TP n=°4 : K plus proches voisins

Informatique tronc commun 2ère année

1) Le première étape est de sauvegarder le dossier du TP n=°4 dans votre espace personnel. Ensuite, il faut ouvrir le fichier Python qui est sauvegardé dans votre espace.

2) Compléter la fonction **separer(donnee,ratio)** qui prend en entrée une liste Python `donnee` contenant les données, un entier représentant un pourcentage, et sépare la liste `donnee` en deux listes `apprentissage` et `test`, tel que `apprentissage` contient `len(donnee) * ratio/100` éléments et `test` le reste des éléments de la liste `donnee`.

3) Écrire une fonction **distance_euclidienne(donnee1,donnee2)** qui prend en entrée deux listes Pythons `donnee1,donnee2` représentant des vecteurs. Les listes sont de même taille `n` et contiennent des flottants aux positions `0` jusqu'à `n-2`, leurs coordonnées. Le dernier élément de la liste est la catégorie à laquelle l'élément représenté par la liste appartient.

La fonction renvoie la distance euclidienne entre les deux vecteurs.

4) Écrire une fonction Python **Lprime(x,apprentissage,distance)** qui prend en entrée une liste `x` représentant une donnée, une liste `apprentissage` contenant plusieurs données, et la fonction `distance` utilisée pour calculer la distance.

La fonction renvoie une liste `lp` contenant des couples `(distance,categorie)` décrite dans la première étape des *K plus proches voisins*.

5) Écrire la fonction **plus_petit(Lp,k)** qui prend entrée une liste `Lp` dont la forme est la réponse de la question précédente. La fonction renvoie les `k` éléments de `Lp` dont la distance est la plus faible. On pourra s'aider de la fonction **insertion(rep,elem)**.

6) Écrire la fonction **dico_classe(Lpk)** qui prend en entrée une liste `Lpk` dans la forme est celle renvoyée par la fonction précédente, et renvoie un dictionnaire dont les clefs sont les classes apparaissant dans `Lpk` et leurs valeurs le nombre d'occurrences dans `Lpk`.

dico_classe([(0.2, 'Iris-versicolor'), (0.5, 'Iris-versicolor'), (0.6, 'Iris-versicolor'), (0.7, 'Iris-virginica')]) renvoie
{'Iris-versicolor' : 3, 'Iris-virginica' : 1}

7) Écrire une fonction **max_occurences(dico)** qui renvoie la clef du dictionnaire `dico` qui a la plus grande valeur.

8) En réutilisant les fonctions de la questions 4 à 7, écrire une fonction **k_plus_proches_voisins(apprentissage,x,k,distance)** qui prend en plus des entrées normales, la fonction `distance` qui sera appliquée pour calculer la distance entre deux données.

9) Écrire une fonction **test(donnee,ratio,k,distance)** qui prend en entrée une liste `donnee`, des entiers `ratio,k` et une fonction `distance`.

La fonction crée une base d'apprentissage et de test, effectue les k plus proches voisins sur tous les données de la base de test et renvoie le taux de réussite.

10) Expliquer ce que fait la fonction écrite à la question 10.

Tester pour différentes valeurs de c et observer le changement du taux de réussite. Quelles paramètres de notre base de données sont les plus importants pour distinguer les différentes espèces d'iris.

La fonction Question 10 teste le taux de réussite lorsqu'on augmente ou diminue l'importance d'une colonne. On se rend compte qu'augmenter trop l'importance de la première colonne réduit le taux de réussite de notre classificateur.

11) Écrire une fonction **k_plus_proches_voisins2(apprentissage,x,k,distance)** qui utilise le tri fusion.

12) Changer dans l'importation des données **"iris.csv"** et indiquer le fichier **"voitures.csv"**. Premièrement, tester en utilisant la fonction question 9 le taux de réussite si k = 5.

13) Le fichier contient 850 instances de voitures différentes. Est-ce que cela veut dire que la base est mieux maillée que la base d'iris ?

Pas forcément, car le fichier prend aussi plus de paramètres en compte. La quantité de données nécessaire pour avoir un bon maillage augmente avec le nombre de paramètre mesuré.

14) Avec l'aide de la fonction **tracer2**, écrire une fonction **taux_k(donnee,ratio,distance)** et qui trace un graphique de taux de réussite en fonction de k. On fera varier k de 1 à 51 par pas de 2 et on n'utilisera pas la fonction test, pour garder des bases

d'apprentissage et de test constantes.

15) Tester différentes dilatations des paramètres de la base de voitures, et essayer de trouver les paramètres qui influent le plus sur le taux de réussite.

16) Comparer le temps de calcul de la méthode avec tri et de la méthode sans tri en fonction du k donnée en entrée.
Quand est-il avantageux de trier ?
Il devient avantageux de trier sur le graphique à partir de k = 50. En pratique, on ne trie presque jamais la liste au préalable pour trouver les k-voisins les plus proches.