Sorbonne université

Rapport de ARF

TME 1 - 6

Samu Tamminen

Année 2017-2018

Table des matières

1 TME 1 : Arbres de décision, sélection de modèles

1.1 Q1.3 Interpretation de l'entropie

Calculer également la différence entre l'entropie et l'entropie conditionnelle pour chaque attribut. A quoi correspond une valeur de 0 ? une valeur de 1 ? Quel est le meilleur attribut pour la première partition ?
Quand la différence entre l'entropie et l'entropie conditionnelle est 0 quand la partition n'est pas du tout bon et un, quand la partition est le meilleur possible.

1.2 Q1.4 - 6 La profondeur d'arbre

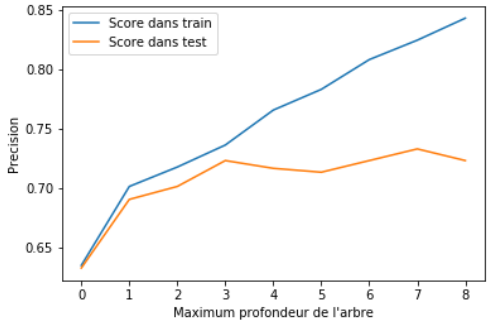
En augmentant la profondeur maximale d'arbre, on obtient meilleur score. Mais ces scores ne sont pas un indicateur fiable, si nous ne faisons pas différence entre les données training et test.

1.3 Q1.7 - Sur et sous apprentissage

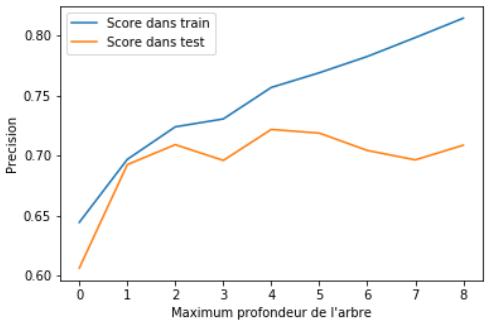
Le partition 0.8 / 0.2 dans Figure 1 (c) semble d'etre le plus robust pour test et training. La difference difference entre le score dans aprentissage et test est le plus petit parmis (a),(b) et (c).

Quand il y a que peu d'examples d'apprentissage, l'erreur est plus éleve dans le test. Par contre quand on a beacoup des d'examples d'aprentissage, l'erreur dans le test est moins d'élevé.

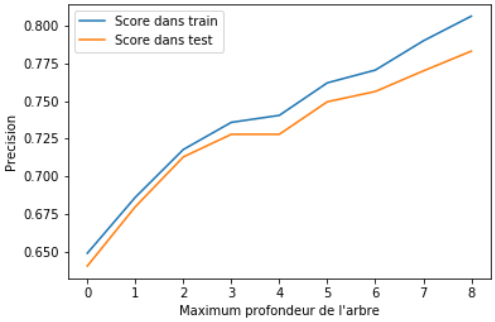
FIGURE 1 – Score d'arbre de decision en faisant varier le profondeur



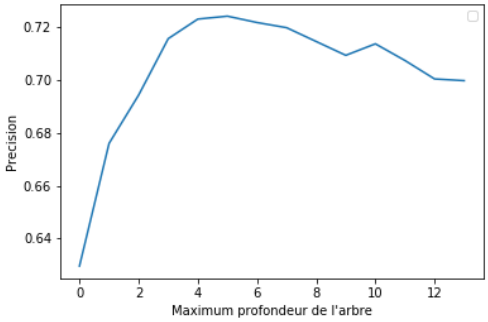
(a) Donnée aprentissage et test : 0.2 / 0.8



(b) Donnée aprentissage et test : 0.5 / 0.5



(c) Donnée aprentissage et test : 0.8 / 0.2



(d) Validation croisee

Avec validation croisée on obtient un score plus fiable et stable. Dans la figure Figure 1 (d) nous pouvons constater, comment le score augmente au début avec la profondeur maximale, mais à la fin il tombe. Ça c'est à cause de sous et sur apprentissage. Au début, on généralise beaucoup avec un arbre peu profond (sous apprentissage). Quand la profondeur se monte sur cinq, nous commençons à apprendre les données d'apprentissage à la façon trop détaillée et le score sur le test se tombe (sur aprentissage). La profondeur maximale 4 semble optimal.

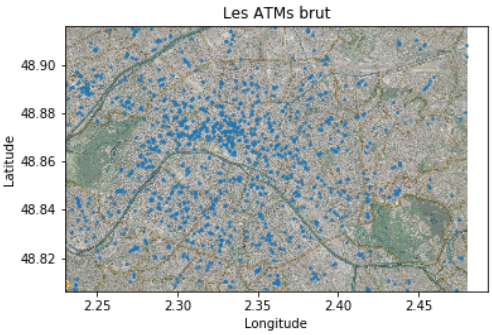
2 TME 2 : Estimation de densité

Nous sommes intéressés de densité des services différents de Paris.

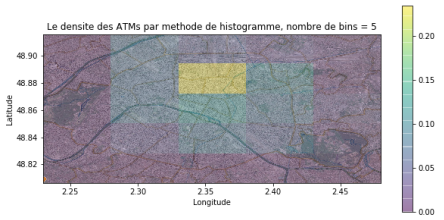
2.1 Méthode des histogrammes

La méthode la plus simple, méthode des histogrammes, il s'agit de partager la valeur continue (latitude et longitude) à les intervalles fixés et compter, combien d'occurrences chacun contient.

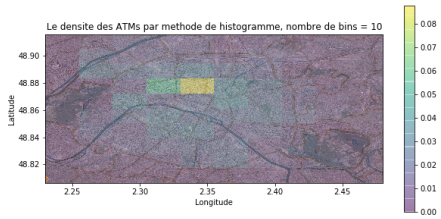
FIGURE 2 – Densite par méthode des histogrammes



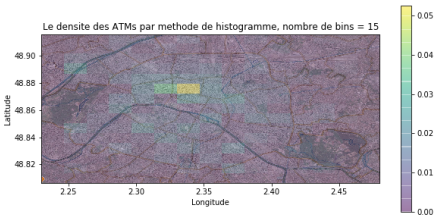
(a) Les ATMs à Paris



(b) Histogramme avec 5 intervalles



(c) Histogramme avec 10 intervalles



(d) Histogramme avec 15 intervalles

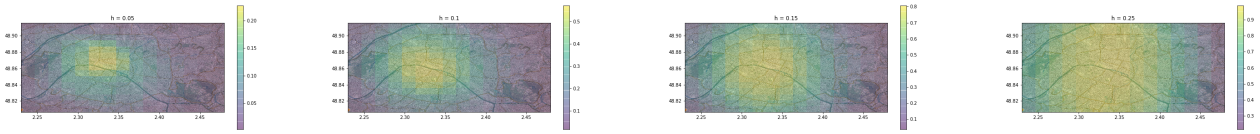
On peut voir bien avec chaque discretization de Figure 2, que la densité maximum des ATMs est dans 9iéme arrondissement. Par contre, la densite n'est pas très visible dans les endroits où la densité est moins forte (par exemple 13iéme arrondissement). Surtout avec 15 étapes, nous ne voyons pas très bien.

2.2 Méthode à noyaux

Comme la méthode de histogramme ne permet pas de visualiser les atms bien avec 15 étapes, je vais fixer le variable étapes à 15 pour l'estimation par noyau. L'estimation par noyau a une variable

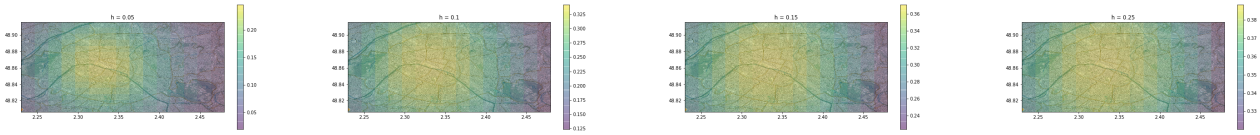
h qui est taille de lissage. On va varier d'abord h.

FIGURE 3 – Parzen comme fonction de Noyau



Dans Figure 3, si h >= 0.25, l'estimation est "oversmoothed". h = 0.15 l'air le plus optimal, car il prend en compte les points partout à Paris, mais la densité est également visible.

FIGURE 4 – Gauss comme fonction de Noyau



Avec le gaussien (Figure 4), le meilleur paramètre de lissage est peut-être 0.05.

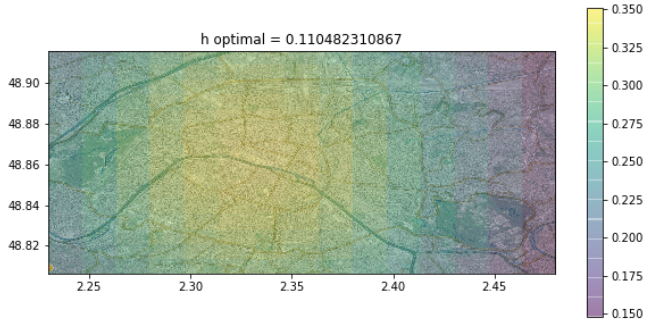
2.2.1 Comment choisir de manière automatique le meilleur taille de lissage h ?

Selon [A rule-of-thumb bandwidth estimator, 2018], nous pouvons estimer l'optimal taille de lissage h pour noyau de Gauss. L'estimation est definit dans Équation 1.

$$h = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{\frac{1}{5}} \approx 1.06\hat{\sigma}n^{-1/5} \tag{1}$$

Nous obtenons h=0.110482310867 pour les données des ATMs à Paris, visualisé dans Figure 5.

FIGURE 5 – Densité avec la méthode de noyaux (Gauss) et lissage optimale



L'optimal "rule-of-thumb" estimateur minimise l'erreur de moindre carrée intégré (Mean integrated squared error). Il permet de montrer le "vraie" distribution, que les POIs vont suivre. J'ai visualisé également les autres POIs avec le noyaux de Gauss et l'optimal h. J'ai constaté que les visualisations ne sont pas très différentes. Les différentes POIs sont distribués à Paris plus ou moins à la même façon : en centre-ville la densité est grande, d'ailleurs moins. Donc je pense que ça sera difficile de classifier les POIs seulement par rapport ses localisations.

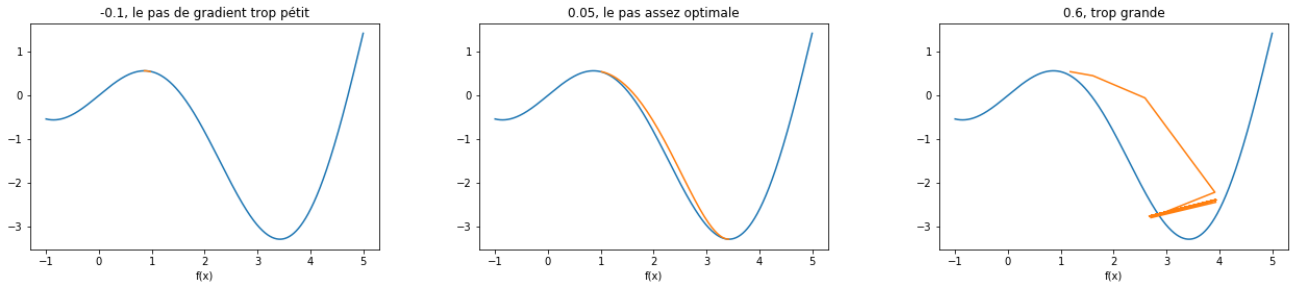
3 TME 3 : Descente de gradient

Cette semaine nous avons implémente la descente de gradient et appliqué ça à régression logistique.

3.1 Optimisation de fonctions

Dans le Figure 6 il y a des visualisations de fonctionnement de descente de gradient. Nous optimisons le fonction f donnée, en cherchent son minimum (locale, car f(x) = x cos(x) n'est pas convexe). L'axe y correspond la valeur de f(x), donc le minimum local est au fond de courbe et l'axe x correspond les valeurs de x.

FIGURE 6 – La descente de gradiente de f(x) = x cos(x)



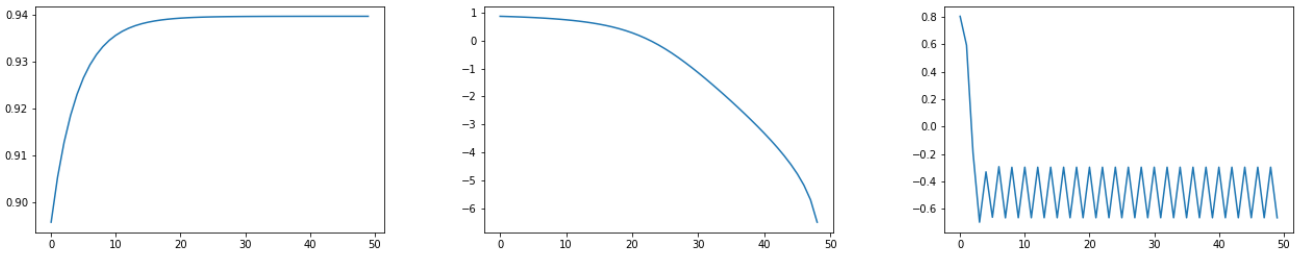
(a) (b) (c)

Si le bas de descente de gradient est trop petit ou grande, il y a une risque que, à la fin des itérations d'algorithme, nous n'arrivons pas à minimum de fonction. (Figure 6 a et c).

Dans le Figure 7 j'ai plotté le direction de descente (defini dans Équation 2) de chaque epsilon de figure precedent. On peut deduire, que le premier ne descend pas, car son epsilon est negative (direction faux). Troisiéme descend jusqu'a certain moment, mais apres il commence bugger (epsilon tros grand). Au milieu, le descente est par contre assez direct vers le minimum.

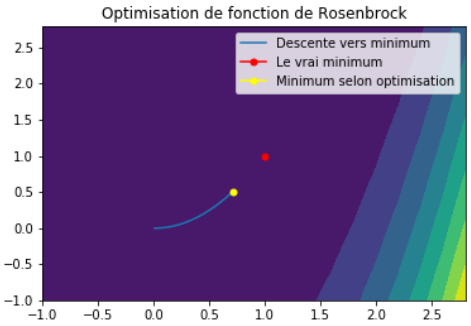
$$\left(t, \log \left\|x^t - x^*\right\|\right) \tag{2}$$

FIGURE 7 – Direction de descente (Équation 2)



Mon implémentation de descente de gradient ne fonctionne pas parfaitement. Dans le Figure 8 il y a une optimisation de fonction de Rosenbrock, où on peut voir que l'algorithme ne converge pas à vrai minimum de fonction. Cette fonction est connue pour ça que c'est difficile de converger à son minimum.

FIGURE 8 – C'est difficile de converger à minimum de fonction de Rosenbrock



3.2 Regression logistique

J'ai implémente[1] régression logistique, qui utilise la descente de gradient pour optimiser log vraisemblance.

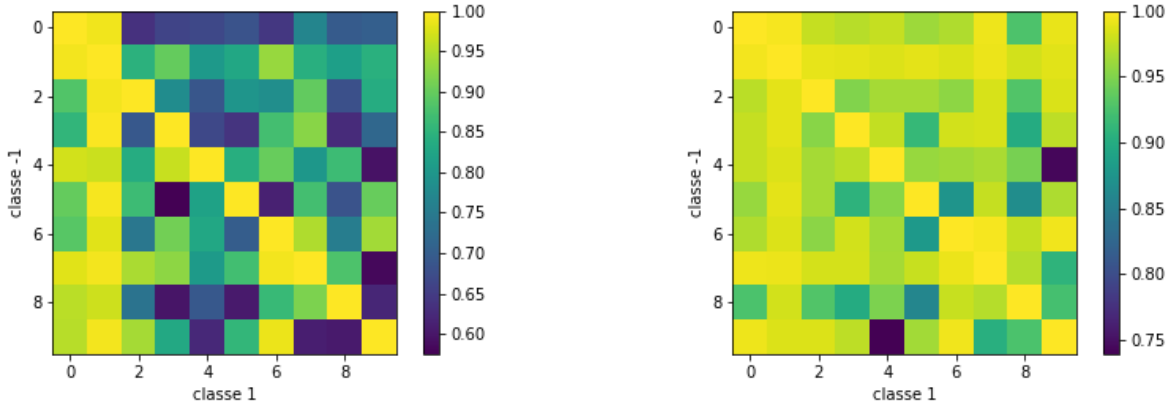
Je vais comparer classification des nombres différents (les données USPS). Pour le pas de descente de gradient j'ai fixé eps = 0.025 qui fonctionne bien avec différents nombres.

Pour chaque combination

$$(i, j) \in \{0...9\} \times \{0...9\}$$

nous calculons dans Figure 9 les scores avec cinq iteration de validation croisée.

FIGURE 9 – Les nombres un contre un



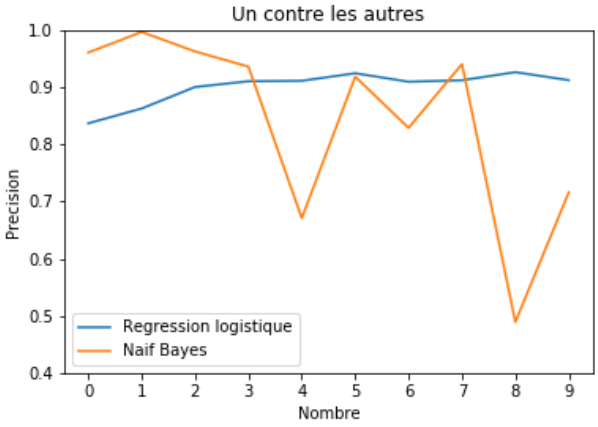
(a) Regression logistique

(b) Naif Bayes

Nous pouvons constater dans Figure 9 a), avec regression logistique, par exemple que le nombre 5 étant classe -1 et 3 étant classe 1 on obtiens score mal. Également le nombre zéro est difficile de classifier, comme on peut voir dans Figure 9 a) sur le premier ligne. La difficulté avec zéro est probablement à cause de fait, qu'on a beaucoup plus de zeros dans les données que les autres et la forme de zero est distribué largement et est donc facile de mélanger avec les autres. Naif bayes semble fonctionner mieux sur classification un contre un (Figure 9 a)). Seulement le score entre 4 et 9 est moins elevée.

1. Code dans TME3/Logistic_regression.py

FIGURE 10 – Les nombres, un contre les autres

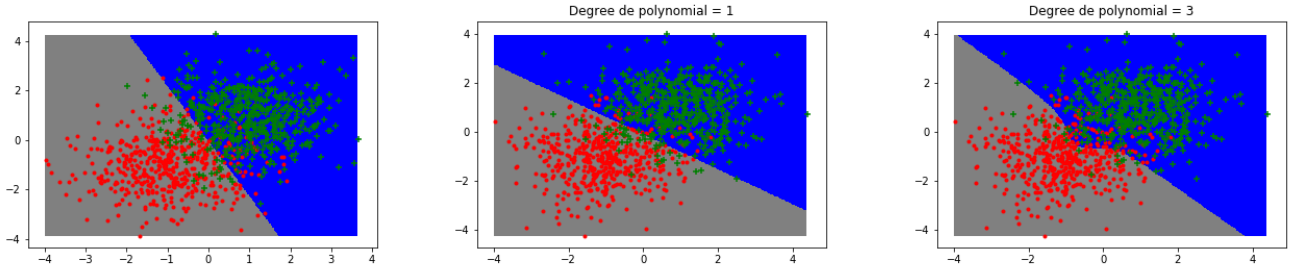


Dans Figure 10 il y a comparison entre les scores de regression logistique et naif bayes [2]. Nous calculons le precision (avec validation croisée de cinq scores) de chaque nombre comparée à tous les autres nombres. Ça va dire, que nous mettons le classe -1 pour cet nombre et classe 1 pour tous les autres. En moyenne, le regression logistique est plus stable et robust, mais parfois on obtient meilleur resultat avec classificateur de naif bayes (comme pour nombre 1 : avec NB c'est presque parfait le classification).

4 TME 4 et 5 : Perceptron

Cette semaine nous avons implémente perceptron [3] et nous avons réalisé plusieurs expériences. Dans Figure 11 a) nous avons visualisation de données artificiel et la frontiére de decision de perceptron. Perceptron cherche le separateur lineaire, (corbe au milieu) qui partage les données.

FIGURE 11 – Perceptron avec les données artifiel en 2 dimension



(a) Sans biais (b) Avec biais (c) Avec polynomial de 3iéme degree

4.1 Prise en compte de biais

Dans Figure 11 b) j'ai utilisé également le biais, qui permet de ne pas passer par origo. Dans le formulation d'algorithme de perceptron, nous utilisons fonction f (Équation 3) en aprentissage (dans

2. Code dans TME3/NaiveBayes_classifier.py
3. Code dans TME4/tme4_etu.py

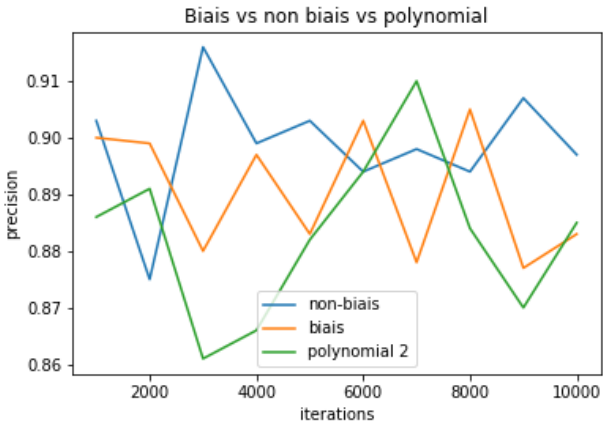
le fonction coût) et en classifiant les nouvelles examples. Le biais correspond le premier terme w_0 dans (Équation 4).

$$f(x_i) = \sum_j x_{ij} w_j = x_i \cdot w^T \tag{3}$$

$$w = [w_0, w_1, .., w_d] \tag{4}$$

À grace de w_0 nous pouvons en theorie augmenter l'expressivité de model. En revanche, la précision de classification n'augmente pas forcement, si on ajoute biais. Dans le figure Figure 12 on voit que la précision (avec validation croisée de 5 fois) avec biais est supérieur de la précision de la formulation non-biais seulement quand on effectue 2000, 6000 ou 8000 itérations. Mais en général il est inférieur.

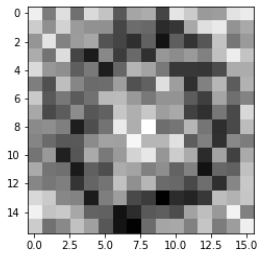
FIGURE 12 – Les scores avec validation croisée de 5 iterations



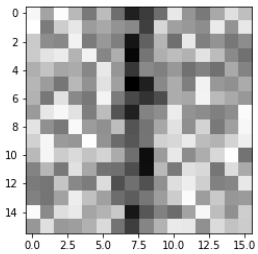
4.2 USPS et sur-aprentissage

Nous allons utiliser les données des USPS, les chiffres manuscrits. Le model w qu'on apprendre en classifiant les chiffres peut être interpretable. Dans le Figure 13 nous voyons les chiffres 0 (a) et 1 (b), mais le model pour reconnetre les chiffres 4 et 9 (presenté en c) n'est pas trés interpretable.

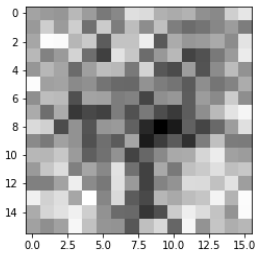
FIGURE 13 – Le model w dans perceptron



(a) 0 contre 1



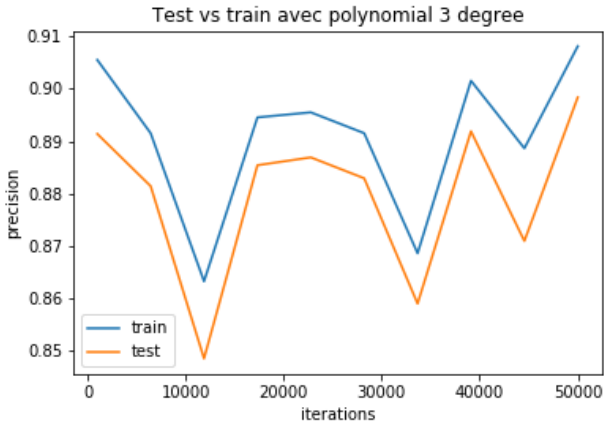
(b) 1 contre 0



(c) 4 contre 9

J'ai etudié, quelle est l'effet de polynomial pour sur-apprentissage. En faisant varier le degree de polynomial, j'ai pas noté grande difference. Dans Figure 14 j'ai plotté les results (chiffre 4) avec polynomial de 3 degrée, et on peut voir que le score ne tombe pas beaucoup.

FIGURE 14 – Le chiffre 4 contre les autres avec polynomial de 3 degrée



J'ai essayé implementer le projection gaussien, mais mon implementation n'arrive pas classifié que 0.5 des examples. Soit il y a un bug dans mon code, soit j'ai mail comprit le formulation de projection gaussien, mais je preferé avancer à SVM et utiliser kernel gaussian.

5 TME 6 : Support Vector Machine

Nous sommes d'abord interessé par les frontières de décisions et les vecteurs supports (les points dont les coefficients sont non nuls). Nous effectuons un premier experience (sans validation croisée) et nous obtenons les résultats suivants (Tableau 1) pour 1000 points aléatoires :

TABLE 1 – Les vecteurs supports parmis 1000 points sans validation croisée

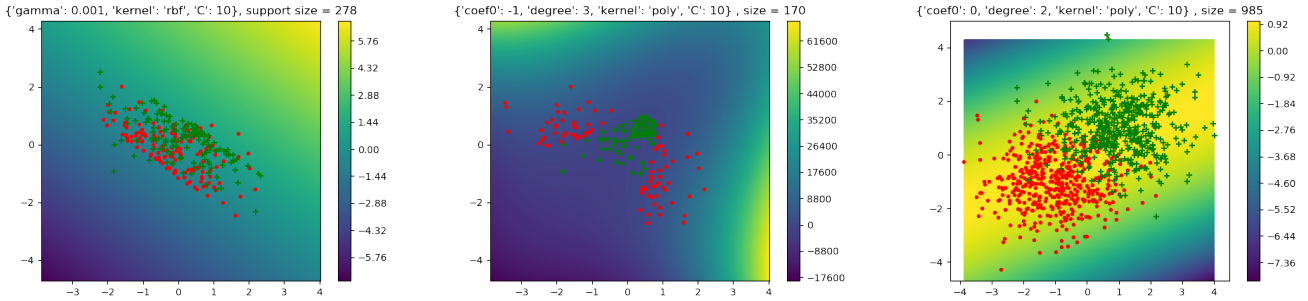
Noyaux utilisé	Taille de vecteurs supports
Polynomial	278
Polynomial (degree 2)	974
Polynomial (degree 3)	278
Polynomial (degree 4)	982
Linear	213
Linear (C = 0.2)	223
Sigmoid	154
Sigmoid (coef0 1)	194
Gaussien	233
Gaussien (gamma 2)	297

Pour polynomial, on peut noter que son degrees paire (2,4) créent beaucoup plus des coefficients non nul que impaires(1,3), propablement à cause de nature des fonctions parabole (degree paire) et cubique (degree impaire). Le nombre de paramètres de SVM, qu'on doit considérer, est grande, comme on peut voir dans le Tableau 1. Pour ça on va utiliser technique GridSearch pour considérer tous les combinations des paramètres et choisir celle avec le meilleur score.

5.1 GridSearch

J'ai utilisé le module GridSearchCV de sklearn pour iterer les combinations des paramètres et choisir le meilleur. Selon cette discussion [4] Sigmoid n'est pas un valid fonction de noyaux, donc je vais considerer que Gaussian,Lineaire et Polynomial comme les fonction des noyaux. Je fais varier gamma, C, degree et coef0.

FIGURE 15 – Les données artifiel et ses vecteurs supports, trouvée par GridSearchCV (validation croisée de cinq fois)



(a) Precision moyenne 0.92003 (b) Precision moyenne 0.81418 (c) Precision moyenne 0.47953

Dans le Figure 15 on a trois combinations differentes des paramètres de SVM. Figure 15 a) est le meilleur classifier, que GridSearchCV a trouvée, puis b) et c) sont classifiers un peu moins precis. L'objectif de SVM est maximiser la marge entre deux classes, pour qu'il soit robuste en face de nouvelles données. Maximiser la largeur de la marge est équivalente que minimiser le vecteur W des vecteurs (ou points) supports.

Intuitivement, ça semble que dans le figure a), les vecteurs supports (les points dans le figure) sont concantre et la marge est grande ; également la précision de classifier est élevée. Par contre dans le figure c) les nombre des points est grande (W est grande ?) donc le marge est plus petit et on arrive pas classifier bien.

5.2 Apprentissage multi-classe

Jusqu'à maintenant nous avons classifié les exemples qu'en cas binaire, mais déjà les données USPS contiennent plusieurs classes. Il y a deux stratégies populaires pour classification multi-classe : One-versus-one et One-versus-all. J'ai implémente que le one-versus-one, car c'est plus simple et je n'ai plus de temps. [5].

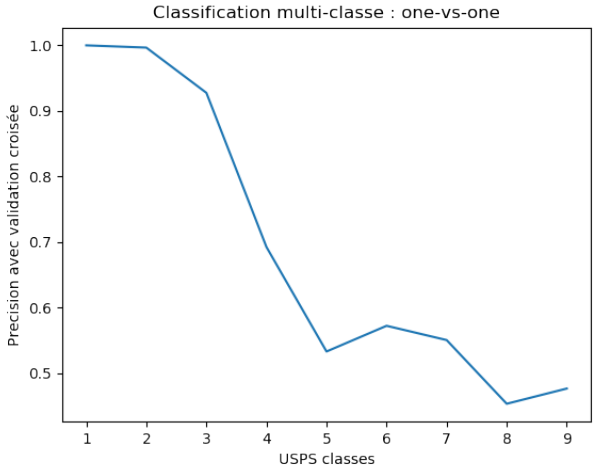
J'ai testé cette methode avec les données USPS (Figure 16). La classe 0 est present toujours et j'ai introduit les classes un par un. Le score se diminue evidement, quand on ajoute plus des classes, mais également parce que certaines classes sont plus difficile de reconnaitre.

5.3 String Kernel

Dans le SVM on peut remplacer le produit matriciel $< w^T \times x >$ par n'importe quelle fonction de noyaux. J'ai implémenté [6] un noyau pour les chaînes des caractères pour classifier des auteurs

4. https ://stackoverflow.com/questions/21390570/scikit-learn-svc-coef0-parameter-range
5. Code dans TME5/MultiClassClassifier.py
6. Code dans TME5/StringKernel.py et text_svm.ipynb

FIGURE 16 – Classification mutli-classe des USPS



différentes des textes, tel que proposé dans l'annonce de TME.

Références

A rule-of-thumb bandwidth estimator. Kernel density estimation — Wikipedia, the free encyclopedia, 2018. URL `https://en.wikipedia.org/wiki/Kernel_density_estimation#A_rule-of-thumb_bandwidth_estimator`. [Enligne ; consulté 20 Mars 2018].