Rapport des TME
Machine Learning

Esther CHOI (3800370)
Master 1 DAC

March 24, 2022

Contents

1 TME1 : Arbres de décision, Sélection de modèles

1.1 L'essentiel sur les arbres de décision

3) Lorsque l'entropie d'un genre vaut 0, cela signifie qu'il n'y a pas de film de ce genre. Lorsque l'entropie d'un genre vaut 1, cela signifie que tous les films sont de ce genre. Ainsi, l'entropie est la plus grande pour le genre Drama (qui vaut 0.69) et l'entropie la plus faible correspond aux genres Adult, News, Reality-TV, Talk-Show et Game-Show (qui vaut 0).

1.1.1 Quelques expériences préliminaires

4) La figure 1 montre un arbre de décision de profondeur 4. Le nombre d'exemples séparés diminue avec la profondeur. C'est normal puisque l'on chercher à discriminer de plus en plus les exemples pour les classer.

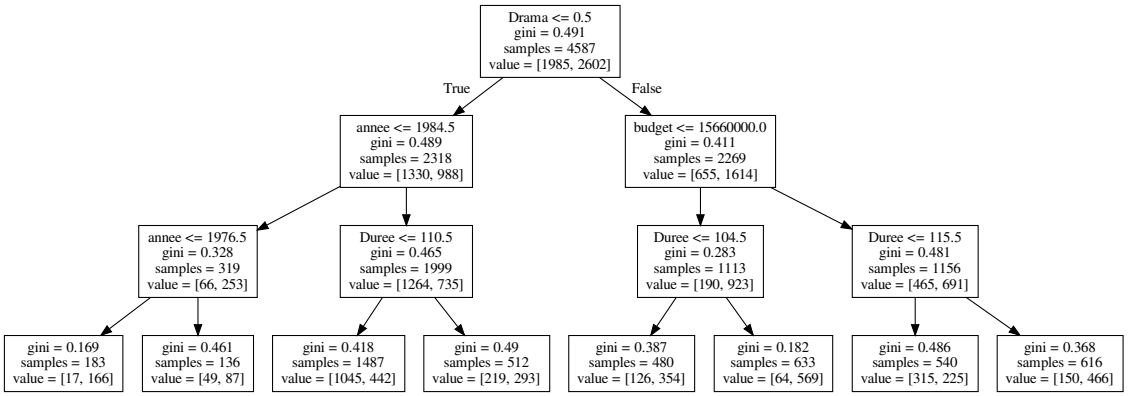


Figure 1: Arbres de décision de profondeur 4

5) Le score augmente (ou de façon équivalente, l'erreur diminue) avec la profondeur. C'est normal puisqu'il y a de plus en plus de feuilles, donc on gagne en précision. Cependant, une erreur trop basse peut-être signe de sur-apprentissage.

6) Ces scores ne sont pas un indicateur fiable puisqu'ils sont calculés sur l'ensemble des données. Il faudrait séparer les données d'entraînement des données de test.

1.2 Sur et sous apprentissage

7) Graphiques de la figure 2

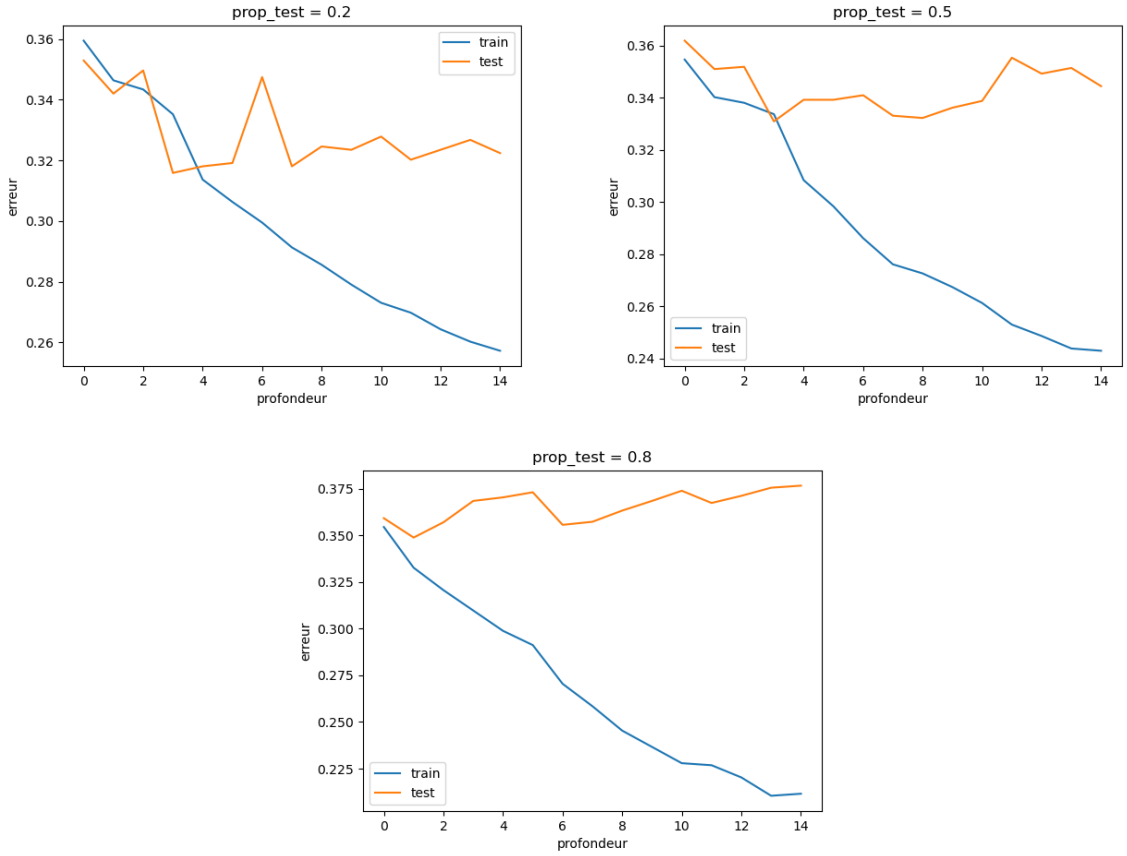


Figure 2: Score en train (bleu) et de test (orange) pour 3 partitionnements (0.2, 0.5, 0.8 de test)

8) Lorsqu'il y a peu d'exemples d'apprentissage, l'erreur d'apprentissage diminue moins vite que lorsqu'il y en a beaucoup. Par contre, plus il y a de données d'apprentissage, plus l'erreur en test est grande.
On peut aussi remarquer que le modèle sur-apprend quand la profondeur de l'arbre est trop grande : l'erreur d'apprentissage est très faible et l'erreur de test augmente. Une profondeur de 4 donne les meilleures performances.

9) Les résultats sont plus fiables qu'avant, mais pour mieux évaluer nos modèles, on peut utiliser la validation croisée (voir figure 3).

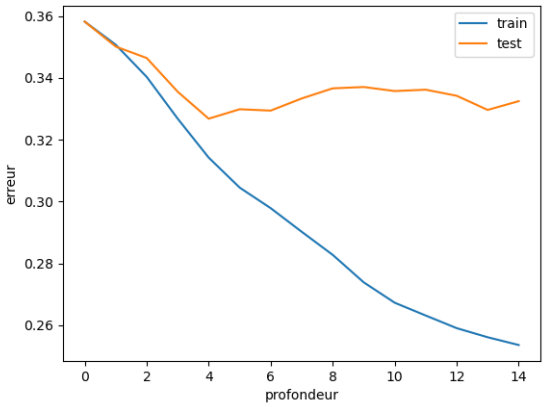


Figure 3: Erreur en train (bleu) et test (orange) en cross validation

2 TME 2 : Estimation de densité

2.1 Méthode par histogramme

Pour bakery : Le paramètre steps correspond au pas de discrétisation. Il fait varier la taille des cases pour construire l'histogramme (voir figure 4).

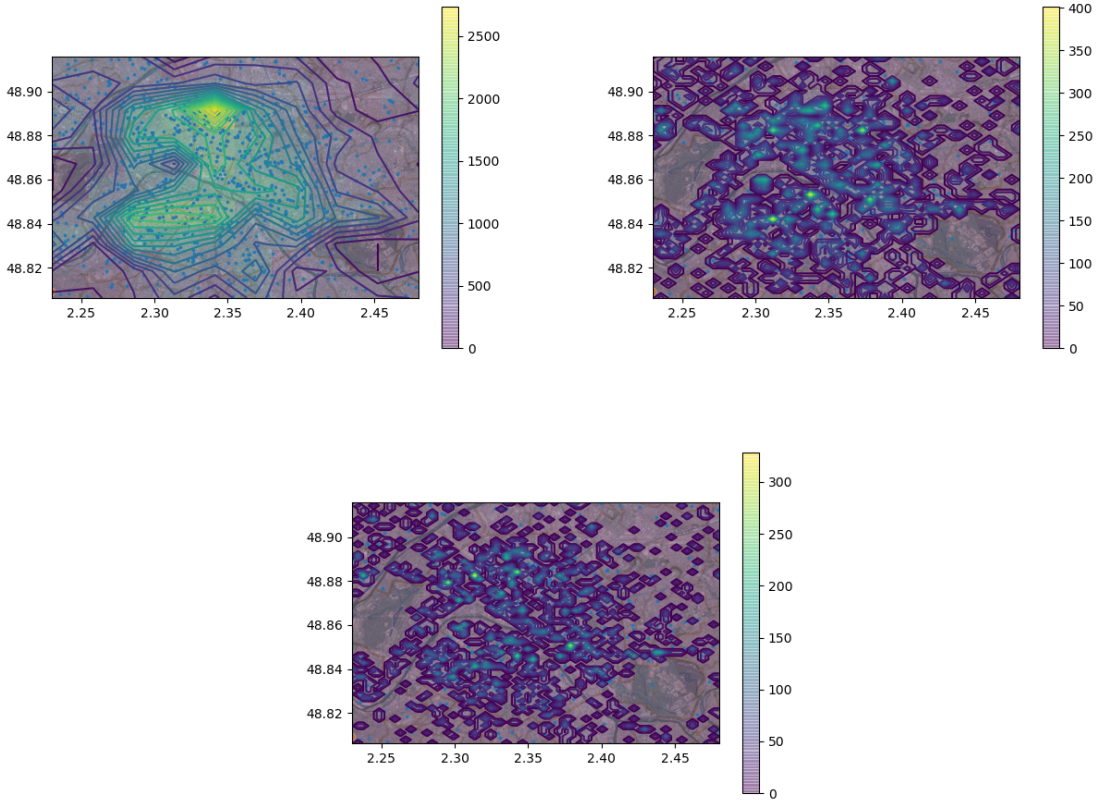


Figure 4: Densité pour 3 valeurs de steps : 10 (gauche), 50 (droite), 70 (bas)

Le meilleur pas (celui qui maximise le score, le score étant la log-vraisemblance des données) est 8, comme on peut le voir sur le graphique de la figure 5.

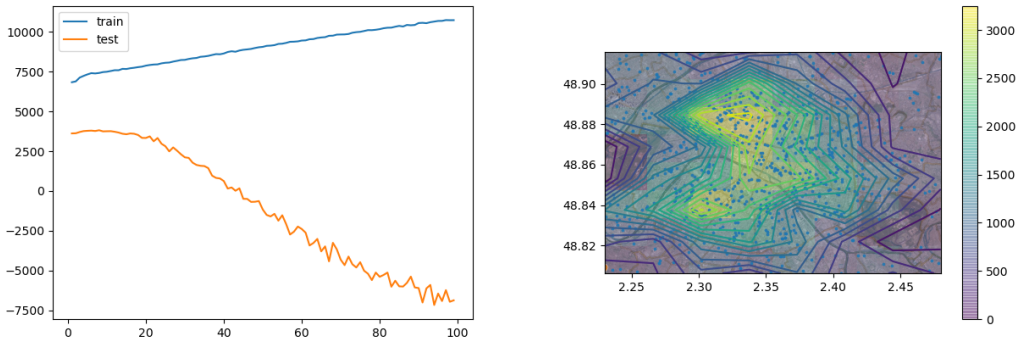


Figure 5: Score en fonction du pas pour bakery et meilleure densité

Pour night club : On obtient 4 comme meilleur pas (figure 6).

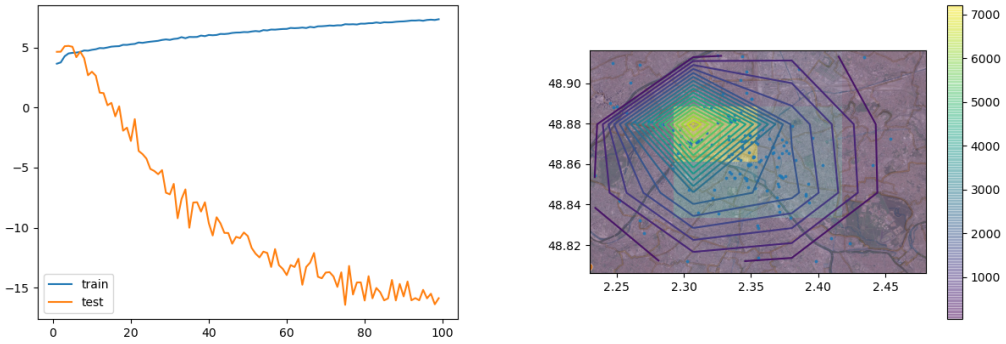


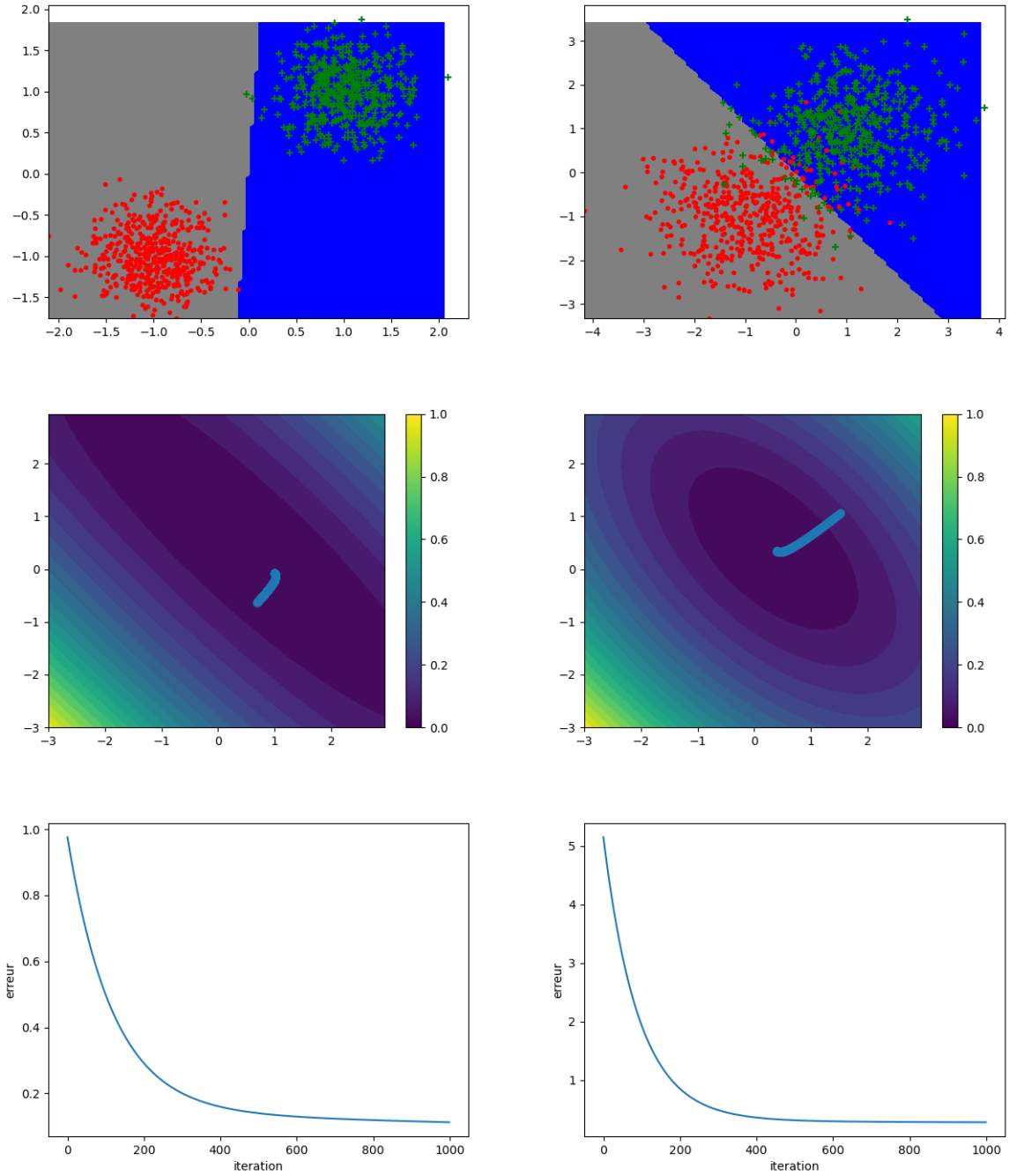
Figure 6: Score en fonction du pas pour night club et meilleure densité

Conclusion Ainsi, le nombre de pas doit augmenter lorsqu'il y a beaucoup de données.

3 TME 3 : Descente de gradient, Régression

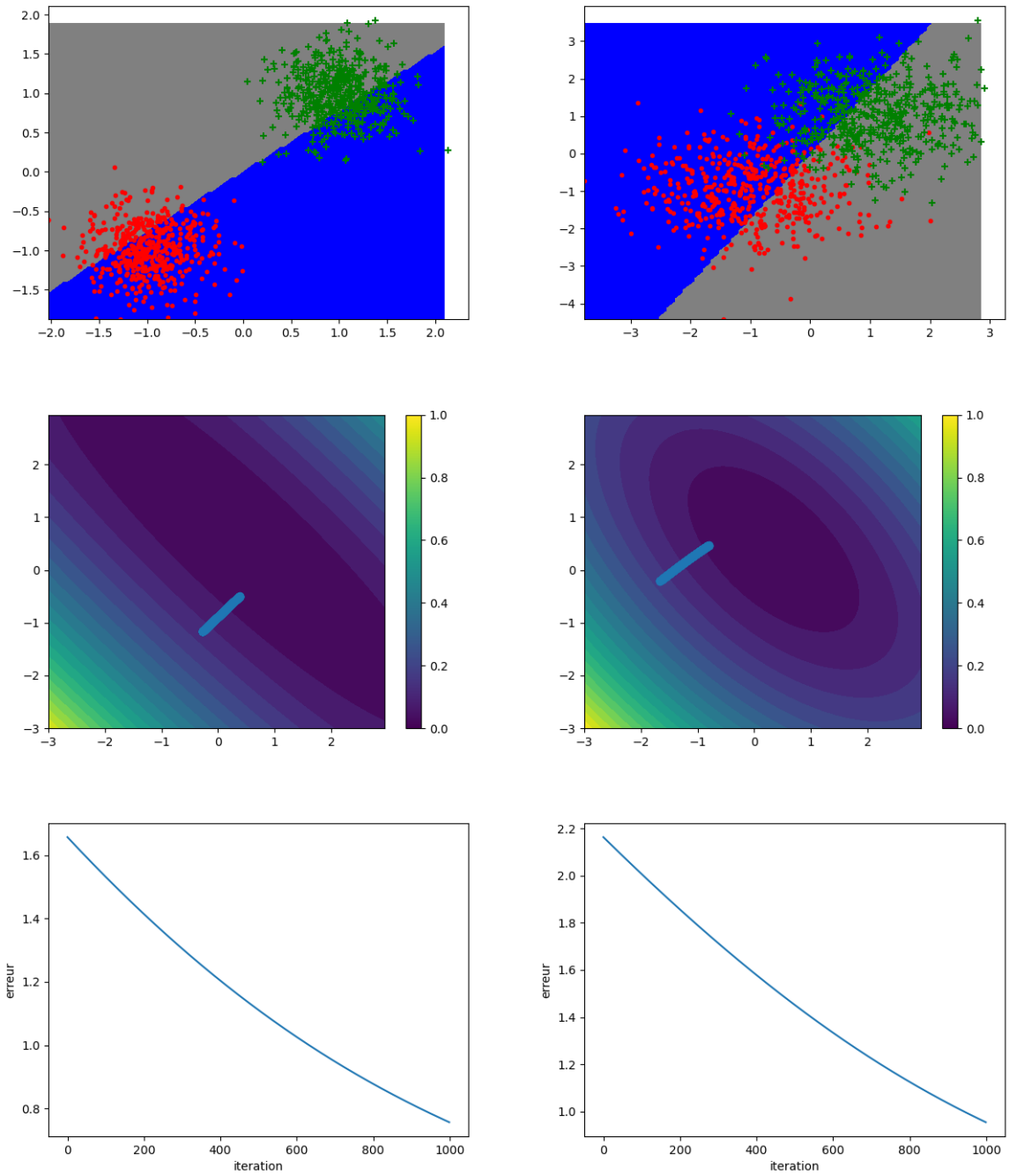
Les modèles sont testés sur un jeu de données aléatoire tiré selon un mélange de deux gaussiennes, d'abord séparable, puis non séparable.

3.1 Régression linéaire



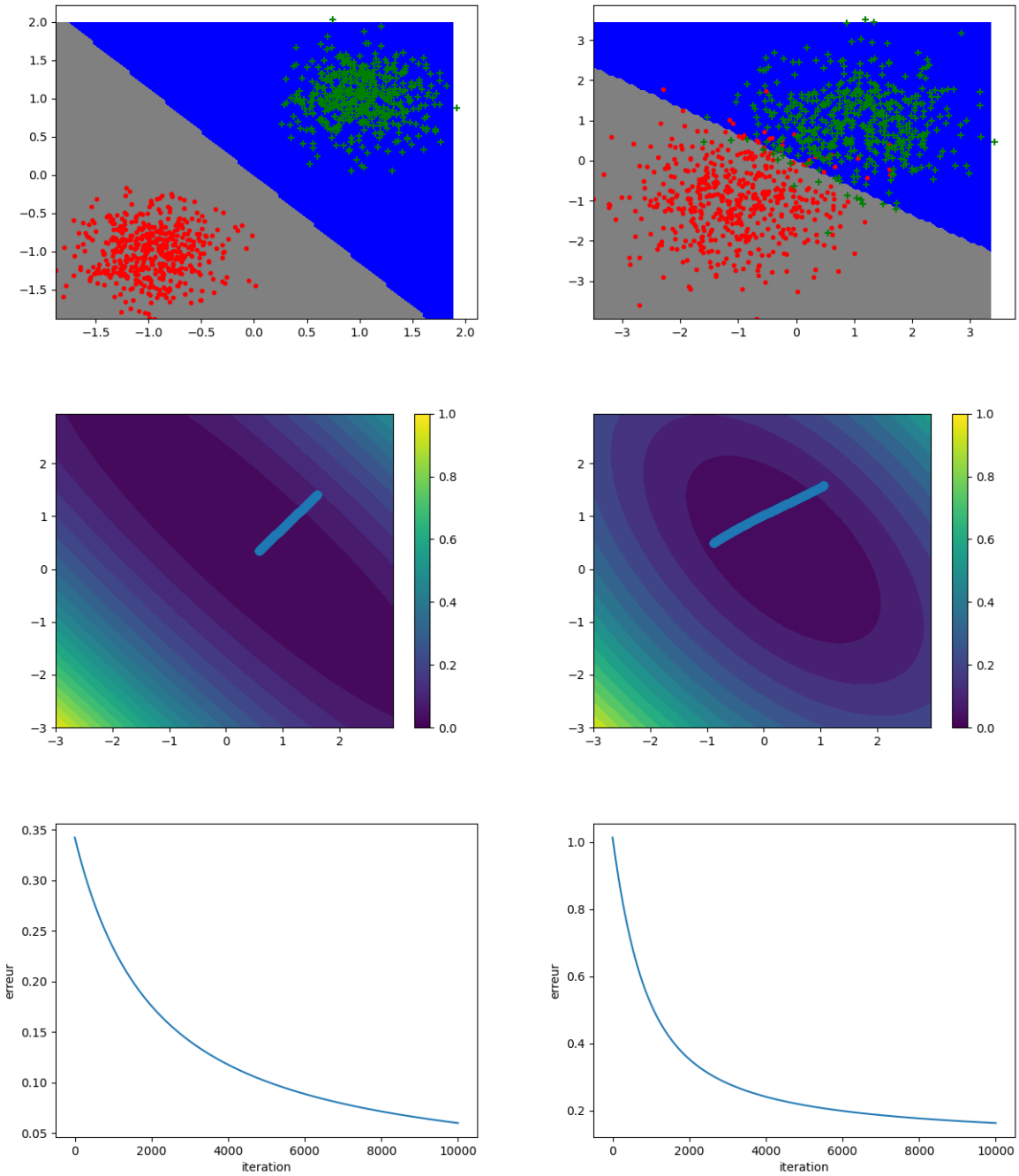
À travers ces graphiques, nous pouvons voir que plus on fait d'itérations, plus l'erreur diminue, et que bien sûr, on fait plus d'erreur avec des données non séparables (droite) qu'avec des données séparables (gauche).

3.2 Régression logistique



Nous pouvons tirer les mêmes conclusions que pour la régression linéaire. Cependant, nous pouvons remarquer que la régression logistique converge moins vite que la régression linéaire. Le tableau suivant montre l'erreur moyenne avec 1000 itérations.

	séparable	non séparable
linéaire	0.1127	0.281
logistique	0.7563	0.9543



Avec 10000 itération, la régression logistique finit par être meilleure que la régression linéaire avec 1000 itérations : on a alors 0.0777 d'erreur pour des données séparables et 0.2532 pour des données non séparables.

4 TME 4 : Perceptron et Projection

4.1 Perceptron et classe `Lineaire`

On teste le perceptron sur des données jouet comme dans le TME3. On voit que le modèle marche et qu'il converge plus vite que les régressions : avec seulement 200 itérations, on obtient déjà 0.2983 d'erreur (figure 7).

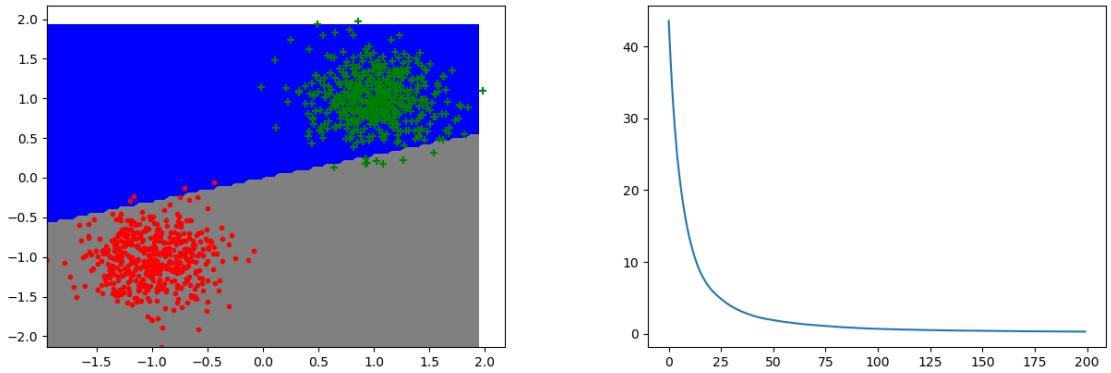


Figure 7: Perceptron sur des données jouet

4.2 Données USPS

Un contre un Lorsque l'on fait du un contre un, on reconnaît la forme des deux chiffres choisis dans la matrice `w` finale (figure 8). Les grandes valeurs dans `w` correspondent au chiffre codé 1, et les petites valeurs correspondent au chiffre codé -1. En effet il faut mettre un grand poids sur les pixels du chiffre positif, et un petit poids sur les pixels du chiffre négatif pour pouvoir au mieux les distinguer.
D'après la courbe d'erreur (de test en fonction du nombre d'itération), on remarque que l'algorithme converge assez vite, mais les résultats ne sont pas satisfaisant : nous n'avons que 50% de taux de bonne classification en apprentissage et 49% en test.

Un contre tous Ici, on ne reconnaît pas de forme particulière pour `w` (figure 9) mais les résultats sont bien meilleurs : on est de l'ordre de 80% de taux de bonne classification en apprentissage et un peu moins en test.
Dans la courbe d'erreur, on voit un petite montée vers la fin.

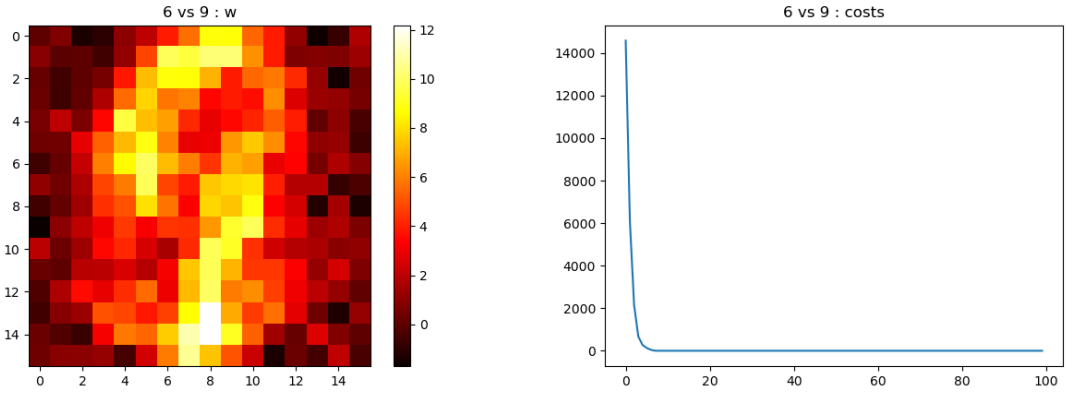


Figure 8: 6 vs 9 (100 itérations)

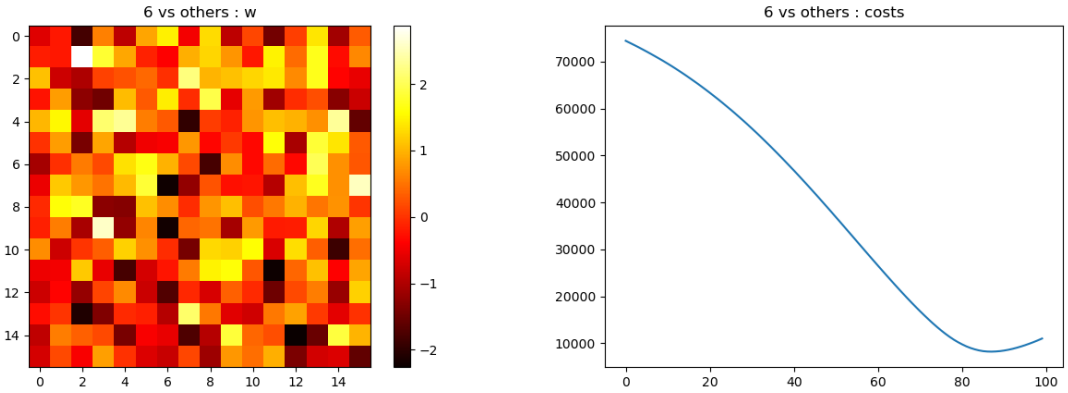


Figure 9: 6 vs all (100 itérations)

5 TME 5 : SVM

5.1 SVM et Grid Search

Prise en main et Tests préliminaires Pour commencer, j'ai essayé la SVM sur des données jouet tirées aléatoirement par mélange de 4 gaussiennes, sur quatre noyaux différents : linéaire (`linear`), polynomial (`poly`), gaussien (`rbf`), et sigmoïde (`sigmoid`). La figure 10 montre les frontières obtenues avec les paramètres par défaut :

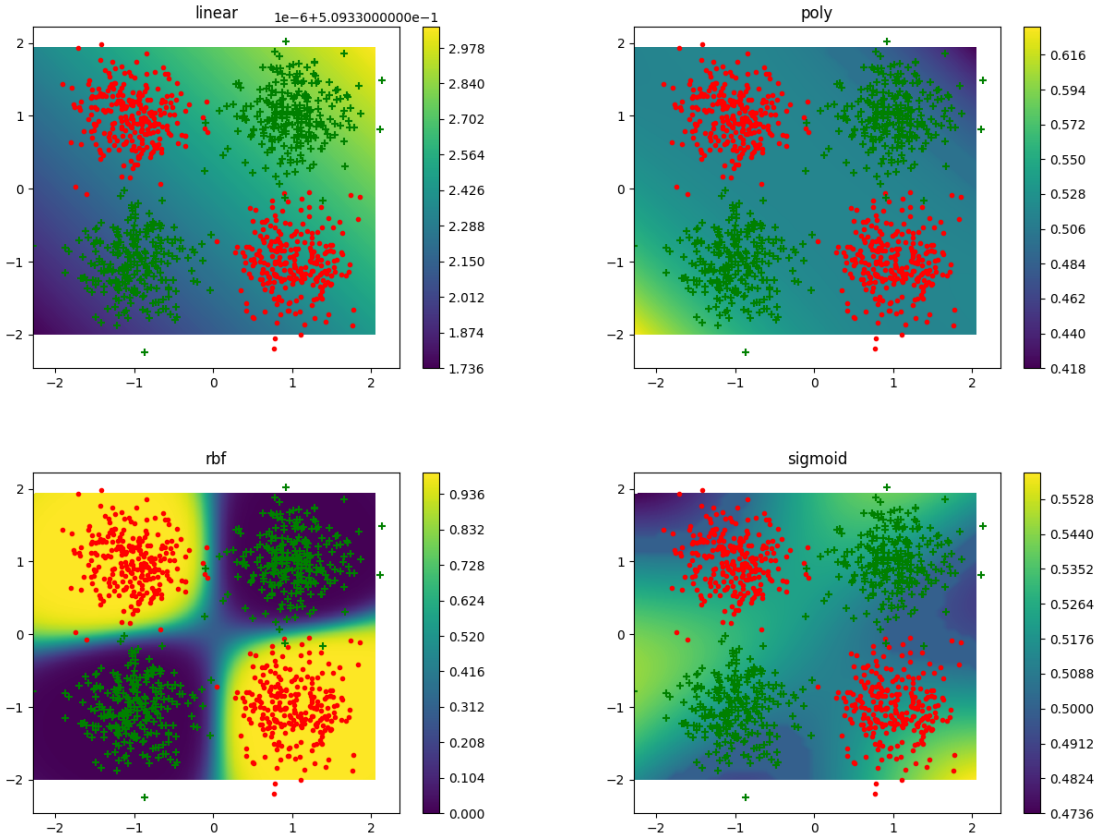


Figure 10: SVM avec quatre noyaux

On obtient les scores suivants : 0.448 pour le noyau linéaire, 0.5 pour le noyau polynomial, 0.992 pour le noyau gaussien et 0.496 pour le noyau sigmoïde. Le noyau gaussien est donc de loin le meilleur pour ce type de données.

Pour déterminer les meilleurs hyper-paramètres, on fait un grid search et on évalue par validation croisée.

Noyau linéaire Le seul paramètre intéressant à étudier est le paramètre de régularisation `C`.

Dans la figure 11, on voit que le meilleur paramètre de régularisation se situe aux alentours de 0.3. D'autre part, le nombre de vecteurs support diminue quand le paramètre de régularisation augmente.

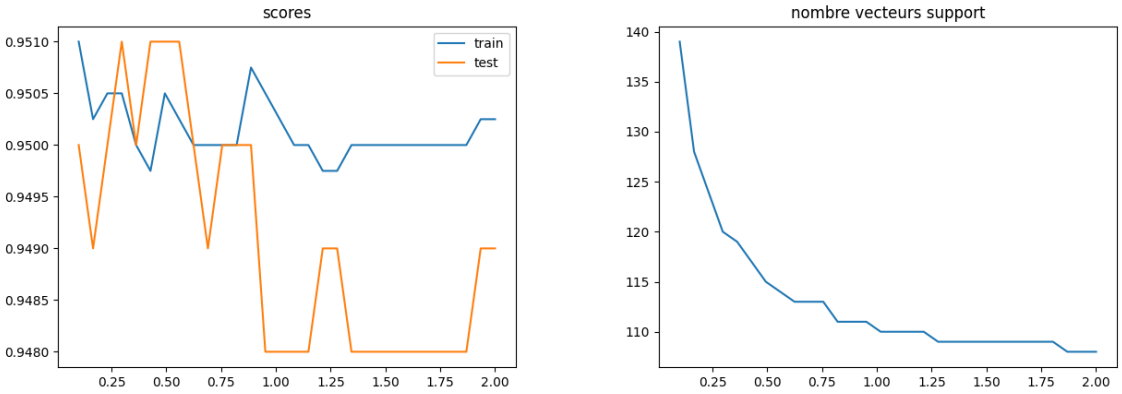


Figure 11: Grid Search pour le modèle linéaire