Statistique en Grande Dimension et Apprentissage - TP 1

Ce TP est consacré aux k-plus proches voisins. On utilisera de préférence le logiciel Python. A travers la mise en oeuvre de cet algorithme, les objectifs sont de se familiariser avec le calcul empirique du risque, la validation simple, la validation croisée ainsi que le choix de paramètres (ou hyper-paramètres). On abordera aussi la notion de surapprentissage.

Exercice 1 (Données simulées)**.** On se place ici dans le cadre suivant. On suppose que $\mathbf{X}$ suit la loi uniforme sur $[0,1]^2$ et que Y est une variable aléatoire à valeurs dans $\{0,1\}$ telle que

$$\mathbb{P}(Y=1|\mathbf{X}=x) = \begin{cases} \alpha & \text{si } |X_1+2X_2| \leq 1 \\ \beta & \text{si } |X_1+2X_2| > 1. \end{cases}$$

1. Ecrire une fonction pour générer un échantillon $\mathcal{D}_n$ de taille n.
2. Simulez un échantillon d'entraînement (`Xtrain`,`Ytrain`) de taille `trainsize` et un échantillon test de taille $n-$`trainsize` également que vous noterez (`Xtest`,`Ytest`). Vous pourrez utiliser le programmer à la main puis utiliser le module train_test_split de sk-learn.
3. Faire une représentation graphique de l'échantillon d'entraînement en attribuant une couleur différente et un symbole différent selon la valeur de Y (on pourra utiliser le module `relplot` de la librairie `seaborn` dont on trouve un tutoriel à ce lien.
4. Appliquez la fonction `KNeighborsClassifier` du package `sk-learn` pour prédire les points de coordonnées $(1/2, 1/2)$ et $(1/4, 3/4)$ (avec par exemple $k = 5$).
5. Prédire les labels de l'échantillon test avec $k = 1$ et $k = 20$.
6. Tracez un graphe affichant bien/mal classés et leur classes.
7. A l'aide de la fonction `plot_confusion_matrix`, afficher la matrice de confusion (qui permet de visualiser les classements prédits et les classements réels) pour ce test.
8. En déduire (l'estimation de) l'erreur de classification associée.
9. Calculez l'erreur par validation simple pour $k = 1 : 20$ et tracez l'évolution sur un graphe. Que décidez-vous ?
10. Calculez l'erreur test pour ce choix optimal. Que constatez-vous ? (On comparera à l'erreur de validation simple)
11. Calculez l'erreur par validation croisée sur tout l'échantillon en utilisant la fonction `knn.cv` (validation croisée de type Leave-One-Out) puis en programmant une validation croisée 5-fold. Que constatez-vous ? Quel est le choix optimal de k dans le cas 5-fold ? (On pourra utiliser le module `GridSearchCV`).
12. Comparez l'erreur de validation croisée à l'erreur test pour le choix optimal de l'hyperparamètre k.

Exercice 2. Récupérer le jeux de données `pulsar` et les afficher. Il s'agit d'un jeu de données astrophysique dont chaque ligne correspond à un objet du ciel nocturne. A partir de 8 paramètres, l'objectif est d'être capable de prévoir si cet objet est un pulsar ou non

1. Fabriquer aléatoirement un échantillon d'entraînement et un échantillon test (de même taille).
2. Représenter une projection en $2d$ des données d'entraînement (et de leur label). Qu'en pensez-vous ?
3. Fabriquer un algorithme de k-plus proches voisins adapté à ce problème (en choisissant l'hyperparamètre par une procédure de validation simple ou croisée avec les proportions de votre choix).
4. Erreur de classification "optimale" ? F_1-score ?
5. Affichez pour $k = 3$ et $k = 30$, le graphe projeté des bien/mal classés ainsi que des classes correspondantes.
6. On pourra aussi représenter la *frontière de décision* pour $k = 30$ puis $k = 15$, puis $k = 1$. Dans quels cas peut-on parler de sur ou sous-apprentissage ? (Il existe par exemple le module `DecisionBoundaryDisplay` pour assurer cette tâche).

Exercice 3. Dans ce denier exercice, on s'intéresse à une base de données célèbre : la base de données `MNIST` qui contient 70000 images de chiffres (en noir et blanc) écrits à la main, en 28×28 pixels.

1. Chargez les jeux de données `mnist_train.csv` et `mnist_test.csv`.
2. En remarquant que la première colonne est celle des réponses, fabriquez des `data.frame`(s) `Xtrain`, `Ytrain`, `Xtest` et `Ytest`.
   `Remarque :` Dans ce jeu de données, on atteint un peu les limites algorithmiques des k-plus proches voisins dont la complexité est en $O(nkd)$ où d est la dimension du problème (ici, $d = 784$).
3. Représentez l'une des observations sous la forme d'un tableau 28×28 afin de vous familiariser avec l'objet.
4. Afin de réduire le temps de calcul, on propose donc dans la suite de travailler seulement avec un sous-échantillon d'entraînement de 12000 observations. Fabriquez ce sous-échantillon en utilisant la fonction `sample`.
5. Mettez en oeuvre l'algorithme sur ce sous-échantillon d'entraînement avec une base test de 500 observations (pour limiter le temps de calcul) et $k = 10$ voisins.
6. Représentez la matrice de confusion et calculez l'erreur de classification.