Travaux Pratiques Apprentissage Automatique

M2 BioInfo

Objectifs

Programmer quelques méthodes de classification statistique vues en cours (estimation de gaussiennes, séparation linéaire, kppv, Parzen), évaluer leur performance sur des jeux de données test, comparer les méthodes, ...

Evaluation

Rapport ou notebook jupyter présentant l'implémentation des méthodes, les résultats obtenus ainsi qu'une petite discussion autour des résultats...

Données de travail

On dispose pour 3 problèmes de classification à 5 classes {1,2,3,4,5} dans R^2 de 100 échantillons par classe pour l'apprentissage (data_tp{1,2,3}.app) et 100 échantillons par classe pour le test (data_tp{1,2,3}.dec) des classifieurs. On peut visualiser la répartition de ces échantillons (data_tp{1,2,3}_{app,dec}.gif).

Estimation de gaussiennes

- Programmer un classifieur par distance euclidienne minimum. Le tester sur les 3 jeux de données. Evaluer ses performances (taux de bonne classificationen Top1 et Top2, matrice de confusion). Conclusion.
- Programmer un classifieur par distance de Mahalanobis minimum. Le tester sur les 3 jeux de données. Evaluer ses performances (taux de bonne classification en Top1 et Top2, matrice de confusion). Conclusion.
- Comparer les performances des deux classifieurs notamment en dissociant l'analyse sur les 3 jeux de données. Conclusion.

Kppv

- Programmer un classifieur 1ppv. Le tester sur les 3 jeux de données. Evaluer ses performances (taux de bonne classification en Top1, matrice de confusion). Conclusion.
- Programmer un classifieur kppv avec vote à la majorité et vote à l'unanimité. Déterminer par 5-CV la valeur de K. Le tester sur les 3 jeux de données. Evaluer ses performances (taux de bonne classification en Top1 et Top2, matrice de confusion). Conclusion.
- Comparer les performances notamment en dissociant l'analyse sur les 3 jeux de données. Conclusion.

Parzen

- Programmer un classifieur de Parzen avec un noyau uniforme et un noyau gaussien. Le tester sur les 3 jeux de données. Evaluer ses performances (taux de bonne classification en Top1 et Top2, matrice de confusion). Conclusion.
- Déterminer par 5-CV la meilleure valeur de l'hyper-paramètre h pour les deux noyaux et sur les 3 jeux de données. Evaluer leurs performances (taux de bonne classification en Top1 et Top2, matrice de confusion). Conclusion.
- Comparer les performances des deux noyaux et des hyper-paramètres notamment en dissociant l'analyse sur les 3 jeux de données. Conclusion.

Séparation linéaire

- Programmer l'algorithme d'apprentissage du perceptron vu en cours en cherchant un hyperplan qui sépare une classe d'une autre (soit pour 5 classes, 5*4/2 hyperplans à déterminer). Le tester sur le jeu de données TP1.
- Adapter l'algorithme pour qu'il converge vers une solution sur les jeux de données T2 et TP3 (non linéairement séparables). Evaluer les performances sur ces deux jeux de données.
- Après avoir supprimé des 3 jeux de données la classe centrale (classe 'e' ou 5), adapter votre algorithme de décision en cherchant un hyperplan qui sépare une classe de toutes les autres (soit pour 4 classes, 4 hyperplans à déterminer). Evaluer les performances (taux de bonne classification en Top1, matrice de confusion) sur le jeu de données TP1 et comparer à la version one-vs-one.
- Comparer l'ensemble des solutions possibles sur les 3 jeux de données. Conclusion.

Bagging

- Programmer l'algorithme du bagging, pour un nombre de classifieurs fixé à l'avance, en utilisant comme classifieur de base la séparation linéaire adaptée (convergeant lorsque non linéairement séparable). Evaluer les performances (taux de bonne classification en Top1, matrice de confusion) sur les 3 jeux de données.
- Déterminer par 5-CV le nombre « optimal » de classifieurs dans l'ensemble. Comparer les performances et la valeur « optimale » du nombre de classifieurs dans l'ensemble sur les 3 jeux de données. Conclusion.
- Comparer avec un bagging de kppv. Conclusion