Université de Rouen

Rapport de projet

Minimal Learning Machine

Étudiant Laziz Hamdi
Encadrant M. Simon Bernard

2020 — 2021

Remerciements

Il était agréable de m'acquitter d'une dette de reconnaissance envers tous ceux, dont la contribution au cours de ce projet, a favorisé son aboutissement.

Ainsi je tiens vivement à remercier mon encadrant Mr Simon Bernard qui n'a pas ménagé aucun effort pour m'aider et m'orienter le long de ce projet, merci Mr Simon Bernard. Je remercie aussi le professeur Mr Maxime Berar pour son encadrement et le soutien qu'il m'a donné.

Je remercie enfin toute personne qui a contribué de près ou de loin à l'élaboration de ce rapport.

Contents

Chapter 1

Introduction

En apprentissage automatique on distingue deux principales situations, dans le cas où les données à prédire ne sont pas étiquetés ce que l'on appelle l'apprentissage non supervisé. Dans le cas contraire lorsque les données à prédire sont étiquetés on appelle ça apprentissage supervisé. Il existe des tas de méthodes d'apprentissage supervisé parmi les plus connues on a les SVM (Support Vector Machine) ou encore RF (Random Forest). Il existe une technique qui est souvent utilisé en machine learning dans différentes tâches tel que la réduction de dimension, cette technique est l'utilisation d'une mesure de similarité ou dissimilarité entre les données.

Il existe plusieurs moyens de calculer une similiarité ou dissimilarité entre les données, le moyen le plus connue est sans doute les distances. Dans ce rapport je vous présente une méthode d'apprentissage supervisé qui se base sur le calcul de dissimilarité entre les données en utilisant les distances. Il s'agit du Minimal Learning Machine ou MLM. Ce document se divise en trois parties principales, chaque partie présente l'un des objectifs de ce TER qui sont dans un premier temps de comprendre et de documenter la méthode ensuite de reprendre les expérimentations présentées dans les articles du MLM pour ensuite implémenter une version de la méthode de référence qui serait "kernelizable" ou "pré calculé" et de tester cette version avec des Random Forest.

Chapter 2

Minimal Learning Machine

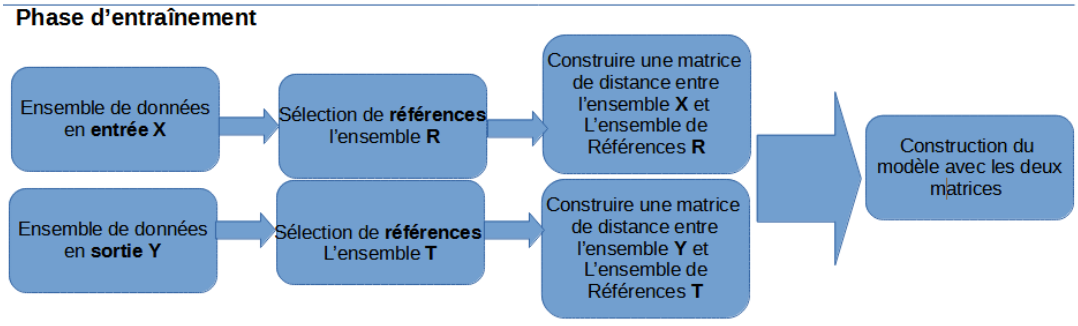
2.1 Définition

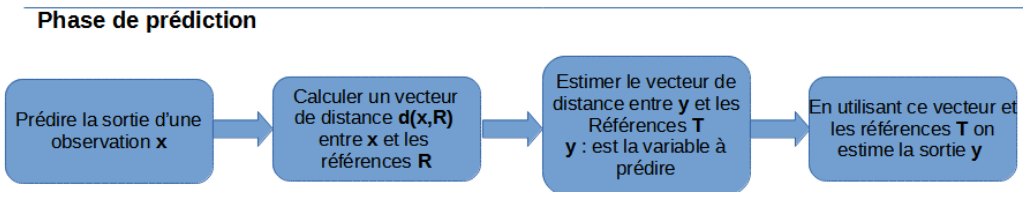
Le Minimal Learning Machine est une technique d'apprentissage supervisé apparu en 2015 qui fonctionne sur des problèmes de régressions ou de classifications. Dans sa phase d'entraînement, les données sont projetées dans un nouvel espace. Pour cela des observations aléatoires sont sélectionnées depuis les données, ensuite des distances qu'on appellera **"dissimilarités"** sont calculées entre ces observations et l'ensemble des données. Dans ce nouvel espace on ne cherche plus à prédire les mêmes données mais plutôt à prédire des distances. Et une fois ces distances prédites, on utilise le processus inverse pour estimer les réelles valeurs.

2.2 Phase d'entraînement

2.2.1 Formulation

Pour un ensemble de données en entrée $X = \{x_i\}_{i=1}^{N}$ avec $x_i \in R$ et sa correspondance en sortie $Y = \{y_i\}_{i=1}^{N}$ avec $y_i \in S$.
On suppose qu'il existe une relation continue entre ces deux espaces qu'on appelle $f : X \to Y$, l'objectif est d'estimer f.
Le processus du MLM se divise en deux étapes principales:





2.2.2 Construire les matrices de distances

Le MLM requière que l'utilisateur précise le nombre de points références k à sélectionner pour construire ce qu'on appelle des matrices de distances ou de dissimilarités. Si on considère D une matrice de distance de taille n, m, la valeur qui se trouve à l'intersection de la ligne i et la colonne j représente la distance euclidienne entre l'observation i et la référence j. De ce fait les lignes de D représentent l'ensemble des observations et les colonnes l'ensemble des points références.
On définit l'ensemble des points références $R = \{m_k\}_{k=1}^{k}$ sélectionnés aléatoirement de l'ensemble X et leurs correspondances $T = \{t_k\}_{k=1}^{k}$ de l'ensemble Y, ensuite on construit la matrice $D_x \in R^{N \times K}$ des distances euclidiennes entre les observations x_i en ligne et les références en colonne, de la même manière est définit la matrice des distances en sortie $\Delta_x \in R^{N \times K}$.
On définit la relation entre ces matrices ainsi $\Delta_y = g(D_x) + E$ avec E qui représente le résidu.
On peut représenter cette relation sous forme matricielle $\boldsymbol{\Delta_y = D_x B + E}$.
$\boldsymbol{B}$ est la matrice des coefficients du modèle.

2.2.3 Construire le modèle

Pour estimer $\boldsymbol{B}$ c'est à dire les coefficients du modèle, plusieurs méthodes peuvent être utiliser comme les moindres carrés moyens, moindres carrés récursifs pour calculer la différence entre les vraies distances et les distances prédites. cette différence est exprimé avec cette fonction.
$\boldsymbol{RSS(B) = tr((\Delta_y - D_x B)'(\Delta_y - D_x B))}$
Minimiser cette fonction revient à chercher le point ou le gradient est null, ce qui conduit à résoudre un système d'équations ou le nombre d'équations est le nombre d'observations $\boldsymbol{N}$ et le nombre d'inconnue est le nombre de références $\boldsymbol{K}$. La solution est différente selon le nombre $\boldsymbol{K}$.

- Pour $\boldsymbol{K < N}$: $\boldsymbol{\hat{B} = (D_x{}' D_x)^{-1} D_x{}' \Delta_y}$
- Pour $\boldsymbol{K = N}$: $\boldsymbol{\hat{B} = D_x{}^{-1} \Delta_y}$
- Pour $\boldsymbol{K > N}$: une infinité de solutions.

Ce dernier cas se produit lorsque après sélection des points références uniquement une partie des données est utilisé pour créer le modèle, ceci donne naissance à un problème indéterminé car le nombre d'équations est plus petit que le nombre d'inconnu avec une infinité de solution.

2.3 Phase de prédiction

Une fois $\boldsymbol{B}$ estimer pour un point en entrée $\boldsymbol{x}$, on construit un vecteur de distances euclidienne entre ce point et l'ensemble des point références $\boldsymbol{d(x, R) = [d(x, m_1)...d(x, m_k)]}$, alors dans le cas ou $\boldsymbol{K = N}$ ou $\boldsymbol{k < N}$, le vecteur des distances en sortie est le produit entre le vecteur des distances en entrée et la matrice des coefficients.
$\boldsymbol{\hat{\delta}(y, T) = d(x, R)\hat{B}}$ avec $\boldsymbol{\hat{\delta}(y, T) = [\hat{\delta}(y, t_1)...\hat{\delta}(y, t_k)]}$

$\boldsymbol{y}$ est estimé en utilisant le vecteur des distances $\boldsymbol{\hat{\delta}(y, T)}$ et les références $\boldsymbol{T}$ avec ce qu'on appelle une **multitaréation**. C'est une technique qui utilise des mesures de distance pour relever les coordonnées spatiales de positions inconnues. En pratique les distances sont mesurées avec erreur, et les méthodes statistiques peuvent quantifier l'incertitude de l'estimation de la

position inconnue. De nombreuses méthodes d'estimation de la position d'un point par multitération peuvent être utiliser comme un estimateur linéaire des moindres carrés, un estimateur des moindres carrés pondéré de manières itérative et une technique non linéaire des moindres carrés. En général la technique des moindres carrés non linéaire est la plus performante.
Pour estimer $\boldsymbol{y}$ on minimise la fonction objective suivante :
$\boldsymbol{\jmath(y) = \sum_{k-1}^{k}((y - t_k)\prime(y - t_k) - \hat{\delta}^2(y, t_k))^2}$
Cette fonction de coût possède un minimum en 0 qui est atteint seulement si la valeur estimer est égal à la vraie valeur, c'est à dire que les valeurs prédites sont égales aux valeurs réels $\boldsymbol{\prime y = y}$. Sinon on approche le plus possible $\boldsymbol{y}$ avec un algorithme de minimisation.

Plusieurs algorithmes de minimisation peuvent être utilisés mais le plus adapter pour ce problème est L'algorithme de **Levenberg-Marquardt** qui permet de trouver une solution numérique à un problème de minimisation d'une fonction non linéaire dépendant de plusieurs variables. Cet algorithme est plus stable et trouve une solution même s'il démarre très loin du minimum.

2.4 Choix du paramètre K

Le principal avantage du MLM est qu'il ne possède qu'un seul hyper paramètre K à optimiser le nombre de points références que l'utilisateur doit rentrer. Pour optimiser le nombre de points références la validation croisée est utilisée, en divisant l'ensemble des données en F sous-ensembles ensuite en testant avec différentes valeurs de $\boldsymbol{k}$ les taux de réussites sont calculés avec les moindres carrés moyens pour les vecteur de distances en sortie $\hat{\boldsymbol{\delta}}$ et l'estimation des $\hat{\boldsymbol{y}}$

$$\boldsymbol{AMSE(\delta) = \frac{1}{K}\sum_{k=1}^{k}\frac{1}{N_v}\sum_{i=1}^{N_v}(\delta(y_i, t_k) - \hat{\delta}(y_i, t_k))^2}$$
$$\boldsymbol{AMSE(y) = \frac{1}{4}\sum_{s=1}^{S}\frac{1}{N_v}\sum_{i=1}^{N_v}(y_i^{(s)} - \hat{y}_i^{(s)})^2}$$

Les points références sont sélectionnés aléatoirement des données.

2.5 Performances et Complexité

La complexité pour l'étape de formation du modèle dépend fortement de la méthode utilisée pour le calcul des inversions de matrice surtout qu'on construit des matrices de distances toujours plus grandes selon la taille des données et le nombre de points références sélectionnées, puisque on construit des matrices de tailles $\boldsymbol{N, K}$.
L'une des méthodes les plus connue est l'inverse de **Moore-Penrose** qui estime une pseudo inverse de la matrice, car dans certains cas les matrices ne sont pas inversibles. L'une des constructions les plus connues de cette méthode est la décomposition en valeurs singulières $\boldsymbol{SVD}$ qui est très précise mais très gourmande en temps de calcule et est plusieurs fois plus élevé que le produit matrice-matrice.

Pour accélérer le calcul, plusieurs méthodes ont été proposé comme le produit entre un type spécial de tenseur et une décomposition $\boldsymbol{QR}$ ou encore un algorithme basé sur une factorisation **Cholesky**.
La complexité de la phase d'entrainement du MLM est $\boldsymbol{\Theta(K^2N)}$ c'est similaire à celle d'un algorithme de machine learning lorsque le nombre de neurones cachés est égal au nombre de références K.

Le MLM est testé sur 12 ensembles de données les plus fréquemment utilisés dans le monde ensuite ces performances sont comparées à celle de cinq autre méthodes de références le

machine learning, extrême learning machine **ELM**, le réseau de fonction à base radial **RBF**, les machines à vecteur de support **SVM**, les processus gaussiens **GP** et le percepteron multicouches **MLP**. Tous les ensembles de données sont près traités de la même manière pour reproduire les expériences à l'identique. Supprimer les données manquantes, supprimer les données catégorielles, normaliser de la même façon et utiliser la même proportion de données pour l'entraînement et le test.

Pour ces tests le seul hyper paramètre K du MLM est optimisé avec une validation croisée sur 10-Fold, avec une sélection aléatoire des références depuis l'ensemble de données pour k allant de 5% à 100% avec un pas de 5%. Tous les modèles sont évalués en utilisant l'erreur quadratique moyenne **MSE** sur 10 tests indépendants. Le MLM obtient le plus petit taux d'erreur pour 5/8 des problèmes de régressions et pour les autres problèmes il obtient des résultats proches des résultats obtenue par les autres méthodes.
Ces expériences montrent qu'en utilisant 20% des points d'apprentissages comme points références semble être un bon choix pour la plupart des ensembles de données.

Chapter 3

Expérimentations

Pour évaluer les performances du Minimal Learning Machine, les chercheurs qui ont inventé la méthode ont comparé les résultats obtenus avec le MLM aux résultats obtenus avec les méthodes de machine learning les plus connus, sur 12 datasets, leur dimensions sont détaillés dans la table ci-dessous.

Dataset	Type	Dim		# Samples	
		In	Out	Train	Test
Ailerons	R	5	1	4752	2377
Elevators	R	6	1	6344	3173
Breast Cancer	R	32	1	129	65
Boston Housing	R	13	1	337	169
Servo	R	4	1	111	56
Abalone	R	8	1	2784	1393
Stocks	R	9	1	633	317
Auto Price	R	15	1	106	53
Wisconsin Breast Cancer	C	30	2	379	190
Pima Indians Diabetes	C	8	2	512	256
Iris	C	4	3	100	50
Wine	C	13	3	118	60

Lors de ce projet on a reproduit certaines des expérimentations présentées dans l'article scientifique de la méthode sur 8/12 des précédents datasets n'ayant pas retrouvés les 4 autres.

On a suivi le même protocole pour tous les datasets, les données sont centrées et réduites avec la moyenne et écart type des données d'entraînements , les variables catégorielles ainsi que les instances qui contiennent des données manquantes sont supprimées. Dix différentes permutations aléatoires sont appliquées pour chaque dataset.2/3 de l'ensemble de données sont utilisés pour la phase d'entrainement du modèle et 1/3 pour la phase de test.
Il est a noté que pour la classification les proportions des classes ont été équilibré. Chaque classe est représentée avec une proportion égale pour la phase d'entrainement et de test.

Le seul hyper paramètre du Minimal Learning Machine c'est à dire le nombre de point référence, est optimisé avec une validation croisée sur 10 cv, en allant de 5% à 100% de la taille totale de l'ensemble de donnée. Les hyper paramètres des SVM sont optimisé avec un grid search et une cross validation avec des valeur logarithmiques de $\mathbf{2^{-2}}$ à $\mathbf{2^{10}}$, pour le SVM on utilise le noyau gaussien. Pour les Random Forest les paramètres optimisés sont la profondeur

maximale de 10 à 100 avec un pas de 20, le critère utilisé est le "mse" pour la régression et l'indice de "gini" pour la classification. Le nombre de données minimal pour continuer de diviser créer des noeuds "n-min-split" vas de 2 à 50 avec un pas de 5 et le nombre d'arbres vas de 100 à 400 avec un pas de 50. Comme pour le SVM le RF est optimisé avec un grid search et une validation croisée.

3.1 Sur des problèmes de régressions

Pour calculer les taux de précisions le Mean Square Error est utilisé, sont affichés dans la figure 3.1 les résultats du MLM comparé au SVM et Random Forest ainsi que leur temps d'exécutions sur les quatre datasets pour la régression.

	MLM (C)	SVM (RBF)	RF
Servo	6.24e-1	6.9-1	6.14e-1
Auto price	3.09+e6	5.74e+6	3.03e+6
Boston Housing	2.41e+1	2.36e+1	1.4e+1
Abalone	4.99	4.5	5.17

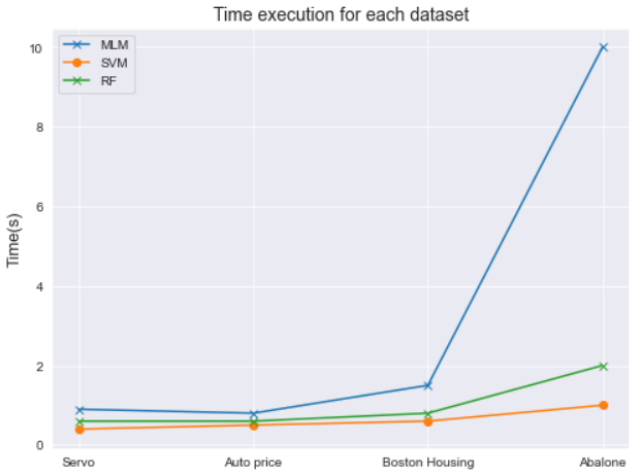


Figure 3.1: MSE results, Time execution

On voit que le RF obtient le meilleur score sur les datasets Servo ,Auto Price et Boston Housing alors que le SVM obtient le meilleur score sur le dataset Abalone. Mais on constate que les résultats du MLM sont très proches des résultats des deux autres méthodes

La figure suivante représente l'évolution de l'erreur en fonction du nombre de références $\boldsymbol{K}$ c'est à dire pour chaque valeur que prend $\boldsymbol{K}$ on calcule le Mean Square Error entre les valeurs prédites et les valeurs réelles ensuite cette valeur est normalisée en la divisant sur le Mean Square Error de ces valeurs prédites.

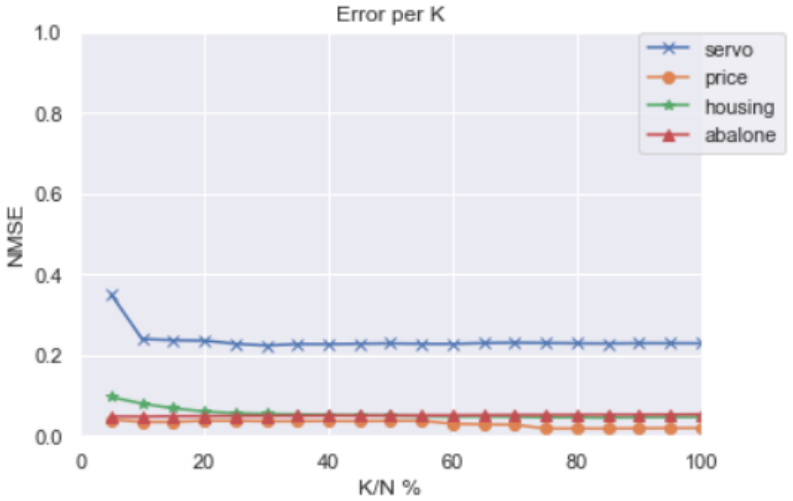
On voit que les 4 courbes évoluent de la même façon.

Dans la figure qui suit, on affiche le nombre optimal de points références pour chaque dataset sur 10 différentes exécutions.

3.2 Sur des problèmes de classifications

Pour la classification on utilise le même processus utilisé pour la régression sauf pour le calcul du taux de précision on utilise l'accuracy.

Le Minimal Learning Machine obtient le meilleur score pour le dataset Wine et Iris, le SVM est meilleur sur les dataset Iris et Wisconsin Breast Cancer et les RF sont meilleurs sur



	MLM	SVM	RF
Iris	0.98	0.98	0.96
Wine	0.979	0.875	0.958
Prima Indians Diabetes	0.748	0.782	0.793
Wisconsin Breast Cancer	0.971	0.978	0.978

Figure 3.2: Accuracy results, Time execution

les deux datasets Prima Indians Diabetes et Wisconsin Breast Cancer

L'évolution de l'erreur en fonction du nombre de points références:

Le nombre optimal de points références pour les trois datasets Breast Cancer, Iris et Wine sur 10 différentes exécutions:

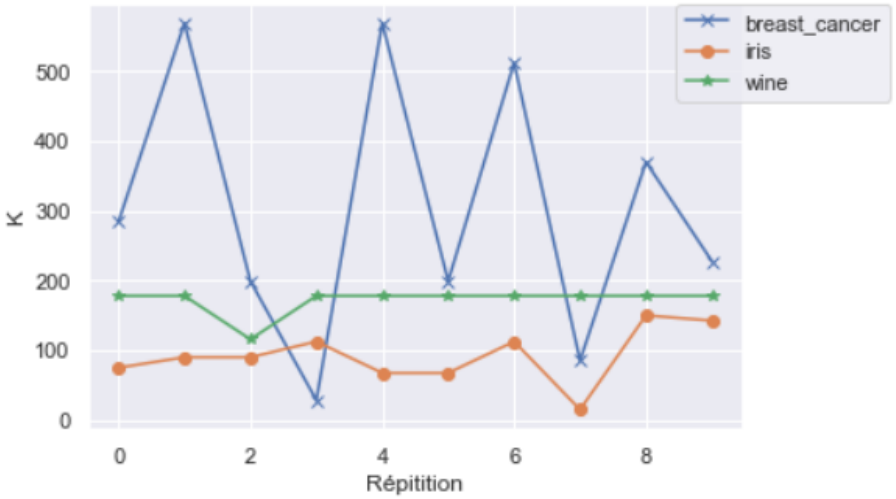


Figure 3.3: Optimal K value over different runs

Comme c'est difficile de faire des tests sur des données réelles car l'exécution peut prendre énormément de temps, surtout pour optimiser les hyper paramètres, avec une validation croisée ou une recherche en grille. Donc trois nouveaux ensembles de données de 100 observations à deux dimensions pour 500 instances sont générés, pour comparer les performances du Minimal Learning Machine au SVM et RF. Les données sont près traitées toujours de la même façon.

Les trois datasets sont générer aléatoirement en utilisant les fonction "make classificatio","make moons" et "make circle".

Les trois modèles n'ont pas été optimisé, le nombre de points références est fixé à 0.3 pour le MLM pour le SVM et RF on utilise aussi les valeurs des paramètres par défaut.

Les données sont près traitées de la même façon que dans la section précédente.

Les résultats des trois modèles sont affichés dans ce tableau :

	RBF SVM	RF	MLM
make moon	0.970588	0.911765	0.911765
make circle	0.941176	0.941176	0.911765
linearly separable	0.970588	0.970588	0.941176

Figure 3.4: Accuracy score for SVM,RF and MLM

On voit que les SVM et RF obtiennent les meilleurs résultats mais l'écart avec les résultats du MLM n'est pas grand.

Pour montrer l'influence du nombre de points référence utilisé sur la complexité de l'algorithme, on test le MLM sur des données générées aléatoirement avec la fonction "make regression" avec 500 instances pour 5 features, on compte le temps d'exécution sur les mêmes données en utilisant un nombre de points références qui vas de 5% à 100% de la taille des données avec un pas de 5%

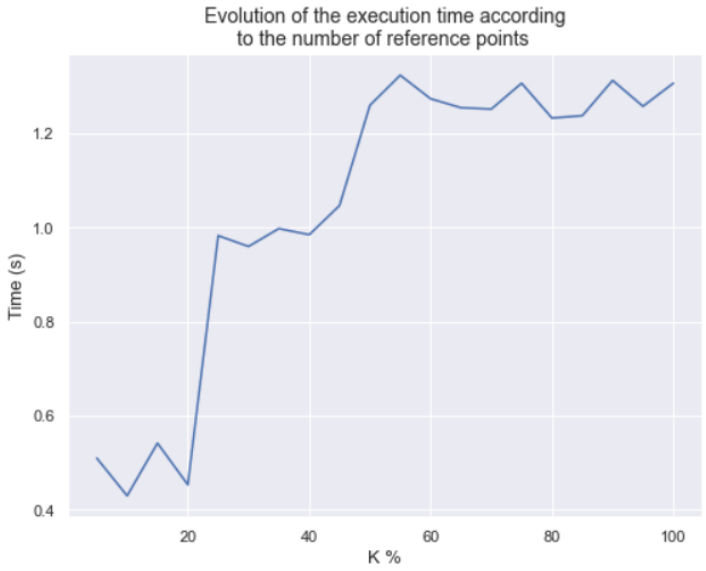


Figure 3.5: Time evolution per K

Comme attendu la complexité augumente fortement en parallèle au nombre de références, mais ce n'est pas grave du moment qu'à l'étape de prédiction le processus est rapide.

Chapter 4

Implémentation d'une version pré calculé

Maintenant l'objectif est de réussir à implémenter une version pré calculé de la méthode. Le terme **"pré calculé"** veut dire que dans cette version on donne au modèle les matrices de distances près calculés, directement lors de la formation et aussi lors de la prédiction, ainsi on peut définir notre propre manière de calcul des dissimilarités et donner au modèles les matrices prêtes à être utiliser directement.

Avant de pouvoir modifier le code de la méthode, il fallait d'abord coder notre propre version en suivant la procédure décrite dans l'article pour s'approprier la méthode et pouvoir la modifier plus facilement en cas de besoin.

La classe principale de la méthode s'appelle MLM, en plus du constructeur elle contient trois parties essentielles, une partie pour la sélection de points références et calcul des matrices de distances, une partie pour la formation du modèle et calcul de la matrice des poids et une partie pour la prédiction et optimisation de la fonction de coût. Pour la classification le même processus est utilisé seulement on transforme d'abord les labels avec **one hot encoding**.

Après avoir coder la version pré calculé du Minimal Learning Machine, on l'a testé sur les datasets précédemment utilisés avec la même mesure de distance pour vérifier qu'on retrouve bien les mêmes résultats. Ensuite à l'aide de la fonction **cdist** de scipy on calcule les matrices de distances avec différentes mesures et on compare avec les résultats obtenus en utilisant la distance euclidienne. Une explication des mesures utilisés est donné ici :

- La distance **Caneberra** entre deux points $\boldsymbol{u}, \boldsymbol{v}$: $\boldsymbol{d(u,v)} = \sum_i \frac{|u_i - v_i|}{|u_i| + |v_i|}$
- La distance **chebyshev** entre deux points $\boldsymbol{u}, \boldsymbol{v}$: $\boldsymbol{d(u,v)} = \boldsymbol{max_i}|\boldsymbol{u_i} - \boldsymbol{v_i}|$
- La distance **cityblock** entre deux points $\boldsymbol{u}, \boldsymbol{v}$: calcul la distance de Manhattan
- La distance **sqeuclidean** entre deux points $\boldsymbol{u}, \boldsymbol{v}$: $\boldsymbol{d(u,v)} = ||\boldsymbol{u} - \boldsymbol{v}||_2^2$

  On a testé le Minimal Learning Machine sur les 8 datasets précédemment utilisés, les données ont été pré traité de la même façon que dans la partie Expérimentations. Pour chaque mesure de distance sur chaque dataset une cross validation de 5% à 100% de la taille des données avec un pas de 5% pour optimiser le nombre de points références. Pour la régression on utilise le MSE et pour la classification on utilise l'Accuracy comme métriques, les résultats sont affiché dans les figures 4.1 et 4.2 .

Avec la distance euclidienne le MLM obtient le meilleur score pour les datasets Boston Housing et Abalone. Les distances sqeueclidienne et cityblock permettent d'avoir les meilleurs

	euclidean	canberra	chebyshev	cityblock	sqeuclidean
Servo	0.624081	0.792614	0.623477	0.580101	0.497879
Auto price	3234830.651756	2714728.274372	8811892.847048	2165361.789862	7174472.849607
Boston Housing	15.160613	15.315436	25.309695	15.358098	107.667046
Abalone	5.072464	5.317277	5.504687	5.193842	265.565567

Figure 4.1: Regression results (MSE)

	euclidean	canberra	chebyshev	cityblock	sqeuclidean
Iris	0.980000	0.840000	0.900000	0.960000	0.600000
Wine	0.979167	0.854167	0.916667	0.937500	0.520833
Prima Indians Diabetes	0.748603	0.547486	0.776536	0.659218	0.664804
Wisconsin Breast Cancer	0.971831	0.880282	0.957746	0.943662	0.880282

Figure 4.2: Classification results (Accuracy)

résultats pour les datasets Auto price et Servo. Pour la classification la distance euclidienne donne le meilleur score pour les dataset Iris, Wine et Wisconsin Breast Cancer tandis que chebyshev donne le meilleur résultat pour le Prima Indians Diabetes. En général la distance euclidienne permet d'avoir de bons résultats sur la plupart des problèmes, néanmoins sur certains problèmes d'autres mesures de distances sont meilleures.

Chapter 5

Minimal Learning Machine avec Random Forest

Il y a plusieurs façons de calculer des dissimilarités en machine learning, la plus connue d'entre elles est la distance euclidienne. Notre objectif est d'implémenter notre propre version de la méthode Minimal Learning Machine en utilisant les Random Forest pour calculer les dissimilarités, ensuite de comparer les résultats obtenus avec cette méthode aux résultats du Minimal Learning Machine, SVM et Random Forest.

5.1 Forêt aléatoires

Les forêts aléatoires font partie des techniques d'apprentissage automatique, qui combinent les concepts de sous-espaces aléatoires et de **bagging**.L'idée est de créer plusieurs copies d'un même modèle dans notre cas un arbre de décision, en entrainant chaque copie sur une partie aléatoire du dataset. Pour ça on utilise une technique d'échantillonnage appelé **Bootstrapping** et qui consiste à replacer après chaque tirage au sort les données qui ont été sélectionné dans notre dataset, de cette manière on obtient une foule de modèles diversifiés puisque il n'ont pas été nourri sur les mêmes données mais qui partagent quand même certaines connaissances en commun, avec ça on regroupe les résultats de chaque modèle pour faire une prédiction final.

5.2 Arbre de décision

Un arbre de décision est une méthode de machine learning ou l'on construit un arbre sur les données d'apprentissages, chaque nœud interne décrit un test sur une variable d'apprentissage, chaque branchement représente un résultat du test et chaque feuille contient la valeur de la variable cible (une étiquette de classe pour les arbres de classification, une valeur numérique pour les arbres de régression).

5.3 Calcul de dissimilarité

Les Random Forest présentent une mesure de similiarité pour une paire d'instances $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ qui prend en compte les labels correspondant $\boldsymbol{y_i}$ et $\boldsymbol{y_j}$ à l'inverse de la distance euclidienne par exemple qui prend en compte uniquement les deux instances $\boldsymbol{d(x_i, x_j)} = \sqrt{\sum_{k=1}^{d}(x_i^{(k)} - x_j^{(k)})^2}$. On dit que $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ sont similaire si elles sont proches l'une de l'autre, mais aussi si elles appartiennent à la même classe. $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ sont similaire si elles suivent le même chemin dans l'arbre de décision.

- On définit l'ensemble de feuille $\boldsymbol{L_k}$ dans $\boldsymbol{h_k}$

- soit $\boldsymbol{l_k : X \rightarrow L_k}$ une fonction qui attribue à chaque $\boldsymbol{x}$ une prédiction c'est à dire une feuille de l'ensemble $\boldsymbol{L_k}$.
- Dans la figure qui suit $\boldsymbol{l_k(x_i) = N_{12}}$

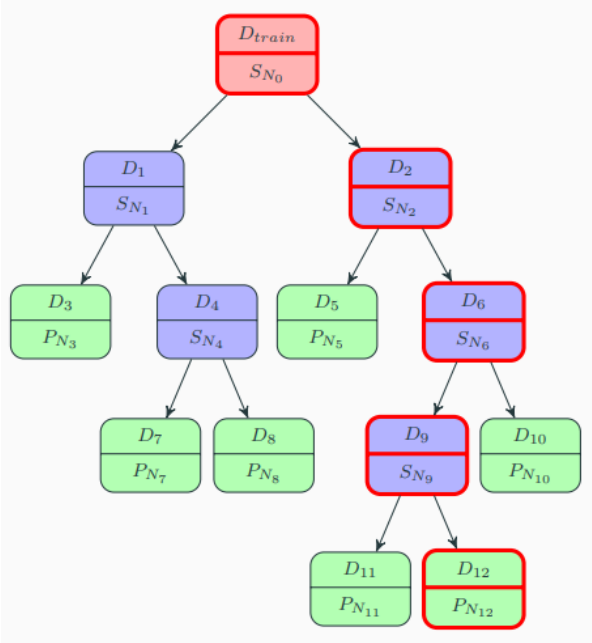
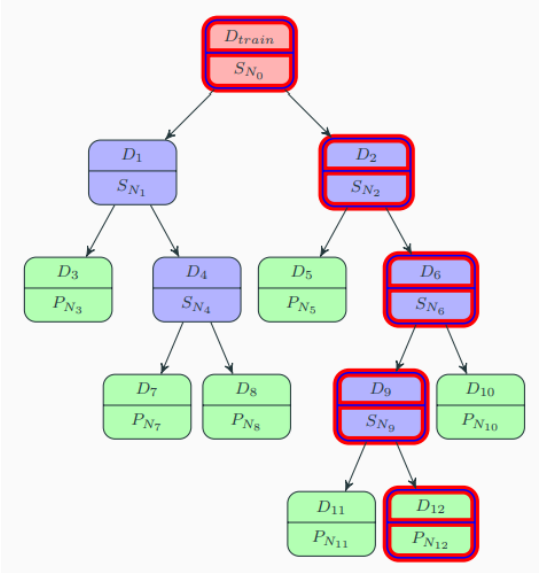


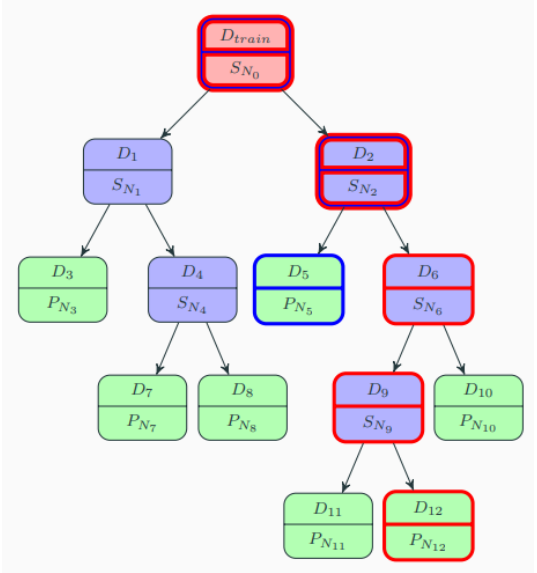
Figure 5.1: Path from root to leaf

La similarité $\boldsymbol{d^{(k)}(x_i, x_j)}$ entre $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ donné par $\boldsymbol{h_k}$ est $\boldsymbol{d^{(k)}(x_i, x_j)} = \begin{cases} \mathbf{1} & \text{if} \boldsymbol{l_k(x_i) = l_k(x_j)} \\ \mathbf{0} & \text{otherwise} \end{cases}$

- La figure (a) représente le cas ou $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ tombent dans la même feuille $\boldsymbol{d^{(k)}(x_i, x_j) = 1}$
- La figure (b) représente le cas ou $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$ ne tombent pas dans la même feuille donc $\boldsymbol{d^{(k)}(x_i, x_j) = 0}$



(a) $\boldsymbol{d^{(k)}(x_i, x_j) = 1}$



(b) $\boldsymbol{d^{(k)}(x_i, x_j) = 0}$

5.4 Comparaison des résultats

Comme cette version du MLM lance un Random forest à l'intérieur et derrière lance aussi un Minimal Learning Machine, la première question qui nous vient à l'esprit c'est est ce qu'on combinant les deux modèles on obtient de meilleurs résultats qu'avec chaque modèle séparément. Donc pour comparer les résultats des trois modèles on les a testés sur les 8 datasets utilisés précédemment. Les données ont été traité en suivant le même protocole expliqué dans la partie 3 Expérimentations. Concernant les modèles pour une comparaison équitable,pour le Minimal Learning Machine on prend le nombre de points références égale au nombre de données et comme mesure de distance on prend la distance euclidienne. Le Random Forest on le laisse avec les paramètres par défauts donc le nombre d'estimateurs = 100. Le Minimal Learning Machine avec Random Forest dissimilarités prend les mêmes valeurs des paramètres que pour ces deux modèles.

	RFD_MLM	RF	MLM
Servo	1.15	1.14	1.25
Auto price	3.59e+6	3.13e+6	6.80e+6
Boston Housing	12.1	13.5	10.69
Abalone	5.15	4.80	5.07

(a) Regression results (MSE)

	RFD_MLM	RF	MLM
Iris	0.935	0.935	0.84
Wine	0.962	0.975	0.975
Prima Indians Diabetes	0.739	0.750	0.779
Wisconsin Breast Cancer	0.969	0.963	0.967

(b) Classification results (Accuracy)

Figure 5.3: Comparison between RF, RFD-MLM, MLM

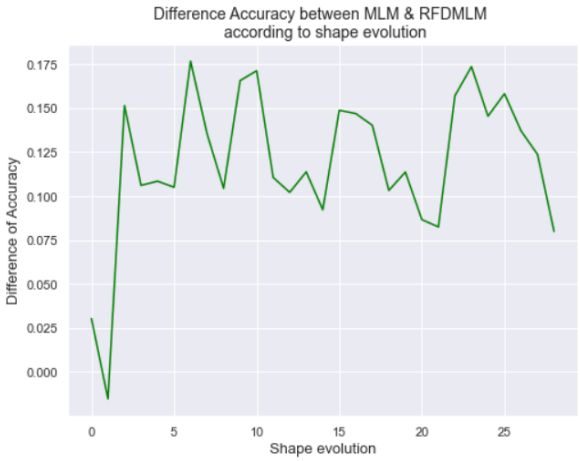
On voit que le Random Forest seul obtient le meilleur score sur 3/4 des datasets pour la régression et aussi sur 2/4 des datasets pour la classification. Le MLM avec Random Forest dissimilarités est meilleur sur 2/3 des datasets pour la classification tant dis que le Minimal Learning Machine est meilleur sur 2/4 des problèmes de classification et 1/4 des problèmes de régression.

On voit que le Random Forest tout seul reste meilleur que le Minimal Learning Machine ou en combinant les deux. Néanmoins la différence entre les résultats n'est pas très grande entre les trois modèles et surtout entre le RF et RFD-MLM et d'ailleurs en voit qu'en moyenne le RFD-MLM est meilleur que le MLM.

5.5 Comportement du MLM et RF dans des espaces à grandes dimensions

On teste le Minimal Learning Machine et RFD-MLM sur deux problèmes joués, un pour la régression et un autre pour la classification. Les données sont générées aléatoirement en utilisant les fonctions **make classification** et **make regression** de sklearn, pour la classification on fixe les paramètres (n-classes=3,n-informative=2) pour la régression on fixe les paramètres (n-informative=2). Sur chaque un des cas on teste les deux modèles sur 30 tirages aléatoires mais on incrémente le nombre d'instances de 400 et le nombre de features de 40 donc après chaque tour sur 30 itérations. Les deux paramètres varient de 400 à 12000 et de 40 à 1200 respectivement. Pour chaque test les données sont divisées en deux ensembles 2/3 pour le train et 1/3 pour le test, ensuite on standardise les données avec la moyenne et l'écart type des

(a) Regression results (MSE)



(b) Classification results (Accuracy)

données d'entrainements. Pour la classification les proportions des classes ont été équilibrées. Les figures (a) (b) montrent les résultats du test en régression et classification.

On voit que la différence entre les deux résultats est petite au début car les deux modèles donnent des résultats très proches au début avant la cinquième itération ce qui correspond au plages 400 à 2000 instance et 40 à 200 features. A partir de là la différence devient plus grande, alors les résultats du RFD-mlm ne changent pas en moyenne c'est les résultats du MLM qui baissent par exemple en classification sur les 5 premières itérations la moyenne des scores du MLM est 0.86 et sur les 5 itérations suivantes la moyenne est 0.81 alors que pour le RFD-MLM on passe de 0.89 à 0.87.

Les résultats montrent clairement que les mesures de distances se comportent mal en grandes dimensions ce qui n'est pas le cas pour les Random Forest dissimilarités.

Chapter 6

Conclusion et perspectives

On a vu que le Minimal Learning Machine présente des résultats assez bons en utilisant la distance euclidienne comme mesure de dissimilarité, néanmoins ces performances n'égalent pas celles des SVM et RF. Mais ça reste une méthode de machine learning qui présente des avantages et des inconvénients et qui peut se révéler être un atout pour résoudre certains problèmes.

La nouvelle version de la méthode qui utilise les RF dissimilarités présente beaucoup d'avantages à savoir que ces performances ne changent pas que ce soit en petites ou grande dimension. Et on a vu qu'elle présente des résultats encourageant si on la compare au Minimal Learning Machine, mais ces résultats demeurent moins bon comparés aux résultats des Random Forest. Mais on a beaucoup d'idées pour améliorer le RFD-MLM, en commençant par optimiser le nombre de point référence qui permettra d'un coté de réduire sa complexité et peut être améliorer ces performances.

On peut aussi optimiser le RF utilisé à l'intérieur en commençant par augmenter le nombre d'estimateurs. On a la possibilité d'utiliser d'autre mesures de distances pour les données en sortie, on pourrait même penser à une façon de calculer les dissimilarités pour les données en sortie en utilisant les Random Forest.

En plus il y a des variantes du Minimal Learning Machine dont on n'a pas parlé dans ce projet, par exemple il y a une version qui s'appelle OS-MLM (Optimal Selected Minimal Learning Machine) qui au lieux de sélectionner aléatoirement les références des données, choisit de sélectionner certaines données en particuliers, ainsi on pourrait éviter de sélectionner des outliers par exemple ou dans le cas de la classification on pourrai veiller à sélectionner la même proportions de points références pour chaque classe. Il y a aussi une variante qui s'appelle NN-MLM(Nearest Neighbors Minimal Learning Machine) qui se base sur une approche au plus proche voisin. Donc voilà il y a pleins d'améliorations possibles qu'on pourrait ajouter au RFD-MLM.

Références

@articleICML, author = "Cortes, C, Mohris, M. Weston, J.", title = "A general regression technique for learning transductions", year = "2005", journal = "Proceedings of the 22Nd International Conference on Machine Learning.", pages = "153–160"

@miscscikit-mlm, author = "Madson Luiz Dantas Dias", year = "2019", title = "scikit-mlm: An implementation of MLM for scikit-learn framework", url = "https://github.com/omadson/scikit-mlm", doi = "10.5281/zenodo.2875802", institution = "Federal University of Ceará, Department of Computer Science"

Liens vers l'article MLM : https://www.sciencedirect.com/science/article/abs/pii/S092523121500302

Liens vers Distance Mesures Data : https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa