

Rapport des TME

Apprentissage et Reconnaissance des Formes

Celia Kherfallah, Stieban Fernandez

Encadrant : Sylvain Lamprier

Avril 2018

Introduction

Ce rapport est une synthèse des six premiers travaux encadrés de l'UE Apprentissage et Reconnaissance des Formes. Ces derniers se portent sur l'étude des différents algorithmes et outils utilisés dans le Machine Learning. Nous aborderons plus particulièrement les TME 4 et 5 dédiés au Perceptron.

TME 1 - Arbres de décision, sélection de modèles

Arbres de décision

Un arbre de décision est un classifieur modélisé par un arbre dont chaque noeud intermédiaire est un test sur une dimension de X et chaque feuille un label de Y . Chaque branche est alors un résultat du test effectué par le noeud dont il découle.

Pour chaque noeud de l'arbre, le choix de l'attribut à tester est déterminé par la répartition de ses valeurs calculée par l'entropie de Shannon. Dans le cas de la classification, on calcule la différence entre l'entropie et l'entropie conditionnelle. Elle permet de connaître la dépendance entre les variables aléatoires associées à un attribut X et un label Y . Une valeur de 0 signifie qu'il y a indépendance entre X et Y tandis qu'une valeur de 1 montre une dépendance, Y est donc déterminée par X .

On cherche ainsi à maximiser l'équiprobabilité (plus grande valeur pour $H(Y)$) et maximiser l'homogénéité (minimiser $H(Y|X)$) à chaque noeud.

Expériences

Les expériences ont été effectuées sur un extrait de la base imdb (Internet Movie Database). Nous procédons par la séparation de la base initiale en deux sous-ensembles, une base d'apprentissage et une base de test. Les figures ci-dessous représentent les performances avec différents partages de notre base initiale.

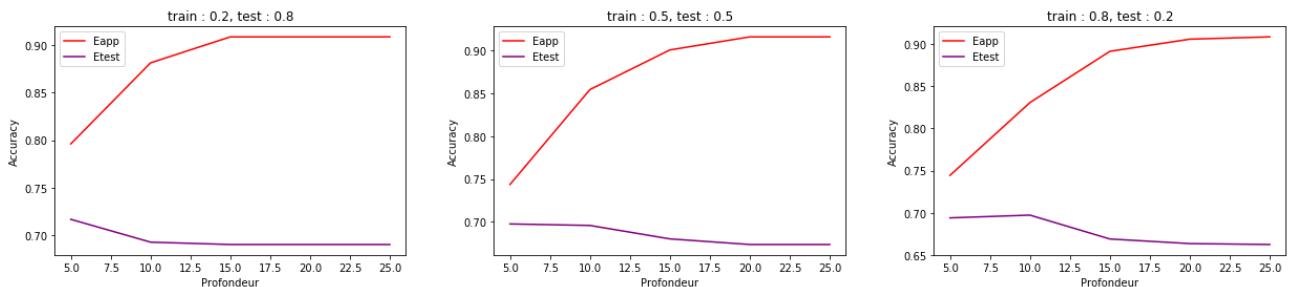


FIGURE 1 – Performances avec différentes séparations

La profondeur de l'arbre détermine le nombre de partitionnements. Nous observons que plus ce nombre est grand, meilleures sont les performances sur la base d'apprentissage (Fig.1). Néanmoins, les résultats du test le sont moins,

nous pouvons parler ici de sur-apprentissage. En effet, le classifieur est trop spécifique à la base d'apprentissage, il n'est donc pas capable de généraliser d'où le mauvais score en test. Notons également qu'une faible profondeur peut-être à l'origine d'un sous-apprentissage.

Par ailleurs, les exemples sont choisis aléatoirement pour construire notre base d'apprentissage qui est une partie de la base initiale. Par conséquent, toute la base n'est pas exploitée pour l'apprentissage et les résultats ne sont pas très fiables.

Validation croisée

Suite à ce problème, une autre approche semble intéressante : la validation croisée. Elle consiste à partitionner la base initiale en N partitions. On effectue N itérations, et à chacune d'elle, on considère une partition comme base de test et le reste comme base d'apprentissage. Les résultats sont davantage fiables mais ils restent identiques aux précédents (Fig.2).

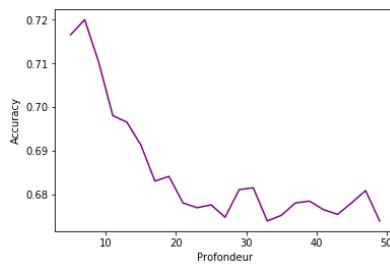


FIGURE 2 – Performances en test avec la validation croisée

TME 2 – Estimation de densité

Dans cette deuxième partie, l'étude se porte sur une base décrivant l'emplacement des points d'intérêt (POI) dans la région parisienne.

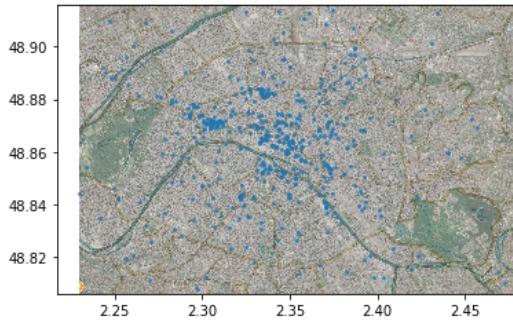


FIGURE 3 – Carte avec le POI : night club

Soit G la variable aléatoire modélisant la présence d'un POI à un emplacement donné sur la carte. L'objectif est d'estimer la densité de G

Méthode des histogrammes

La méthode des histogrammes correspond ici à la discréétisation des coordonnées géographiques et consiste ensuite à compter le nombre de points qui « tombent » dans chaque case.

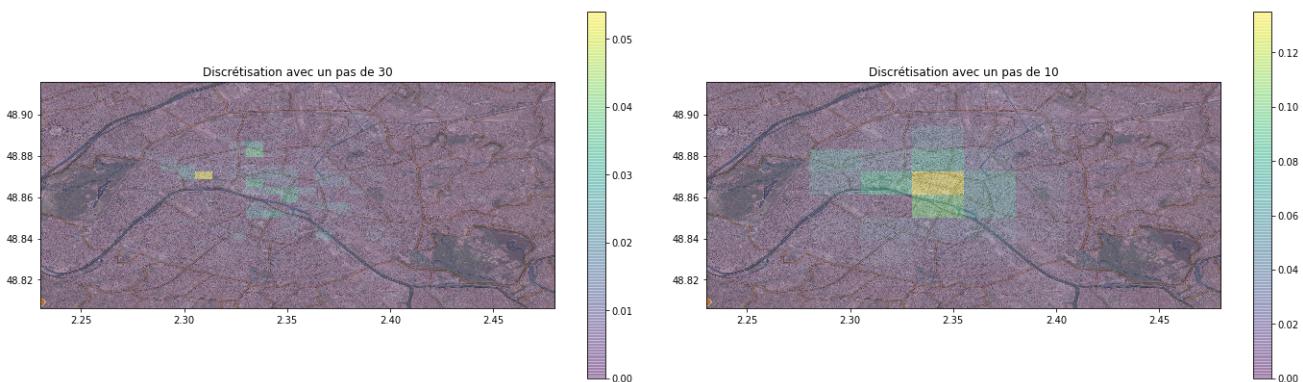


FIGURE 4 – Estimation de densité par la méthode des histogrammes

Lorsque le pas de discréétisation est grand, il y a une grande précision. Néanmoins, cela engendre des « cassures » entre les cases où le nombre d'observations varie énormément. Il est donc difficile d'obtenir une estimation globale sur une zone. Au contraire, en réduisant le pas, on obtient des estimations trop généralisées.

Estimation non paramétrique par noyaux

Une nouvelle approche consiste à prendre en considération les cases voisines. Il existe différentes manières d'y procéder et ce, par l'utilisation d'une fenêtre ou d'un noyau.

Fenêtre de Parzen

Le principe est de considérer une fenêtre autour d'un point et prendre en compte tous les points à l'intérieur de celle-ci. La taille de la fenêtre est déterminée par un paramètre hn .

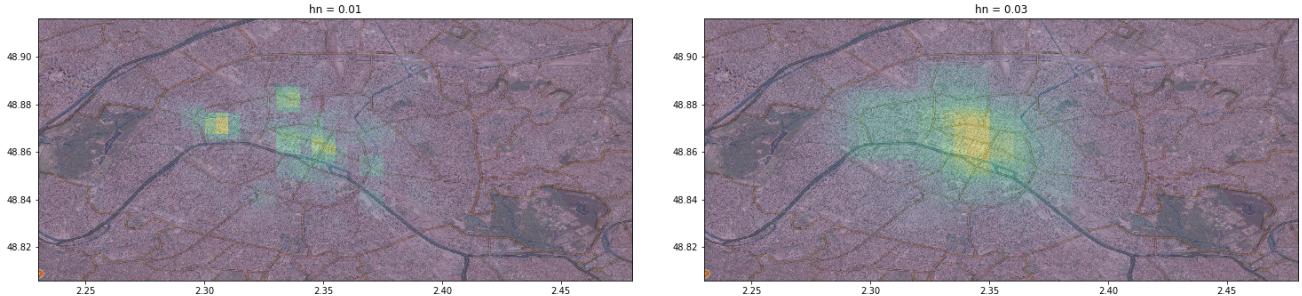


FIGURE 5 – Estimation de densité avec la fenêtre de Parzen

Noyaux

On procède cette fois-ci à l'utilisation d'un noyau gaussien. L'importance du paramètre hn dépend grandement de la taille des données. Si cette dernière est trop petite ou tend vers l'infini, le paramètre n'a pas vraiment d'impact. Sinon, il permet de plus ou moins nous renseigner sur la répartition des points.

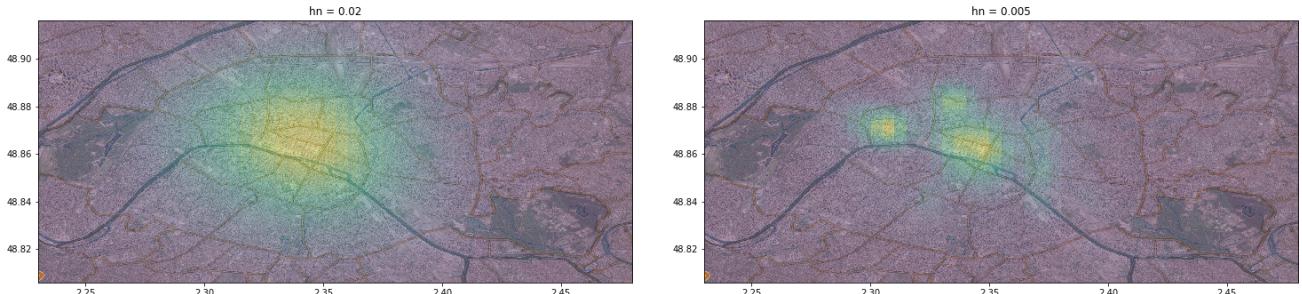


FIGURE 6 – Estimation de densité avec un noyau gaussien

De la densité à la classification

L'estimation de densité peut servir à la classification, notamment dans le cas de l'estimateur Watson-Nadaraya qui utilise la fenêtre de Parzen ou un noyau. On remarque que la fenêtre de Parzen est problématique lorsqu'il n'y a pas de points dans la zone considérée.

L'algorithme du KNN est quant à lui, une autre méthode de classification qui consiste à prendre en compte les K plus proches voisins du point considéré et prédire par vote majoritaire. Néanmoins, celui-ci se base uniquement sur une mesure de distance et prend un nombre de points fixe. Une valeur de K trop petite conduit au sur-apprentissage tandis qu'un K trop grand, on prend la classe majoritaire.

Dans les deux cas, le paramétrage définit l'efficacité du classifieur et doit être testé à la main.

TME 3 - Descente de gradient

L'optimisation d'un classifieur est effectuée par une fonction de coût. Le but est de trouver le vecteur de poids optimal qui la minimise. On peut procéder par solution analytique mais aussi par la descente de gradient. Ce dernier est un algorithme qui permet d'atteindre un minimum local d'une fonction.

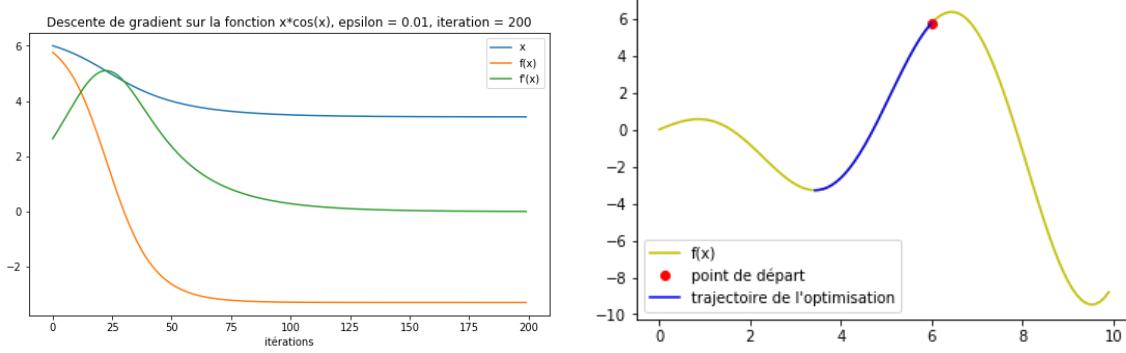


FIGURE 7 – Descente de gradient sur $xcos(x)$ avec $\epsilon = 0.01$

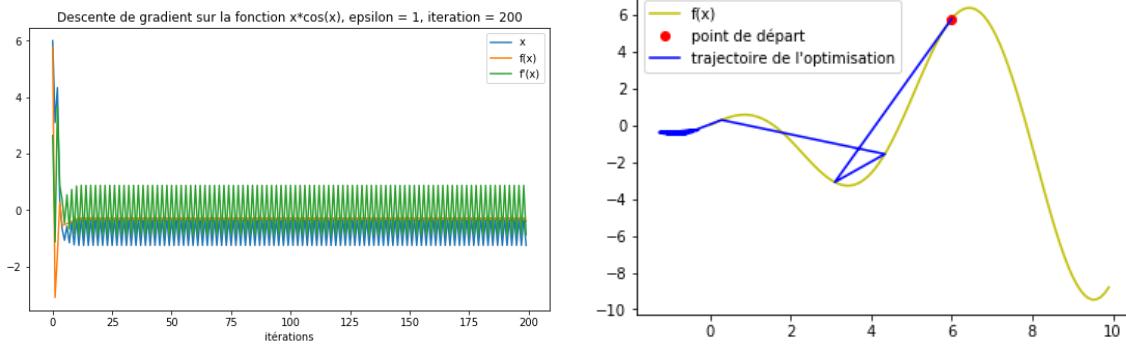


FIGURE 8 – Descente de gradient sur $xcos(x)$ avec $\epsilon = 1$

On remarque ici qu'un pas de gradient peu élevé nécessite beaucoup plus d'itérations pour arriver au minimum alors qu'un pas trop grand provoque des oscillations et le minimum peut ne pas être atteint. Une des solutions utilisées est de réduire le pas du gradient progressivement.

L'algorithme de la descente de gradient est préférable à la solution analytique car il est moins coûteux. Il est notamment utilisé dans la régression logistique.

TME 4-5 - Perceptron

Le perceptron est un algorithme d'apprentissage supervisé de classification binaire qui utilise la descente de gradient. Notre étude se porte sur deux fonctions de coût : les moindres carrés (MSE) et hinge.

Données linéairement séparables

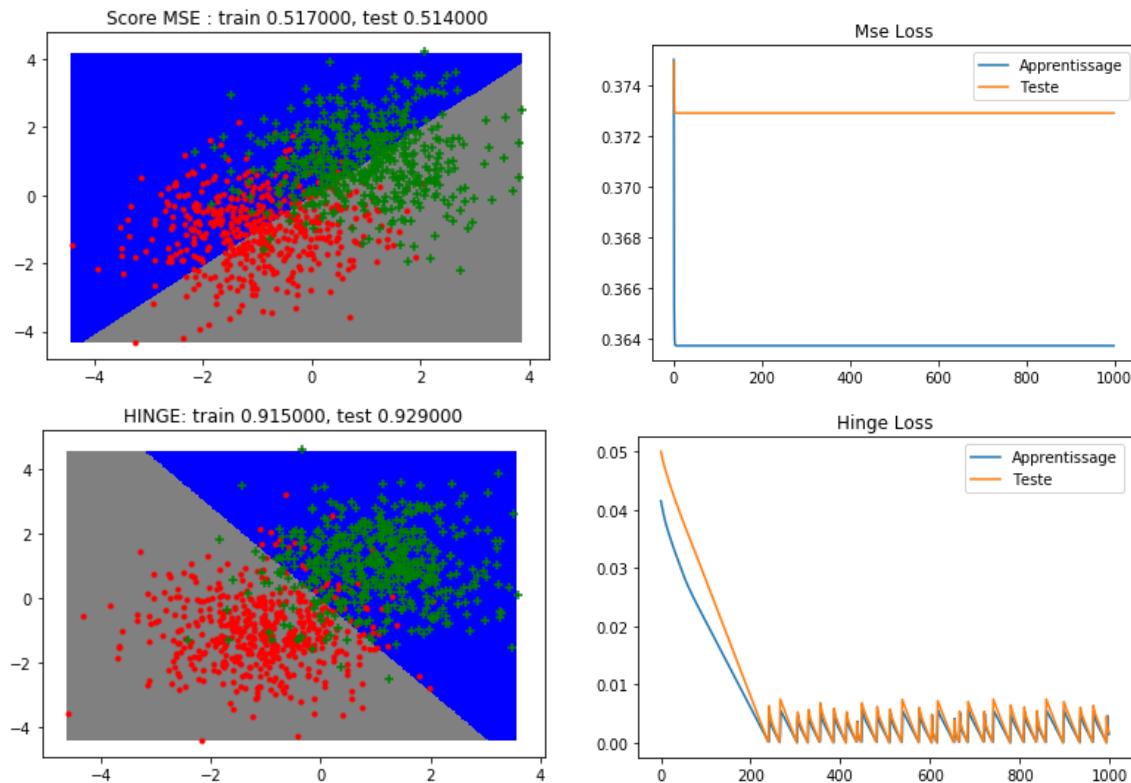


FIGURE 9 – Frontières de décision et évolution de l'erreur avec MSE (gauche) et hinge (droite)

On remarque qu'avec MSE, la frontière de décision est erronée contrairement à celle obtenue avec hinge. Cela s'explique par le fait que MSE est principalement utilisé dans le cadre de la régression linéaire, qui consiste dans notre cas à trouver la meilleure relation affine limitant l'erreur quadratique moyenne. MSE pénalise donc aussi bien les points bien classés que ceux mal classés à cause de leur distance par rapport à la frontière. Par conséquent, cette fonction n'est pas adaptée à la classification.

Contrairement au MSE, la fonction hinge ne pénalise que les points qui sont du mauvais côté de la frontière. D'où nous obtenons de meilleurs résultats avec hinge.

Par ailleurs, les oscillations dans la 4e figure sont dues au bruit car l'algorithme rencontre toujours un point mal classé et réajuste le vecteur de poids continuellement.

Données non linéairement séparables

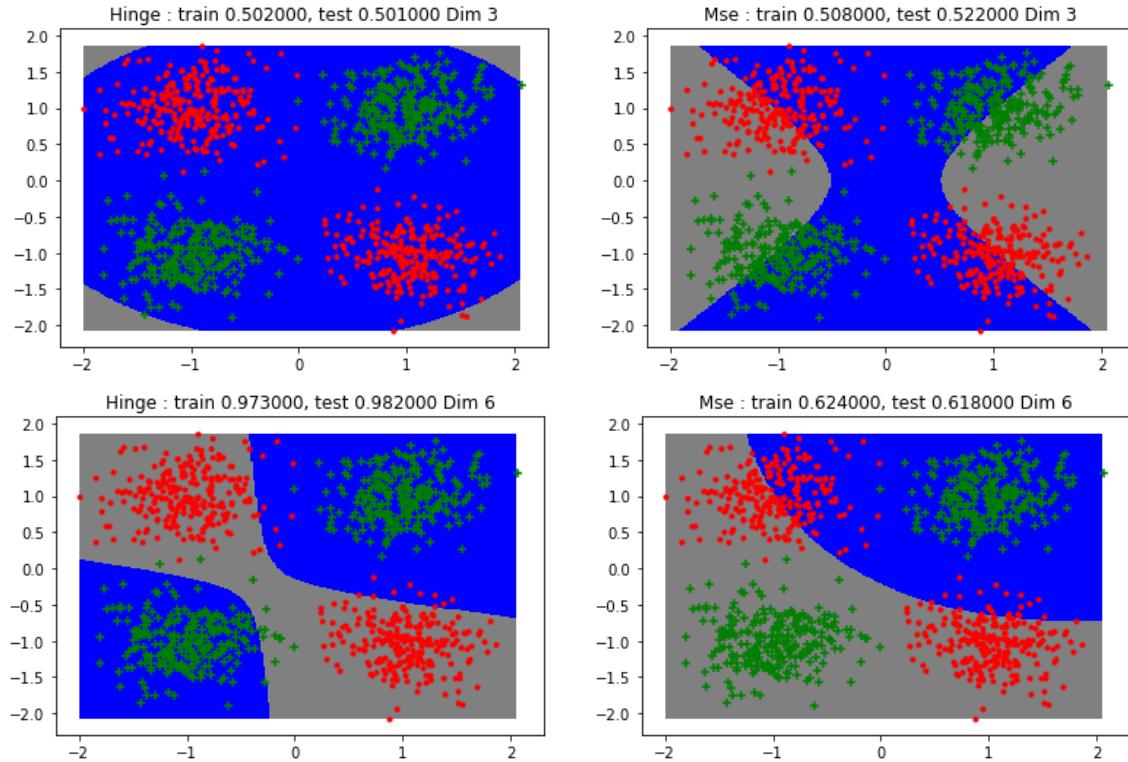


FIGURE 10 – Frontières de décision avec projection polynomiale

Nous constatons que plus le nombre de dimensions pour la projection augmente, plus les scores sont bons. Néanmoins, le risque de sur-apprentissage s'accroît.

Jeu de données USPS

Les expériences se portent sur un jeu de données où chaque exemple correspond à un vecteur de 256 pixels représentant un chiffre. Nous allons traiter deux approches différentes : One vs One et One vs All.

L'approche One vs One consiste à choisir deux classes différentes i et j . On considère alors la classe i comme le label +1 et la classe j comme -1. On fait en sorte que les exemples de l'ensemble de chaque classe soient mélangés et que la distribution soit équilibrée (ie, autant d'exemples positifs que d'exemples négatifs).

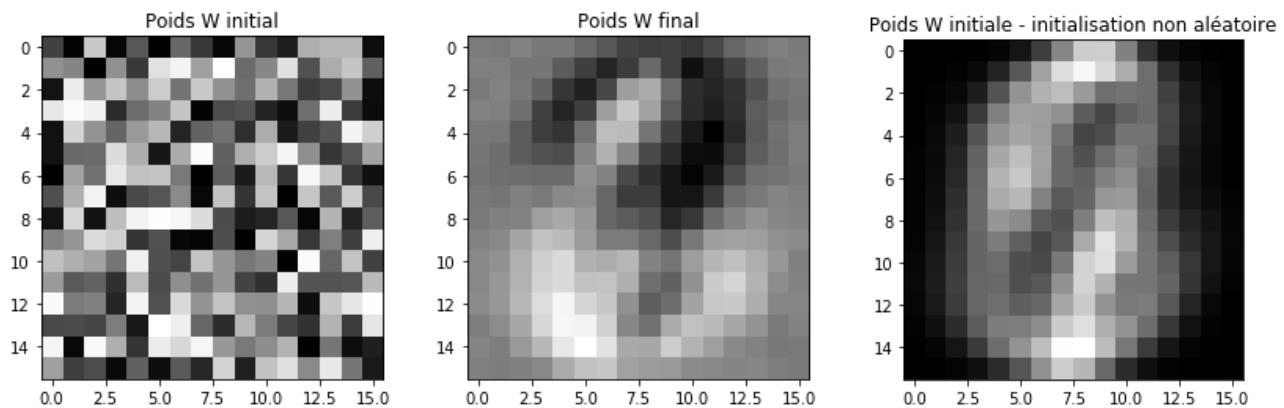


FIGURE 11 – Score : train 1.0, test : 0.997

En choisissant les classes 6 et 9, on constate que le vecteur de poids final est plutôt satisfaisant. La partie basse gauche est plus claire pour former la boucle du 6 et la partie haute droite est plus foncée pour former la boucle du 9. Nous avons testé une initialisation non aléatoire du vecteur qui consiste à faire la moyenne des exemples présents dans notre base contenant 6 et 9 pour permettre à notre perceptron d'avoir un a priori sur les pixels pertinents (d'où la couleur blanche au milieu, pixels communs aux deux chiffres) et une couleur noire sur les côtés car aucun autre exemple présent dans la base ne lui a permis d'interpréter ces pixels.

L'approche One vs All considère cette fois-ci une classe comme positive et toutes les autres comme la classe négative. On procède de la même manière pour le balancement et le tirage aléatoire des exemples.

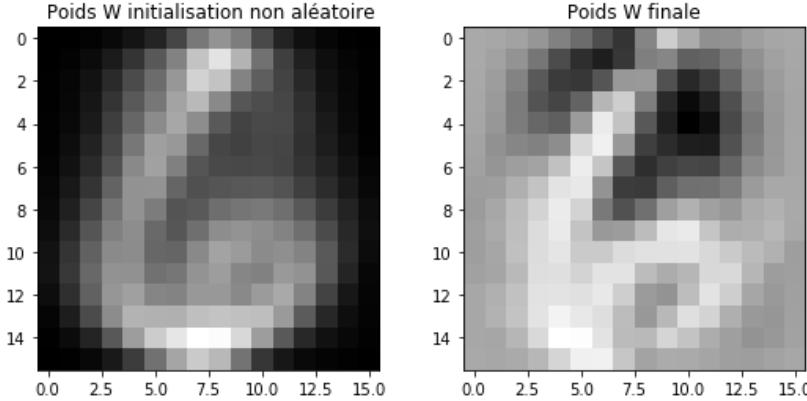


FIGURE 12 – Score : train 0.949, test 0.941

One vs One obtient de meilleurs résultats et converge plus rapidement, ce qui est normal puisqu'il ne compare que deux classes.

Notre modèle sur-apprend-il ?

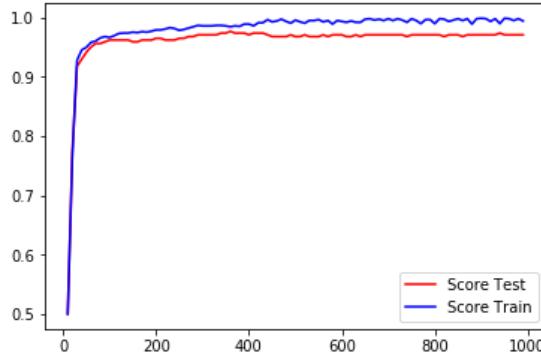


FIGURE 13 – Performances sur l'ensemble d'apprentissage et de test

La courbe des performances en test approche la courbe en apprentissage, il ne s'agit donc pas de surapprentissage.

Batch vs stochastique

Nous avons utilisé jusqu'ici un modèle batch qui consiste à prendre en compte toute la base d'exemples à chaque itération. Nous allons le comparer à la version stochastique.

La version batch converge plus rapidement mais présente un coût élevé tandis que la version stochastique est nettement moins coûteuse car elle ne prend en compte qu'un seul exemple à chaque itération, ce qui réduit le temps de calcul. Néanmoins, elle est moins précise car étant donné le tirage aléatoire des exemples, ces derniers ne le dirigent pas forcément vers la même direction on peut observer ce phénomène à travers les oscillations, contrairement à la version batch qui est lisse avec quelques itérations.

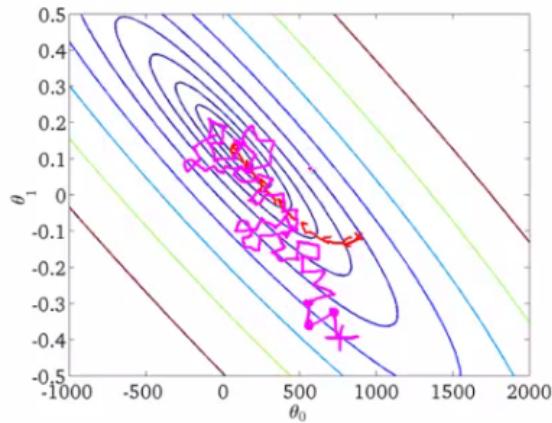


FIGURE 14 – Descente de gradient avec version batch (rouge) et stochastique (rose)

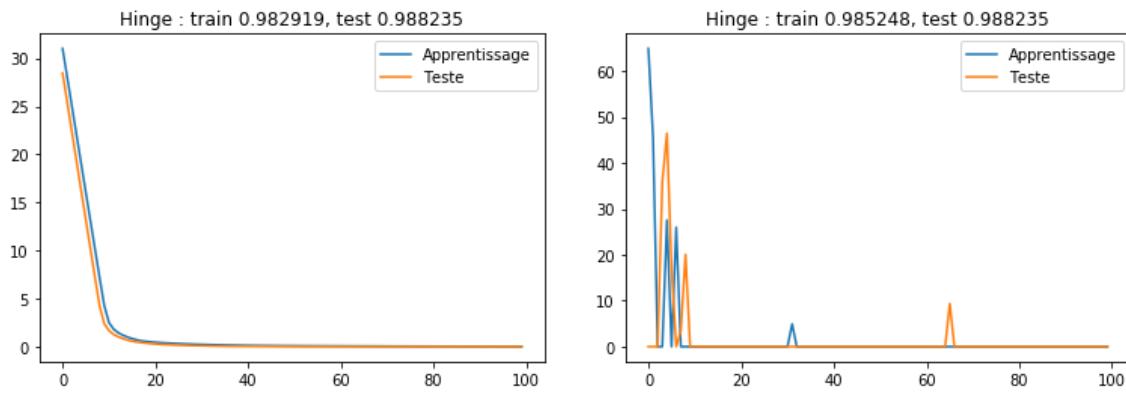


FIGURE 15 – Évolution de l'erreur avec batch (gauche) et stochastique (droite)

Bagging

Nous avons testé la méthode du Bagging avec 20 perceptrons afin d'avoir des résultats plus fiables. Nous obtenons alors un score moyen de 0.998 avec un temps de calcul raisonnable.

TME 6 - SVM, Noyaux

Nous aborderons dans cette partie la famille des SVM, acronyme de Support Vector Machines. Un algorithme d'apprentissage automatique, et qui est très efficace dans les problèmes de classification. Il appartient à la catégorie des classificateurs linéaires. Nous avons remarqué précédemment que la frontière de décision du perceptron n'est pas unique sachant qu'elle dépend du dernier ajustement et de plus, elle n'est pas optimale. Nous voulons une frontière de décision aussi loin que possible des exemples de chacune des classes afin d'avoir une meilleure capacité de généralisation. Ainsi, SVM permet de trouver la distance minimale entre les vecteurs d'entraînement (appelés vecteurs supports) et l'hyperplan.

On ajoute la notion de variable reçoit (slack variable) qui autorise la tolérance au bruit et une constante C qui la règle.

Nous utiliserons notamment des projections particulièrement dans le cas des données non linéairement séparables. Les noyaux utilisés sont : linéaire, polynomial, RBF (gaussien) et sigmoïde.

Nous voulons mettre en évidence l'influence de C sur la frontière de décision et les marges.

Données linéairement séparables

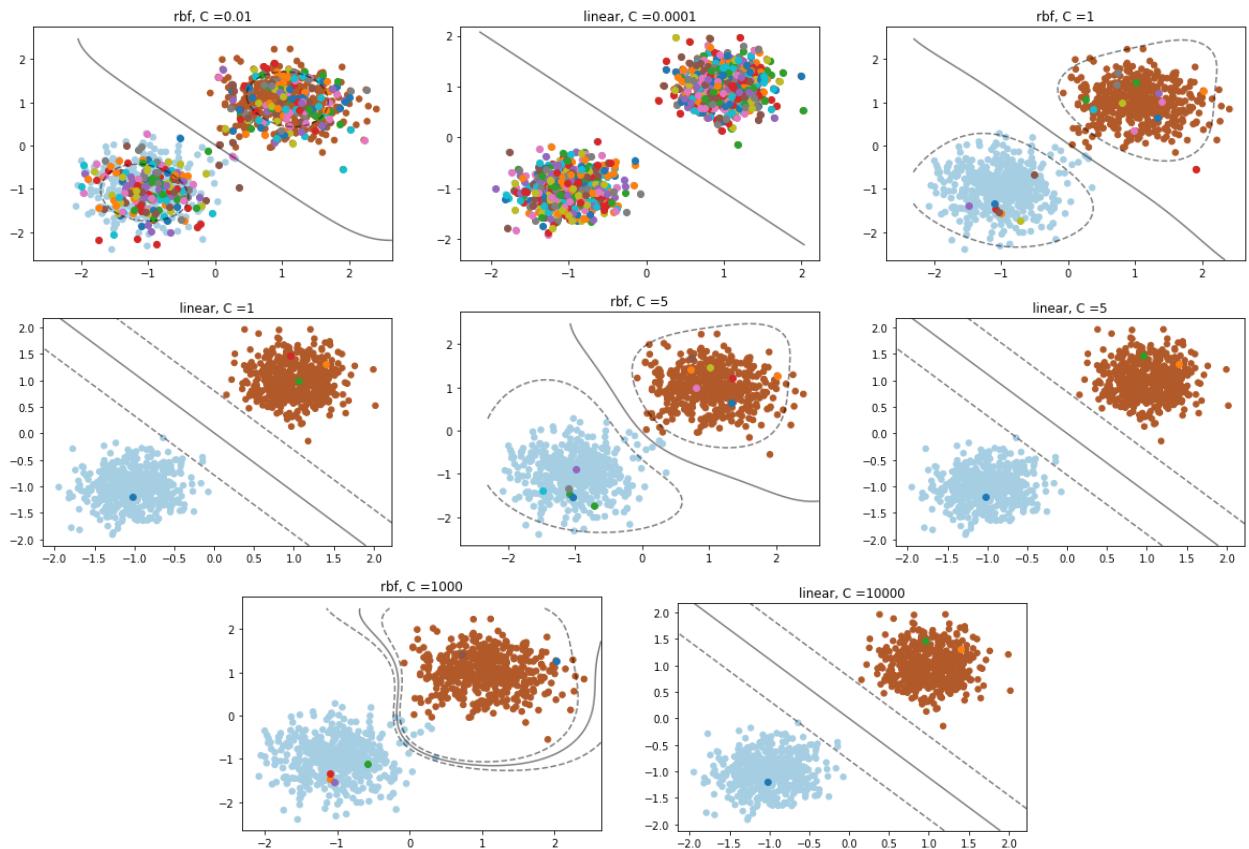


FIGURE 16 – Frontière de décision et marges avec noyau gaussien (RBF) et linéaire. Classe +1 (marrons), classe -1 (bleus), vecteurs supports (multicolores)

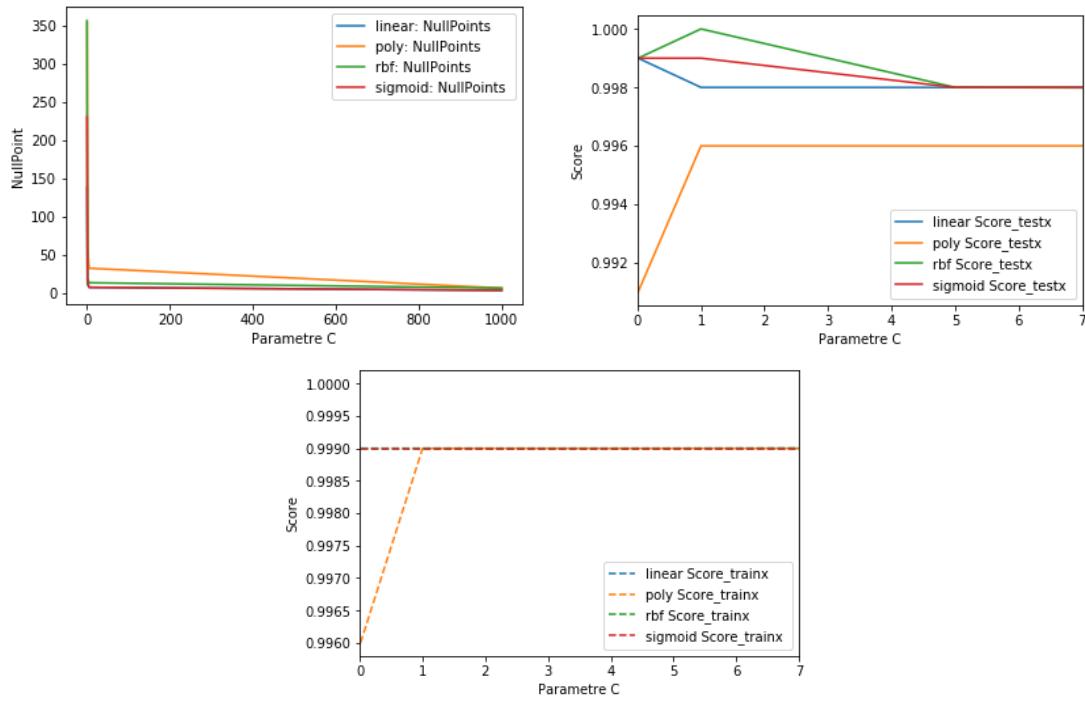


FIGURE 17 – Score en fonction du paramètre C et différentes projections

Données non linéairement séparables

Voir frontières de décision et marges en annexe (Fig.20).

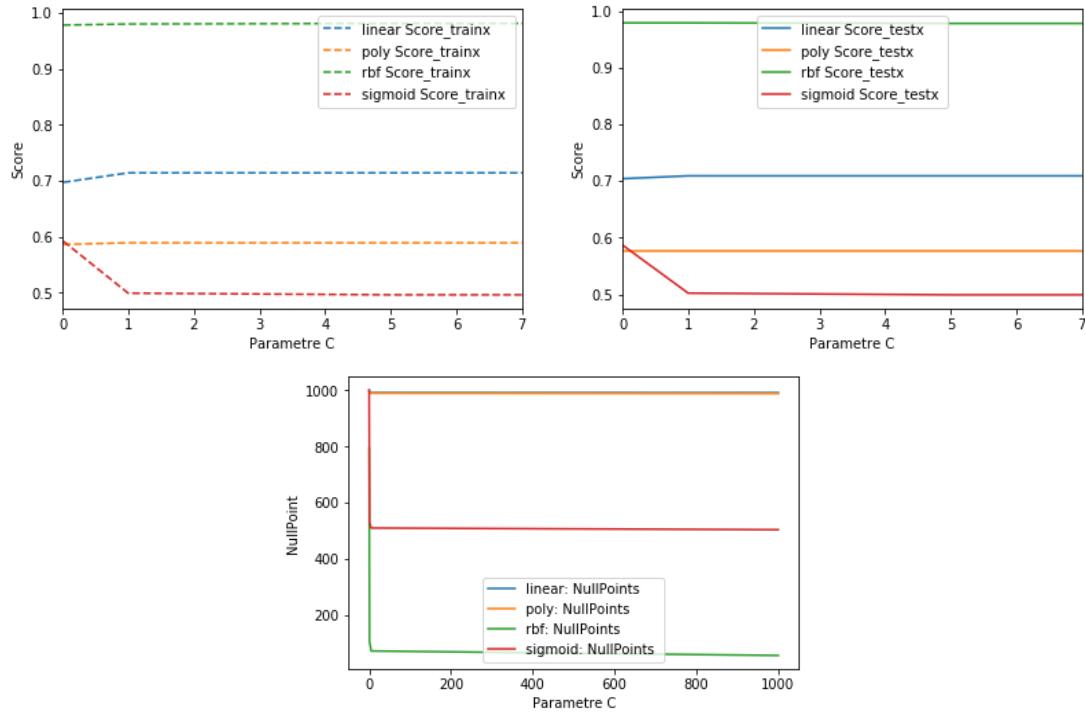


FIGURE 18 – Score en fonction du paramètre C et différentes projections

Un C faible rend la surface de décision lisse et on obtient plus de points avec un coefficient non nul. Alors qu'un C élevé vise à classer correctement tous les exemples d'entraînement en donnant au modèle la liberté de sélectionner plus d'échantillons comme vecteurs support.

Le noyau gaussien possède la meilleure performance d'après les expériences. Pour le noyau sigmoïde, on observe que plus C augmente, plus le score diminue (Fig.18).

One vs One et One vs All

La stratégie one-vs-all consiste à entraîner un seul classificateur par classe, avec les échantillons de cette classe comme échantillons positifs et tous les autres échantillons comme négatifs, avec rebalancer des classes comme expliqué plus haut. La prise de décision signifie appliquer tous les classificateurs à un échantillon invisible x et prédire l'étiquette k pour laquelle le classificateur correspondant rapporte le score de confiance le plus élevé : $y = \text{argmax } f_k(x)$ avec $k \in \{1 \dots K\}$. Cette stratégie est très populaire et donne d'assez bons résultats en étant peu coûteux en temps de calcul.

La stratégie One vs One consiste à former $\frac{N(N-1)}{2}$ classificateurs binaires. Chacun reçoit les échantillons d'une paire de classes de l'ensemble d'apprentissage original et s'entraîne sur celle-ci. Lors de la prédiction, on procède par vote majoritaire. La classe ayant été prédite le plus grand nombre de fois est choisie.

	One vs one	One vs all
linéaire :	0.92326856004	0.884404583956
RBF (gaussien) :	0.941205779771	0.925759840558
polynomial :	0.930244145491	0.922770303936

TABLE 1 – Scores

Contrairement à One vs All, One vs one est beaucoup moins sensible aux problèmes avec des ensembles de données déséquilibrés. Cependant, il est beaucoup plus coûteux en temps de calcul en dépit de ses performances qui meilleures.

Kernel String

Nous avons testé le String Kernel sur les textes suivants issus d'époques différentes

Ainsworth, William Harrison, 1805-1882 Ainsworth, William Harrison, 1805-1882 tome 2 Ainsworth, William Harrison, 1805-1882 tome 3 Molière , L'avare, 1668 Molière, Don Juan, 1665 Molière, Médecin malgré lui, 1666

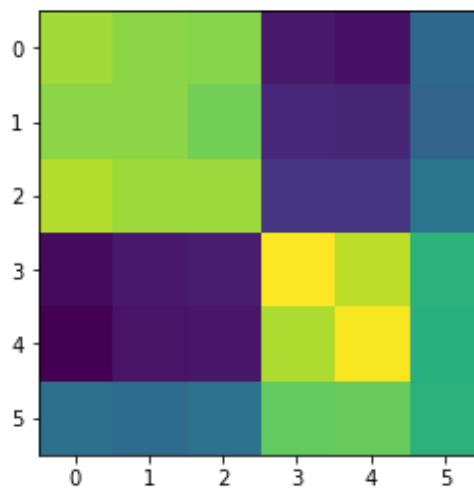


FIGURE 19 – Matrice de confusion sur les 6 textes

Après filtrage des textes et suppression des mots vides, on constate que les deux auteurs ont un style littéraire différent.

Annexe

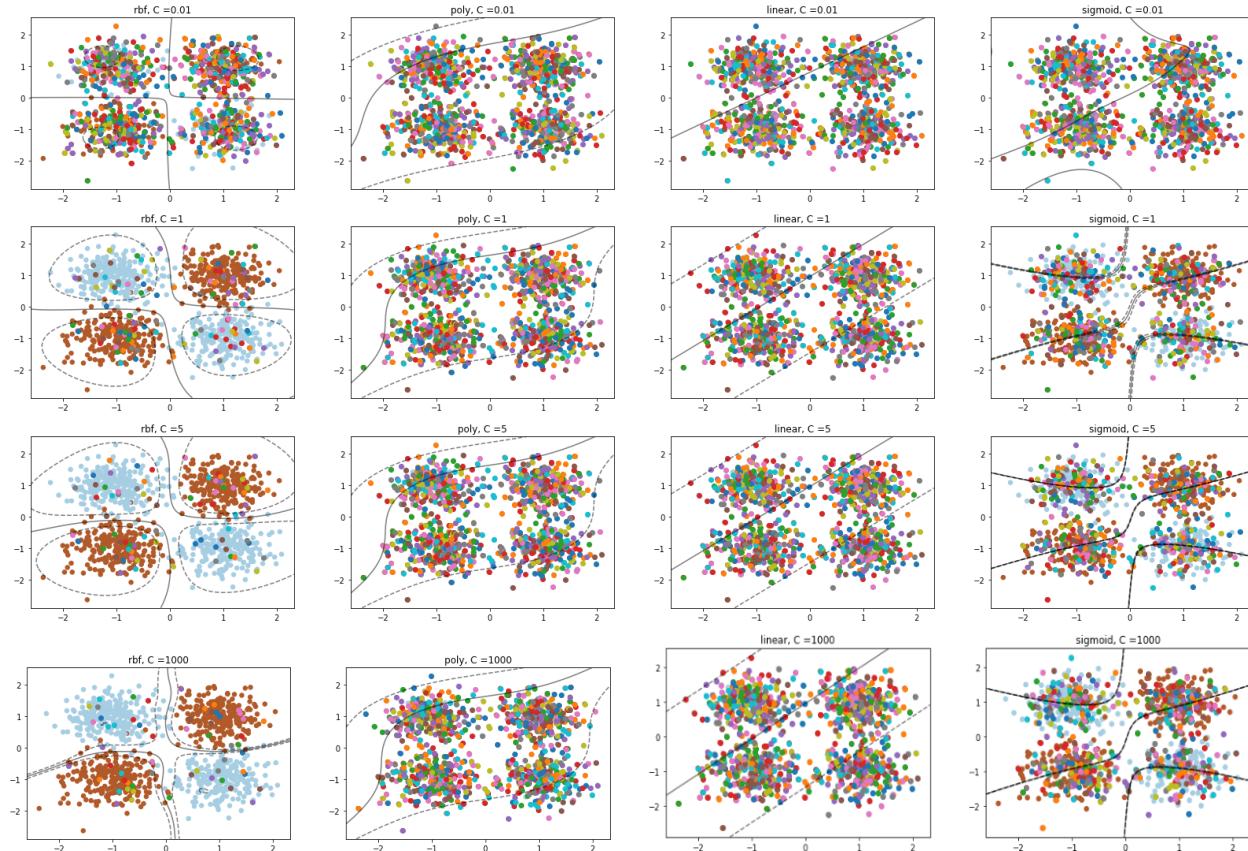


FIGURE 20 – Frontière de décision et marges avec noyau gaussien, polynomial, linéaire et sigmoïde