algorithmes comparés. Les références répétées sont associées à plusieurs ensembles de comparaisons expérimentales.

Ref.	CD/$r_{\max}$	Ref.	CD/$r_{\max}$	Ref.	CD/$r_{\max}$
[54]	0.613	[105]	0.800	[118]	0.576
[44]	0.754	[107]	0.750	[43] 1	0.462
[73]	0.530	[76]	1.657	[43] 2	1.132
[77]	0.653	[74]	0.800	[121]	0.764
[79]	0.877	[114]	0.693	[123] 1	0.867
[93]	1.386	[108]	0.453	[123] 2	0.676
[94]	1.106	[111]	0.800	[124]	0.828
[99]	0.871	[112]	1.172	[113]	0.778
[102]	0.591	[116]	0.750	[110]	0.762
[104]	0.623	[87]	0.672		

Tableau 3.4 – Algorithmes dominants détectés avec des tests statistiques avec $\alpha = 0.05$ pour chaque mesure $\mathcal{H}$ et $\mathcal{M}$ et pour chaque communauté du multigraphe G_c. Les méthodes dominantes pour les deux mesures apparaissent en gras. Différents seuils significatifs ($\alpha = 0.01$ à 0.1) ont été testés et, à l'exception de REML pour $\alpha = 0.01$, les algorithmes de ce tableau restent en haut du classement.

$\mathcal{D}(\mathcal{M})$			$\mathcal{D}(\mathcal{H})$		
C_1	C_{1-2}	C_2	C_1	C_{1-2}	C_2
MVMD	BMLPL	**MDDM**	**MVMD**	MLC-BMaD	**MDDM**
SSMDDM	REML	MLLS	**SSMDDM**		
HOPLS	SLEEC	BiDir	LS-CCA		
LDA	FaIE		HCCA		
	SDSR				

nimise le critère d'indépendance de Hilbert Schmidt entre l'espace des attributs réduits et l'espace des labels et que MVMD et SSMDDM sont des méthodes hybrides dont l'objectif est un compromis entre l'objectif de MDDM et celui d'une autre méthode (voir tableau 1a). MVDM et SSMDDM sont des approches récentes qui ont été largement comparées à d'autres mais dans un seul article, alors que MDDM, qui est plus ancien, résiste à un plus grand nombre de comparaisons.

La communauté C_2 est très petite dans $G_D(\mathcal{H})$: les auteurs des algorithmes de C_2 s'intéressent principalement à des données avec un grand nombre de labels et accordent plus d'importance aux mesures de classement qu'à celles de reconstruction globale telles que le Hamming Loss. Par conséquent, il n'y a pas de méthode qui domine simultanément pour les deux mesures. Pour la mesure $\mathcal{M}$, SLEEC et REML dominent dans plusieurs articles. Ces méthodes ont été spécialement conçues pour l'apprentissage multi-label extrême, et contrairement aux autres qui construisent des représentations de données de faible rang pouvant manquer l'information apportée par la distribution à longue traîne des labels, elles calculent une représentation de haut rang qui capture une plus grande partie de l'information.

En plus de ces résultats les plus significatifs, il est intéressant d'identifier les approches dominantes pour une mesure et dominées pour l'autre. LS-CCA et HCCA (respectivement LDA) sont dominants dans $G_D(\mathcal{H})$ (respectivement $G_D(\mathcal{M})$) et dominent dans $G_D(\mathcal{M})$ (resp. $G_D(\mathcal{H})$). Ces méthodes ne sont pas intrinsèquement supposées être plus efficaces pour le Hamming Loss que pour les mesures de $\mathcal{M}$, et en raison de l'absence de corrélation entre $\mathcal{M}$ et $\mathcal{H}$, et il n'est donc pas surprenant de trouver des comportements différents. Ce résultat confirme l'intérêt de la double analyse.

Les méthodes dominées pour les deux mesures sont HSL, DMLDA, MLSI et CCA. Elles appartiennent toutes à C_1 ou C_{1-2} en raison de l'absence de mesures de Hamming Loss en C_2. Elles illustrent l'effet bibliographique : ce sont des méthodes précoces dominées par des propositions plus récentes. Il faut rester prudent sur les interprétations de ces résultats. En effet, HSL est dominé dans les expériences disponibles mais il n'a été

considéré qu'une seule fois dans le corpus $\mathcal{C}$. En outre, CCA pourrait avoir été utilisé dans la majorité des articles avec une version qui n'est pas optimale. Dans les expériences, les matrices d'attributs, de labels et de covariance attributs/labels sont souvent mal conditionnées en raison des caractéristiques des jeux de données multi-label et il est connu que l'intégration d'une légère régularisation avec une inversion généralisée de Penrose conduit à de bien meilleures performances pour CCA [118].

Pour l'ensemble des algorithmes de $\mathcal{A}$ qui n'appartiennent pas au groupe des dominés ou des dominants, deux cas doivent être considérés. Pour les algorithmes qui ne sont pas dans les graphes G_D, notre méta-analyse ne peut rien conclure de plus que le manque de significativité des comparaisons qui les prennent en compte. Pour les autres, qui ont à la fois un degré entrant et sortant non nul, une recommandation raisonnable est de les remplacer par l'un des algorithmes dominants de leur communauté définis pour une tâche similaire.

3.3 Discussion

En complément des spécificités théoriques des approches présentées précédemment, les résultats expérimentaux restent un critère de sélection majeur. Une méta-analyse a été menée pour identifier les performances les plus significatives des algorithmes à partir des comparaisons numériques répertoriées dans les publications. Les résultats dépendent à la fois de la mesure de qualité utilisée et des informations principales guidant la réduction de dimension (espace des attributs, des labels ou les deux). Trois méthodes MVMD, SSMDDM et MDDM basées sur la réduction de l'espace des attributs dominent pour les deux mesures conservées non corrélées (Hamming Loss et une mesure sélectionnée parmi un grand ensemble de mesures corrélées, comprenant par exemple le Micro F1, le Macro F1 et l'AUC). Les résultats mettent également en évidence SLEEC et REML, des approches récentes spécialement conçues pour l'apprentissage multi-label extrême. Un double examen des relations de domination complète l'analyse en soulignant les méthodes dominées pour les deux mesures. Cependant, d'un point de vue méthodologique, la généralisation des conclusions doit être considérée avec prudence. Comme de nombreuses comparaisons par paires sont absentes des expériences publiées, la méta-analyse a été réalisée sur un graphe non complet. De plus, l'hétérogénéité des jeux de données utilisés dans les différentes études et le nombre de fois où chaque algorithme a été évalué ajoutent des biais aux comparaisons. Malgré ces limitations, nous pensons néanmoins que cette première méta-analyse peut aider à identifier des propriétés récurrentes dans les approches les plus efficaces et également des failles dans les protocoles expérimentaux (par exemple, l'absence de certaines comparaisons par paires). De manière plus générale, le développement des publications dans l'apprentissage automatique favorisera certainement les procédures

de méta-analyse dans un avenir proche.

4 Proposition d'une nouvelle méthode de réduction couplée avec le classifieur ML-kNN : ML-ARP

La meilleure méthode de réduction de dimension n'est pas nécessairement la même selon le classifieur [54]. Ce résultat stimule le développement d'approches intégrant un couplage entre réduction de dimension et apprentissage. Les méthodes couplées existantes ont plutôt recours à un couplage avec des classifieurs qui s'approchent des SVM [93][92]. Mais leur processus d'optimisation tente de combiner explicitement deux objectifs différents. Dans [92], la fonction de perte exprimée est la somme de deux erreurs de reconstruction : réduction de dimension et apprentissage. Dans [93], la combinaison des deux formulations conduit à un problème d'optimisation à deux paramètres où chaque paramètre est calculé alternativement. Cette stratégie multi-objectifs peut converger vers une solution de mauvaise qualité pour le classifieur. De plus, ces approches précédentes ne considèrent pas un couplage avec le célèbre algorithme des k-plus proches voisins (ML-kNN) qui est, avec son caractère multi-label intrinsèque et ses règles de classification puissantes, un candidat potentiellement très intéressant. Nous proposons ici une nouvelle approche de réduction de dimension des attributs couplée avec ML-kNN. Dans son problème d'optimisation, la projection vers l'espace d'attributs réduits est l'unique paramètre et la performance de ML-kNN dans l'espace projeté est l'unique objectif. Pour le résoudre, l'approche appelée Multi-Label Adaptative Random Projection (ML-ARP), initialise une projection linéaire aléatoire et l'adapte itérativement avec une recherche à voisinage variable réduite. Puisque la complexité et les performances de ML-kNN dépendent principalement des caractéristiques de l'espace des attributs, notre proposition ainsi que les méthodes auxquelles nous la comparons réduisent et transforment cet espace. En réduisant la dimension des données et la complexité de recherche de voisinage jusqu'à 90%, ML-ARP est non seulement meilleur en moyenne que ML-kNN sans reduction de dimension, mais il est compétitif avec plusieurs algorithmes standards de réduction de dimension cités précédemment (PCA, CCA, MDDM et OPLS).

4.1 Description de l'algorithme ML-ARP

4.1.1 Rappels sur ML-kNN

Parmi les méthodes d'apprentissage multi-label, ML-kNN fait probablement partie des meilleurs. Basé sur le principe du maximum *a posteriori*, ML-kNN exploite une stratégie d'apprentissage basé instance. Des comparaisons numériques avec de nombreuses méthodes ont confirmé la grande qualité de ses résultats.

Rappelons que ML-kNN [29] combine le principe de l'algorithme des k plus proches voisins avec une puissante règle de décision multi-label. Plus précisément, pour un vecteur d'attributs donné $x \in \mathcal{R}^{d_x}$ il détermine d'abord son voisinage dans $\mathcal{T}$ en utilisant la distance Euclidienne. Ensuite, il prédit une sortie à valeur réelle $\widehat{y_{int}} \in \mathbb{R}^{d_y}$ en additionnant les labels des k voisins, puis il convertit sa prédiction en classification avec une règle de maximum *a posteriori*. Cette règle, basée sur des statistiques globales et locales sur les labels, nécessite une phase d'apprentissage où deux quantités sont calculées pour chaque label l : (i) la probabilité *a priori* de la présence (ou de l'absence) du label l qui est sa fréquence (ou le complémentaire) dans $\mathcal{T}$ et (ii) la probabilité dans $\mathcal{T}$ qu'une instance associée au label l ait exactement j voisins avec le label l, pour $j \in \{0, ..., k\}$. Avec ces deux informations et $\widehat{y_{int}}$, ML-kNN détermine la probabilité *a posteriori* de la présence/absence de chaque label avec une règle de Bayes : si la probabilité de présence est supérieure à la probabilité d'absence, le label est prédit (valeur 1) dans le vecteur final de prédiction des label $\widehat{y} \in \{0, 1\}^{d_y}$.

Un lissage de Laplace est optionnellement appliqué pour éviter que les événements qui ne se produisent pas dans l'ensemble d'apprentissage $\mathcal{T}$ aient une probabilité égale à zéro. Dans nos expériences, sans aucune connaissance préalable des données, nous préférons éviter d'utiliser ce lissage. De plus, comme ML-kNN est ici appliqué pour chaque méthode que nous comparons, le paramètre de lissage affecterait uniquement les performances absolues et non les comparaisons relatives.

4.1.2 ML-ARP : Multi-Label Adaptative Random Projection

Notre but est de construire une projection qui optimise explicitement les performances m_q de ML-kNN dans l'espace des attributs réduits (de dimension d'_x). Ceci a pour objectif de corriger des défauts de ML-kNN : comme il s'appuie sur une fonction de distance, il est très sensible aux attributs bruyants, redondants et non pertinents [202]. La réduction de dimension est un levier prometteur pour ces problèmes car elle peut filtrer implicitement les attributs non pertinents pour l'apprentissage. En sus, elle permet de réduire la complexité de l'évaluation de la distance en passant de d_x opérations à d'_x opérations. Les performances sont ici mesurées avec le Hamming Loss (HL). Le problème de minimisation sur l'objectif $m_q(P)$ est alors défini par :

$$\min_{P \in \mathbb{R}^{d_x \times d'_x}} m_q(P) = \min_{P \in \mathbb{R}^{d_x \times d'_x}} \sum_{i=1}^{n} HL(y_i, \widehat{y_i} = \text{ML-}k\text{NN}(\mathcal{T}^P, x_i)) \tag{3.7}$$

où ML-kNN$(\mathcal{T}^P, x_i)$ est la prédiction pour x_i de ML-kNN appliqué sur l'ensemble d'apprentissage après projection des attributs avec P.

Comme la dérivée de ML-kNN$(\mathcal{T}^P, x_i)$ en fonction de P ne peut pas être explicitée, les approches standards d'optimisation sont impraticables et nous recourons à une

heuristique RVNS (Reduced Variable Neighorhood Search) pour calculer une solution au problème (3.7). Notre implémentation du RVNS modifie le paramètre P du problème itérativement et aléatoirement et conserve les changements qui améliorent l'objectif m_q. Plus précisément, les différentes étapes de l'algorithme sont les suivantes :

1. Initialiser P comme une projection aléatoire tirée d'une distribution gaussienne à variance unitaire et à moyenne nulle.
2. Effectuer une légère modification de la solution P en une nouvelle solution P' en utilisant une matrice de "vitesse" ΔP : $P' = P + \Delta P$.
3. Évaluer la perte $m_q(P')$ liée au nouveau paramètre P'.
4. Si $m_q(P')$ est inférieure à $m_q(P)$, alors P' est considérée comme la nouvelle solution actuelle ; sinon, on conserve le paramètre P.
5. Si la nouvelle solution est P', les étapes 2, 3 et 4 sont répétées avec la même matrice de "vitesse" ΔP ; dans le cas contraire, ces étapes sont répétées avec une nouvelle matrice de "vitesse". La matrice de "vitesse" choisie est creuse afin que seuls quelques paramètres soient modifiés à chaque itération de RVNS. Le processus de construction de ΔP est le suivant :
   On sélectionne aléatoirement un taux de mutation α dans $[0, 1]$. Ensuite, pour chaque terme de la matrice ΔP, on réalise un essai de Bernoulli avec une probabilité de α. Si le résultat est négatif, le terme est fixé à 0 ; sinon, le terme est généré aléatoirement à partir d'une distribution gaussienne standard.

Le processus s'arrête après un nombre fixe d'itérations ou un temps de calcul maximum. Remarquons que les conditions du lemme de Johnson-Lindenstrauss ne sont pas valables ici : en choisissant des modifications spécifiques, l'algorithme ML-ARP produit une solution finale P qui n'est plus une projection aléatoire. Par conséquent, les distances initiales dans l'espace original ne sont pas conservées ; elles sont modifiées afin d'améliorer les performances de ML-kNN. Une transformation réductrice non linéaire aurait aussi pu être candidate. Cependant, sans plus d'informations sur les données, nous avons privilégié ici le choix le plus simple d'une projection linéaire. La non-linéarité globale du modèle est indirectement gérée par la combinaison de la réduction avec le classifieur non linéaire ML-kNN.

4.2 Expérimentations

Nous décrivons d'abord le protocole expérimental et présentons ensuite les comparaisons obtenues avec six approches différentes sur douze ensembles de données de différentes tailles.

4.2.1 Protocole expérimental

Données Nous avons effectué nos comparaisons expérimentales sur douze jeux de données provenant de divers domaines : annotation musicale (Emotions), annotation d'images (Scène, Corel5k), vidéo (Mediamill), compréhension de textes (Enron, Bibtex, Delicious, Bookmarks, Reuters) et santé (Yeast, Genbase, Medical). Leurs principales propriétés sont décrites dans l'annexe 1 et nous référons à Mulan [203] pour plus de détails.

Algorithmes Le nouvel algorithme ML-ARP a été comparé à quatre autres approches de réduction de dimension de l'état de l'art (PCA [64], CCA [118], MDDM [54], OPLS [101]). Suite à la réduction de dimension, la classification a été effectuée avec ML-kNN. Nous avons ajouté deux autres comparaisons qui jouent le rôle de référence : une avec une projection aléatoire normalisée (RP) tirée d'une distribution gaussienne standard et une autre avec le classificateur ML-kNN original sans réduction de dimension.

Dans nos expériences, la dimension d'_x de l'espace réduit des attributs est du même ordre de grandeur que celles classiquement utilisées dans la littérature : 128 ou 64 si la dimension de l'espace des attributs d'origine est inférieure à 128. Plus la dimension de l'espace d'_x est grande, plus la projection est expressive. Nous avons donc fixé la même valeur pour chaque méthode pour une comparaison équitable. Pour vérifier la pertinence des résultats, nous avons également implémenté une baseline qui prédit systématiquement, pour tout $x \in \mathcal{S}$, les fréquences des labels calculées sur $\mathcal{T}$. Les valeurs réelles du vecteur de fréquences sont binarisées avec un seuil de 0.5. Comme nous nous limitons ici à la comparaison des différentes performances d'approches, nous n'avons pas exploré l'impact du nombre de voisins. Nous avons suivi la recommandation de [29] et fixé $k = 5$. Le temps de calcul maximal a été fixé à deux heures pour répondre à nos contraintes opérationnelles.

Évaluation de la qualité Pour évaluer les performances des algorithmes sur chaque jeu de données, nous avons effectué une "10-fold cross-validation" et calculé la performance moyenne et l'écart-type de 11 mesures différentes [31, 34, 30] évaluant des performances de classement, de précision de classification et de reconstruction globale : Ranking Loss, One Error, Coverage, Jaccard Loss, Hamming Loss, Accuracy, Recall, Precision, Subset Accuracy, Average Precision, F1-Score.

Une analyse plus poussée avec des tests statistiques sur le Hamming Loss a été effectuée pour évaluer les différences significatives et les similitudes entre les algorithmes. En utilisant le package R *scmamp* [204], nous avons appliqué le test de Friedman avec $\alpha = 0.1$ (confiance 90%) et l'avons complété avec le test post-hoc de Nemenyi.

Tableau 3.5 – Hamming Loss des prédictions des différentes méthodes sur les jeux de données de l'étude (N/A désigne les valeurs non disponibles)

		ML-ARP	Baseline	ML-kNN	RP	MDDM	PCA	OPLS	CCA
Yeast	$d'_x = 64$	**0.191**	0.232	0.195	0.202	0.227	0.194	0.203	0.204
Emotions	$d'_x = 64$	**0.226**	0.313	0.262	0.261	0.31	0.262	0.256	0.252
Scene	$d'_x = 128$	0.091	0.179	**0.088**	0.108	0.089	0.097	0.166	0.162
Enron	$d'_x = 128$	0.05	0.062	0.051	0.053	**0.049**	**0.049**	0.067	0.064
Genbase	$d'_x = 128$	0.047	0.047	0.047	0.047	0.047	0.047	0.047	N/A
Corel5k	$d'_x = 128$	0.009	0.009	0.009	0.009	0.009	0.009	0.009	N/A
Delicious	$d'_x = 128$	**0.014**	0.019	**0.014**	0.020	0.018	0.018	N/A	N/A
Medical	$d'_x = 128$	0.017	0.028	0.015	0.018	**0.013**	0.015	N/A	N/A
Bibtex	$d'_x = 128$	0.014	0.015	0.014	0.014	**0.012**	0.013	N/A	N/A
Mediamill	$d'_x = 64$	0.027	0.035	0.027	0.028	N/A	N/A	**0.025**	N/A
Bookmarks	$d'_x = 128$	0.008	0.009	0.008	0.008	N/A	N/A	N/A	N/A
Rcv1	$d'_x = 4000$	0.026	0.028	0.026	0.026	N/A	N/A	N/A	N/A

4.2.2 Résultats

Les résultats détaillés (intégralité des mesures de performances pour chaque algorithme sur chaque jeu de données) sont présentés en Annexe 2. Dans ce paragraphe, nous en présentons une synthèse. Les résultats obtenus avec le Hamming Loss pour les différentes approches sont résumés dans le tableau 3.5.

Premièrement, ils montrent que ML-ARP dépasse les autres approches de réduction de dimension (MDDM, PCA, OPLS, CCA) pour trois ensembles de données (Yeast, Emotions, Delicious) et qu'il est très proche des meilleures valeurs pour les autres ensembles de données. Deuxièmement, ils suggèrent que ML-ARP est empiriquement meilleur que ML-kNN, RP et la baseline, mais la significativité statistique de la domination n'est confirmée que par rapport à la baseline par le test de Nemenyi (Figure 3.4). Troisièmement, les performances du ML-kNN sans réduction sont toujours améliorées par au moins une approche de réduction de dimension. Mais, pour certains jeux de données, la réduction de dimension indépendante peut conduire à des résultats dégradés (par exemple, MDDM pour Emotions et CCA pour Scene). MDDM, CCA, OPLS, PCA ne sont pas appliqués sur certains ensembles de données (valeurs N/A) parce que leur complexité (spatiale et temporelle) est trop élevée ou parce qu'ils nécessitent une inversion d'une matrice non inversible.

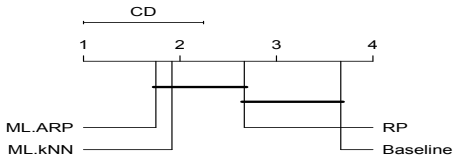

Figure 3.4 – Résultat du test statistique de Nemenyi pour ML-ARP, une projection aléatoire (RP), ML-kNN et la baseline pour tous les jeux de données.

Tableau 3.6 – Rangs moyens des algorithmes pour toutes les mesures de performances considérées pour quatre jeux de données (Emotions, Scene, Enron et Yeast).

	ML.ARP	Baseline	ML-kNN	RP	MDDM	PCA	OPLS	CCA
Accuracy	**2.00**	8.00	3.25	4.75	4.00	3.00	6.5	4.5
Average Precision	**2.00**	7.75	3.375	5.00	4.25	2.875	6.125	4.625
Coverage	3.25	6.75	**2.375**	4.375	4.00	3.00	6.5	5.75
F1	**2.5**	8.00	3.00	5.25	4.00	2.625	6.125	4.5
Hamming Loss	**2.00**	7.5	3.375	4.5	4.375	3.25	5.75	5.25
Jaccard Loss	**2.5**	8.00	3.00	4.75	4.00	2.75	6.5	4.5
One Error	2.75	8.00	**2.625**	5.00	5.00	**2.625**	5.75	5.25
Precision	3.375	5.75	4.375	4.625	3.75	**3.125**	6.00	5.00
Ranking Loss	3.00	8.00	**2.75**	4.25	4.00	**2.75**	6.5	4.75
Recall	**2.625**	8.00	3.875	5.25	4.00	3.00	5.5	3.75
Subset Accuracy	**2.00**	8.00	3.25	4.25	4.00	3.5	6.00	5.00
Moyenne	**2.55**	7.52	3.2	4.73	4.13	2.95	6.11	4.81
Rang global	**1.64**	7.91	2.64	5.41	4.05	1.91	7.09	5.36

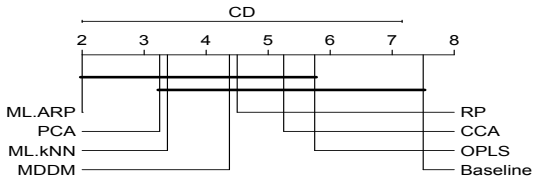

Figure 3.5 – Résultat du test statistique de Nemenyi pour tous les algorithmes sur quatre jeux de données (Yeast, Emotions, Scene, Enron).

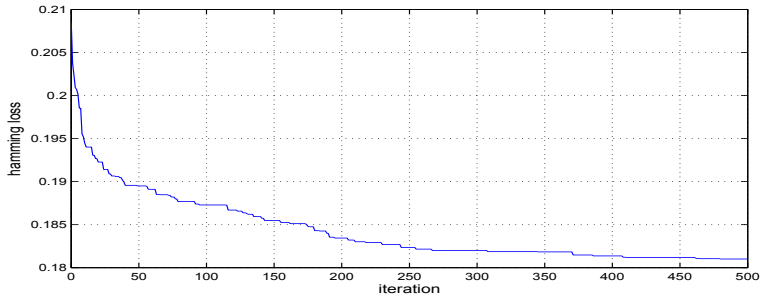

Figure 3.6 – Évolution pour ML-ARP, sur le jeu de données Emotions, de la performance de Hamming loss sur l'ensemble d'apprentissage au cours des itérations (courbe moyenne obtenue sur 10 répétitions)

Pour les quatre jeux de données où tous les algorithmes ont été appliqués (Emotions, Scene, Enron et Yeast), ML-ARP obtient le rang moyen le plus élevé pour une majorité de mesures de performance (Tableau 3.6). Il est meilleur pour les mesures d'erreur de reconstruction globale (Hamming Loss, perte de Jaccard, précision). Pour certaines mesures sensibles au rang (Coverage, One Error, Precision, Ranking Loss), il est légèrement dépassé par ML-k NN et PCA avec des performances très proches mais les différences

ne sont pas statistiquement significatives (Figure 3.5). De plus, même si un examen plus approfondi du temps de convergence dépasse l'objectif de cet étude, nous avons observé que, en moyenne, le RVNS dans ML-ARP converge après plusieurs centaines d'itérations (Figure 3.6).

5 Vers l'apprentissage extrême : limites de la littérature et des contributions complémentaires

Le couplage entre la réduction de dimension et l'apprentissage apparaît intuitivement comme un élément prometteur pour améliorer l'état de l'art actuel et cela est confirmé par quelques premiers résultats expérimentaux. Pour mieux approfondir cet apport, nous avons exploré deux pistes de recherche.

D'abord, nous avons proposé ML-ARP, une nouvelle méthode de réduction linéaire pour spécifiquement optimiser les performances de classification de ML-kNN qui est un des meilleurs algorithmes multi-label, et l'avons comparé à l'état de l'art en réduction de dimension de l'espace des attributs sur un banc d'expériences multi-label standard. Cette étude a permis de tirer plusieurs conclusions. Quel que soit le jeu de données, il a été observé qu'il existe un espace réduit pour lequel les performances de ML-kNN peuvent être améliorées ou maintenues. Ainsi, les approches de réduction de dimension ont non seulement l'avantage de réduire le nombre d'attributs et d'accélérer la recherche de voisinage, mais elles ont aussi le potentiel d'améliorer les performances de ML-kNN. Cependant, dans la pratique, les approches de réduction classiques ont obtenu des performances intéressantes pour certains jeux de données mais ont détérioré l'apprentissage pour d'autres parce que leur objectif indépendant basé sur leur propre critère (covariance, dépendance, co-occurrence) ne garantit pas d'obtenir un voisinage pertinent dans ML-kNN. A l'inverse, ML-ARP, en tant que "wrapper" conçu pour améliorer spécifiquement l'objectif de ML-kNN, présente les performances les plus régulières et le meilleur classement moyen face à une grande variété de méthodes. Par contre, sa complexité est très élevée et nous avons donc rencontré des problèmes pour le passage à l'échelle. En effet, ML-ARP nécessite d'évaluer ML-kNN à chaque itération. De plus, résolu par RVNS, il ne peut pas apprendre une projection avec un nombre important de composantes. Il devient difficile d'apprendre dès lors que $d_x \times d'_x$ est de l'ordre de 10^5 or, en XML, ce nombre peut facilement atteindre 10^9 (avec $d_x \sim 10^6$ et $d'_x \sim 1000$).

Ainsi, pour tenter de se confronter à l'échelle nouvelle du XML, nous nous sommes orientés vers une famille d'approches très populaires aujourd'hui et qui permet aussi de faire une réduction de dimension de manière totalement couplée avec l'apprentissage : les réseaux de neurones (figure 3.7). Les résultats n'ont pas été présentés en détails dans ce chapitre car les conclusions nous ont finalement conduits à changer d'orientation de

recherche. Nous résumons néanmoins ici une ébauche des résultats et les premières conclusions que nous avons pu en tirer.

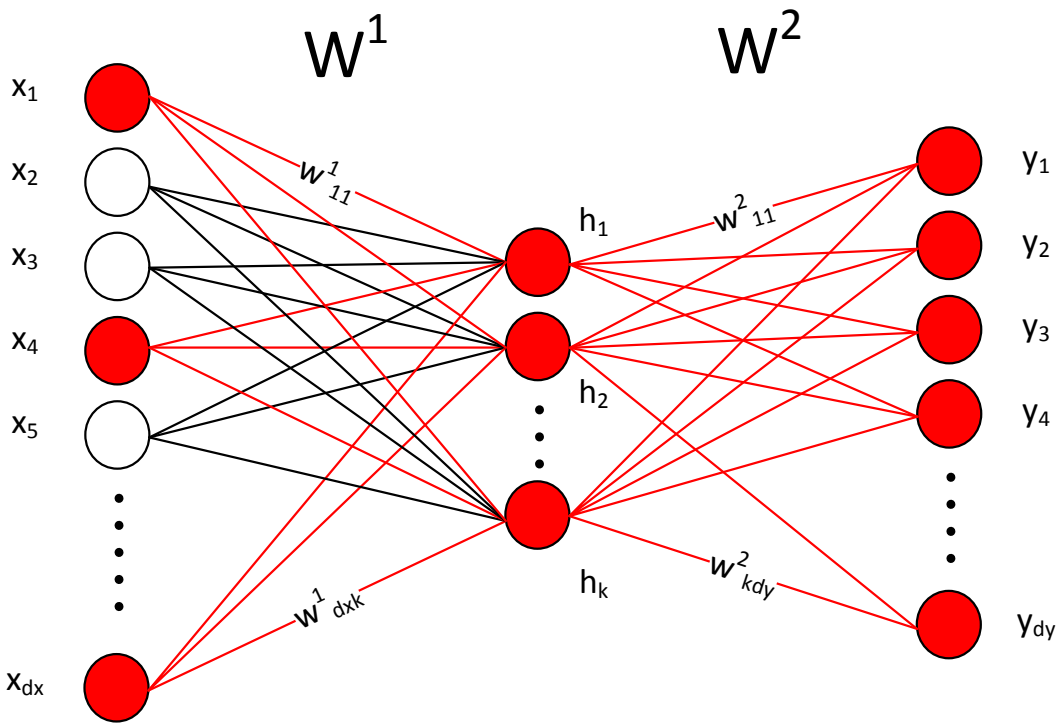

Figure 3.7 – Prédiction multi-label avec un perceptron multi-couches. Les neurones rouges sont non nuls et les connexions rouges correspondent à celles qui ont nécessité une opération dans la descente de gradient.

Premièrement, sur les jeux de données multi-label de plus petite dimension dans la littérature extrême (Bibtex, Mediamill, Delicious et Eurlex), nos expérimentations ont d'abord montré qu'un perceptron multi-couche (MLP) avec une seule couche cachée était plus performant qu'une architecture un peu plus profonde (3 couches et 5 couches). Ce résultat, qui pourrait être surprenant au premier abord, est en fait assez récurrent sur des applications où les données sont creuses comme, par exemple, la classification de textes [205]. Deuxièmement, sur ces mêmes jeux de données, notre meilleure variante du perceptron multi-couche (une couche cachée), entraînée avec la méthode ADAM [206] et avec une stratégie de "early stopping" avec patience, obtient des performances proches de l'état de l'art SLEEC en réduction de dimension pour l'apprentissage multi-label extrême (tableau 3.7)

Ces deux premiers résultats sont donc encourageants pour explorer les réseaux de neurones dans le cadre de l'apprentissage multi-label extrême. Cependant, le très grand nombre de labels constitue un obstacle majeur car, bien que les calculs puissent être simplifiés en tirant profit du caractère creux des attributs, cela ne peut être fait sur les

Tableau 3.7 – Comparaison entre les performances d'un perceptron multi-couche (MLP) et l'état de l'art en reduction de dimension multi-label extrême sur quatre jeux de données standards.

	Eurlex			Mediamill			Bibtex			Delicious		
	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
MLP	76,62	62,31	50,97	83.16	65.85	51.48	63.00	39.33	29.18	63.53	62.99	58.16
SLEEC	79.26	64.30	52.33	87.82	73.45	59.17	65.08	39.64	28.87	67.59	61.38	56.56

labels. La figure 3.7 illustre le fait que dans un cas où le vecteur d'attributs est creux (peu d'attributs non nuls en rouge), le nombre de calculs peut être réduit vis à vis de x mais que les prédictions des labels restent denses. Ainsi, pendant la phase d'apprentissage, l'erreur de prédiction à minimiser l'est également. La complexité du modèle est donc de l'ordre de $O(s_x \times d_y \times n)$, ce qui compromet le passage à l'échelle en apprentissage extrême. Face à ce problème, nous avons donc envisagé deux solutions : (1) la parallélisation sur GPU et (2) des astuces d'optimisation. La première solution était difficile à mettre en oeuvre car elle nécessitait le sous dimensionnement du réseau de neurones : par exemple, sur Wiki-LSHTC, un perceptron avec seulement 1000 neurones sur une unique couche cachée nécessite déjà une dizaine de Go de mémoire confrontant alors une majorité de GPUs à un problème de gestion de la mémoire. La deuxième solution a consisté en l'échantillonnage du gradient de l'erreur de prédiction. Une des stratégies les plus connues est le "negative sampling" [1] qui consiste à ne pas rétropropager uniquement la partie du gradient de l'erreur associée aux labels qui sont non nuls dans le vecteur vérité terrain et à un échantillon des labels nuls tiré aléatoirement. La complexité devient donc proportionnelle à s_y ($O(s_x \times s_y \times n)$) et l'algorithme passe finalement à l'échelle. En revanche, les performances obtenues empiriquement avec cette stratégie étaient équivalentes à celle de LEML et donc nettement inférieures à celles de SLEEC.

Ces médiocres performances pourraient être expliquées par l'approximation de rang faible effectuée par le perceptron multi-couche. L'analyse des différentes pistes de recherche nous a conduits à explorer deux stratégies alternatives :

1. Éviter l'approximation du rang faible avec une stratégie proche de celle de REML. Cela peut consister à combiner linéairement le MLP avec un modèle linéaire/logistique prédisant les labels directement à partir des attributs (sans réduction). Dans le chapitre 4, nous nous intéressons donc à l'apprentissage d'un tel modèle linéaire dans un cadre extrême. Ce problème pose un double challenge : en complexité temporelle et en complexité mémoire. On propose une stratégie efficace permettant de réduire très significativement la complexité mémoire.
2. Éviter l'approximation du rang faible avec une stratégie proche de celle de SLEEC

1. idée exploitée dans l'apprentissage du célèbre "word2vec" [55] pour pouvoir gérer un vocabulaire important

ou AnnexML. Pour éviter l'approximation, ces dernières découpent préalablement le problème XML avec des méthodes de partitionnement hiérarchiques et effectuent la réduction localement sur les sous-problèmes. Dans le chapitre 5, nous proposons une méthode arborescente pour tenter de résoudre le problème multi-label extrême.

Notons, *a posteriori*, que des améliorations intéressantes (par exemple l'apprentissage de plusieurs plongements locaux et l'exploitation d'un graphe de labels) des réseaux de neurones pour permettre la classification multi-label extrême ont été proposées récemment dans la nouvelle approche DeepXML [207].

Chapitre 4

Astuce d'optimisation pour réduire la complexité mémoire

Sommaire

1 Introduction

Les caractéristiques des données multi-label de grande dimension confrontent l'apprentissage à un défi de gestion de la mémoire. La montée en puissance des supercalculateurs permet d'affronter les changements d'échelle [208]. Cependant, leur utilisation est sévèrement limitée par l'accès technologique et les ressources financières, et de nombreuses applications fonctionnent encore sur des ordinateurs standards. Et de plus en plus de nouvelles applications ciblent les appareils à ressources limitées tels que les smartphones et les capteurs de l'Internet des Objets (IoT) [209, 210]. Dans ce contexte, les limitations sur la puissance de calcul allouée peuvent restreindre, voire empêcher, le déploiement de certaines tâches d'apprentissage automatique. Par exemple, pour le jeu de données de référence Wiki10-31K [211] un modèle linéaire simple nécessite le stockage de $d = d_x \times d_y = 101938 \times 30938 = 3.15 \times 10^9$ paramètres, ce qui représente environ 11.7 Go de RAM avec une précision flottante. De telles dimensions vont à l'encontre des capacités

de stockage de la plupart des ordinateurs ou des smartphones standards.

Cependant, plusieurs expériences numériques ont montré que seuls les plus grands paramètres des modèles d'apprentissage automatique sont réellement utiles pour faire des prédictions précises (voir Figure 4.1 et [212, 213]). Ces observations sont confirmées pour la régression linéaire par un résultat théorique classique [214] : un sous-ensemble limité de variables qui sont naturellement sélectionnées sous régularisation L_0 [215] conduit au risque de l'oracle et par conséquent aux meilleures garanties sur la qualité des prédictions. Mais en pratique, il est bien connu que la régularisation L_0 n'est pas applicable pour des raisons de stabilité/dérivabilité. Elle est souvent remplacée par les régularisations L_1 et L_2 ou par une pénalisation de type seuillage dur [212]. Ces stratégies fournissent des solutions parcimonieuses mais nécessitent le stockage en mémoire de tous les paramètres du modèle, et même de ceux qui sont petits ou nuls, pendant le processus d'optimisation.

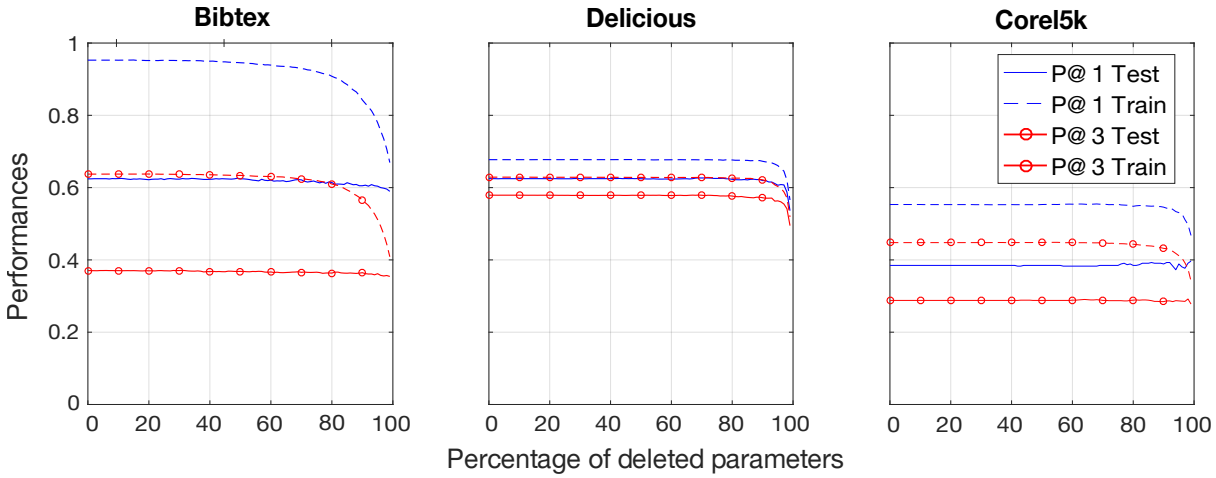

Figure 4.1 – Pour trois jeux de données classiques : évolution des performances du modèle des moindres carrés ordinaires lorsque ses paramètres les plus bas - en valeur absolue - sont progressivement supprimés. On peut noter que les performances sont maintenues avec une petite partie des plus grands paramètres.

Dans ce chapitre, nous proposons une nouvelle approche pour estimer avec précision les paramètres les plus importants tout en économisant de la mémoire. Cette approche peut être intégrée dans n'importe quel modèle qui est entraîné avec une technique basée sur la descente de gradient. Pendant le processus d'optimisation, la strategie proposée ne stocke en mémoire qu'une matrice de $m << d$ valeurs à partir de laquelle il peut reconstruire les d paramètres d'origine du modèle et identifier les plus grands avec une bonne précision. Notre approche repose sur l'algorithme de "count-sketch" [216] qui appartient à la famille des méthodes d'identification des "heavy-hitters" [217]. Ces méthodes ont été développées à l'origine dans les moteurs de recherche et dans les boutiques en ligne pour déterminer les requêtes les plus fréquentes ou les produits les plus demandés sans avoir à stocker des compteurs de tous les éléments [218]. L'algorithme consiste à stocker de manière indépendante t (~ 10) fois toutes les valeurs de paramètres sur différents ensembles de

$r << d$ compteurs en agrégeant aléatoirement plusieurs valeurs dans les mêmes compteurs avec une fonction de hachage. L'approximation d'un paramètre donné est ensuite estimée par la médiane des valeurs stockées dans les t compteurs qui lui sont associés. La qualité de cette approximation s'avère bonne pour les plus grandes valeurs de paramètres.

Nous avons testé cette approche pour le cas populaire du problème de régression linéaire one-vs-rest résolu avec une descente de gradient stochastique [219]. Des expériences numériques sur neuf jeux de données de la littérature d'apprentissage multi-label avec diverses cardinalités de paramètres (de douze mille à trois milliards) confirment que la stratégie de stockage proposée mène à des performances identiques au modèle de base qui stocke toutes les valeurs des paramètres en mémoire alors qu'elle permet de réduire très significativement la consommation mémoire (de 80% à 99,9%). Nous avons également comparé notre approche à la stratégie de hachage des attributs [220] qui est l'alternative standard pour réduire la consommation de mémoire : nos performances sont meilleures (de 10% en moyenne et jusqu'à 50%) pour la même économie de mémoire. Nous complétons les résultats expérimentaux avec une borne théorique de l'erreur d'approximation relative des paramètres et nous détaillons le cas particulier d'une distribution des paramètres suivant une loi de Zipf couramment observée.

Le reste du chapitre est organisé comme suit. La **section 2** rappelle les travaux précédents sur l'économie de mémoire. La **section 3** détaille l'approche par "count-sketch" proposée. La **section 4** montre les bornes théoriques de l'erreur d'approximation des paramètres et discute des hyperparamètres de notre proposition. La **section 5** présente les comparaisons expérimentales.

2 Travaux connexes

Dans cette section, nous décrivons d'abord l'approche de hachage des attributs qui est le moyen standard pour limiter la consommation de mémoire en réduisant le nombre de paramètres des modèles d'apprentissage. Ensuite, nous présentons les principales méthodes d'identification des "heavy hitters" qui nous semblent pertinentes pour parvenir aux mêmes finalités mais qui ont été initialement développées dans la littérature sur l'analyse des flux.

2.1 Réduction de la consommation mémoire avec du hachage d'attributs

Le nombre de paramètres d'un modèle d'apprentissage est généralement fonction du nombre d'attributs dans les données. Par conséquent, il peut être réduit en appliquant une réduction de dimension sur l'espace des attributs [221, 222]. On considère ici les

projections aléatoires car elles réduisent la dimension et n'entraînent pas de coût d'apprentissage supplémentaire.

L'une des stratégies de projection aléatoire les plus populaires est le hashing trick [220]. Il projette un vecteur d'attributs $x \in \mathbb{R}^{d_x}$ en un vecteur d'attributs réduits $x' \in \mathbb{R}^{d'_x}$ avec deux fonctions de hachage ϕ et s qui associent un index $i \in \{0, ..., d_x - 1\}$ à respectivement un autre index $\phi(i) \in \{0, ..., d'_x - 1\}$ et un signe $s(i) \in \{-1, 1\}$. Le vecteur d'attributs réduits x' est calculé comme suit :

1. x' est initialisé avec des zéros.
2. $\forall i \in \{0, ..., d_x - 1\}$, $x'_{\phi(i)} = x'_{\phi(i)} + s(i) \times x_i$.

Ce type de projections aléatoires a été appliqué avec succès pour réduire la consommation de mémoire dans de nombreux modèles d'apprentissage ou de data mining [223] tels que l'algorithme de clustering k-means [224], les k plus proches voisins [225] ou les réseaux de neurones [226]. Elles peuvent cependant être limitantes en compressant trop les attributs ou en faisant l'approximation de rang faible.

2.2 Réduction de la consommation mémoire avec des méthodes d'identification de "heavy-hitters" dans des flux

Pour dépasser les limites des méthodes de réduction de dimension, nous souhaitons développer dans ce chapitre une approche qui ne réduit pas les données initiales avant apprentissage. Cependant, sans réduire les données, le modèle peut avoir un très grand nombre de paramètres donc notre proposition a pour but d'estimer uniquement les plus grands paramètres -qui sont les plus utiles pour les prédictions- sans avoir à tous les stocker en mémoire. Ce problème est proche d'un problème rencontré dans l'analyse des flux dans lequel on considère un très grand nombre de compteurs mis à jour progressivement par un flux d'incréments et où le but est d'estimer les "heavy-hitters" qui sont les plus grands. Le scénario d'utilisation le plus courant consiste à déterminer les requêtes les plus fréquentes dans les moteurs de recherche.

Les méthodes d'identification des "heavy-hitters" développées dans la littérature sur l'analyse des flux peuvent être organisées en trois familles principales : [227] : (i) les méthodes basées sur des compteurs, (ii) les méthodes basées sur les quantiles et (iii) les méthodes basées sur les "sketchs". Les méthodes basées sur les "sketchs" se sont révélées être les plus efficaces et les plus polyvalentes [227]. Elles utilisent des fonctions de hachage pour projeter un grand ensemble de valeurs p dans un plus petit M qui permet d'estimer avec précision les "heavy-hitters" de p. Dans le scenario le plus courant, les composants de p, qui sont les compteurs associés à chaque requête, sont incrémentés de un à chaque événement du flux. Des généralisations ont été proposées pour des mises à jour avec des valeurs réelles et, dans ce cas, les méthodes basées sur les "sketchs" les plus utilisées

sont le "count-min-sketch" [217] et le "count-sketch" [216]. Cependant, le "count-min-sketch" est un modèle de type tourniquet strict qui ne traite que des problèmes pour lesquels la somme des mises à jour d'un compteur reste positive à tout instant. Par conséquent, nous considérons ici la stratégie de "count-sketch" qui convient le mieux pour l'apprentissage d'un modèle d'apprentissage automatique dont les paramètres sont habituellement *a priori* non contraints à être positifs.

3 Proposition : stockage des paramètres basé sur le count-sketch

Notons $p = (p_0, p_1, ..., p_{d-1})$ le vecteur des d paramètres associés à un modèle qui est appris par descente de gradient. A chaque pas de descente, la mise à jour d'un paramètre donné p_i est définie par $p_i \leftarrow p_i + \Delta_i$. Au lieu de stocker p en mémoire, le "count-sketch" stocke une matrice M de cardinalité $m = t \times r$ composée de t ensembles (lignes) de r compteurs où t est petit (de l'ordre de 10) et $r << d$. Deux ensembles de fonctions de hachage associent chaque paramètre p_i respectivement à un signe et à un indice de compteur pour chacun des t ensembles, et à chaque mise à jour Δ_i du paramètre p_i ses t compteurs sont incrémentés de $\pm\Delta_i$ (figure 4.2). Comme $r << d$, de nombreux paramètres entrent en collision mais comme les fonctions de hachage des différents ensembles sont indépendantes, la probabilité de collision d'un paramètre avec les mêmes paramètres dans chaque ensemble est très faible. Et l'effet de collision est réduit en estimant chaque paramètre p_i avec la médiane de son approximation dans chaque ensemble. Le signe aléatoire introduit pour le stockage des paramètres permet de réduire l'espérance des effets de collision.

Plus formellement, on note $(\phi_0, ..., \phi_{t-1})$ et $(s_0, ..., s_{t-1})$ deux ensembles de t fonctions de hachage indépendantes par paire associant tout indice de paramètre $i \in \{0, ..., d-1\}$ à respectivement un index de compteur $\phi_j(i) \in \{0, ..., r-1\}$ et un signe $s_j(i) \in \{-1, 1\}$. L'algorithme de "count-sketch" stocke la mise à jour Δ_i du $i^{\text{ème}}$ paramètre p_i comme suit :

$$\forall j \in \{0, ..., t-1\}, M_{j,\phi_j(i)} = M_{j,\phi_j(i)} + s_j(i)\Delta_i \tag{4.1}$$

Par conséquent, la valeur de chaque paramètre p_i est stockée t fois (positivement ou négativement) dans $M_{0,\phi_0(i)}$, $M_{1,\phi_1(i)}$,. .., et $M_{t-1,\phi_{t-1}(i)}$. A n'importe quelle étape, l'approximation $\widehat{p_i}$ de p_i est définie par la médiane des compteurs associés :

$$\widehat{p_i} = \underset{j \in \{0,...,t-1\}}{\text{median}} s_j(i) M_{j,\phi_j(i)} \tag{4.2}$$

Dans la section suivante, nous démontrons que ce processus permet d'estimer avec

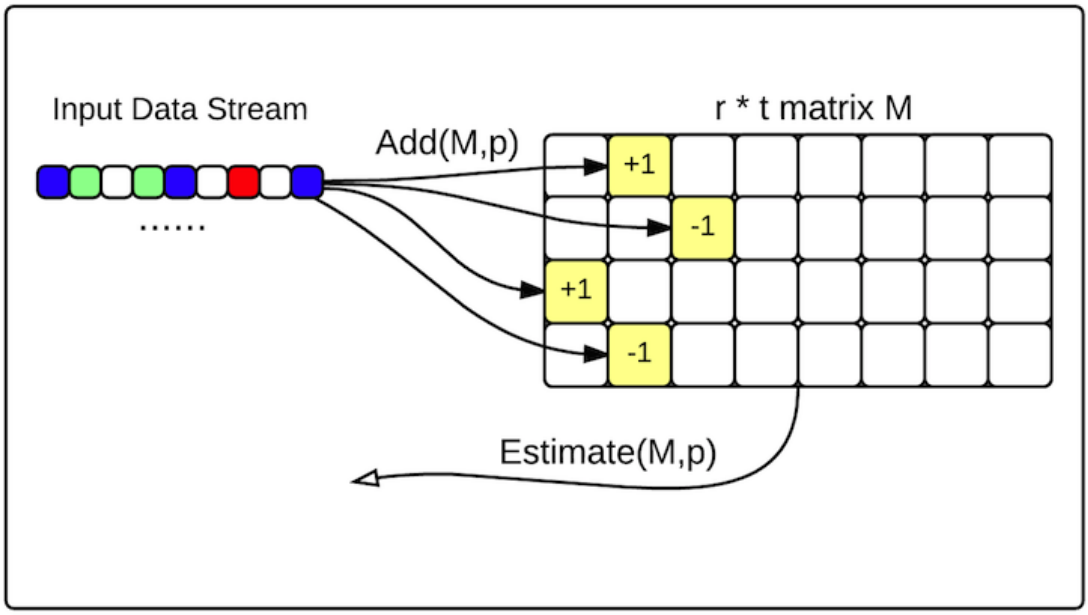

Figure 4.2 – Schéma du principe du "count-sketch". Image extraite de `http://www.cs.jhu.edu/~zliu39/clustering/`

précision les paramètres les plus grands de p qui jouent un rôle clé dans le processus d'apprentissage.

4 Analyse de l'erreur du "count-sketch" et impact sur le dimensionnement

Dans cette section, nous analysons l'erreur d'approximation normalisée $e_i = \frac{|p_i - \widehat{p_i}|}{\|p\|_2}$ pour chaque paramètre p_i. Nous définissons d'abord une borne théorique, valable à n'importe quelle étape de la descente, pour ce que nous appelons la probabilité d'échec $\mathbb{P}(e_i \geq \epsilon)$. Ensuite, nous discutons du choix des valeurs de t et r qui contrôlent la taille de la matrice M du "count-sketch" et nous étudions précisément le comportement de la probabilité d'échec dans le cas d'une loi de Zipf pour la distribution des paramètres.

4.1 Borne d'erreur pour l'estimation des paramètres

Rappelons tout d'abord un résultat pour un "sketch" avec un seul ensemble de compteurs ($t = 1$) [228]. L'approximation de p_i est donnée par $\widehat{p_i}^{(1)} = s(i)M_{0\phi(i)}$ où s et ϕ sont deux fonctions de hachage tels que $s(i) \in \{-1, 1\}$ et $\phi(i) \in \{0, ..., r-1\}$. Grâce à l'inégalité de Chebychev, on peut prouver que, pour tout $\epsilon > 0$, la probabilité d'échec est

bornée par $\frac{1}{r\epsilon^2}$:

$$\mathbb{P}\left(e_i \geq \epsilon\right) \leq \frac{1}{r\epsilon^2} \tag{4.3}$$

Démonstration. L'approximation $\widehat{p_i}$ est détériorée par des collisions avec d'autres paramètres. L'impact Y_{oi} du paramètre p_o $(o \neq i)$ sur $\widehat{p_i}$ est :

$$Y_{oi} = \begin{cases} p_o, & \text{avec une probabilité } \frac{1}{2r} \text{ (collision et signe positif)} \\ -p_o, & \text{avec une probabilité } \frac{1}{2r} \text{ (collision et signe négatif)} \\ 0 & \text{avec une probabilité } 1 - \frac{1}{r} \text{ (pas de collision)} \end{cases}$$

On a alors $\mathbb{E}[Y_{oi}] = 0$ et $\mathrm{Var}[Y_{oi}] = \frac{p_o^2}{r}$.

Soit Y_i l'impact de tous les paramètres sur $\widehat{p_i}$. Il suit que $Y_i = \sum_{o=0,o\neq i}^{d-1} Y_{o_i}$, $\mathbb{E}[Y_i] = 0$ et $\mathrm{Var}[Y_i] = \sum_{o=0,o\neq i}^{d-1} \frac{p_o^2}{r}$. De plus, Y_i est également l'erreur d'approximation de p_i : $|Y_i| = |p_i - \widehat{p_i}|$.

En appliquant l'inégalité de Chebychev sur Y_i, on obtient :
$\mathbb{P}\left(|Y_i - \mathbb{E}[Y_i]| \geq a\sqrt{\mathrm{Var}[Y_i]}\right) \leq \frac{1}{a^2} \Leftrightarrow \mathbb{P}\left(|Y_i - 0| \geq a\sqrt{\sum_{o=0,o\neq i}^{d-1} \frac{p_o^2}{r}}\right) \leq \frac{1}{a^2}$

Puisque $\sqrt{\sum_{o=0,o\neq i}^{d-1} p_o^2} \leq \|p\|_2$, l'événement $|p_i - \widehat{p_i}^{(j)}| \geq \epsilon\|p\|_2$ est inclus dans l'événement $|p_i - \widehat{p_i}^{(j)}| \geq \epsilon\sqrt{\sum_{o=1,o\neq i}^{d} p_o^2}$. En choisissant $a = \sqrt{r}\epsilon$, il suit (4.3). □

Par conséquent, pour un seul jeu de compteurs, la probabilité d'échec augmente avec la borne ϵ sur l'erreur d'approximation normalisée et décroît avec la largeur de "sketch" choisie r. Elle peut être réduite d'avantage avec l'utilisation de plusieurs compteurs et l'agrégation des t approximations (Proposition 1).

Proposition 1. *Soit $p = (p_0, p_1, ..., p_{d-1})$ le vecteur des paramètres d'un modèle d'apprentissage stocké de façon efficace dans une matrice M de "count-sketch" avec t ensembles de r compteurs. A toute étape de la descente, l'erreur normalisée $e_i = \frac{|p_i - \widehat{p_i}|}{\|p\|_2}$ de l'approximation $\widehat{p_i} = \underset{j\in\{0,...,t-1\}}{\mathrm{median}} s_j(i)M_{j,\phi_j(i)}$ de tout paramètre p_i satisfait le résultat suivant :*

$$\forall \epsilon \ s.t. \ r\epsilon^2 > 2, \mathbb{P}\left(e_i \geq \epsilon\right) \leq e^{\frac{t}{2} - \frac{t}{r\epsilon^2} - ln(\frac{r\epsilon^2}{2})\frac{t}{2}} \tag{4.4}$$

Les ingrédients de la preuve (astuce de la médiane et borne de Chernoff) sont familiers dans l'analyse de flux [217] mais, pour faciliter la lecture de ce chapitre, nous écrivons ici la preuve dans notre contexte d'apprentissage de modèle.

Démonstration. Soit W_j une variable aléatoire égale à 1 lorsque $\frac{|p_i - s_j(i)M_{j,\phi_j(i)}|}{\|p\|_2} \geq \epsilon$ et 0 sinon, et $W = \sum_{j=0}^{t-1} W_j$. Si l'inégalité est vérifiée pour la médiane, c'est-à-dire que $\frac{|p_i - \underset{j\in\{0,...,t-1\}}{\mathrm{median}} s_j(i)M_{j,\phi_j(i)}|}{\|p\|_2} \geq \epsilon$, alors elle est vérifiée pour plus de la moitié des $s_j(i)M_{j,\phi_j(i)}$ et donc $W \geq \frac{t}{2}$.

La probabilité $\mathbb{P}(W \geq \frac{t}{2})$ est donc plus grande que la probabilité $\mathbb{P}(\frac{|p_i - \underset{j\in\{0,\ldots,t-1\}}{\text{median}} s_j(i)M_{j,\phi_j(i)}|}{\|p\|_2} \geq \epsilon)$. Nous établissons une borne pour $\mathbb{P}(W \geq \frac{t}{2})$ dans ce qui suit.

Pour tout $x > 0$, $\mathbb{P}\left(W \geq \frac{t}{2}\right) = \mathbb{P}\left(e^{xW} \geq e^{x\frac{t}{2}}\right)$

$$\leq \frac{\mathbb{E}[e^{xW}]}{e^{x\frac{t}{2}}} \qquad \textit{(Inégalité de Markov)}$$

Puisque les événements W_0, W_1, ..., W_{t-1} sont indépendants du fait de l'indépendance des fonctions de hachage de chaque ensemble de compteurs, on obtient :

$$\begin{aligned}
\text{pour tout } x > 0,\ \mathbb{P}\left(W \geq \frac{t}{2}\right) &\leq e^{-x\frac{t}{2}} \prod_{j=0}^{t-1} \mathbb{E}[e^{xW_j}] \\
&\leq e^{-x\frac{t}{2}} \prod_{j=0}^{t-1} (\mathbb{P}(W_j = 1)e^x + (1 - \mathbb{P}(W_j = 1))) \\
&\leq e^{-x\frac{t}{2}} \prod_{j=0}^{t-1} e^{\mathbb{P}(W_j=1)(e^x-1)} \quad \textit{(Taylor : } 1 + y < e^y\textit{)}
\end{aligned}$$

En utilisant (4.3), on a $\mathbb{P}(W_j = 1) = \mathbb{P}\left(\frac{|p_i - s_j(i)M_{j,\phi_j(i)}|}{\|p\|_2} \geq \epsilon\right) \leq \frac{1}{r\epsilon^2}$, ce qui entraine $\mathbb{P}\left(W \geq \frac{t}{2}\right) \leq e^{-x\frac{t}{2}} e^{\frac{t}{r\epsilon^2}(e^x-1)}$. Puisque $e^{-x\frac{t}{2}} e^{\frac{t}{r\epsilon^2}(e^x-1)}$ est minimal pour $x = \ln(\frac{r\epsilon^2}{2})$, il s'en suit (4.4).

□

L'équation (4.4) donne un aperçu de l'impact de t et r sur l'erreur d'approximation normalisée des paramètres. Premièrement, $t \sim 10$ conduit à une très faible probabilité d'échec. En effet, il est facile de déduire que, si $r = \frac{3}{\epsilon^2}$, la probabilité d'échec est bornée par δ, où δ est lié à t avec la relation suivante : $t = T\ln(\frac{1}{\delta})$ où $T = \frac{2}{\ln(3/2)-1/3}$. Par conséquent, en pratique, une probabilité d'échec de $\delta = 10^{-4}$ est atteinte pour un nombre de jeux de compteurs t de l'ordre de 10 ($\sim \ln(\frac{1}{10^{-4}})$). Deuxièmement, l'équation (4.4) montre que l'erreur d'approximation normalisée e_i, bornée par $\epsilon = \sqrt{\frac{3}{r}}$ avec une probabilité de $1 - \delta$, diminue avec la racine du nombre de compteurs r. Enfin, en raison de la normalisation par $\|p\|_2$ dans l'équation (4.4), l'erreur d'approximation relative $e_{\%i} = \frac{|p_i - \widehat{p_i}|}{p_i}$ pour chaque paramètre p_i dépend de la taille de p_i par rapport aux autres. Par conséquent, la qualité de l'approximation pour les plus grands paramètres est étroitement liée à la distribution des paramètres du modèle appris.

4.2 Borne d'erreur dans le cas d'une distribution de Zipf

Pour les ensembles de données considérés dans nos expériences présentées dans la section suivante, les paramètres du modèle bien connu des moindres carrés ordinaires

suivent une distribution semblable à la loi de Zipf (figure 4.3). Nous fournissons ici une borne pour l'erreur d'approximation dans ce cas très commun.

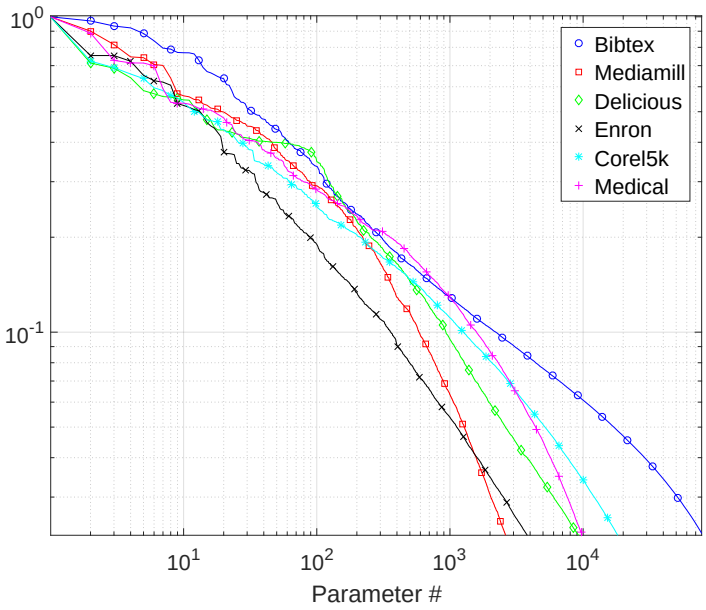

Figure 4.3 – Distribution des paramètres estimés par les moindres carrés ordinaires pour six ensembles de données (échelle logarithmique). Les paramètres sont triés et représentés en valeur absolue.

Proposition 2. *Soit $p = (p_0, p_1, ..., p_{d-1})$ le vecteur des paramètres du modèle d'apprentissage triés par ordre décroissant dont les valeurs sont définies par $|p_i| = Cx^i$ où $x \in]0, 1[$ est le décrément et $C = |p_0|$ est le plus grand paramètre. À chaque étape de la descente, pour chaque paramètre p_i, l'erreur relative $e_{\%i} = \frac{|p_i - \widehat{p_i}|}{p_i}$ est bornée, avec une probabilité d'au moins $1 - \delta$, comme suit :*

$$e_{\%i} \leq \sqrt{\frac{3}{r}} \sqrt{\frac{1 - x^{2d}}{1 - x^2}} x^{-i} \tag{4.5}$$

Démonstration. Comme $|p_i| = Cx^i$,

$$\|p\|_2^2 = \sum_{i=0}^{d-1} C^2 x^{2i} = C^2 \frac{1 - (x^2)^d}{1 - x^2} \tag{4.6}$$

De (4.4) il suit que :

$$\mathbb{P}\left(|p_i - \widehat{p_i}| \geq \sqrt{\frac{3}{r}} \sqrt{\frac{C^2(1 - x^{2d})}{1 - x^2}} \right) \leq \delta \tag{4.7}$$

En divisant par $|p_i|$, on obtient l'inégalité (4.5) avec une probabilité d'au moins $1 - \delta$. □

L'équation (4.5) montre que plus le décrément x est élevé, meilleure est l'approximation des plus grands paramètres. De plus, elle confirme que plus la largeur du sketch r est élevée, plus la borne ϵ sur l'erreur d'approximation relative est petite. Plus précisément, ϵ diminue avec la racine carrée de r.

5 Étude expérimentale

Dans cette section, nous analysons l'impact de notre stratégie de "count-sketch" pour le modèle populaire de régression linéaire one-vs-rest (LR) appris avec une descente de gradient stochastique. Nous nous concentrons ici sur les données multi-label. Rappelons que les modèles d'apprentissage appliqués aux données multi-label en grande dimension - en particulier la régression one-vs-rest - sont utiles mais nécessitent le stockage de beaucoup de paramètres (voir Tableau 4.1). Ce cadre est donc particulièrement bien adapté à l'évaluation de l'intérêt de la stratégie d'économie de la mémoire que nous proposons. L'objectif des expériences est double. Premièrement, nous comparons la qualité des prédictions obtenues avec la baseline (modèle LR entraîné par descente de gradient stochastique classique) qui stocke tous les paramètres en mémoire à celles obtenues avec deux alternatives économiques en terme de consommation mémoire : notre stratégie basée sur un "count-sketch" et la stratégie standard basée sur le hachage d'attributs. Deuxièmement, nous mesurons la quantité de mémoire économisée par notre proposition. Nous commençons par décrire le protocole expérimental, les jeux de données sélectionnés et la méthodologie pour définir les hyperparamètres (dimensions r et t) du "count-sketch", et enfin nous discutons des résultats obtenus.

5.1 Protocole

Soit $\mathcal{T}$ un échantillon de n instances (x_i, y_i) où $x_i = (x_{ik}) \in \mathbb{R}^{d_x}$ est le vecteur d'attributs et $y_i = (y_{ij}) \in \mathbb{R}^{d_y}$ est le vecteur de labels.

5.1.1 Baseline

Le problème LR vise à trouver la projection P de cardinalité $d = d_y \times d_x$ minimisant l'erreur quadratique moyenne entre les labels "vérité terrain" y_i et les prédictions faites par un modèle linéaire $x_i \mapsto \widehat{y_i} = Px_i$. L'objectif précis est défini par :

$$\min_{P \in \mathbb{R}^{d_y \times d_x}} \sum_{i=1}^{n} \|y_i - Px_i\|_2^2 = \min_{P \in \mathbb{R}^{d_y \times d_x}} \sum_{i=1}^{n} \sum_{j=1}^{d_y} \underbrace{(y_{ij} - \sum_{k=1}^{d_x} P_{jk} \times x_{ik})}_{l_{ij}}{}^2 \tag{4.8}$$

Le problème est ici résolu avec une descente de gradient stochastique [229]. Étant

donnée la perte l_{ij} sur le j^{eme} label de l'instance (x_i, y_i), la direction du gradient pour le paramètre P_{jk} est calculée comme suit :

$$\frac{\partial l_{ij}^2}{\partial P_{jk}} = -2 \times l_{ij} \times x_{ik} \tag{4.9}$$

Les stratégies de "count-sketch" et de hachage d'attributs sont aussi testées pour ce même problème.

5.1.2 Stratégie basée "count-sketch"

La descente de gradient est définie de la même manière que pour la baseline mais la projection n'est pas stockée en mémoire : les paramètres $(P_{jk})_{j=1...d_y, k=1...d_x}$ sont stockés dans la matrice de "count-sketch" M et sont récupérés à chaque besoin. Le schéma de la descente de gradient stochastique avec la stratégie de "count-sketch" est décrit par l'algorithme 1 où :

Algorithme 1 : Stochastic gradient descent for the LR problem

while the validation error decreases **do**
  for random $(i, j) \in \{1, ..., n\} \times \{1, ..., d_y\}$ **do**
    $l \leftarrow y_{ij}$
    for $l \in (1...d_x)$ **do**
      $l \leftarrow l - APPROXPARAM(j, k, M) \times x_{ik}$
    end for
    for $k \in (1...d_x)$ **do**
      $\Delta_{jk} \leftarrow \gamma \times l \times x_{ik}$
      $UPDATE(j, k, M, \Delta_{jk})$
    end for
  end for
end while

— $APPROXPARAM(j, k, M)$ estime le paramètre P_{jk} depuis M avec la médiane des compteurs comme défini dans la formule (4.2) donnée en Section 3. Remarquons que, dans cette description algorithmique, P_{jk} correspond à p_i dans (4.2) à un changement d'indice près (par exemple $i = j \times d_x + k$).
— $UPDATE(j, k, M, \Delta_{jk})$ met à jour M par rapport l'étape de descente $P_{jk} \leftarrow P_{jk} + \Delta_{jk}$ selon la formule (4.1).
— γ est le pas d'apprentissage.

Dans les expériences, nous utilisons le hash "Murmur" [230] avec des graines différentes pour les fonctions de hachage. Cette stratégie de hachage s'est avérée rapide et produit des distributions aléatoires de bonne qualité.

5.1.3 Stratégie basée hachage d'attributs

Le "hashing trick" défini dans la section 2.1 est appliquée à l'ensemble de données initial pour obtenir des instances réduites (x'_i, y_i) où le vecteur d'attribut réduit x'_i a une dimension $d'_x < d_x$. Les équations définies pour la baseline sont ici appliquées à l'ensemble de données réduit et donc la projection du LR, qu'on note ici P', a un cardinal $d' = d'_x \times d_y < d = d_x \times d_y$. Le choix de la dimension d'_x est fixé pour que la consommation de mémoire soit ici égale à celle de la stratégie "count-sketch" :

$$\frac{d_x}{d'_x} = \frac{d}{m} \tag{4.10}$$

Pour mieux comprendre les liens entre les méthodes comparées, la figure 4.4 schématise les différences et similarités entre la baseline, la stratégie basée sur le "count-sketch" et la stratégie basée sur le hachage des attributs.

5.2 Jeux de données d'évaluation

Les différentes approches sont comparées sur neuf jeux de données classiques issus de la littérature multi-label (Mediamill, Enron, Medical, Corel5k, Bibtex, Delicious, Rcv1, WikiSmall et Wiki10-31K) dont les caractéristiques sont détaillées dans l'annexe 1. Le tableau 4.1 donne pour chacun d'entre eux, le nombre de paramètre $d = d_x \times d_y$ de la baseline.

Tableau 4.1 – Nombre de paramètres de la baseline pour les jeux de données utilisés.

Jeu	d
Mediamill	12,120
Enron	53,053
Medical	65,205
Corel5k	186,626
Bibtex	291,924
Delicious	491,500
Rcv1	4,770,129
WikiSmall	82,718,500
Wiki10-31K	3,153,757,844

WikiSmall est une version réduite de Wiki-LSHTC [145] dans laquelle nous avons seulement gardé les 500 labels les plus fréquents et un sous-ensemble des instances qui ont au moins l'un des labels sélectionnés.

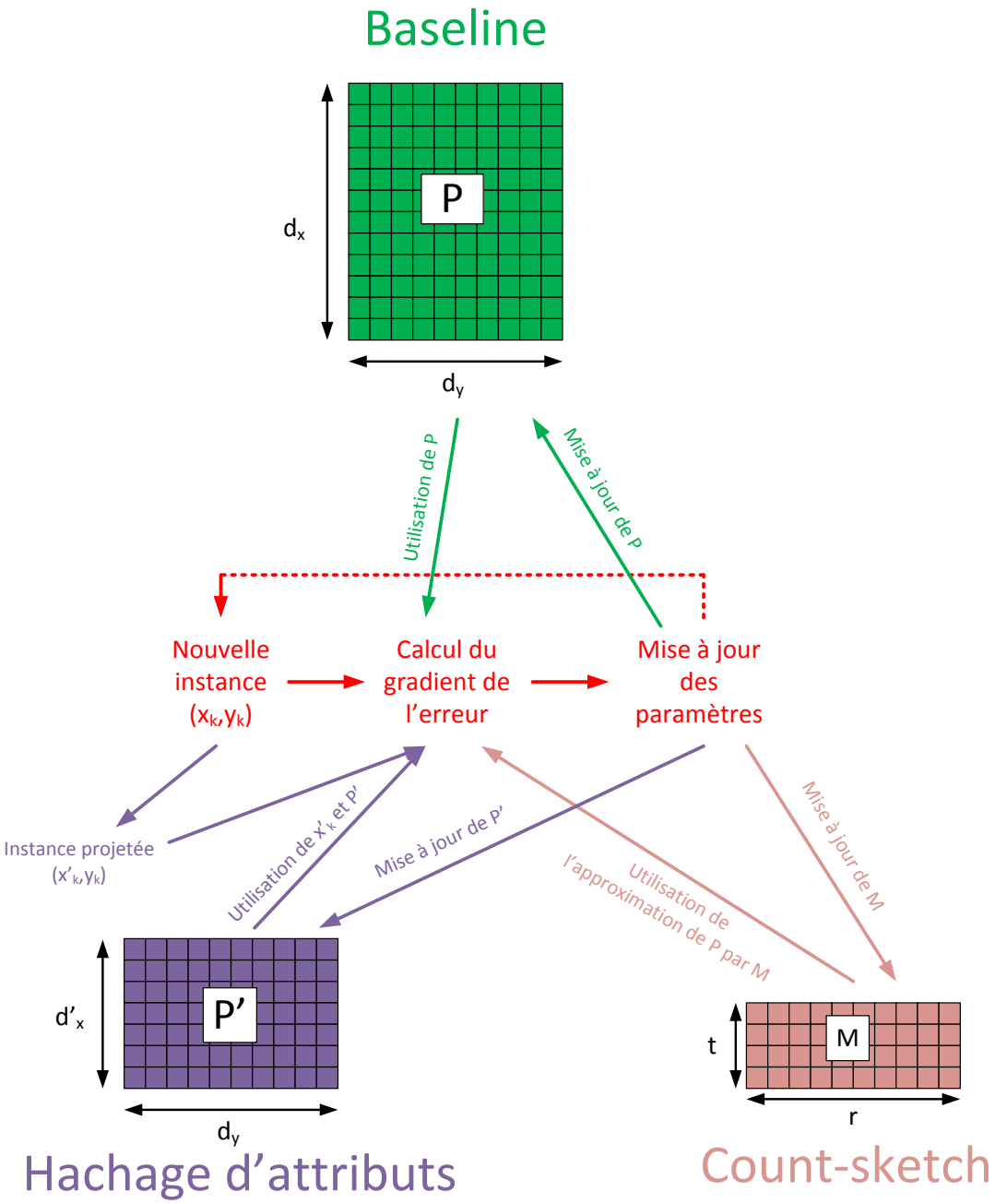

Figure 4.4 – Schéma comparatif, pour une descente de gradient (étapes en rouge), entre la baseline et les deux stratégies d'économie mémoire.

5.3 Méthodologie de validation croisée pour dimensionner automatiquement le sketch

La taille de la matrice de "count-sketch" M dépend des deux hyperparamètres t (nombre d'ensembles de compteurs) et r (nombre de compteurs par ensemble). Dans les expériences, la valeur de t a été fixée à 10 comme précisé dans la section 4.1. La valeur de

r a été fixée avec une méthodologie de validation croisée. La figure 4.5 montre l'évolution d'une erreur de validation par rapport à la taille du "sketch" $m = t \times r$ sur les six plus petits ensembles de données.

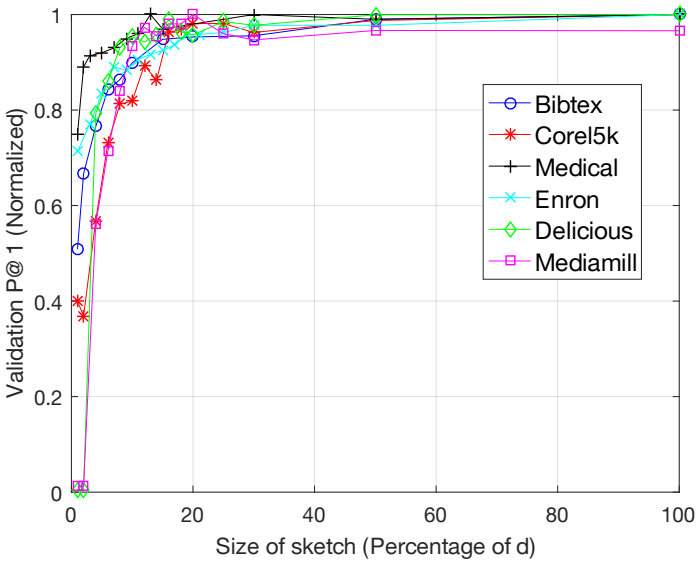

Figure 4.5 – Évolution de la performance en validation $P@1$ (normalisée par sa valeur maximal) en fonction de la taille m du "count-sketch" (exprimée comme un pourcentage de d) pour $t = 10$.

Dans tous les cas, on observe que les performances de validation semblent atteindre un plateau à un pourcentage donné. Pour chaque jeu de données, le pourcentage peut être utilisé pour dimensionner le sketch. De plus, puisque la qualité de la performance n'est pas détériorée au-delà de cette taille, un surdimensionnement est possible si la mémoire disponible est suffisante.

5.4 Résultats

Les résultats des comparaisons sont donnés dans le tableau 4.2 et la figure 4.6. $P@k$ dénote la mesure "Precision à k" qui calcule le pourcentage de labels vérité terrain parmi les k labels prédits avec les scores les plus élevés. Cette mesure est couramment utilisée dans la littérature multi-label extrême.

Notre approche a des performances similaires à celles de la baseline tout en économisant entre 80% et 99,9% de mémoire. C'est un avantage majeur : par exemple, au lieu de consommer 11,7 Go de RAM pour Wiki10-31K comme la baseline, le modèle ne consomme que 12 Mo à n'importe quel instant de l'apprentissage, ce qui est très largement compatible avec les capacités de n'importe quel ordinateur standard ou smartphone. Pour la même quantité de mémoire sauvegardée, les performances du hachage des attributs sont bien plus mauvaises que celles de la baseline : la dégradation relative est de 10% en moyenne et atteint jusqu'à 50% (P@5 sur Wiki10-31K). Ce dernier résultat n'est pas très

Tableau 4.2 – Performances prédictives de la baseline, de la stratégie de hachage des attributs et de notre stratégie basée "count-sketch" pour le problème de régression linéaire. "Gain Mem" dénote le gain en mémoire par rapport à la baseline. Il est défini par $1 - \frac{d'_x}{d_x}$ pour le hachage des attributs et par $1 - \frac{m}{d}$ pour le "count-sketch".

Dataset	Baseline				Feature Hashing				Our strategy ("count-sketch")			
	$P@1$	$P@3$	$P@5$	Size	$P@1$	$P@3$	$P@5$	Gain Mem.	$P@1$	$P@3$	$P@5$	Gain Mem.
Mediamill	0.819	0.637	0.481	47.3 KB	0.821	0.654	0.511	80%	0.806	0.642	0.500	80%
Enron	0.712	0.586	0.429	207.2 KB	0.706	0.596	0.442	80%	0.729	0.588	0.452	80%
Medical	0.876	0.364	0.223	254.7 KB	0.814	0.364	0.223	90%	0.876	0.371	0.223	90%
Corel5k	0.401	0.283	0.229	729.0 KB	0.337	0.244	0.205	80%	0.409	0.285	0.231	80%
Bibtex	0.649	0.397	0.291	1.1 MB	0.540	0.310	0.231	80%	0.631	0.371	0.279	80%
Delicious	0.587	0.539	0.502	1.9 MB	0.542	0.490	0.451	85%	0.586	0.532	0.491	85%
Rcv1	0.910	0.631	0.445	18.2 MB	0.805	0.597	0.434	99%	0.911	0.644	0.451	99%
WikiSmall	0.679	0.360	0.249	315.5 MB	0.618	0.322	0.223	99.5%	0.678	0.359	0.246	99.5%
Wiki10-31K	0.826	0.710	0.627	11.7 GB	0.607	0.397	0.310	99.9%	0.809	0.691	0.593	99.9%

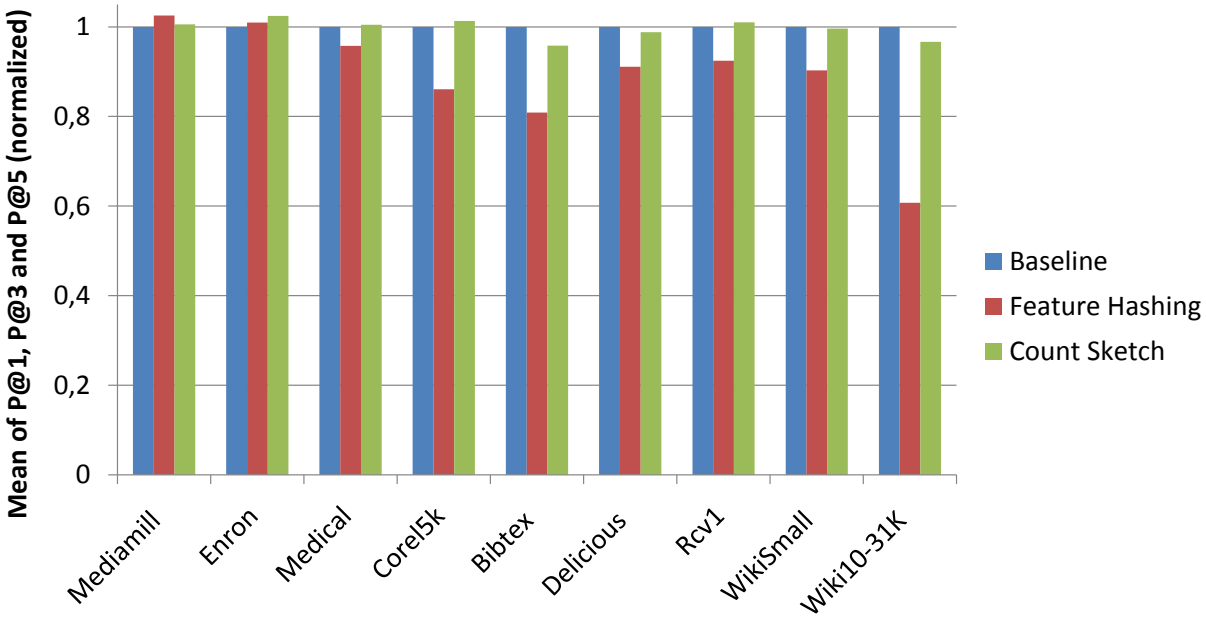

Figure 4.6 – Performances moyennes des trois algorithmes comparés normalisées par celles de la baseline.

surprenant. En effet, dans certains cas, pour atteindre un gain de mémoire important, certains jeux de données (par exemple Wiki10-31K) nécessitent une réduction drastique de l'espace des attributs pouvant entraîner une perte importante d'informations. De plus, on retrouve le problème déjà évoqué de l'approximation de rang faible qui détériore les performances [44].

6 Conclusion

Dans ce chapitre, nous avons proposé une stratégie de stockage de paramètres inspirée de la méthode du "count-sketch" pour réduire considérablement la taille mémoire requise par un modèle entraîné par descente de gradient. Des comparaisons expérimen-

tales complétées par des bornes théoriques confirment la capacité de la stratégie à réduire la consommation de mémoire de plusieurs ordres de grandeur (jusqu'à 1000x) tout en préservant les bonnes qualités prédictives.

Dans la pratique, l'intérêt principal de ce travail est de donner la possibilité à plusieurs modèles potentiellement gourmands en mémoire d'être explorés dans une classification à grande échelle sans avoir à recourir à des environnements de calcul coûteux. Cela ouvre la porte à de nombreuses applications, en particulier sur des appareils tels que les smartphones et les capteurs IoT. Une perspective d'évolution intéressante pour ce travail serait de remplacer la structure "count-sketch" par des propositions plus récentes, comme celle du SF-sketch [231] par exemple, pour explorer le gain obtenu notamment en complexité temporelle.

Dans la suite, la complexité temporelle a été abordée par une approche arborescente qui nous est apparue finalement comme la plus prometteuse pour affronter les dimensions de l'XML.

Chapitre 5

La méthode hiérarchique CRAFTML : efficacité mémoire, rapidité et précision

Sommaire

1 Introduction

Pour faire face à un problème extrême, rappelons qu'une des stratégies efficaces consiste à partitionner hiérarchiquement le problème initial en sous-problèmes à petite échelle [45, 38]. Cette décomposition arborescente présente plusieurs avantages. En découpant l'apprentissage en sous-tâches, elle réduit la complexité de l'entraînement et des prédictions, et sa séquence de décisions successives permet une grande expressivité.

Motivés par ces propriétés, nous présentons ici une nouvelle approche arborescente rapide et précise appelée CRAFTML (Clustering-based RAndom Forest of predictive Trees for extreme Multi-label Learning). Comme PFastReXML [38] qui fait partie des meilleures approches arborescentes pour l'apprentissage multi-label extrême, CRAFTML est une forêt d'arbres de décision entraînés avec la supervision des labels où les conditions de séparations des instances à chaque noeud sont multivariées. Mais CRAFTML a deux différences de fond avec PFastReXML : (i) pour obtenir la diversité, il exploite une stratégie qui s'approche de celle des "forêts aléatoires" mais qui s'en distingue non seulement car il remplace les sélections aléatoires par des projections aléatoires pour préserver plus

d'informations mais aussi car il les applique aux labels en plus des attributs ; (ii) il utilise une nouvelle stratégie de séparation qui a une très faible complexité et qui ne résout pas un problème d'optimisation multi-objectif à chaque noeud.

CRAFTML présente un très bon compromis entre la qualité des performances prédictives, les ressources de calcul nécessaires et la vitesse d'exécution. Des expériences numériques sur neuf jeux de données de la littérature XML montrent qu'il dépasse les approches basées sur les arbres avec un temps d'apprentissage inférieur et une consommation de mémoire plus faible. Il est également compétitif avec les approches les plus performantes de l'état de l'art actuel (DISMEC [42] et PPDSparse [2]). Bien que ses prédictions soient moins bonnes sur quelques jeux de données, ses complexités temporelle et spatiale sont beaucoup plus faibles. Ainsi, contrairement aux deux méthodes qui sont implémentées sur des superordinateurs, il peut être entraîné rapidement sans parallélisation.

Ce chapitre est organisé comme suit. La **section 2** décrit notre nouvelle proposition CRAFTML. La **section 3** analyse sa complexité temporelle et spatiale et discute du choix des valeurs des hyperparamètres. La **section 4** compare les performances de CRAFTML avec les meilleures méthodes de la littérature récente.

2 CRAFTML

Les approches récentes de la littérature (FastXML, PLT, ...) ont obtenu des résultats intéressants sur les données extrêmes. Mais il reste encore une place à l'amélioration en explorant deux directions : utiliser des stratégies de partitionnement très rapides et exploiter la randomisation des arbres. En effet, pour construire les séparateurs dans les noeuds, les approches XML actuelles recourent à des processus d'optimisation complexes qui peuvent potentiellement être remplacés par des opérations plus simples avec des complexités plus faibles. De plus, la diversité des arbres, mise en oeuvre avec une sélection aléatoire des attributs, a contribué au succès des forêts aléatoires [151] et de RF-PCT dans l'apprentissage multi-label. Dans la littérature XML, ceci a été testé avec une sélection aléatoire des attributs dans l'approche MLRF [158] mais les performances prédictives obtenues sont limitées par rapport à FastXML pour une raison principale : sa stratégie de partitionnement [36]. En outre, une projection serait capable de conserver plus d'informations qu'une sélection pour un même taux de compression et une projection aléatoire conjointe des attributs et des labels est plus prometteuse pour traiter le nombre extrême de labels. En suivant ces directions, nous introduisons une nouvelle approche arborescente appelée CRAFTML.

2.1 Les étapes clés

Algorithme 2 : trainTree

Input : Training set with a feature matrix X and a label matrix Y.
Initialize node v
v.isLeaf $\leftarrow$testStopCondition(X, Y)
if v.isLeaf $=$ false **then**
  v.classif $\leftarrow$ trainNodeClassifier(X, Y)
  $(X_{child_i}, Y_{child_i})_{i=0,..,k-1} \leftarrow$ split$(v.\text{classif}, X, Y)$
  for i **from** 0 **to** $k-1$ **do**
    $v.child_i \leftarrow$trainTree$(X_{child_i}, Y_{child_i})$
  end for
else
  $v.\widehat{y} \leftarrow$computeMeanLabelVector(Y)
end if
Output : node v

Algorithme 3 : trainNodeClassifier

Input : feature matrix (X_v) and label matrix (Y_v) of the instance set of the node v.
$X_s, Y_s \leftarrow$ sampleRows(X_v, Y_v, n_s)
$X'_s \leftarrow X_s P_x$ # Étape 1 : projection aléatoire des attributs
$Y'_s \leftarrow Y_s P_y$ # Étape 1 : projection aléatoire des labels
$\boldsymbol{c} \leftarrow k$-means$(Y'_s, k)$ # Étape 2 : clustering par rapport aux labels
$(\boldsymbol{c} \in \{0, ..., k-1\}^{\min(n_v, n_s)})$
for i **from** 0 **to** $k-1$ **do**
  $(\text{classif})_{i,.} \leftarrow$ computeCentroid$(\{(X'_s)_{j,.} | c_j = i\})$ # Étape 3 : Apprentissage du classifieur multi-classe simple.
end for
Output : Classifier classif $(\in \mathbb{R}^{k \times d'_x})$.

$\boldsymbol{c}$ est un vecteur tel que la $j^{\text{ème}}$ composante c_j correspond à l'indice du cluster auquel la $j^{\text{ème}}$ instance, associée à $(X'_s)_{j,.}$ et $(Y'_s)_{j,.}$, appartient.

CRAFTML construit une forêt F de m_F arbres d'instances k-aires (e.g binaires si $k = 2$) dont la construction suit le schéma commun des méthodes basées sur l'arbre d'instances (voir Algorithme 2) rappelé dans la section 3 du chapitre 2. La condition d'arrêt du partitionnement récursif d'un arbre est classique : soit (i) le nombre d'instances dans le noeud est inférieur à un seuil donné n_{leaf}, soit (ii) toutes les instances du noeud ont les mêmes attributs, soit (iii) toutes les instances du noeud ont les mêmes labels. Une fois qu'un arbre a été formé, chaque feuille stocke un vecteur correspondant à la moyenne des vecteurs de labels de ses instances. Similairement à FastXML, la raison du partitionnement des noeuds de CRAFTML est de regrouper des instances avec des labels communs dans un même sous-ensemble mais notre stratégie est très différente car nous visons à satisfaire deux contraintes : le calcul de la partition doit être basé sur des instances projetées aléatoirement pour assurer la diversité et il doit effectuer des opérations de faible complexité pour le passage à l'échelle. Par conséquent, l'étape

d'apprentissage des noeuds dans CRAFTML est décomposée en trois étapes consécutives (voir Algorithme 3 et figure 5.1) :

1. Projeter aléatoirement, dans des espaces de dimension réduite, les attributs et les labels des instances du noeud.
2. Partitionner, avec l'algorithme des k-moyennes, les instances en k sous-ensembles temporaires à partir de leurs labels projetés.
3. Apprendre un classifieur multi-classe simple pour assigner chaque instance à son sous-ensemble temporaire pertinent (c'est-à-dire l'indice de cluster calculé à l'étape 2) à partir de son vecteur d'attributs. Puis, partitionner les instances en k sous-ensembles finaux (noeuds enfants) avec le classifieur appris ("split" dans Algorithm 2).

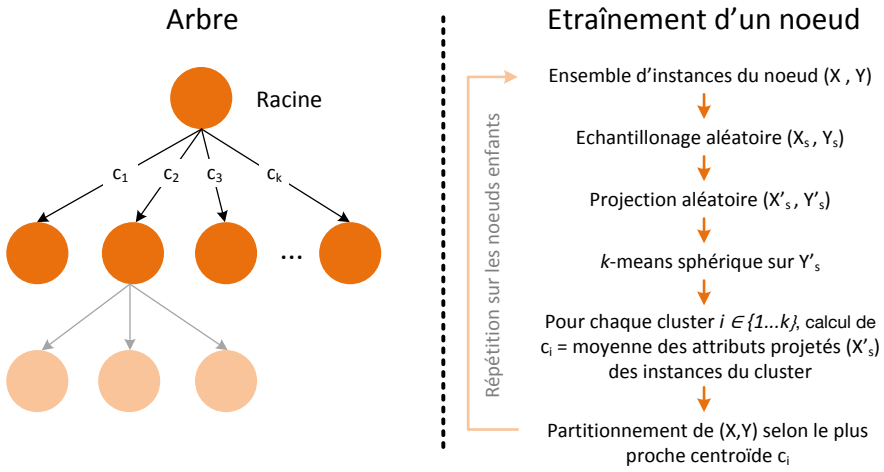

Figure 5.1 – Schéma du principe d'apprentissage de CRAFTML.

Dans la phase de prédiction, pour chaque arbre, l'instance d'entrée suit un chemin racine-feuille déterminé par les décisions successives des classifieurs et la prédiction fournie est le vecteur de labels stocké dans la feuille atteinte. La forêt agrège les prédictions des arbres avec une simple moyenne.

Précisons que contrairement aux forêts aléatoires classiques qui utilisent des "bootstraps", chaque arbre de CRAFTML est entraîné sur l'ensemble de données initial complet car, en XML, les échantillons d'instances peuvent couvrir un faible nombre de labels. La diversité des arbres reste néanmoins assurée grâce aux projections aléatoires. Les principales similitudes et différences entre CRAFTML et les autres approches arborescentes de l'état de l'art sont résumées dans le tableau 5.1.

Tableau 5.1 – Comparaison des caractéristiques des méthodes arborescentes. **depend de la taille mémoire, **selection aléatoire d'attributs.*

	Arbres de labels		Arbres d'instances			
	HOMER	PLT	RF-PCT	LPSR	FastXML	CRAFTML
Adapté à l'XML	Non	Oui	Non	Oui	Oui	Oui
Projections des attributs	Non	Oui*	Oui**	Non	Non	Oui
Projections des labels	Non	Non	Non	Non	Non	Oui
Plusieurs arbres	Non	Non	Oui	Non	Oui	Oui
Arbres binaires	Non	Non	Oui	Non	Oui	Non
Condition de séparation multivariée	Oui	Oui	Non	Oui	Oui	Oui

2.2 Détails sur l'apprentissage du noeud

Nous détaillons ici les trois étapes du processus de partitionnement d'instances dans chaque noeud v d'un arbre T d'une forêt F.

Étape 1 : projection aléatoire des instances de v Les vecteurs d'attributs et de labels x et y de chaque instance de v sont projetés dans un espace de dimension inférieure : $x' = xP_x$ et $y' = yP_y$ où P_x (resp P_y) est une matrice de projection aléatoire de $\mathbb{R}^{d_x \times d'_x}$ (resp. $\mathbb{R}^{d_y \times d'_y}$) et d'_x (resp. d'_y) est la dimension de l'espace des attributs réduits (resp. labels réduits). Les matrices de projection sont différentes d'un arbre à l'autre. Pour optimiser la mémoire, les coefficients des matrices de projection ne sont pas stockés mais générés avec une graine et un "hash" sur demande. Nous avons testé deux projections aléatoires : une projection générée à partir d'une distribution gaussienne standard et une projection orthogonale creuse de type hashing trick [220], décrit dans la section 2.1 du chapitre 4, qui n'a qu'un seul paramètre non nul avec une valeur de -1 ou $+1$ sur chaque ligne. Les comparaisons nous ont amenés à privilégier le "hashing trick". Il conduit statistiquement à des performances légèrement meilleures. De plus, grâce à la faible densité de sa projection, il est beaucoup plus rapide. En pratique, avec le "hashing trick", les vecteurs d'attributs (resp. de labels) projetés ont au plus autant d'éléments non nuls que les originaux qui en ont s_x (resp. s_y) en moyenne et d'autres composants de la forêt peuvent donc être accélérés.

En outre, nous avons exploré l'impact de la diversité des matrices de projection entre les noeuds d'un même arbre T. Nous avons considéré quatre combinaisons : SxSy, SxDy, DxSy et DxDy où Sx (resp. Sy) est le cas où les projections des attributs (resp. labels) sont les mêmes dans chaque noeud de T et Dx (resp. Dy) est le cas où les projections des attributs (resp. labels) sont différentes d'un noeud à l'autre. Les comparaisons présentées dans la section 3.2 nous ont conduits à privilégier SxSy.

Étape 2 : partitionnement des instances en k sous-ensembles temporaires Soit Y_s la matrice des labels d'un échantillon tiré sans remise de taille au plus n_s de l'ensemble des instances de v. L'échantillon est partitionné avec l'algorithme des k-moyennes sphériques (algorithme de Loyd) appliqué sur $Y_s P_y$. La métrique du cosinus est utilisée car elle est rapide et adaptée aux données creuses. Les centroïdes des clusters sont initialisés avec la stratégie k-means ++ [232] qui améliore la stabilité des clusters et les performances de l'algorithme par rapport à une initialisation aléatoire.

Étape 3 : assigner une instance à un sous-ensemble à partir de ses attributs projetés Le classifieur multi-classe qui réalise cette tache est très simple : dans chaque sous-ensemble temporaire, il calcule le centroïde des vecteurs d'attributs projetés des instances. Dans la phase de prédiction, le classifieur assigner le sous-ensemble dont le centroïde est le plus proche du vecteur d'attributs projetés de l'instance d'entrée. Deux métriques (cosinus et euclidienne) ont été comparées et, pour des raisons similaires à celles de l'étape 2, le cosinus est plus adapté. En outre, à la place de ce classifieur très simple, nous avons également testé un modèle linéaire one-vs-rest, mais la variante de CRAFTML en résultant était moins précise et beaucoup plus lente.

Afin de faciliter la lecture de la suite de ce chapitre, quelques notations sont résumées ici :

- m_F : nombre d'arbres dans la forêt.
- n_s : taille de l'échantillon pour l'apprentissage du séparateur dans les noeuds.
- d'_x : dimension du vecteur d'attributs projeté.
- d'_y : dimension du vecteur de labels projeté.
- k : facteur de branchement de l'arbre.
- v : noeud.
- T : arbre.
- F : forêt.
- n_{leaf} : nombre d'instances maximum par feuille.

3 Analyse de l'algorithme et choix des hyperparamètres

Dans cette section, les complexités temporelle et spatiale de CRAFTML sont explicitées. Puis, l'impact des hyperparamètres sur les propriétés de l'algorithme sont analysées pour déduire un paramétrage standard adapté aux données multi-label extrême.

3.1 Complexités spatiales et temporelles

Pour un noeud v d'un arbre T, notons n_v le nombre d'instances du sous-ensemble associé. Le nombre i d'itérations du k-means sphérique est fixé *a priori*.

Lemme 1. *Pour un noeud v d'un arbre T, la complexité temporelle C_v est $O\big(n_v \times C\big)$ où $C = k \times (i \times s_y + s_x)$ est la complexité par instance.*

Démonstration. La complexité temporelle d'un noeud v est la somme des complexités du k-means sphérique initialisé avec k-means++, de l'apprentissage du classifieur multi-classe et de ses prédictions sur les instances du sous-ensemble d'instances associé à v. L'algorithme k-means est appliqué sur les labels projetés et sa complexité est bornée par la somme de la complexité $O(i \times n_v \times s_y \times k)$ de l'algorithme de Loyd et de la complexité $O(n_v \times s_y \times k)$ de k-means++. L'apprentissage du classifieur est basé sur le calcul du centroïde de chaque cluster dans l'espace des attributs projetés, ce qui conduit à une complexité totale $O(n_v \times s_x)$. Pour ses prédictions, le classifieur calcule la distance aux k centroïdes pour chaque instance, ce qui nécessite $O(n_v \times k \times s_x)$ opérations. □

En pratique, la complexité de l'apprentissage du noeud est inférieure à la borne donnée dans le lemme 1. En effet, le k-means et le classifieur sont appliqués sur un échantillon d'instances de taille $\min(n_v, n_s) \leq n_v$. En outre, dans nos expériences, CRAFTML atteint déjà ses meilleures performances avec seulement $i = 2$ itérations de k-means.

Considérons maintenant un arbre k-aire T et notons l_T son nombre de feuilles, $m_T = \frac{l_T - 1}{k-1}$ son nombre de noeuds et $\overline{n}_T = \frac{\sum_{v \in T} n_v}{m_T}$ le nombre moyen d'instances dans ses noeuds.

Proposition 3. *Si l'arbre T est équilibré, sa complexité temporelle d'apprentissage C_T est $O\big(log_k(\frac{n}{n_{leaf}}) \times n \times C\big)$. Sinon, C_T est $O\big(\frac{l_T - 1}{k-1} \times \overline{n}_T \times C\big)$.*

Démonstration. Si T est un arbre équilibré, le j^{eme} niveau T_j de T a k^j noeuds et pour chaque noeud $v \in T_j$, $n_v = \frac{n}{k^j}$. D'après le lemme 1, la complexité du j^{eme} niveaux est donc $O\big(\frac{n}{k^j} \times k^j \times C\big)$ qui est indépendent de j. En outre, du fait de la condition terminale choisie (le noeud n'est pas divisé s'il a moins de n_{leaf} instances), le nombre de niveaux est $O(log_k(\frac{n}{n_{leaf}}))$. Finalement, le produit entre le nombre de niveau et la complexité d'apprentissage de chaque niveau donne la complexité C_T de l'arbre.

Si T est un arbre déséquilibré, la complexité temporelle de T qui est la somme des complexités de ses noeuds est $O\big(\sum_{v \in T} n_v \times C\big)$. Il suffit de substituer $\sum_{v \in T} n_v$ par $m_T \times \overline{n}_T$ pour terminer la preuve. □

La complexité dépend du nombre de feuilles l_T et du nombre moyen d'instances dans les noeuds $\overline{n}_T$. En théorie, l_T est compris entre $\frac{n}{n_{leaf}}$ pour le meilleur des cas et n pour le pire des cas. Dans nos expériences, nous avons observé sur tous les jeux de données XML que, en fixant $n_{leaf} = 10$ et $k = 2$, on obtient $l_T = \frac{n}{2.83 \pm 0.41}$ et $\overline{n}_T = 24.32 \pm 2.61$.

Notons qu'avec le ratio $\frac{C}{k-1} = \frac{k \times (i \times s_y + s_x)}{k-1}$, la contribution de k dans la complexité temporelle disparaît pour l'arbre déséquilibré. En outre, en exploitant la parcimonie des données (s_x et s_y), la complexité temporelle est indépendante des dimensions de projection d'_x et d'_y. Dans les jeux de données XML où les instances sont très parcimonieuses, s_x et s_y

sont beaucoup plus petits que d_x et d_y. De plus, la complexité de la projection aléatoire qui est négligeable par rapport aux autres opérations n'est pas considérée ici. Par exemple, dans un arbre T, la complexité de la combinaison SxSy la plus rapide choisie dans nos expériences est égale à $O\big(n \times (s_x + s_y) \times C_{gen}\big)$ où C_{gen} est la complexité pour générer un coefficient des matrices de projection [1]. Il est également important de souligner que, en pratique, C_T est au-dessus de la réalité en raison de l'échantillonnage d'instances dans chaque noeud v qui réduit les complexités des k-means sphériques et des calculs de classifieurs.

Proposition 4. *La complexité mémoire d'un arbre T est $O\big(n \times s_y + m_T \times k \times d'_x\big)$.*

Démonstration. Chaque feuille de T stocke la moyenne des vecteurs de labels de ses instances. Dans le meilleur des cas, toutes les $\frac{n}{l_T}$ instances ont les mêmes labels et le vecteur contient alors en moyenne s_y éléments non nuls. Dans le pire des cas, les $\frac{n}{l_T}$ instances ont des labels différents et le vecteur contient approximativement $\frac{s_y \times n}{l_T}$ éléments non nuls en moyenne. Par conséquent, la mémoire requise pour les feuilles est $O\big(l_T \times \frac{s_y \times n}{l_T}\big)$. Chaque noeud de T stocke un classifieur multi-classe représenté par les k vecteurs d'attributs des centroïdes de dimension d'_x ; la mémoire requise est $m_T \times k \times d'_x$. □

En pratique, la limite déterminée dans la proposition 4 est significativement supérieure à la mémoire requise pour deux raisons. Premièrement, les instances regroupées dans une même feuille avec le clustering partagent de nombreux labels communs et le pire cas ne se produit pas. Deuxièmement, les centroïdes stockés dans les noeuds ont des composantes parcimonieuses, en particulier pour les noeuds distants de la racine car ils sont calculés comme la moyenne sur un petit sous-ensemble de vecteurs similaires.

3.2 Analyse des performances

Dans cette section, nous explorons l'effet respectif de cinq hyperparamètres (d'_x, d'_y, m_F, n_{leaf}, n_s) sur les performances prédictives de CRAFTML pour cinq jeux de données provenant du repository apprentissage multi-label extrême [2] : quatre jeux de données standards issus de la littérature sur l'apprentissage multi-label (Bibtex, Mediamill, Delicious, EURLex-4K) avec un maximum de 5000 attributs et 3993 labels et un grand jeu (Wiki10-31K) avec 101938 attributs et 30938 labels (voir Tableau 5.2 pour plus de détails). La qualité des résultats est mesurée avec la précision à 1 (P@1), 3 (P@3) et 5 (P@5). Pour éviter le sur-apprentissage des données de tests, nous nous limitons dans cette section à la partie apprentissage des jeux de données : un ensemble de validation avec 20% des instances est utilisé pour l'évaluation.

1. Dans nos expériences, les coefficients sont générés avec MurmurHash3 [230] pour les raisons rappelées page 81.

2. `http://manikvarma.org/downloads/XC/XMLRepository.html`

Impact de la projection Les figures 5.2 et 5.3 montrent l'impact des dimensions de projection et du choix de la stratégie de diversité sur les performances de CRAFTML avec $m_F = 50$. Les performances augmentent significativement avec d'_x, moins avec d'_y et les deux évolutions atteignent un plateau qui varie selon les caractéristiques du jeu de données. Lorsque les deux dimensions d'_x et d'_y sont assez grandes (> 500), l'effet des projections aléatoires devient positif (comparaison avec ou sans projection dans la figure 5.2). Les quatre combinaisons de présence/absence de diversité entre les noeuds d'un même arbre SxSy, SxDy, DxSy et DxDy conduisent à des performances très proches (Figure 5.3).

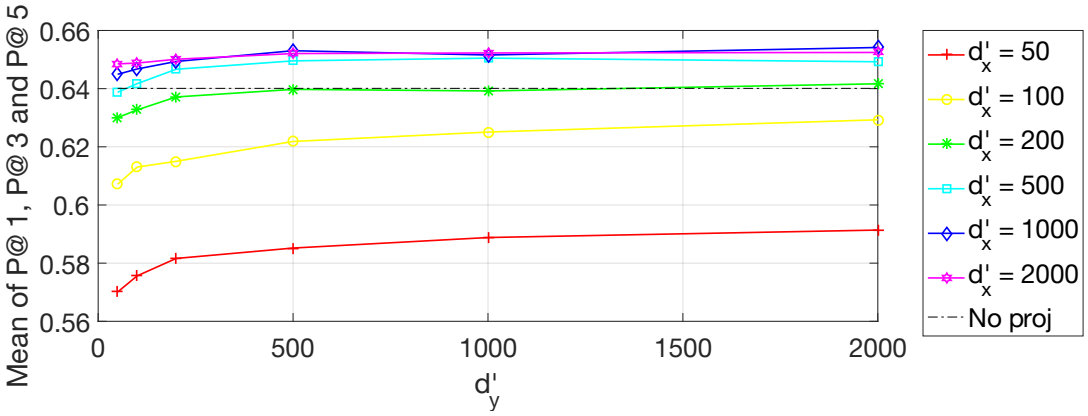

Figure 5.2 – Évolution, sur l'exemple de Eurlex-4K et sur la variante "SxDy", des performances d'une forêt de cinquante arbres en fonction de d'_y, pour six valeur de d'_x. Les courbes pour les autres variantes (SxSy,DxSy,DxDy) et les autres jeux de données ont la même forme. Les performances d'une forêt sans projection aléatoire apparaissent en trait pointillé sur la figure.

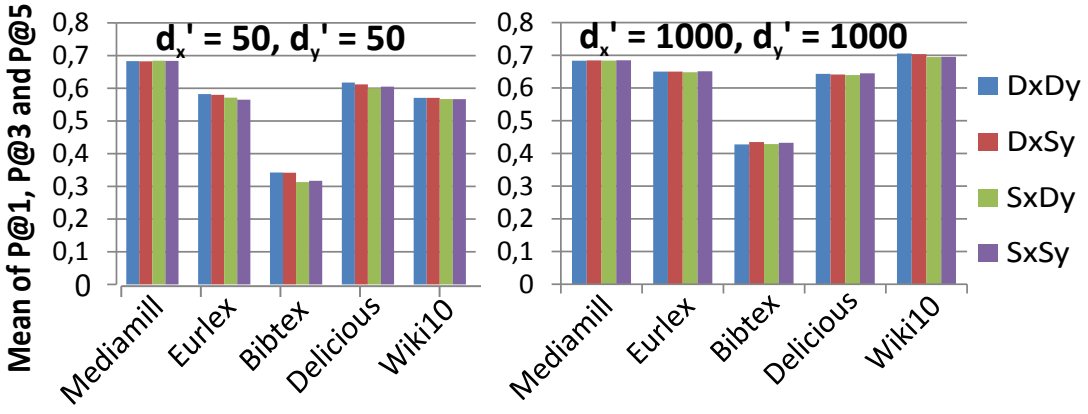

Figure 5.3 – Comparaison des quatre variantes de diversité (SxSy,SxDy,DxSy,DxDy) pour deux valeurs de (d'_x,d'_y) pour une forêt de cinquante arbres.

Impact du nombre d'arbres La contribution des noeuds dans la complexité mémoire de la forêt (proposition 4) dépend linéairement du produit $m_F \times d'_x$. Cela soulève une question : les meilleures performances sont-elles obtenues avec quelques arbres ayant une grande dimension d'_x ou avec un grand nombre d'arbres de petite dimension ? Intuitivement plusieurs arbres sont nécessaires pour bénéficier de l'effet d'agrégation mais la dimension doit être raisonnable pour préserver la précision de chaque arbre. La figure 5.4 confirme cette intuition. Elle montre que l'optimum est atteint pour un point d'équilibre entre m_F et d'_x dont la valeur varie avec l'ensemble de données.

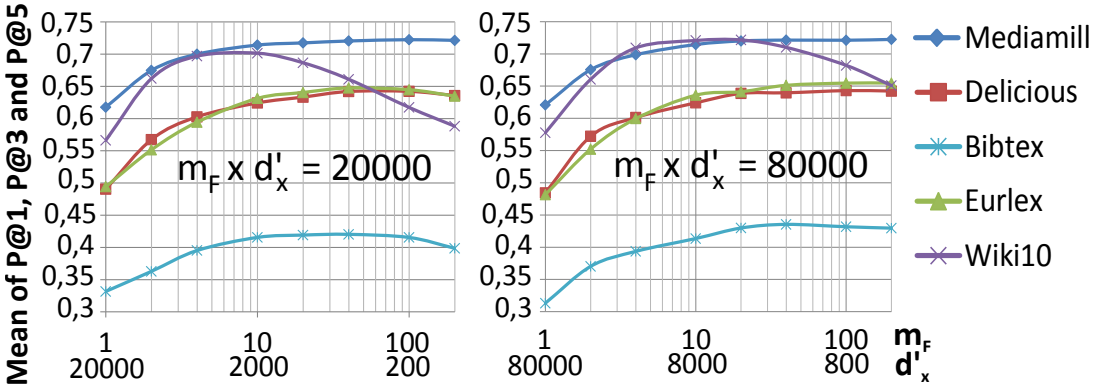

Figure 5.4 – Évolution des performances en fonction du nombre d'arbre m_F, à iso-produit $m_F \times d'_x$.

Impact de la profondeur La profondeur de l'arbre est contrôlée par la condition d'arrêt et surtout par le nombre maximal d'instances par feuille n_{leaf}. Pour un unique arbre, un grand n_{leaf} est nécessaire pour éviter le sur-apprentissage des données d'entraînement (graphique gauche sur la Figure 5.5), mais pour une forêt, un ensemble faible d'instances par feuille crée plus de diversité entre les arbres et améliore les performances (graphique de droite sur la Figure 5.5). Ce phénomène est bien connu dans la littérature des forêts aléatoires [151].

Impact de la taille de l'échantillon dans le k-means sphérique Pour l'ensemble de données avec le plus grand nombre d'instances $n = 1.7M$ (WikiLSHTC-325K), les performances s'améliorent avec la taille de l'échantillon jusqu'à atteindre un plateau pour $n_s = 20000$. Les impacts sur $P@1$, sur $P@3$ et sur $P@5$ sont presque identiques.

3.3 Paramétrage standard pour CRAFTML

Des expériences ont montré que CRAFTML nécessite quelques dizaines d'arbres profonds et de grandes dimensions de projection ($d'_x > 5000$ et $d'_y > 5000$). Pour limiter

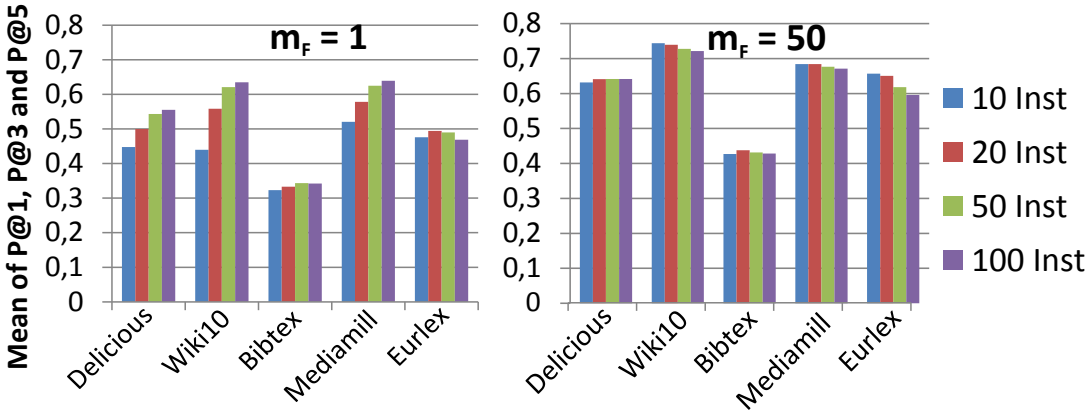

Figure 5.5 – Performances d'un arbre ($m_F = 1$) et d'une forêt de cinquante arbres ($m_F = 50$) pour quatre valeurs différentes de n_{leaf}. Les performances d'un seul arbre sont moyennées sur trente répétitions.

les effets de taille dans les comparaisons expérimentales, le nombre choisi d'arbres et la condition d'arrêt sont les mêmes que pour FastXML : $m_F = 50$ et $n_{leaf} = 10$ [36]. Comme indiqué dans la section 3.1, la dimension de projection des labels d'_y n'affecte pas les complexités de temps et de mémoire et a donc été fixée à une valeur élevée arbitraire : $d'_y = min(d_y, 10000)$. La dimension de projection des attributs d'_x n'a pas non plus d'effet sur le temps et très peu sur la mémoire en pratique. CRAFTML atteint semble atteindre un plateau des performance pour chaque ensemble de données pour une taille d'échantillon $n_s = 20000$ et une dimension $d'_x = min(d_x, 10000)$. Et sa taille de modèle mesurée est faible (Table 5.3) pour ce paramétrage. De plus, pour les valeurs choisies de m_F, n_{leaf}, d'_x et d'_y, CRAFTML obtient des performances proches avec les quatre variations SxSy, SxDy, DxSy et DxDy. Nous utilisons SxSy qui est la combinaison la plus rapide.

4 Comparaisons expérimentales

Dans cette section, nous comparons CRAFTML avec les neuf meilleures méthodes de l'état de l'art dans l'apprentissage multi-label extrême présentées dans le chapitre 2 : quatre méthodes arborescentes (FastXML, PFastReXML, LPSR, PLT) et cinq autres en distinguant celles adaptées aux machines avec un seul coeur (SLEEC, AnnexML, PDSparse) et celles spécialement conçues pour une implémentation parallèle (DISMEC, PPDSparse). Les expériences numériques sont réalisées sur neuf ensembles de données XML provenant de différents domaines d'application : Bibtex, Mediamill, Delicious, EURLex-4K, Wiki10-31K, Delicious-200K, AmazonCat-13K, WikiLSHTC-325K, Amazon-670K. Les nombres d'instances, d'attributs et de labels sont rappelés dans la première colonne du tableau 5.2 et des détails supplémentaires sont disponibles dans le repository

Tableau 5.2 – Comparaison entre CRAFTML et l'état de l'art sur les ensembles de test des jeux de données classiques de l'apprentissage multi-label extrême. Le nombre de labels d_y, le nombre d'attributs d_x, le nombre d'instances d'apprentissage n et le nombre d'instances de test $n_{\mathcal{S}}$ sont rappelées. Le meilleur résultat parmi les méthodes arborescentes est présenté en gras. Le meilleur résultat parmi toutes les méthodes est souligné.

		Méthodes arborescentes					Autres méthodes				
		CRAFTML	PFastReXML	FasTXML	LPSR	PLT	SLEEC	AnnexML	PDSparse	DISMEC	PPDSparse
Mediamill	P@1	**85.86**	83.98	84.22	83.57		87.82	-	83.64	84.83	84.42
$d_x = 120$, $d_y = 101$	P@3	**69.01**	67.37	67.33	65.78	-	73.45	-	66.13	67.17	67.26
$n = 30993$, $n_{\mathcal{S}} = 12914$	P@5	**54.65**	53.02	53.04	49.97	-	59.17	-	50.90	52.80	52.78
Bibtex	P@1	**65.15**	63.46	63.42	62.11	-	65.08	-	62.36	63.69	63.69
$d_x = 1836$, $d_y = 159$	P@3	**39.83**	39.22	39.23	36.65	-	39.64	-	36.50	38.80	39.43
$n = 4880$, $n_{\mathcal{S}} = 2515$	P@5	28.99	**29.14**	28.86	26.53	-	28.87	-	26.50	28.30	28.67
Delicious	P@1	**70.26**	67.13	69.61	65.01	-	67.59	-	-	-	-
$d_x = 500$, $d_y = 983$	P@3	63.98	62.33	**64.12**	58.96	-	61.38	-	-	-	-
$n = 12920$, $n_{\mathcal{S}} = 3185$	P@5	59.00	58.62	**59.27**	53.49	-	56.56	-	-	-	-
EURLex-4K	P@1	**78.81**	75.45	71.36	76.37	-	79.26	-	75.90	82.40	74.61
$d_x = 5000$, $d_y = 3993$	P@3	**65.21**	62.70	59.90	63.36	-	64.30	-	61.16	68.50	59.56
$n = 15539$, $n_{\mathcal{S}} = 3809$	P@5	**53.71**	52.51	50.39	52.03	-	52.33	-	50.83	57.70	48.43
Wiki10-31K	P@1	**85.19**	83.57	83.03	72.72	84.34	85.88	86.50	-	85.20	-
$d_x = 101938$, $d_y = 30938$	P@3	**73.17**	68.61	67.47	58.51	72.34	72.98	74.28	-	74.60	-
$n = 14146$, $n_{\mathcal{S}} = 6616$	P@5	**63.27**	59.10	57.76	49.50	62.72	62.70	64.19	-	65.90	-
WikiLSHTC-325K	P@1	**56.57**	56.05	49.75	27.44	45.67	54.83	63.36	60.70	64.40	64.13
$d_x = 1617899$, $d_y = 325056$	P@3	34.73	**36.79**	33.10	16.23	29.13	33.42	40.66	39.62	42.50	42.10
$n = 1778351$, $n_{\mathcal{S}} = 587084$	P@5	25.03	**27.09**	24.45	11.77	21.95	23.85	29.79	29.20	31.50	31.14
Delicious-200K	P@1	**47.87**	41.72	43.07	18.59	45.37	47.85	46.66	37.69	45.50	45.05
$d_x = 782585$, $d_y = 205443$	P@3	**41.28**	37.83	38.66	15.43	38.94	42.21	40.79	30.16	38.70	38.34
$n = 196606$, $n_{\mathcal{S}} = 100095$	P@5	**38.01**	35.58	36.19	14.07	35.88	39.43	37.64	27.01	35.50	34.9
Amazon-670K	P@1	37.35	**39.46**	36.99	28.65	36.65	35.05	42.08	-	44.70	43.04
$d_x = 135909$, $d_y = 670091$	P@3	33.31	**35.81**	33.28	24.88	32.12	31.25	36.65	-	39.70	38.24
$n = 490449$, $n_{\mathcal{S}} = 153025$	P@5	30.62	**33.05**	30.53	22.37	28.85	28.56	32.76	-	36.10	34.94
AmazonCat-13K	P@1	92.78	91.75	**93.11**	-	91.47	90.53	93.55	87.43	93.40	92.72
$d_x = 203882$, $d_y = 13330$	P@3	**78.48**	77.97	78.20	-	75.84	76.33	78.38	70.48	79.10	78.14
$n = 1186239$, $n_{\mathcal{S}} = 306782$	P@5	63.58	**63.68**	63.41	-	61.02	61.52	63.32	56.70	64.10	63.41

XML et dans l'annexe 1. Les précisions des algorithmes de l'état de l'art considérés sont extraits des derniers résultats publiés [3]. Les hyperparamètres choisis pour CRAFTML sont ceux décrits dans la section précédente. Le tableau 5.2 présente les résultats des performances prédictives et le tableau 5.3 les temps d'apprentissage/prediction et la taille du modèle.

4.1 Comparaison avec les méthodes arborescentes

CRAFTML dépasse les meilleures méthodes arborescentes dans la plupart des cas. Pour les jeux de données WikiLSHTC-325K et Amazon-670K, la domination de PFastreXML s'explique en partie par le fait qu'il est entraîné avec les propensions des labels calculées avec des informations externes supplémentaires (hiérarchie des labels de Wikipedia et Amazon) [38]. Les comparaisons de temps de calcul sont à considérer avec précaution car les approches ont été développées avec des langages différents (Java pour CRAFTML) et les temps ont été mesurés sur différentes machines. Néanmoins le tableau

3. du repository XML pour PFastReXML, FastXML, LPSR-NB, SLEEC et DISMEC, de [45] pour PLT, de [2] pour PDSparse, PPDSparse et les résultats restants de DISMEC, et de [144] pour AnnexML

Tableau 5.3 – Temps d'apprentissage, temps de prédiction et tailles de modèles pour les algorithmes comparés sur des jeux de données de grandes dimensions. Les valeurs pour FastXML, PFastReXML, SLEEC, PDSparse, DISMEC et PPDSparse sont extraites de [2] et des conditions similaire (même quantité de RAM, CPU de la même gamme) ont été fixées pour nos mesures. Pour CRAFTML, le temps d'apprentissage entre parenthèses a été calculé avec une parallélisation sur cinq coeurs.

Machine		1 core						100 cores	
Langage		Java		C++				C++	
Algorithme		CRAFTML Forêt	CRAFTML Arbre (Forêt/50)	FastXML	PFastReXML	SLEEC	PDSparse	DISMEC	PPDSparse
EURLex-4K	Apprentissage (s)	196 (47.75)	3.92	315.9	324.4	4543.4	773.2	76.07	9.95
	Test (ms)	5.39	0.1078	3.65	5.43	3.67	0.73	2.26	1.5
	Taille modèle (Mb)	30	0.6	384	455	121	25	15	9.5
WikiLSHTC-325K	Apprentissage (s)	18508 (5092)	370.16	19160	20070	39000	94343	271407	353
	Test (ms)	7.67	0.1534	1.02	1.47	4.85	3.89	65	290
	Taille modèle (Gb)	1.06	0.021	14	16	0.635	0.534	8.1	4.9
Delicious-200K	Apprentissage (s)	4754 (1174)	95.08	8832.46	8807.51	4838.7	5137.4	38814	2869
	Test (ms)	8.6	0.172	1.28	7.4	2.685	0.432	311.4	275
	Taille modèle (Gb)	0.346	0.007	1.3	20	2.1	0.004	18	9.4
Amazon-670K	Apprentissage (s)	5653 (1487)	113.06	5624	6559	20904	-	174135	921.9
	Test (ms)	5.02	0.1004	1.41	1.98	6.94	-	148	20
	Taille modèle (Gb)	0.494	0.010	4.0	6.3	6.6	-	8.1	5.3
AmazonCat-13K	Apprentissage (s)	10606 (2876)	212.12	11535	13985	119840	2789	11828	122.8
	Test (ms)	5.12	0.1024	1.21	1.34	13.36	0.87	0.2	1.82
	Taille modèle (Gb)	0.659	0.013	9.7	11	12	0.015	2.1	0.347

5.3 confirme que CRAFTML est très compétitif. Comparé aux autres méthodes basées sur les arbres, son temps d'entraînement observé est inférieur en moyenne et la taille de son modèle est plus petite. Ces mesures sont cohérentes avec les résultats théoriques : en raison de la stratégie d'échantillonnage et de la réduction de la dimension résultant des projections aléatoires, le temps d'apprentissage et les complexités de la mémoire de CRAFTML sont les plus faibles. Le temps de prédiction de CRAFTML est plus élevé que les autres arbres même si sa complexité est équivalente.

4.2 Comparaison avec d'autres modèles sans parallélisation (SLEEC, PDSparse, AnnexML)

Les performances de CRAFTML sont meilleures que celles de PD-Sparse à l'exception de WikiLSHTC-325K. Elles sont équivalentes à celles de SLEEC mais il y a une légère domination de CRAFTML sur les quatre plus grands jeux de données. De plus, CRAFTML est plus rapide que SLEEC et que PDSparse excepté sur le jeu de données AmazonCat-13K. Les spécificités de ce jeu de données - un petit nombre de labels et un grand nombre d'instances- favorisent PDSparse. La taille du modèle CRAFTML est inférieure à celle de SLEEC -sauf pour WikiLSHTC-325K-, mais elle est supérieure à celle de PDSparse : 1,2 fois pour EURLex-4K, 1,98 fois pour WikiLSHTC-325K, 93 fois pour Delicious-200K et 45 fois pour Amazon-13K. Avec un seuil appliqué sur ses paramètres après l'entraînement, la taille finale du modèle PDSparse est très faible, mais PDSparse nécessite une grande quantité de mémoire pendant l'entraînement ; par exemple : il ne peut pas être

entraîné sur l'ensemble de données Amazon-670K avec 100 Go de mémoire [2]. La comparaison avec AnnexML est plus sensible car son temps d'apprentissage et la taille de son modèle ne sont pas publiés pour une implémentation sur un coeur. Les performances publiées montrent que CRAFTML est proche de AnnexML, sauf pour Amazon-670K et WikiLSHTC-325K. Mais pour ce dernier, le temps d'apprentissage d'AnnexML (4 heures) uniquement mentionné pour une implémentation à 24 coeurs suggère que CRAFTML est plus rapide (1.5 heure sur une machine à 5 coeurs).

4.3 Comparaison avec les méthodes parallèles (DISMEC, PPDSparse)

Les résultats des modèles linéaires DISMEC et PPDSparse reportés dans les tableaux 5.2 et 5.3 ont été obtenus sur une machine à cent coeurs. Rappelons que le modèle DISMEC a été spécialement conçu pour la parallélisation et qu'il est inapplicable sur une machine monocoeur. Les résultats de CRAFTML CRAFTML ont eux été obtenus sur une machine monocoeur. Les conclusions de la comparaison sont mixtes et dépendent du jeu de données. Les jeux de données WikiLSHTC-325K et Amazon-670K semblent favoriser les deux approches basées sur un modèle linéaire par rapport à CRAFTML mais aussi toutes les autres méthodes arborescentes.

Pour la plupart des jeux de données, la taille du modèle CRAFTML est inférieure à celle de DISMEC et PPDSparse. Le temps de prédiction de CRAFTML obtenu sur une machine monocoeur est souvent inférieur à celui de DISMEC et de PPDSparse obtenu sur machine à cent coeurs. Son temps d'apprentissage est également inférieur à celui de DISMEC pour les grands ensembles de données et similaire pour les plus petits mais supérieur à celui de PPDSparse. En outre, nous avons mesuré les gains de temps de CRAFTML avec une machine à cinq coeurs. Dans ce cas, le temps d'apprentissage de CRAFTML est inférieur à celui de DISMEC pour tous les jeux de données et à celui de PPDSparse pour Delicious-200K. Et, il se rapproche du temps d'apprentissage de PPDSparse pour Amazon-670K. Par conséquent, avec seulement cinq coeurs, CRAFTML est compétitif avec les meilleures approches parallélisées. Plus important encore, son facteur d'accélération d'environ quatre entre une implémentation monocoeur et cinq coeurs et sa faible complexité de temps d'apprentissage/prédiction nous permet d'envisager être en moyenne plus rapide que PPDSparse sur un supercalculateur comparable.

5 Conclusion

Notre nouvelle méthode d'apprentissage multi-label extreme CRAFTML est compétitive avec les autres méthodes arborescentes avec une implémentation sur un seul coeur et elle est compétitive avec PPDSparse, même avec une implémentation parallèle restreinte.

Contrairement à la plupart des méthodes XML actuelles, CRAFTML ne s'appuie pas sur un schéma d'optimisation complexe. Il combine des blocs d'apprentissage simples et rapides (par exemple un clustering avec k-means, un classifieur multi-classe très naïf) ce qui permet d'envisager des extensions pour atteindre les performances requises par les défis sociétaux et techniques actuels [233]. Avec la dimension croissante des données, l'apprentissage automatique recourt de plus en plus aux supercalculateurs. Mais cet accès est loin d'être disponible partout aujourd'hui et son coût va fixer des limites à l'avenir. Par conséquent, (i) des algorithmes d'apprentissage machine économes en ressources et évolutifs sont nécessaires pour favoriser la démocratisation des nombreuses applications du monde réel qui dépendent encore du calcul standard. En contraste, le cloud computing [234] et le développement croissant des supercalculateurs [235] nécessitent également (ii) des méthodes qui exploitent pleinement les ressources de calcul disponibles en étant, en particulier, facilement parallélisables. CRAFTML s'inscrit dans les deux cadres (i) et (ii).

Chapitre 6

Applications

Sommaire

1 Introduction

L'apprentissage multi-label a de nombreuses applications dans des domaines variés comme la vision par ordinateur [1][21], la compréhension de textes [22][23] ou la santé [24][25]. Dans cette thèse, menée dans le cadre d'un contrat CIFRE en partenariat avec le groupe Orange, nous nous sommes intéressés en particulier à l'analyse d'opinions pour le marketing. Nous avons également récemment mené des analyses exploratoires sur des données textuelles.

Dans la première partie du chapitre, nous présentons un outil interactif d'apprentissage multi-label, appelé VIPE (« Visual Interactive and Personalized Exploration of data »), et utilisé au sein du groupe Orange pour l'analyse d'opinions. Basé dans sa première version sur un algorithme de factorisation rapide de matrice, il permet à un utilisateur d'importer des textes courts (tweets, mails, enquêtes, ...), de définir des labels d'intérêts (« client globalement satisfait », « évoque la rapidité du débit »,...) et de proposer pour chaque texte des recommandations de labels et pour chaque label des recommandations de textes. Les dernières contributions algorithmiques de cette thèse pour permettre l'apprentissage multi-label extrême n'ont pas encore été intégrées mais leur exploitation est prévue dans un avenir proche pour rendre l'outil plus performant.

Nous présentons ensuite deux études exploratoires réalisées avec la proposition récente

CRAFTML. La première consiste à évaluer sa capacité à traiter des problèmes de compréhension de textes, reformulés comme des problèmes d'apprentissage multi-label extrême. La deuxième consiste en l'extension de CRAFTML vers le paradigme multi-classe.

2 VIPE : un outil interactif pour l'apprentissage multi-label sur des messages courts

L'analyse d'opinions est un enjeu majeur pour les entreprises qui visent à améliorer en permanence leur relation client. Aux enquêtes par sondage s'ajoutent pour l'analyse les informations extraites sur les médias sociaux. Ces informations contribuent à déterminer le degré d'engouement suscité par les offres d'entreprises, à identifier les différents points de vue et les points de convergence entre les clients, et à recueillir de l'information « fraîche » ([236]). Cependant, l'acquisition des informations utiles est une tâche difficile car les sources complémentaires dont elles sont extraites sont hétérogènes et contiennent des données volumineuses, bruitées et non structurées. Les problèmes associés à l'analyse de ces données rendent le traitement automatique délicat en pratique et l'implication de l'utilisateur est cruciale ([237]).

L'intégration de l'humain dans la boucle d'apprentissage connaît en effet un essor croissant et des systèmes d'apprentissage interactifs ont été développés pour des applications variées : e.g. classification d'images (cueFlick), sélection de fichiers (Smart Selection), classification de gestes (Wekinator), classification de documents (iCluster), tri d'alarmes (CueT). Dans ce cadre, l'utilisateur annote, via une interface adaptée, un nombre limité d'exemples et, à partir de ces quelques exemples, un algorithme d'apprentissage tente de capturer l'expertise pour apprendre un premier modèle prédictif. En fonction de sa satisfaction, l'utilisateur peut arrêter l'apprentissage ou continuer à entraîner le modèle. Les retours expérimentaux menés sur des petits échantillons d'utilisateurs semblent très prometteurs. Cependant, la plupart des systèmes existants se limitent à une classification monolabel où un seul label peut être affecté à la fois à un exemple ; ce qui est peu expressif d'autant moins que les données sont très souvent de nature multi-label. Dans le cadre de l'analyse d'opinions, il s'agit effectivement de dépasser le cadre positif/négatif pour prendre en compte des comportements plus subtils. Mais comme l'ont montré récemment [39], peu d'approches multi-label résistent aux contraintes d'interactivité.

Pour palier cette limitation, nous avons contribué au développement d'un nouvel outil VIPE permettant un apprentissage interactif multi-label de textes courts provenant de transcriptions de résultats d'enquêtes d'opinions sur le Web ou de centres d'appels, de forums spécialisés ou de Twitter. L'utilisateur définit initialement un ensemble de labels (par exemple : Efficacité, Innovation, Couverture réseau, Négatif, Positif) puis il procède

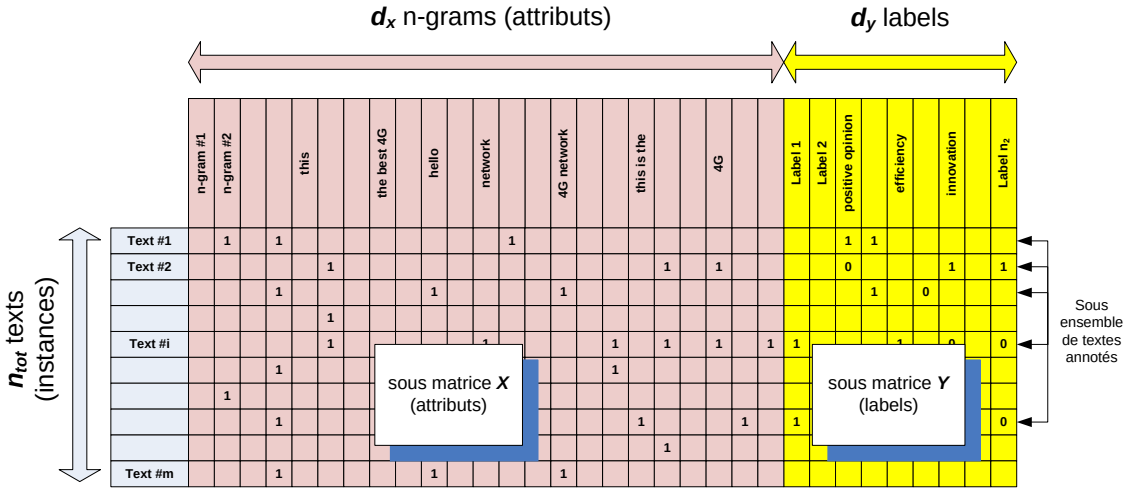

Figure 6.1 – Matrice des données dans VIPE.

à la classification manuelle d'un ensemble restreint de textes. Par exemple, il annote le tweet "ça c'est de la #4G !" en positif avec les deux labels : Efficacité et Innovation. En se basant sur l'ensemble de textes étiquetés, un algorithme d'apprentissage assiste ensuite l'agent en lui prédisant les labels les plus probables pour un ensemble de textes ou les textes les plus probables pour un label ou une combinaison sélectionnés.

2.1 Apprentissage multi-label avec VIPE

2.1.1 Modélisation du problème

On considère un ensemble de n_{tot} textes, chacun automatiquement encodé comme un sac de n-grams. Plus précisement, nous retenons tous les 1-grams, 2-grams et 3-grams occurrents au moins deux fois dans le corpus (e.g comme 3-grams, nous avons "this is the", "is the best", ... "best 4G network"). L'ensemble $\mathcal{F}$ de taille d_x contient tout le dictionnaire de n-grams et joue le rôle de l'ensemble d'attributs du problème.

Au début du processus d'apprentissage, l'utilisateur définit un ensemble $\mathcal{L}$ de d_y labels qui l'intéressent et annote un petit ensemble $\mathcal{T}$ de n textes avec les labels pertinents. Le reste du corpus constitue l'ensemble $\mathcal{S}$ de $n_{\mathcal{S}}$ textes non annotés. Dans VIPE, ce deuxième ensemble est généralement beaucoup plus grand que $\mathcal{T}$.

On appelle M la matrice de taille $n_{tot} \times d$ de $n_{tot} = n + n_{\mathcal{S}}$ textes décrit par $d = d_x + d_y$ variables. La matrice M peut être décomposée en deux sous matrices X et Y (figure 6.1). La matrice X encode la présence/absence des n-grams pour chaque texte : $X_{ij} = 1$ si le $i^{\text{ème}}$ texte contient le $j^{\text{ème}}$ n-gram de $\mathcal{F}$ et X_{ij} est vide sinon. Cette configuration optimise la gestion de la mémoire. La matrice Y encode les labels des textes : $Y_{ij} = 1$ (resp. 0) si le $i^{\text{ème}}$ texte est étiqueté positivement (resp. négativement) avec le $j^{\text{ème}}$ label.

Si l'information est manquante, la composante de la matrice est vide. La matrice Y est vide pour les $n_{\mathcal{S}}$ lignes correspondant aux textes non annotés et le but de VIPE est de prédire ces composantes inconnues.

2.1.2 Algorithme actuel

L'algorithme actuel est basé sur l'algorithme de factorisation de matrice Molécule qui est une adaptation de l'algorithme "Gravity" [238], gagnant du concours Netflix démarré en 2006 [239], pour les données "positive-only" (matrice creuse dont les composantes non nulles ne sont que des 1). Le principe est d'approximer une matrice creuse M de taille $n_{tot} \times d$ par une matrice dense de rang faible $\widehat{M} = P^T Q$ où $P \in \mathbb{R}^{k \times n_{tot}}$ et $Q \in \mathbb{R}^{k \times d}$ sont respectivement des représentations latentes des lignes et colonnes de la matrice M. La matrice de rang faible (au plus k) obtenue est utilisée comme approximation des valeurs manquantes de M.

Les matrices P et Q sont apprises en minimisant l'erreur quadratique moyenne (RMSE) qui est la moyenne des e_{ij}^2 (erreur entre l'approximation de la matrice de rang faible $\widehat{M}_{ij} = \sum_{w=1}^{k} P_{wi} Q_{wj}$ et la vérité terrain M_{ij}) pour chaque cellule d'indice (i, j) non vide de M. La RMSE est une fonction d'erreur quadratique facile à dériver qui peut être optimisée efficacement avec un algorithme de descente de gradient.

Les matrices P et Q sont initialisées aléatoirement (distribution gaussienne et normalisation). Ensuite, l'algorithme de factorisation considère dans un ordre aléatoire toutes les cellules non nulles de la matrice M : pour chaque cellule M_{ij}, il calcule le gradient de l'erreur quadratique e_{ij}^2 et l'utilise pour mettre à jour les facteurs $P_{.i}$ et $Q_{.j}$. Le gradient de e_{ij}^2 selon les paramètres impliqué est calculé comme suit :

$$\begin{aligned} &\forall w \in \{1, ..., k\}, \\ &\frac{\partial e_{ij}^2}{\partial P_{wi}} = -2 \times e_{ij} \times Q_{wj} \\ &\frac{\partial e_{ij}^2}{\partial Q_{wj}} = -2 \times e_{ij} \times P_{wi} \end{aligned} \tag{6.1}$$

Pour réduire l'erreur de prédiction du modèle et approximer la solution de $M = P^T Q$, les facteurs P_{wi} et Q_{wj} sont mis à jour dans le sens inverse du gradient avec un taux d'apprentissage α (généralement de l'ordre de 10^{-2}). Pour éviter le sur-apprentissage, on ajoute un terme de régularisation L_2 dans l'apprentissage. De plus, pour éviter la divergence à l'infini, les facteurs sont contraints de rester dans l'intervalle $[-1, 1]$. Enfin, nous utilisons également une procédure de "early stopping" avec patience pour éviter le sur-apprentissage. A la fin de chaque itération d'apprentissage, la RMSE est évaluée sur un petit ensemble de validation : si elle ne diminue pas pendant un nombre fixé d'itérations, le processus d'apprentissage s'arrête et les dernières matrices optimales P et Q sont retenues.

Puisque la factorisation de la matrice ne prend en compte que des valeurs non vides, M risque d'être confondue avec une matrice remplie de 1. En effet, toutes les cellules non vides contiennent la valeur 1 et cela peut conduire à une approximation dépourvue de sens. Pour éviter cette situation, l'algorithme doit être informé que certaines des cellules vides portent une valeur nulle. Pour cela, lors de la phase d'apprentissage, chaque fois qu'un gradient est calculé et retropropagé par rapport à une cellule non vide de M, un gradient est également calculé et retropropagé par rapport à une cellule vide de M sélectionnée de manière aléatoire ("negative sampling"). D'autres stratégies ont été testées (par exemple de normalisation) mais celle de "negative sampling" s'est avérée être la plus intéressante.

La complexité de l'algorithme est linéaire par rapport au nombre d'éléments non nuls de la matrice M et du nombre de passes. Expérimentalement, le nombre de passes est de l'ordre de la dizaine.

2.2 Architecture de VIPE

VIPE est une application Web composée de 4 modules. Le **module 1** est un gestionnaire d'import des textes courts par chargement de fichiers textes et d'export des résultats de l'apprentissage. Les textes sources sont partagés entre les utilisateurs mais les labels sont du domaine privé de chacun. VIPE apprend un modèle unique sur les différentes sources et labels, mais chaque utilisateur peut uniquement annoter ou consulter les résultats pour ses propres labels. Les résultats peuvent être de trois types : *(i)* les textes correspondants à des labels donnés - sélectionnés par un ranking sur la matrice des facteurs calculée par l'algorithme de factorisation - ; *(ii)* les labels prédits pour un texte donné ; *(iii)* la matrice des scores des labels prédite pour l'ensemble des textes.

Le **module 2** est un gestionnaire de la matrice M de données qui contient en ligne les textes et en colonne deux informations : *(i)* une description des textes en sacs de n-grams (matrice X) et *(ii)* une description binaire de l'affectation des textes aux labels (matrice Y).

Le **module 3** est composé d'un serveur et d'une interface Web qui permet de manipuler les textes et d'effectuer l'étiquetage interactif (figure 6.2). VIPE intègre trois actions pour l'apprentissage interactif : *(i)* par correction manuelle des résultats proposés ; *(ii)* par sélection d'exemples positifs et négatifs à partir de la base de textes. Cela est fait manuellement par une requête à base de mots-clés ou en parcourant la liste de textes ; *(iii)* par ajout d'exemples virtuels grâce à une boîte de dialogue qui permet à l'utilisateur de construire un exemple et de l'étiqueter.

Le **module 4** contient l'algorithme d'apprentissage qui est anytime et qui est appliqué selon une boucle sans fin sur les données d'apprentissage (une passe dure quelques secondes pour 1 million de cellules traitées dans la matrice).

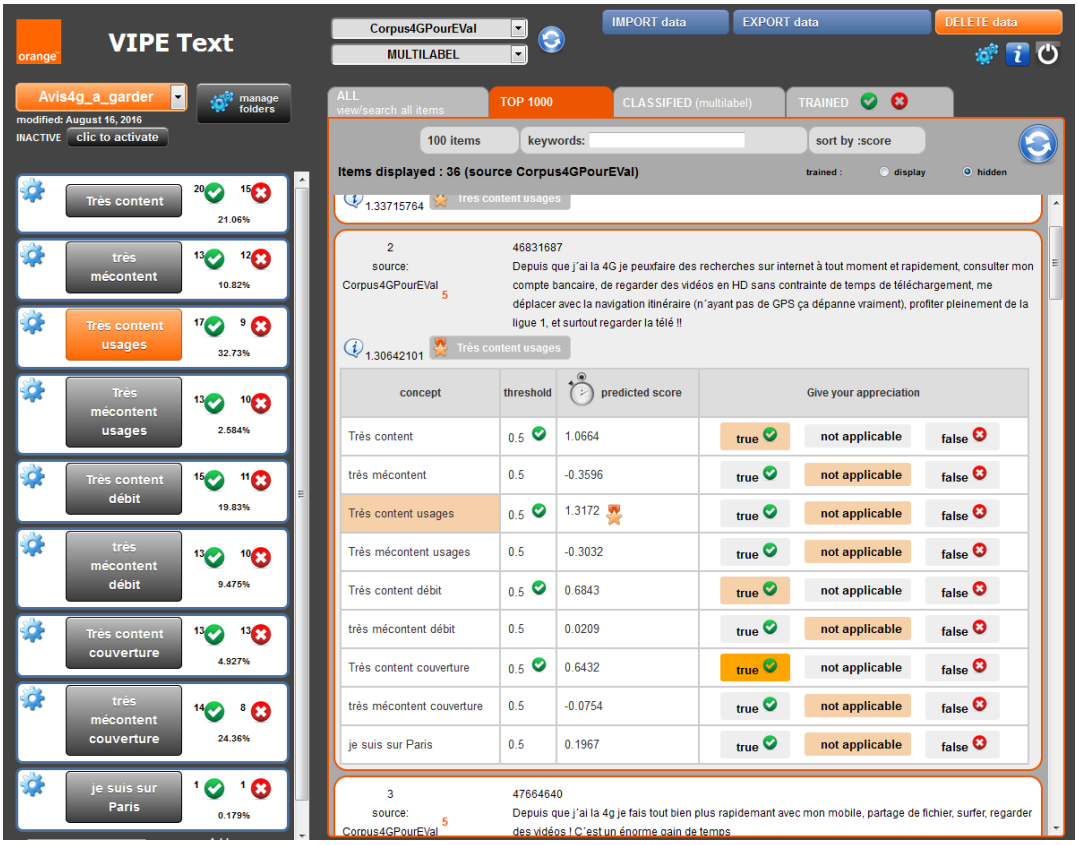

Figure 6.2 – L'interface de VIPE pour l'étiquetage des textes

2.3 Usages et performances

Actuellement, VIPE est utilisé par une vingtaine d'utilisateurs au sein d'Orange, principalement dans les services marketing pour analyser les opinions sur les différents produits et services de l'entreprise. En moyenne, la phase interactive dure moins de 2 heures ; une fois la confiance établie dans l'outil, l'utilisateur exporte les résultats. VIPE produit une table avec les scores prédits. Ces derniers peuvent être utilisés pour de l'apprentissage mais aussi pour du ranking selon les objectifs de l'analyse. En 2018, la volumétrie des données stockées pour l'analyse d'opinions était de : 1M+ textes dans la base (n_{tot}), 1.5M+ n-grammes (d_x), ∼200 labels (d_y), ∼ 20M (resp. ∼ 5k) cellules non vides dans la matrice X (resp. Y).

L'algorithme d'apprentissage Molecule a préalablement été testé sur quatre bases de tailles variées - de 6000 à 330000 lignes (n_{tot}) et de 3600 à 480 000 colonnes ($d = d_x + d_y$) - : MovieLens IM, Netflix et deux extraits de catalogue VOD (Video On Demand) IMDB (table films/mot-clés extraite en 2012) et Orange. Les données, codées en « positive only » (binarisées selon la moyenne globale des notes dans les cas de Netflix et MovieLens), ont été partitionnées en 10 ensembles avec un processus classique de validation croisée. Les résultats avec le critère BER multi-label qui évalue le ratio des labels mal classés sont les suivants : 11.3% (Movie Lens), 12.9% (Netflix), 7.4% (IMDB), 20.2% (Orange).

L'apprentissage résiste bien à la très forte parcimonie des données (au mieux, plus de 96% de valeurs manquantes).

2.4 Vers l'intégration des contributions de la thèse

En conclusion, VIPE est un prototype de système d'aide à l'apprentissage interactif multi-label de textes courts qui permet de gérer une volumétrie importante, avec des bonnes performances prédictives. Des évolutions dans un futur proche prévoient de rendre l'outil multi-utilisateur (plusieurs milliers d'utilisateurs) et plus performant. Dans un tel cadre, en considérant que chacun définisse une centaine de labels et que l'ensemble des données soit traité avec un unique moteur d'apprentissage, le problème (plusieurs dizaines de milliers de labels) devient extrême et rejoint ainsi le cadre de la problématique de cette thèse. De plus, si l'algorithme Molecule actuel est intéressant car il tire profit des exemples non annotés, il a été initialement conçu pour l'apprentissage non supervisé et il n'obtient généralement pas les performances de l'état de l'art en apprentissage multi-label [39].

Ainsi, il est prévu de remplacer le moteur actuel d'apprentissage de VIPE par le nouvel algorithme CRAFTML. Mais, cela nécessitera une évolution de ce dernier dans deux directions : (i) vers une version semi-supervisée capable d'apprendre avec peu d'exemples annotés et beaucoup d'exemples non annotés, et (ii) vers une version online pour l'interactivité. Ces perspectives sont détaillées dans le dernier chapitre.

3 Tests applicatifs sur CRAFTML

Outre l'application logiciel VIPE, différentes études exploratoires sollicitent actuellement l'utilisation de l'algorithme CRAFTML. C'est pourquoi le prototype expérimental a conduit au développement d'une API java (voir Figure 6.3). Elle permet d'appeler les fonctionnalités facilement avec un fichier de configuration indiquant l'action ou les actions à réaliser parmi les possibilités suivantes :

- Apprendre : A partir de données multi-label annotées fournies par l'utilisateur et des hyperparamètres choisis (ou par défaut), le modèle CRAFTML est appris et stocké sous forme de fichiers sur le disque.
- Prédire : A partir de données non annotées et d'un modèle appris, CRAFTML fournit des prédictions.
- Mesurer les performances prédictives : A partir de données multi-label annotées de test et d'un modèle appris, CRAFTML fournit des prédictions et mesure ses performances par rapport à la vérité terrain.

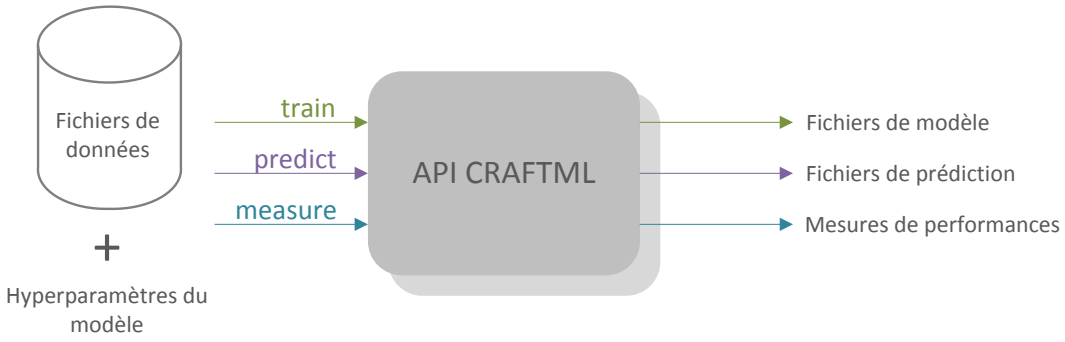

Figure 6.3 – Principe de l'API de CraftML

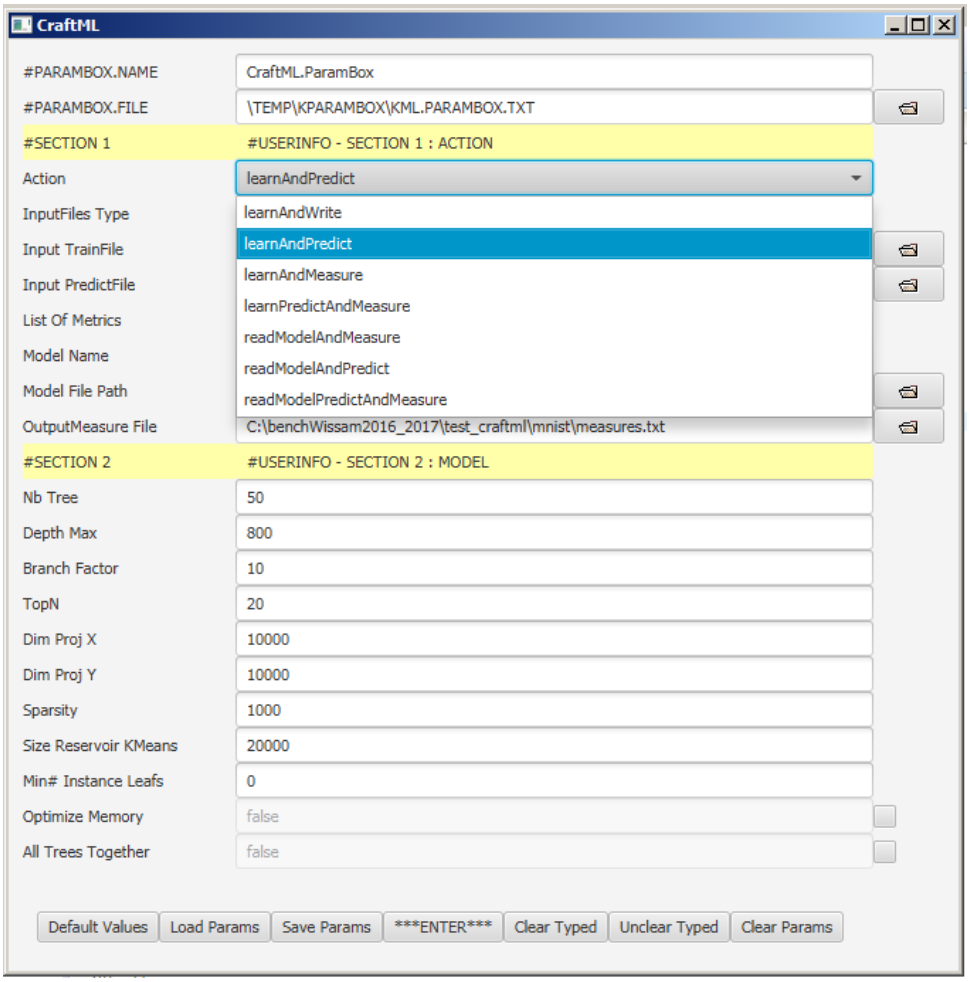

Figure 6.4 – Version actuelle de l'interface graphique pour la génération de fichier de configuration pour l'API.

Les actions sont réalisées à partir de fichiers de données (apprentissage, test) supportant plusieurs formats (libsvm, tabulaire, de logs, en texte brut) et elles produisent

des fichiers contenant le modèle, les prédictions ou bien un récapitulatif des mesures. Une interface utilisateur facilitant l'accès à l'API est en cours de développement. Il s'agit notamment de proposer un formulaire dans une fenêtre permettant la génération d'un fichier de configuration interprétable par l'API (Figure 6.4).

De plus, deux types de tests exploratoires sont en cours de réalisation :

— L'algorithme étant particulièrement adapté au traitement de textes bruts, des tests sont réalisés sur des thématiques textuelles (traduction automatique, clustering de textes, complétion de séquences).

— Si l'algorithme actuel a été initialement conçu pour l'apprentissage multi-label, il n'a pas de raison structurelle de s'y limiter. Un autre test exploratoire consiste à vérifier la capacité de l'algorithme à s'adapter à des problèmes multi-classe.

Les premiers résultats obtenus sont présentés dans les deux sous-sections suivantes.

3.1 Applications textuelles

CRAFTML semble prometteur pour les applications textuelles car il ne nécessite pas de connaitre initialement la liste des attributs (dictionnaire de mots/n-grams) ou des labels pour apprendre et il peut donc être entraîné directement sur du texte brut. Cette caractéristique est due aux projections aléatoires des attributs et des labels pendant l'apprentissage. Pour projeter un texte brut sur un vecteur d'attribut réduit, la procédure consiste à boucler sur les n-grams qu'il contient, à les hacher pour obtenir un indice du vecteur réduit, et à mettre à jour la composante associée. Il n'est donc pas nécessaire d'indexer le vocabulaire à l'avance car la fonction de hachage, appliquée directement sur un n-gram, produit un index sur l'espace réduit.

Pour commencer à tester le caractère prometteur de ce domaine applicatif, nous avons effectué des premiers tests exploratoires. A ce stade nous sommes loin d'avoir exploré l'abondante littérature associée et nous n'avons pas encore pour objectif de comparer nos résultats aux méthodes sophistiquées de l'état de l'art. Nous avons juste obtenu un aperçu des résultats préliminaires de notre algorithme dans trois situations : traduction automatique, complétion de séquences, clustering de textes.

Traduction automatique CRAFTML pouvant associer un très grand ensemble de variables (e.g. un dictionnaire de mots/n-grams dans une langue) à un autre grand ensemble de variables (e.g. dictionnaire de mots/n-grams dans une autre langue), la traduction automatique apparaît comme un domaine d'application à évaluer. Les premiers résultats obtenus portent sur le cadre très simplifié de la traduction sac-de-mots à sac-de-mots de l'anglais au français.

Pour cela nous avons appris un modèle CRAFTML sur 100000 phrases réalignées anglais-français de la base Europarl de retranscriptions de réunions du parlement euro-

péen. Les phrases en anglais (resp. français) sont transformées en sacs-de-mots qui sont les attributs (resp. labels) du problème. Sur chaque label i, la pondération $-ln(f_i)$, où f_i est la fréquence du label, est appliquée pour éviter de donner un poids trop important aux mots vides dans les feuilles. Quelques exemples de prédictions sont donnés dans le tableau 8.15 de l'annexe 3. Sur des phrases avec un vocabulaire adapté au contexte du parlement européen, les résultats sont encourageants : plusieurs mots des phrases en anglais sont bien disponibles en français dans les prédictions. Mais une partie est parfois manquante dans les prédictions actuelles et nous avons constaté que les résultats sont très sensibles à l'encodage, notamment aux pondérations qui favorisent soit les mots vides et fréquents, soit les mots rares.

Complétion de séquences Nous avons exploré la complétion de séquence dans le cadre textuel qui consiste à prédire le prochain mot (resp. lettre) à partir de tous les mots (resp. lettres) précédents dans la phrase. Les modèles classiques, tels que le réseau de neurones word2vec, apprennent avec une fenêtre glissante qui consiste à prédire un élément à partir des l éléments précédents où l est le paramètre de largeur de la fenêtre. Nous avons testé un encodage qui ne se limite pas à une fenêtre de mots afin de conserver une mémoire de l'ensemble des éléments précédents tout en donnant plus d'importance aux éléments proches. Une phrase de taille n_m éléments (mots ou caractères) est donc encodée comme $n_m - 1$ instances telles que la i-ème instance a pour sortie le $i+1$-ème élément et pour entrée les i éléments précédents pondérés selon leur distance au $i+1$-ème élément (le poids du j-ème élément, où $j \in \{1, .., i\}$, est 0.9^{i+1-j}). Un test a été effectué avec ce codage sur la base « avis 4g » et les éléments pour l'encodage sont les caractères. Des exemples de prédictions sont montrés en Annexe 3 dans le tableau 8.16. Les résultats sont encourageants mais présentent quelques limites qui resteront à lever dans la suite. En particulier, si l'enchaînement des lettres et des mots est souvent réaliste, les fins de séquences comportent des erreurs.

Clustering de textes Pour tenter d'étendre CRAFTML à des applications non supervisées, nous avons considéré l'astuce suivante : CRAFTML est entraîné sur des données non annotées en considérant que les labels sont ici les attributs. Il devient donc simplement un algorithme de clustering hiérarchique très rapide. Nous l'avons ainsi testé sur une base de 40 000 mots encodés par Glove [3]. Les clusters obtenus (voir exemples dabs le Tableau 8.17 de l'Annexe 3) semblent qualitativement intéressants car ils regroupent les mots selon des thèmes identifiables (bijouterie, art, mode, apéritif, etc..). Ils témoignent à la fois de la bonne représentation des mots fournie par le modèle Glove mais aussi de la qualité des regroupements réalisés par CRAFTML en mode non supervisé.

Tableau 6.1 – Comparaison des performances de CRAFTML et des forêts aléatoires classiques (RF) sur 13 jeux de données de l'UCI

	CRAFTML	RF
audiology	0.8131 ± 0.0809	0.8127 ± 0.0780
iris	0.9733 ± 0.0249	0.9400 ± 0.0327
balance-scale	0.8593 ± 0.0128	0.8320 ± 0.0195
ecoli	0.8782 ± 0.0406	0.8873 ± 0.0472
zoo	0.9905 ± 0.0190	0.9905 ± 0.0190
breast-cancer	0.7376 ± 0.0259	0.7202 ± 0.0126
car	0.9612 ± 0.0133	0.9606 ± 0.0190
Congressional Voting Records	0.9586 ± 0.0277	0.9610 ± 0.0256
letter	0.9075 ± 0.0051	0.8988 ± 0.0045
pendigits	0.9944 ± 0.0022	0.9913 ± 0.0016
optdigits	0.9859 ± 0.0059	0.9808 ± 0.0031
primary-tumor	0.4404 ± 0.0493	0.4371 ± 0.0453
soybean	0.9383 ± 0.0223	0.9415 ± 0.0075

3.2 Extension vers d'autres paradigmes

Différentes problématiques applicatives, telles que par exemple la complétion de séquences ci-dessus, se modélisent comme des problèmes d'apprentissage multi-classe de petite dimension. Nous avons donc commencé à évaluer la capacité de CRAFTML à s'adapter à ce cadre pour des jeux de données de taille restreinte avec un petit nombre de classes, de l'ordre de la dizaine au maximum. Une première comparaison partielle avec le modèle de forêt aléatoire multi-classe standard a été effectuée récemment sur un banc d'expériences de l'UCI. A ce stade, nous disposons de résultats sur 13 jeux de données : iris, zoo, audiology, ecoli, balance-scale, breast-cancer, car, congressional voting records, letter, pendigits, optdigits, primary-tumor et soybean. Le tableau 6.1 présente les résultats obtenus pour la mesure d'*accuracy* avec une évaluation 5-fold stratifiée et un partitionnement apprentissage/test identique pour CRAFTML et pour les forêts aléatoires. Pour les deux algorithmes, nous avons choisi l'hyperparamétrage par défaut et nous avons fixé le nombre d'arbres à 100. L'implémentation des forêts aléatoires utilisée est celle de la librairie "scikit-learn".

Sur ces premiers tests, les performances des deux algorithmes sont très similaires sur une majorité des jeux. Ce résultat montre que le comportement asymptotique de CRAFTML dans une configuration très différente de celle dans laquelle il a été initialement développé est proche de celui d'une forêt aléatoire classique qui est considérée comme une approche très compétitive en classification multi-classe.

3.3 Conclusion

Les préoccupations applicatives de cette thèse conduisent à deux directions pour les développements futurs. Dans un premier temps, le nouvel algorithme proposé CRAFTML doit être intégré dans le logiciel VIPE développé par Orange et testé pour d'autres applications telles que la catégorisation de demandes clients et la catégorisation de brevets. De plus, l'API que nous avons développée permettra d'explorer le comportement de CRAFTML pour d'autres paradigmes que celui pour lequel il a été initialement développé. Les premières expérimentations menées sur des données textuelles et dans le cadre multi-classe semblent prometteuses. Elles nécessiteront bien évidemment à l'avenir des approfondissements pour identifier le rôle spécifique de chacun des composants de CRAFTML pour ces nouveaux problèmes et pour améliorer les résultats.

Chapitre 7

Conclusion et perspectives

Sommaire

1 Conclusions

Dans cette thèse, nous nous sommes intéressés au récent problème d'apprentissage multi-label extrême pour lequel nous avons apporté des contributions *méthodologiques*, *algorithmiques* et *applicatives*. Une analyse macroscopique des méthodes existantes complétée par un focus sur les stratégies adaptées à l'apprentissage extrême nous a conduits à développer de nouvelles méthodes permettant un passage à l'échelle tout en restant économes en ressources.

D'un point de vue *méthodologique*, un état de l'art approfondi des méthodes de réduction de dimension appliquées à l'apprentissage multi-label nous a permis de regrouper plus d'une cinquantaine de méthodes publiées sous une même typologie structurée autour de cinq critères discriminants (le ou les espaces réduits par la méthode, la dépendance entre attributs et labels dans la réduction, le couplage entre la réduction et le classifieur, le type de transformation appliquée et les contraintes imposées sur le problème de réduction) et de décrire la grande majorité des problèmes posés dans la littérature sous deux formalismes unificateurs. Cette unification permet de comparer plus finement les différentes approches, d'identifier les ingrédients encore peu exploités dans les recherches, et de guider la sélection d'une méthode mieux appropriée pour un problème spécifié. En complément, une méta-analyse basée sur l'agrégation de relations de domination statistiquement significatives dans l'ensemble des résultats expérimentaux disponibles dans la

littérature nous a permis d'identifier les approches les plus performantes. Il ressort notamment que les meilleures approches eu égard aux mesures d'évaluation classiques de la qualité de la prédiction utilisent une stratégie qui couple l'objectif de la réduction de dimension et celui de la classification. Selon cette caractéristique les réseaux de neurones apparaissent comme des candidats prometteurs mais des expérimentations préliminaires nous ont montré deux limites face aux données extrêmes : (i) la nécessité d'éviter les approximations de rang faible en apprenant par exemple plusieurs modèles locaux et/ou en transformant préalablement l'espace des labels, et (ii) des moyens de calculs conséquents qui n'étaient pas dans notre cahier des charges qui visait à développer des approches applicables sur des environnements standards. Nous aurions pu approfondir les premières expérimentations avec les réseaux de neurones mais nous avons choisi une autre voie qui nous a conduits à développer un algorithme multi-label avec de bonnes performances prédictives et à faible complexité en temps et en mémoire.

Les contributions *algorithmiques* portent sur les trois axes majeurs de la littérature XML : réduction de dimension, astuces d'implémentation/d'optimisation, et partitionnement hiérarchique du problème. En **réduction de dimension**, nous avons proposé une approche, appelée ML-ARP, pour explorer un couplage total entre réduction de dimension et classification : ML-ARP réduit linéairement les attributs pour spécifiquement optimiser les performances de classification de ML-kNN, un des meilleurs algorithmes multi-label standard. Ce couplage permet d'obtenir de bonnes performances, plus stables que l'état de l'art sur un ensemble de jeux classiques de données multi-label. En revanche, en faisant une approximation de rang faible et en ayant une complexité trop élevée, ML-ARP se heurte au passage à l'échelle. Pour **optimiser l'implémentation**, nous avons proposé une stratégie qui permet d'avoir de bonnes performances et une faible complexité spatiale. Il s'agit d'une stratégie de stockage économe des paramètres d'un modèle entraîné avec une méthode d'optimisation itérative comme la descente de gradient. Inspirée de la méthode du « count-sketch », elle permet de réduire considérablement la taille mémoire du modèle et les résultats théoriques suggèrent une bonne approximation des plus grands paramètres. Expérimentalement nous l'avons testée sur le modèle de régression one-vs-rest qui est une des approches ayant les performances prédictives les plus élevées en XML mais dont la complexité mémoire est trop grande pour les calculateurs classiques. Notre approche réduit significativement la consommation mémoire (par exemple, en passant de 12Go à seulement 12Mo sur Wiki10-31K) et préserve les performances prédictives. Elle ne permet cependant pas de résoudre le challenge temporel. Pour cela, nous avons proposé une méthode arborescente appelée CRAFTML (Clustering-based RAndom Forest of predictive Trees for extreme Multi-label Learning) qui, en **partitionnant hiérarchiquement** le problème XML, permet de répondre raisonnablement aux trois attentes de l'apprentissage extrême. En tirant parti du caractère creux des données et de la réduction

de la dimension basée sur des projections aléatoires, et en implémentant une stratégie de partitionnement très rapide, CRAFTML permet de passer à l'échelle sur des problèmes de très grandes dimensions en préservant les ressources mémoires et temporelles.

Nos contributions *applicatives*, stimulées par le contexte CIFRE de cette thèse au sein du groupe Orange, concernent le développement d'outils logiciels complété par quelques études exploratoires. Plus précisément, nous avons contribué au développement du logiciel de classification multi-label de textes VIPE qui est utilisé pour l'analyse d'opinion et d'une API de CRAFTML pour permettre son intégration dans diverses analyses de compréhension de textes.

L'ensemble de nos contributions qui s'inscrit dans la dynamique actuelle de l'apprentissage multi-label laisse entrevoir des pistes d'amélioration et ouvre la voie à une extension de notre approche CRAFTML vers des problèmes plus contraints.

2 Perspectives

Nos travaux ouvrent des perspectives dans plusieurs directions : méthodologique, algorithmique et applicative.

La *direction méthodologique* est motivée par une partie conséquente de la thèse qui a porté sur l'analyse approfondie des méthodes de réduction de dimension. Notre démarche a consisté à établir un cadre formel fédérateur puis à mener une méta-analyse sur les résultats publiés dans la littérature. Cette démarche nous permet d'identifier les pistes d'amélioration pour l'avenir. D'une part, la combinatoire des composantes clés des formulations génériques (section 2 du chapitre 3) pourrait être exploitée pour de futures propositions. De plus, la méta-analyse ouvre la discussion vers la construction d'un protocole expérimental partagé qui devrait permettre de mieux évaluer les performances avec un biais limité. Enfin, cette revue de la littérature nous a permis de constater que la robustesse à l'échantillonnage, au bruit, aux transformations géométriques et au type de données sont des préoccupations qui ne sont que partiellement abordées en réduction de dimension multi-label.

La *direction algorithmique* concerne l'amélioration et l'extension de CRAFTML. Nous avons identifié trois perspectives dans cette direction. Tout d'abord, soulignons que tout récemment de nouveaux algorithmes d'apprentissage multi-label extrême comme Parabel [240] ou ProXML [241] ont à nouveau élevé les performances de l'état de l'art. Le challenge qui vise à améliorer les performances prédictives avec une complexité restreinte reste donc d'actualité. Deux voies sont envisageables pour tenter d'améliorer CRAFTML : (i) la combinaison avec les autres contributions algorithmiques de cette thèse et (ii) l'extension des blocs actuels de l'algorithme vers des stratégies plus sophistiquées.

Pour le point (i), l'astuce de stockage des paramètres développée dans le chapitre 4 pourrait être combinée à la forêt CRAFTML pour améliorer sa consommation mémoire. On peut également envisager d'exploiter des techniques de réduction de dimension analysées dans le chapitre 3 dans la construction du séparateur à chaque noeud pour en améliorer la qualité. Pour le point (ii), il est nécessaire d'évaluer plus finement les contributions respectives de chacun des blocs de CRAFTML afin de proposer des variantes pertinentes. Les directions majeures sont l'implémentation de modèles locaux plus expressifs dans les feuilles qui tiendraient compte des attributs pour prédire les labels (e.g. classifieur linéaire ou de type kNN) [38], l'introduction de pondérations entre les arbres de la forêt voire entre les différents chemins racine/feuille dans un même arbre, et le remplacement de l'algorithme de clustering dans la construction du séparateur.

La *direction applicative* concerne entre autres l'intégration de CRAFTML dans l'outil logiciel VIPE développé chez Orange. En sus des extensions présentées dans le chapitre précédent, deux pistes sont actuellement à l'étude : la semi-supervision et une variante on-line adaptée à l'interactivité.

La semi-supervision est très importante dans VIPE car elle permet non seulement d'apprendre à partir des exemples non annotés mais aussi à partir d'exemples annotés. Les premiers sont nombreux et les seconds sont peu abondants car les utilisateurs chargent généralement de grandes bases de textes mais n'en annotent que quelques uns. Le deuxième intérêt de la semi-supervision dans VIPE est lié au caractère textuel des données traitées. En effet, l'apprentissage sur les informations contenues dans les attributs d'un corpus non annoté, comme la co-occurrence des mots par exemple, permet de produire une meilleure représentation des attributs qui facilite ensuite la généralisation à partir d'un nombre limité d'exemples annotés.

Dans la thèse nous n'avons pas proposé d'algorithme semi-supervisé car les jeux de données du banc d'expériences XML ont généralement un nombre d'instances assez important (de l'ordre du millier d'instances sur les petits jeux de données et de la centaine de milliers d'instances sur les jeux extrêmes). Cependant, si la version actuelle de CRAFTML est uniquement supervisée, des extensions vers une semi-supervision sont envisageables. Par exemple, les arbres pourraient être construits en non supervisé, puis les labels des quelques instances annotées pourraient être propagées dans les feuilles des sous-arbres qui les contiennent. Les séparateurs au niveau des noeuds pourraient également être définis dans un cadre semi-supervisé en remplaçant le clustering supervisé actuel sur les labels (étape 2 de CRAFTML dans la section 2 du chapitre 5) par un clustering semi-supervisé [242]. Pour évaluer, à l'avenir, l'apport de ces stratégies par rapport à la version initiale de CRAFTML, des protocoles expérimentaux sont adaptés [39][243]. Ils consistent à calculer les performances des algorithmes appris sur l'ensemble d'apprentissage après avoir supprimé les labels d'un certain pourcentage (variant de 0% à 100%) des instances.

La perspective portant sur l'interactivité nécessite le développement d'une variante on-line de CRAFTML qui ne soit pas contrainte par les dimensions de la base d'apprentissage et qui permette de traiter des flux d'exemples en oubliant les exemples passés. Actuellement, le comportement de CRAFTML n'est pas très éloigné d'un comportement interactif lorsque le nombre total d'exemples et de variables d'apprentissage est de l'ordre de la dizaine de milliers. Il est en effet capable dans ce cas d'apprendre très rapidement (quelques secondes sur Eurlex-4K) et peut réapprendre complètement un nouveau modèle à la réception d'un nouvel exemple. Mais la construction d'un arbre on-line est un problème difficile pour trois raisons majeures [244] : (i) une variante on-line nécessite plus d'exemples avant sa stabilisation, (ii) elle est plus sensible au bruit, aux données manquantes et au sur-apprentissage, et (iii) l'équilibre entre l'information apportée par les exemples récents et celle apportée par les exemples plus anciens est délicat à trouver. Mais heureusement de nouvelles stratégies prometteuses ont récemment été publiées pour envisager l'implémentation d'une forêt on-line [244][245][246].

3 Liste des travaux

Les travaux contenus dans cette thèse ont donné lieu à des communications dans trois conférences internationales (ICML 2018, CIE47 et IEA/AIE 2017), dans deux conférences nationales (EGC 2017 et AAFD & SFC'16) et dans un atelier (Atelier TextMine EGC 2018). En outre, un article est actuellement en cours de révision pour un journal et un autre est rédigé en vue d'une soumission à une conférence.

Conférences internationales avec actes

— Siblini, W., Meyer, F., Kuntz, P., CRAFTML, an Efficient Clustering-based Random Forest for Extreme Multi-label Learning. **ICML 2018**. Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. [247]

— Siblini, W., Meyer, F., Kuntz, P., Vipe : A New Interactive Classification Framework for Large Sets of Short Texts - Application to Opinion Mining. **CIE47**. The 47th International Conference on Computers & Industrial Engineering, Oct 2017, Lisbon, Portugal. [248]

— Siblini, W., Alami, R., Meyer, F., Kuntz, P., Supervised Feature Space Reduction for Multi-Label Nearest Neighbors. **IEA/AIE 2017**. The 30th International Conference on Industrial, Engineering, Other Applications of Applied Intelligent Systems. [110]

Conférences nationales avec actes

— Meyer, F., Tricot, S., Kuntz, P., Siblini, W., VIPE : un outil interactif de classification multi-label de messages courts. **EGC 2017**. Conférence Extraction et Gestion des Connaissances. (**Prix de la meilleure démonstration logiciel**) [249]

— Siblini, W., Meyer, F., Kuntz, P., Vers un apprentissage multi-label rapide en grande dimension – Une étude préliminaire. **AAFD & SFC'16** : Conférence Internationale Francophone sur la Science des Données. [250]

Divers

El Asry, I., Siblini, W., Meyer, F., Réseau neuronal convolutif sur des séquences de caractères pour la classification de textes. **Atelier TextMine EGC 2018**.

En cours de soumission

Siblini, W., Meyer, F., Kuntz, P., A review on dimensionality reduction for multi-label classification. En cours de révision pour une revue internationale.

Siblini, W., Meyer, F., Kuntz, P., A Count-sketch to Reduce Memory Consumption when Training a Model with a Gradient Descent. Doît être soumis à une conférence internationale.

Bibliographie

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, pages 248–255. IEEE, 2009.

[2] Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. Ppdsparse : A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553. ACM, 2017.

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove : Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[4] Ase Dragland. Big data–for better or worse. *SINTEF. no. 22 May 2013. Web. 27 Oct*, 2013.

[5] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[6] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] Thorsten Joachims. Text categorization with support vector machines : Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[9] Sellappan Palaniappan and Rafiah Awang. Intelligent heart disease prediction system using data mining techniques. In *Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on*, pages 108–115. IEEE, 2008.

[10] Andrew McAfee, Erik Brynjolfsson, Thomas H Davenport, DJ Patil, and Dominic Barton. Big data. *The management revolution. Harvard Bus Rev*, 90(10) :61–67, 2012.

[11] Kenneth Cukier and Viktor Mayer-Schoenberger. The rise of big data : How it's changing the way we think about the world. *Foreign Aff.*, 92 :28, 2013.

[12] Cynthia Rudin and Kiri L Wagstaff. Machine learning for science and society, 2014.

[13] Michael J Berry and Gordon Linoff. *Data mining techniques : for marketing, sales, and customer support*. John Wiley & Sons, Inc., 1997.

[14] Yutaka Matsuo and Mitsuru Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01) :157–169, 2004.

[15] Yufeng Kou, Chang-Tien Lu, Sirirat Sirwongwattana, and Yo-Ping Huang. Survey of fraud detection techniques. In *IEEE international conference on Networking, sensing and control*, volume 2, pages 749–754. IEEE, 2004.

[16] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification : An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3) :1–13, 2007.

[17] Patrick Pantel, Dekang Lin, et al. Spamcop : A spam classification & organization program. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, pages 95–98, 1998.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[19] Hung-Yi Lo, Ju-Chiang Wang, Hsin-Min Wang, and Shou-De Lin. Cost-sensitive multi-label learning for audio tag annotation and retrieval. *IEEE Transactions on Multimedia*, 13(3) :518–529, 2011.

[20] Forrest Briggs, Yonghong Huang, Raviv Raich, Konstantinos Eftaxias, Zhong Lei, William Cukierski, Sarah Frey Hadley, Adam Hadley, Matthew Betts, Xiaoli Z Fern, et al. The 9th annual mlsp competition : New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–8. IEEE, 2013.

[21] Ricardo Cabral, Fernando De la Torre, Joao Paulo Costeira, and Alexandre Bernardino. Matrix completion for weakly-supervised multi-label image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1) :121–135, 2015.

[22] Jung-Yi Jiang, Shian-Chi Tsai, and Shie-Jue Lee. Fsknn : multi-label text categorization based on fuzzy similarity and k nearest neighbors. *Expert Systems with Applications*, 39(3) :2813–2821, 2012.

[23] Haytham Elghazel, Alex Aussem, Ouadie Gharroudi, and Wafa Saadaoui. Ensemble multi-label text categorization based on rotation forest and latent semantic indexing. *Expert Systems with Applications*, 57 :1–11, 2016.

[24] Xiao Wang, Weiwei Zhang, Qiuwen Zhang, and Guo-Zheng Li. Multip-schlo : multi-label protein subchloroplast localization prediction with chou's pseudo amino acid composition and a novel multi-label classifier. *Bioinformatics*, 31(16) :2639–2645, 2015.

[25] Jian-Sheng Wu, Sheng-Jun Huang, and Zhi-Hua Zhou. Genome-wide protein function prediction through multi-instance multi-label learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(5) :891–902, 2014.

[26] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1) :5–53, 2004.

[27] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. Social collaborative filtering by trust. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8) :1633–1647, 2017.

[28] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9) :1757–1771, 2004.

[29] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn : A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7) :2038–2048, 2007.

[30] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification : An overview. *International Journal of Data Warehousing and Mining*, 3(3) :1–13, 2006.

[31] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8) :1819–1837, 2014.

[32] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[33] Mohammad S Sorower. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 2010.

[34] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9) :3084–3104, 2012.

[35] J Ross Quinlan et al. Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.

[36] Yashoteja Prabhu and Manik Varma. Fastxml : A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD*

international conference on Knowledge discovery and data mining, pages 263–272. ACM, 2014.

[37] Jason Weston, Ameesh Makadia, and Hector Yee. Label partitioning for sublinear ranking. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 181–189, 2013.

[38] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 935–944. ACM, 2016.

[39] Noureddine-Yassine Nair-Benrekia, Pascale Kuntz, and Frank Meyer. Learning from multi-label data with interactivity constraints : an extensive experimental study. *Expert Systems with Applications*, 42(13) :5723–5736, 2015.

[40] Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski. Ensembles of multi-objective decision trees. *Machine Learning : ECML 2007*, pages 624–631, 2007.

[41] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, pages 30–44, 2008.

[42] Rohit Babbar and Bernhard Schölkopf. Dismec : Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 721–729. ACM, 2017.

[43] Hsiang-fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. Large-scale multi-label learning with missing labels. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 593–601, 2014.

[44] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738, 2015.

[45] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hullermeier. Extreme f-measure maximization using sparse probability estimates. In *International Conference on Machine Learning*, pages 1435–1444, 2016.

[46] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. Pd-sparse : A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning*, pages 3069–3077, 2016.

[47] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11) :559–572, 1901.

[48] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4) :321–377, 1936.

[49] Christopher JC Burges. Geometric methods for feature extraction and dimensional reduction-a guided tour. In *Data mining and knowledge discovery handbook*, pages 53–82. Springer, 2009.

[50] Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*, volume 22, pages 772–780, 2009.

[51] Sunil K Jha and RDS Yadava. Denoising by singular value decomposition and its application to electronic nose data processing. *IEEE Sensors Journal*, 11(1) :35–44, 2011.

[52] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning : A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8) :1798–1828, 2013.

[53] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[54] Yin Zhang and Zhi-Hua Zhou. Multilabel dimensionality reduction via dependence maximization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(3) :14, 2010.

[55] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751, 2013.

[56] Newton Spolaôr, Maria Carolina Monard, Grigorios Tsoumakas, and Huei Diana Lee. A systematic review of multi-label feature selection and a new method based on label construction. *Neurocomputing*, 180 :3–15, 2016.

[57] George H John, Ron Kohavi, Karl Pfleger, et al. Irrelevant features and the subset selection problem. In *Machine learning : proceedings of the eleventh international conference*, pages 121–129, 1994.

[58] Jianping Hua, Waibhav D Tembe, and Edward R Dougherty. Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, 42(3) :409–424, 2009.

[59] Monica Rogati and Yiming Yang. High-performing feature selection for text classification. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 659–661. ACM, 2002.

[60] Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Weinberger. Supervised semantic indexing. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 187–196. ACM, 2009.

[61] Stéphane Clinchant and Florent Perronnin. Aggregating continuous word embeddings for information retrieval. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 100–109, 2013.

[62] Andrew L Maas and Andrew Y Ng. A probabilistic model for semantic word vectors. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[63] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006.

[64] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews : Computational Statistics*, 2(4) :433–459, 2010.

[65] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786) :504–507, 2006.

[66] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004.

[67] Avrim Blum. Random projection, margins, kernels, and feature-selection. In *Subspace, Latent Structure and Feature Selection*, pages 52–68. Springer, 2006.

[68] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8) :832–844, 1998.

[69] Andreas Janecek, Wilfried Gansterer, Michael Demel, and Gerhard Ecker. On the relationship between feature selection and classification accuracy. In *New Challenges for Feature Selection in Data Mining and Knowledge Discovery*, pages 90–105, 2008.

[70] Bo Li, Chao Wang, and De-Shuang Huang. Supervised feature extraction based on orthogonal discriminant projection. *Neurocomputing*, 73(1) :191–196, 2009.

[71] Liang Sun, Shuiwang Ji, and Jieping Ye. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 668–676. ACM, 2008.

[72] Kai Yu, Shipeng Yu, and Volker Tresp. Multi-label informed latent semantic indexing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–265. ACM, 2005.

[73] Jianhua Xu, Jiali Liu, Jing Yin, and Chengyu Sun. A multi-label feature extraction algorithm via maximizing feature variance and feature-label dependence simultaneously. *Knowledge-Based Systems*, 98 :172–184, 2016.

[74] Farbound Tai and Hsuan-Tien Lin. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9) :2508–2542, 2012.

[75] Kuan-Hao Huang and Hsuan-Tien Lin. Cost-sensitive label embedding for multi-label classification. *arXiv preprint arXiv :1603.09048*, 2016.

[76] Paul Mineiro and Nikos Karampatziakis. Fast label embeddings via randomized linear algebra. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 37–51. Springer, 2015.

[77] Tianyi Zhou and Dacheng Tao. Multi-label subspace ensemble. In *AISTATS*, pages 1444–1452, 2012.

[78] Ju-Jie Zhang, Min Fang, Hongchun Wang, and Xiao Li. Dependence maximization based label space dimension reduction for multi-label classification. *Engineering Applications of Artificial Intelligence*, 45 :453–463, 2015.

[79] Xin Li and Yuhong Guo. Bi-directional representation learning for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 209–224. Springer, 2014.

[80] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning (ICML-04)*, page 104. ACM, 2004.

[81] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12) :3397–3415, 1993.

[82] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1) :129–159, 2001.

[83] Liang Sun, Shuiwang Ji, and Jieping Ye. *Multi-label dimensionality reduction*. CRC Press, 2013.

[84] Qiaomin Ye, Luo Luo, and Zhihua Zhang. Frequent direction algorithms for approximate matrix multiplication with applications in cca. *Computational Complexity*, 1(m3) :2, 2016.

[85] Yi Zhang and Jeff Schneider. Multi-label output codes using canonical correlation analysis. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 873–882, 2011.

[86] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning (ICML-13)*, pages 1247–1255, 2013.

[87] Eakasit Pacharawongsakda and Thanaruk Theeramunkong. A two-stage dual space reduction framework for multi-label classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 330–341. Springer, 2013.

[88] Eakasit Pacharawongsakda and Thanaruk Theeramunkong. Multi-label classification using dependent and independent dual space reduction. *The Computer Journal*, 56(9) :1113–1135, 2013.

[89] Kalyanmoy Deb, Karthik Sindhya, and Jussi Hakanen. Multi-objective optimization. In *Decision Sciences : Theory and Practice*, pages 145–184. CRC Press, 2016.

[90] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6) :369–395, 2004.

[91] Ioannis Giagkiozis and Peter J Fleming. Pareto front estimation for decision making. *Evolutionary computation*, 22(4) :651–678, 2014.

[92] Yuhong Guo and Dale Schuurmans. Semi-supervised multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 355–370. Springer, 2012.

[93] Shuiwang Ji and Jieping Ye. Linear dimensionality reduction for multi-label classification. In *Proceedings of the 21st international jont conference on Artifical intelligence, IJCAI-09*, pages 1077–1082, 2009.

[94] Tingting Mu and Sophia Ananiadou. Proximity-based graph embeddings for multi-label classification. In *International Conference on Knowledge Discovery and Information Retrieval*, pages 74–84, 2010.

[95] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6) :391, 1990.

[96] Yi Liu, Rong Jin, and Liu Yang. Semi-supervised multi-label learning by constrained non-negative matrix factorization. In *Proceedings of the 21st national conference on Artificial intelligence (AAAI-06)*, pages 421–426. AAAI Press, 2006.

[97] Manfred K Warmuth and Dima Kuzmin. Randomized pca algorithms with regret bounds that are logarithmic in the dimension. In *Advances in neural information processing systems*, pages 1481–1488, 2006.

[98] Rong Yan, Jelena Tesic, and John R Smith. Model-shared subspace boosting for multi-label classification. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 834–843. ACM, 2007.

[99] Effrosyni Kokiopoulou and Yousef Saad. Orthogonal neighborhood preserving projections : A projection-based dimensionality reduction technique. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12) :2143–2156, 2007.

[100] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[101] Roman Rosipal and Nicole Krämer. Overview and recent advances in partial least squares. In *Subspace, latent structure and feature selection*, pages 34–51. Springer, 2006.

[102] Shuiwang Ji, Lei Tang, Shipeng Yu, and Jieping Ye. Extracting shared subspace for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 381–389. ACM, 2008.

[103] Buyue Qian and Ian Davidson. Semi-supervised dimension reduction for multi-label classification. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 569–574. AAAI Press, 2010.

[104] Hua Wang, Chris Ding, and Heng Huang. Multi-label linear discriminant analysis. In *European Conference on Computer Vision*, pages 126–139. Springer, 2010.

[105] Maria Oikonomou and Anastasios Tefas. Direct multi-label linear discriminant analysis. In *International Conference on Engineering Applications of Neural Networks*, pages 414–423. Springer, 2013.

[106] Ping Li, Hong Li, and Min Wu. Multi-label ensemble based on variable pairwise constraint projection. *Information Sciences*, 222 :269–281, 2013.

[107] Gaofeng Luo, Tongcheng Huang, and Zijuan Shi. Multi-label classification using hypergraph orthonormalized partial least squares. *Journal of Computers*, 9(6) :1364–1370, 2014.

[108] Xin Shu, Darong Lai, Huanliang Xu, and Liang Tao. Learning shared subspace for multi-label dimensionality reduction via dependence maximization. *Neurocomputing*, 168 :356–364, 2015.

[109] Weiwei Liu and Ivor W Tsang. Large margin metric learning for multi-label prediction. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 2800–2806. AAAI Press, 2015.

[110] Wissam Siblini, Reda Alami, Frank Meyer, and Pascale Kuntz. Supervised feature space reduction for multi-label nearest neighbors. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 182–191. Springer, 2017.

[111] Jörg Wicker, Bernhard Pfahringer, and Stefan Kramer. Multi-label classification using boolean matrix decomposition. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 179–186. ACM, 2012.

[112] Krishnakumar Balasubramanian and Guy Lebanon. The landmark selection method for multiple output prediction. *arXiv preprint arXiv :1206.6479*, 2012.

[113] Wei Bi and James Kwok. Efficient multi-label classification with many labels. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 405–413, 2013.

[114] Yao-Nan Chen and Hsuan-Tien Lin. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems*, pages 1529–1537, 2012.

[115] Ashish Kapoor, Raajay Viswanathan, and Prateek Jain. Multilabel classification using bayesian compressed sensing. In *Advances in Neural Information Processing Systems*, pages 2645–2653, 2012.

[116] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. Multi-label classification via feature-aware implicit label space encoding. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 325–333, 2014.

[117] Eakasit Pacharawongsakda and Thanaruk Theeramunkong. A comparative study on single and dual space reduction in multi-label classification. In *Knowledge, Information and Creativity Support Systems : Recent Trends, Advances and Solutions*, pages 389–400. Springer, 2016.

[118] Liang Sun, Shuiwang Ji, and Jieping Ye. Canonical correlation analysis for multilabel classification : A least-squares formulation, extensions, and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1) :194–200, 2011.

[119] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie : scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence (IJCAI-11)*, pages 2764–2770. AAAI Press, 2011.

[120] Farzaneh Mirzazadeh, Yuhong Guo, and Dale Schuurmans. Convex co-embedding. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, pages 1989–1996. AAAI Press, 2014.

[121] Piyush Rai, Changwei Hu, Ricardo Henao, and Lawrence Carin. Large-scale bayesian multi-label learning via topic-based label embeddings. In *Advances in Neural Information Processing Systems*, pages 3222–3230, 2015.

[122] Xin Li and Yuhong Guo. Multi-label classification with feature-aware non-linear label space transformation. In *Proceedings of the 24th International Conference on Artificial Intelligence (AAAI-15)*, pages 3635–3642. AAAI Press, 2015.

[123] Chang Xu, Dacheng Tao, and Chao Xu. Robust extreme multi-label learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284. ACM, 2016.

[124] Si Si, Kai-Yang Chiang, Cho-Jui Hsieh, Nikhil Rao, and Inderjit S Dhillon. Goal-directed inductive matrix completion. In *Proceedings of the 22nd ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, pages 1165–1174. ACM, 2016.

[125] Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. Learning deep latent spaces for multi-label classification. pages 2838–2844, 2017.

[126] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

[127] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis : An overview with application to learning methods. *Neural computation*, 16(12) :2639–2664, 2004.

[128] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500) :2323–2326, 2000.

[129] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10) :1338–1351, 2006.

[130] Min-Ling Zhang. Ml-rbf : Rbf neural networks for multi-label learning. *Neural Processing Letters*, 29(2) :61–74, 2009.

[131] Yan Huang, Wei Wang, Liang Wang, and Tieniu Tan. Multi-task deep neural network for multi-label learning. In *20th IEEE International Conference on Image Processing (ICIP)*, pages 2897–2900. IEEE, 2013.

[132] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. Large-scale multi-label text classification - revisiting neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2014.

[133] Min-Ling Zhang, José M Peña, and Victor Robles. Feature selection for multi-label naive bayes classification. *Information Sciences*, 179(19) :3218–3229, 2009.

[134] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1) :1–27, 1964.

[135] David L Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6) :797–829, 2006.

[136] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted l 1 minimization. *Journal of Fourier analysis and applications*, 14(5-6) :877–905, 2008.

[137] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 67(2) :301–320, 2005.

[138] Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 477–485. Association for Computational Linguistics, 2009.

[139] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1) :1929–1958, 2014.

[140] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining : Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[141] Chih-Jen Lin, Ruby C Weng, and S Sathiya Keerthi. Trust region newton method for logistic regression. *Journal of Machine Learning Research*, 9(Apr) :627–650, 2008.

[142] Prateek Jain, Raghu Meka, and Inderjit S Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 937–945, 2010.

[143] Pablo Sprechmann, Roee Litman, Tal Ben Yakar, Alexander M Bronstein, and Guillermo Sapiro. Supervised sparse analysis and synthesis operators. In *Advances in Neural Information Processing Systems*, pages 908–916, 2013.

[144] Yukihiro Tagami. Annexml : Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 455–464. ACM, 2017.

[145] Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Paliouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. Lshtc : A benchmark for large-scale text classification. *arXiv preprint arXiv :1503.08581*, 2015.

[146] Moustapha Cissé, Maruan Al-Shedivat, and Samy Bengio. Adios : Architectures deep in output space. In *International Conference on Machine Learning*, pages 2770–2779, 2016.

[147] Moustapha M Cisse, Nicolas Usunier, Thierry Artieres, and Patrick Gallinari. Robust bloom filters for large multilabel classification tasks. In *Advances in Neural Information Processing Systems*, pages 1851–1859, 2013.

[148] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *Esann*, volume 99, pages 219–224, 1999.

[149] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[150] Zhi-Hua Zhou. *Ensemble methods : foundations and algorithms*. Chapman and Hall/CRC, 2012.

[151] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[152] Tin Kam Ho. Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE, 1995.

[153] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7) :1545–1588, 1997.

[154] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC bioinformatics*, 9(1) :307, 2008.

[155] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1) :3–42, 2006.

[156] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets : An ensemble method for multilabel classification. In *European conference on machine learning*, pages 406–417. Springer, 2007.

[157] Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski. Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3) :817–833, 2013.

[158] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. Multi-label learning with millions of labels : Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24. ACM, 2013.

[159] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3) :4, 2006.

[160] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[161] Koby Crammer and Yoram Singer. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3(Feb) :1025–1058, 2003.

[162] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *International Conference on Machine Learning*, pages 53–61, 2013.

[163] Deng Cai, Xiaofei He, and Jiawei Han. Efficient kernel discriminant analysis via spectral regression. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 427–432. IEEE, 2007.

[164] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[165] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5) :403–420, 1970.

[166] Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4) :551–566, 1993.

[167] Beresford N Parlett. *The Symmetric Eigenvalue Problem*, volume 20. SIAM, 1998.

[168] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[169] RV Mises and Hilda Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9(2) :152–164, 1929.

[170] Cornelius Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

[171] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–588. ACM, 2013.

[172] Xin Shu, Huanliang Xu, and Liang Tao. A least squares formulation of multi-label linear discriminant analysis. *Neurocomputing*, 156 :221–230, 2015.

[173] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[174] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions : A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1), 2007.

[175] Mee Young Park and Trevor Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 69(4) :659–677, 2007.

[176] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[177] Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Semi-supervised dimensionality reduction. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 629–634. SIAM, 2007.

[178] Philip E Gill, Walter Murray, and Margaret H Wright. Practical optimization. *London : Academic Press, 1981*, 1981.

[179] Augustin Cauchy. Méthode générale pour la résolution des systemes d'"équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847) :536–538, 1847.

[180] Emmanuel J Candès and Terence Tao. The power of convex relaxation : Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5) :2053–2080, 2010.

[181] Stephen J Wright. *Primal-dual interior-point methods*. SIAM, 1997.

[182] Michele Benzi. Preconditioning techniques for large linear systems : a survey. *Journal of computational Physics*, 182(2) :418–477, 2002.

[183] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004, 2003.

[184] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[185] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3) :475–494, 2001.

[186] Cornelis Roos, Tamás Terlaky, and Jean-Philippe Vial. *Theory and algorithms for linear optimization*. Wiley, 1998.

[187] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2) :95–110, 1956.

[188] Joseph Louis Lagrange. *Mécanique analytique*, volume 1. Mallet-Bachelier, 1853.

[189] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1) :1–122, 2011.

[190] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[191] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[192] William G Cochran. Problems arising in the analysis of a series of similar experiments. *Supplement to the Journal of the Royal Statistical Society*, 4(1) :102–118, 1937.

[193] Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in Statistics*, pages 66–70. Springer, 1992.

[194] Khalili Ahmad and Shashaani Lily. The effectiveness of computer applications : A meta-analysis. *Journal of Research on computing in Education*, 27(1) :48–61, 1994.

[195] So Young Sohn. Meta analysis of classification algorithms for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11) :1137–1144, 1999.

[196] Adrien Jamain and David J Hand. Mining supervised classification performance studies : A meta-analytic investigation. *Journal of Classification*, 25(1) :87–112, 2008.

[197] Rafael B Pereira, Alexandre Plastino, Bianca Zadrozny, and Luiz HC Merschmann. Correlation analysis of performance measures for multi-label classification. *Information Processing & Management*, 54(3) :359–369, 2018.

[198] Olivier Hudry and Bernard Monjardet. Consensus theories. an oriented survey. *Mathématiques et sciences humaines. Mathematics and social sciences*, (190) :139–167, 2010.

[199] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the American Control Conference*, pages 1859–1864. IEEE, 2005.

[200] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan) :1–30, 2006.

[201] Arthur B Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11) :558–562, 1962.

[202] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv :1306.6709*, 2013.

[203] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan : A java library for multi-label learning. *Journal of Machine Learning Research*, 12(Jul) :2411–2414, 2011.

[204] Borja Calvo and Guzman Santafe. scmamp : Statistical comparison of multiple algorithms in multiple problems. *The R Journal*, Accepted for publication, 2015.

[205] Hoa T Le, Christophe Cerisara, and Alexandre Denis. Do convolutional networks need to be deep for text classification ? *arXiv preprint arXiv :1707.04108*, 2017.

[206] Diederik P Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

[207] Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. Deep extreme multi-label learning. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 100–107. ACM, 2018.

[208] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. Dadiannao : A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622. IEEE Computer Society, 2014.

[209] Ashish Kumar, Saurabh Goyal, and Manik Varma. Resource-efficient machine learning in 2 kb ram for the internet of things. In *International Conference on Machine Learning*, pages 1935–1944, 2017.

[210] Chirag Gupta, Arun Sai Suggala, Ankit Goyal, Harsha Vardhan Simhadri, Bhargavi Paranjape, Ashish Kumar, Saurabh Goyal, Raghavendra Udupa, Manik Varma, and Prateek Jain. Protonn : Compressed and accurate knn for resource-scarce devices. In *International Conference on Machine Learning*, pages 1331–1340, 2017.

[211] Arkaitz Zubiaga. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv :1202.5469*, 2012.

[212] Zemin Zheng, Yingying Fan, and Jinchi Lv. High dimensional thresholded regression and shrinkage effect. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 76(3) :627–649, 2014.

[213] R. Babbar and B. Schölkopf. Dismec distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM 2017)*, pages 721–729, 2017.

[214] Andrew Barron, Lucien Birgé, and Pascal Massart. Risk bounds for model selection via penalization. *Probability theory and related fields*, 113(3) :301–413, 1999.

[215] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1) :34–81, 2009.

[216] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[217] Graham Cormode and Shan Muthukrishnan. An improved data stream summary : the count-min sketch and its applications. *Journal of Algorithms*, 55(1) :58–75, 2005.

[218] Shanmugavelayutham Muthukrishnan et al. Data streams : Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2) :117–236, 2005.

[219] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[220] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

[221] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction : a comparative. *J Mach Learn Res*, 10 :66–71, 2009.

[222] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction : survey, insights, and generalizations. *Journal of Machine Learning Research*, 16(1) :2859–2900, 2015.

[223] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction : applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

[224] Christos Boutsidis, Anastasios Zouzias, and Petros Drineas. Random projections for k-means clustering. In *Advances in Neural Information Processing Systems*, pages 298–306, 2010.

[225] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 459–468. IEEE, 2006.

[226] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3422–3426. IEEE, 2013.

[227] Graham Cormode and Marios Hadjieleftheriou. Finding frequent items in data streams. *Proceedings of the VLDB Endowment*, 1(2) :1530–1541, 2008.

[228] Sergei Vassilvitskii. Lecture 3 : Counting on streams. `http://www.cs.columbia.edu/~coms699812/lecture3.pdf`, Columbia University. [Online ; accessed 2017-05-10].

[229] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[230] Austin Appleby. Murmurhash 2.0 - `https://sites.google.com/site/murmurhash/`, 2008.

[231] Tong Yang, Lingtong Liu, Yibo Yan, Muhammad Shahzad, Yulong Shen, Xiaoming Li, Bin Cui, and Gaogang Xie. Sf-sketch : A fast, accurate, and memory efficient data structure to store frequencies of data items. In *International Conference on Data Engineering*, 2017.

[232] David Arthur and Sergei Vassilvitskii. k-means++ : The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[233] Karthik Kambatla, Giorgos Kollias, Vipin Kumar, and Ananth Grama. Trends in big data analytics. *Journal of Parallel and Distributed Computing*, 74(7) :2561–2573, 2014.

[234] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing : Review and open research issues. *Information Systems*, 47 :98–115, 2015.

[235] Jeffrey Dean, David Patterson, and Cliff Young. A new golden age in computer architecture : Empowering the machine learning revolution. *IEEE Micro*, 2018.

[236] Claire Gauzente, Pierre Volle, et al. Développer l'intelligence client. In *Stratégie clients : Points de vue d'experts sur le management de la relation client, P. Volle (ed)*. Pearson, 2012.

[237] Daniel A Keim, Milos Krstajic, Christian Rohrdantz, and Tobias Schreck. Real-time visual analytics for text streams. *Computer*, (7) :47–55, 2013.

[238] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Major components of the gravity recommendation system. *ACM SIGKDD Explorations Newsletter*, 9(2) :80–83, 2007.

[239] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.

[240] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Parabel : Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 993–1002. International World Wide Web Conferences Steering Committee, 2018.

[241] Rohit Babbar and Bernhard Schölkopf. Adversarial extreme multi-label classification. *arXiv preprint arXiv :1803.01570*, 2018.

[242] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[243] Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 441–449. ACM, 2018.

[244] Corbin Rosset. A review of online decision tree learning algorithms. 2015.

[245] Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec, and Horst Bischof. On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1393–1400. IEEE, 2009.

[246] Wen Sun, Alina Beygelzimer, Hal Daumé III, John Langford, and Paul Mineiro. Contextual memory trees. *arXiv preprint arXiv :1807.06473*, 2018.

[247] Wissam Siblini, Frank Meyer, and Pascale Kuntz. Craftml, an efficient clustering-based random forest for extreme multi-label learning. In *International Conference on Machine Learning*, pages 4671–4680, 2018.

[248] Wissam Siblini, Frank Meyer, and Pascale Kuntz. Vipe : A new interactive classification framework for large sets of short texts-application to opinion mining. *arXiv preprint arXiv :1803.02101*, 2018.

[249] Frank Meyer, Sylvie Tricot, Pascale Kuntz, and Wissam Siblini. Vipe : un outil interactif de classification multilabel de messages courts. In *Extraction et Gestion des Connaissances. EGC 2017*, 2017.

[250] Wissam Siblini, Pascale Kuntz, and Frank Meyer. Vers un apprentissage multi-label rapide en grande dimension–une étude préliminaire. In *Conférence AAFD & SFC 2016*, 2016.

Chapitre 8

Annexes

Sommaire

1 Annexe 1 - Jeux de données de l'apprentissage multi-label standard et extrême

Dans cette annexe, nous détaillons les caractéristiques des jeux de données multi-label utilisés dans la thèse. Le tableau 8.1 présente les jeux de données standards et le tableau 8.2 présente les jeux de données utilisés en XML.

Tableau 8.1 – Description de treize jeux de données multi-label standards : domaine d'application, nombre d'instances d'apprentissage (n), nombre d'instances de test ($n_{\mathcal{S}}$), nombre d'attributs (d_x), nombre de labels (d_y).

Jeu	Application	n	$n_{\mathcal{S}}$	d_x	d_y
Yeast	Santé	2173	244	103	14
Emotions	Audio	533	60	72	6
Mediamill	Video	39516	4391	120	101
Scene	Image	2166	241	294	6
Corel5k	Image	4500	500	499	374
Delicious	Page Web	14495	1610	500	983
Enron	Texte	1531	171	1001	53
Genbase	Santé	595	67	1186	27
Medical	Santé	880	98	1449	45
Bibtex	Texte	6656	739	1836	159
Bookmarks	Texte	79070	8786	2150	208
Rcv1	Texte	3000	3000	47229	101
Reuters	Texte	5400	600	47229	101

Tableau 8.2 – Description des neuf jeux de données d'apprentissage multi-label extrême utilisés dans l'étude sur CRAFTML : domaine d'application, nombre d'instances d'apprentissage (n), nombre d'instances de test ($n_{\mathcal{S}}$), nombre d'attributs (d_x), nombre de labels (d_y).

Jeu	Application	d_x	d_y	n	$n_{\mathcal{S}}$
Mediamill	Vidéo	120	101	30993	12914
Bibtex	Texte	1836	159	4880	2515
Delicious	Page Web	500	983	12920	3185
EURLex-4K	Texte	5000	3993	15539	3809
AmazonCat-13K	Recommandation	203882	13330	1186239	306782
Wiki10-31K	Texte	101938	30938	14146	6616
Delicious-200K	Page Web	782585	205443	196606	100095
WikiLSHTC-325K	Texte	1617899	325056	1778351	587084
Amazon-670K	Recommandation	135909	670091	490449	153025

2 Annexe 2 - Résultats détaillés pour l'étude de ML-ARP

Dans cette annexe, nous présentons les résultats détaillés de l'étude sur ML-ARP. Les tableaux qui suivent présentent, pour chaque jeu de données de l'étude, douze mesures de performances prédictives pour les huit algorithmes considérés.

2.1 Yeast

— $k = 5$: Nombre de voisins.

— $d'_x = 64$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.335 ± 0.01	0.512 ± 0.0169	0.484 ± 0.013	**0.53** ± 0.014	0.376 ± 0.014	0.514 ± 0.0144	0.47 ± 0.011	0.471 ± 0.018
Average Precision	0.703 ± 0.01	0.759 ± 0.0165	0.750 ± 0.016	**0.761** ± 0.013	0.714 ± 0.01	0.759 ± 0.014	0.746 ± 0.015	0.746 ± 0.013
Coverage	6.79 ± 0.089	**6.34** ± 0.15	6.44 ± 0.116	6.37 ± 0.14	6.727 ± 0.075	6.35 ± 0.142	6.494 ± 0.134	6.496 ± 0.145
F1	0.228 ± 0.005	0.308 ± 0.008	0.295 ± 0.008	0.303 ± 0.007	0.246 ± 0.008	**0.309** ± 0.008	0.29 ± 0.006	0.29 ± 0.010
Hamming Loss	0.232 ± 0.003	0.195 ± 0.007	0.202 ± 0.006	**0.191** ± 0.005	0.227 ± 0.003	0.194 ± 0.007	0.203 ± 0.006	0.204 ± 0.005
Jaccard Loss	0.664 ± 0.01	0.487 ± 0.016	0.515 ± 0.013	0.49 ± 0.014	0.623 ± 0.014	**0.485** ± 0.014	0.529 ± 0.011	0.528 ± 0.018
One Error	0.247 ± 0.014	**0.238** ± 0.028	0.24 ± 0.023	0.24 ± 0.019	0.25 ± 0.016	**0.238** ± 0.02	0.24 ± 0.021	0.243 ± 0.018
Precision	**0.749** ± 0.015	0.716 ± 0.016	0.717 ± 0.019	0.717 ± 0.013	0.72 ± 0.019	0.721 ± 0.017	0.723 ± 0.012	0.725 ± 0.014
RMSE	0.327 ± 0.004	**0.309** ± 0.004	0.311 ± 0.001	0.310 ± 0.002	0.324 ± 0.004	**0.309** ± 0.004	0.313 ± 0.002	0.314 ± 0.001
Ranking Loss	0.211 ± 0.009	**0.172** ± 0.012	0.178 ± 0.011	0.174 ± 0.01	0.204 ± 0.009	**0.172** ± 0.012	0.182 ± 0.011	0.181 ± 0.011
Recall	0.335 ± 0.009	0.589 ± 0.024	0.547 ± 0.019	**0.59** ± 0.023	0.397 ± 0.022	**0.59** ± 0.022	0.528 ± 0.017	0.531 ± 0.025
Subset Accuracy	0.014 ± 0.008	0.185 ± 0.019	0.152 ± 0.016	**0.198** ± 0.017	0.056 ± 0.014	0.183 ± 0.0169	0.142 ± 0.011	0.139 ± 0.015

Tableau 8.3 – Résultats expérimentaux sur le jeu de données Yeast

2.2 Emotions

— $k = 5$: Nombre de voisins.

— $d'_x = 64$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.374 ± 0.042	0.374 ± 0.04	**0.477** ± 0.063	0.109 ± 0.042	0.374 ± 0.042	0.359 ± 0.058	0.384 ± 0.065
Average Precision	0.578 ± 0.014	0.715 ± 0.023	0.718 ± 0.0225	**0.761** ± 0.037	0.592 ± 0.034	0.719 ± 0.023	0.716 ± 0.0437	0.725 ± 0.034
Coverage	3.15 ± 0.165	2.27 ± 0.128	2.27 ± 0.123	**2.002** ± 0.20	2.98 ± 0.19	2.278 ± 0.128	2.315 ± 0.218	2.272 ± 0.149
F1	0.0 ± 0.0	0.226 ± 0.025	0.22 ± 0.023	**0.277** ± 0.029	0.069 ± 0.027	0.226 ± 0.025	0.215 ± 0.031	0.229 ± 0.031
Hamming Loss	0.313 ± 0.002	0.262 ± 0.013	0.261 ± 0.02	**0.226** ± 0.022	0.31 ± 0.0189	0.262 ± 0.013	0.256 ± 0.0259	0.252 ± 0.029
Jaccard Loss	1.0 ± 0.0	0.625 ± 0.04	0.625 ± 0.04	**0.522** ± 0.063	0.89 ± 0.042	0.625 ± 0.042	0.64 ± 0.058	0.615 ± 0.065
One Error	0.542 ± 0.0327	0.374 ± 0.041	0.376 ± 0.047	**0.321** ± 0.059	0.553 ± 0.073	0.374 ± 0.041	0.374 ± 0.08	0.357 ± 0.06
Precision	0.0 ± 0.0	0.550 ± 0.058	0.554 ± 0.062	**0.627** ± 0.052	0.2 ± 0.085	0.55 ± 0.058	0.521 ± 0.06	0.544 ± 0.063
RMSE	0.431 ± 0.00049	0.41 ± 0.003	0.409 ± 0.004	**0.393** ± 0.01	0.43 ± 0.003	0.41 ± 0.003	0.409 ± 0.008	0.406 ± 0.008
Ranking Loss	0.412 ± 0.016	0.259 ± 0.025	0.256 ± 0.022	**0.207** ± 0.036	0.401 ± 0.034	0.259 ± 0.025	0.263 ± 0.046	0.249 ± 0.036
Recall	0.0 ± 0.0	0.421 ± 0.054	0.423 ± 0.05	**0.545** ± 0.064	0.111 ± 0.044	0.421 ± 0.054	0.402 ± 0.066	0.437 ± 0.06
Subset Accuracy	0.0 ± 0.0	0.154 ± 0.038	0.156 ± 0.026	**0.241** ± 0.083	0.033 ± 0.021	0.154 ± 0.038	0.154 ± 0.049	0.172 ± 0.07

Tableau 8.4 – Résultats expérimentaux sur le jeu de données Emotions

2.3 Mediamill

— $k = 5$: Nombre de voisins.

— $d'_x = 64$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.339 ± 0.004	0.4767 ± 0.003	0.469 ± 0.004	0.4825 ± 0.002	N\A	N\A	**0.518** ± 0.003	N\A
Average Precision	0.614 ± 0.002	0.7298 ± 0.002	0.726 ± 0.0035	0.7474 ± 0.0023	N\A	N\A	**0.758** ± 0.0029	N\A
Coverage	20.86 ± 0.156	14.18 ± 0.237	14.36 ± 0.30	14.09 ± 0.26	N\A	N\A	**12.55** ± 0.227	N\A
F1	0.223 ± 0.004	0.2910 ± 0.001	0.288 ± 0.002	0.299 ± 0.001	N\A	N\A	**0.307** ± 0.001	N\A
Hamming Loss	0.035 ± 0.003	0.027 ± 0.00023	0.0279 ± 0.00021	0.027 ± 0.00021	N\A	N\A	**0.025** ± 0.00014	N\A
Jaccard Loss	0.66 ± 0.009	0.5105 ± 0.003	0.517 ± 0.004	0.505 ± 0.003	N\A	N\A	**0.466** ± 0.003	N\A
One Error	0.23 ± 0.004	0.154 ± 0.005	0.157 ± 0.005	**0.1408** ± 0.006	N\A	N\A	0.145 ± 0.003	N\A
Precision	0.7 ± 0.008	0.7498 ± 0.005	0748 ± 0.005	0.761 ± 0.004	N\A	N\A	**0.764** ± 0.004	N\A
RMSE	0.127 ± 0.007	0.116 ± 0.00023	0.116 ± 0.00027	**0.104** ± 0.00022	N\A	N\A	0.112 ± 0.00018	N\A
Ranking Loss	0.064 ± 0.0089	0.037 ± 0.00085	0.0386 ± 0.001	0.037 ± 0.00087	N\A	N\A	**0.032** ± 0.000815	N\A
Recall	0.344 ± 0.04	0.526 ± 0.0028	0.519 ± 0.004	0.539 ± 0.003	N\A	N\A	**0.566** ± 0.003	N\A
Subset Accuracy	0.048 ± 0.0012	0.167 ± 0.0038	0.160 ± 0.004	0.171 ± 0.004	N\A	N\A	**0.217**± 0.005	N\A

Tableau 8.5 – Résultats expérimentaux sur le jeu de données Mediamill

2.4 Scene

— $k = 5$: Nombre de voisins.

— $d'_x = 128$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.669 ± 0.03	0.425 ± 0.02	0.658 ± 0.024	**0.683** ± 0.025	0.64 ± 0.033	0.191 ± 0.0379	0.197 ± 0.027
Average Precision	0.426 ± 0.001	**0.859** ± 0.018	0.625 ± 0.014	0.851 ± 0.016	0.857 ± 0.016	0.845 ± 0.02	0.635 ± 0.021	0.643 ± 0.02
Coverage	2.366 ± 0.08	0.521 ± 0.07	0.621 ± 0.1	0.546 ± 0.06	**0.5** ± 0.054	0.526 ± 0.07	1.418 ± 0.094	1.390 ± 0.084
F1	0.0 ± 0.0	0.341 ± 0.016	0.269 ± 0.018	0.336 ± 0.012	**0.347** ± 0.013	0.326 ± 0.01	0.097 ± 0.019	0.100 ± 0.014
Hamming Loss	0.179 ± 0.004	**0.088** ± 0.008	0.108 ± 0.005	0.091 ± 0.007	0.089 ± 0.0077	0.097 ± 0.01	0.166 ± 0.006	0.162 ±0.004
Jaccard Loss	1.0 ± 0.0	0.330 ± 0.03	0.56 ± 0.054	0.341 ± 0.024	**0.316** ± 0.025	0.359 ± 0.03	0.808 ± 0.037	0.802 ± 0.027
One Error	0.781 ± 0.0012	**0.228** ± 0.027	0.29 ± 0.035	0.241 ± 0.025	0.239 ± 0.026	0.262 ± 0.035	0.545 ± 0.03	0.537 ±0.029
Precision	0.0 ± 0.0	0.697 ± 0.033	0.598 ± 0.032	0.684 ± 0.026	**0.710** ± 0.025	0.665 ± 0.034	0.199 ± 0.037	0.205 ± 0.029
RMSE	0.394 ± 0.002	**0.289** ± 0.007	0.315 ± 0.0057	0.292 ± 0.005	**0.289** ± 0.0057	0.298 ± 0.007	0.372 ± 0.0036	0.371 ± 0.003
Ranking Loss	0.485 ± 0.0085	0.086 ± 0.014	0.251 ± 0.009	0.091 ± 0.013	**0.083** ± 0.012	0.088 ± 0.014	0.264 ± 0.018	0.259 ± 0.016
Recall	0.0 ± 0.0	0.684 ± 0.0325	0.184 ± 0.014	0.673 ± 0.023	**0.691** ± 0.027	0.653 ± 0.035	0.194 ± 0.038	0.199 ±0.028
Subset Accuracy	0.0 ± 0.0	0.627 ± 0.0288	0.516 ± 0.063	0.618 ± 0.025	**0.648** ± 0.025	0.602 ± 0.032	0.181 ± 0.037	0.186 ± 0.024

Tableau 8.6 – Résultats expérimentaux sur le jeu de données Scene

2.5 Corel5k

— $k = 5$: Nombre de voisins.

— $d'_x = 128$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.016 ± 0.0036	0.01 ± 0.0017	0.01 ± 0.0038	**0.03** ± 0.0059	0.01 ± 0.002	0.029 ± 0.007	N\A
Average Precision	**0.290** ± 0.005	0.251 ± 0.006	0.241 ± 0.01	0.245 ± 0.0057	0.273 ± 0.008	0.254 ± 0.007	0.260 ± 0.007	N\A
Coverage	**48.13** ± 2.1	105.6 ± 3.24	109.61 ± 3.12	108.76 ± 3.1	107.09 ± 2.96	110.097 ± 4.17	106.16 ± 2.43	N\A
F1	0.0 ± 0.0	0.010 ± 0.0022	0.007 ± 0.0011	0.0069 ± 0.002	**0.0194** ± 0.0035	0.007 ± 0.0014	0.019 ± 0.004	N\A
Hamming Loss	**0.009** ± 0.0	**0.009** ± 0.0	0.009 ± 0.0	**0.009** ± 0.0	**0.009** ± 0.0	**0.009** ± 0.0	**0.009** ± 0.0	N\A
Jaccard Loss	1.0 ± 0.0	0.983 ± 0.0036	0.988 ± 0.0017	0.989 ± 0.0038	**0.969** ± 0.0059	0.989 ± 0.002	0.970 ± 0.007	N\A
One Error	0.776 ± 0.008	0.737 ± 0.009	0.749 ± 0.02	0.738 ± 0.012	**0.707** ± 0.017	0.727 ± 0.011	0.724 ± 0.014	N\A
Precision	0.0 ± 0.0	0.031 ± 0.007	0.0265 ± 0.004	0.024 ± 0.0065	0.059 ± 0.01	0.023 ± 0.005	**0.060** ± 0.013	N\A
RMSE	**0.067** ± 0.0	0.069 ± 0.0	0.069 ± 0.0	0.069 ± 0.0	0.069 ± 0.0	0.069 ± 0.0	0.069 ± 0.0	N\A
Ranking Loss	0.147 ± 0.0021	0.139 ± 0.0036	0.141 ± 0.004	0.140 ± 0.003	**0.134** ± 0.003	0.136 ± 0.003	0.137 ± 0.003	N\A
Recall	0.0 ± 0.0	0.017 ± 0.0037	0.012 ± 0.0018	0.01 ± 0.003	**0.031** ± 0.006	0.011 ± 0.002	**0.031** ± 0.007	N\A
Subset Accuracy	0.0 ± 0.0	0.005 ± 0.0024	0.0006 ± 0.0009	0.001 ± 0.0009	**0.007** ± 0.0049	0.0008 ± 0.0009	0.005 ± 0.003	N\A

Tableau 8.7 – Résultats expérimentaux sur le jeu de données Corel5k

2.6 Delicious

— $k = 5$: Nombre de voisins.

— $d'_x = 128$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.104 ± 0.002	0.292 ± 0.01	**0.119** ± 0.0037	0.115 ± 0.002	0.101 ± 0.002	N/A	N/A
Average Precision	**0.419** ± 0.011	0.326 ± 0.0037	0.567 ± 0.01	0.319 ± 0.0038	0.337 ± 0.002	0.324 ± 0.002	N/A	N/A
Coverage	**393.19** ± 4.3	626.73 ± 8.19	20.31 ± 0.75	623.65 ± 8.35	618.01 ± 6.61	627.92 ± 5.13	N/A	N/A
F1	0.0 ± 0.0	0.081 ± 0.0019	0.183 ± 0.0059	0.076 ± 0.003	**0.089** ± 0.001	0.079 ± 0.002	N/A	N/A
Hamming Loss	0.0193 ± 0.0	**0.014** ± 0.0	0.02 ± 0.0	**0.014** ± 0.0	0.018 ± 0.0	0.018 ± 0.0	N/A	N/A
Jaccard Loss	0.999 ± 0.0005	0.894 ± 0.0029	0.625 ± 0.009	0.902 ± 0.0038	**0.884** ± 0.002	0.897 ± 0.0029	N/A	N/A
One Error	0.596 ± 0.009	0.402 ± 0.01	0.379 ± 0.012	0.414 ± 0.013	**0.389** ± 0.004	0.406 ± 0.0069	N/A	N/A
Precision	0.0 ± 0.0	0.457 ± 0.006	0.472 ± 0.012	0.443 ± 0.015	**0.477** ± 0.011	0.440 ± 0.01	N/A	N/A
RMSE	0.064 ± 0.000046	0.063 ± 0.00006	0.102 ± 0.0003	**0.062** ± 0.00004	**0.062** ± 0.0	0.063 ± 0.0	N/A	N/A
Ranking Loss	0.173 ± 0.0029	0.135 ± 0.002	0.0596 ± 0.003	0.137 ± 0.002	**0.131** ± 0.002	0.135 ± 0.002	N/A	N/A
Recall	0.0 ± 0.0	0.114 ± 0.003	0.33 ± 0.011	**0.105** ± 0.004	0.126 ± 0.002	0.111 ± 0.003	N/A	N/A
Subset Accuracy	0.0008 ± 0.00049	**0.002** ± 0.001	0.137 ± 0.006	**0.002** ± 0.0006	**0.002** ± 0.0007	**0.002** ± 0.00099	N/A	N/A

Tableau 8.8 – Résultats expérimentaux sur le jeu de données Delicious

2.7 Enron

— $k = 5$: Nombre de voisins.

— $d'_x = 128$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.222 ± 0.038	0.328 ± 0.018	0.314 ± 0.015	0.359 ± 0.024	**0.397** ± 0.039	0.387 ± 0.038	0.3108 ± 0.034	0.318 ± 0.016
Average Precision	0.539 ± 0.015	0.627 ± 0.017	0.619 ± 0.017	0.637 ± 0.016	**0.666** ± 0.018	0.661 ± 0.018	0.531 ± 0.022	0.542 ± 0.014
Coverage	13.55 ± 0.62	13.68 ± 0.547	14.02 ± 0.657	14.01 ± 0.657	**12.77** ± 0.76	13.28 ± 0.62	15.024 ± 0.67	14.86 ± 0.67
F1	0.165 ± 0.026	0.213 ± 0.009	0.207 ± 0.008	0.231 ± 0.013	**0.251** ± 0.019	0.247 ± 0.019	0.209 ± 0.018	0.213 ± 0.009
Hamming Loss	0.062 ± 0.001	0.051 ± 0.001	0.053 ± 0.001	0.05 ± 0.001	**0.049** ± 0.001	**0.049** ± 0.001	0.067 ± 0.003	0.064 ± 0.002
Jaccard Loss	0.777 ± 0.038	0.671 ± 0.018	0.685 ± 0.015	0.64 ± 0.02	**0.602** ± 0.039	0.612 ± 0.038	0.689 ± 0.034	0.681 ± 0.016
One Error	0.463 ± 0.02	0.309 ± 0.028	0.33± 0.028	0.288 ± 0.03	**0.260** ± 0.028	0.253 ± 0.023	0.500 ± 0.037	0.475 ± 0.036
Precision	0.519 ± 0.015	0.579 ± 0.025	0.564 ± 0.024	0.6117 ± 0.02	0.629 ± 0.03	**0.632** ± 0.038	0.477 ± 0.04	0.488 ± 0.026
RMSE	0.186 ± 0.004	0.163 ± 0.0	0.164 ± 0.0	**0.162** ± 0.0009	0.174 ± 0.004	0.175 ± 0.023	0.176 ± 0.0016	0.175 ± 0.001
Ranking Loss	0.118 ± 0.009	0.096 ± 0.007	0.1 ± 0.007	0.099 ± 0.007	**0.088** ± 0.008	0.091 ± 0.007	0.117 ± 0.009	0.115 ± 0.009
Recall	0.257 ± 0.055	0.371 ± 0.02	0.359 ± 0.016	0.409 ± 0.03	**0.457** ± 0.04	0.442 ± 0.04	0.418 ± 0.04	0.422 ± 0.025
Subset Accuracy	0.001 ± 0.002	0.072 ± 0.018	0.06 ± 0.017	0.078 ± 0.022	**0.104** ± 0.032	0.088 ± 0.027	0.048 ± 0.025	0.054 ± 0.016

Tableau 8.9 – Résultats expérimentaux sur le jeu de données Enron

2.8 Genbase

— $k = 5$: Nombre de voisins.

— $d'_x = 128$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ±	0.0 ±	0.0 ±	N\A
Average Precision	0.423 ± 0.013	**0.427** ± 0.012	0.427± 0.012	**0.427** ± 0.012	**0.427** ±	**0.427** ±	**0.427** ±	N\A
Coverage	**4.241** ± 0.26	4.365 ± 0.27	4.365 ± 0.27	4.365 ± 0.27	4.365 ±	4.365 ±	4.365 ±	N\A
F1	0.0 ± 0.0	0.0 ± 0.0	0.0± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N\A
Hamming Loss	**0.047** ± 0.001	**0.047** ± 0.001	0.047± 0.001	**0.047** ± 0.001	**0.047** ± 0.001	**0.047** ± 0.001	**0.047** ± 0.001	N\A
Jaccard Loss	1.0 ± 0.0	1.0 ± 0.0	1.0± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	N\A
One Error	**0.752** ± 0.015	**0.752** ± 0.015	0.752± 0.015	**0.752** ± 0.015	**0.752** ± 0.015	**0.752** ± 0.015	**0.752** ± 0.015	N\A
Precision	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N\A
RMSE	**0.195** ± 0.002	**0.195** ± 0.002	0.195± 0.002	**0.195** ± 0.002	**0.195** ± 0.002	**0.195** ± 0.002	**0.195** ± 0.002	N\A
Ranking Loss	**0.156** ± 0.005	**0.156** ± 0.005	0.156± 0.005	**0.156** ± 0.005	**0.156** ± 0.005	**0.156** ± 0.005	**0.156** ± 0.005	N\A
Recall	0.0 ± 0.0	0.0 ± 0.0	0.0± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N\A
Subset Accuracy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N\A

Tableau 8.10 – Résultats expérimentaux sur le jeu de données Genbase

2.9 Medical

— $k = 5$: Nombre de voisins.

— $d'_x = 128$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.612 ± 0.052	0.535 ± 0.033	0.580 ± 0.05	**0.679** ± 0.042	0.621 ± 0.041	N\A	N\A
Average Precision	0.399 ± 0.025	0.795 ± 0.035	0.752 ± 0.022	0.777 ± 0.026	**0.829** ± 0.035	0.804 ± 0.035	N\A	N\A
Coverage	5.313 ± 0.57	3.198 ± 0.994	3.369 ± 0.521	3.144 ± 0.809	**2.462** ± 0.61	2.987 ± 0.64	N\A	N\A
F1	0.0 ± 0.0	0.321 ± 0.025	0.282 ± 0.016	0.305 ± 0.026	**0.354** ± 0.02	0.325 ± 0.02	N\A	N\A
Hamming Loss	0.028 ± 0.0	0.015 ± 0.002	0.018 ± 0.001	0.017 ± 0.001	**0.013** ± 0.001	0.015 ± 0.02	N\A	N\A
Jaccard Loss	1.0 ± 0.0	0.387 ± 0.05	0.464 ± 0.03	0.419 ± 0.053	**0.320** ± 0.042	0.378 ± 0.002	N\A	N\A
One Error	0.735 ± 0.0355	0.254 ± 0.043	0.315 ± 0.034	0.277 ± 0.0369	**0.210** ± 0.046	0.241 ± 0.04	N\A	N\A
Precision	0.0 ± 0.0	0.671 ± 0.048	0.593 ± 0.036	0.643 ± 0.0522	**0.738** ± 0.042	0.682 ± 0.04	N\A	N\A
RMSE	0.204 ± 0.007	0.159 ± 0.007	0.126 ± 0.002	**0.122** ± 0.004	0.148 ± 0.008	0.157 ± 0.04	N\A	N\A
Ranking Loss	0.145 ± 0.009	0.063 ± 0.014	0.067 ± 0.011	0.062 ± 0.011	**0.051** ± 0.011	0.057 ± 0.007	N\A	N\A
Recall	0.0 ± 0.0	0.643 ± 0.054	0.564 ± 0.033	0.605 ± 0.05	**0.709** ± 0.042	0.643 ± 0.01	N\A	N\A
Subset Accuracy	0.0 ± 0.0	0.519 ± 0.0539	0.445 ± 0.034	0.490 ± 0.057	**0.587** ± 0.046	0.537 ± 0.04	N\A	N\A

Tableau 8.11 – Résultats expérimentaux sur le jeu de données Medical

2.10 Bibtex

— $k = 5$: Nombre de voisins.

— $d'_x = 128$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.14 ± 0.008	0.101 ± 0.008	0.122 ± 0.008	**0.305** ± 0.008	0.195 ± 0.006	N\A	N\A
Average Precision	0.203 ± 0.01	0.334 ± 0.0123	0.279 ± 0.0126	0.295 ± 0.01	**0.509** ± 0.01	0.391 ± 0.007	N\A	N\A
Coverage	42.13 ± 1.59	58.61 ± 2.02	64.46 ± 1.49	62.45 ± 1.02	**40.39** ± 1.32	50.94 ± 1.62	N\A	N\A
F1	0.0 ± 0.0	0.087 ± 0.004	0.063 ± 0.004	0.076 ± 0.0035	**0.178** ± 0.004	0.116 ± 0.03	N\A	N\A
Hamming Loss	0.015 ± 0.0001	0.0135 ± 0.0002	0.014 ± 0.0002	0.0139 ± 0.0002	**0.0123** ± 0.0002	0.013 ± 0.0002	N\A	N\A
Jaccard Loss	1.0 ± 0.0	0.859 ± 0.009	0.898 ± 0.008	0.877 ± 0.008	**0.694** ± 0.008	0.804 ± 0.006	N\A	N\A
One Error	0.853 ± 0.014	0.604 ± 0.019	0.681 ± 0.02	0.641 ± 0.018	**0.425** ± 0.017	0.552 ± 0.008	N\A	N\A
Precision	0.0 ± 0.0	0.270 ± 0.01	0.203 ± 0.01	0.241 ± 0.011	**0.456** ± 0.012	0.335 ± 0.01	N\A	N\A
RMSE	0.095 ± 0.0	0.091 ± 0.0	0.092 ± 0.0	0.092 ± 0.0	**0.086** ± 0.0004	0.089 ± 0.0003	N\A	N\A
Ranking Loss	0.330 ± 0.007	0.228 ± 0.006	0.258 ± 0.008	0.251 ± 0.008	**0.142** ± 0.006	0.194 ± 0.007	N\A	N\A
Recall	0.0 ± 0.0	0.144 ± 0.007	0.104 ± 0.008	0.124 ± 0.008	**0.327** ± 0.008	0.201 ± 0.006	N\A	N\A
Subset Accuracy	0.0 ± 0.0	0.059 ± 0.006	0.046 ± 0.006	0.055 ± 0.006	**0.172** ± 0.011	0.109 ± 0.0069	N\A	N\A

Tableau 8.12 – Résultats expérimentaux sur le jeu de données Bibtex

2.11 Bookmarks

— $k = 5$: Nombre de voisins.

— $d'_x = 128$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	0.209 ± 0.0028	0.193 ± 0.0028	**0.210** ± 0.0029	N\A	N\A	N\A	N\A
Average Precision	0.158 ± 0.002	**0.376** ± 0.0029	0.346 ± 0.002	0.357 ± 0.0021	N\A	N\A	N\A	N\A
Coverage	66.61 ± 1.268	**56.75** ± 0.457	59.8 ± 0.5	59.99 ± 0.414	N\A	N\A	N\A	N\A
F1	0.0 ± 0.0	**0.107** ± 0.0014	0.098 ± 0.001	0.098 ± 0.001	N\A	N\A	N\A	N\A
Hamming Loss	0.009 ± 0.0	**0.008** ± 0.0	0.008 ± 0.0	**0.008** ± 0.0	N\A	N\A	N\A	N\A
Jaccard Loss	1.0 ± 0.0	0.790 ± 0.002	0.806 ± 0.002	**0.785** ± 0.002	N\A	N\A	N\A	N\A
One Error	0.922 ± 0.0019	**0.641** ± 0.003	0.679 ± 0.003	0.67 ± 0.003	N\A	N\A	N\A	N\A
Precision	0.0 ± 0.0	**0.228** ± 0.003	0.204 ± 0.003	0.225 ± 0.0	N\A	N\A	N\A	N\A
RMSE	0.079 ± 0.0	**0.073** ± 0.0	0.074 ± 0.0	**0.073** ± 0.001	N\A	N\A	N\A	N\A
Ranking Loss	0.263 ± 0.002	**0.187** ± 0.002	0.2 ± 0.002	0.201 ± 0.0029	N\A	N\A	N\A	N\A
Recall	0.0 ± 0.0	**0.214** ± 0.002	0.197 ± 0.003	0.197 ± 0.0032	N\A	N\A	N\A	N\A
Subset Accuracy	0.0 ± 0.0	0.191 ± 0.002	0.18 ± 0.002	**0.195** ± 0.003	N\A	N\A	N\A	N\A

Tableau 8.13 – Résultats expérimentaux sur le jeu de données Bookmarks

2.12 Reuters subset 1

— $k = 5$: Nombre de voisins.

— $d'_x = 4000$: Nombre d'attributs réduits.

	BaseLine	ML-kNN	RP	ML-ARP	MDDM	PCA	OPLS	CCA
Accuracy	0.0 ± 0.0	**0.193** ± 0.0139	0.193 ± 0.014	0.189 ± 0.01	N\A	N\A	N\A	N\A
Average Precision	0.272 ± 0.012	**0.581** ± 0.01	0.568 ± 0.008	0.559 ± 0.012	N\A	N\A	N\A	N\A
Coverage	28.76 ± 1.529	**16.52** ± 0.825	17.34 ± 0.69	17.93 ± 0.83	N\A	N\A	N\A	N\A
F1	0.0 ± 0.0	**0.123** ± 0.008	0.121 ± 0.008	0.1192 ± 0.006	N\A	N\A	N\A	N\A
Hamming Loss	0.028 ± 0.0001	**0.026** ± 0.0004	0.026 ± 0.0004	**0.026** ± 0.0002	N\A	N\A	N\A	N\A
Jaccard Loss	1.0 ± 0.0	**0.806** ± 0.013	0.806 ± 0.014	0.810 ± 0.01	N\A	N\A	N\A	N\A
One Error	0.769 ± 0.017	**0.435** ± 0.014	0.459 ± 0.015	0.454 ± 0.019	N\A	N\A	N\A	N\A
Precision	0.0 ± 0.0	**0.359** ± 0.023	0.355 ± 0.019	0.355 ± 0.02	N\A	N\A	N\A	N\A
RMSE	0.129 ± 0.015	0.119 ± 0.001	0.116 ± 0.0005	**0.115** ± 0.001	N\A	N\A	N\A	N\A
Ranking Loss	0.167 0.003±	**0.071** ± 0.004	0.076 ± 0.003	0.080 ± 0.005	N\A	N\A	N\A	N\A
Recall	0.0 ± 0.0	**0.205** ± 0.016	0.203 ± 0.016	0.194 ± 0.01	N\A	N\A	N\A	N\A
Subset Accuracy	0.0 ± 0.0	0.062 ± 0.012	0.066 ± 0.007	**0.063** ± 0.007	N\A	N\A	N\A	N\A

Tableau 8.14 – Résultats expérimentaux sur le jeu de données Reuters

3 Annexe 3 - Applications textuelles avec CRAFTML

Cette annexe présente des résultats préliminaires sur les études applicatives exploratoires textuelles réalisées avec CRAFTML.

Tableau 8.15 – Exemples de traductions sac-de-mots à sac-de-mots avec CRAFTML entraîné sur la base Europarl.

Vous avez saisi : we must stop this war Voici la prédiction pour : must=1.0 stop=1.0 this=1.0 war=1.0 we=1.0 0 devons 2.9719677 1 nous 1.3313631 2 stopper 1.2767069 3 évolution 0.8132383 4 faut 0.49489406 5 cette 0.4863202 6 possibilité 0.34010455 ... 41 conflit 0.119121954 ...	Vous avez saisi : the meeting Voici la prédiction pour : meeting=1.0 the=1.0 0 réunion 1.8400643 1 lors 0.96069795 2 conseil 0.7127227 3 du 0.5892083 4 commission 0.54489845 5 a 0.49276412 6 au 0.4570371 7 eu 0.45354474 8 mars 0.45111978 9 en 0.45066226
Vous avez saisi : the war Voici la prédiction pour : the=1.0 war=1.0 0 guerre 2.7347791 1 décadent 0.73682725 2 sous-développement 0.57267046 3 cruelle 0.56600904 4 prolonge 0.5526204 5 est 0.54672444 6 mondiale 0.5245664 7 continuité 0.5040217 8 angola 0.47412044 9 tchétchénie 0.44143814	Vous avez saisi : agriculture is important Voici la prédiction pour : agriculture=1.0 important=1.0 is=1.0 0 important 2.2855823 1 c'est 2.0585666 2 importante 0.63127303 3 essentiel 0.5537987 4 s'agit 0.5258978 5 d'une 0.44573352 6 question 0.4047897 ... 26 l'agriculture 0.10119945 ...
Vous avez saisi : I would like to mention one final point (*exemple d'apprentissage*) Voici la prédiction pour : final=1.0 i=1.0 like=1.0 mention=1.0 one=1.0 point=1.0 to=1.0 would=1.0 0 dernier 2.5997846 1 point 2.503704 2 soulever 1.6814073 3 permets 1.4321631 4 voudrais 1.343031 5 je 1.1206332 6 un 1.0766649 7 me 0.7817933 8 encore 0.51368475 9 président 0.47233883	Vous avez saisi : Resumption of the session (*exemple d'apprentissage*) Voici la prédiction pour : of=1.0 resumption=1.0 session=1.0 the=1.0 0 session 2.6617866 1 reprise 2.4922042 2 la 0.51057255 3 du 0.38679272 4 est 0.3212504 5 de 0.3167285 6 une 0.3032409 7 annuelle 0.29608205 8 l' 0.25995567 9 pasok 0.23025851

Tableau 8.16 – Exemples de complétions de phrases caractère par caractère avec CRAFTML.

Vous avez saisi : je suis memoire de l'amorce : 0.5904899 e 0.5314409 i 0.80999994 j 0.47829682 s 0.9 u 0.7289999 prédiction de la suite : passé de la 3G à la 4G et je ne peux plus rapide que la 3G et qu en Wifi.Gain de temps et bien plu	Vous avez saisi : je vais aller chez memoire de l'amorce : 0.5904899 j 0.15009457 a 0.34867835 l 0.43046713 c 0.6560999 r 0.5314409 e 0.80999994 s 0.28242943 h 0.7289999 v 0.20589104 i 0.25418648 z 0.9 prédiction de la suite : moi je suis en Wifi donc le a nae ou le ci est souvent en 3G et maintenant je vais en parler de
Vous avez saisi : très mécont memoire de l'amorce : 0.47829682 é 0.5904899 c 0.6560999 m 0.5314409 n 0.80999994 o 0.7289999 r 0.34867835 s 0.43046713 t 0.9 è 0.38742042 prédiction de la suite : ent de la 4G pour le même prix mais depuis j'ai des soucis de réseau dans les transports en commun	Vous avez saisi : aucune diff memoire de l'amorce : 0.5904899 a 0.3138105 c 0.38742042 d 0.6560999 e 0.5314409 f 0.9 i 0.7289999 n 0.47829682 u 0.43046713 prédiction de la suite : érence avec la 3G # super————+++++++++ ++-++++++-++++++++++++++++ # super+++++++-+++++ ++++++++++
Vous avez saisi : depuis que memoire de l'amorce : 0.6560999 u 0.80999994 d 0.34867835 e 0.9 i 0.5314409 p 0.43046713 q 0.7289999 s 0.5904899 prédiction de la suite : j'ai la 4G je n'ai pas changé grand chose a mis je n'ai pas de réseau pour téléphone portable pour	Vous avez saisi : je n'ai memoire de l'amorce : 0.5904899 ' 0.7289999 a 0.80999994 e 0.5314409 i 0.9 j 0.47829682 n 0.6560999 prédiction de la suite : pas de réseau pour téléphone portable très rapidement sur le net en déplacement et j'ai besoin de r
Vous avez saisi : bon memoire de l'amorce : b 0.7289999 n 0.9 o 0.80999994 prédiction de la suite : jour je suis passé en 4G et pour le même prix !Savez vous que chez vous et je continue à payer un a	Vous avez saisi : le réseau est memoire de l'amorce : 0.6560999 s 0.80999994 a 0.5314409 t 0.9 e 0.7289999 u 0.5904899 l 0.25418648 é 0.38742042 r 0.34867835 prédiction de la suite : très faible en 4G (meme en ville) et ce n'est pas le cas partout et il est comme si j'étais en 3G

Tableau 8.17 – Exemples de clusters de mots construits à partir de leurs représentations Glove [3] et du clustering de CRAFTML en non supervisé.

bling, chains, claddagh, crown, crystal, diamond, diamonds, emerald, gem, gems, gemstones, gold, imitation, jewel, jeweler, jewelry, jewels, platinum, ring, rings, ruby, studded, wristwatch
art, artwork, artworks, collage, collages, collection, drawings, framed, framing, graffiti, graphic, illustration, illustrations, miscellaneous, originals, photo, photograph, photographs, photography, pictorial, portrait, portraits, postcard, postcards, poster, posters, print, prints, reproduction, reproductions, typography
canvas, canvases, etchings, l'oeil, mosaic, mural, murals, painted, painting, paintings, sculpture, sculptures, tapestries, tapestry, trompe, woodblock, woodcut
artisan, craft, crafted, crafting, crafts, creations, decorator, eco, handcrafted, handmade, heirloom, housewares, upcycled
ceramics, coin, coins, collectors, cutlery, deco, dolls, figurines, glass, knives, miniature, nautical, nouveau, porcelain, pottery, royal, souvenir, swords, teapot, thrift, tin, vase, victorian, vintage, ware, wood
apparel, beanie, camouflage, cardigan, clothes, clothing, coat, coats, denim, fleece, fur, gloves, hardy, hat, hats, hoodie, jacket, jackets, jeans, jersey, jumper, pants, plaid, polo, shirt, shirts, shorts, sleeved, socks, suit, suits, sweater, sweaters, sweatshirt, tee, tops, trousers, vest, wear, worn
ballerina, butterfly, coloring, daisy, dragon, fairies, fairy, jigsaw, kaleidoscope, mandala, mermaid, nativity, origami, peacock, princess, puzzle, rainbow, seashell, skeleton, snowflake, wonderland, zodiac
appetizing, crunchy, delectable, delicious, delish, hearty, savory, sweet, tasty, yum, yummy
albacore, anchovies, beansprouts, bouillabaisse, breaded, broiled, calamari, caviar, clams, codfish, confit, crabmeat, crawfish, crusted, curried, foie, gefilte, gras, haddock, linguine, mackerel, mahi, marinated, miso, monkfish, mussels, oysters, pickled, pilaf, portobello, prawns, prosciutto, ragu, rigatoni, risotto, salted, sardines, sashimi, scallops, seared, seaweed, shrimps, simmered, snapper, stewed, swordfish, truffle, wagyu, wasabi
ate, calorie, calories, carb, carbs, cook, cookbook, cooking, cooks, eating, food, foods, gluten, meal, meals, paleo, recipe, recipes, servings, snack, snacks, vegan, vegans, vegetarian, vegetarians, watchers
booze, champagne, cocktail, cocktails, drank, gin, libation, libations, liquor, mead, moonshine, rum, scotch, sip, sips, tequila, vodka, whiskey, whisky
gram, kilo, kilos, ounce, ounces, oz, pound, quarts
broiling, caramelize, drizzling, grilling, marinade, marinate, marinating, moisten, preheat, preheated, roasting, salting
beverage, beverages, carbonated, coke, cokes, cola, drink, drinks, fizzy, flavored, gum, juice, juices, kombucha, lemonade, milk, mints, popsicles, soda, sodas, straws, sugar, sugary, sweeten, sweetener, syrup

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE NANTES

Titre : Apprentissage multi-label extrême : comparaisons d'approches et nouvelles propositions.

Mots clés : multi-label extrême, réduction de dimension, arbre de décision, méthodes économes

Résumé : Stimulé par des applications comme l'annotation de documents ou d'images, l'apprentissage multi-label a connu un fort développement cette dernière décennie. Mais les algorithmes classiques se heurtent aux nouveaux volumes des données multi-label extrême (XML) où le nombre de labels peut atteindre le million. Cette thèse explore trois directions pour aborder la complexité en temps et en mémoire du problème : la réduction de dimension multi-label, les astuces d'optimisation et d'implémentation et le découpage arborescent. Elle propose d'unifier les approches de réduction à travers une typologie et deux formulations génériques et d'identifier des plus performantes avec une méta-analyse originale des résultats de la littérature. Une nouvelle approche est développée pour analyser l'apport du couplage entre le problème de réduction et celui de classification. Pour réduire la complexité mémoire en maintenant les capacités prédictives, nous proposons également un algorithme d'estimation des plus grands paramètres utiles d'un modèle classique de régression one-vs-rest qui suit une stratégie inspirée de l'analyse de données en flux. Enfin, nous présentons un nouvel algorithme CRAFTML qui apprend un ensemble d'arbres de décision diversifiés. Chaque arbre effectue une réduction aléatoire conjointe des espaces d'attributs et de labels et implémente un partitionnement récursif très rapide. CRAFTML est plus performant que les autres méthodes arborescentes XML et compétitif avec les meilleures méthodes qui nécessitent des supercalculateurs. Les apports de la thèse sont complétés par la présentation d'un outil logiciel VIPE développé avec Orange Labs pour l'analyse d'opinions multi-label.

Title : Extreme multi-label learning: comparisons of approaches and new proposals.

Keywords : extreme multi-label, dimensionality reduction, decision tree, efficient methods

Abstract : Stimulated by many applications such as documents or images annotation, multi-label learning have gained a strong interest during the last decade. But, standard algorithms cannot cope with the volumes of the recent extreme multi-label data (XML) where the number of labels can reach millions. This thesis explores three directions to address the complexity in time and memory of the problem: multi-label dimension reduction, optimization and implementation tricks, and tree-based methods. It proposes to unify the reduction approaches through a typology and two generic formulations and to identify the most efficient ones with an original meta-analysis of the results of the literature. A new approach is developed to analyze the interest of coupling the reduction problem and the classification problem. To reduce the memory complexity of a classical one-vs-rest regression model while maintaining its predictive performances, we also propose an algorithm for estimating the largest useful parameters that follows a strategy inspired by data stream analysis. Finally, we present a new algorithm called CRAFTML that learns an ensemble of diversified decision trees. Each tree performs a joint random reduction of the feature and the label spaces and implements a very fast recursive partitioning strategy. CRAFTML performs better than other XML tree-based methods and is competitive with the most accurate methods that require supercomputers. The contributions of the thesis are completed by the presentation of a software called VIPE that is developed with Orange Labs for multi-label opinion analysis.