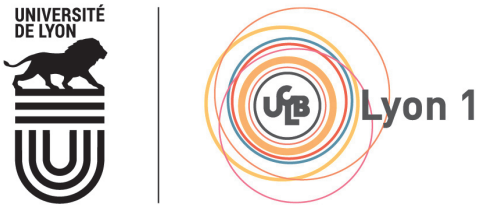
Apprentissage semi-supervisé pour la régression multi-labels : application à l'annotation automatique de pneumatiques

Vivien Kraus



N° d'ordre NNT : 2021LYSE1052

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

École Doctorale ED512
Informatique et Mathématiques (InfoMaths)

Spécialité de doctorat : Informatique

Soutenue publiquement le 12/03/2021, par :
Vivien KRAUS

Apprentissage semi-supervisé pour la régression multi-labels : Application à l'annotation automatique de pneumatiques.

Devant le jury composé de :

— Mme Marianne CLAUSEL, Professeur, Université de Lorraine, Rapporteur
— M. Engelbert MEPHU NGUIFO, Professeur, Université Clermont-Auvergne, Rapporteur
— M. Hamamache KHEDDOUCI, Professeur, Université Lyon 1, Examinateur
— Mme Pascale KUNTZ, Professeur, Université de Nantes, Examinatrice
— Mme Farida ZEHRAOUI, Maître de conférences, Université d'Évry, Examinatrice
— M. Khalid BENABDESLEM, MCF-HDR, Université Lyon 1, Directeur de thèse
— M. Bruno CANITIA, Responsable R&D, Lizeo IT, Invité

Résumé

Avec l'avènement et le développement rapide des technologies numériques, les données sont devenues à la fois un bien précieux et très abondant. Cependant, avec une telle profusion, se posent des questions relatives à la qualité et l'étiquetage de ces données. En effet, à cause de l'augmentation des volumes de données disponibles, alors que le coût de l'étiquetage par des experts humains reste très important, il est de plus en plus nécessaire de pouvoir renforcer l'apprentissage semi-supervisé grâce l'exploitation des données non-labellisées. Ce problème est d'autant plus marqué dans le cas de l'apprentissage multi-labels, et en particulier pour la régression, où chaque unité statistique est guidée par plusieurs cibles différentes, qui prennent la forme de scores numériques.

C'est dans ce cadre fondamental, que s'inscrit cette thèse. Tout d'abord, nous commençons par proposer une méthode d'apprentissage pour la régression semi-supervisée, que nous mettons à l'épreuve à travers une étude expérimentale détaillée. Grâce à cette nouvelle méthode, nous présentons une deuxième contribution, plus adaptée au contexte multi-labels. Nous montrons également son efficacité par une étude comparative, sur des jeux de données issues de la littérature. Par ailleurs, la dimensionnalité du problème demeure toujours la difficulté de l'apprentissage automatique, et sa réduction suscite l'intérêt de plusieurs chercheurs dans la communauté. Une des tâches majeures répondant à cette problématique est la sélection de variables, que nous proposons d'étudier ici dans un cadre complexe : semi-supervisé, multi-labels et pour la régression.

Mots-clés : apprentissage semi-supervisé, apprentissage multi-labels, régression, régularisation Laplacienne, sélection de variables, sélection de labels, optimisation.

Abstract

With the advent and rapid growth of digital technologies, data has become a precious asset as well as plentiful. However, with such an abundance come issues about data quality and labelling. Because of growing numbers of available data volumes, while human expert labelling is still important, it is more and more necessary to reinforce semi-supervised learning with the exploitation of unlabeled data. This problem is all the more noticeable in the multi-label learning framework, and in particular for regression, where each statistical unit is guided by many different targets, taking the form of numerical scores.

This thesis focuses on this fundamental framework. First, we begin by proposing a method for semi-supervised regression, that we challenge through a detailed experimental study. Thanks to this new method, we present a second contribution, more fitted to the multi-label framework. We also show its efficiency with a comparative study on literature data sets. Furthermore, the problem dimension is always a pain point of machine learning, and reducing it sparks the interest of many researchers. Feature selection is one of the major tasks addressing this problem, and we propose to study it here in a complex framework : for semi-supervised, multi-label regression.

Finally, an experimental validation is proposed on a real problem about automatic annotation of tires, to tackle the needs expressed by the industrial partner of this thesis.

Keywords : semi-supervised learning, multi-label learning, regression, Laplacian regularization, feature selection, label selection, optimization.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Chapitre 1

Introduction générale

1.1 Contexte et motivation

L'apprentissage automatique est une discipline de l'intelligence artificielle qui consiste à extraire à partir de données complexes, des connaissances utiles pour la prise de décision dans différents domaines d'application. Ces données peuvent êtres présentées sous différentes formes : observations, objets, transactions, signaux, graphes, textes, etc.

L'apprentissage automatique à partir des données vise à extraire des connaissances directement à partir des observations que l'on peut effectuer de l'environnement. Selon le cadre d'application, les observations peuvent être très différentes. Il peut s'agir de textes et d'images, par exemple, afin d'effectuer de la classification de documents textuels ou de la reconnaissance d'images.

Dans ce contexte, le principe général consiste à effectuer une modélisation statistique d'une fonction qui nous permet de résoudre le problème abordé. Cette modélisation est jugée pertinente si elle permet de généraliser la décision à de nouvelles formes de données inconnues au modèle, a priori.

Dans le cadre de cette thèse, la modélisation que nous cherchons à mettre en œuvre se présente sous la forme $y = f(x)$ où x représente les variables explicatives du problème, par exemple les mots employés dans un texte. y représente la variable cible à expliquer, appelée aussi cible ou label, c'est-à-dire celle que l'on cherche à expliquer : le ou les thèmes du document textuel.

Pour trouver cette fonction f, il y a plusieurs régimes d'application. Si la variable cible est complètement connue (pour toutes les observations), il s'agit d'apprentissage supervisé. Si elle est inconnue, il s'agit d'apprentissage non supervisé. Dans le cas qui nous intéresse, elle n'est que partiellement connue, ce qui définit l'apprentissage semi-supervisé.

Selon le type d'apprentissage envisagé, nous pouvons utiliser un certain nombre d'algorithmes, c'est-à-dire une fonction qui à partir des données construit une approximation du modèle f. Dans le cadre qui nous intéresse ici, l'algorithme consiste à minimiser une fonction de coût, en cherchant le modèle parmi une certaine classe de fonctions. La fonction de coût accumule donc les erreurs commises sur l'ensemble du jeu de données. En la

minimisant, l'objectif est d'obtenir le modèle le plus adapté à la distribution des données.

Par ailleurs, en apprentissage automatique, l'extraction de variables permet de synthétiser l'information contenue dans les caractéristiques du problème, de façon à en obtenir de nouvelles où chaque observation est décrite différemment. L'utilisation de ces nouvelles variables, souvent moins nombreuses, permet d'effectuer une réduction de dimension. D'autre part, la sélection de variables est un cas particulier, pour lequel chaque variable obtenue est une variable de description de l'observation, mais seules les variables pertinentes sont sélectionnées. La sélection de variables a donc un avantage considérable sur l'extraction de variables, puisqu'elle permet d'avoir un modèle plus interprétable. Par exemple, dans le cas de l'apprentissage à partir de données textuelles, elle permet de sélectionner le champ lexical du thème qui nous intéresse, ce qui est en soi un apport de connaissances.

La performance des algorithmes d'apprentissage en classification, régression ou sélection de variables, s'évalue sur les données n'ayant pas servi à la construction du modèle. Si la fonction résultante f est performante sur ces données, elle permettra de généraliser la connaissance, et ainsi conduire à une décision fiable. Principalement, on distingue deux obstacles qui s'opposent à la généralisation de la fonction de décision : le *sous-apprentissage*, pour lequel f ignore l'information utile des données, et le *sur-apprentissage*, qui donne trop d'importance au bruit dans les données. Le sous-apprentissage est assez simple à détecter, puisque la fonction f n'est pas performante sur les données connues.

Il est parfois possible de contrer l'effet du sur-apprentissage, en introduisant une régularisation explicite dans l'algorithme préconisé. Ainsi, celui-ci minimise l'erreur accumulée sur l'ensemble des données, mais tente également de minimiser la complexité du modèle sous-jacent. Une certaine définition de la complexité du modèle informe le choix de la régularisation employée.

Selon la nature de la variable cible, différents algorithmes peuvent être envisagés. Dans le cas de la régression, cette variable est de nature continue, contrairement à la classification dans laquelle elle est discrète. De plus, dans le cas d'un apprentissage multi-labels, l'espace de cette variable cible s'élargit et rejoint celui des variables explicatives dans toute sa complexité liée à la corrélation, au bruit et à la difficulté à modéliser la fonction de décision avec de multiples cibles, simultanément. C'est enfin, dans ce cadre complexe, que s'inscrit l'aspect fondamental de cette thèse.

Sur l'aspect applicatif, cette thèse s'inscrit dans le cadre d'une convention CIFRE entre le laboratoire LIRIS et la société Lizeo IT du groupe Lizeo [1]. En effet, dans le cadre de ses activités, l'entreprise récolte de nombreux documents textuels issus de multiples sources et décrivant les qualités des pneumatiques, à travers un certain nombre de caractéristiques étudiées pour les pneumatiques. Chacune de ces caractéristiques représente un score d'appréciation continu (sous forme d'une note) : ce sont donc des variables cible, réelles. La connaissance extraite de ces données d'appréciation sont d'une très grande im-

1. Cette thèse fait l'objet d'une "Convention Industrielle de Formation par la Recherche" (CIFRE) proposée par l'Agence Nationale de la Recherche Technique (ANRT). Ce mode de financement consiste en un partenariat entre une entreprise, ici Lizeo IT du groupe Lizeo et une université (ici, l'université Claude Bernard Lyon 1 , UCBL).

portance pour les manufacturiers et les distributeurs, mais l'annotation manuelle est très délicate, puisqu'elle requiert des connaissances vis-à-vis des produits, et coûteuse puisqu'elle doit s'effectuer sur plusieurs critères différents. Par conséquent, l'apprentissage doit s'inscrire dans le cadre semi-supervisé pour la régression multi-labels.

1.2 Contributions

Tout d'abord, nous avons commencé à aborder le problème de la régression dans le cadre mono-label. Pour ce faire, nous nous sommes fondés sur deux algorithmes représentatifs de l'état de l'art : *SSSL* (Simple algorithm for Semi-supervised Learning, [1]) et celui de la régularisation Laplacienne [2]. *SSSL* effectue une régression simple après un changement d'espace des données ; nous proposons de reprendre ce changement d'espace et d'apporter une régularisation Laplacienne à la régression, pour obtenir l'algorithme *Laplacian-regularized Simple Semi-Supervised Learning*, ou *LapS3L*.

Nous avons proposé par la suite une adaptation de *LapS3L* pour l'apprentissage multi-labels, *Laplacian-based Semi-supervised Multi-regression*, ou *LSMR*. L'idée principale de cette extension consiste à imposer, au travers d'une régularisation, que les labels similaires doivent être modélisés de manière similaire.

Nous avons également développé un algorithme de sélection de variables multi-labels et semi-supervisé, appelé *RSMS* pour *Robust Semi-supervised Multi-label feature Selection for regression*. En effet, dans le cas de l'apprentissage multi-labels, la sélection de variables propose de sélectionner les variables utiles pour tous les labels, simultanément. Cet algorithme de sélection de variables introduit l'idée que la sélection de labels permet de guider la sélection de variables.

Enfin, nous étions confrontés à un jeu de données textuel pour la résolution d'un problème réel, qui consiste à évaluer l'appréciation de diverses propriétés associées à des pneumatiques dans des commentaires d'internautes. C'est la raison pour laquelle nous avons développé un démonstrateur en situation permettant d'adapter nos algorithmes ainsi développés aux contingences de cette application à forte valeur ajoutée pour l'entreprise.

1.3 Organisation du manuscrit

L'organisation de ce manuscrit suit l'ordre chronologique de nos propositions. Dans le chapitre qui suit, nous dresserons un état de l'art sur l'apprentissage semi-supervisé et multi-labels, en s'intéressant également à la sélection de variables. Ensuite, nous présenterons *LapS3L* (*Laplacian-regularised Simple Semi-Supervised Learning*), en décrivant les travaux utilisés, l'approche proposée, l'algorithme d'optimisation retenu et l'étude expérimentale mise en œuvre. Le chapitre 4 se consacrera à la deuxième contribution, *LSMR* (*Laplacian-based semi-supervised multi-regression*) qui est son extension dans le cadre multi-labels. Le chapitre 5 présentera *RSMS* (*Robust Semi-supervised Multi-label feature Selection for regression*), pour répondre à la problématique de la sélection de variables, dans le cadre semi-supervisé et multi-labels. Finalement, le chapitre 6 présentera

l'application de nos travaux sur un jeu de données réel, pour répondre à la problématique de l'entreprise sur l'annotation automatique à partir de grandes masses de données sur les pneumatiques. Enfin, nous conclurons ce rapport dans le dernier chapitre avec un bilan sur les contributions proposées et les résultats obtenus via les différentes expérimentations menées durant cette thèse. Nous discuterons aussi quelques propositions d'amélioration en guise de perspectives à court et à moyen termes.

1.4 Notations

Les notations utilisées dans la suite de ce document sont rappelées ici.

Voici les opérateurs employés :

— M' désigne la transposée de lamatrice M ;
— $\bar{v}$ désigne la moyenne d'un vecteur v ;
— $M \otimes P$ désigne le produit de Kronecker de M par P ;
— $M \succeq 0$ ou $M \in S_N^+(\mathbb{R})$ signifie que M est une matrice symétrique réelle semi-définie positive.

Les notations suivantes sont employées pour la dimension du problème d'apprentissage :

— n_l : le nombre d'individus labellisés de l'ensemble d'apprentissage ;
— N : le nombre total d'individus, y compris ceux non labellisés ;
— n_t : le nombre d'individus de test ;
— d : la dimension de l'espace des individus, ou le nombre de variables ;
— m : le nombre de labels ;
— c : le nombre de classes pour un classifieur ;
— o : le nombre de pseudo-labels ;
— s : le nombre de valeurs propres considérées, par ordre croissant.

Nous utilisons ces notations pour écrire une fonction objectif :

— $\alpha > 0$, $\beta > 0$, $\gamma > 0$, $\delta > 0$: régulariseurs ;
— $b \in \mathbb{R}^m$: le vecteur de biais pour l'apprentissage de régression ;
— $B \in \mathbb{R}^{o,m}$: le facteur à droite dans la décomposition du modèle ;
— Γ : les coefficients du modèle SSSL ;
— D, D_s : la matrice de degré du graphe des individus (l'indice s sert à la démarquer de D_m) ;
— D_m : la matrice de degré du graphe des labels ;
— D_W, D_B : matrices diagonales pour la relaxation lisse de la norme $l_{2,1}$ et $l_{1,2}$;
— $\epsilon > 0$: paramètre de la fonction de coût de l'algorithme SVM (*Support Vector Machine*), ou notation pour un nombre strictement positif pouvant être choisi arbitrairement proche de 0 ;
— $J \in \mathbb{R}^{N,N}$: matrice diagonale indicatrice des individus labellisés ;
— $L \in \mathbb{R}^{N,N}$, L_s : la matrice Laplacienne du graphe des individus ;
— $L_m \in \mathbb{R}^{m,m}$: la matrice Laplacienne du graphe des labels ;
— $M \in \mathbb{R}^{N,N}$: la matrice d'adjacence du graphe des individus ;
— $M_m \in \mathbb{R}^{m,m}$: la matrice d'adjacence du graphe des labels ;

— P, Q : deux matrices décomposant un modèle W, $W = P + Q$ ou $W = PQ$;
— $\mathcal{R}$: lorsqu'une régularisation quelconque s'applique sur un modèle W, nous la noterons $\mathcal{R}(W)$;
— tr : fonction calculant la trace d'une matrice (la somme des valeurs de sa diagonale) ;
— $V \in \mathbb{R}^{N,o}$: une matrice de pseudo-labels ;
— $V_l \in \mathbb{R}^{n_l,o}$: les lignes de V correspondant aux individus labellisés ;
— $\mathcal{V}$: les individus du jeu d'apprentissage pour SSSL ;
— $\mathcal{V}_t$: le jeu de test pour SSSL ;
— $w \in \mathbb{R}^d$: le modèle, sous forme de vecteur de coefficients, pour un apprentissage mono-label ;
— $W \in \mathbb{R}^{d,m}$: le modèle pour un apprentissage multi-labels ;
— $y \in \mathbb{R}^{n_l}$: le vecteur de labels (pour de la régression mono-label) ;
— $Y \in \mathbb{R}^{n,m}$: la matrice de labels, de dimension $N \times m$. Elle contient des lignes de valeur non spécifiée pour les individus non labellisés. De dimension $n_l \times m$, il ne s'agit que des individus labellisés ;
— $\hat{Y} \in \mathbb{R}^{n_t,m}$: la sortie du modèle pour la prédiction des labels d'un ensemble de n_t individus ;
— ξ, ξ^* : paramètres de marge pour l'algorithme SVR (*Support Vector Regression*) ;
— $X \in \mathbb{R}^{N,d}$ la matrice de données servant pour l'apprentissage ;
— $X_l \in \mathbb{R}^{n_l,d}$ les lignes de X correspondant aux individus labellisés.

Certains algorithmes utilisent un noyau. Les notations associées sont :

— $\mathcal{H}_\kappa$: espace de Hilbert à noyau reproduisant ;
— $K \in \mathbb{R}^{N,N}$: application de la fonction de noyau à chaque paire d'individus ;
— $K_b \in \mathbb{R}^{N,n_t}$: application de la fonction de noyau à chaque paire (individu d'apprentissage, individu de test) ;
— $\kappa \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, une fonction de noyau ;
— $\sigma > 0$: l'hyperparamètre de la RBF (/Radial Basis Function/) ;
— $U \in \mathbb{R}^{N,s}$: les vecteurs propres de la matrice de noyau K.

Pour décrire un algorithme, nous utilisons les notations suivantes :

— $M^{(0)}$ désigne une valeur initiale avant itérations ;
— $\mathbf{1}$: un vecteur dont toutes les composantes sont égales à 1 ;
— $C \in \mathbb{R}^{m,o}$: l'indicatrice des clusters, vaut 1 si m est dans le cluster o ;
— C_W, C_V, C_B : la constante de Lipschitz de la fonction de gradient vis-à-vis de W, V et B ;
— $\mathcal{C}$: un cluster ;
— f, ou h : une fonction de prédiction, prend en entrée un individu ou un ensemble d'individus et retourne les valeurs de tous les labels pour ces individus ;
— i : indice d'itération d'individu ;
— I : la matrice identité ;
— j : indice d'itération de variables ;
— k : indice d'itération de labels, ou nombre de plus proches voisins ;
— l : indice d'itération de pseudo-labels ;

— $\hat{L}_N$: l'opérateur d'évaluation d'une fonction de prédiction ;
— $\left(\hat{\lambda}\right)_{i=1}^{N}$ les valeurs propres de $\hat{L}_N$;
— min, max : minimum et maximum d'un ensemble de valeurs ;
— $\mathcal{O}$: notation pour la complexité temporelle ou spatiale ;
— p : indice de la norme de Minkowski ;
— $(\phi)_{i=1}^{N}$ les fonctions propres de $\hat{L}_N$;
— $\rho(M)$ désigne le rayon spectral de M, c'est-à-dire sa plus grande valeur propre ;
— $\mathcal{S} \subset \{1, ..., m\}$ les labels sélectionnés ;
— $\sigma_i(W)$: la $i^{\text{ième}}$ valeur propre de W ;
— $x_i, \quad i = \{1...N\}$: un individu de l'ensemble d'apprentissage ;
— $x_{li}, \quad i = \{1...n_l\}$: un individu labellisé de l'ensemble d'apprentissage ;
— $\hat{y}_i$ désigne la prédiction mono-label d'un individu de test i ;
— $\hat{Y}_{i,k}$ désigne la prédiction d'un individu de test i pour un label k ;
— z : le vecteur de label pour SSSL ;
— $\hat{z}$: la prédiction pour SSSL.

Enfin, $\mathcal{X}$ désigne l'ensemble d'apprentissage ; dans tous les cas étudiés il s'agit d'un espace réel de dimension d.

Nous rappelons également les normes utilisées :

— la norme de Minkowski, pour un indice p : $\|x_i\|_p = (\sum_{i=1}^{n} |x|^p)^{\frac{1}{p}}$;
— la norme l_2, cas particulier pour $p = 2$: $\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$;
— la norme l_1, autre cas particulier pour $p = 1$: $\|x\|_1 = \sum_{i=1}^{n} |x_i|$;
— la norme matricielle $l_{p,q}$ pour deux indices de Minkowski, est la norme q du vecteur constitué des normes p de chacune des lignes de la matrice ;
— la norme $l_{2,1}$ s'écrit donc pour une matrice $W \in \mathbb{R}^{d,m}$: $\|W\|_{2,1} = \sum_{j=1}^{d} \|W_{j,.}\|_2$;
— la norme $l_{1,1}$ s'écrit : $\|W\|_{1,1} = \sum_{j=1}^{d} \|W_{j,.}\|_1$;
— la norme $l_{1,2}$ est la norme $l_{2,1}$ de la transposée ;
— la norme $l_{\infty,1}$ s'écrit : $\|W\|_{\infty,1} = \sum_{j=1}^{d} \max_k |W_{j,k}|$;
— la norme de Frobenius $l_{2,2}$ s'écrit : $\|W\|_F = \sqrt{\sum_{j=1}^{d} \sum_{k=1}^{m} W_{j,k}^2}$;
— la norme $l_{2,1-2}$ est la différence entre la norme $l_{2,1}$ et la norme de Frobenius ;
— la norme trace s'écrit : $\|W\|_* = \sqrt{tr\,(W'W)} = \sum_{i=1}^{m} \sigma_i(W)$, si $\sigma_i(W)$ est la $i^{\text{ième}}$ valeur propre de W.

Chapitre 2

État de l'art : Régression semi-supervisée multi-labels

Dans ce chapitre, nous dressons un état de l'art de l'apprentissage de régression semi-supervisée et multi-labels.

Pour l'apprentissage semi-supervisé, nous détaillons diverses approches qui peuvent utiliser les données non labellisées. Nous commençons par les approches de pénalisation utilisant la régularisation Laplacienne, puis son adaptation dans le cadre de l'algorithme SVM, ainsi que les méta-méthodes semi-supervisées : *self-training* et *co-training*. Nous nous intéressons également à la sélection de variables pour l'apprentissage semi-supervisé.

Nous nous concentrons ensuite sur la régression multi-labels. On peut également traiter le problème de régression multi-labels grâce à des méta-méthodes, mais aussi des régularisations spéciales conçues pour l'apprentissage multi-labels. Nous terminons ce chapitre par un exposé de la sélection de variables multi-labels.

2.1 Régression semi-supervisée

Dans cette section, nous nous intéressons à la notion de régression semi-supervisée. L'apprentissage semi-supervisé est un sujet de recherche important. Le but est de tirer profit de données non labellisées en plus des données labellisées, dans le cas relativement fréquent où labelliser des données est très coûteux, car ce processus nécessite la collaboration d'un expert, alors que dans le même temps de grandes masses de données non labellisées sont disponibles.

Les problèmes de régression peuvent aussi être traités dans ce contexte d'apprentissage semi-supervisé. Ce cas dans lequel la variable cible est continue suscite un intérêt différent du cas, beaucoup plus fréquent, où la variable cible est discrète (classification binaire ou multiple). Ainsi, de nombreuses approches en apprentissage semi-supervisé sont d'abord pensées pour une tâche de classification, ce qui parfois rend leur adaptation plus difficile.

2.1.1 Apprentissage semi-supervisé

L'objectif de l'apprentissage semi-supervisé est d'exploiter les données non labellisées, disponibles en abondance, pour améliorer les performances en généralisation de l'algorithme. Par exemple, prenons le cas du jeu de données 2-lunes (*two moons*, figure 2.1). Dans ce jeu de données (en haut), il n'y a que deux points labellisés. En ignorant les points non labellisés, la classification donnerait le résultat en bas à gauche, alors que le résultat en bas à droite semble plus naturel.

Ce problème de classification binaire semi-supervisée est assez inintéressant au premier abord, puisqu'il ne propose qu'un seul point par classe. Dans ce cas, n'importe quel algorithme de classification binaire peut obtenir une erreur résiduelle nulle en traçant une zone de séparation entre les deux points.

Cependant, le résultat n'est pas très intuitif. On s'attendrait à ce que chacune des deux branches du jeu de données utilise la même étiquette de manière cohérente, en utilisant la justification suivante : dans les zones de points à haute densité, toutes les étiquettes devraient être les mêmes.

Si le résultat de classification est différent, on ne peut cependant pas affirmer qu'il sera meilleur. Cependant, comme nous le verrons plus tard, certaines connaissances a priori de cette nature peuvent donner de meilleurs résultats expérimentaux.

Selon [3], il est nécessaire de satisfaire plusieurs hypothèses pour améliorer la performance de l'apprentissage, rappelées ici.

L'hypothèse de *régularité semi-supervisée* affirme que la prédiction doit être plus lisse dans les zones à haute densité que dans les zones à basse densité. Dans notre exemple ci-dessus, les deux lunes sont des zones à haute densité ; la prédiction ne doit donc pas varier rapidement dans cette zone. En revanche, entre les deux formes lunaires se trouve une zone à faible densité, où la prédiction peut très abruptement basculer entre négatif et positif.

L'hypothèse de *clusterisation* affirme que si les données se divisent en groupes, alors la prédiction doit être la même sur tout le groupe. C'est principalement cette hypothèse

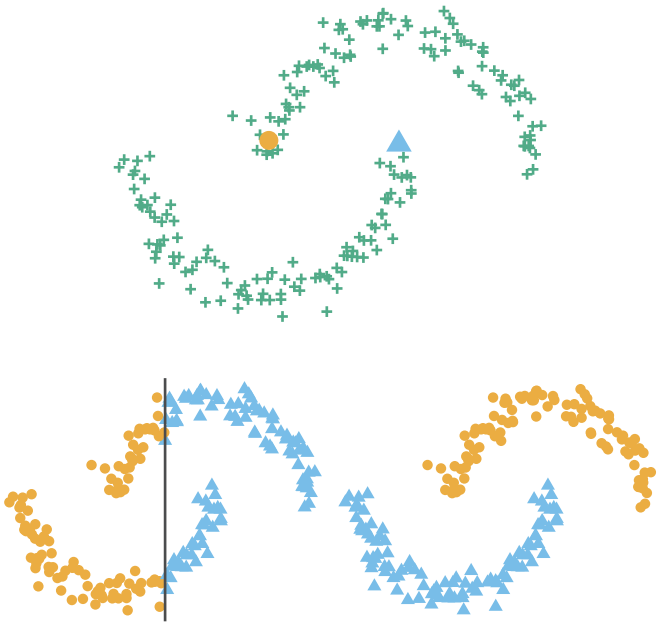


FIGURE 2.1 – Le jeu de données *two moons*

que nous aurions utilisée pour définir le résultat attendu de la figure 2.1. Ce n'est pas exactement la même chose que l'hypothèse de régularité. En effet, elle n'exprime rien quant à la zone de séparation, et les regroupements ne sont pas nécessairement les zones à plus haute densité.

Finalement, l'hypothèse de *manifold* affirme que les données vivent en réalité dans un espace de plus faible dimension. C'est en effet le cas des données de la figure : on pourrait paramétrer cet espace par un réel unique, qui indiquerait sur quelle lune on se trouve et à quelle position.

L'intérêt d'un algorithme de régression semi-supervisée réside donc dans le fait qu'il soit capable de tirer avantage de ces hypothèses, au moyen par exemple de connaissances a priori injectées dans le modèle. Dans la suite, nous allons voir quelques exemples d'algorithmes qui utilisent ces hypothèses.

Notations

Pour garder une certaine cohérence dans la liste des méthodes évoquées, nous définissons ici les notations employées pour modéliser un problème d'apprentissage semi-supervisé.

Notations générales Le problème d'apprentissage semi-supervisé consiste à apprendre une fonction de prédiction, f, à partir des données. Ces données sont modélisées comme

des points $(x_i)_{i=1}^N$ dans un espace $\mathcal{X}$. Dans le cas des problèmes de régression, on considèrera uniquement que les individus $(x_i)_{i=1}^N$ sont des vecteurs dans un espace $\mathbb{R}^d$, où d est la dimension de l'espace. La fonction de prédiction retourne un nombre réel, dans le cadre de la régression. On a donc :

$$f \colon \mathcal{X} \to \mathbb{R}$$

Parmi cet ensemble d'apprentissage semi-supervisé, on peut en extraire un sous-ensemble supervisé constitué de paires $(x_{li}, y_i)_{i=1}^{n_l} \in \mathbb{R}^d \times \mathbb{R}$. En pratique, l'ensemble supervisé constitue généralement les n_l premiers individus de l'ensemble total.

Dans le cas linéaire, la fonction f est paramétrée par un modèle noté $w \in \mathbb{R}^d$, et optionnellement par un biais $b \in \mathbb{R}$. Le problème d'optimisation consiste donc à trouver la valeur de w (et b) minimisant le coût. La fonction f s'écrit alors :

$$x \mapsto \sum_{j=1}^{d} x_j w_j + b$$

Dans le cas d'un apprentissage à noyau, on définit une fonction

$$\kappa \colon \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$$

qui agit sur une paire d'individus. La fonction f dépend donc de l'ensemble d'apprentissage $(x_i)_{i=1}^N$, et elle adopte la forme suivante :

$$x \mapsto \sum_{i=1}^{N} \kappa(x_i, x) w_i + b$$

Notations matricielles Pour la notation matricielle, on note $X \in \mathbb{R}^{N,d}$ la matrice de données, de sorte que pour tout individu d'apprentissage $i \in \{1, ..., N\}$, $X_{i,\cdot} = x_i$. De la même façon, on introduit $X_l \in \mathbb{R}^{n_l,d}$ et $y \in \mathbb{R}_l^n$ pour l'ensemble labellisé des données. Dans certains cas, les opérations sur y sont plus communément décrites en utilisant une notation matricielle, c'est-à-dire en considérant la matrice-colonne $Y \in \mathbb{R}^{n_l,1}$ telle que pour tout $1 \leq i \leq n_l$, $Y_{i,1} = y_i$. On peut reprendre cette notation pour la valeur de la prédiction : pour un certain ensemble d'apprentissage, comprenant n_t individus, on obtient une matrice $\hat{Y} \in \mathbb{R}^{n_t,1}$.

2.1.2 Approches de régression semi-supervisée

Nous allons présenter maintenant les approches les plus importantes pour l'apprentissage semi-supervisé en général, et plus particulièrement la régression semi-supervisée. En effet, l'apprentissage semi-supervisé concentre principalement des travaux de classification. Certains peuvent être directement adaptés au cadre de la régression, même si d'autres utilisent des normalisations ou des régularisations qui ne font sens que pour un problème de classification. Enfin, certains sont spécifiques à la régression.

Régression Laplacienne

Cette famille d'approches est très employée pour les problèmes d'apprentissage semi-supervisé, et permet de traiter indifféremment des problèmes de régression et des problèmes de classification. De plus, la simplicité de l'idée qu'elle développe lui procure une notoriété supplémentaire.

Champs gaussiens et fonctions harmoniques L'idée développée par cette approche se résume en deux points [4]) :

1. La prédiction sur l'ensemble labellisé doit correspondre aux valeurs de labels de l'apprentissage ;
2. Si deux individus sont proches (dans un sens à définir), alors la prédiction pour ces deux individus doit être similaire.

Tout d'abord, il faut indiquer dans quel sens deux individus sont proches. À partir de la matrice de données X, on construit un graphe, où chaque individu correspond à un nœud, et dans lequel deux individus sont reliés si la distance est suffisamment faible (ou si la similitude est suffisamment élevée). Dans la pratique, lorsque l'on trouve une approche qui mobilise un graphe des individus, la similarité retenue est presque toujours donnée au travers d'un noyau gaussien, aussi appelé *radial basis function* (*RBF*, [4], [5], [6], [7], par exemple) : si deux individus i et j ont pour caractéristiques x_i et x_j, la similarité $M_{ij} \in \mathbb{R}$ s'exprime par :

$$M_{ij} = \exp\left(\frac{-\|x_i - x_j\|_2^2}{2\sigma^2}\right) \tag{2.1}$$

Ce qui introduit un hyperparamètre à fixer a priori, $2\sigma^2 > 0$, qui est aussi parfois noté γ, ou $4t$ ([2]). La matrice M ainsi créée est souvent appelée W pour désigner les poids du graphe.

Dans les cas où les données sont en grande dimension, par exemple pour la classification de textes ([8]), la RBF ne peut pas s'appliquer directement, à cause de sa dépendance à la norme l_2, qui peut difficilement distinguer des distances entre individus en grande dimension. Par conséquent, on peut la remplacer par la similarité cosinus :

$$M_{ij} = \frac{\langle x_i, x_j \rangle}{\|x_i\| \, \|x_j\|} \tag{2.2}$$

Cette matrice M est généralement rendue parcimonieuse (*sparse*), pour faciliter les calculs quand le nombre d'individus est très élevé. En ce sens, on affecte souvent à 0 la valeur de M_{ij} si i et j sont trop éloignés, c'est-à-dire si x_i n'est pas un des k plus proches voisins de x_j et réciproquement x_j n'est pas non plus l'un des k plus proches voisins de x_i. (par exemple, [9] ou [10]).

Dans tous les cas, on s'assure que la matrice M soit symétrique.

Minimiser l'écart de prédiction entre deux individus proches revient en fait à minimiser :

$$\sum_{i_1=1}^{N}\sum_{i_2=1}^{N} M_{i_1 i_2} \left\| \hat{y}_{i_1} - \hat{y}_{i_2} \right\|_2^2 \tag{2.3}$$

En développant le carré, on obtient :

$$\sum_{i_1=1}^{N}\sum_{i_2=1}^{N} M_{i_1 i_2} \hat{y}_{i_1}^2 + \sum_{i_1=1}^{N}\sum_{i_2=1}^{N} M_{i_1 i_2} \hat{y}_{i_2}^2 - 2\sum_{i_1=1}^{N}\sum_{i_2=1}^{N} M_{i_1 i_2} \hat{y}_{i_1} \hat{y}_{i_2} \tag{2.4}$$

En utilisant le fait que la matrice M est symétrique, on peut le ramener à :

$$2\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij} \hat{y}_i^2 - 2\sum_{i_1=1}^{N}\sum_{i_2=1}^{N} M_{i_1 i_2} \hat{y}_{i_1} \hat{y}_{i_2} \tag{2.5}$$

qui se simplifie en :

$$2\sum_{i=1}^{N} \left(\sum_{j=1}^{N} M_{ij} \right) \hat{y}_i^2 - 2tr\left(\hat{Y}' M \hat{Y} \right) \tag{2.6}$$

Ceci fait apparaître la somme de chaque ligne de M, que l'on écrit dans une matrice diagonale D (de « degré », c'est-à-dire contenant sur la diagonale le degré de chaque nœud du graphe) :

$$D = \mathrm{diag}(M\mathbf{1})$$

Le problème revient donc à minimiser :

$$tr\left(\hat{Y}'(D - M)\hat{Y} \right) \tag{2.7}$$

ce qui fait apparaître tout naturellement la matrice Laplacienne du graphe, $L = D - M$. En rajoutant la contrainte concernant la prédiction sur l'ensemble labellisé, on obtient le problème :

$$\begin{array}{ll} \underset{\hat{Y} \in \mathbb{R}^{n,1}}{\text{minimize}} & tr\left(\hat{Y}' L \hat{Y} \right) \\ \text{subject to} & \forall 1 \leq i \leq n, \quad \hat{Y}_i = Y_i \end{array} \tag{2.8}$$

Ce problème d'optimisation est convexe, à condition que la matrice L soit semi-définie positive. Il est généralement employé pour des problèmes de classification multi-classes ([7]), ce qui demande une autre forme de normalisation pour la matrice L, de façon à avoir des prédictions positives et de somme 1. Pour des problèmes de régression, nous utiliserons principalement la normalisation la plus simple ($L = D - M$, [11]).

Ce problème fait partie de la famille de problèmes *transductifs*. Un problème d'apprentissage transductif permet d'obtenir une prédiction pour les individus non labellisés de l'ensemble d'apprentissage, cependant il est impossible de généraliser à de nouveaux cas.

Régularisation Laplacienne LapRLS Le problème précédent d'apprentissage transductif peut se traduire en problème d'apprentissage inductif semi-supervisé. Résoudre un tel problème permet d'obtenir un modèle qui peut être utilisé pour obtenir une prédiction à partir de n'importe quel point de la distribution des données.

L'algorithme LapRLS ([2]) peut être interprété comme une extension de l'algorithme précédent. Avec les notations suivantes :

— $X \in \mathbb{R}^{N,d}$ désigne la matrice de données ;
— $X_l \in \mathbb{R}^{n,d}$ désigne le sous-ensemble labellisé de cette matrice ;
— $W \in \mathbb{R}^{d,1}$ désigne le modèle ;

la prédiction s'écrit tout simplement :

$$f \colon X \mapsto \hat{Y} = XW \tag{2.9}$$

Si l'on se concentre sur le problème de régression, on peut réécrire l'équation 2.8 en :

$$\begin{array}{ll} \underset{W \in \mathbb{R}^{d,1}}{\text{minimize}} & \text{tr}\left((XW)'L(XW)\right) \\ \text{subject to} & \forall 1 \leq i \leq n (X_l W)_i = Y_i \end{array} \tag{2.10}$$

Cette formulation n'est pas complète, car il n'est pas possible de résoudre la contrainte. Cependant, on peut utiliser une régression à la place, aux moindres carrés (*Least Squares*) : en introduisant un régulariseur $\beta > 0$, le problème se transforme donc en :

$$\underset{W \in \mathbb{R}^{d,1}}{\text{minimize}} \quad \|X_l W - Y\|_F^2 + \beta \text{tr}\left((XW)'L(XW)\right) \tag{2.11}$$

Dans les cas où il est question d'apprentissage avec des données à haute dimension, c'est-à-dire où $d > N$, la régression aux moindres carrés ne fonctionne pas correctement. En effet, le modèle contient trop de paramètres, qui ne peuvent pas être déterminés par l'algorithme. Pour pallier ce problème, on introduit généralement un terme de régularisation supplémentaire, dit régularisation Ridge ou Tikhonov : en introduisant un hyperparamètre supplémentaire, $\alpha > 0$, on obtient :

$$\underset{W \in \mathbb{R}^{d,1}}{\text{minimize}} \quad \|X_l W - Y\|_F^2 + \alpha \|W\|_F^2 + \beta \text{tr}\left((XW)'L(XW)\right) \tag{2.12}$$

Ces deux formulations (2.11 et 2.12) sont la base de nombreux algorithmes d'apprentissage semi-supervisé. Le problème 2.12 peut notamment être résolu avec une solution analytique :

$$W = [(X_l' X_l + \alpha I + \beta X' L X]^{-1} X_l' Y \tag{2.13}$$

Cette résolution de système linéaire nécessite un temps de calcul en $\mathcal{O}(N^2 d^3)$, ce qui est considérable. Pour les approches effectuant une descente de gradient, le gradient se calcule en :

$$\nabla_W \mathcal{L} = 2X_l'[X_l W - Y] + 2\alpha W + 2\beta X' L X W \tag{2.14}$$

Dans les cas où la dimension d est plus grande que le nombre d'individus N, le théorème de représentation ([2]) nous permet d'introduire une fonction à noyau $\kappa\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ et de ne considérer comme variables que les interactions entre un individu et le jeu d'apprentissage. On applique donc la fonction κ entre tous les individus de l'ensemble d'apprentissage (à gauche et à droite), ce qui donne une matrice $K \in \mathbb{R}^{N,N}$. La solution fait intervenir une inversion de la matrice K, ce qui n'est pas recommandable du fait de sa dimension $(N \times N)$. On voit rarement cette approche appliquée dans les travaux récents, puisque selon l'hypothèse de l'apprentissage semi-supervisé, on dispose d'un grand nombre d'individus non labellisés.

Algorithme LapSVM

L'algorithme SVR est une adaptation de SVM pour la régression [12]. Le SVM a été lui-même adapté au cadre semi-supervisé avec le *SVM transductif*, d'une part, et avec une régression Laplacienne d'autre part ([13], [14]). Cependant, la régularisation Laplacienne peut aussi s'utiliser avec les SVM en régression ([15]).

Dans ce cas, on utilise la fonction de coût ϵ-insensible, où ϵ est un hyperparamètre, en combinaison de la matrice Laplacienne d'un graphe que l'on aura calculé de la même façon qu'à la section précédente.

On peut donc définir le problème d'apprentissage du LapSVM en introduisant une marge supérieure ξ et une marge inférieure ξ^* pour chaque point labellisé, et en pénalisant les modèles qui donnent une prédiction hors de ces marges.

$$\begin{array}{ll} \underset{W \in \mathbb{R}^{d,1}, \xi, \xi^*}{\text{minimize}} & \|W\|_F^2 + \alpha \sum_{i=1}^{n_l} (\xi_i + \xi_i^*) + \beta \text{tr}\left(W' X' L X W\right) \\ \text{subject to} & \forall 1 \leq i \leq n_l, \quad X_i W \leq Y_i + \epsilon + \xi_i, \\ & \forall 1 \leq i \leq n_l, \quad X_i W \geq Y_i - \epsilon - \xi_i^*, \\ & \forall 1 \leq i \leq n_l, \quad \xi_i \geq 0 \wedge \xi_i^* \geq 0 \end{array} \tag{2.15}$$

On constate que la modification du SVM pour la régression apportée pour inclure cet algorithme dans le cadre semi-supervisé revient exactement à ajouter un terme de régularisation Laplacienne.

Par ailleurs, une modification, nommée LapESVR ([15]), introduit une nouvelle variable, V, qui fait office *d'embedding* de labels intermédiaires, d'où le nom de l'algorithme. Concrètement, on introduit un nouvel hyperparamètre γ, et on cherche à résoudre simultanément les contraintes suivantes :

1. Les valeurs du label Y sont bien reconstruites par V sur l'ensemble labellisé ;
2. Si deux individus sont proches, la valeur des lignes de V doit aussi être proche ;
3. Le modèle permet de bien reconstruire V avec la fonction de coût ϵ-insensible.

Il s'agit donc de résoudre le problème suivant :

$$\begin{aligned}
\underset{W \in \mathbb{R}^{d,1}, \xi, \xi^*, V}{\text{minimize}} \quad & \|W\|_F^2 + \alpha \sum_{i=1}^{n} (\xi_i + \xi_i^*) + \beta \text{tr}\left(W' X' L X W\right) \\
& + \gamma \text{tr}\left((V - Y)' J (V - Y)\right) \\
\text{subject to} \quad & \forall 1 \leq i \leq n, \quad X_i W \leq V_i + \epsilon + \xi_i, \\
& \forall 1 \leq i \leq n, \quad X_i W \geq V_i - \epsilon - \xi_i^*, \\
& \forall 1 \leq i \leq n, \quad \xi_i \geq 0 \wedge \xi_i^* \geq 0
\end{aligned} \tag{2.16}$$

Avec la matrice diagonale J qui fait office d'indicatrice des individus labellisés :
— hors de la diagonale, elle vaut 0 : $\forall i \neq j, J_{ij} = 0$
— sur la diagonale, elle indique les n premiers individus : $\forall i, J_{ii} = [i \leq n]$

Méta-algorithmes de régression semi-supervisée

Parmi les approches d'apprentissage semi-supervisé, on trouve aussi des méta-méthodes qui fonctionnent à partir d'un apprentissage supervisé.

Self-training Le *self-training* est un algorithme d'apprentissage semi-supervisé relativement simple [16]. Étant donné un régresseur de base, $h \colon \mathcal{X} \to \mathbb{R}$, on obtient un modèle semi-supervisé en itérant l'algorithme suivant :

1. Faire un apprentissage de h sur l'ensemble labellisé (X_l, Y) ;
2. Obtenir une prédiction sur la partie non supervisée : $\hat{Y} \leftarrow h(X)$;
3. Sélectionner les indices des prédictions pour lesquelles la confiance est la plus grande, i. Si la confiance est trop faible pour tous les points non labellisés, arrêter l'algorithme ;
4. Transférer $X_{i.}, \hat{Y}_{i.}$ depuis l'ensemble non labellisé vers l'ensemble labellisé.

Cet algorithme n'est pas complètement spécifié, il reste à définir comment l'étape 3 est implémentée. Pour certains classifieurs, comme la régression logistique [17] ou le SVM, il est possible d'utiliser les scores d'appartenance à une classe.

Dans le cas d'un apprentissage pour la classification, on peut également utiliser une méthode ensembliste pour remplacer h. Dans ce cas, h est composé d'un ensemble $(h_i)_{i=1}^c$ de classifieurs, ce qui permet de définir pour un point de l'ensemble non-labellisé x la prédiction comme étant la classe majoritaire de l'ensemble $\{h_i(x)\}_{i=1}^c$. En ce qui concerne la confiance dans la prédiction, on peut considérer l'entropie.

Dans le cas de la régression, on ne peut pas utiliser d'approche fournissant directement un score de confiance. On peut cependant utiliser des méthodes ensemblistes, similaires au cas de classification [18]. Dans ce cas, un ensemble de régresseurs est utilisé à la place de h. Chaque régresseur donne une prédiction pour chaque individu non labellisé. La prédiction de h correspond à la moyenne des prédictions de chacun des régresseurs, et la confiance est calculée à partir de la variance des prédictions des régresseurs. Lorsque la variance augmente, la confiance diminue.

Cette mesure de confiance est particulièrement intéressante parce qu'il est possible d'utiliser n'importe quel régresseur pour les $(h_i)_i$. Par conséquent, cette méthode permet de construire un algorithme d'apprentissage semi-supervisé ensembliste à partir d'un algorithme de régression supervisée *quelconque*.

Les auteurs dans [18] définissent aussi une autre mesure de confiance, qui ne s'applique que dans le cas d'une forêt aléatoire. Dans ce cas, il est possible de suivre le parcours d'un individu à travers chacun des arbres. Si deux individus aboutissent à la même feuille pour un certain nombre d'arbres, on peut considérer qu'ils sont voisins. Étant donné que chaque régresseur de l'ensemble a été entraîné à partir d'un jeu d'apprentissage différent, sélectionné par échantillonnage avec remise, pour chaque régresseur, il existe des points de l'ensemble labellisé qui n'ont en fait pas servi pour l'entraîner : ce sont les exemples « hors du lot » d'apprentissage (*Out of bag*). Pour évaluer la confiance d'un point de l'ensemble d'apprentissage, on peut sélectionner les individus de l'ensemble d'apprentissage proches, et pour chacun de ces voisins labellisés, évaluer l'erreur de régression des régresseurs qui n'ont pas utilisé ce point pour l'entraînement. On obtient donc une estimation de l'erreur pour chacun des points de l'ensemble d'apprentissage non labellisé, qui est l'opposé de la confiance accordée à la prédiction pour ce point.

Il est nécessaire de définir un critère d'arrêt pour l'algorithme. En effet, si celui-ci se poursuit jusqu'à épuisement du jeu d'apprentissage non-labellisé, il est possible que la performance se dégrade. On choisit donc généralement un seuil de confiance, en-dessous duquel on ne considèrera pas un point de l'ensemble non labellisé. On peut également conserver un ensemble de validation sur lequel évaluer chaque itération de l'algorithme, et l'arrêter en cas de dégradation de performance. [18] propose ansi de réutiliser les points *out-of-bag* pour évaluer la performance de l'algorithme.

Le méta-algorithme *self training* peut être vu comme étant *transductif*, puisque l'algorithme ne peut pas généraliser la prédiction à un nouveau point. En effet, si un point est ajouté dans l'ensemble non labellisé, l'ordre de sélection peut changer à partir d'un certain point, et toutes les prédictions et sélections suivantes seront modifiées.

En revanche, si le régresseur h (ou les $(h_i)_{i=1}^c$) sont inductifs, il peut être utilisé tel quel pour effectuer une prédiction sur de nouveaux individus. Ceci doit être considéré pour une application d'un algorithme de *self-training*, mais les études expérimentales n'ont pas besoin de s'y intéresser.

Le self-training est une méthode qui utilise les prédictions d'un algorithme pour renforcer le jeu d'apprentissage de ce même algorithme. En conséquence, il souffre d'un problème de biais de confirmation [19], ce qui peut dégrader significativement les performances dans des cas similaires.

Co-training Le *co-training*[20] peut être vu comme une extension de l'algorithme de *self-training*. Pour éviter de renforcer les erreurs de l'algorithme à chaque itération, on introduit une certaine forme de variabilité. Traditionnellement, l'ensemble des variables est séparé en deux « vues » différentes. Par exemple, une page web peut être décrite par le texte qui y figure, mais aussi par les liens qui s'y trouvent. Chacune des vues est donc un ensemble de variables. Les vues doivent être suffisantes pour obtenir une classification

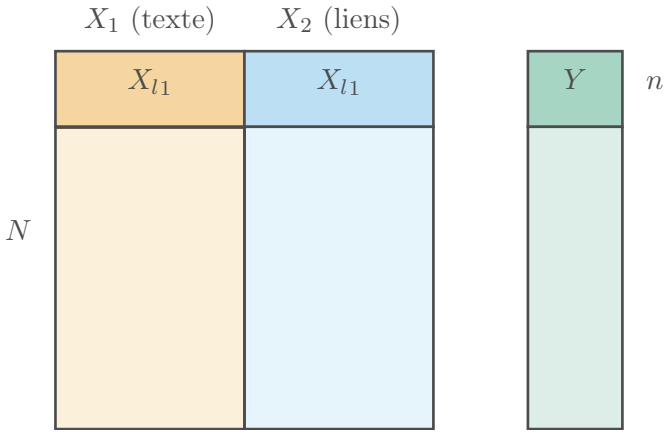


FIGURE 2.2 – Apprentissage semi-supervisé multi-vues pour le *co-training*

des pages web, mais l'information ne doit pas être répétée.

Une modification simple de l'algorithme de co-training peut avoir lieu. Pour rappel, on dispose d'un classifieur de base h. Les deux vues sont séparées en X_1 et X_2 ; se rapporter à la figure 2.2.

Le principal consiste à :

1. Faire un apprentissage d'une première instance du classifieur, h_1, de X_{l1} vers Y ;
2. Effectuer la prédiction pour l'ensemble non labellisé : prédire $\hat{Y}_1$ à partir de X_1 avec le classifieur h_1, et $\hat{Y}_2$ à partir de X_2 avec le classifieur h_2. Calculer également la confiance dans la prédiction pour les deux algorithmes ;
3. Augmenter l'ensemble d'apprentissage du classifieur h_2 avec les prédictions les plus confiantes de h_1, et vice versa.

Visuellement, on peut se fier à la figure 2.3 à titre d'exemple illustratif. En haut à gauche, le début d'une itération montre que certains individus sont labellisés pour les régresseurs h_1 et h_2 (en blanc), et d'autres non (en gris). En haut à droite, chaque régresseur effectue une prédiction, et on retient celles de confiance maximale. En bas à gauche, les valeurs prédites sont communiquées à l'autre régresseur. En bas à droite, les nouveaux individus sont labellisés.

En ce qui concerne l'adaptation à la régression, on trouve principalement l'algorithme *COREG*[21], qui, au lieu d'utiliser plusieurs vues pour entraîner deux instances différentes de l'algorithme, utilise deux algorithmes différents. Il s'agit de régresseurs aux plus proches voisins, avec deux indices de Minkowski différents : $p = 1$ et $p = 2$. C'est-à-dire que les deux régresseurs produisent une moyenne des labels des k plus proches voisins, le premier avec la distance l_1, le second avec la distance l_2.

Pour évaluer la certitude de la prédiction d'un des régresseurs, l'algorithme vérifie si l'erreur de régression locale diminue en ajoutant le point. Pour ce faire, on sélectionne k voisins du point considéré dans l'ensemble d'apprentissage labellisé, et on apprend une nouvelle instance du régresseur h_i, h'_i, mais uniquement sur ce voisinage agrémenté du

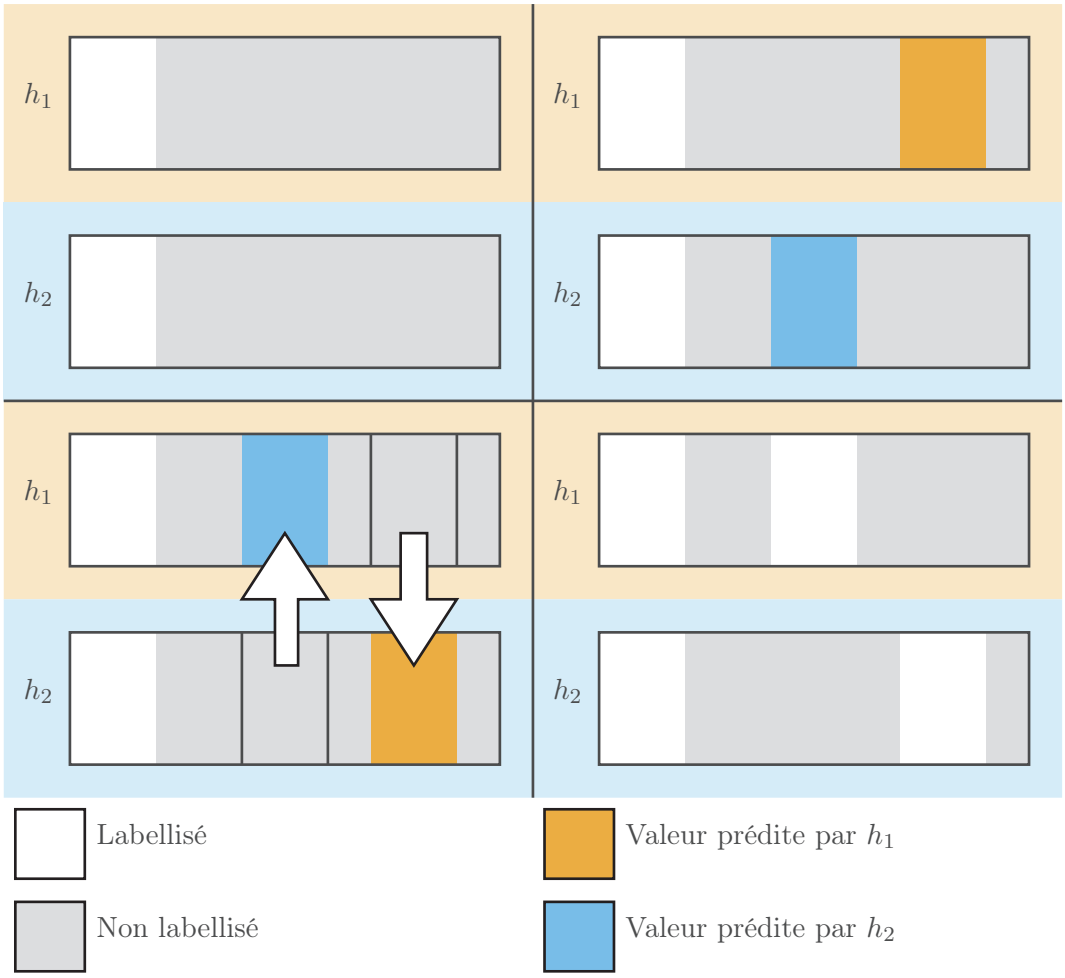


FIGURE 2.3 – Apprentissage semi-supervisé avec le *co-training*

point considéré. En comparant les résidus de h_i et h'_i sur les voisins labellisés, on peut savoir si ajouter le point augmentera ou pas les performances locales de l'algorithme.

Il est à noter que cet algorithme fonctionne grâce au fait que les régresseurs sont des k-NN (k plus proches voisins). En effet, il faut effectuer un apprentissage à chaque itération, et à chaque point non encore labellisé.

2.1.3 Sélection de variables pour l'apprentissage semi-supervisé

La sélection de variables consiste à étudier l'ensemble des variables du jeu de données et à n'en retenir qu'un petit nombre. Il y a plusieurs avantages à effectuer une sélection de variables, surtout dans les applications à grande dimension (par exemple, des applications textuelles) :

1. La dimension du problème est réduite, ce qui permet d'appliquer d'autres algorithmes ;
2. Le modèle est plus interprétable, c'est-à-dire que la connaissance extraite par l'apprentissage est compréhensible par l'être humain.

En ce qui concerne l'apprentissage semi-supervisé, il existe plusieurs méthodes de sélection de variables qui permettent de tirer profit de la présence d'individus non labellisés dans le jeu d'apprentissage.

L'évaluation de variable est une approche de sélection de variables qui donne un score à chacune des variables. Pour en effectuer la sélection, il suffit donc simplement de ne conserver que les variables dont le score est le plus élevé.

Le cadre d'apprentissage semi-supervisé appelle bien souvent la construction d'un graphe et son exploitation au travers de la matrice Laplacienne. Pour l'évaluation des variables, on dispose du *score Laplacien*[22]. La construction du graphe utilise encore une fois le même procédé : les k plus proches voisins (symétriques) sont reliés dans un graphe, et on affecte à l'arête entre ces voisins un poids calculé à partir de la fonction RBF. Une fois la matrice d'adjacence M obtenue, on calcule la matrice de degré D comme étant la diagonale de la somme des lignes (ou des colonnes) de M, et on utilise toujours la normalisation $L = D - M$.

Notons X^j la colonne j de la matrice X. Ainsi, X^j contient les données de la $j^{\text{ième}}$ variable.

Une première étape consiste à normaliser chaque variable afin d'en retirer la moyenne une fois appliquée sur le graphe :

$$\tilde{X}^j \leftarrow X^j - \frac{\sum_{i=1}^{N} X_{ij} D_{ii}}{\sum_{i=1}^{N} D_{ii}} \tag{2.17}$$

Ainsi, pour chaque variable X^j, le score Laplacien est égal à :

$$\frac{\left(\tilde{X}^j\right)' L \tilde{X}^j}{\left(\tilde{X}^j\right)' D \tilde{X}^j} \tag{2.18}$$

Cette méthode permet donc d'effectuer une sélection de variables non supervisée.

L'utilisation du score Laplacien peut aussi être détourné pour une sélection de variables semi-supervisée. Des travaux ([23], [6], [24], [25], [26]) se sont concentrés sur les approches en classification ; cependant on peut aussi l'utiliser pour des applications de régression. L'algorithme SSLS [5] se sert de la définition du score Laplacien non-supervisé, ainsi que d'une adaptation de ce score pour la régression supervisée, pour fournir un score semi-supervisé.

Le score Laplacien supervisé pour la régression est très similaire au score Laplacien non supervisé : on commence par construire un graphe des individus, labellisés cette fois, mais au lieu de considérer la distance entre individus selon leurs variables, on considère la distance entre leurs labels. Ainsi, on obtient une matrice M^{sup} définie pour une paire d'indices d'individus labellisés $i, j \leq n_l$:

$$M_{i,j}^{\text{sup}} = \exp\left(\frac{-(y_i - y_j)^2}{2\sigma^2}\right) \tag{2.19}$$

Ce graphe peut aussi être rendu *éparse* en ne sélectionnant que les arêtes les plus fortes. Le score semi-supervisé consiste donc à combiner ces deux formulations pour obtenir le graphe suivant :

$$M_{i,j}^{\text{semi}} = \begin{cases} \alpha M_{i,j}^{\text{sup}}, & i, j \leq n \\ M_{i,j}, & i > n \vee j > n \end{cases} \tag{2.20}$$

avec M le score Laplacien non supervisé vu plus haut, et α un nouvel hyperparamètre qui permet de donner plus ou moins d'importance à l'utilisation des individus labellisés par rapport aux individus non labellisés.

2.2 Régression multi-labels

L'apprentissage multi-tâches consiste à réutiliser un modèle ou une partie d'un modèle afin de résoudre plusieurs tâches d'apprentissage simultanément. C'est un cadre très utilisé dans l'apprentissage à base de données textuelles, car les différentes tâches d'apprentissage sont reliées. Dans [27], l'apprentissage multi-tâches s'effectue en réutilisant certaines couches du modèle dans un réseau de neurones profond.

La régression multi-labels est un cas particulier de la régression multi-tâches. Dans le cadre de la régression multi-labels, toutes les tâches sont continues et à une dimension. La prédiction pour un individu consiste en exactement une valeur pour chacun des labels. Cette distinction peut être comprise en comparant un jeu de données multi-tâches très utilisé, *schools*[28], et un jeu de données multi-labels, *sarcos*[29].

Le jeu de données *schools* est une agrégation des résultats d'élèves différents dans des écoles différentes ; pour chaque école, la variable cible est la note de l'élève, et les variables consistent en des constantes par école et des variables descriptives de l'élève. Tous les élèves sont décrits dans le même espace de variables, on peut donc considérer le

problème comme multi-tâches. Cependant, chaque élève n'est inscrit que dans une seule école, il est donc impossible de considérer ce jeu de données comme multi-tâches.

En comparaison, *sarcos* est tiré d'un robot, dont on a mesuré les positions, vitesses et accélérations des articulations, et dont on cherche à prédire les moments aux articulations. Cette fois-ci, il s'agit d'un jeu de données multi-labels puisque chaque mesure donne une valeur pour tous les labels.

L'apprentissage multi-labels a concerné historiquement des problèmes de classification binaire, il faut donc faire attention à ne pas le confondre avec la classification multi-classes. Dans l'apprentissage multi-labels, on suppose que les labels partagent de l'information. On peut notamment s'intéresser à la corrélation entre labels.

2.2.1 Métriques de régression multi-labels

Il y a différentes façons d'évaluer un algorithme de régression multi-labels. On cherche en effet à comparer les résultats de la prédiction, $\hat{Y} \in \mathbb{R}^{d,m}$, avec les vraies valeurs, $Y \in \mathbb{R}^{d,m}$.

On note :

— la moyenne du label $1 \leq k \leq m$: $\bar{Y}_k$;
— la matrice moyenne, $\bar{Y} \in \mathbb{R}^{n_t,m}$, définie par : $\bar{Y}_{i,k} = \left[\bar{Y}_k\right]_{i,k}$.

Les métriques usuelles de régression mono-label peuvent être réutilisées pour définir des métriques multi-labels. Pour rappel, on trouve :

— la *RMSE*, pour *root-mean-squared error*, i.e. la racine de l'erreur quadratique moyenne ;
— la *MAE*, pour *mean absolute error*, i.e. erreur absolue moyenne ;
— les métriques relatives : *RRSE* pour *root of relative squared error*, i.e. racine de l'erreur quadratique relative, et *RAE* pour *root of absolute error*, i.e. racine de l'erreur absolue. Ces métriques expriment le rapport entre la métrique associée à la prédiction et la métrique associée à la prédiction constante de la moyenne du label ;
— le coefficient de corrélation *CC* entre $\hat{Y}_{\cdot 1}$ et $Y_{\cdot 1}$.

L'extension de ces métriques en multi-labels consiste à définir une métrique *moyenne* ou *macro*, et une métrique *micro*. La métrique moyenne, préfixée par un « a » pour *average*, consiste à calculer la métrique sur tous les labels et à en retenir la moyenne. La métrique micro, préfixée par un « µ », consiste à mettre bout à bout les labels pour appliquer la métrique mono-label.

En ce qui concerne la métrique *MAE*, il n'y a pas de différence, puisqu'on peut intervertir l'ordre des sommations. On ne conservera donc que la *aMAE*.

$$
\begin{aligned}
\text{aRMSE} &= \frac{1}{m}\sum_{k=1}^{m}\sqrt{\frac{1}{n_t}\sum_{i=1}^{n_t}\left(\hat{Y}_{i,k}-Y_{i,k}\right)^2} & \mu\text{RMSE} &= \sqrt{\frac{1}{n_t m}\sum_{i,k=1}^{n_t,m}\left(\hat{Y}_{i,k}-Y_{i,k}\right)^2}\\
\text{aMAE} &= \frac{1}{n_t m}\sum_{i,k=1}^{n_t,m}\left|\hat{Y}_{i,k}-Y_{i,k}\right| & &\\
\text{aRRSE} &= \frac{1}{m}\sum_{k=1}^{m}\sqrt{\frac{\sum_{i=1}^{n_t}\left(\hat{Y}_{i,k}-Y_{i,k}\right)^2}{\sum_{i=1}^{n_t}\left(\bar{Y}_k-Y_{i,k}\right)^2}} & \mu\text{RRSE} &= \sqrt{\frac{\sum_{i,k=1}^{n_t,m}\left(\hat{Y}_{i,k}-Y_{i,k}\right)^2}{\sum_{i,k=1}^{n_t,m}\left(\bar{Y}_k-Y_{i,k}\right)^2}}\\
\text{aRAE} &= \frac{1}{m}\sum_{k=1}^{m}\frac{\sum_{i=1}^{n_t}\left|\hat{Y}_{i,k}-Y_{i,k}\right|}{\sum_{i=1}^{n_t}\left|\bar{Y}_k-Y_{i,k}\right|} & \mu\text{RAE} &= \frac{\sum_{i,k=1}^{n_t,m}\left|\hat{Y}_{i,k}-Y_{i,k}\right|}{\sum_{i,k=1}^{n_t,m}\left|\bar{Y}_k-Y_{i,k}\right|}\\
\text{aCC} &= \frac{1}{m}\sum_{k=1}^{m}\frac{\sum_{i=1}^{n_t}\left(\hat{Y}_{i,k}-\bar{\hat{Y}}_k\right)\left(Y_{i,k}-\bar{Y}_k\right)}{\left\|\hat{Y}_{.,k}-h\bar{a}tY_k\right\|\left\|Y_{.,k}-\bar{Y}_k\right\|} & \mu\text{CC} &= \frac{\text{tr}\left(\left(\hat{Y}-\bar{\hat{Y}}\right)'\left(Y-\bar{Y}\right)\right)}{\left\|\hat{Y}-\bar{\hat{Y}}\right\|_F\left\|Y-\bar{Y}\right\|_F}
\end{aligned}
\tag{2.21}
$$

Parmi toutes ces métriques, la *aRMSE* est la plus employée.

2.2.2 Chaînes de régresseurs pour l'apprentissage multi-labels

Nous allons commencer par étudier les approches algorithmiques qui permettent de convertir un régresseur mono-label en régresseur multi-labels [30].

Stacking pour la régression multi-labels

Pour réutiliser l'information des labels, l'approche de transformation la plus populaire est l'empilement de régresseurs (*stacking*). Dans ce contexte, un régresseur est entraîné pour chaque label. On obtient un nouvel espace de description des individus, en combinant les variables du jeu d'apprentissage et les valeurs des labels. Un deuxième ensemble de régresseurs est entraîné, ce qui permet d'effectuer une prédiction en deux temps tout en conservant de l'information entre tous les labels.

Chaînes de régresseurs

Il est possible de développer cette approche. Pour construire une chaîne de régresseurs, il faut sélectionner un ordre des labels, puis pour chacun, entraîner un régresseur pour obtenir une prédiction pour ce label à partir de l'ensemble des variables ainsi que l'ensemble des prédictions pour les labels précédemment sélectionnés. La prédiction se fait dans le même ordre, en utilisant les prédictions précédentes. On obtient l'algorithme *RCC* pour chaîne de régresseurs corrigée (*regressor chains corrected*, la correction venant du fait que chaque étape effectue une prédiction à partir des prédictions et non des labels).

Chaînes de régresseurs ensemblistes

Cette méthode ne donne pas le même résultat selon l'ordre de sélection des labels. C'est un problème, car la prédiction pour le premier label ne peut pas s'appuyer sur la prédiction des autres : il est de fait traité comme un problème mono-label. Ce problème est résolu en prenant plusieurs ordres différents, et en effectuant une prédiction ensembliste, où la valeur prédite pour chaque label est la moyenne des valeurs prédites par chaque ordre de sélection.

2.2.3 Approches portant sur la régularisation multi-labels

En ce qui concerne les approches de régression, l'optimisation de méthodes de régularisation multi-labels pose un problème supplémentaire par rapport à la régression mono-label. En effet, si l'on veut appliquer une simple régularisation Ridge en multi-labels, la solution est simple à calculer :

$$W = [X'X + \alpha I]^{-1} X'Y \tag{2.22}$$

Malheureusement, en ajoutant des régularisations relatives aux labels, par exemple en remplaçant le terme Ridge par $\alpha \|WA\|$ avec A une matrice quelconque, on obtient :

$$\left[[X'X] \otimes I_m + \alpha I_d \otimes [AA']\right]^{-1} X'Y \tag{2.23}$$

où I_m (I_d) est la matrice identité de dimension m (d) et $\_ \otimes \_$ désigne le produit de Kronecker. La dimension du système à résoudre n'est donc plus $d \times d$, mais bien $d \times d \times m \times m$, et on ne peut plus se satisfaire de solution analytique au problème.

MALSAR (Multi task Learning via structural regularization)

MALSAR [1] est une implémentation d'un ensemble de méthodes partageant l'utilisation d'une fonction objectif régularisée pour traiter le problème multi-labels. L'optimisation se fait toujours par descente de gradient accélérée, ce qui résout le problème de complexité soulevé plus haut.

Lasso par groupe

La régularisation du Lasso par groupe, *group lasso*[31], utilise l'hypothèse suivante :

> Parmi l'ensemble des variables du problème, seul un certain nombre d'entre elles sont utiles pour la prédiction simultanée de tous les labels.

Pour utiliser cette hypothèse, le Lasso [32], ou régularisation l_1, est un bon candidat. En effet, si l'on considère le problème suivant :

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad \|Xw - y\|_2^2 + \alpha \|w\|_1 \tag{2.24}$$

1. https://github.com/jiayuzhou/MALSAR

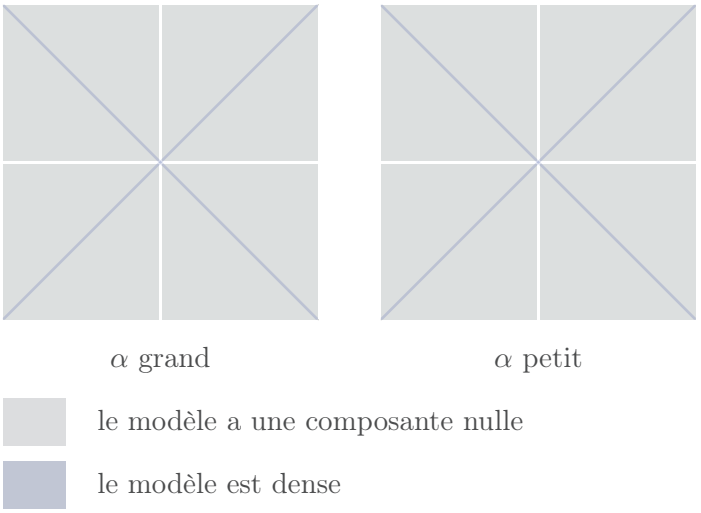


FIGURE 2.4 – Comparaison de modèles pour Lasso : lorsque le régulariseur α est grand, les zones dans lesquelles le modèle perd une composante sont plus étendues.

La solution, en fonction de la valeur de α, sera plus ou moins éparse, c'est-à-dire que le modèle w aura un certain nombre de composantes entièrement nulles. Plus précisément, l'optimisation de ce problème par descente de gradient proximal nous indique que la solution se situe à une distance l_1 de α^{-1} de l'origine. Si le problème a deux dimensions, une grande valeur de α laisse une plus grande partie de l'espace menant à une solution où l'une des coordonnées du modèle est nulle, cf figure 2.4.

Pour l'apprentissage multi-labels, l'extension du lasso retenue est le lasso par groupe [31]. En effet, on suppose que tous les labels peuvent être prédits à partir d'un sous-ensemble des variables. On souhaite donc appliquer une régularisation équivalente, en n'appliquant la régularisation que sur les *lignes* de W. On obtient donc le problème 2.25 :

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|XW - Y\|_F^2 + \alpha \sum_{j=1}^{d} \|W_{j.}\|_2 \tag{2.25}$$

La norme ainsi obtenue est notée $\|W\|_{2,1}$.

Cette formulation utilise la norme $l_{2,1}$. Il est possible d'utiliser d'autres normes pour obtenir un résultat comparable. L'algorithme nommé *Dirty Model*[33] utilise deux termes pour la régularisation, afin de pouvoir traiter à la fois les cas où l'on peut réutiliser les mêmes variables pour tous les labels, et ceux où il faut un modèle différent par label. Le modèle s'exprime de la façon suivante 2.26 :

$$\underset{W = P + Q, P \in \mathbb{R}^{d,m}, Q \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|X(P+Q) - Y\|_F^2 + \alpha \|P\|_{1,1} + \beta \|Q\|_{\infty,1} \tag{2.26}$$

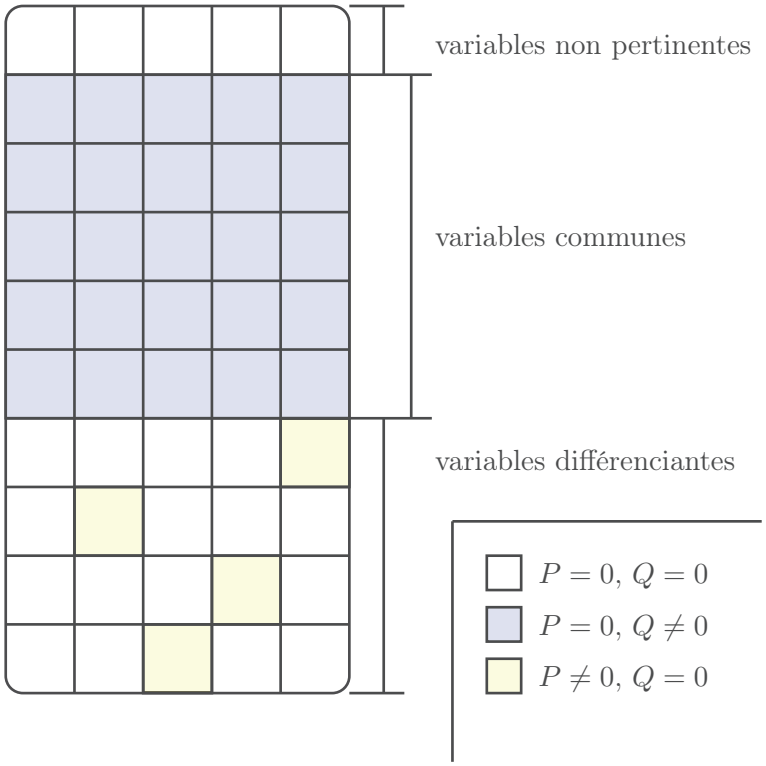


FIGURE 2.5 – Un régresseur produit par le *dirty model*

Pour plus de clarté, on utilise la notation $\|Q\|_{\infty,1}$ et pas la notation employée dans l'article original [33].

En sélectionnant les hyperparamètres α et β, il est possible de traiter mieux plus de cas que le problème 2.25. Le modèle se décompose donc en deux termes, comme le montre la figure 2.5.

Régularisation Laplacienne multi-labels

La régularisation Laplacienne évoquée tout au long de ce document peut également être utilisée pour une régularisation multi-labels [34]. En effet, si l'on considère que chaque label correspond à un nœud d'un graphe, on peut définir une arête entre deux labels comme la similarité entre ces deux labels. Pour cela, si l'on considère deux labels quelconques, on peut comparer pour chaque individu les valeurs assignées à ces deux labels. On peut ensuite agréger le résultat sur tous les individus. Concrètement, la construction du graphe se fait en utilisant non pas X comme pour l'utilisation semi-supervisée, mais Y'_l. L'hypothèse que l'on souhaite mettre en œuvre peut être résumée de la façon suivante :

> Si deux labels sont similaires, les valeurs du modèle pour ces deux labels doivent être similaires.

On peut donc utiliser la fonction objectif suivante :

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|XW - Y\|_F^2 + \alpha \text{tr}(W L_m W') \tag{2.27}$$

Cette régularisation est utilisée dans l'algorithme GLOCAL [35], qui part du principe que les données peuvent être partitionnées en sous-ensembles dans lesquels la corrélation entre labels est différente.

Régularisation sur le rang

L'hypothèse selon laquelle les labels partagent de l'information peut aussi se traduire par une contrainte sur le rang (des colonnes) du modèle W. En effet, si l'on considère que le modèle W s'écrit comme un produit $P \times Q$, avec deux matrices $P \in \mathbb{R}^{d,o}$ et $Q \in \mathbb{R}^{o,m}$, alors le modèle est de rang au plus o, et on peut constater que l'information contenue dans P est réutilisée pour tous les labels.

Dans la pratique, cette idée se traduit par une régularisation sur la norme trace de la matrice W[36].

$$\|W\|_* = \sqrt{tr\left(W'W\right)} = \sum_{i=1}^{m} \sigma_i(W) \tag{2.28}$$

où $(\sigma_i(W))_{i=1}^m$ désigne l'ensemble des valeurs propres de la matrice W. Dans ce cas, le problème s'écrit :

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|XW - Y\|_F^2 + \alpha \|W\|_* \tag{2.29}$$

Classification des labels

L'approche *CMTL* pour *clustered multi-task learning* propose de faire une classification des labels, à partir de l'algorithme des k-moyennes. Selon [37], si l'on a o clusters de labels, en supposant que les clusters sont $(\mathcal{C}_l)_{l=1}^o$, dont chacun est de taille n_l, et l'indicatrice des clusters se note $C \in \mathbb{R}^{k,m}$ telle que pour un label k et un cluster o :

$$C_{kl} = \begin{cases} \frac{1}{\sqrt{n_l}}, & k \in \mathcal{C}_l \\ 0, & k \notin \mathcal{C}_l \end{cases} \tag{2.30}$$

on peut écrire le coût de la façon suivante :

$$\sum_{l=1}^{o} \sum_{k \in \mathcal{C}_l} \left\| W_{.k} - \frac{1}{n_l} \sum_{k' \in \mathcal{C}_l} W_{.k} \right\|_2^2 = \text{tr}\left(W'W\right) - \text{tr}\left(C'W'WC\right) \tag{2.31}$$

En relaxant la définition de C pour avoir simplement $C'C = I$, le problème des k-moyennes devient :

$$\begin{array}{ll} \underset{C \in \mathbb{R}^{m,o}}{\text{minimize}} & \operatorname{tr}\left(W'W\right) - \operatorname{tr}\left(C'W'WC\right) \\ \text{subject to} & C'C = I \end{array} \tag{2.32}$$

À partir de cette formulation, il est possible de construire une fonction objectif multi-labels [38] :

$$\begin{array}{ll} \underset{W \in \mathbb{R}^{d,m}, C \in \mathbb{R}^{m,o}}{\text{minimize}} & \|XW - Y\|_F^2 + \alpha \left(\operatorname{tr}\left(W'W\right) - \operatorname{tr}\left(C'W'WC\right)\right) + \beta \|W\|_F^2 \\ \text{subject to} & C'C = I \end{array} \tag{2.33}$$

La formulation n'étant pas convexe, elle n'est pas satisfaisante. [38] propose donc de poser $M = CC'$, $\eta = \frac{\beta}{\alpha}$, pour obtenir la fonction objectif suivante :

$$\begin{array}{ll} \underset{W \in \mathbb{R}^{d,m}, M \in \mathbb{R}^{m,m}}{\text{minimize}} & \|XW - Y\|_F^2 + \alpha\eta\,(1+\eta) \operatorname{tr}\left(W\left(\eta I + M\right)^{-1} W'\right) \\ \text{subject to} & M \succeq 0, \\ & I - M \succeq 0, \\ & \operatorname{tr}\left(M\right) = o \end{array} \tag{2.34}$$

La notation $M \succeq 0$ signifie que M est une matrice réelle symétrique semi-définie positive.

Régularisation non-convexe avec une norme plafonnée

Jusqu'à présent, toutes les fonctions objectifs que nous avons décrites sont convexes, ou convexes selon chacune de leurs variables. Cependant, on peut également envisager des régularisations non convexes. On n'obtient en général pas de garantie de convergence, mais les résultats expérimentaux sont souvent suffisamment importants pour qu'il ne soit pas possible d'ignorer ces approches.

La norme l_1 plafonnée [39, 40] consiste simplement à régulariser certaines *lignes* du modèle W avec la norme l_1, mais pas toutes. Les lignes ayant des valeurs importantes ne sont pas régularisées. Ce n'est pas la même chose que la norme $l_{2,1}$ vue précédemment : ici, l'application de la norme pour une variable régularisée signifie qu'il n'y aura que quelques labels utilisant cette variable.

Le problème s'écrit de la façon suivante :

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|XW - Y\|_F^2 + \alpha \sum_{j=1}^{d} \min\left(\|W_{j\cdot}\|_1, \beta\right) \tag{2.35}$$

Ce problème n'est pas convexe ; il peut cependant être relaxé pour être résolu itérativement, en utilisant plusieurs étapes. À chaque itération, on détermine les lignes pour

lesquelles la régularisation s'applique, en calculant la norme l_1 du modèle à l'itération précédente. Puis on résout le problème en appliquant la régularisation uniquement sur ces lignes.

La relaxation converge vers la solution du problème non convexe, sous certaines conditions [40].

En ce qui concerne la régularisation sur le rang, il est aussi possible de traiter un problème de norme plafonnée [41]. Le problème s'écrit alors :

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|XW - Y\|_F^2 + \alpha \sum_{k=1}^{m} \min\left(\sigma_k\left(W\right), \beta\right) \tag{2.36}$$

La notation $\sigma_k(W)$ désigne la $k^{\text{ième}}$ valeur propre de W. Si l'on suit le même processus de relaxation que pour MSMTFL [40], la régularisation ne doit s'appliquer que sur les plus petites valeurs propres. Si l'on note s le nombre de valeurs propres supérieures à β, et $\|W\|_{s-}$ la somme des valeurs propres de W, sauf les s premières, chaque itération résout le problème suivant :

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|XW - Y\|_F^2 + \alpha \|W\|_{s-} \tag{2.37}$$

Cette norme peut en fait se réécrire comme la norme trace, moins un terme calculé à partir de la décomposition en valeurs propres du modèle à l'itération précédente, en tronquant la décomposition à s termes. C'est un problème convexe, mais avec un terme non lisse (la norme trace), qui peut être résolu de la même manière que la régularisation trace seule.

2.2.4 Semi-Supervised Multi-Task Regression, avec contraintes

L'algorithme *SSMTR*[10] est une approche non linéaire, utilisant un noyau, permettant de traiter le problème de régression semi-supervisée en multi-labels. Il s'agit d'une extension de la régression multi-labels semi-supervisée *SMTR*, présentée également dans [10].

Dans *SMTR*, chaque label est traité séparément par un apprentissage à noyau, mais les paramètres des noyaux pour tous les labels sont issus de la même distribution. C'est une façon de réutiliser de l'information entre labels. L'extension au cadre semi-supervisé vise à définir pour chaque label un graphe d'individus, en déduire la matrice Laplacienne et l'utiliser en complément du noyau multi-labels.

Bien qu'une matrice Laplacienne soit calculée pour chaque label, l'inférence du modèle évite de calculer un produit de Kronecker. En revanche, cette extension correspond bien à la régression Laplacienne mono-label classique : une fois le jeu de données non labellisé utilisé pour apprendre le modèle, il est éliminé et ne sert plus pour la prédiction. [10] propose donc d'ajouter deux types de contraintes :

1. La prédiction pour un individu est supérieure à la prédiction pour un autre individu (avec une marge) ;

2. Un écart de prédiction entre deux individus doit être supérieur à un écart de prédiction entre deux autres individus.

L'inférence devient plus délicate, mais donne de meilleurs résultats pour les jeux de données évoqués précédemment *sarcos* et *schools*.

2.2.5 Le problème de sélection de variables multi-labels

Dans un problème d'apprentissage multi-labels, la sélection de variables a un sens légèrement plus précis : on doit trouver un sous-ensemble des variables qui permet d'obtenir une bonne prédiction pour *tous* les labels simultanément. Ainsi, on peut difficilement généraliser les approches de sélection de variables mono-label directement au multi-labels. Par exemple, l'utilisation du score Laplacien supervisé, utilisé dans *SSLS*[5] permet de faire de la sélection de variables, en s'appliquant sur un jeu de données à plusieurs labels, mais ne correspond pas à notre problème de sélection de variables multi-labels. En effet, la construction du graphe supervisé met en relation les individus de l'ensemble d'apprentissage dont la distance dans l'espace des labels est faible. La sélection de variables consiste donc en une sélection *moyenne* pour les différents labels, et elle n'est pas conçue pour traiter tous les labels en même temps.

De la même façon, une adaptation naïve du Lasso en multi-labels ne permet pas de résoudre le problème de sélection de variables multi-labels. En effet, si l'on cherche à résoudre le problème suivant :

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|XW - Y\|_F^2 + \alpha \sum_{j=1}^{d} \sum_{k=1}^{m} |W_{jk}| \tag{2.38}$$

On obtiendra une matrice W sparse, mais ce n'est pas suffisant : il se peut que chaque label demande un sous-ensemble différent de variables. Il faut que tous les labels partagent un même sous-ensemble de variables.

Utilisation du Lasso par groupe

Pour pallier ce problème, il faut utiliser une version un peu moins naïve du Lasso. En construisant un modèle linéaire W, les lignes correspondent aux coefficients pour une variable et les colonnes correspondent aux coefficients pour un label. Ainsi, si l'on veut pouvoir exprimer tous les labels avec un petit nombre de variables, il faut que la matrice W ait un petit nombre de lignes non nulles. Il faut donc appliquer une régularisation de type Lasso sur les lignes de W, c'est-à-dire un lasso par groupe. On retrouve donc directement l'algorithme *JFS*[31].

Régularisation $l_{2,1-2}$ non convexe pour la sélection de variables

Si l'on examine la norme $l_{2,1}$ pour un modèle à 2 variables et 1 label, on obtient la figure 2.6 haut. On constate que la valeur de la fonction de coût est plus faible si l'une des coordonnées du modèle est nulle, et elle est globalement minimale si les deux coordonnées

sont nulles. C'est utile pour la sélection de variables, puisque l'optimisation du problème pourra éliminer l'une des deux variables. Cependant, [42] propose une amélioration pour la sélection de variables multi-labels, en utilisant la différence entre la norme $l_{2,1}$ et la norme l_2. Les effets de cette régularisation sont très parlants pour un modèle à 2 variables (figure 2.6 bas) : le coût est exactement nul dès que l'une des variables est éliminée.

Ce problème n'est pas convexe, cependant il est possible de l'optimiser en relaxant la partie non convexe par le plan tangent à chaque itération.

Multi-Labels Informed Feature Selection

Pour les problèmes de sélection de variables, il n'est pas toujours pertinent d'obtenir de bons résultats en régression. Ceci débloque une opportunité pour utiliser des variables latentes. C'est l'approche développée par l'algorithme *MIFS*, pour *Multi-Labels Informed Feature Selection*[11]. Elle se fonde sur plusieurs hypothèses, qui sont intégrées dans une même fonction objectif.

Tout d'abord, *MIFS* propose une abstraction des labels. Concrètement, il faut sélectionner un nombre de labels latents, $1 \leq o \leq m$, et introduire une matrice de pseudo-labels $V \in \mathbb{R}^{n,o}$ ainsi qu'une matrice $B \in \mathbb{R}^{o,m}$ de sorte à substituer VB à Y, et à proposer un apprentissage de V :

$$\underset{W \in \mathbb{R}^{d,o}, V \in \mathbb{R}^{n,o}, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad \|XW - V\|_F^2 + \alpha \|Y - VB\|_F^2 \tag{2.39}$$

Ensuite, une régularisation Laplacienne introduit l'hypothèse suivante : pour chaque pseudo-label, deux individus proches ont des valeurs proches. Enfin, la régularisation sur les lignes de la matrice W permet d'obtenir un modèle sparse par ligne, ce qui permet de guider la sélection de variables.

$$\underset{W \in \mathbb{R}^{d,o}, V \in \mathbb{R}^{n,o}, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad \|XW - V\|_F^2 + \alpha \|Y - VB\|_F^2 + \beta \text{tr}\left(V'LV\right) + \gamma \|W\|_{2,1} \tag{2.40}$$

La fonction objectif est convexe en chacune de ces variables ; l'algorithme MIFS choisit une optimisation alternée pour l'optimisation.

2.3 Conclusion

Dans ce chapitre, nous avons présenté des méthodes de l'état de l'art pour résoudre des problèmes d'apprentissage semi-supervisés, et pour la régression multi-labels. Nous nous sommes également intéressés au problème de sélection de variables.

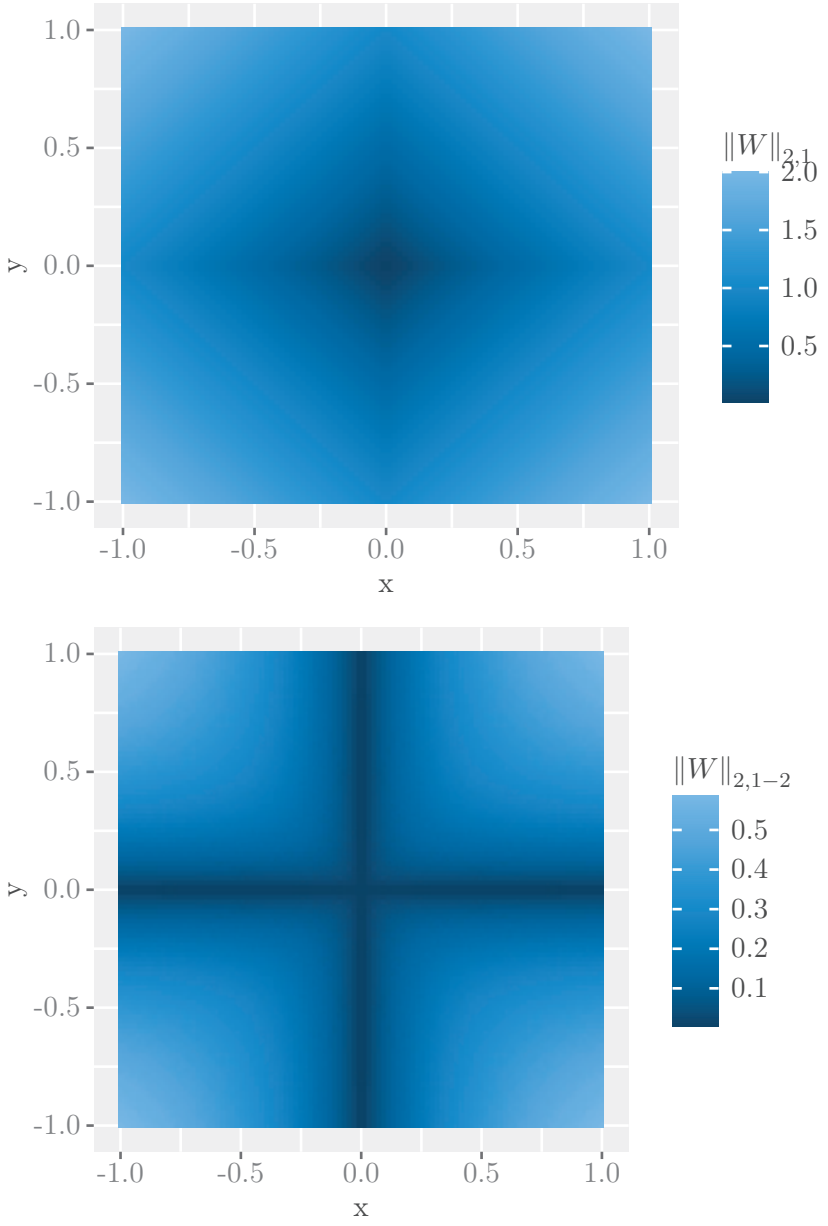


FIGURE 2.6 – Norme $l_{2,1}$ et $l_{2,1-2}$ d'un modèle à 2 variables et 1 label.

Chapitre 3

Régression Laplacienne semi-supervisée

Dans ce chapitre, nous présentons notre première contribution sur la régression semi-supervisée utilisant une version régularisée de l'algorithme *SSSL*[1]. Nous commençons par décrire les travaux liés, puis nous décrivons l'approche proposée, après quoi nous montrerons à travers deux études expérimentales l'apport de notre proposition.

L'algorithme *SSSL* de base consiste en deux étapes distinctes : un changement d'espace non supervisé, puis une régression simple dans ce nouvel espace. Nous proposons d'emprunter la même idée du changement d'espace original, mais en adoptant une régularisation Laplacienne dans le nouvel espace ainsi obtenu.

3.1 Introduction

À cause de la prolifération des données partiellement labellisées, l'apprentissage automatique a connu des développements majeurs dans le contexte semi-supervisé [3]. Cette tendance est due d'une part à la difficulté de la tâche de labellisation, et d'autre part au coût associé lorsque la labellisation est possible. L'apprentissage semi-supervisé est un cas particulier de l'apprentissage faiblement labellisé [43], qui consiste en général en l'apprentissage d'une fonction objectif à partir de données contenant à la fois des individus labellisés et des individus non-labellisés. Ces situations peuvent être abordées par deux classes d'approches : l'une fondée sur la propagation des labels, dans le but d'utiliser des méthodes supervisées [4], et l'autre fondée sur la transformation des données labellisées en contraintes à intégrer dans un processus non-supervisé de clustering [44]. Nous nous concentrons sur la première classe d'approches, avec un défi particulier : il s'agit d'apprendre à partir d'un petit sous-ensemble supervisé, en comparaison de l'ensemble non supervisé. Dans ce contexte semi-supervisé, la littérature a connu une extension significative depuis les vingt dernières années, en particulier pour la classification, grâce à des approches populaires comme le self-training [3], le co-training [20], le SVM transductif ou S^3VM [45, 46], les approches fondées sur des graphes [47] et des approches génératives [48]. Dans ce paradigme particulier, les problèmes de régression ont aussi suscité l'intérêt de plusieurs travaux de recherche que nous pouvons citer de façon non exhaustive. Les approches sont diverses : fondées sur la régression linéaire [49, 1], logistique [50], ou Laplacienne [51, 2] ; d'autres approches sont fondées sur le co-training [52] ; ou fonctionnent avec des contraintes spécifiques liées à l'ordre de préférence des données non labellisées [4] ou leur distribution géométrique dans des espaces de haute dimension [53, 54].

3.2 Travaux liés

L'algorithme SSSL (*Simple algorithm for Semi-Supervised Learning*) [1] se décompose en deux étapes simples :

1. Extraction de variables non supervisée au moyen d'une ACP à noyau (*k-PCA*, *kernel PCA*) ;
2. Apprentissage supervisé dans ce nouvel espace.

Sous certaines hypothèses vis-à-vis des données, il est possible de garantir que dans la classe de fonctions recherchée, l'algorithme fournit une fonction ayant une erreur de régression minimale (à une constante près), cette erreur de généralisation théorique restant inférieure à ce que l'on obtiendrait sans les individus non labellisés. Ces hypothèses portent sur :

— l'erreur de régression minimale sur la classe de recherche des fonctions ;
— la distribution des valeurs propres de la matrice de noyau ;
— les vecteurs propres de cette matrice ;
— le nombre d'individus non labellisés.

L'algorithme *SSSL* permet de rechercher une fonction de prédiction qui puisse s'écrire de la manière suivante. Si N désigne le nombre d'individus y compris les non labellisés de l'ensemble d'apprentissage, et $x_i \in \mathcal{X}$ désigne le $i^{\text{ième}}$ individu de l'ensemble d'apprentissage, on définit une fonction de noyau $\kappa\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. On peut également définir la fonction $\hat{L}_N$ suivante :

$$\hat{L}_N\colon \begin{array}{ccc} (\mathcal{X} \to \mathbb{R}) & \to & (\mathcal{X} \to \mathbb{R}) \\ f & \mapsto & x \mapsto \frac{1}{N}\sum_{i=1}^{N} \kappa(x_i, x) f(x_i) \end{array} \tag{3.1}$$

On note les valeurs propres de cet opérateur linéaire $\left(\hat{\lambda}\right)_{k=1}^{N}$ et ses fonctions propres $(\phi)_{k=1}^{N}$. La fonction de prédiction est recherchée comme une combinaison linéaire des $s \leq N$ fonctions propres ayant la plus grande valeur propre :

$$f\colon x \mapsto \sum_{k=1}^{s} \Gamma_k \hat{\phi}_k(x) \tag{3.2}$$

Les coefficients Γ peuvent être obtenus tout simplement par une régression aux moindres carrés, c'est-à-dire en minimisant l'erreur de régression (z désigne le vecteur de labels pour les individus labellisés) :

$$\underset{\Gamma \in \mathbb{R}^d}{\text{minimize}} \quad \|f(x) - z\|_2^2 \tag{3.3}$$

Par conséquent, les valeurs propres elles-mêmes de $\hat{L}_N$ ne sont pas utilisées.

La régularisation Laplacienne que nous utilisons [2] ne considère pas les labels des données, ce qui permet d'utiliser tous les individus, y compris ceux de l'ensemble non-supervisé. À partir de leurs similarités dans l'espace des variables, un graphe est construit. Les nœuds sont les individus, et les arêtes indiquent des individus similaires. La régularisation consiste à indiquer que les individus similaires doivent avoir des valeurs de la variable cible similaires, en minimisant le terme suivant :

$$\sum_{i,j \text{ similaires}} (f(x_i) - f(x_j))^2 \tag{3.4}$$

3.3 Approche proposée : *LapS3L*

Bien que le changement d'espace introduit par *SSSL*(présenté ci-dessus) permette de mieux traiter certains problèmes de régression semi-supervisée [1], elle n'utilise pas les mêmes hypothèses que la régularisation Laplacienne. Dans le cas général, il n'est pas possible de conclure à la proximité de deux individus dans l'espace extrait à partir de deux individus proches dans l'espace réel. Nous proposons donc de réguler l'apprentissage en introduisant l'hypothèse suivante :

> Si deux individus sont proches dans l'espace réel, alors l'écart de prédiction doit être faible.

Nous proposons une nouvelle approche, fondée sur cette hypothèse, que nous appelons *LapS3L*, pour *Laplacian-regularized Simple algorithm for Semi-Supervised Learning*. Nous décrivons dans ce qui suit, ses différentes étapes.

3.3.1 Changement d'espace

En reprenant le cadre de l'algorithme *SSSL*, il est possible de tisser un parallèle entre la décomposition en valeurs et fonctions propres de $\hat{L}_N$ et la décomposition en valeurs propres et vecteurs propres de la matrice noyau. Ce parallèle permet d'obtenir un algorithme d'apprentissage en deux temps : obtention des vecteurs propres de la matrice noyau, puis obtention de ses coefficients.

Tout d'abord, cette matrice notée $K \in S_N^+(\mathbb{R})$ est définie comme l'application symétrique de la fonction κ entre chaque individu de l'ensemble d'apprentissage :

$$\forall i, j \in \{1, ..., N\}, \quad K_{i,j} = \kappa(x_i, x_j) \tag{3.5}$$

L'analyse des vecteurs propres de K permet d'obtenir une matrice de vecteurs propres $U \in \mathbb{R}^{N,s}$, dont les colonnes sont unitaires. On peut donc poser le changement de variables suivant :

$$X \leftarrow K'U \tag{3.6}$$

Dans ce changement de variables, chaque ligne de la matrice X correspond à un individu de l'ensemble d'apprentissage. Chaque colonne correspond à une variable extraite. Le nombre de variables extraites est compris entre 1 et N. Par conséquent, dans le cas où la matrice de données est plus longue que large, on peut avoir $s > d$. Pour les applications où les données sont en grande dimension, comme par exemple pour les données textuelles, d'images ou génétiques, ce n'est pas un problème. Dans les autres cas, la réduction de dimensionnalité ne peut s'opérer que si la valeur de l'hyperparamètre s est relativement faible.

Puisque les lignes de X sont aussi les lignes de la matrice de données dans l'espace original, $\mathcal{V}$, il est possible de ne conserver que les lignes correspondant aux individus labellisés. On obtient ainsi une sous-matrice $X_l \in \mathbb{R}^{n,s}$.

3.3.2 Régression régularisée

Dans la deuxième partie de l'algorithme, l'implémentation consiste à effectuer une régression simple à partir de ces nouvelles variables. On obtient donc une liste de coefficients, w, de dimension s.

$$\underset{w \in \mathbb{R}^s}{\text{minimize}} \quad \|X_l w - z\|_2^2 \tag{3.7}$$

Dans le cas où la fonction κ est en fait le produit scalaire, la formulation de la régression dans le nouvel ensemble de variables X, 3.7, montre qu'il n'y a pas de solution dans le cas où $s > d$. L'extraction de variables permet donc dans ce cas une réduction

de dimensionnalité. Dans le cas général, cette remarque ne tient pas. Par exemple, pour n'importe quel hyperparamètre σ du noyau RBF :

$$\kappa\colon (x_i, x_j) \mapsto e^{-\frac{\left\|x_i - x_j\right\|_2^2}{2\sigma^2}} \tag{3.8}$$

La matrice noyau obtenue K est à diagonale strictement dominante, donc inversible ; par conséquent la matrice de covariance $X'X$ l'est aussi en extrayant N valeurs propres. Dans la pratique, les approximations numériques ne permettent pas de choisir une grande valeur de s.

Étant donné que la prédiction s'effectue directement dans l'espace extrait, il suffit donc de construire le graphe des individus à partir des variables d'origine, et d'ajouter un terme de régularisation dans la fonction objectif linéaire de la seconde étape. Concrètement, on remplace la régression 3.7 par la version utilisant la régularisation Laplacienne 3.9 :

$$\underset{w \in \mathbb{R}^s}{\text{minimize}} \quad \|X_l w - z_l\|_2^2 + \alpha \|w\|^2 + \beta [Xw]' L [Xw] \tag{3.9}$$

Cette approche ne nécessite pas de changer la prédiction ; elle n'intervient que dans l'apprentissage.

3.3.3 Prédiction

Pour la prédiction sur un ensemble de test $\mathcal{V}_t$ à n_t individus, on commence par appliquer la matrice κ entre tous les individus de l'ensemble labellisé, à gauche, et tous les individus de l'ensemble de test, à droite. On note le résultat $K_b \in \mathbb{R}^{N,n_t}$:

$$\forall i \in \{1, ..., N\}, \quad \forall j \in \{1, ...n_t\}, \quad K_{bi,j} = \kappa(\mathcal{V}_{i,.}, \mathcal{V}_{tj,.}) \tag{3.10}$$

Le changement de variables est identique à celui effectué pour l'apprentissage :

$$X_t \leftarrow K_b' U \in \mathbb{R}^{n_t, s} \tag{3.11}$$

La prédiction est donc opérée par :

$$X_t w \tag{3.12}$$

3.3.4 Algorithme

L'algorithme 1 montre les étapes de l'apprentissage, et l'algorithme 2 décrit la fonction de prédiction.

Le problème de minimisation de la fonction objectif, 3.9, admet une solution analytique [3] :

La partie déterminante de la complexité de l'algorithme réside dans la décomposition en valeurs propres et vecteurs propres de la matrice de noyau, de dimension $N \times N$. Comme nous ne nous intéressons qu'aux valeurs propres les plus grandes en valeur absolue de cette matrice symétrique réelle, nous pouvons appliquer un algorithme itératif [55].

$$W \leftarrow \left[X_l' X_l + \alpha I + \beta X' L X\right]^{-1} X_l' Y \tag{3.13}$$

Algorithm 1 Algorithme *LapS3L* : apprentissage

Donnée : $\mathcal{V} \in \mathbb{R}^{N,d}$
Donnée : $z \in \mathbb{R}^{n_l}$
Hyperparamètre : $s \in \{1, ..., N\}$
Hyperparamètre : $\kappa \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Hyperparamètre : matrice Laplacienne du graphe des individus, $L \in \mathbb{R}^{N,N}$
Hyperparamètre : $\alpha > 0$
Hyperparamètre : $\beta > 0$
1. Construire la matrice $K \in \mathbb{R}^{N,N}$:

$$\forall i, j, \quad K_{i,j} = \kappa(\mathcal{V}_{i,.}, \mathcal{V}_{j,.})$$

2. Décomposer la matrice K en valeurs propres et vecteurs propres, sélectionner les s vecteurs propres ayant la plus grande valeur propre associée : $U \in \mathbb{R}^{N,s}$
3. Poser $X := KU$
4. Sélectionner la sous-matrice $Xl \in \mathbb{R}^{n,s}$ composée des lignes de X correspondant aux individus labellisés
5.

$$\underset{w \in \mathbb{R}^s}{\text{minimize}} \quad \|X_l w - z_l\|_2^2 + \alpha \|w\|^2 + \beta [Xw]' L [Xw] \tag{3.14}$$

Résultat : $U \in \mathbb{R}^{N,s}$
Résultat : $w \in \mathbb{R}^s$

3.4 Validation expérimentale

Dans cette section, nous présentons une première étude expérimentale pour valider l'approche proposée sur des données publiques issues de la littérature.

3.4.1 Jeux de données utilisés

Les auteurs dans [1] ont identifié trois jeux de données pour lesquels les hypothèses de fonctionnement sont établies : *wine*, *insurance* et *temperature*.

— *wine* regroupe une évaluation de la composition chimique de 4898 vins, exprimée sur 11 variables continues, ainsi qu'un score de qualité attribué à chaque vin [1]. La cible, la qualité, est une valeur continue, ce qui en fait un jeu de données de régression. La qualité ne suit pas une loi uniforme : 45% des vins ont une qualité égale à 6, 30% ont une qualité égale à 5, et 18% une qualité égale à 7. Ainsi,

1. https://archive.ics.uci.edu/ml/datasets/wine+quality

Algorithm 2 Algorithme *LapS3L* : prédiction

Donnée : $\mathcal{V} \in \mathbb{R}^{N,d}$, $\mathcal{V}_t \in \mathbb{R}^{n_t,d}$
Hyperparamètre : $\kappa \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Modèle : $U \in \mathbb{R}^{N,s}$
Modèle : $w \in \mathbb{R}^s$
1. Construire la matrice $K_b \in \mathbb{R}^{N,n_t}$:

$$\forall i, j, \quad K_{bi,j} = \kappa(\mathcal{V}_{i,.}, \mathcal{V}_{tj,.})$$

2. Poser $X_t := K_b' U$
Résultat : $\hat{z} \leftarrow X_t w$

les valeurs d'erreur de régression doivent être inférieures à la base suivante : en prédisant toujours 6, on obtient une *RSME* de 0.89 et une *MAE* de 0.63.

— *insurance*, aussi appelé COIL2000 [2], regroupe des profils de personnes ayant souscrit une assurance, établis sur 85 variables, et la cible est la variable binaire indiquant si la personne assurée a une assurance pour une caravane. Là encore, la cible est déséquilibrée, puisque seuls 6% des clients ont une assurance pour une caravane. La base consistant à prédire la valeur moyenne donne une *RMSE* de 0.237, et prédire 0 donne une *RMSE* de 0.244 et une *MAE* de 0.06.

— *temperature* décrit la température de la haute atmosphère, dans un jeu de données qui n'est pas public. Par conséquent, nous ne l'utilisons pas dans notre étude.

Comme dans l'article original, nous conservons de 2% à 9% des labels des jeux de données.

3.4.2 Protocole expérimental

L'algorithme *LapS3L* a plusieurs hyperparamètres :

— la définition du noyau, et éventuellement la valeur de la bande passante si c'est un noyau gaussien ;
— le nombre de composantes principales ;
— le régulariseur ridge α ;
— le régulariseur Laplacien β ;
— la définition du graphe des individus.

Les valeurs de ces hyperparamètres sont recherchées dans les ensembles suivants :

— noyau : l'ensemble composé du noyau utilisant la similarité cosinus, du noyau constitué par la matrice de Gram, et du noyau gaussien dont la bande passante σ prend les valeurs dans $\{0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100\}$;
— le nombre de composantes principales est sélectionné dans l'ensemble entre 1, 2, 5, 10, 20, 50, 100, 200. Une valeur supérieure à 200 nuit à l'apprentissage du *SSSL*, puisque la matrice de variables extraites X a un rang trop faible, d'une part, et le temps d'apprentissage est trop long, d'autre part ;

2. https://archive.ics.uci.edu/ml/datasets/Insurance+Company+Benchmark+(COIL+2000)

— le graphe des individus est défini comme suit : on commence par le construire complet, les arêtes étant valuées par la similarité cosinus entre individus dans l'ensemble des variables original, puis il est élagué de façon à ne conserver que les 10 plus proches voisins symétriques ;
— les deux régulariseurs sont choisis dans l'ensemble $\{(10^k)_{k=-4}^{4}\}$.

La procédure de tuning est la suivante :
— Répéter 10 fois les opérations suivantes :
  — Sélectionner 90% du jeu de données labellisé pour l'apprentissage, 10% pour le test (les données non labellisées sont utilisées pour l'apprentissage) ;
  — Répéter 10 fois (insurance) ou 20 fois (wine) :
    — Tirer avec remise une valeur pour chaque hyperparamètre,
    — Évaluer la performance (*RMSE*) de *LapS3L* et *SSSL* en validation croisée à 10 folds sur l'ensemble d'apprentissage labellisé, en y ajoutant les individus non labellisés ;
  — Sélectionner la meilleure valeur des hyperparamètres ;
  — Entraîner *SSSL* et *LapS3L* sur l'ensemble d'apprentissage en utilisant ces hyperparamètres ;
  — Tester sur l'ensemble de test : *RMSE*, *MAE*, *RRSE*, *RAE*.
— Retourner la moyenne et l'écart-type de chacune de ces métriques.

Le nombre de points du paramétrage est plus restreint pour le jeu de données *insurance*, car il comporte environ deux fois plus d'individus que *wine*.

Nous utilisons les quatre métriques suivantes pour évaluer la régression semi-supervisée, où z désigne le label de l'ensemble de test, $\hat{z}$ la prédiction, et $\bar{z}$ la valeur moyenne sur l'ensemble d'apprentissage :

— *RMSE* ou *Root Mean Squared Error*, racine de l'erreur quadratique moyenne :

$$\mathrm{RMSE} = \sqrt{\frac{1}{n_t} \|z - \hat{z}\|_2^2}$$

— *MAE* ou *Mean Absolute Error*, erreur absolue moyenne :

$$\mathrm{MAE} = \frac{1}{n_t} |z - \hat{z}|$$

— *RRSE* ou *Root Relative Squared Error*, racine de l'erreur quadratique relative,

$$\mathrm{RRSE} = \sqrt{\frac{\|z - \hat{z}\|_2^2}{\|z - \bar{z}\|_2^2}}$$

— *RAE* ou *Relative Absolute Error*, erreur en valeur absolue relative,

$$\mathrm{RAE} = \frac{|z - \hat{z}|_2^2}{|z - \bar{z}|_2^2}$$

TABLE 3.1 – Comparaison entre *LapS3L* et *SSSL* sur *wine* et *insurance*, métrique RMSE.

dataset	**LapS3L**	*SSSL*
insurance (2%)	**0.20 ± 0.03**	0.21 ± 0.03
insurance (3%)	**0.21 ± 0.02**	0.21 ± 0.02
insurance (4%)	**0.21 ± 0.02**	0.22 ± 0.02
insurance (5%)	**0.22 ± 0.02**	0.22 ± 0.02
insurance (6%)	**0.22 ± 0.02**	0.22 ± 0.02
insurance (7%)	**0.22 ± 0.02**	0.22 ± 0.02
insurance (8%)	**0.22 ± 0.02**	0.23 ± 0.02
insurance (9%)	**0.23 ± 0.01**	0.23 ± 0.02
wine (2%)	**0.77 ± 0.06**	0.82 ± 0.06
wine (3%)	**0.75 ± 0.05**	0.80 ± 0.05
wine (4%)	**0.77 ± 0.06**	0.82 ± 0.05
wine (5%)	**0.78 ± 0.05**	0.84 ± 0.06
wine (6%)	**0.78 ± 0.03**	0.84 ± 0.04
wine (7%)	**0.77 ± 0.04**	0.84 ± 0.03
wine (8%)	**0.77 ± 0.04**	0.84 ± 0.04
wine (9%)	**0.77 ± 0.04**	0.84 ± 0.04

TABLE 3.2 – Comparaison entre *LapS3L* et *SSSL* sur *wine* et *insurance*, métrique MAE.

dataset	**LapS3L**	*SSSL*
insurance (2%)	**0.07 ± 0.02**	0.12 ± 0.02
insurance (3%)	**0.07 ± 0.03**	0.11 ± 0.01
insurance (4%)	**0.09 ± 0.03**	0.11 ± 0.02
insurance (5%)	**0.09 ± 0.03**	0.11 ± 0.02
insurance (6%)	**0.09 ± 0.03**	0.11 ± 0.02
insurance (7%)	**0.10 ± 0.02**	0.11 ± 0.02
insurance (8%)	**0.10 ± 0.03**	0.11 ± 0.02
insurance (9%)	**0.11 ± 0.01**	0.11 ± 0.01
wine (2%)	**0.62 ± 0.05**	0.67 ± 0.05
wine (3%)	**0.60 ± 0.04**	0.65 ± 0.05
wine (4%)	**0.60 ± 0.05**	0.66 ± 0.04
wine (5%)	**0.61 ± 0.04**	0.67 ± 0.04
wine (6%)	**0.61 ± 0.03**	0.68 ± 0.03
wine (7%)	**0.61 ± 0.03**	0.67 ± 0.03
wine (8%)	**0.61 ± 0.03**	0.67 ± 0.04
wine (9%)	**0.61 ± 0.03**	0.67 ± 0.04

TABLE 3.3 – Comparaison entre *LapS3L* et *SSSL* sur *wine* et *insurance*, métrique RRSE.

dataset	**LapS3L**	*SSSL*
insurance (2%)	**0.71 ± 0.13**	1.04 ± 0.04
insurance (3%)	**0.86 ± 0.10**	1.03 ± 0.03
insurance (4%)	**0.91 ± 0.11**	1.02 ± 0.03
insurance (5%)	**0.95 ± 0.07**	1.01 ± 0.01
insurance (6%)	**0.96 ± 0.05**	1.00 ± 0.01
insurance (7%)	**0.99 ± 0.03**	1.00 ± 0.01
insurance (8%)	**0.97 ± 0.03**	1.00 ± 0.01
insurance (9%)	**0.99 ± 0.01**	1.00 ± 0.00
wine (2%)	**0.92 ± 0.05**	0.97 ± 0.03
wine (3%)	**0.90 ± 0.05**	0.96 ± 0.04
wine (4%)	**0.90 ± 0.03**	0.96 ± 0.02
wine (5%)	**0.90 ± 0.03**	0.96 ± 0.02
wine (6%)	**0.89 ± 0.02**	0.96 ± 0.01
wine (7%)	**0.89 ± 0.02**	0.96 ± 0.02
wine (8%)	**0.88 ± 0.02**	0.96 ± 0.02
wine (9%)	**0.88 ± 0.02**	0.96 ± 0.01

TABLE 3.4 – Comparaison entre *LapS3L* et *SSSL* sur *wine* et *insurance*, métrique RAE.

dataset	**LapS3L**	*SSSL*
insurance (2%)	**0.47 ± 0.17**	1.02 ± 0.02
insurance (3%)	**0.61 ± 0.25**	1.01 ± 0.02
insurance (4%)	**0.76 ± 0.28**	1.01 ± 0.02
insurance (5%)	**0.82 ± 0.25**	1.00 ± 0.02
insurance (6%)	**0.80 ± 0.25**	1.00 ± 0.01
insurance (7%)	**0.93 ± 0.15**	1.00 ± 0.01
insurance (8%)	**0.89 ± 0.20**	0.99 ± 0.01
insurance (9%)	**0.99 ± 0.01**	1.00 ± 0.02
wine (2%)	**0.95 ± 0.07**	1.01 ± 0.03
wine (3%)	**0.92 ± 0.07**	0.99 ± 0.03
wine (4%)	**0.91 ± 0.06**	1.00 ± 0.03
wine (5%)	**0.92 ± 0.05**	1.00 ± 0.03
wine (6%)	**0.92 ± 0.04**	1.01 ± 0.02
wine (7%)	**0.91 ± 0.03**	1.00 ± 0.03
wine (8%)	**0.91 ± 0.04**	1.00 ± 0.03
wine (9%)	**0.91 ± 0.03**	1.00 ± 0.03

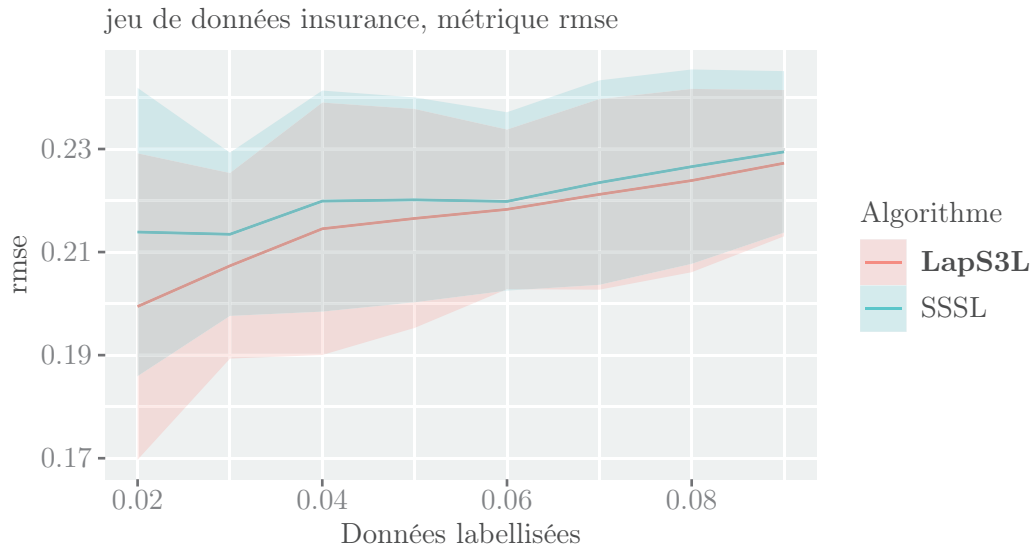


FIGURE 3.1 – Performance de *LapS3L* et *SSSL* sur *insurance* (métrique RMSE)

3.4.3 Résultats

Les résultats de la comparaison sont visibles dans les tables 3.1-3.4.

Pour la métrique RMSE (table 3.1), *LapS3L* et *SSSL* obtiennent une erreur de régression inférieure à 0.244 (pour insurance) et 0.89 (pour wine), qui étaient les lignes de base à dépasser. La métrique RMSE a été utilisée pour la recherche des hyperparamètres (ci-après dénotée *tuning*), ce qui n'est pas le cas pour la métrique MAE, dont la ligne de base n'est pas toujours respectée.

Plus précisément, la table 3.2 montre que *LapS3L* reste inférieur à la ligne de base pour wine (0.63), mais pas pour insurance (0.06). *SSSL* ne respecte aucune de ces lignes de base.

Sur les deux jeux de données, quel que soit le nombre d'individus labellisés (entre 2% et 9% du total) et quelle que soit la métrique envisagée, l'approche *LapS3L* donne un meilleur résultat que l'approche *SSSL*. Ce n'est pas surprenant, puisque *LapS3L* est une généralisation de *SSSL*. Cela signifie en revanche qu'un tuning à 10 points (respectivement 20) pour insurance (respectivement wine) est suffisant pour explorer un espace de recherche de la solution plus grand. Pour les métriques relatives, l'écart-type est plus faible pour *SSSL*, mais ce n'est pas très pertinent : en effet, pour les métriques relatives, les deux approches tendent à donner un résultat proche de 1. Par conséquent, *LapS3L* donnant parfois une erreur de régression bien plus faible, obtient un écart-type plus grand.

En regardant de plus près le jeu de données *insurance*, on s'aperçoit que les performances sont bonnes pour un petit nombre de données labellisées (cf figure 3.1). En effet, les limites imposées pour le tuning ne permettent pas de contrer le sur-apprentissage.

La métrique MAE, toujours sur le jeu de données *insurance* (figure 3.2), montre une

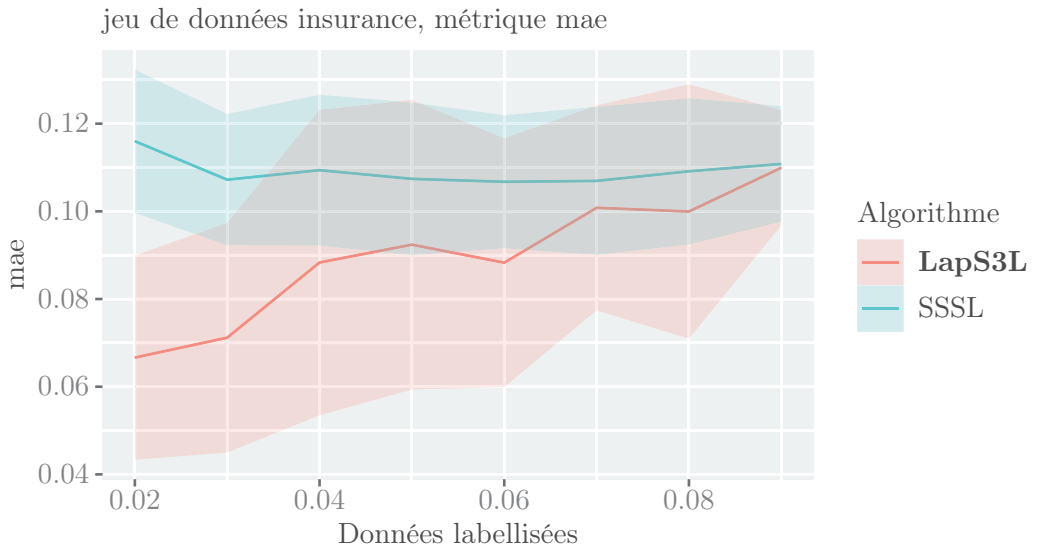


FIGURE 3.2 – Performance de *LapS3L* et *SSSL* sur *insurance* (métrique MAE)

tendance similaire pour *LapS3L*, mais pas pour *SSSL*, ce qui justifie la pertinence de notre approche.

Nous pouvons donc conclure que l'approche proposée, en tant que généralisation stricte de *SSSL*, ne nécessite pas un tuning plus précis pour donner de meilleures performances. Il nous reste à vérifier si notre approche fonctionne hors des hypothèses de fonctionnement de *SSSL*.

3.5 Étude comparative

Dans cette section, nous élargissons la comparaison des performance à différents algorithmes sur d'autres jeux de données publics divers.

3.5.1 Jeux de données

Nous utilisons les jeux de données publics suivants :

— *Concrete Compressive Strength* (*concrete*) [3] observe des échantillons de béton comparant leur composition et leur âge, afin d'en déduire la pression de compression. Ce jeu de données comprend 1030 échantillons pour 9 variables. Afin de pouvoir appliquer la fonction de noyau, toutes les variables du jeu de données sont normalisées de manière à avoir une moyenne nulle et un écart-type égal à 1 ;

— *Airfoil Self-Noise* (*airfoil*) [4] observe des hélices, afin d'en déduire le niveau de bruit. Il comprend 1503 observations pour 5 variables. Pour ce jeu de données,

3. https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength
4. https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise

toutes les variables ont été redimensionnées de manière à obtenir une moyenne nulle et un écart-type égal à 1 ;

— *Behavior of the urban traffic of the city of Sao Paulo in Brazil* (*behavior*) [5] mesure le retard du réseau comme la réponse à divers événements. Il ne comprend que 135 individus pour 18 variables ;

— *Communities and Crime Data Set* (*crimepredict*) [6] mesure le nombre de crimes violents dans un certain nombre de communes aux États-Unis. Il comprend 1994 individus pour 123 variables (en enlevant les variables d'index) décrivant les communes, y compris leur démographie. Ce jeu de données a été sélectionné parce qu'il contenait un nombre d'individus suffisamment faible pour l'apprentissage, avec des variables normalisées (ce qui permet d'appliquer directement une fonction de noyau).

3.5.2 Algorithmes

Dans la version non linéaire de l'algorithme *LapRLS*[2], il s'agit de minimiser la fonction objectif suivante :

$$\underset{f \in \mathcal{H}_\kappa}{\text{minimize}} \quad \left\| \hat{f}_l - y_l \right\|_2^2 + \alpha \left\| f \right\|_\kappa^2 + \beta \hat{f}' L \hat{f} \tag{3.15}$$

Ici, $\mathcal{H}_\kappa$ désigne un espace de Hilbert à noyau reproduisant. Dans cet espace, une fonction f s'écrit de la façon suivante :

$$x \mapsto \sum_{i=1}^{N} \alpha_i \kappa(x_i, x) \tag{3.16}$$

On note également $\hat{f}$ l'application de la fonction f sur le jeu d'apprentissage ($\hat{f}_l$ uniquement sur les individus labellisés). La fonction objectif peut être résolue de manière analytique en posant :

$$\alpha \leftarrow \left[JK + \alpha I + \beta LK \right]^{-1} JY \tag{3.17}$$

avec J la matrice diagonale de dimension N, N dont la diagonale en (i, i) vaut 1 si l'individu i est labellisé, 0 sinon.

En plus de la version non linéaire de *LapRLS*, ainsi que de *SSSL*, nous comparons avec une forêt aléatoire (*TB* pour *Tree Bagging*[56]) comprenant entre 1 et 1000 arbres, et l'algorithme *ν-SVR*[57].

La comparaison de ces algorithmes sur les 4 jeux de données donnent les résultats dans la table 3.5.

L'approche *LapS3L* donne de meilleurs résultats sur les jeux de données *behavior*, *concrete* et *crimepredict*. Seul le jeu de données *airfoil* est partagé entre *LapS3L* et *LapRLS*. Notons que le jeu de données *behavior*, qui comporte un très faible nombre

5. https://archive.ics.uci.edu/ml/datasets/Behavior+of+the+urban+traffic+of+the+city+of+Sao+Paulo+in+Brazil
6. https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

TABLE 3.5 – Comparaison de *LapS3L* contre d'autres algorithmes de régression, métrique RMSE

dataset	**LapS3L**	*LapRLS*	*SSSL*	*ν-SVR*	*TB*
airfoil (2%)	**0.65 ± 0.09**	0.73 ± 0.11	0.70 ± 0.13	0.78 ± 0.09	0.74 ± 0.07
airfoil (3%)	**0.67 ± 0.08**	0.69 ± 0.11	0.75 ± 0.10	0.80 ± 0.11	0.75 ± 0.08
airfoil (4%)	**0.64 ± 0.09**	0.66 ± 0.10	0.71 ± 0.07	0.78 ± 0.10	0.68 ± 0.07
airfoil (5%)	**0.62 ± 0.08**	0.63 ± 0.09	0.72 ± 0.05	0.77 ± 0.06	0.68 ± 0.06
airfoil (6%)	0.63 ± 0.06	**0.62 ± 0.07**	0.72 ± 0.04	0.77 ± 0.05	0.66 ± 0.05
airfoil (7%)	**0.59 ± 0.05**	0.60 ± 0.06	0.71 ± 0.05	0.75 ± 0.04	0.64 ± 0.04
airfoil (8%)	0.59 ± 0.04	**0.57 ± 0.06**	0.70 ± 0.05	0.74 ± 0.03	0.61 ± 0.04
airfoil (9%)	0.57 ± 0.03	**0.55 ± 0.05**	0.70 ± 0.05	0.73 ± 0.03	0.58 ± 0.03
behavior (2%)	**4.15 ± 2.73**	5.04 ± 2.58	*erreur*	*erreur*	5.26 ± 3.59
behavior (3%)	**2.64 ± 1.56**	3.84 ± 1.76	3.64 ± 3.68	3.08 ± 1.74	3.26 ± 1.81
behavior (4%)	**2.42 ± 0.82**	3.60 ± 1.48	3.98 ± 2.91	3.14 ± 0.97	3.09 ± 1.21
behavior (5%)	**2.48 ± 0.61**	3.39 ± 1.03	3.83 ± 2.25	2.61 ± 0.67	3.00 ± 0.92
behavior (6%)	**2.58 ± 0.74**	3.66 ± 0.60	3.64 ± 1.75	2.72 ± 0.53	3.06 ± 0.76
behavior (7%)	**2.49 ± 0.97**	3.52 ± 0.76	3.05 ± 1.19	2.76 ± 0.96	2.93 ± 0.88
behavior (8%)	**2.57 ± 0.83**	3.32 ± 0.68	3.13 ± 0.77	2.77 ± 0.87	2.79 ± 0.69
behavior (9%)	**2.55 ± 0.73**	3.35 ± 0.60	2.69 ± 0.81	2.61 ± 0.87	2.78 ± 0.52
concrete (2%)	**0.49 ± 0.12**	0.60 ± 0.14	0.60 ± 0.17	0.60 ± 0.10	0.61 ± 0.13
concrete (3%)	**0.46 ± 0.09**	0.57 ± 0.11	0.58 ± 0.14	0.59 ± 0.10	0.62 ± 0.10
concrete (4%)	**0.46 ± 0.12**	0.58 ± 0.12	0.53 ± 0.14	0.56 ± 0.12	0.61 ± 0.11
concrete (5%)	**0.49 ± 0.11**	0.59 ± 0.12	0.55 ± 0.13	0.56 ± 0.09	0.62 ± 0.10
concrete (6%)	**0.48 ± 0.08**	0.53 ± 0.09	0.53 ± 0.10	0.53 ± 0.08	0.59 ± 0.11
concrete (7%)	**0.47 ± 0.07**	0.52 ± 0.10	0.50 ± 0.09	0.51 ± 0.08	0.58 ± 0.09
concrete (8%)	**0.45 ± 0.07**	0.53 ± 0.09	0.47 ± 0.09	0.48 ± 0.06	0.55 ± 0.08
concrete (9%)	**0.43 ± 0.06**	0.47 ± 0.07	0.47 ± 0.07	0.47 ± 0.05	0.54 ± 0.07
crimepredict (2%)	**0.15 ± 0.02**	0.18 ± 0.03	0.15 ± 0.03	0.16 ± 0.02	0.15 ± 0.02
crimepredict (3%)	**0.15 ± 0.03**	0.17 ± 0.03	0.15 ± 0.03	0.16 ± 0.04	0.15 ± 0.02
crimepredict (4%)	**0.14 ± 0.02**	0.16 ± 0.02	0.14 ± 0.02	0.16 ± 0.03	0.14 ± 0.01
crimepredict (5%)	**0.14 ± 0.02**	0.17 ± 0.02	0.15 ± 0.02	0.17 ± 0.02	0.15 ± 0.02
crimepredict (6%)	**0.15 ± 0.01**	0.17 ± 0.01	0.15 ± 0.01	0.17 ± 0.02	0.15 ± 0.01
crimepredict (7%)	**0.14 ± 0.01**	0.17 ± 0.01	0.15 ± 0.01	0.17 ± 0.02	0.15 ± 0.01
crimepredict (8%)	**0.14 ± 0.01**	0.17 ± 0.01	0.14 ± 0.01	0.16 ± 0.02	0.14 ± 0.00
crimepredict (9%)	**0.14 ± 0.01**	0.16 ± 0.01	0.14 ± 0.01	0.16 ± 0.02	0.14 ± 0.01

d'individus, ne permet pas d'obtenir suffisamment de vecteurs supports pour le ν-SVR en ne conservant que 2% des labels, et ne permet pas à l'algorithme *SSSL* de trouver une solution, puisque la matrice de covariance est toujours trop faible.

3.6 Application aux données de réseau d'assainissement

Dans cette section, nous utilisons un jeu de données dont le but est la reconstitution de l'âge des canalisations des réseaux d'assainissement de la ville de Lyon. L'inspection des canalisations est une tâche complexe, et étant donné le faible nombre de canalisations inspectées chaque année, il est nécessaire de privilégier l'inspection des canalisations les plus dégradées.

Pour savoir quelles canalisations sont dégradées, il est nécessaire d'en connaître la date de pose [58, 59, 60], qui indique l'âge et les techniques employées (celles-ci ont évolué au cours du dernier siècle).

Parmi les 85766 canalisations recensées, 24% seulement ont une date de pose connue. Le jeu de données est composé de 4 variables catégorielles (forme, matériau, type d'effluent, et si la canalisation fait partie du réseau structurant), et 3 variables continues (largeur, hauteur, longueur).

3.6.1 Pré-traitement des données

D'après les registres de pose, il y a toujours eu chaque année une quantité comparable de canalisations posées, si l'on exclut les périodes de guerre. En revanche, les informations historiques qui nous sont parvenues ne nous permettent de connaître principalement que les dates de pose des canalisations récentes (voir figure 3.3). Le réseau ne s'est pas construit uniquement à partir de 1975 ; les dates de pose des canalisations anciennes (qui sont l'objectif principal de l'application) sont majoritairement inconnues.

En effectuant un apprentissage sur ces données, les canalisations anciennes sont majoritairement ignorées ou considérées comme des observations anormales. Or, ce sont précisément ces canalisations dont on veut connaître la date de pose.

Nous avons donc ré-échantillonné le jeu de données, afin de sélectionner 2000 conduites dont la date de pose suit une loi uniforme (en sachant qu'il existe des périodes entières pendant lesquelles aucune canalisation n'est connue ; la canalisation est donc tirée au bord de ces périodes), et 2000 canalisations parmi tout l'ensemble de données pour le jeu de données non labellisé.

3.6.2 Résultats

La métrique évaluée est la RMSE. Les trois algorithmes comparés sont *LapS3L*, *SSSL* et *LapRLS* (non-linéaire). L'erreur de régression en année est donnée par la table 3.6. Notre approche donne une erreur de régression plus faible.

L'évolution de la performance de l'algorithme en fonction du nombre de composantes s (figure 3.4) montre qu'il est nécessaire de conserver un grand nombre de composantes pour un bon fonctionnement de l'algorithme. C'est un avantage de *LapS3L*, puisque lorsque le

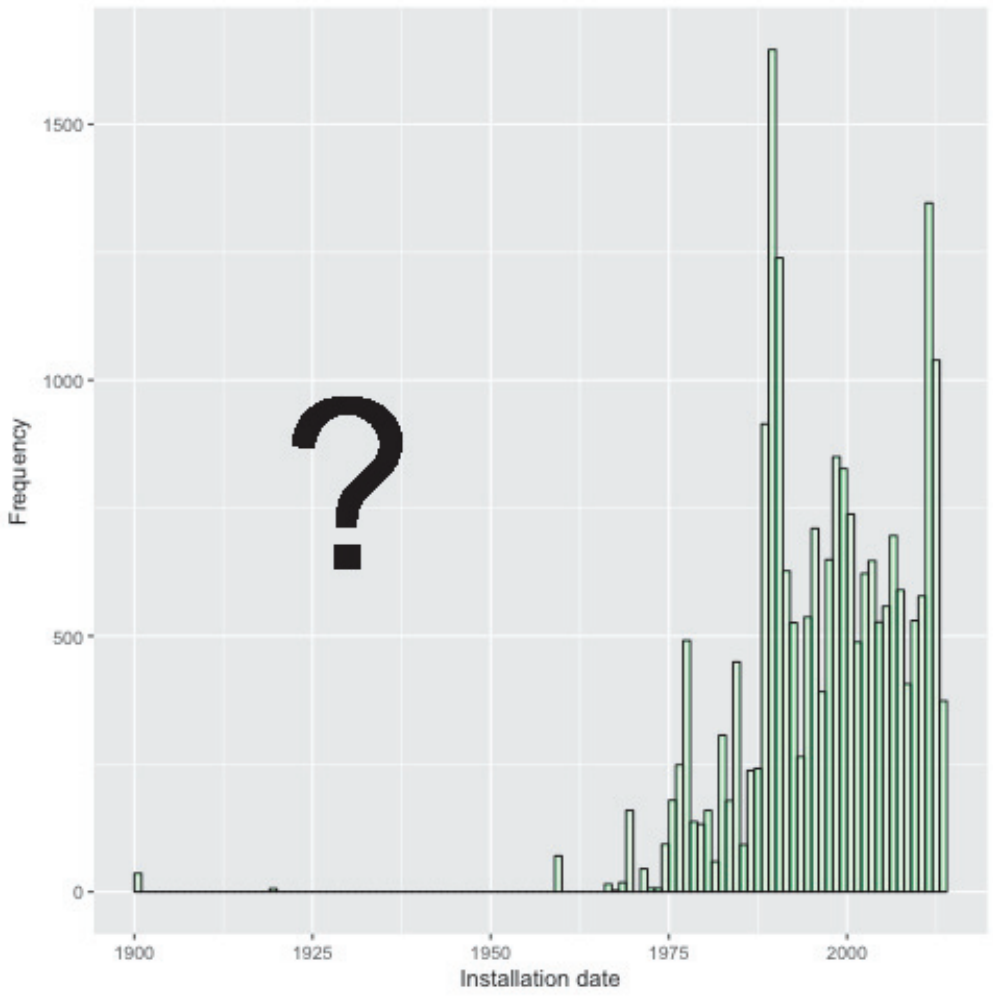


FIGURE 3.3 – Histogramme montrant la répartition des dates de pose des canalisations connues.

TABLE 3.6 – Métrique RMSE pour le jeu de données d'assainissement en année pour *LapS3L*, *SSSL*, *LapRLS*

LapS3L	*SSSL*	*LapRLS*
8.69	10.03	13.70

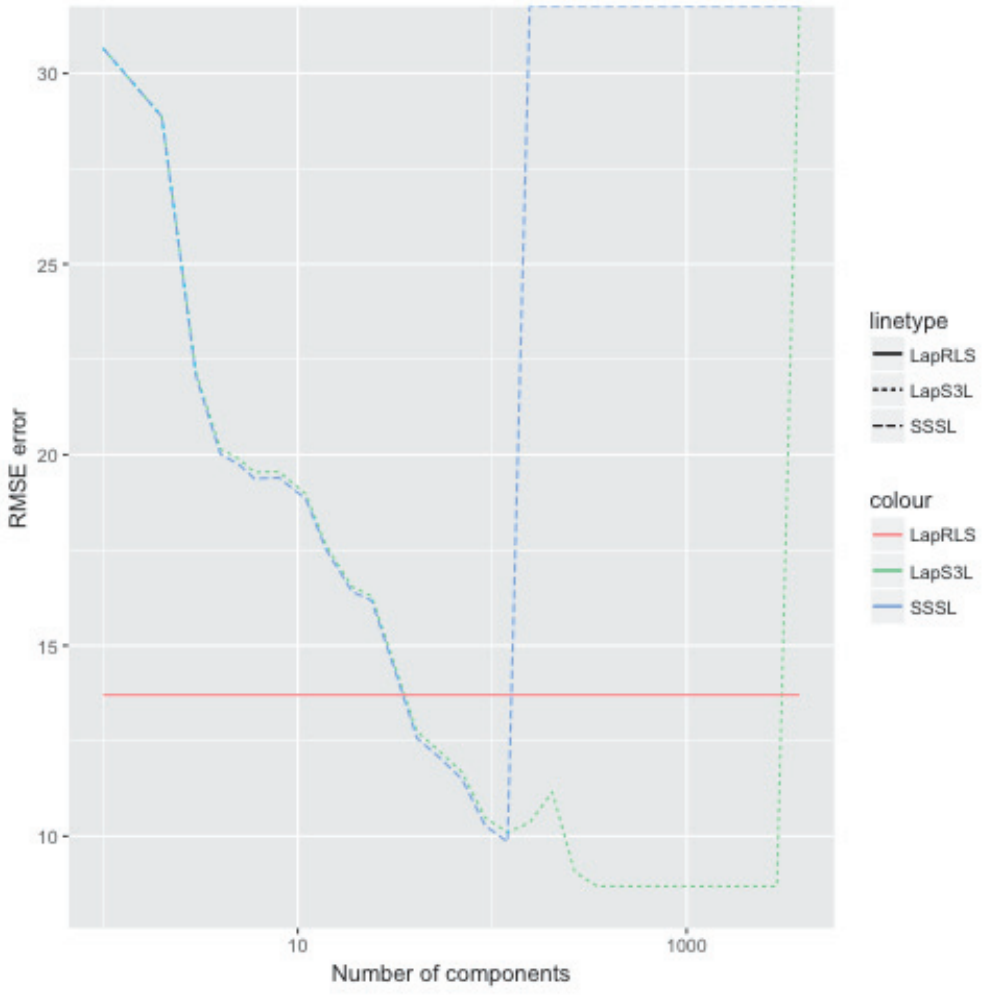


FIGURE 3.4 – Évolution de la performance de *LapS3L*, *SSSL*, et *LapRLS*, en fonction du nombre de composantes sélectionnées (en échelle logarithmique)

nombre de composantes extraites est supérieur au rang de la matrice de covariance, *SSSL* ne donne pas de solution, alors que la régularisation effectuée par *LapS3L* le permet.

L'erreur de régression de *LapRLS* est une constante, puisqu'elle ne dépend pas du nombre de composantes. L'erreur de régression pour *SSSL* diminue jusqu'à atteindre son minimum pour 100 composantes principales, puis elle monte très abruptement. L'erreur de régression de *LapS3L* suit celle de *SSSL*, remonte moins abruptement entre 100 et 200 composantes, puis baisse à nouveau jusqu'à 2000 composantes, en atteignant le minimum pour les trois algorithmes, après quoi elle augmente elle aussi très abruptement.

3.7 Conclusion

Nous avons proposé un algorithme de régression semi-supervisé, *LapS3L*, comme une extension stricte de l'algorithme *SSSL*. Par conséquent, au prix d'un *tuning* plus précis, *LapS3L* a donné de meilleures performances que son concurrent. Nous avons montré expérimentalement que pour un tuning de même complexité pour *SSSL* et *LapS3L*, *LapS3L* donne une plus faible erreur de régression sur les deux jeux de données publics utilisés dans l'article d'origine de *SSSL*. Nous avons montré également que *LapS3L* est compétitif sur d'autres jeux de données de régression. Enfin, nous avons montré expérimentalement l'importance de la régularisation employée dans un cas d'application réelle.

Chapitre 4

Régression semi-supervisée multi-labels : une approche régularisée

Dans ce chapitre, nous reprenons le travail décrit dans le chapitre précédent, pour l'étendre dans le cadre de l'apprentissage multi-labels. Nous menons également une étude expérimentale détaillée sur des jeux de données publics spécifiques.

L'idée de cette extension consiste à modifier la régularisation employée. En effet, il est possible d'employer des régularisations multi-labels, or la régularisation employée pour *LapS3L* ne l'est pas.

4.1 Introduction

L'un des défis principaux de l'apprentissage automatique moderne consiste à apprendre à partir de données labellisées à la main ainsi que de données non labellisées, qui sont généralement plus faciles à obtenir. L'apprentissage faiblement labellisé [43], et plus particulièrement l'apprentissage semi-supervisé [3, 4], abordent ce défi en utilisant soit des approches de propagation de labels pour suppléer des méthodes supervisées [4, 61], ou utilisent l'information des labels comme contraintes pour des méthodes non supervisées [44, 62].

De nombreuses méthodes semi-supervisées fondées sur des méthodes supervisées ont été proposées. Par exemple, on retrouve des adaptations telles que le *self-training*[16] ou *co-training*[20, 21], des cas d'apprentissage *transductif*[45, 46], ou des méthodes génératives [48].

Dans le cas particulier de la régression, avec une prédiction numérique, des applications spécifiques sont proposées [49, 53], et on retrouve principalement des approches fondées sur l'apprentissage de représentation de l'espace [1] ou utilisant un graphe entre individus tenant compte de la similarité [54], avec une insistance particulière sur la régularisation Laplacienne [51, 2].

De plus, l'apprentissage multi-labels vise à exploiter les relations entre *labels* pour apporter plus d'information à la tâche d'apprentissage. Des travaux théoriques ont été proposés [63], et de nombreuses méthodes multi-labels ont été proposées [64, 30]. Certaines stratégies transforment un problème multi-labels en plusieurs problèmes mono-label [65, 66], et il est parfois possible d'étendre directement le problème au cadre multi-labels, comme pour les processus gaussiens [67].

De nombreux algorithmes d'apprentissage multi-labels minimisent une fonction objectif régularisée. Ce peut être avec une régularisation non-convexe [40, 42], mais les approches convexes sont plus fréquentes, avec l'adaptation de l'algorithme LASSO [32] au cadre multi-labels [31, 11], ou le *Clustered Multik-Task Learning*[38], ou d'autres approches d'apprentissage de représentation de l'espace des labels [68, 69].

Pour l'apprentissage de régression, les méthodes spécifiques sont plus rares. Il est d'usage de modifier la fonction objectif pour utiliser des labels à valeurs réelles [34].

4.2 Travaux liés

Pour l'apprentissage de régression mono-label, nous avons démontré l'intérêt d'utiliser une version régularisée de l'algorithme *SSSL*[1] grâce à la régularisation Laplacienne. La caractéristique principale de cette famille d'algorithmes réside dans leur fonctionnement en deux étapes, qui permettent dans un premier temps de faire un changement d'espace *non-supervisé* propice à l'apprentissage de régression de certains jeux de données, puis d'adopter une *régression linéaire* dans ce nouvel espace.

Pour rappel, l'utilisation de la régularisation Laplacienne dans le nouvel espace permet de lier les deux espaces de description des individus, dans le but de satisfaire l'hypothèse suivante :

> Si deux individus sont proches dans l'espace réel, alors l'écart de prédiction doit être faible.

Nous avons montré expérimentalement que la complexité supplémentaire due à l'introduction de cette régularisation ne nécessite pas de procédure de tuning plus complexe.

Comme évoqué ci-dessus, il est aussi possible d'utiliser la régression Laplacienne pour traiter le problème d'apprentissage multi-labels [34]. Dans cette approche, on construit un graphe des labels, et il s'agit de régulariser le modèle de façon à introduire l'hypothèse suivante :

> Si deux labels sont similaires, les valeurs du modèle pour ces deux labels doivent être similaires.

4.3 Approche proposée : *LSMR*

4.3.1 Notations

Les individus sont décrits dans l'espace des variables supposé être de dimension d, $\mathbb{R}^d$. L'ensemble des labels est de dimension m. Étant donné que le problème est celui de la régression, cet espace est décrit dans $\mathbb{R}^m$.

Notons $\mathcal{V} \in \mathbb{R}^{N,d}$ la matrice de données, dont les lignes correspondent aux individus. On note n_l le nombre d'individus labellisés au total.

Associée à cette matrice de données, la matrice de labels de régression $Y \in \mathbb{R}^{N,m}$ contient une ligne par individu ; seules les lignes correspondant aux individus labellisés sont renseignées. On suppose que tous les labels sont manquants simultanément : si un individu est labellisé, les valeurs de tous les labels sont renseignés. Sinon, aucune valeur n'est renseignée.

4.3.2 Première étape : changement d'espace

Comme pour l'algorithme *SSSL* et *LapS3L*, une première étape consiste à choisir une fonction noyau symétrique, $\kappa \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, et à construire la matrice noyau entre tous les individus de l'ensemble d'apprentissage :

$$\forall i, j \in \{1, ..., N\}, \quad K_{i,j} = \kappa(\mathcal{V}_{i,.}, \mathcal{V}_{j,.}) \tag{4.1}$$

K étant une matrice symétrique réelle, on peut en extraire les s valeurs propres réelles les plus élevées, pour $s \leq N$, et les vecteurs propres associés. Ces derniers forment une matrice $U \in \mathbb{R}^{N,s}$, dont les colonnes sont de norme unitaire, où chaque ligne correspond à un individu de l'ensemble d'apprentissage. Étant donné que l'on n'utilise pas la matrice de labels, cette première étape est non supervisée.

Le changement d'espace s'obtient en posant :

$$X \leftarrow KU \tag{4.2}$$

La matrice X comprend donc une ligne par individu, et nous pouvons en extraire la sous-matrice correspondant aux individus labellisés comme $X_l \in \mathbb{R}^{n,s}$.

4.3.3 Régularisation semi-supervisée multi-labels

En reprenant l'algorithme *LapS3L*, la régression semi-supervisée qui définit la seconde étape de l'algorithme utilise deux termes de régularisation :

— une régularisation *Laplacienne*,

— une régularisation *Ridge*, qui pénalise la complexité du modèle.

Nous proposons de remplacer le terme de régularisation *Ridge* en régularisation Laplacienne multi-labels. Ainsi, en définissant le graphe des labels par la matrice d'adjacence $M_m \in \mathbb{R}^{m,m}$, et le graphe des individus par la matrice d'adjacence $M_s \in \mathbb{R}^{N,N}$, le terme à pénaliser devient :

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|X_l W - Y_l\|_F^2 + \alpha \text{tr}(W' X' L_s X W) + \beta \text{tr}(W L_m W') \tag{4.3}$$

où :

— α est le régulariseur semi-supervisé ;

— β est le régulariseur multi-labels ;

— L_s est la matrice Laplacienne du graphe des individus,

$$L_s = D_s - M_s$$

avec D_s la matrice de degré du graphe des individus,

$$\forall i \in \{1, ..., N\}, \quad D_s(i,i) = \sum_{j=1}^{N} M_s(i,j)$$

— L_m est la matrice Laplacienne du graphe des labels,

$$L_m = D_m - M_m$$

avec D_m la matrice de degré du graphe des labels,

$$\forall k \in \{1, ..., m\}, \quad D_m(k,k) = \sum_{l=1}^{m} M_m(k,l)$$

La pénalisation est rendue explicite sur des exemples figure 4.1. En haut, le graphe est construit à partir de 4 individus ; la proximité dans l'espace des variables est indiquée par une arête. En bas, le graphe est construit à partir de 3 labels ; une arête relie deux labels ayant souvent des valeurs similaires sur beaucoup d'individus. Pour l'apprentissage semi-supervisé, en haut, les labels sont considérés de manière indépendante. On note à l'intérieur du cercle la valeur prédite par le modèle pour un label. Pour l'apprentissage multi-labels, en bas, chaque label a un modèle linéaire, dont les poids sont représentés sur une colonne. La valeur de chaque poids du modèle est représentée par une couleur différente. Dans la partie gauche de la figure, la pénalisation est faible, puisque le modèle donne des valeurs similaires aux individus liés (en haut) et les modèles sont similaires pour des labels reliés (en bas). À droite, la pénalisation est plus forte.

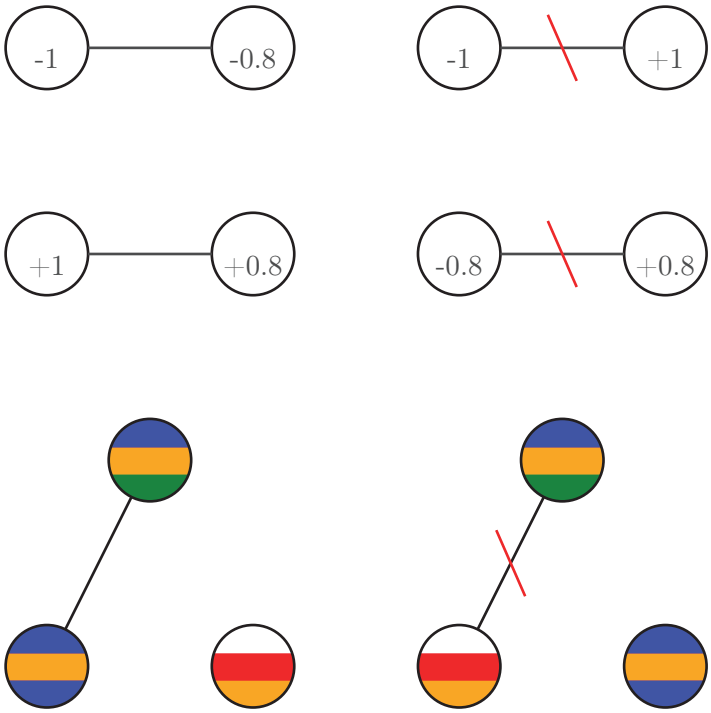


FIGURE 4.1 – Exemples de pénalisation pour l'apprentissage semi-supervisé (en haut) et l'apprentissage multi-labels (en bas)

Dans la suite, nous appellerons la modification proposée *LSMR*, pour *Laplacian-regularized Simple algorithm for Semi-supervised multi-labels Regression*. La modification proposée de l'algorithme *LapS3L* consiste toujours en une extension de l'algorithme *SSSL*, mais sa différence avec *LapS3L* réside dans un autre choix pour la régularisation Ridge.

4.4 Algorithme d'optimisation

La fonction objectif employée dans l'algorithme *LapS3L* admet une solution analytique, qui permet d'obtenir la valeur du modèle W grâce à une résolution d'un système linéaire de dimension $s \times s$. Malheureusement, on ne peut pas l'appliquer directement pour *LSMR*, à moins d'avoir à résoudre m^2 systèmes linéaires distincts. Cette limitation est commune à beaucoup de méthodes de régularisation multi-labels, ce qui pousse les méthodes de MALSAR à adopter une méthode d'optimisation différente.

4.4.1 Descente de gradient

La fonction objectif obtenue dans 4.3 présente les caractéristiques suivantes :
— la variable d'optimisation, W, est réelle ;
— la fonction objectif est convexe, si les arêtes des matrices d'adjacence des graphes des individus et des labels sont à poids positifs ;
— la fonction objectif est lisse.

Par conséquent, le problème peut être résolu par descente de gradient, dont le calcul est donné par (4.4).

$$2\left[X_l' X_l + \alpha X' L_s X\right] W - 2X_l' Y_l + 2\beta W L_m \tag{4.4}$$

4.4.2 Pas d'apprentissage

L'algorithme de descente du gradient produit des itérations de descente de l'erreur d'apprentissage jusqu'à convergence de la variable d'optimisation. Chaque itération consiste à calculer la valeur du gradient au point donné pour la valeur courante de la variable d'optimisation, ∇_W (selon (4.4)), puis poser :

$$W \leftarrow W - \eta \nabla_W \tag{4.5}$$

où η est le *pas d'apprentissage*. Dans notre cas, le gradient 4.4 est une fonction Lipschitzienne : pour deux valeurs quelconques du modèle, $P, Q \in \mathbb{R}^{s,m}$,

$$\|\nabla_W(P) - \nabla_W(Q)\|_F^2 \leq C \|P - Q\|_F^2 \tag{4.6}$$

avec :

$$C = 2\left(\rho\left(X_l' X_l + \alpha X' L_s X\right) + \beta\rho(L_m)\right) \tag{4.7}$$

Pour toute matrice M symétrique réelle, $\rho(M)$ désigne le rayon spectral de M, c'est-à-dire sa plus grande valeur propre. En posant :

$$\eta \leftarrow \frac{1}{C} \tag{4.8}$$

La convergence de l'algorithme de descente de gradient est assurée [70].

4.4.3 Initialisation du modèle

Le modèle $W \in \mathbb{R}^{s,m}$ est initialisé comme la solution du problème de régression linéaire, avec un terme de régularisation Ridge utilisant une valeur de régulariseur faible.

$$W \leftarrow \left[X_l' X_l + \epsilon I_s\right]^{-1} X_l' Y_l \tag{4.9}$$

avec ϵ faible (pour l'implémentation, nous avons retenu $\epsilon = 10^{-6}$), et $I_s \in \mathbb{R}^{s,s}$ la matrice identité en dimension s.

4.4.4 Descente de gradient accélérée

L'algorithme de descente de gradient accélérée [70] est une amélioration de l'algorithme de descente de gradient originale qui propose une convergence plus rapide. Pour notre application, l'algorithme d'optimisation est résumé dans l'algorithme 3. La prédiction s'effectue ainsi avec l'algorithme 4, qui est simplement la version multi-labels de l'algorithme de prédiction de l'algorithme *LapS3L* développé dans le chapitre précédent, en remplaçant le modèle de dimension s par un modèle de dimension $s \times m$.

4.5 Étude comparative

Afin de s'assurer de la performance de notre approche, nous proposons deux études expérimentales. Tout d'abord, puisqu'elle généralise une approche existante, nous devons nous assurer que cette extension est nécessaire. Puis nous montrerons quelques comparaisons avec d'autres régularisations pour de la régression multi-labels.

4.5.1 Jeux de données utilisés

Nous avons utilisé des jeux de données du projet MULAN [1], décrits dans [30], à quoi nous avons ajouté un échantillon de 1000 individus du jeu de données SARCOS [29]. Nous n'avons pas conservé tous les jeux de données, les plus grands contenant trop d'individus pour appliquer directement l'algorithme LSMR. Les jeux de données et leurs caractéristiques sont résumés dans la table 4.1.

Nous avons divisé les jeux de données en une partie pour l'apprentissage et une partie pour le test. Ces jeux de données étant labellisés, nous avons sélectionné 30% des individus de l'ensemble d'apprentissage pour en supprimer les labels, pour tous les labels simultanément. Le nombre de labels enlevé est bas, surtout comparé au travail effectué

1. http://mulan.sourceforge.net/datasets-mtr.html

Algorithm 3 Algorithme *LSMR* : apprentissage

Donnée : $\mathcal{V} \in \mathbb{R}^{N,d}$
Donnée : $Y \in \mathbb{R}^{n_l,m}$
Hyperparamètre : $s \in \{1, ..., N\}$
Hyperparamètre : $\kappa \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Hyperparamètre : matrice Laplacienne du graphe des individus, $L_s \in \mathbb{R}^{N,N}$
Hyperparamètre : matrice Laplacienne du graphe des labels, $L_m \in \mathbb{R}^{m,m}$
Hyperparamètre : $\alpha > 0$
Hyperparamètre : $\beta > 0$
1. Construire la matrice $K \in \mathbb{R}^{N,N}$:

$$\forall i, j, \quad K_{i,j} = \kappa(\mathcal{V}_{i,.}, \mathcal{V}_{j,.})$$

2. Décomposer la matrice K en valeurs propres et vecteurs propres, sélectionner les s vecteurs propres ayant la plus grande valeur propre associée : $U \in \mathbb{R}^{N,s}$
3. Poser $X := KU$
4. Sélectionner la sous-matrice $Xl \in \mathbb{R}^{n,s}$ composée des lignes de X correspondant aux individus labellisés
5. Appliquer l'algorithme de descente de gradient accélérée, en utilisant les paramètres suivants :
5.a. Pas d'apprentissage : $\frac{1}{C}$, $C = 2\left(\rho\left(X_l' X_l + \alpha X' L_s X\right) + \beta\rho(L_m)\right)$
5.b. Modèle initial : $W \in \mathbb{R}^{s,m} = \left[X_l' X_l + \epsilon I_s\right]^{-1} X_l' Y_l$
5.c. Calcul du gradient : $\nabla_W \leftarrow 2\left[X_l' X_l + \alpha X' L_s X\right] W - 2X_l' Y_l + 2\beta W L_m$
Résultat : $U \in \mathbb{R}^{N,s}$
Résultat : $W \in \mathbb{R}^{s,m}$

Algorithm 4 Algorithme *LSMR* : prédiction

Donnée : $\mathcal{V} \in \mathbb{R}^{N,d}$, $\mathcal{V}_t \in \mathbb{R}^{n_t,d}$
Hyperparamètre : $\kappa \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Modèle : $U \in \mathbb{R}^{N,s}$
Modèle : $W \in \mathbb{R}^{s,m}$
1. Construire la matrice $K_b \in \mathbb{R}^{N,n_t}$:

$$\forall i, j, \quad K_{bi,j} = \kappa(\mathcal{V}_{i,.}, \mathcal{V}_{tj,.})$$

2. Poser $X_t := K_b' U$
Résultat : $\hat{Y} \leftarrow X_t W$

TABLE 4.1 – Jeux de données utilisés pour l'étude expérimentale de notre approche, **LSMR**

Jeu de données	**Nb. indiv. (avec label + sans)**	**Test**	**Variables**	**Labels**
atp1d	262 (76 + 186)	165	411	6
atp7d	234 (67 + 167)	147	411	6
edm	121 (35 + 86)	73	16	2
enb	601 (173 + 428)	366	8	2
jura	281 (81 + 200)	173	15	3
oes10	314 (91 + 223)	198	298	16
oes97	257 (75 + 182)	163	263	16
osales	495 (144 + 351)	309	401	12
sarcossub	779 (225 + 554)	467	21	7
scpf	889 (256 + 633)	521	23	3
sf1	250 (73 + 177)	158	31	3
sf2	832 (240 + 592)	501	31	3
wq	827 (238 + 589)	487	16	14

pour *LapS3L*. Cependant le nombre d'individus total étant relativement faible, il n'est pas possible d'en enlever davantage.

Nous avons centré et réduit toutes les variables, et tous les labels, en soustrayant la moyenne et divisant par l'écart-type. Ceci garantit que l'on peut appliquer à la fois une fonction noyau entre les individus à partir des variables, et à la fois une fonction noyau entre les labels à partir des données labellisées. D'autre part, si les valeurs de l'un des labels sont négligeables devant les valeurs d'un autre, le calcul des métriques multi-labels risque de ne pas montrer la pertinence des algorithmes en tant qu'algorithmes multi-labels.

4.5.2 Protocole expérimental

La procédure de tuning consiste à tirer une valeur pour chaque hyperparamètre, selon la recherche aléatoire [71]. Cette méthode permet d'éviter l'écueil de la *recherche de grille*, qui considère tous les hyperparamètres comme équitablement importants. Dans le cas où l'algorithme présente de nombreux hyperparamètres, comme par exemple **LSMR**, la recherche aléatoire est à préférer.

Les hyperparamètres donnant la meilleure métrique aRMSE en validation croisée à 10 *folds* sont retenus. Pour rappel, la métrique aRMSE est définie par :

$$\text{aRMSE} = \frac{1}{m}\sum_{k=1}^{m}\sqrt{\frac{\left\|Y - \hat{Y}\right\|_F^2}{n_t}} \tag{4.10}$$

$\hat{Y}$ désignant la prédiction sur le jeu de test, Y les vrais labels du jeu de test, et n_t le nombre d'individu du jeu de test.

TABLE 4.2 – Valeur des hyperparamètres minimisant l'erreur de régression de l'algorithme *SSSL* sur les jeux de données étudiés

Jeu de données	Noyau	Nombre de composantes
atp1d	RBF, $\gamma = 3.140 \cdot 10^{-05}$	202
atp7d	produit scalaire	5
edm	RBF, $\gamma = 1.703 \cdot 10^{+01}$	107
enb	similarité cosinus	10
jura	RBF, $\gamma = 4.272 \cdot 10^{-03}$	87
oes10	produit scalaire	31
oes97	RBF, $\gamma = 3.785 \cdot 10^{-05}$	55
osales	RBF, $\gamma = 1.254 \cdot 10^{-02}$	475
scpf	similarité cosinus	2
sf1	produit scalaire	5
sf2	similarité cosinus	3
wq	RBF, $\gamma = 1.545 \cdot 10^{+01}$	774

La métrique doit être minimisée. Puisque l'on a normalisé les labels individuellement, si les labels du jeu de test suivent exactement la même distribution que le jeu d'apprentissage, la métrique est égale à 1 en prédisant toujours 0.

La validation croisée en 10 *folds* consiste à partitionner les données labellisées de l'ensemble d'apprentissage en 10 échantillons. L'apprentissage se fait sur 9 échantillons, plus toutes les données non labellisées. Le test se fait sur l'échantillon restant, en calculant la métrique aRMSE. En répétant l'opération sur les 10 échantillons pour le test, la métrique aRMSE moyenne est calculée.

Notre approche utilise 4 hyperparamètres de différentes natures :

1. La fonction noyau, κ ;
2. Le nombre de composantes principales, s ;
3. Le régulariseur semi-supervisé, α ;
4. Le régulariseur multi-labels, β.

Comme pour l'algorithme *LapS3L*, nous étudions différentes fonction de noyau :
— le produit scalaire ;
— la similarité cosinus ;
— le noyau RBF.

4.5.3 Tuning local

En tant qu'extension de l'algorithme *SSSL*, nous vérifions la pertinence des hyperparamètres introduits. Nous commençons par rechercher le noyau et le nombre de composantes pour minimiser l'erreur de régression de l'algorithme *SSSL*. Les valeurs obtenues sont résumées dans la table 4.2.

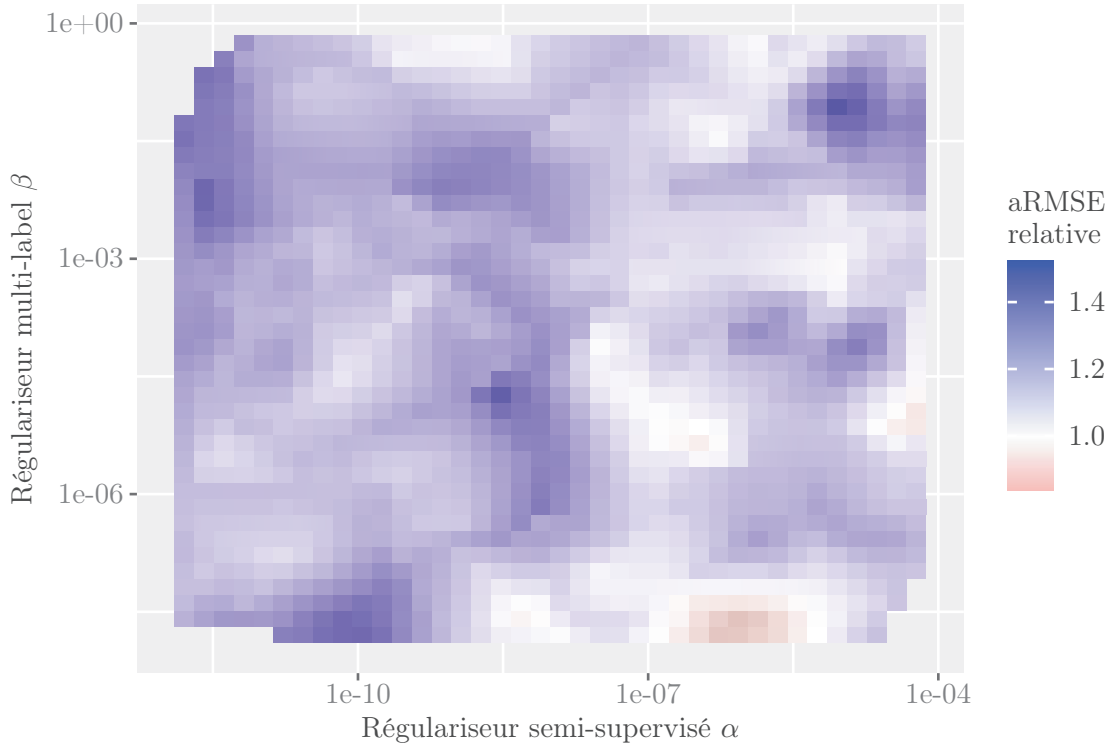


FIGURE 4.2 – aRMSE relative pour le tuning local de *LSMR* sur le jeu de données *scpf*.

En réutilisant ces valeurs, nous cherchons celles pour les deux autres hyperparamètres. Nous comparons avec la performance de *SSSL*, en calculant l'aRMSE relative comme le rapport entre la performance de *LSMR* et la performance de *SSSL*. Nous obtenons plusieurs cas de figure différents, en fonction des jeux de données. Les principaux types sont reportés dans la suite.

Tuning local : *scpf*

Le jeu de données *scpf* regroupe des notices émises par des habitants de certaines villes aux États-Unis à destination de la municipalité. Ces notices sont décrites par différentes variables, telles que la nature et la localisation du problème signalé. Le but est de comprendre l'importance du problème, en prédisant le nombre de vues, de clics et de commentaires.

Le tuning local donne la figure 4.2. Le régulariseur multi-labels n'est pas très pertinent dans ce cas ; il faut utiliser une valeur très faible pour obtenir une amélioration de l'algorithme *SSSL*. La zone où le tuning donne de bonnes performances est très restreinte.

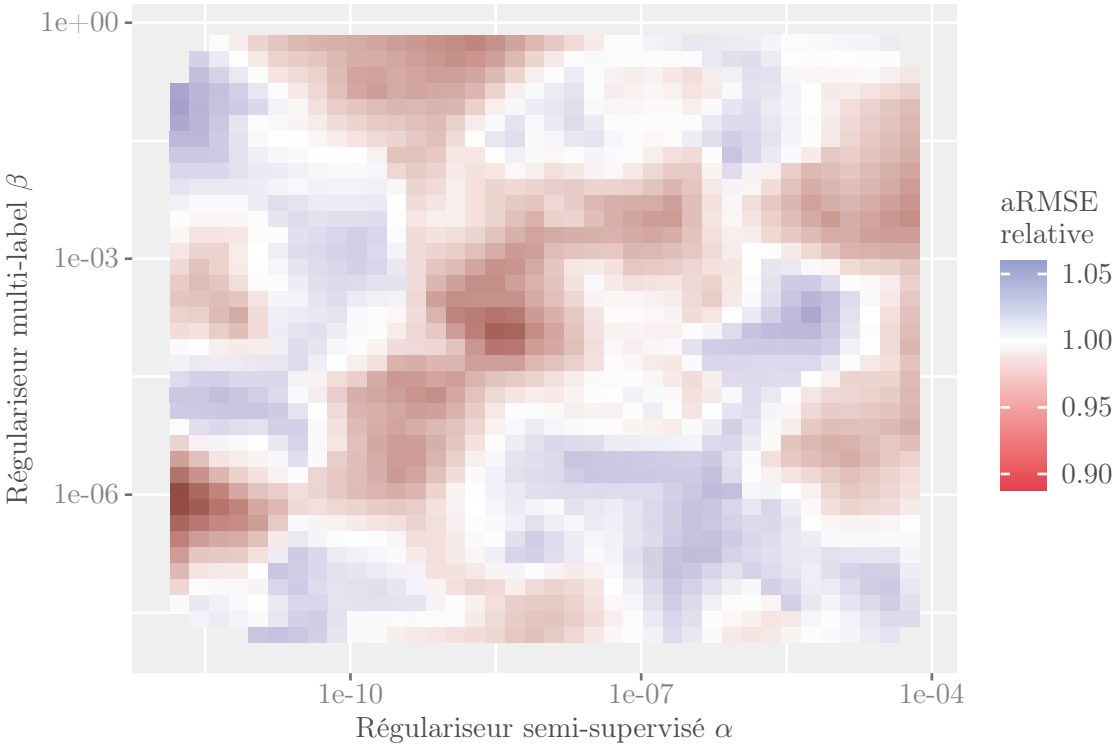


FIGURE 4.3 – aRMSE relative pour le tuning local de *LSMR* sur le jeu de données *osales*.

Tuning local : *osales*

Le jeu de données *osales* contient des produits ayant eu droit à une campagne publicitaire. L'objectif est de prédire le volume de vente de ces produits sur les 12 prochains mois, un label par mois.

Le tuning local donne la figure 4.3. La zone de tuning des hyperparamètres α et β donne une place plus large pour lequel *LSMR* améliore les résultats de *SSSL*, mais le meilleur point de tuning se situe pour un regulariseur semi-supervisé faible, avec une valeur non nulle du régulariseur multi-labels.

Tuning local : *sf2*

Le jeu de données *sf2* considère comme individu une période de 24h, et compte le nombre d'éruptions solaires de différentes catégories dans cette période. La deuxième version utilise des mesures prises en 1978.

Le tuning local donne la figure 4.4. La zone de tuning est là aussi plus étendue, mais le résultat est meilleur pour une grande valeur de chacun des deux hyperparamètres.

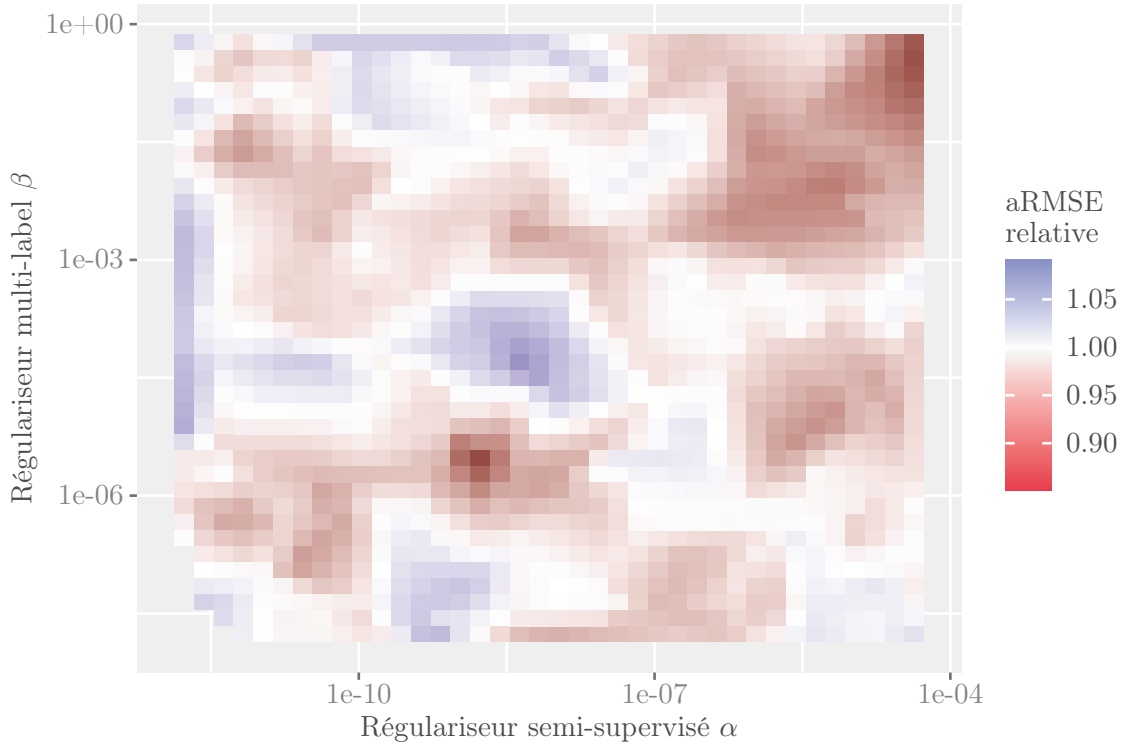


FIGURE 4.4 – aRMSE relative pour le tuning local de *LSMR* sur le jeu de données *sf2*.

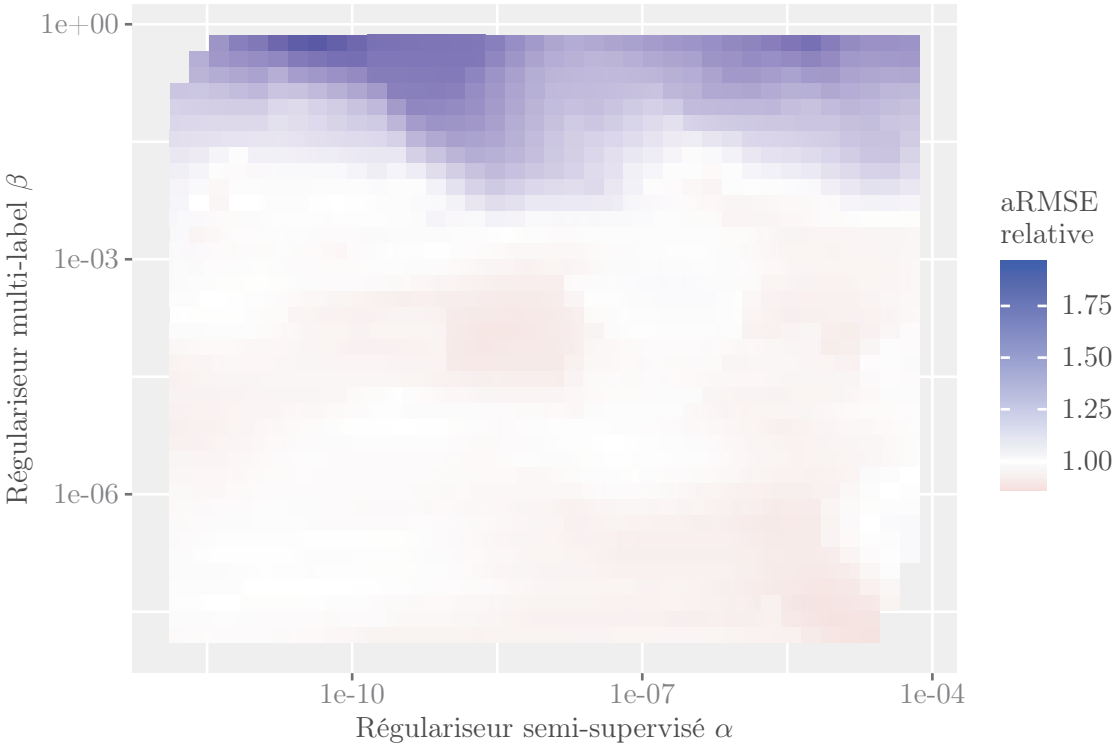


FIGURE 4.5 – aRMSE relative pour le tuning local de *LSMR* sur le jeu de données *oes97*.

Tuning local : *oes97*

Enfin, *oes97* étudie différentes villes des États-Unis (en 1997) et étudie la répartition des emplois dans ces zones. Le but est de faire la prédiction de certaines catégories d'emploi en fonction des autres pour une ville donnée.

Le tuning local donne la figure 4.5. La zone de tuning est trop étendue ; une valeur trop élevée du régulariseur multi-labels fait chuter la performance de l'algorithme.

Résultats du tuning local

En agrégeant les performances relatives sur tous les points de tuning, on obtient la table 4.3. Celle-ci montre que beaucoup de points du tuning local donnent un meilleur résultat que *SSSL*. L'agrégation des scores relatifs se fait avec la moyenne, le premier et troisième quartile, le minimum et le maximum. Sur tous les jeux de données, il existe un point où *LSMR* est bien meilleur que *SSSL* (valeur inférieure à 1), et un point où *LSMR* est pire. En regardant les quartiles, il existe de larges zones dans lesquelles *LSMR* tuné est meilleur que *SSSL*.

TABLE 4.3 – Tuning local : aRMSE moyenne relative de **LSMR** par rapport à *SSSL*

Jeu de données	Moyenne	Q1	Q3	Meilleur point	Pire point
sf2	**0.983**	**0.898**	1.086	**0.553**	1.245
scpf	1.191	**0.862**	1.554	**0.564**	1.684
osales	**0.989**	**0.938**	1.046	**0.635**	1.191
oes97	1.078	**0.872**	1.170	**0.662**	2.344
sf1	**0.994**	**0.922**	1.062	**0.682**	1.206
oes10	1.015	**0.908**	1.110	**0.715**	1.448
jura	1.014	**0.954**	1.065	**0.799**	1.485
atp1d	1.048	**0.958**	1.070	**0.812**	1.922
edm	**0.998**	**0.964**	1.037	**0.821**	1.190
atp7d	1.006	**0.961**	1.053	**0.833**	1.242
enb	**0.998**	**0.964**	1.027	**0.864**	1.124
wq	**0.999**	**0.992**	1.007	**0.960**	1.032

4.5.4 Tuning global

Nous mettons maintenant en place une approche de tuning global de façon à comparer notre approche proposée à d'autres approches de régularisation multi-labels pour la régression. Les méthodes évoquées résolvent un problème de régression multi-labels linéaire, en optimisant une fonction objectif convexe, au moyen de l'algorithme de descente de gradient accélérée.

Ligne de base : sans régularisation

Notre ligne de base n'effectue pas de régularisation, elle est notée *LSQ* (pour *least squares*).

Régularisation Lasso

La régularisation l_1 permet d'obtenir des parties du modèle ayant une valeur nulle [32]. Nous le notons *MTL* (*multi-task learning*). La régularisation est donnée par :

$$\|W\|_1 = \sum_{j=1}^{d} \sum_{k=1}^{m} |W_{j,k}| \tag{4.11}$$

Régularisation Laplacienne

C'est la régularisation employée dans l'algorithme *LSMR*. On la désigne par *SGR*, pour *sparse graph regularization*[34]. Le graphe est construit avec des poids utilisant la similarité cosinus entre les labels, d'après les valeurs qu'ont les individus. Avec M la matrice d'adjacence, D la matrice de degré et L la matrice Laplacienne du graphe, la régularisation est :

$$\mathrm{tr}\left(WLW'\right) = \sum_{k_1,k_2=1}^{m} M_{k_1,k_2} \left\|W_{.,k_1} - W_{.,k_2}\right\|_2^2 \tag{4.12}$$

Régularisation Lasso par groupe

L'algorithme *JFS* (*joint feature selection*) tel que défini par [31] utilise le Lasso par groupe. La régularisation employée est :

$$\|W\|_{2,1} = \sum_{j=1}^{d} \|W_{j,.}\|_2^2 \tag{4.13}$$

Régularisation du rang

La régularisation du rang peut s'effectuer en régularisant la norme trace [36], *TNR* (*trace norm regularization*), c'est-à-dire la somme de toutes les valeurs propres. La régularisation employée est :

$$\|W\|_* = \sum_{\sigma \text{ valeur propre de } W} \sigma \tag{4.14}$$

Dirty model

La régularisation du *Dirty Model*[33] (*DM*) est en deux termes, pour un modèle qui se décompose en $W = P + Q$:

$$\alpha \|P\|_{1,1} + \beta \|Q\|_{\infty,1} = \alpha \sum_{j=1}^{d} \sum_{k=1}^{m} |P_{j,k}| + \beta \sum_{j=1}^{d} \|Q_{j,.}\|_\infty \tag{4.15}$$

Clustered multi-task learning

CMTL est la version convexe de cet algorithme [38]. L'optimisation se fait sur deux variables, le modèle W et une matrice symétrique semi-définie positive M telle que $I - M$ est aussi semi-définie positive.

$$\alpha \mathrm{tr}\left(W\left(M + \beta I\right)^{-1} W'\right) \tag{4.16}$$

Résultats

La table 4.4 montre les résultats des différentes approches. Nous remarquons que :
— notre approche, pour un tuning global, est meilleure que *SSSL*, cependant le tuning local est insuffisant ;
— notre approche est souvent meilleure que la régularisation multi-labels *SGR* ;

— le tuning global pour notre approche donne très souvent de meilleurs résultats que le tuning local ;
— contrairement à *SSSL*, notre approche donne des résultats comparables à l'état de l'art sur les jeux de données considérés, pour des approches de régularisation.

4.5.5 Validation statistique

Afin de savoir comment se positionnent les algorithmes les uns par rapport aux autres, nous effectuons un test statistique en suivant la méthodologie indiquée dans [72].

t-test par paires

En considérant une paire d'algorithmes, il est possible de calculer la différence entre les performances de ces deux algorithmes sur chaque jeu de données. Lorsque l'on prend la moyenne de cette différence, à supposer qu'elle s'applique sur un grand nombre de jeux de données, on peut effectuer un test pour savoir si cette moyenne est positive, ou pour savoir si elle est négative. Cela permettrait de savoir si l'un des algorithmes est relativement meilleur que l'autre.

Comme nous n'avons que 13 jeux de données, il n'est pas possible de considérer les performances de chaque algorithme comme un échantillon gausssien, ce qui pose une limite assez claire à cette approche.

De plus, les jeux de données contenant très peu d'individus ont une performance très variable.

Test de Wilcoxon

Ce test permet aussi de départager deux algorithmes, mais il ne se fonde plus sur les valeurs de la métrique envisagée mais sur les rangs des algorithmes. Plus précisément, on calcule les différences entre les deux algorithmes sur chaque jeu de données en valeur absolue, puis on trie ces différences absolues. Pour chaque algorithme, on sélectionne les différences qui sont en faveur de cet algorithme, et on somme leurs rangs. En prenant une valeur de $\alpha = 0.1$, on obtient la figure 4.6. Sur cette figure, une cellule bleue indique que l'algorithme en ligne bat l'algorithme en colonne. Une cellule rouge indique que l'algorithme en colonne bat l'algorithme en ligne. Notre généralisation est meilleure que l'algorithme *SSSL*.

Tests de Friedman et Nemenyi

Le test de Friedman permet de rejeter l'hypothèse suivante en se fondant sur les rangs des approches : tous les régresseurs sont équivalents, c'est-à-dire que le rang moyen de tous les régresseurs est égal. Cette hypothèse est rejetée dans notre cas pour un risque $\alpha = 0.05$.

Puisque le rang moyen n'est pas égal, le test de Nemenyi permet d'établir la distance critique entre rangs moyens. Si des algorithmes ont une différence entre leurs rangs moyens supérieure à cette distance critique, ils ne sont pas équivalents. La figure 4.7

TABLE 4.4 – Résultats du tuning global, métrique aRMSE

Jeu de données	**LSMR**	**LSMR local**	*SSSL*	*CMTL*	*DM*	*JFS*	*LSQ*	*MTL*	*SGR*	*TNR*
atp1d	**0.472**	1.006	0.481	0.533	0.522	0.543	0.549	0.534	0.548	0.548
atp7d	0.888	**0.713**	0.779	0.727	0.782	0.764	0.787	0.787	0.787	0.787
edm	**0.841**	1.005	0.895	0.849	0.857	0.856	1.198	0.855	1.207	0.852
enb	**0.320**	0.541	0.339	0.332	0.333	0.332	0.332	0.332	0.332	0.332
jura	0.661	0.847	0.727	0.597	0.593	**0.591**	0.609	0.593	0.611	0.598
oes10	**0.390**	0.488	0.395	0.398	0.402	0.402	0.402	0.402	0.402	0.402
oes97	**0.445**	0.918	0.494	0.515	0.510	0.534	0.534	0.534	0.534	0.534
osales	1.012	0.957	0.938	0.882	**0.839**	0.861	0.921	0.873	0.873	0.906
sarcossub	0.363	0.816	0.428	0.357	0.357	**0.354**	0.357	0.354	0.357	0.357
scpf	1.111	0.926	1.253	**0.621**	0.636	0.635	0.635	0.635	0.635	0.635
sf1	1.070	**0.998**	1.092	0.999	0.999	0.999	1.292	0.999	1.285	0.999
sf2	**0.973**	1.041	1.114	1.022	1.161	1.024	1.262	1.024	1.249	1.024
wq	0.999	0.999	1.003	**0.946**	1.006	0.982	1.079	0.947	1.079	1.023

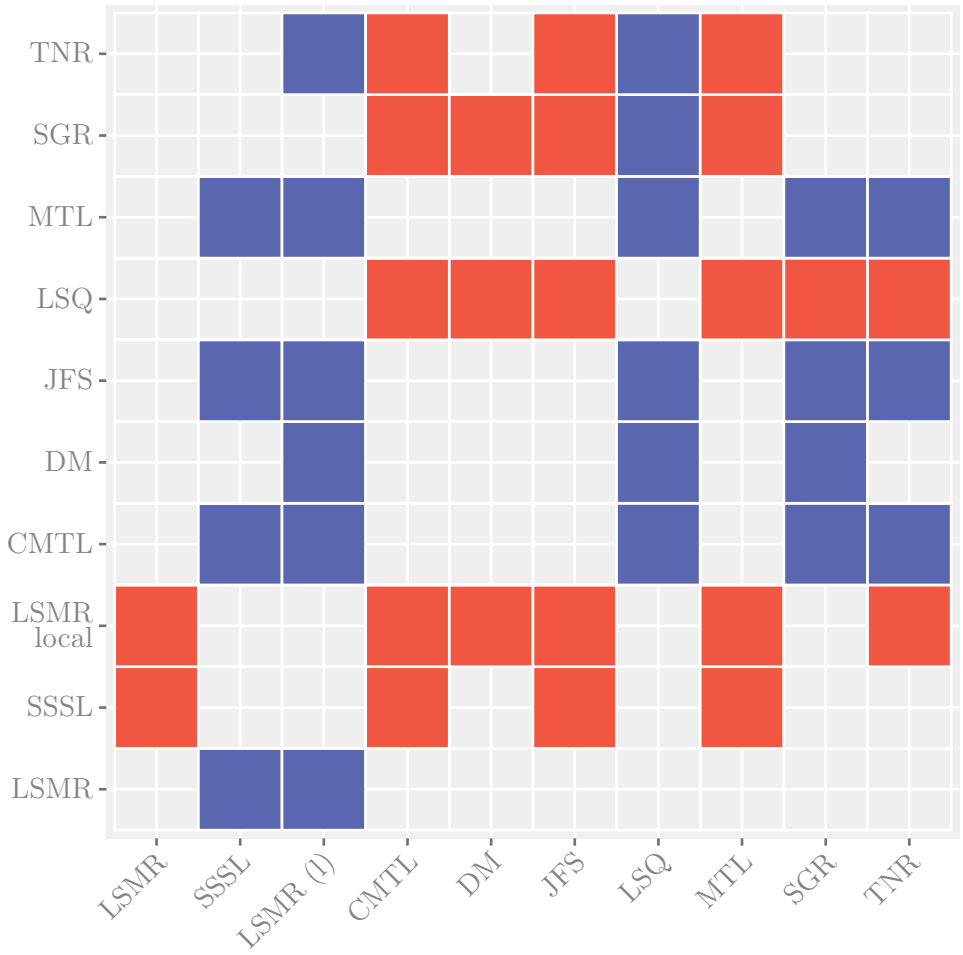


FIGURE 4.6 – Résultat du test de comparaison par paires d'algorithmes de Wilcoxon

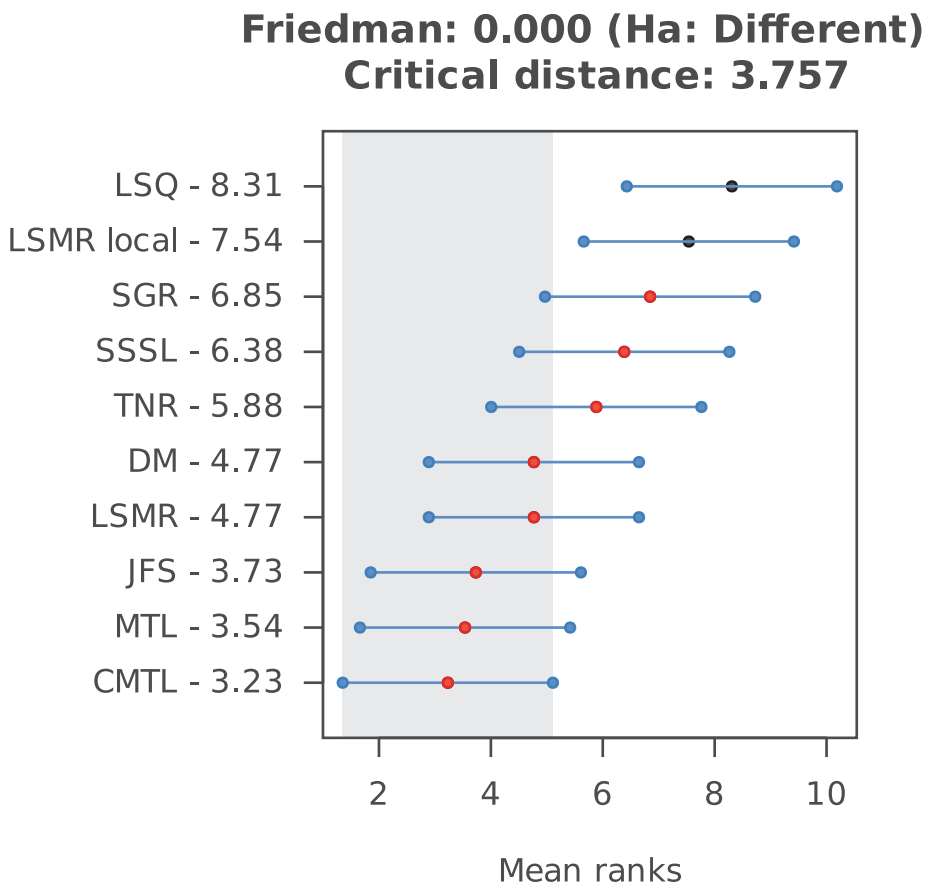


FIGURE 4.7 – Résultat du test de comparaison par paires d'algorithmes de Nemenyi

montre les rangs des algorithmes, ainsi que la distance critique. On constate que notre approche, même si elle n'obtient pas le meilleur rang moyen, est dans le groupe du meilleur algorithme, contrairement aux deux approches sur lesquelles elle est fondée.

4.6 Conclusion

Nous avons proposé une extension de l'algorithme *SSSL* adapté à l'apprentissage multi-labels, en réutilisant la régularisation Laplacienne que nous avions développée dans l'algorithme *LapS3L*. Expérimentalement, cette approche donne une erreur de régression multi-labels plus faible que *SSSL*, même en réutilisant une partie du tuning des hyper-paramètres. En optimisant les hyperparamètres de façon globale, l'approche demeure compétitive avec des méthodes représentatives de l'état de l'art.

Chapitre 5

Sélection de variables semi-supervisée en multi-régressions

Les deux travaux présentés dans les chapitres précédents, **LapS3L** et sa version multi-labels, **LSMR**, utilisent une *extraction de variables* non supervisée pour réduire la dimension du problème.

Dans certains cas d'application, cependant, la *sélection de variables* est une tâche plus abordée puisqu'elle rend le modèle *interprétable*. Dans ce chapitre, nous proposons de résoudre cette tâche dans un cadre à la fois multi-labels et semi-supervisé pour les problèmes de régression, et montrons son efficacité sur des jeux de données multi-labels publics issus de la littérature.

L'idée de cette méthode consiste à guider la sélection de variables par la *sélection de labels* : en effet, si les performances d'apprentissage sont dégradées pour certains labels, il n'est pas nécessaire de les conserver pour la sélection de variables.

5.1 Introduction

Dans le cas d'apprentissage multi-labels dont l'espace de description des individus est de grande dimension, ce qui arrive pour des données textuelles ou d'images par exemple, la présence de trop de variables par rapport au nombre d'individus peut mener à un sur-apprentissage. Pour empêcher cela, la sélection de variables multi-labels a pour objectif de trouver un sous-ensemble de variables pertinentes pour tous les labels, permettant d'obtenir une meilleure erreur de régression, de réduire la dimension du problème et de produire un modèle plus simple et interprétable.

Dans ce chapitre, nous proposons un algorithme de sélection de variables semi-supervisé et multi-labels, adapté aux problèmes de régression. Tout d'abord, nous rappelons le contexte ainsi que les travaux existants sur lesquels nous nous appuyons. Ensuite, nous développons l'algorithme, à partir de l'explicitation de notre intuition. Enfin, pour valider notre approche, nous proposons une étude expérimentale comparative sur des jeux de données de régression multi-labels.

5.2 Travaux liés

L'approche que nous proposons s'inspire principalement de l'algorithme *MIFS* (*Multi-label Informed Feature Selection*) dans les travaux de [11], de façon à fonctionner dans le cadre semi-supervisé, et en régression. Nous proposons également des éléments de l'apprentissage robuste et de la sélection de labels.

Dans le cadre de la sélection de variables, l'algorithme *MIFS* propose un *embedding* des labels. Ainsi, les variables ne sont pas sélectionnées pour améliorer la prédiction des vrais labels, mais des labels extraits. L'avantage de cette approche est que des valeurs bruitées dans l'ensemble d'apprentissage perturberont peu la sélection de variables. En revanche, la méthode ne peut pas servir directement en prédiction pour faire simultanément de la sélection de variables et de la régresssion multi-labels.

Pour rappel, l'algorithme *MIFS* vise à minimiser la fonction objectif suivante :

$$\underset{W \in \mathbb{R}^{d,o}, V \in \mathbb{R}^{n,o}, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad \|XW - V\|_F^2 + \alpha \|Y - VB\|_F^2 + \beta \text{tr}\left(V'LV\right) + \gamma \|W\|_{2,1} \tag{5.1}$$

Cette fonction objectif se décompose en 4 termes :

1. L'erreur de régression pour prédire les labels extraits ;
2. La précision de l'extraction des labels ;
3. La cohérence des labels extraits ;
4. La sélection de variables, sur le modèle prédisant les labels extraits.

Cependant, la pénalisation de l'extraction de labels prévue par *MIFS* n'est pas suffisante dans les cas où certains des labels dont on souhaite effectuer la prédiction ne peuvent tout simplement pas être traités. Si de tels labels sont présents dans le jeu de

données, la prédiction pour ces labels sera mauvaise quelles que soient les variables sélectionnées : il ne faut pas sélectionner de variables pertinentes pour ces labels. De ce point de vue, la sélection de labels au service de la sélection de variables joue un rôle différent de l'extraction de labels telle que mise en œuvre dans l'algorithme *MIFS*.

La sélection de labels peut donc être utilisée pour l'apprentissage multi-labels [73]. Dans ce cas, le modèle multi-labels est décomposé en un terme utilisant une structure commune pour la plupart des labels, et un terme indiquant les labels non traités.

$$\underset{A,\,B}{\text{minimize}} \quad \|X(A+B)-Y\|_F^2 + \alpha\mathcal{R}(A) + \beta\,\|B\|_{1,2} \tag{5.2}$$

La notation $\|B\|_{1,2}$ désigne la norme $l_{1,2}$ de B, c'est-à-dire la norme $l_{2,1}$ de la transposée de B :

$$\|B\|_{1,2} = \sum_{l=1}^{m} \|B_{.,l}\|_2 \tag{5.3}$$

$\mathcal{R}$ désigne un terme de régularisation multi-labels quelconque. Dans le cas de [73], il s'agit de la norme trace. La régularisation $\|B\|_{1,2}$ ressemble au terme de sparsité du *dirty model*, mais l'hypothèse est plus forte, puisque ce terme ne correspond qu'à certains labels.

5.3 Approche proposée : RSMS

Dans cette section nous décrivons notre approche que nous dénommons par la suite **RSMS** (*Robust Semi-supervised Multi-label feature Selection*).

On dispose d'un ensemble d'apprentissage supervisé, $(x_i \in \mathcal{X}, y_i \in \mathbb{R}^m)_{i=1}^{n_l}$, auquel est adjoint un ensemble non supervisé, $(x_i \in \mathcal{X})_{i=n_l+1}^{N}$.

Tout d'abord, l'algorithme cherche à effectuer une extraction de labels. Pour cela, on écrit la matrice de labels Y comme suit :

$$Y = VB$$

Ceci introduit la matrice de pseudo-labels $V \in \mathbb{R}^{N,o}$, et la matrice $B \in \mathbb{R}^{o,m}$. Concrètement, chaque individu labellisé de l'ensemble d'apprentissage correspond à une ligne de V, et chaque pseudo-label correspond à une colonne de V. Dans le cadre semi-supervisé, on définit une matrice diagonale $J \in \mathbb{R}^{N,N}$, dont la diagonale est l'indicatrice des individus labellisés. Pour chaque individu $i \in \{1, ..., N\}$, $J_{ii} = 1$ si et seulement si i est labellisé. Cette matrice nous permet donc d'affiner la décomposition, qui ne doit porter que sur la partie labellisée :

$$\underset{V \in \mathbb{R}^{N,o},\, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad \|JVB - JY\|_F^2 \tag{5.4}$$

L'obtention d'un modèle d'apprentissage semi-supervisé peut ainsi s'effectuer uniquement à partir des labels extraits. On introduit un modèle $W \in \mathbb{R}^{d,o}$, et un régulariseur

$\alpha > 0$. Plus α est grand, plus les labels extraits représenteront fidèlement les vrais labels Y.

$$\underset{W \in \mathbb{R}^{d,o}, V \in \mathbb{R}^{N,o}, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad \|XW - V\|_F^2 + \alpha \|JVB - JY\|_F^2 \tag{5.5}$$

L'optimisation par rapport à V permet d'assigner des pseudo-labels même aux individus non labellisés. Ce n'est cependant pas suffisant pour tenir compte des hypothèses de l'apprentissage semi-supervisé, puisque ces valeurs ne sont pas contraintes. La traduction des hypothèses semi-supervisées, dans le cadre de l'algorithme *MIFS*, se traduit de la façon suivante :

> Si deux individus sont proches, alors les valeurs de leurs pseudo-labels doivent être proches.

En suivant cette hypothèse, on peut être amené à introduire une matrice Laplacienne $L \in \mathbb{R}^{N,N}$ et un nouveau régulariseur $\beta > 0$, de sorte à minimiser le problème suivant :

$$\underset{W \in \mathbb{R}^{d,o}, V \in \mathbb{R}^{N,o}, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad \|XW - V\|_F^2 + \alpha \|JVB - JY\|_F^2 + \beta \text{tr}\left(V'LV\right) \tag{5.6}$$

Il reste donc deux problèmes à traiter : la sélection de variables, et la sélection de labels au service de la sélection de variables. Pour la sélection de variables, l'utilisation de la norme $l_{2,1}$ est très répandue [74]. Pour rappel, il s'agit de régulariser le terme s'appliquant sur les lignes de W :

$$\sum_{j=1}^{d} \sqrt{\sum_{k=1}^{o} W_{jk}^2} = \|W\|_{2,1}$$

Ce terme permet de rendre les lignes de W éparses, c'est-à-dire que des lignes de W tendent à être entièrement nulles. Les lignes comportant des valeurs non nulles indiquent des variables qui sont importantes pour l'apprentissage de tous les pseudo-labels. Le problème devient donc :

$$\begin{aligned} \underset{W \in \mathbb{R}^{d,o}, V \in \mathbb{R}^{N,o}, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad & \|XW - V\|_F^2 + \alpha \|JVB - JY\|_F^2 \\ & + \beta \text{tr}\left(V'LV\right) + \gamma \|W\|_{2,1} \end{aligned} \tag{5.7}$$

Pour effectuer la sélection de labels, notre hypothèse se traduit donc par le fait que certaines des colonnes de la matrice B, qui correspondent donc aux vrais labels, seront entièrement nulles. En introduisant un terme de régularisation supplémentaire, on obtient :

$$\begin{aligned} \underset{W \in \mathbb{R}^{d,o}, V \in \mathbb{R}^{N,o}, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad & \|XW - V\|_F^2 + \alpha \|JVB - JY\|_F^2 \\ & + \beta \text{tr}\left(V'LV\right) + \gamma \|W\|_{2,1} + \delta \|B\|_{1,2} \end{aligned} \tag{5.8}$$

L'hypothèse semi-supervisée change également. La régularisation $\mathrm{tr}(VLV')$ n'est pas pertinente, parce qu'elle s'applique de façon indiscriminée sur tous les pseudo-labels. Ainsi, si un pseudo-label représente l'un des labels ignorés, il ne faut pas tenir compte de ses valeurs pour la régularisation. Ceci nous mène à modifier l'hypothèse semi-supervisée pour la sélection de variables aidée par la sélection de labels :

> Si deux individus sont proches, alors pour chaque label sélectionné, la reconstruction des valeurs de ce label doivent être proches.

Nous considérons pour la reconstruction des valeurs des labels le produit VB. Le problème final devient donc 5.9 :

$$\begin{aligned} \underset{W \in \mathbb{R}^{d,o}, V \in \mathbb{R}^{N,o}, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad & \|XW - V\|_F^2 + \alpha \|JVB - JY\|_F^2 \\ & + \beta \mathrm{tr}\left(B'V'LVB\right) + \gamma \|W\|_{2,1} + \delta \|B\|_{1,2} \end{aligned} \tag{5.9}$$

Le terme de régularisation semi-supervisé $\mathrm{tr}\left(B'V'LVB\right)$ peut se réécrire de la façon suivante :

$$\sum_{k=1}^{m} \sum_{i_1=1, i_2=1}^{N} M_{i_1,i_2} \left(\sum_{l=1}^{o} \left(V_{i_1,l} - V_{i_2,l}\right) B_{l,k} \right)^2 \tag{5.10}$$

Par hypothèse, si un label k est ignoré, alors pour tout $l \in \{1, ..., o\}$, $B_{l,k} = 0$. Par conséquent, la régularisation ne porte que sur les labels non ignorés, notés $\mathcal{S} \subset \{1, ..., m\}$:

$$\sum_{k \in \mathcal{S}} \sum_{i_1=1, i_2=1}^{N} M_{i_1,i_2} \left(\sum_{l=1}^{o} \left(V_{i_1,l} - V_{i_2,l}\right) B_{l,k} \right)^2 \tag{5.11}$$

Contrairement à l'apprentissage robuste [73], où le modèle doit rendre compte à la fois des labels sélectionnés et des labels non sélectionnés, nous ne nous intéressons qu'aux variables pour les labels sélectionnés. Il n'est donc pas nécessaire de décomposer le terme B.

5.4 Optimisation

La question de l'initialisation des variables pose problème, puisque le résultat final en dépend. Il est possible d'utiliser des variables indépendantes suivant une loi normale, mais cela ne permet pas de rendre l'algorithme déterministe.

En ce qui concerne W, nous avons choisi d'utiliser l'initialisation suivante :

$$W^{(0)} \leftarrow \left[X'JX + \epsilon I\right]^{-1} X'JV \tag{5.12}$$

Il s'agit de la solution aux moindres carrés, régularisée par ϵ. La valeur a été fixée à 10^{-6}, qui est suffisamment faible pour ne pas trop influencer l'apprentissage pour les jeux de données du type de ceux étudiés dans l'étude expérimentale. Le terme V_l désigne le sous-ensemble labellisé des lignes de V.

Pour V et B, la question est plus complexe. Pour conserver l'aspect déterministe de l'initialisation, on ne peut pas recourir à un algorithme de clustering de type *k-means*, et il n'est pas non plus possible d'effectuer une décomposition *SVD* de la matrice Y. L'initialisation retenue pour B est une matrice diagonale rectangulaire (la diagonale s'arrêtant prématurément, laissant des colonnes nulles). L'initialisation pour V_l correspond donc aux valeurs des o premiers labels. Le reste des valeurs de V est obtenue en utilisant la valeur initiale $W^{(0)}$ de W sous la forme $V^{(0)} \leftarrow XW$.

L'initialisation de l'algorithme consiste donc à faire un apprentissage des o premiers labels avec une régularisation Ridge de paramètre 10^{-6}.

Comme pour l'algorithme *MIFS*, notre approche utilise 3 variables d'optimisation : W, V et B. En effectuant une optimisation alternée vis-à-vis de chacune des variables, nous pouvons vérifier que la valeur de la fonction objectif décroît à chaque étape, ce qui montre la convergence de notre approche.

5.4.1 Optimisation vis-à-vis de W, V et B étant fixés

Le problème 5.9, si l'on ne considère que W, peut se réécrire en :

$$\underset{W \in \mathbb{R}^{d,o}}{\text{minimize}} \quad \|XW - V\|_F^2 + \gamma \|W\|_{2,1} \tag{5.13}$$

La difficulté de ce problème réside dans le fait que le deuxième terme n'est pas lisse. Il existe plusieurs façons de traiter ce cas, par exemple l'implémentation MALSAR [34] propose une descente de gradient proximal. Dans notre cas, nous utilisons une constante $\epsilon > 0$, et nous adoptons la solution de l'algorithme *RFS*[74], en posant la matrice diagonale D_W telle que :

$$\forall j \in \{1, ..., d\}, D_{W\,j,j} = \begin{cases} \frac{1}{2\|W_{j,.}\|_2}, & \|W_{j,.}\|_2 \neq 0 \\ \epsilon, & \|W_{j,.}\|_2 = 0 \end{cases} \tag{5.14}$$

Cette définition permet d'écrire :

$$2\text{tr}(W' D_W W) = \sum_{j=1, \|W_{j,.}\|_2 \neq 0}^{d} \|W_{j,.}\|_2 + \sum_{j=1, \|W_{j,.}\|_2 = 0} \epsilon \|W_{j,.}\|_2^2 \tag{5.15}$$

Il s'agit donc de la norme $l_{2,1}$. En pratique, pour vérifier la convergence de l'algorithme, si le terme $\text{tr}(W' D_W W)$ décroît, alors le terme $\|W\|_{2,1}$ décroît aussi.

Le gradient vis-à-vis de W s'écrit donc :

$$2\left(X'(XW - V) + \gamma D_W W\right) \tag{5.16}$$

Si l'on pose :

$$W^* \leftarrow W - \frac{2}{C_W}(X'(XW - V) + \gamma D_W W) \tag{5.17}$$

on applique une itération de l'algorithme de descente du gradient, avec un pas d'apprentissage $\frac{1}{C_W}$. On constate que la fonction gradient est Lipschitzienne, c'est-à-dire que pour toute paire de modèles $W^{(1)}, W^{(2)}$,

$$\left\|\nabla_W(W^{(1)}) - \nabla_W(W^{(2)})\right\|_F^2 \leq C_W \left\|W^{(1)} - W^{(2)}\right\|_F^2 \tag{5.18}$$

avec $C_W = \rho(X'X) + \gamma \max(D_W)$, où $\rho(X'X)$ est le rayon spectral de la matrice $X'X$.

Par conséquent, l'itération de descente de gradient fait décroître la fonction objectif.

5.4.2 Optimisation vis-à-vis de V

En fixant W et B, le problème d'optimisation s'écrit :

$$\underset{V \in \mathbb{R}^{N,o}}{\text{minimize}} \quad \|XW - V\|_F^2 + \alpha \|JVB - JY\|_F^2 + \beta \text{tr}\left(B'V'LVB\right) \tag{5.19}$$

Le gradient par rapport à V s'écrit :

$$2\left(V - XW + \alpha JVB' - \alpha JYB' + \beta LVBB'\right) \tag{5.20}$$

La fonction objectif est convexe par rapport à V, et la fonction de gradient est Lipschitzienne : pour toute paire de modèles $V^{(1)}, V^{(2)}$,

$$\left\|\nabla_V(V^{(1)}) - \nabla_V(V^{(2)})\right\|_F^2 \leq C_V \left\|V^{(1)} - V^{(2)}\right\|_F^2 \tag{5.21}$$

avec :

$$C_V = 1 + (\alpha + \beta\rho(L))\,\rho(BB') \tag{5.22}$$

5.4.3 Optimisation vis-à-vis de B

En fixant W et V, le problème d'optimisation s'écrit :

$$\underset{B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad \alpha \|JVB - JY\|_F^2 + \beta \text{tr}\left(B'V'LVB\right) + \delta \|B\|_{1,2} \tag{5.23}$$

Cette fonction n'est pas lisse, à cause de la norme $l_{1,2}$ appliquée à B. On applique la même astuce que pour l'optimisation par rapport à W : on pose la matrice diagonale D_B dont la diagonale vaut :

$$\forall l \in \{1, ..., m\}, \quad D_{Bl,l} \leftarrow \begin{cases} \frac{1}{2\left\|B_{.,l}\right\|_2}, & \left\|B_{.,l}\right\|_2 \neq 0 \\ \epsilon, & \left\|B_{.,l}\right\|_2 = 0 \end{cases} \tag{5.24}$$

Nous devons donc résoudre le problème suivant :

$$\underset{B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad \alpha \|JVB - JY\|_F^2 + \beta \text{tr}\left(B'V'LVB\right) + \delta \text{tr}\left(BD_BB'\right) \tag{5.25}$$

Le gradient peut donc s'écrire :

$$2\left(\alpha V'(JVB - JY) + \beta V'LVB + \delta D_B B\right) \tag{5.26}$$

La fonction objectif est encore convexe par rapport à B, et la fonction de gradient est aussi Lipschitzienne, de constante :

$$C_B = \alpha\rho(V'JV) + \beta\rho(L)\rho(V'V) + \delta \max(D_B) \tag{5.27}$$

Une itération de descente de gradient de pas $\frac{1}{C_B}$ permet donc de diminuer la fonction de coût.

5.4.4 Algorithme final

L'algorithme complet 5 consiste simplement à alterner les étapes d'optimisation jusqu'à convergence. On remarque que le calcul du pas d'apprentissage nécessite d'obtenir la plus grande valeur propre de certaines matrices : BB' de dimension $m \times m$, $V'V$ de dimension $o \times o$, et $X'X$ de dimension $d \times d$ (dans le cas de la sélection de variables, d peut être grand) et L de dimension $N \times N$ (pour un problème d'apprentissage semi-supervisé, N peut être grand). Heureusement, le calcul pour ces deux dernières matrices peut être fait en amont.

5.5 Étude expérimentale

Pour évaluer la pertinence de notre approche de sélection de variables et de labels, nous proposons une étude expérimentale qui met en concurrence **RSMS** avec d'autres algorithmes de sélection de variables multi-labels adaptés à la régression, *SFUS*[75], *RFS*[74] et *MIFS*[11].

L'algorithme *MIFS* est l'algorithme sur lequel nous nous sommes fondés. Il reprend la sélection de variables multi-labels semi-supervisée pour la régression 5.28, et son optimisation est également effectuée par descente de gradient alternée.

$$\underset{W \in \mathbb{R}^{d,o}, V \in \mathbb{R}^{n,o}, B \in \mathbb{R}^{o,m}}{\text{minimize}} \quad \|XW - V\|_F^2 + \alpha \|Y - VB\|_F^2 + \beta \text{tr}\left(V'LV\right) + \gamma \|W\|_{2,1} \tag{5.28}$$

L'algorithme RFS utilise la norme $l_{2,1}$ exclusivement dans la fonction objectif. Ceci permet d'ignorer certains individus pour la construction du modèle W. Si le jeu de données contient des anomalies, la sélection de variables est rendue plus robuste de ce point de vue, d'où son nom, *Robust Feature Selection*.

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|XW - Y\|_{2,1}^2 + \alpha \|W\|_{2,1} \tag{5.29}$$

Enfin, l'algorithme *SFUS* se fonde sur l'idée de *RFS*, à savoir l'utilisation de la norme $l_{2,1}$ pour pouvoir ignorer certains individus, et une décomposition de rang faible du modèle.

Algorithm 5 Algorithme **RSMS** : sélection de variables

Donnée : $X \in \mathbb{R}^{N,d}$
Donnée : J, diagonale, indicatrice des données labellisées
Donnée : $Y \in \mathbb{R}^{N,m}$
Hyperparamètre : $\alpha > 0$, $\beta > 0$, $\gamma > 0$, $\delta > 0$
Hyperparamètre : $o \in \{1, ..., m\}$
Hyperparamètre : matrice Laplacienne du graphe des individus, $L \in \mathbb{R}^{N,N}$
Initialisation :
1. B : matrice diagonale rectangulaire
2. $W \leftarrow [X'JX + \epsilon I]^{-1} X'JYB$
3. $V \leftarrow XW$
4. $D_W \leftarrow I_d$
5. $D_B \leftarrow I_m$

Itération jusqu'à convergence :
1. $C_W \leftarrow \rho(X'X) + \gamma \max(D_W)$
2. $W \leftarrow W - \frac{2}{C_W}(X'(XW - V) + \gamma D_W W)$
3. $\forall j, \quad D_{W\,j,j} \leftarrow \frac{1}{2\|W_{j,.}\|_2}$
4. $C_V \leftarrow 1 + (\alpha + \beta\rho(L))\,\rho(BB')$
5. $V \leftarrow V - \frac{2}{C_V}(V - XW + \alpha JVB' - \alpha JYB' + \beta LVBB')$
6. $C_B \leftarrow \alpha\rho(V'JV) + \beta\rho(L)\rho(V'V) + \delta \max(D_B)$
7. $B \leftarrow B - \frac{2}{C_B}(\alpha V'(JVB - JY) + \beta V'LVB + \delta D_B B)$
8. $\forall l, \quad D_{Bl,l} \leftarrow \frac{1}{2\|B_{.,l}\|_2}$

Résultat : les variables sélectionnées sont les lignes de W de norme minimale
Résultat : les labels sélectionnés sont les colonnes de B de norme minimale

$$\begin{array}{ll} \underset{W \in \mathbb{R}^{d,m}, P \in \mathbb{R}^{o,m}, Q \in \mathbb{R}^{d,o}}{\text{minimize}} & \|XW - Y\|_{2,1}^2 + \alpha \|W\|_{2,1} + \beta \|W - QP\|_F^2 \\ \text{subject to} & Q'Q = I \end{array} \tag{5.30}$$

5.5.1 Jeux de données utilisées

Les jeux de données utilisés sont les suivants [30, 29] : *atp1d*, *atp7d*, *edm*, *enb*, *oes10*, *oes97*, *osales*, *scpf*, *sf1*, *sf2*, *wq*, comme pour le test de l'algorithme *LSMR* défini dans le chapitre précédent.

5.5.2 Algorithmes utilisés

Pour cette étude expérimentale, nous avons sélectionné quatre algorithmes de sélection de variables multi-labels.

Premièrement, l'algorithme *MIFS*. Cet algorithme n'est pas conçu pour l'apprentissage semi-supervisé ; cependant sa fonction objectif ainsi que son algorithme d'optimisation peuvent très simplement utiliser l'information de tous les individus, y compris ceux non labellisés. C'est cette version que nous avons utilisée pour la comparaison. À cet algorithme, nous ajoutons deux algorithmes de sélection de variables multi-labels, l'algorithme *SFUS*[75] et l'algorithme *RFS*[74] décrit ci-dessus.

Puisque notre approche est destinée à effectuer de la sélection de variables, nous devons tester les performances d'un algorithme de régression multi-labels quelconque qui apprendrait avec différents sous-ensembles des variables (et éventuellement des labels) sélectionnés. Étant donné qu'il faille effectuer un apprentissage différent pour chaque sous-ensemble de variables et chaque sous-ensemble de labels, sans possibilité de réutilisation des valeurs des hyperparamètres, nous avons choisi l'algorithme avec un tuning plus limité. Par conséquent, les valeurs obtenues ne sont pas exactement les mêmes que pour le chapitre précédent.

5.5.3 Protocole expérimental

Pour chaque algorithme et chaque jeu de données, on recherche les valeurs optimales des hyperparamètres en suivant une procédure de tuning optimisant la valeur moyenne de la métrique *aRMSE* selon une validation croisée à 10 folds. Une fois ces hyperparamètres déterminés, les variables (et dans le cas de **RSMS**, les labels) sont triés par ordre d'importance.

L'évaluation de la sélection de variables s'effectue de la façon suivante. Pour une certaine fraction de variables, pour chaque algorithme, on détermine les valeurs des hyperparamètres de l'algorithme d'évaluation *LSMR* en suivant une recherche aléatoire qui minimise l'erreur aRMSE moyenne sur les 10 folds de la validation croisée. La partie de test réservée pour chaque jeu de données permet d'évaluer l'algorithme *LSMR*. Le nombre d'individus labellisés est fixé à 30%. L'opération est répétée 10 fois avec différents ensembles de test et différents individus labellisés.

TABLE 5.1 – Rang moyen de *LSMR* après sélection de variables multi-labels

Jeu de données	**RSMS**	*MIFS*	*SFUS*	*RFS*
atp1d	**1.9 [1]**	2.1 [2]	3.2 [4]	2.8 [3]
atp7d	2.5 [2]	3.2 [4]	**1.3 [1]**	3.0 [3]
edm	**1.8 [1]**	3.2 [4]	2.3 [2]	2.7 [3]
enb	2.3 [2]	3.4 [4]	**1.8 [1]**	2.5 [3]
oes10	2.8 [3]	2.2 [2]	3.0 [4]	**2.0 [1]**
oes97	2.5 [3]	3.6 [4]	**1.8 [1]**	2.1 [2]
osales	2.8 [4]	2.0 [2]	**1.25 [1]**	2.4 [3]
scpf	**2.1 [1]**	2.4 [2]	3.1 [4]	2.4 [3]
sf1	**1.9 [1]**	2.7 [3]	3.4 [4]	2.0 [3]
sf2	2.3 [2.5]	**1.7 [1]**	3.7 [4]	2.3 [2.5]
wq	**1.7 [1]**	2.6 [3]	2.4 [2]	3.3 [4]

5.5.4 Évaluation de la sélection de variables

Nous avons tout d'abord étudié la performance de la sélection de variables pour les différents algorithmes sur les différents jeux de données. En fixant le nombre de labels conservés à 100%, et le nombre d'individus labellisés à 30%, on obtient une courbe pour chaque jeu de données, pour la métrique aRMSE 5.1.

Les résultats sont agrégés dans la table 5.1. Nous avons pour chaque jeu de données, chaque algorithme, et chaque pourcentage de variables conservées, calculé le rang de chaque algorithme (entre 1 et 4). Nous montrons le rang moyen de chaque approche sur l'ensemble des pourcentages de variables sélectionnées.

Si l'on calcule le rang moyen de chaque algorithme sur toutes les fractions de variables sélectionnées et tous les jeux de données, on obtient le résultat suivant :

1. **RSMS** : **2.21**
2. *RFS* : 2.50
3. *SFUS* : 2.55
4. *MIFS* : 2.70

La figure 5.1 montre les résultats obtenus pour chaque jeu de données. On observe les faits suivants :

— Pour *atp1d*, *atp7d*, *edm*, *enb*, *scpf* et *sf1*, la courbe d'erreur a un minimum global en ne sélectionnant qu'un certain nombre de variables. Pour *edm*, le choix de la première variable sélectionnée est très important.

— Pour *oes10*, *oes97*, *sf2* et *wq*, la sélection de variables n'est pas pertinente puisque l'erreur minimale s'obtient en sélectionnant toutes les variables, quelle que soit l'approche étudiée.

— Pour *osales*, la sélection de variables est très mauvaise pour RSMS. Pour les autres jeux de données pour lesquels la sélection de variables est pertinente, *atp1d*, *atp7d*, *edm*, et *enb* obtiennent le minimum global avec **RSMS**. *sf1* observe le

FIGURE 5.1 – Évaluation de la sélection de variables par *LSMR*

minimum global avec *RFS*, et *scpf* avec *MIFS* (0.9575 pour *MIFS* contre 0.9581 pour **RSMS**).

5.5.5 Sélection de labels

Pour la sélection de labels, nous cherchons à observer quels sont les labels qui sont les plus faciles à apprendre. Pour ce faire, on garde 30% des variables pour l'apprentissage, et on évalue la performance de l'algorithme *LSMR* en ne conservant que 20%, 40%, 60%, 80% et 100% des labels.

La figure 5.2 montre la courbe de sélection de labels pour chaque jeu de données. On retrouve différent cas de figure :

— Pour *atp1d*, *atp7d*, *enb*, *osales* et *wq*, la sélection de label permet de trouver un nombre de labels pour lequel l'erreur est minimale, de sorte qu'en sélectionnant moins de labels l'erreur augmente et en en sélectionnant plus l'erreur augmente également.

— Pour *oes10*, *scpf* et *sf1*, l'erreur est minimale si l'on ne garde que le nombre minimum de labels.

— Pour *edm*, *oes97* et *sf2*, sélectionner tous les labels donne une erreur de régression plus faible.

5.5.6 Sélection de labels au service de la sélection de variables

Concrètement, si certains labels ne peuvent pas être bien expliqués globalement par les variables en jeu, ils sont simplement ignorés dans la sélection de variables. Dans ce cas, nous comparons le résultat de l'apprentissage en conservant 80% des labels, contre le résultat pour 100% des labels. Dans le cas où le jeu de données a moins de 5 labels, on en conserve moins, pour observer une différence :

— pour *edm* et *enb*, on ne conserve qu'un label ;

— pour *scpf*, *sf1* et *sf2*, on en conserve 2 ;

La figure 5.3 montre la différence entre la sélection de variables seule et la sélection de variables avec sélection de labels. Les courbes obtenues indiquent qu'éliminer 20% des labels donne une meilleure erreur de régression. Les exceptions sont *enb*, d'une part, et *oes10*, *oes97* et *sf2* d'autre part, pour lesquels la sélection de variables n'est tout simplement pas pertinente. Le fait que les courbes semblent simplement translatées en sélectionnant les labels, sans changer la position du minimum, indique que les variables sélectionnées ne sont pas pertinentes pour les labels ignorés, ce qui était le but de notre approche.

5.5.7 Convergence de l'algorithme

Dans cette section, nous vérifions expérimentalement que notre approche converge en 100 itérations, ce qui est utilisé pour l'obtention de l'ordre des variables et des labels, comme utilisé dans les deux sections précédentes.

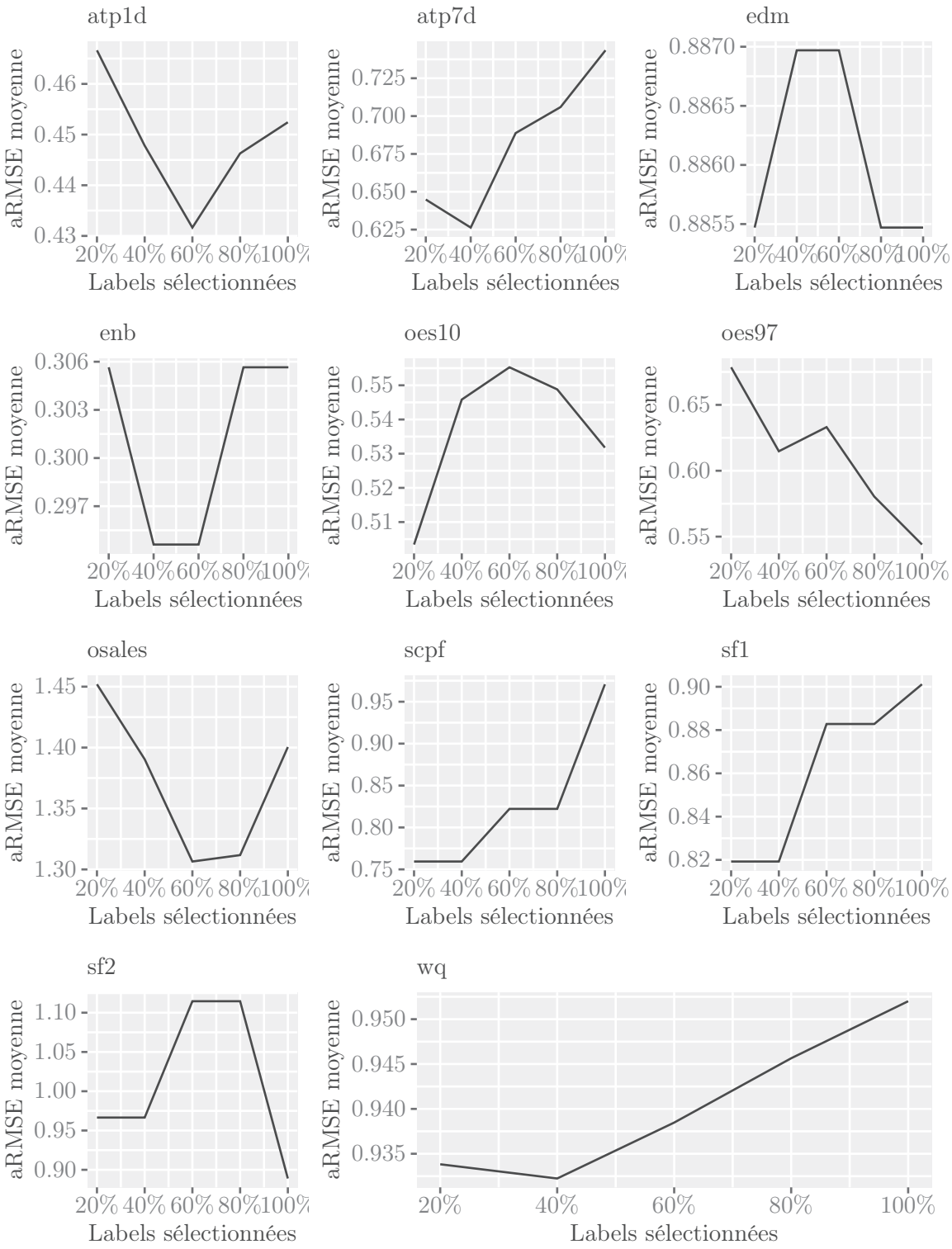


FIGURE 5.2 – Évaluation de la sélection de labels de **RSMS** par *LSMR*

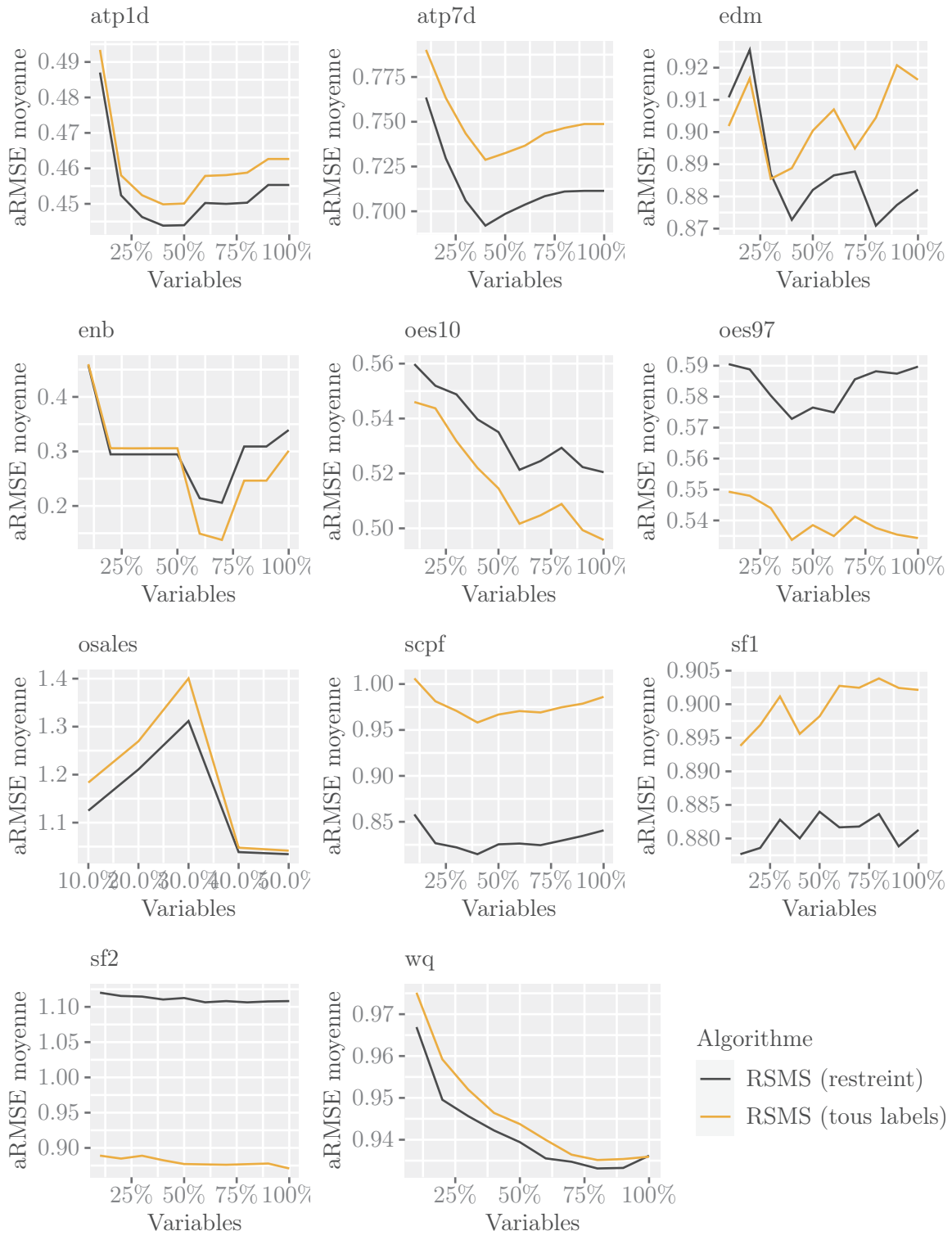


FIGURE 5.3 – Comparaison de la sélection de variables pour tous les labels, et pour un nombre restreint de labels

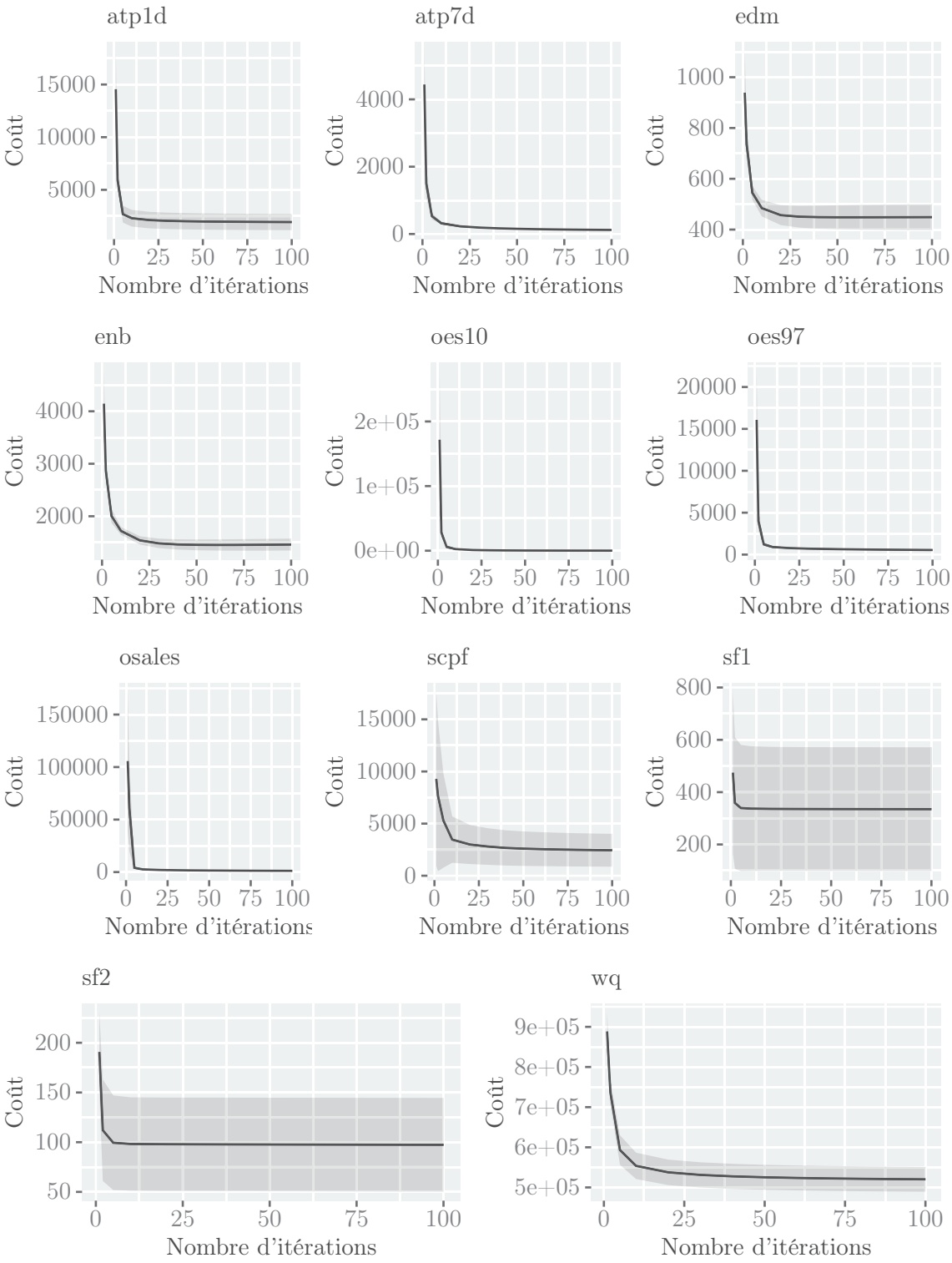


FIGURE 5.4 – Courbes de convergence de l'algorithme de sélection de variables *RSMS* avec écart-type

La figure 5.4 montre les résultats obtenus pour chaque jeu de données. On observe les faits suivants :

— La méthode converge en moins de 100 itérations.
— La convergence s'observe entre 10 itérations (pour *osales*) et 50 itérations (pour *wq*)
— Pour les jeux de données *sf1* et *sf2*, la convergence s'opère très rapidement, mais la fonction de coût reste importante.

5.6 Conclusion

Dans ce chapitre, nous avons proposé un algorithme de sélection de variables pour la régression multi-labels en mode semi-supervisée. Nous avons utilisé l'hypothèse selon laquelle la sélection de labels peut être au service de la sélection de variables.

Nous avons montré expérimentalement que notre approche est compétitive avec l'état de l'art de sélection de variables en régression multi-labels. Nous avons également montré la pertinence de la sélection de labels, pour déduire les labels dont l'apprentissage est plus facile, ce qui guide la sélection de variables à de meilleures performances.

Chapitre 6

Application à l'annotation automatique de pneumatiques

Dans ce chapitre, nous présentons le cadre d'annotation automatique que nous avons développé pour notre partenaire industriel Lizeo IT. Dans le cadre de cette thèse, nous sommes confrontés à un problème réel nécessitant un apprentissage semi-supervisé multi-labels. Nous évaluons la batterie d'algorithmes développés pour les deux tâches principales, à savoir la régression et la sélection de variables. Nous proposons également une adaptation de ces algorithmes afin de traiter le problème dans le cas où le nombre d'individus est important.

6.1 Introduction

Étant dans le cadre d'une convention CIFRE, cette thèse doit proposer un bon compromis entre l'aspect fondamental développé dans les chapitres précédents et l'aspect applicatif en validant nos modèles sur les jeux de données de l'entreprise. Ces données sont de nature textuelle et définissent différentes caractéristiques sur les pneumatiques.

6.2 Présentation de l'entreprise

Le cœur de métier du groupe international Lizeo (et de ses filiales aux États-Unis, à Shanghai et à Lyon) consiste à étudier les données relatives à l'industrie automobile, et en particulier les pneumatiques. L'activité de Lizeo se décompose en deux axes :

— l'activité *B2B* (*Business to Business*), qui consiste à traiter des données telles que les caractéristiques techniques, les prix, les évaluations, voire les photos et descriptions, à destination des entreprises s'intéressant au marché du pneumatique,
— l'activité *B2C* (*Business to Consumer*), en particulier représentée par son site vitrine http://pneu.rezulteo.fr, un comparateur de pneumatiques à destination du public.

Parmi l'ensemble des données dont dispose Lizeo IT, nous retrouvons (cf figure 6.1) :

— des produits, c'est-à-dire des pneumatiques, avec leurs caractéristiques techniques, les résultats des tests, ou les évaluations d'experts ;
— des prix, collectés auprès des revendeurs ou par aspiration de sites vendeurs ;
— des données de véhicules associées aux pneumatiques ;
— des données des distributeurs, qui permettent notamment de les identifier et les localiser ;
— et enfin, ce qui nous intéresse tout particulièrement, des avis des consommateurs, présents sur des forums, et les média sociaux.

Les travaux actuels de l'équipe R&D du groupe portent sur différents thèmes :

— l'analyse textuelle des avis du public et des experts des produits, dont cette thèse fait partie, mais qui vise aussi à la découverte automatique de thématiques ;
— l'analyse du cycle de vie des produits à partir des volumes de prix aspirés sur les sites de vente en ligne, afin de savoir si un pneumatique est récent, ou plus utilisé ;
— l'analyse des prix des produits, avec en particulier la détection d'anomalies dans les séries temporelles, afin de détecter les problèmes dans l'aspiration des prix des pneumatiques sur le web (ce qui est préjudiciable pour un comparateur de pneumatiques) ;
— la prédiction de volumes marchés et de prix.

6.3 Description du jeu de données

L'entreprise Lizeo IT, dans le cadre de son activité B2B, propose d'effectuer une veille des pneumatiques dans des ressources sur le web, telles que des forums, des articles

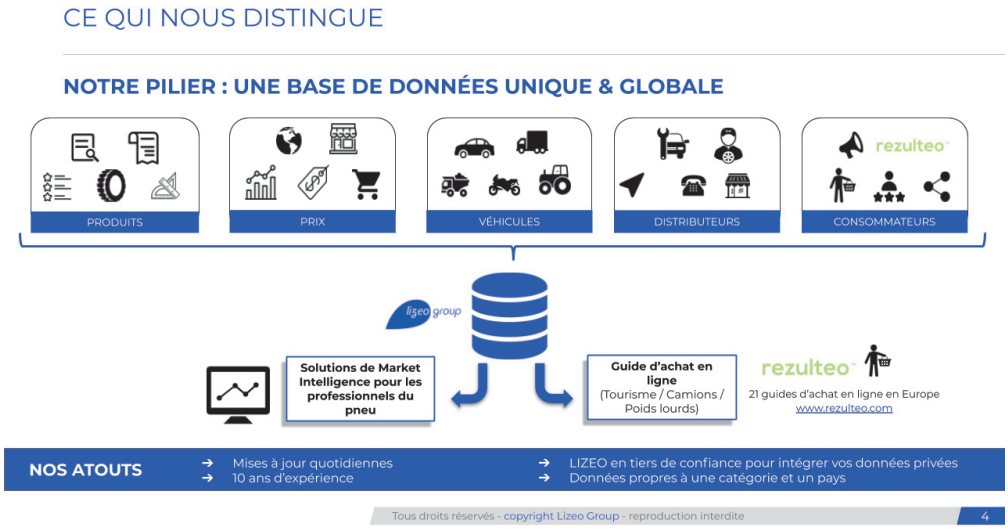


FIGURE 6.1 – Résumé des données collectées par Lizeo dans le cadre de son activité (communication interne)

spécialisés et des tweets. Concrètement, chaque document est un texte, la plupart étant plutôt courts (note de forum ou tweet), traitant d'un ou de plusieurs produits.

L'analyse de ces documents est cruciale pour Lizeo. La perception qu'ont les consommateurs des produits est une donnée d'importance, tant pour les manufacturiers que pour les distributeurs.

On peut décrire chaque produit à partir de différentes caractéristiques, appelées *qualifier* dans le jeu de données. Les experts ont identifié de nombreux *qualifier*. On peut par exemple citer le prix du produit, le bruit, la tenue sur route, ainsi que des *qualifier* plus précis : la tenue sur route humide, ou enneigée. Les *qualifier* sont ainsi conçus comme une hiérarchie. Il y a de nombreux *qualifier* dans le jeu de données, mais nous avons retenu les 97 qui sont employés plus de 10 fois.

Pour un document donné à propos d'un certain produit, un premier objectif du traitement consiste à indiquer, pour chaque *qualifier*, si celui-ci est pertinent. Concrètement, si le document comporte un commentaire sur le bruit d'un produit, il faut indiquer le *qualifier* "bruit".

Ce problème se traduit en un problème d'apprentissage de classification multi-labels : l'unité statistique est le document, les variables descriptives sont des variables textuelles, et les labels (binaires) sont les *qualifier*. Ce problème de classification, résumé par la figure 6.2, n'est cependant pas notre objectif principal.

Si un *qualifier* est pertinent pour un certain produit de l'un des documents, la question qui nous intéresse est la suivante : les termes employés évoquent-ils ce qualifier d'une façon positive ou péjorative ? Nous définissons pour cela la *tonalité* d'une qualification d'un document comme un nombre réel. Une valeur positive de la tonalité indique un

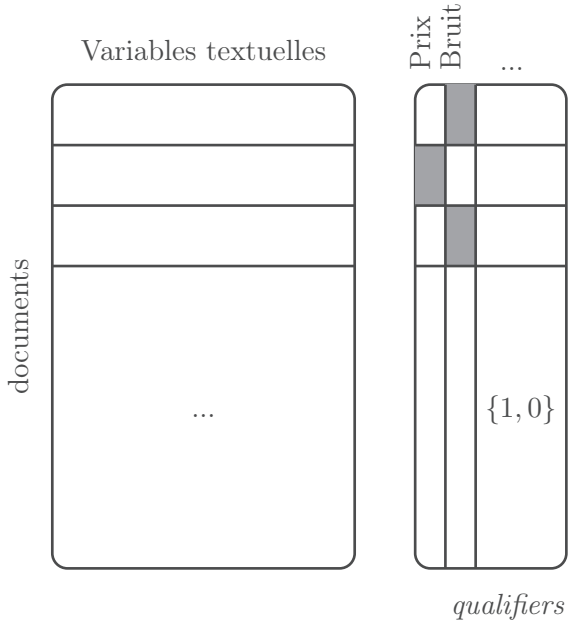


FIGURE 6.2 – Première étape du traitement du jeu de données Lizeo IT : apprentissage de classification binaire multi-labels

sentiment positif. Une valeur nulle indique un sentiment neutre, et une valeur négative un sentiment négatif. Nous traitons ce problème par un apprentissage de régression multi-labels, figure .

6.4 Traitement des données textuelles

Dans le jeu de données utilisé, il se peut qu'il y ait une comparaison entre deux produits sur les mêmes critères. Dans ce cas, on ne verra que la moyenne pour l'apprentissage. Ceci donne parfois des valeurs non entières pour les labels.

Afin d'évaluer les performances de la sélection de variables et de la sélection de labels, nous étudions un sous-ensemble des données regroupant les caractéristiques suivantes :

1. Il s'agit de documents (appelés parfois posts) en français ;
2. Les documents sont tous labellisés ;
3. Il y a 84 907 documents ;
4. La distribution du nombre de /qualifiers/ est documentée par la table 6.1. On constate qu'il y a 30 145 documents associés à aucun qualifier, il ne s'agit pourtant pas d'individus non labellisés, car nous savons qu'aucun des critères retenus ne s'applique sur ces documents. Dans notre cadre de régression, ceci signifie que toutes les tonalités doivent être neutres ;
5. Les 10 qualifiers les plus fréquents sont résumés dans la table 6.2, ce qui correspond à 69% de toutes les qualifications.

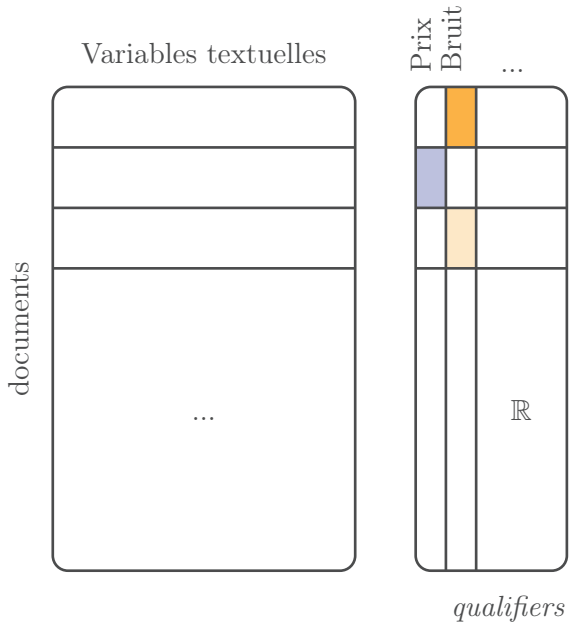


FIGURE 6.3 – Apprentissage de régression multi-labels de la tonalité

TABLE 6.1 – Distribution du nombre de qualifiers par document

Nombre de *qualifiers*	Nombre de documents
0	30145
1	31723
2	12372
3	5350
4	2710
5	1321
6	653
7	310
8	160
9	89
10	48
Entre 11 et 37	53

Les labels ont une arborescence. On retrouve :
— la réputation d'un produit ;
— l'usure ;
— la satisfaction du consommateur ;
— la sécurité, en évaluant la traction sur route sèche, humide ou enneigée ;
— le prix ;
— le confort ;
— et même l'esthétique du produit.

La table 6.2 permet de faire la remarque suivante : pour les qualifieurs les plus fréquents, la moyenne est souvent négative, et la médiane n'est jamais positive. Le fait que la médiane ne soit pas toujours entière tient aux posts faisant une comparaison de plusieurs produits, comme expliqué plus haut.

6.5 Application de l'algorithme LSMR

Dans le cadre de cette application, nous sommes confrontés à un jeu de données réel, ce qui pose de nouvelles contraintes. Pour appliquer l'algorithme *LSMR*, nous avons dû effectuer des adaptations de celui-ci afin de pouvoir fonctionner dans un cas où le nombre d'individus, N, est très grand. En effet, l'application de l'algorithme *LSMR* demande une décomposition en valeurs propres et vecteurs propres (partielle) de la matrice noyau, K, de dimension $N \times N$. Dans ce cas, on peut appliquer l'algorithme de Lanczos [76], qui permet d'obtenir le résultat avec une complexité de $\mathcal{O}(sN^2)$. La complexité spatiale, de manière bien plus prosaïque, est déterminée par la construction et le stockage de la matrice Laplacienne du graphe, L, et de la matrice noyau, K, toutes deux de dimension $N \times N$.

Afin de pouvoir effectuer une approximation de l'apprentissage en un temps raisonnable, nous proposons une modification ensembliste de l'algorithme *LSMR* adaptée à ce jeu de données.

6.5.1 Méthode ensembliste : Bootstrap Aggregating

Afin de conserver une valeur de N acceptable, nous tirons 20 différents sous-ensembles de 4000 points du jeu de données avec remise, afin d'obtenir 20 jeux de données différents de taille plus faible. Cette méthode est nommée *Bootstrap Aggregating*, ou *Bagging*[77].

Comme notre algorithme nécessite un certain nombre d'hyperparamètres, nous considérons un sous-ensemble de validation en plus des 4000 points pour le jeu d'apprentissage. La métrique de validation *aRMSE* est calculée pour chaque modèle, ce qui nous permet de sélectionner les 10 candidats donnant la meilleure erreur de régression.

La méthode de *bagging* prévoit de traiter l'apprentissage d'un modèle par ensemble. La prédiction pour un nouvel individu s'effectue en prenant la moyenne de la prédiction de tous les modèles, dans le cas de la régression.

Cette méthode a l'avantage de s'appliquer de manière naturelle dans le cas de la régression multi-labels.

TABLE 6.2 – Résumé des qualifieurs les plus fréquents parmi les 97 (*Cons. feedback* est l'abréviation de *Consumer feedback*, retour client, et *sat.* de *satisfaction*).

N°	Nombre	Nom	Médiane	Moyenne
59	31.82%	Characteristics / Image / Reputation	-1.000	-0.598
483	25.07%	Performances / Wear life / Wear life	0.000	-0.198
52	19.41%	Performances / Cons. feedback / Overall sat.	-1.000	-0.936
549	11.07%	Performances / Safety / Traction	-0.750	-0.538
506	7.99%	Performances / Ride performance / Noise	0.000	-0.163
494	7.68%	Performances / Safety / Handling	-1.000	-0.579
780	6.95%	Performances / Safety / Wet traction	-0.500	-0.419
782	5.72%	Performances / Safety / Snow/Winter traction	-0.667	-0.582
495	5.30%	Performances / Safety / Steering	0.000	-0.090
61	4.78%	Characteristics / Image / Intent to purchase	-1.000	-0.583
21	4.11%	Performances / Safety / Safety	-0.333	-0.386
60	3.87%	Characteristics / Image / Test results	-0.333	-0.572
491	3.61%	Performances / Environment / Environment	0.000	-0.473
58	3.49%	Performances / Cons. feedback / Recommendation	-1.000	-0.720
781	3.12%	Performances / Safety / Dry traction	-1.000	-0.831
132	3.03%	Attitude / Towards tyre type	-0.500	-0.389
488	2.55%	Performances / Wear life / Abnormal Wear	1.000	0.525
487	2.37%	Performances / Toughness/Robustness / <*idem*>	0.000	-0.055
10	2.23%	Characteristics / Price / Value for money	-1.000	-1.014
484	2.16%	Performances / Wear life / Ageing	0.000	0.404
497	2.12%	Performances / Safety / Stopping distance	0.000	-0.390
489	2.01%	Characteristics / Price / Deal / Discount	-1.000	-0.573
558	2.00%	Performances / Track/Sport perf. / Track perf.	0.000	-0.474
504	1.92%	Performances / Ride perf. / Ride experience	-1.000	-0.616
505	1.65%	Performances / Ride perf. / Ride comfort	-0.500	-0.440
133	1.33%	Attitude / Towards tests	0.000	-0.288
784	1.31%	Performances / Safety / Off road traction	-0.667	-0.546
508	1.29%	Characteristics / Image / Look/aesthetic	0.000	-0.352

TABLE 6.3 – Application de l'algorithme *LSMR*, avec *bootstrap*, toutes les variables, et les deux agrégations possibles

Agrégation moyenne	Agrégation Ridge
0.819	0.809

6.5.2 Stacking

Nous utilisons aussi une deuxième approche d'agrégation, nommée *stacking*[78]. Dans cette approche, on considère un nouveau jeu de données. Dans celui-ci, l'individu statistique est toujours le même, mais il est décrit cette fois par les prédictions de chacun des modèles retenus, et les labels sont les mêmes que pour le jeu d'apprentissage original.

Étant donné le cadre multi-labels de l'apprentissage original, nous considérons pour l'apprentissage par *stacking* que les labels à ce stade sont indépendants. Pour chaque label, on cherche donc à prédire la valeur du label en fonction des prédictions de tous les modèles pour ce label.

Cette étape est également l'occasion de ne considérer dans le jeu de données que les valeurs pertinentes. En effet, si l'on revient à la construction du jeu de données, et si un *qualifier* (label) n'est pas présent, la valeur cible sera la valeur neutre, 0. Cependant, cette valeur n'est pas vraiment pertinente pour l'apprentissage, ce qui pousse le modèle à apprendre des valeurs faibles. Les prédictions sont donc naturellement de norme inférieure à la vérité terrain. Pour compenser ce biais, l'agrégation par label ne retient comme individus que les posts présentant le *qualifier* en question.

Enfin, pour éviter le sur-apprentissage, on ne considère pour chaque modèle que les individus n'ayant pas servi à l'apprentissage du modèle de *bootstrap*. Pour les tâches de *stacking*, les valeurs de prédiction de chaque modèle sur son jeu d'apprentissage sont remplacées par 0.

Le régresseur retenu est une régularisation *ridge*, dont le régulariseur est recherché comme une puissance de 10 entre 10^{-4} et 10^4.

6.5.3 Résultats

En appliquant l'algorithme, sans effectuer de sélection de variables, on obtient le résultat présenté en table 6.3. Nous retenons que l'agrégation par *stacking* avec régularisation Ridge donne un meilleur résultat que l'agrégation moyenne.

6.6 Comparaison d'algorithmes de sélection de variables

Afin d'évaluer la sélection de variables, nous avons sélectionné 20 sous-ensembles de 4000 individus. La valeur des hyperparamètres est obtenue en minimisant la métrique aRMSE du modèle final. Nous évaluons deux autres algorithmes de sélection de variables multi-labels : *RFS*[74] et *MIFS*[11]. Nous n'avons pas pu appliquer l'algorithme *SFUS* en un temps raisonnable, à cause de la décomposition en valeurs propres et vecteurs propres d'une matrice $d \times d$.

6.6.1 RFS

L'algorithme *RFS*[74] effectue la sélection de variables en minimisant le problème suivant :

$$\underset{W \in \mathbb{R}^{d,m}}{\text{minimize}} \quad \|XW - Y\|_{2,1}^2 + \gamma \|W\|_{2,1} \tag{6.1}$$

Ce problème utilise une régularisation $l_{2,1}$, mais utilise aussi cette même norme pour la fonction de coût. Concrètement, cela permet d'ignorer certains individus pour la sélection de variables.

6.6.2 MIFS

Nous utilisons aussi l'algorithme *MIFS*[11], qui est la base de notre approche proposée. Pour rappel, le problème résolu par *MIFS* est la minimisation suivante :

$$\underset{\substack{W \in \mathbb{R}^{d,o} \\ V \in \mathbb{R}^{n,o} \\ B \in \mathbb{R}^{o,m}}}{\text{minimize}} \quad \|XW - V\|_F^2 + \alpha \|Y - VB\|_F^2 + \beta \text{tr}\left(V'LV\right) + \gamma \|W\|_{2,1} \tag{6.2}$$

6.6.3 Sélection de variables : étude préliminaire

Pour chaque sous-ensemble d'individus, on obtient une liste ordonnée de variables. Pour calculer la sélection de variables finale, nous calculons les rangs moyens de chaque variable pour chaque algorithme. Les premières variables sont listées dans la table 6.4.

Qualitativement, les 30 premières variables indiquent un avantage pour *RFS*. Les premières variables sélectionnées par **RSMS** sont principalement des éléments de dimension, c'est-à-dire des caractéristiques de forme des pneumatiques, ce qui n'est pas l'objet de notre étude.

6.6.4 Évaluation de la sélection de variables

Nous pouvons aussi évaluer quantitativement la sélection de variables. Nous choisissons un sous-ensemble de 4000 points, en appliquant l'algorithme *LSMR* non modifié, pour accélérer les calculs.

En évaluant le résultat de la sélection de variables (figure 6.4), on remarque cependant que c'est bien notre approche qui procure le meilleur résultat de sélection. En effet, l'erreur de régression est minimale pour 5% des variables sélectionnées (soit environ 1000 variables), contrairement à *MIFS* et *RFS* dont la sélection n'atteint la meilleure efficacité qu'en sélectionnant toutes les variables. On remarque aussi que la courbe de **RSMS** est plus abrupte que les autres, ce qui signifie que la sélection est plus efficace dès les premières centaines de variables sélectionnées.

Nous retenons la valeur des hyperparamètres issus du tuning pour **RSMS** :

— rang : $o = 30$;
— $\alpha = 10^3$;
— $\beta = 2.5 \times 10^{-2}$;

TABLE 6.4 – Sélection des 30 premières variables pour chaque algorithme.

RSMS	*MIFS*	*RFS*
disponible	classement_adac	silence
d'origine	litres	accrochent
actuels	vii	flottement
sport	def_90	bluffé
hiver	l'emergency	sécurisant
55	assez_bon	décroche
véhicule	remonte	aquaplaning
malheureusement	90_cv	regrette
3	defacyde	azenis
prix	rapprochement	l'abs
205	solidité	négatif
recherche	optimisées	routiers
r16	dois_prendre	excellents
non	extended	pépère
4	attendant	demi
r17	l'exige	l'esp
été	justifie_pas	soleil
l'indice	virages_sans	mou
235	prévenir	confortables
215	promener	châssis
225	financière	catastrophique
marquage	pas_1	pourri
cherche	carcasses	chauffe
195	écrit_1	reproche
45	n'avait_pas	gain
65	s_max	feeling
vitesse	marketing	fou
charge	225_75	d'urgence
xl	contrainte	savonnette
svp	moyens	clairement

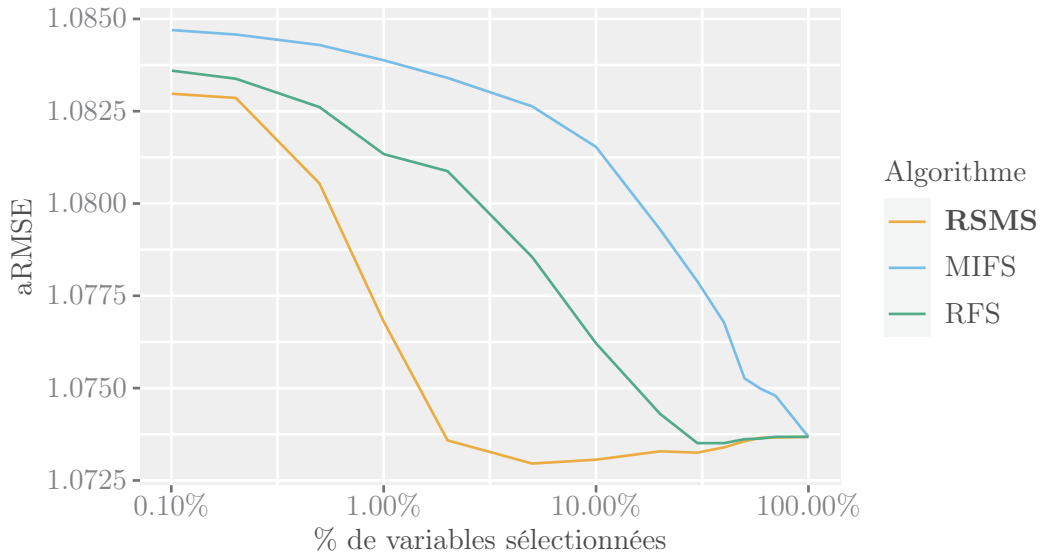


FIGURE 6.4 – Sélection de variables comparée pour avec LSMR non modifié comme évaluateur

— $\gamma = 4 \times 10^6$;
— $\delta = 4 \times 10^2$.

6.7 Application de RSMS sur l'ensemble du jeu de données

Étant donné que le jeu de données contient trop d'individus pour construire et utiliser la matrice Laplacienne du graphe des individus, nous considérons une version légèrement modifiée de l'algorithme.

6.7.1 Optimisation par époques

Chaque étape de l'algorithme **RSMS** est une itération de la descente de gradient. Il est possible d'utiliser la version stochastique avec *minibatch*, c'est-à-dire qu'au lieu de sélectionner un individu à chaque itération, un sous-ensemble est sélectionné à la place. Cette stratégie est souvent employée car elle permet expérimentalement d'obtenir le résultat en moins d'itération qu'avec la version stochastique. En contrepartie, le calcul du gradient doit se faire pour tous les individus sélectionnés simultanément, ce qui n'est pas un problème pour les architectures de calcul parallèle modernes [79].

Afin de pouvoir réutiliser les matrices Laplacienne de graphe, le partitionnement est le même à chaque époque. Nous obtenons le résultat pour 10 époques.

6.7.2 Résultats

Les premières variables ainsi obtenues sont exposées dans la table 6.5.

TABLE 6.5 – Sélection des 30 premières variables de la sélection de RSMS par époques

RSMS par époques
très
pas
content
bon
4
sport
bien
meilleurs
plus
neige
top
nokian
satisfait
prix
trop
emoji
3
2
vredestein
route
rapport
mal
excellent
dunlop
hiver
meilleur
bonne
sécurité
5
qualité

TABLE 6.6 – Variables probablement utiles pour la tonalité de tous les qualifieurs simultanément

RSMS par époques	*RSMS*	*MIFS*	*RFS*
très	malheureusement	assez_bon	bluffé
pas	non	optimisées	regrette
content		justifie_pas	négatif
bon			excellents
bien			catastrophique
meilleurs			pourri
plus			reproche
top			gain
satisfait			
trop			
mal			
excellent			
meilleur			
bonne			
qualité			

Contrairement aux sélections de variables effectuées sur des sous-ensembles des données (table 6.4), il est plus clair que les premiers termes sélectionnés sont applicables à tous les labels. En effet, dans les 30 premières variables, on peut repérer les termes suivants :

— la combinaison de "très" et "pas" permet d'indiquer une tonalité légèrement positive par défaut, et la modifier en plus positif ou négatif, indépendamment du label. La variable "très" n'est pas présente du tout dans les 30 premières variables des sélections par sous-ensemble, et seul *MIFS* sélectionne "pas" (comme membre d'un bi-gramme). **RSMS** par sous-ensembles sélectionne cependant "non". Notons qu'il est très important de conserver ces mots. Les tâches communes de classification textuelle, comme la classification multi-labels de documents, ne sont souvent pas affectées par la présence de ces mots, qui sont alors considérés comme *stopword* (mots-outil) et retirés du texte. Dans notre cas, ils sont essentiels.

— Nous avons relevé les termes d'appréciation généraux dans la table 6.6.

La figure 6.5 montre que l'optimisation par époques obtient son meilleur résultat en sélectionnant seulement 1% des variables (soit 5 fois moins que l'agrégation de la sélection par sous-ensemble). Nous retenons ces 200 meilleures variables.

6.7.3 Sélection de labels

En comparant les normes des colonnes de la matrice B, on peut trouver les labels ayant une plus forte norme, et donc ceux les mieux expliqués par les variables sélectionnées. Grâce à cette information, on peut identifier les labels difficiles. Nous constatons que

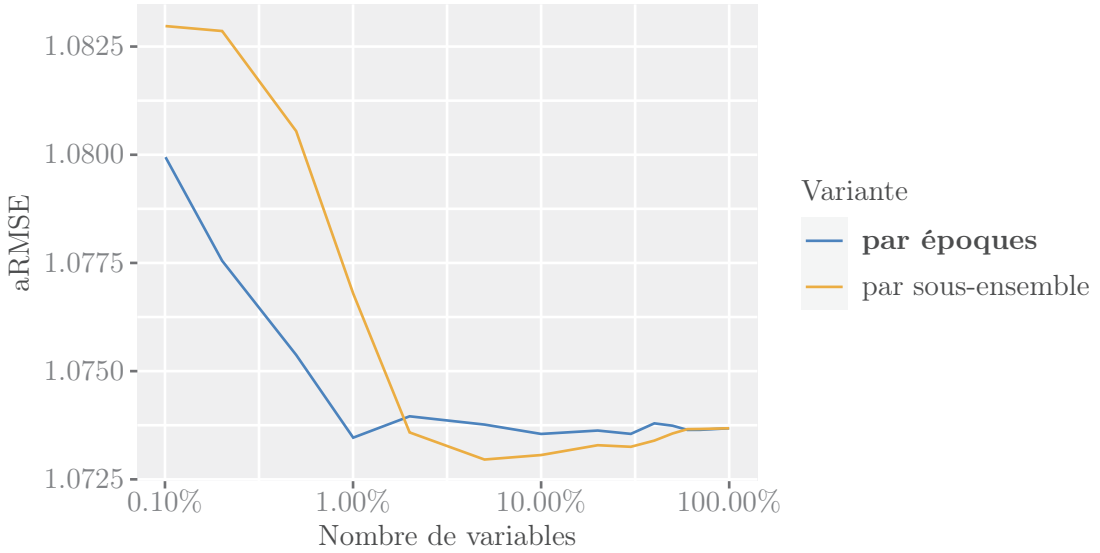


FIGURE 6.5 – Sélection de variables pour **RSMS** en comparant l'optimisation par époques, et l'agrégation de l'optimisation par sous-ensembles

les labels fréquents ont plus tendance à être sélectionnés que les labels moins fréquents. Pour les autres cas, nous présentons dans la table 6.7 la liste des labels présents dans moins de 1% des documents, et dans la table 6.8 la liste des labels présents dans plus de 1% des documents. La table montre la liste des qualifieurs relativement rares qui sont bien représentés par le champ lexical sélectionné, tandis que la table 6.8 montre que certains labels fréquents sont ignorés pour la sélection de variables. On note que l'image des produits est bien retranscrite, ainsi que les caractéristiques techniques les plus précises.

6.8 Application : algorithme LSMR après sélection de variables et de labels

Pour terminer, nous appliquons l'algorithme **LSMR** avec *bagging* sur les deux sous-ensembles de variables obtenus auparavant : les 993 variables de la sélection de l'agrégation de l'algorithme **RSMS**, et les 199 variables de la sélection de **RSMS** par époques. Nous ajoutons également les 993 et 199 meilleures variables selon *RFS* et *MIFS*. Le résultat est présenté dans la table 6.9. Pour rappel, effectuer l'apprentissage pour toutes les variables avec **LSMR** donne une métrique aRMSE de 0.809.

Le processus général pour l'apprentissage est résumé dans la figure 6.6. Une première étape consiste à appliquer la sélection de variables avec **RSMS**, en effectuant un découpage par mini-batch. Une fois la liste des variables obtenues, l'application de **LSMR** sur des sous-ensembles tirés avec remise permet d'obtenir plusieurs modèles, chacun

TABLE 6.7 – Rang de sélection des labels présents dans moins de 1% des documents, parmi les 30 meilleurs labels

N°	Rang	Description
65	9	Characteristics / Labelling / Braking on wet surfaces
135	10	Attitude / Towards brands / Towards asian and budget brands
54	11	Characteristics / Image / Origin
136	13	Attitude / Towards brands / Towards premium brands
67	14	Characteristics / Labelling / Rolling resistance
202	15	Image/Reputation / Reputation/Quality
50	16	Performances / Sportiness
30	17	Performances / Safety / Grip
57	18	Characteristics / Image / Awareness
204	19	Global Satisfaction
16	20	Performances / Services / Michelin OnWay / Warranty
212	21	Performances / Safety / Road Holding / On wet road / Under extreme conditions
211	22	Performances / Safety / Road Holding / On dry road or not specified
214	23	Performances / Safety / Safety Generic) / On wet road / Under extreme conditions
220	24	Performances / Safety / Grip / On dry road or not specified
221	25	Performances / Safety / Grip / On wet road
213	26	Performances / Safety / Safety Generic) / On dry road or not specified
207	27	Performances / Performances Generic)
19	28	Performances / Services / Other / Warranty
203	29	Image/Reputation / Recommendation
222	30	Performances / Safety / Motivity / On dry road or not specified

TABLE 6.8 – Rang de sélection des labels présents dans plus de 1% des documents

N°	Rang	Description
52	1	Performances / Consumer feedback / Overall satisfaction
59	2	Characteristics / Image / Reputation
60	3	Characteristics / Image / Test results
21	4	Performances / Safety / Safety
61	5	Characteristics / Image / Intent to purchase
58	6	Performances / Consumer feedback / Recommendation
10	7	Characteristics / Price / Value for money
132	8	Attitude / Towards tyre type
133	12	Attitude / Towards tests
549	31	Performances / Safety / Traction
483	32	Performances / Wear life / Wear life
780	33	Performances / Safety / Wet traction
506	34	Performances / Ride performance / Noise
494	35	Performances / Safety / Handling
504	36	Performances / Ride performance / Ride experience
781	37	Performances / Safety / Dry traction
782	38	Performances / Safety / Snow/Winter traction
497	39	Performances / Safety / Stopping distance
505	40	Performances / Ride performance / Ride comfort
495	41	Performances / Safety / Steering
487	44	Performances / Toughness/Robustness / Toughness/Robustness
484	47	Performances / Wear life / Ageing
491	48	Performances / Environment / Environment
508	49	Characteristics / Image / Look/aesthetic
488	50	Performances / Wear life / Abnormal Wear
489	52	Characteristics / Price / Deal / Discount
558	53	Performances / Track/Sport performance / Track performance
784	58	Performances / Safety / Off road traction

TABLE 6.9 – Évaluation avec **LSMR** + bagging (aRMSE) des deux ensembles de variables retenus : agrégation de **RSMS** sur des sous-ensembles, et optimisation de **RSMS** sur le jeu de données complet par époques

Pourcentage de variables	**RSMS** par époques	**RSMS** agrégé	*RFS*	*MIFS*
5% (993 variables)	0.810	**0.807**	0.815	0.819
1% (199 variables)	**0.810**	0.812	0.817	0.820

étant évalué sur un ensemble de validation afin d'obtenir une métrique. Nous obtenons également un ensemble hors du sac (*out of bag*), non utilisé pour l'apprentissage ou la validation, avec la sortie du modèle. La métrique obtenue par validation permet de sélectionner les modèles les plus pertinents. L'agrégation s'effectue ensuite en cherchant à prédire pour chaque label indépendamment, sa valeur à partir des valeurs prédites par les modèles de bootstrap. Afin de ne pas réutiliser les individus ayant servi à l'apprentissage ou à la validation, la valeur de ceux-ci est remplacée par la valeur de prédiction moyenne. On obtient donc les variables à identifier, des modèles de bootstrap, et un modèle de stacking.

Pour l'application du modèle sur un nouveau post, les étapes sont décrites dans la figure 6.7. Les mots du texte correspondants aux variables sélectionnés sont reconnus, puis chaque modèle de bootstrap effectue une prédiction, et enfin le modèle de stacking permet d'obtenir une valeur pour tous les labels.

6.9 Conclusion

Nous avons évalué notre algorithme de sélection de variables multi-labels, **RSMS**, sur un jeu de données textuel réel, en appliquant notre approche de régression multi-labels **LSMR** pour le valider. Selon notre mode d'application de **RSMS**, par époques ou en agrégeant les rangs des variables sélectionnées sur des sous-ensembles, nous obtenons respectivement une meilleure sélection de variables qualitativement, et une sélection plus adaptée au problème, en comparaison avec les algorithmes *RFS* et *MIFS*. Dans le premier cas, cette évaluation a montré l'utilité de la sélection de variables pour la réduction de dimension. Dans le second cas, elle a montré que la sélection de variables permet d'obtenir une plus faible erreur de régression.

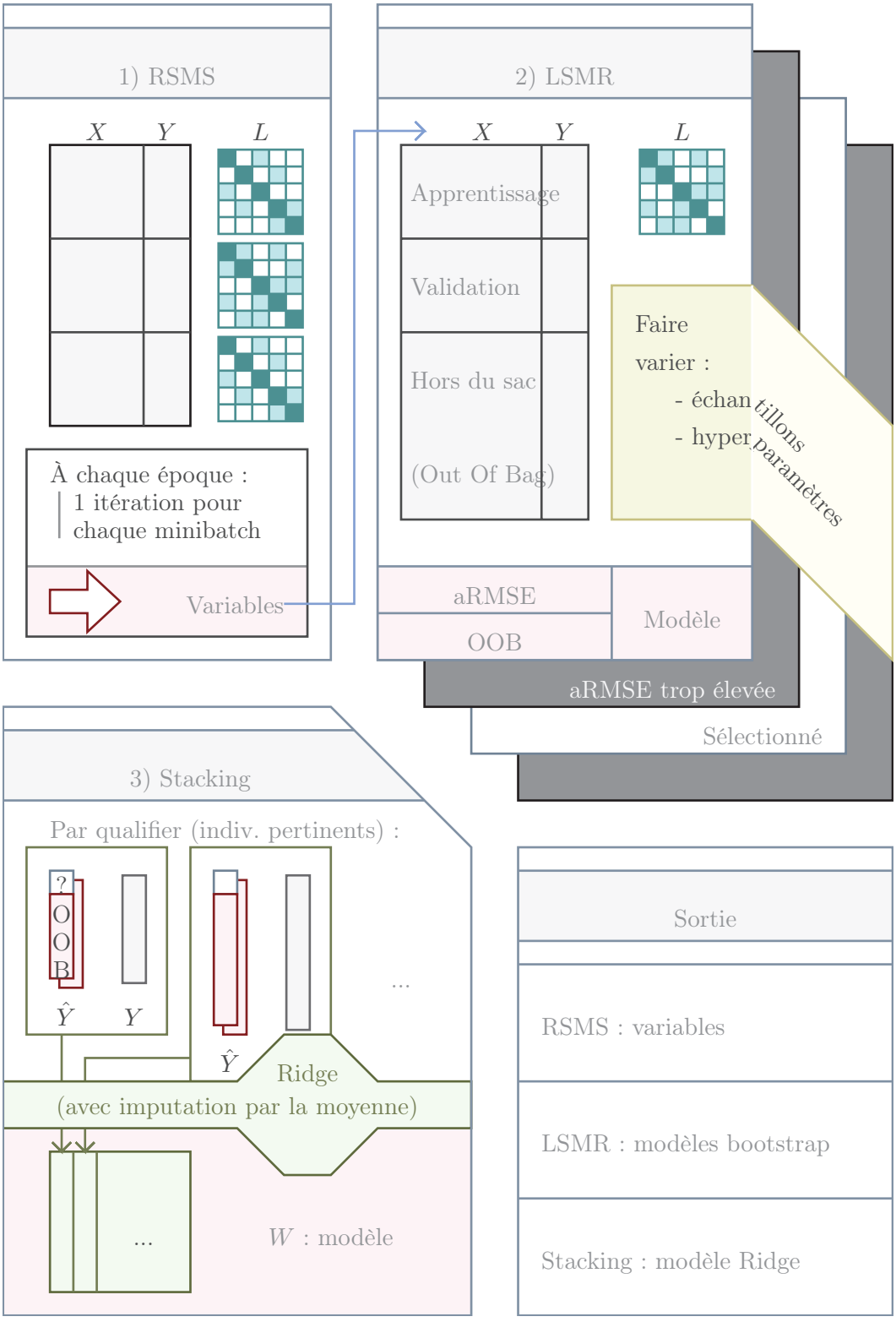


FIGURE 6.6 – Schéma de synthèse de l'application de **RSMS** et **LSMR** sur le jeu de données total

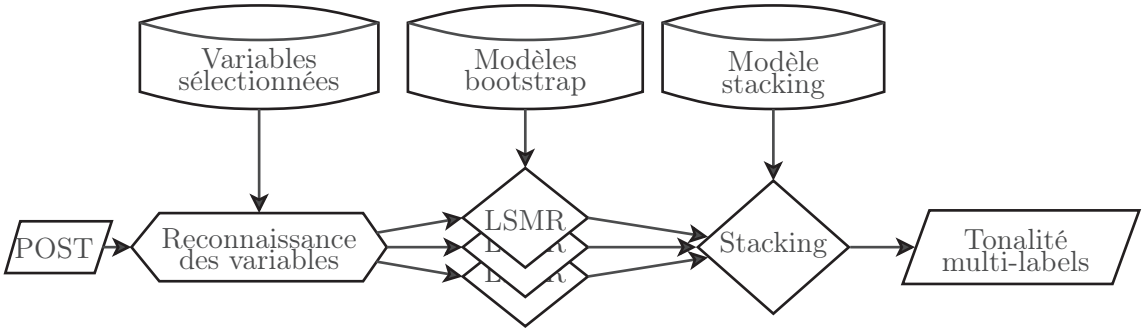


FIGURE 6.7 – Schéma de synthèse de l'application de **RSMS** et **LSMR** sur le jeu de données total : application sur un nouveau post non labellisé

Chapitre 7

Conclusion et perspectives

7.1 Bilan des travaux effectués

Dans cette thèse, nous avons présenté un état de l'art de l'apprentissage semi-supervisé et multi-labels pour la régression, en insistant également sur la sélection de variables. Ceci nous a amené à proposer un algorithme de régression semi-supervisé, *LapS3L*.

Cet algorithme utilise les données non labellisées de deux façons :

— une première étape d'extraction de variables non supervisée permet d'obtenir de nouvelles variables permettant d'exprimer une relation linéaire entre les variables et la variable cible. Cette étape est tirée de l'algorithme *SSSL*.

— une seconde étape qui utilise une régularisation Laplacienne, semi-supervisée, permettant de relier les variables réelles et les variables extraites.

LapS3L obtient de bons résultats de généralisation pour la régression, sur des jeux de données publics, et sur un jeu de données spécifique.

Afin d'adapter *LapS3L* aux problèmes de régression multi-labels, nous avons modifié la régularisation de la seconde partie en ajoutant un terme multi-labels : concrètement, si deux labels obtiennent des valeurs similaires sur les mêmes individus, alors on peut les considérer comme similaires. À partir de ces similarités entre labels, il est possible d'utiliser pour la régularisation l'hypothèse suivante : si deux labels sont similaires, alors les modèles servant à effectuer leur prédiction doivent être similaires.

Nous avons montré que l'algorithme résultant, *LSMR*, est compétitif avec l'état de l'art de la régression multi-labels, tout en donnant de meilleures performances que la régularisation multi-labels seule.

Pour faciliter l'apprentissage, nous nous sommes aussi penchés sur la sélection de variables semi-supervisée et multi-labels. Étant donné le nombre important de labels que nous serions amenés à traiter, nous avons proposé un compromis entre la sélection de variables mono-label et multi-labels en guidant la sélection de variables par la sélection de labels. Concrètement, nous avons conçu notre algorithme pour résoudre simultanément le problème de sélection de variables, et la sélection de labels, afin que les variables retenues ne puissent pas être influencées par les labels dont la modélisation est trop difficile ou impossible.

Enfin, nous avons appliqué **LSMR** et **RSMS** sur un jeu de données réel, de l'entreprise Lizeo, ayant de nombreuses particularités qui nous ont amené à adapter légèrement nos travaux. La version retenue de **LSMR** utilise la technique de *Bootstrap Aggregating*, ou *Bagging*, qui vise à ré-échantillonner (avec remise) le jeu de données pour apprendre un modèle par échantillon, puis une aggrégation des prédictions par un modèle de *stacking*. Ce modèle final nous permet d'ignorer les valeurs non pertinentes, une spécificité du jeu de données. Contrairement à *MIFS* et *RFS*, les variables sélectionnées par *RSMS* permettent soit d'obtenir une erreur de régression plus faible de *LSMR*, soit d'obtenir un modèle très simple.

7.2 Perspectives d'amélioration

Les pistes d'amélioration sont nombreuses pour nos travaux.

En ce qui concerne **LSMR**, nous pourrions tenter d'utiliser d'autres régularisations multi-labels. En effet, indépendamment de la performance de **LSMR**, les résultats obtenus par la régularisation Laplacienne multi-labels semblent montrer les limites de cette approche, en comparant avec la régularisation *CMTL*. Cette dernière est similaire, à ceci près que l'a-priori sur la structure des labels est inféré à partir des données. Nous pourrions également envisager d'intégrer la sélection de labels dans la méthode d'apprentissage **LSMR**, de façon à ne pas contraindre le modèle sur les labels qu'il ne pourra de toutes façons pas représenter fidèlement.

Nous pourrions aussi tenter de modifier les a-priori dont nous disposons sur les labels grâce aux connaissances métier. En effet, les labels forment une hiérarchie : certains sont plus génériques et recouvrent des sous-labels plus spécifiques. Par exemple, si le terme "savonnette" est tantôt employé pour la tenue sur route humide ou sur route enneigée, cela traduit une proximité naturelle entre ces deux labels. Cette proximité pourrait être retrouvée en inspectant simplement la hiérarchie des *qualifiers*.

Pour *RSMS*, nous pourrions envisager d'utiliser une régularisation non convexe, telle que la régularisation $\ell_{2,1-2}$.

En ce qui concerne l'application, il reste à harmoniser la classification et la régression. En effet, le score prédit pour chaque label n'est pas toujours intéressant ; il ne l'est que si le *qualifier* est bien mentionné dans le texte. Bien que l'agrégation par *stacking* permette d'ignorer le score prédit pour les qualifier non présents, les modèles bootstrap cherchent à obtenir la valeur 0 si le qualifier n'est pas présent. Cette idée devrait donc être au cœur de l'optimisation d'un algorithme unifiant la classification et la régression.

Annexe A

Publications personnelles

A.1 Revues internationales

— Seif-Eddine Benkabou, Khalid Benabdeslem, **Vivien Kraus**, Kilian Bourhis, Bruno Canitia, *Local Anomaly Detection for Multivariate Time Series by Temporal Dependency Based on Poisson Model*, in revision, *IEEE Transactions on Neural Networks and Learning Systems*

A.2 Conférences internationales

— **Vivien Kraus**, Khalid Benabdeslem, Bruno Canitia, *Laplacian-based Semi-supervised Multi-Label Regression.*, in *International Joint Conference on Neural Networks (IJCNN).* IEEE, 2020.

— **Vivien Kraus**, Seif-Eddine Benkabou, Khalid Benabdeslem, Frédéric Cherqui, *An improved Laplacian semi-supervised regression*, in *IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI).* IEEE, 2018. p 564-570.

A.3 Conférences nationales

— Seif-Eddine Benkabou, Khalid Benabdeslem, **Vivien Kraus**, Kilian Bourhis and Bruno Canitia. *Détection contextuelle d'anomalies à partir de séries temporelles multi-variées à base de modèle de Poisson*, in *Conférence sur l'Apprentissage automatique (CAp 2019).*

— **Vivien Kraus**, Khalid Benabdeslem, Frédéric Cherqui, *Régression Laplacienne semi-supervisée pour la reconstitution des dates de pose des réseaux d'assainissement*, *Extraction et Gestion des Connaissances (EGC18)* EGC 2018.

A.4 Travaux en cours

— **Vivien Kraus**, Khalid Benabdeslem, Bruno Canitia. *RSMS : Robust Semi-supervised Multi-label feature Selection for regression.* Preprint, 2021.

Bibliographie

[1] Ming Ji, Tianbao Yang, Binbin Lin, Rong Jin, and Jiawei Han. A simple algorithm for semi-supervised learning with improved generalization error bound. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 835–842, 2012.

[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization : A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, pages 2399–2434, 2006.

[3] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006.

[4] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[5] Gauthier Doquire and Michel Verleysen. A graph laplacian based approach to semi-supervised feature selection for regression problems. *Neurocomputing*, pages 5–13, 2013.

[6] Abdelouahid Alalga, Khalid Benabdeslem, and Nora Taleb. Soft-constrained laplacian score for semi-supervised multi-label feature selection. *Knowledge and Information Systems*, pages 75–98, 2016.

[7] Celso A R Sousa. Kernelized constrained gaussian fields and harmonic functions for semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[8] 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-Volume 1*, pages 421–426, 2006.

[9] Khalid Benabdeslem and Mohammed Hindawi. Constrained laplacian score for semi-supervised feature selection. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 204–218. Springer Berlin Heidelberg, 2011.

[10] Yu Zhang and Dit-Yan Yeung. Semi-supervised multi-task regression. *Machine Learning and Knowledge Discovery in Databases*, pages 617–631, 2009.

[11] Ling Jian, Jundong Li, Kai Shu, and Huan Liu. Multi-label informed feature selection. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1627–1633, 2016.

[12] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[13] Olivier Chapelle, Vikas Sindhwani, and Sathiya S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of machine learning research*, pages 203–233, 2008.

[14] Kristin P. Bennett and Ayhan Demiriz. Semi-supervised support vector machines. In *Proceedings of the 11th International Conference on Neural Information Processing Systems*, pages 368–374. MIT Press, 1998.

[15] Lin Chen, Ivor W Tsang, and Dong Xu. Laplacian embedded regression for scalable manifold regularization. *IEEE transactions on neural networks and learning systems*, pages 902–915, 2012.

[16] H. Scudder. Probability of error of some adaptive pattern-recognition machines,. *IEEE Transactions on Information Theory,*, pages 363–371, 1965.

[17] JS Cramer. The origins of logistic regression. *Tinbergen Institute Working Paper*, 2002.

[18] Jurica Levatić, Michelangelo Ceci, Dragi Kocev, and Sašo Džeroski. Self-training for multi-target regression with tree ensembles. *Knowledge-Based Systems*, pages 41–60, 2017.

[19] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[20] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100. ACM, 1998.

[21] Zhi-Hua Zhou and Ming Li. Semi-supervised regression with co-training. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 908–913, 2005.

[22] Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. *Advances in neural information processing systems*, pages 507–514, 2005.

[23] Hasna Barkia, Haytham Elghazel, and Alex Aussem. Semi-supervised feature importance evaluation with ensemble learning. In *2011 IEEE 11th International Conference on Data Mining*, pages 31–40. IEEE, 2011.

[24] Zheng Zhao and Huan Liu. Semi-supervised feature selection via spectral analysis. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 641–646. SIAM, 2007.

[25] Khalid Benabdeslem and Mohammed Hindawi. Efficient semi-supervised feature selection : Constraint, relevance, and redundancy. *IEEE Transactions on Knowledge and Data Engineering*, pages 1131–1143, 2014.

[26] Mohammed Hindawi and Khalid Benabdeslem. Local-to-global semi-supervised feature selection. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2159–2168. ACM, 2013.

[27] Ronan Collobert and Jason Weston. A unified architecture for natural language processing : Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[28] Andreas Argyriou, Massimiliano Pontil, Yiming Ying, and Charles A Micchelli. A spectral regularization framework for multi-task structure learning. In *Advances in neural information processing systems*, pages 25–32, 2008.

[29] S. Vijayakumar and S. Schaal. Locally weighted projection regression : An $\mathcal{O}(n)$ algorithm for incremental real time learning in high dimensional spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 288–293, 2000.

[30] Eleftherios Spyromitros-Xioufis, Grigorios Tsoumakas, William Groves, and Ioannis Vlahavas. Multi-target regression via input space expansion : Treating targets as inputs. *Machine Learning*, pages 55–98, 2016.

[31] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, pages 41–48, 2007.

[32] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society : Series B (Methodological)*, pages 267–288, 1996.

[33] Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep K Ravikumar. A dirty model for multi-task learning. *Advances in neural information processing systems*, pages 964–972, 2010.

[34] Jiayu Zhou, J Chen, and J Ye. Malsar : Multi-task learning via structural regularization. 2012.

[35] Yue Zhu, James T. Kwok, and Zhi-Hua Zhou. Multi-label learning with global and local label correlation. *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[36] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine learning*, pages 243–272, 2008.

[37] Hongyuan Zha, Xiaofeng He, Chris Ding, Ming Gu, and Horst D Simon. Spectral relaxation for k-means clustering. *Advances in neural information processing systems*, pages 1057–1064, 2001.

[38] Jiayu Zhou, Jianhui Chen, and Jieping Ye. Clustered multi-task learning via alternating structure optimization. *Advances in neural information processing systems*, pages 702–710, 2011.

[39] Tong Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 2010.

[40] Pinghua Gong, Jieping Ye, and Chang-shui Zhang. Multi-stage multi-task feature learning. In *Advances in neural information processing systems*, pages 1988–1996, 2012.

[41] Lei Han and Yu Zhang. Multi-stage multi-task learning with reduced rank. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1638–1644, 2016.

[42] Yong Shi, Jianyu Miao, Zhengyu Wang, Peng Zhang, and Lingfeng Niu. Feature selection with $\ell_{2,1-2}$ regularization. *IEEE Trans. Neural Netw. Learning Syst.*, pages 4967–4982, 2018.

[43] Yu-Feng Li, Ivor W. Tsang, James T. Kwok, and Zhi-Hua Zhou. Convex and scalable weakly labeled svms. *The Journal of Machine Learning Research*, pages 2151–2188, 2013.

[44] Sugato Basu, Ian Davidson, and Kiri Lou Wagstaff, editors. *Constrained clustering : advances in algorithms, theory, and applications*. CRC Press, 2009.

[45] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, 1999.

[46] Kristin P Bennett and Ayhan Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information processing systems*, pages 368–374, 1999.

[47] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 19–26. Morgan Kaufmann Publishers Inc., 2001.

[48] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, pages 103–134, 2000.

[49] David Azriel, Lawrence D Brown, Michael Sklar, Richard Berk, Andreas Buja, and Linda Zhao. Semi-supervised linear regression. *arXiv preprint arXiv :1612.02391*, 2016.

[50] Massih-Reza Amini and Patrick Gallinari. Semi-supervised logistic regression. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 390–394. IOS Press, 2002.

[51] Deng Cai, Xiaofei He, and Jiawei Han. Semi-supervised regression using spectral techniques. Technical report, 2006.

[52] Zhi-Hua Zhou and Ming Li. Semi-supervised regression with co-training. page 6.

[53] Kenneth Joseph Ryan and Mark Vere Culp. On semi-supervised linear regression in covariate shift problems. *Journal of Machine Learning Research*, pages 3183–3217, 2015.

[54] Amit Moscovich, Ariel Jaffe, and Boaz Nadler. Minimax-optimal semi-supervised regression on unknown manifolds. pages 933–942.

[55] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide : solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

[56] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, pages 278–282, 1995.

[57] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, pages 1207–1245, 2000.

[58] JP Davies, BA Clarke, JT Whiter, and RJ Cunningham. Factors influencing the structural deterioration and collapse of rigid sewer pipes. *Urban water*, pages 73–89, 2001.

[59] Mehdi Ahmadi, Frédéric Cherqui, Jean-Christophe De Massiac, and Pascal Le Gauffre. Influence of available data on sewer inspection program efficiency. *Urban Water Journal*, pages 641–656, 2014.

[60] Robert Richard Harvey and Edward Arthur McBean. Predicting the structural condition of individual sanitary sewer pipes with random forests. *Canadian Journal of Civil Engineering*, pages 294–303, 2014.

[61] Mingbo Zhao, Tommy W.S. Chow, Zhou Wu, Zhao Zhang, and Bing Li. Learning from normalized local and global discriminative information for semi-supervised regression and dimensionality reduction. *Information Sciences*, pages 286–309, 2015.

[62] Zhao Zhang, Tommy WS Chow, and Mingbo Zhao. Trace ratio optimization-based semi-supervised nonlinear dimensionality reduction for marginal manifold visualization. *IEEE Transactions on Knowledge and Data Engineering*, pages 1148–1161, 2012.

[63] Maxime Gasse, Alex Aussem, and Haytham Elghazel. On the optimality of multi-label classification under subset zero-one loss for distributions satisfying the composition property. In Francis R. Bach and David M. Blei, editors, *International Conference on Machine Learning*, pages 2531–2539, 2015.

[64] Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, pages 216–233, 2015.

[65] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, page 333, 2011.

[66] Weiwei Liu, Ivor W. Tsang, and Klaus-Robert Müller. An easy-to-hard learning paradigm for multiple classes and multiple labels. *The Journal of Machine Learning Research*, pages 3300–3337, 2017.

[67] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on Machine learning*, pages 1012–1019. ACM, 2005.

[68] Yao-nan Chen and Hsuan-tien Lin. Feature-aware label space dimension reduction for multi-label classification. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q.

Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1529–1537. Curran Associates, Inc., 2012.

[69] Minnan Luo, Lingling Zhang, Feiping Nie, Xiaojun Chang, Buyue Qian, and Qinghua Zheng. Adaptive semi-supervised learning with discriminative least squares regression. pages 2421–2427. International Joint Conferences on Artificial Intelligence Organization, 2017.

[70] Yu Nesterov. Gradient methods for minimizing composite functions. *Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), CORE Discussion Papers*, 2007.

[71] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, pages 281–305, 2012.

[72] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of machine learning research*, pages 1–30, 2006.

[73] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 42–50, 2011.

[74] Feiping Nie, Heng Huang, Xiao Cai, and Chris H. Ding. Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1813–1821. Curran Associates, Inc., 2010.

[75] Zhigang Ma, Feiping Nie, Yi Yang, Jasper RR Uijlings, and Nicu Sebe. Web image annotation via subspace-sparsity collaborated feature selection. *IEEE Transactions on Multimedia*, pages 1021–1030, 2012.

[76] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 1950.

[77] Leo Breiman. Bagging predictors. *Machine Learning*, pages 123–140, 1996.

[78] David Wolpert. Stacked generalization. *Neural Networks*, pages 241–259, 1992.

[79] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, pages 84–90, 2017.