Cécile Mallet, Sylvie Thiria

TP EX22 : Algorithme des k plus proches voisins (kppv)

Pour une réalisation en Python de cet exercice, utiliser le document « triedpython », et consulter la table d'équivalence Matlab/Python des noms de fonction.

1°) Expliquer le principe de fonctionnement, la nature et les différentes étapes d'un algorithme des k plus proches voisins, puis retrouver les différentes étapes spécifiquement associées à cet algorithme dans le programme **kppvo.m** fourni.

2°) Soit un ensemble de données d'apprentissage en dimension 2 labellisées en 2 classes. Les données sont contenues dans le fichier **DataA.txt** et les étiquettes dans le fichier **labelA.txt.** Cet ensemble d'apprentissage comprend 100 exemples pour chacune des classes, soit 200 au total.
Le programme **demokppvEuMa.m** utilise ces données pour faire ressortir, à l'aide d'un ensemble de test sur un maillage 2D, les frontières de décision obtenues par l'algorithme kppv. On vous demande d'exécuter ce programme, et de rendre compte des résultats.

3°) On dispose maintenant d'un ensemble de 132 données d'apprentissage en dimension 2, labellisées en 3 classes d'égale taille. Les données sont contenues dans le fichier **Data1.mat** et les étiquettes dans le fichier **Label1.mat**. Avec ces données d'apprentissage, le programme **demokppvDatas.m** classe les 99 données contenues dans **Data1test.mat** avec algorithme du plus proche voisin (code **kppvo.m**). Les données à classer ont été étiquetées par un expert, les labels correspondants sont contenus dans **Label1test.mat**. Les labels sont utilisés pour calculer la matrice de confusion (MCO).
En utilisant la fonction **trace.m** de matlab, ajoutez après chaque classification, le calcul du pourcentage de données bien classées. Produire les résultats obtenus pour différentes valeurs de k (1, 2, 3, 4, 5, 10, 15, 20) selon la distance utilisée (faire un tableau).
Pour chacune des distances, reproduire les figures de classification obtenues pour le meilleur et le moins bon des résultats ainsi que la matrice de confusion.

Refaire l'étude avec les fichiers de données suivants : **Data2.mat, Label2.mat, Data2test.mat, Label2test.mat**

4°) On considère un ensemble de 5 points :

x1=(2,2), x2=(2,-2), x3=(-2,-2) qui sont de classe 1 et
x4=(-2,2), x5=(0,0) qui sont de classe 2.

Vous aller devoir écrire un script qui permet de représenter graphiquement les frontières de décision, en utilisant la distance euclidienne. Pour se faire, vous aurez besoin de définir un grillage 2D. Ce grillage pourra être défini en abscisse et en ordonnés de -3 à 3 par pas de 0.2. Les points de ce maillage constitueront l'ensemble de test qui sera utilisé pour faire apparaitre les frontières de décision. Pour construire cet ensemble de points, vous pouvez faire une boucle imbriquée sur les 2 dimensions ou encore utilisées les fonctions **meshgrid.m** et **reshape.m**. Comme on ne veut que faire apparaître les frontières de décision on attribue comme classe désirée la même classe pour tous les éléments de l'ensemble de test (ceci étant demandé par la fonction **kppvo** utilisée). Vous pouvez prendre exemple sur le code **demokppvEuMa.m**.
Faire apparaître graphiquement les frontières de décision en même temps que les 5 points d'apprentissage pour k=1, k=2, etc.

Que se passe-t-il si l'on échange les noms des classes 1 et 2 :

classe 2 : x1=(2,2) x2=(2,-2), x3=(-2,-2)

classe 1 : x4=(-2,2) x5=(0,0)

Ne présenter les frontières de décision que lorsqu'elles sont différentes des cas précédents. Expliquer le résultat.

On recommence maintenant en choisissant d'une manière aléatoire la classe à laquelle est affecté un point de la grille en cas d'égalité du vote (pour cela il faut que vous interveniez dans le code kppvo.m).

Pour monter l'effet de votre intervention, définissez un maillage plus resserré (-3:0.05:3), et montrez la figure de la frontière de décision pour k = 2 voisins uniquement. Commentez.