TP : Algorithme des k-plus proches voisins

Dans ce TP vous allez implémenter une version de l'algorithme kNN pour déterminer l'espèce d'une fleur en fonction de ses caractéristiques. La base de données contient 150 iris, avec la longueur et la largeur des pétales et sépales pour chacune, ainsi que son espèce entre *Iris setosa*, *Iris versicolor* et *Iris virginica*.

I Manipulation de la base de données

La première étape consiste à extraire et mettre en forme la base de données. Il convient d'utiliser :

```
1 from sklearn.datasets import load_iris
2 iris = load_iris()
3
4 # séparer les données entre caractéristiques C et espèce e
5 C = iris.data
6 e = iris.target.astype(int)
```

Il faut alors utiliser la fonction `train_test_split` du module `sklearn.model_selection` afin de séparer les données entre données de test et d'entrainement. La fonction prend en argument le tableaux des données (ici les caractéristiques), celui des valeurs (ici l'espèce) et un argument optionnel `test_size`, et renvoie 4 tableaux correspondants respectivement aux données d'entrainement, aux données de test, aux valeurs d'entrainement et aux valeurs de test.

I.1)
Quel est le type de `C` et de `e` ?

I.2)
Ecrire un dictionnaire permettant d'affecter à chaque espèce représentée dans `e` son nom.

I.3)
Séparer les données en `C_train`,`C_test`,`e_train`,`e_test` avec une répartition entrainement/test 80/20.

I.4)
Les caractéristiques correspondent dans l'ordre à la longueur et largeur des sépales, puis celles des pétales. Représenter sur un graphe (en important matplotlib) la répartition des fleurs test sur un graphe avec en abscisse la longueur des sépales, en ordonnée, la longueur des pétales, et distinguant les fleurs selon leur espèce.

II Définition de la distance entre deux iris

Il faut maintenant choisir comment définir que deux iris sont proches. Pour ça, on va utiliser les 4 caractéristiques, donc chaque fleur est représentée par un vecteur de dimension 4. Dans un espace vectoriel de dimension n, on peut définir plusieurs types de distances entre un vecteur X de corrodnnées x_i et un vecteur Y de coordonnées y_i. Par exemple, la distance de Manhattan $d_M(X,Y) = \sum_i |x_i - y_i|$ ou la distance de Tchebychev $d_T(X,Y) = \max_i |x_i - y_i|$, mais le programme stipule qu'on ne travaillera qu'avec la distance euclidienne $d(X,Y) = \sqrt{\sum_i (x_i - y_i)^2}$.

II.1)
Ecrire une fonction `distance` qui prend en argument deux vecteurs (donc des listes) de dimension quelconque et renvoie leur distance euclidienne.

III Détermination des k plus proches voisins

Il s'agit maintenant de déterminer les k-plus proches voisins d'un iris de la liste `C_test` parmi ceux de la liste `C_train`. Dans un premier temps, on va donc garder la liste de ces k voisins comme une liste de doublets `(d,i)` avec d la distance entre l'iris d'indice `i` de `C_train` et l'iris de `C_test` à classer.

Il peut donc être utile d'écrire un programme `insertion` qui prend comme argument un entier `k`, une liste déjà triée `L` de doublets, une distance `d` et un indice `i` et qui renvoie une liste `L2` (de `k` éléments au maximum) dans laquelle a été inséré le doublet `[d,i]` à la bonne place.

III.1)

Ecrire cette fonction `insertion`.

III.2)

L'utiliser dans un algorithme `plus_proches` d'arguments `k` et `index` qui renvoie la liste des k plus proches voisins de l'iris d'indice `index` de la liste `C_test` sous la forme d'une liste de listes `[d,i]`.

III.3)

Modifier cet algorithme afin qu'il renvoie l'espèce de chacun des k plus proches voisins.

III.4)

Adaptez les fonctions vues en cours dans le cas de l'algorithme de reconnaissance de caractère afin de :

III.4.a.) Déterminer l'espèce la plus probable d'un iris en prenant l'espèce la plus représentée parmi les k plus proches voisins ;

III.4.b.) Calculer le score de l'algorithme que vous avez réalisé en comparant l'espèce déterminée par l'algorithme avec l'espèce de la liste `e_test` ;

III.4.c.) Tracer le score de l'algorithme en fonction de la valeur de k pour déterminer la valeur optimale de k avec ce jeu de données ;

III.4.d.) Ajouter sur le graphique de la partie 1 l'iris à classer ainsi que le résultat de l'algorithme ;

III.4.e.) la matrice de confusion de votre algorithme.