PCE : TP KNN

October 14, 2024

[1]:
```
import numpy as np
```

[2]:
```
from sklearn import datasets
```

On importe la célèbre base de sonnées `iris`. Elle contient des informations sur 3 variétés : Setosa, Versicolor et Virginica. Un ensemble de fleurs a été étudié. Pour chacune on a noté les informations suivantes : longueur et largeur des sépales, longueur et largeur des pétales.

[3]:
```
iris = datasets.load_iris()
```

On récupère la liste des informations sur les sépales et pétales :

[4]:
```
D = iris.data
D[:5]
```

[4]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

Chaque ligne de la matrice ci-dessus est l'enregistrement des données pour une fleur.

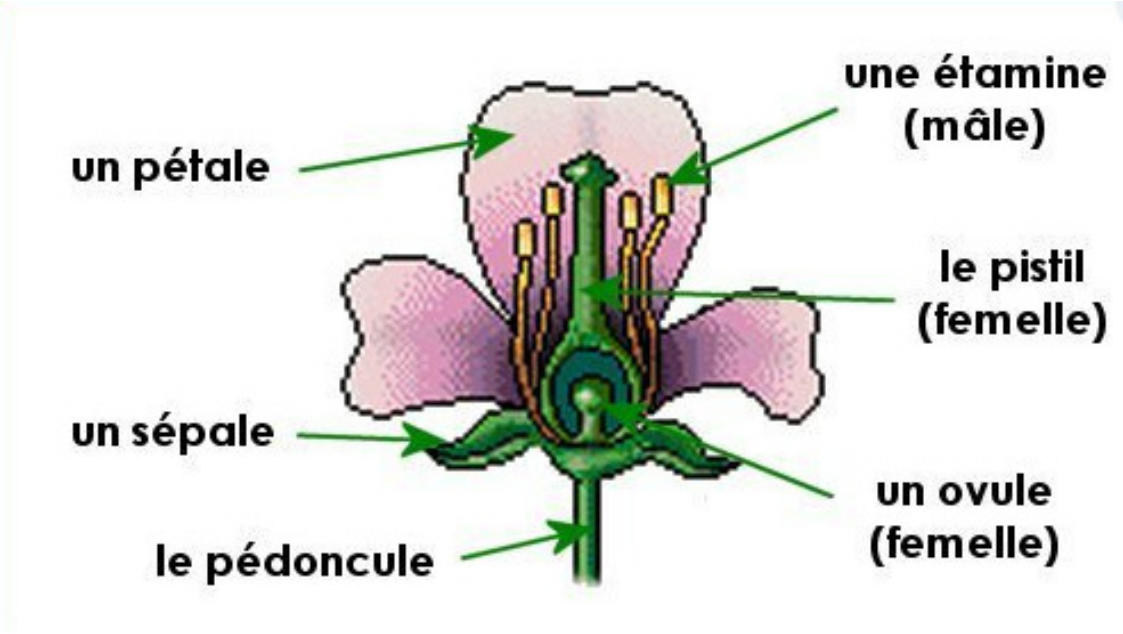
[5]:
```
Y = iris.target
Y, len(Y)
```

[5]:
```
(array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 150)
```

Le tableau `Y` indique, pour chaque numéro de ligne de `D` la catégorie à laquelle appartient la fleur correspondante (0 pour Setosa, 1 pour Versicolor et 3 pour Virginica). Dit autrement, le tableau `Y` représente une partition de l'ensemble des données. La donnée numéro `i` appartient à la classe

d'équivalence numéro `Y[i]`. Ainsi, `D[i]` désigne les caractéristiques longueur et largeur des sépales de la fleur `i`, longueur et largeur des pétales; et `Y[i]` représente la variété à laquelle elle appartient.

Le tableau `D` est appelé *tableau des vecteurs caractéristiques*; `Y` est le *tableau des étiquettes de classe*.



Dans tout ce qui suit D désigne une liste de coordonnées de points à classifier et Y les classes connues de ces points. Pour fixer les idées on prendra D, Y égaux aux tableaux définis ci-dessus mais ce pourrait être tout autre chose.

Question 1

On partitionne `D` et `Y` en deux groupes selon une fonction de choix f. Pour chaque numéro de ligne de D (et donc de Y), f décide si on considère la fleur correspondante comme appartenant aux données d'apprentisage ou aux données de test. L'idéal est que f choisisse aléatoirement, mais pour contrôler nos résultats nous prenons d'abord :

[6]:
```
f = lambda i : i%3==0 # f sélectionne une ligne sur 3
```

Ecrire la fonction `donnees_apprentissage(D,Y,f)` qui prend en paramètres une matrice D de données, un tableau Y de classes de ces données et renvoie 4 tableaux :

- le premier est une matrice constituée des lignes de D qui sont acceptées par la fonction f
- le second est le tableau des classes correspondants aux données acceptées
- les deux derniers tableaux correspondent aux données refusées et leurs classes

Les données acceptées (les deux premiers tableaux renvoyés) servent à l'apprentissage et les données refusées servent aux tests.

[8]:
```
Da,Dt,Ya,Yt = donnees_apprentissage(D,Y,f)
```

[9]:
```
print(Da[:2])# 2 premières données d'apprentissage
print(Dt[:2])# 2 premières données de test
```

```
[array([5.1, 3.5, 1.4, 0.2]), array([4.6, 3.1, 1.5, 0.2])]
[array([4.9, 3. , 1.4, 0.2]), array([4.7, 3.2, 1.3, 0.2])]
```

[10]:
```
print(Yt)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
```

Question 2

Ecrire la fonction `d(x,y)` qui calcule la distance au carré entre deux points x, y de mêmes dimensions

[13]:
```
d([5.1, 3.5, 1.4, 0.2],[4.7, 3.2, 1.3, 0.2])
```

[13]:
```
0.25999999999999945
```

Question 3

Accéder à la partition d'une donnée dont on connaît seulement le vecteur caractéristique et pas la *clé primaire* (c.a.d le "numéro" de la donnée) est coûteux. On écrit la fonction `faire_partitition(D,Y)` qui prend en paramètres le tableau des vecteurs caractéristiques et celui des étiquettes de classe. La fonction retourne un dictionnaire dont les clefs sont des tuples de coordonnées et les valeurs sont les étiquettes de classes.

[15]:
```
partition = faire_partition(Da,Ya)
cpt=0
for k in partition:
    if cpt%10==0:
        print("{}:{}".format(k,partition[k]))
    cpt+=1
```

```
(5.1, 3.5, 1.4, 0.2):0
(4.8, 3.1, 1.6, 0.2):0
(5.0, 2.0, 3.5, 1.0):1
(5.5, 2.6, 4.4, 1.2):1
(6.9, 3.2, 5.7, 2.3):2
```

[16]:
```
partition = faire_partition(D,Y)
partition[(4.9, 3.0, 1.4, 0.2)]
```

[16]:
```
0
```

Question 4

Ecrire la fonction `tri(D,P,d)` qui retourne une version triée du tableau des vecteurs caractéristiques `D` par ordre croissant de distance au point `P`.

[17]:
```
l = [-2,-6,89,3]
l.sort(key= lambda x: abs(x)) # tri par valeur croissante selon val. abs. de x
l
```

[17]:
```
[-2, 3, -6, 89]
```

[19]:
```
M = tri(Da,[0,0,0,0],d)
print(M[:5])
[round(d(M[i],[0,0,0,0]),3) for i in range(5)]
```

```
[array([4.4, 3.2, 1.3, 0.2]), array([4.6, 3.1, 1.5, 0.2]), array([4.8, 3. , 1.4,
0.1]), array([4.8, 3. , 1.4, 0.3]), array([4.6, 3.4, 1.4, 0.3])]
```

[19]:
```
[31.33, 33.06, 34.01, 34.09, 34.77]
```

Question 5

Ecrire la fonction `knn(D,P,d,k)` qui renvoie les k plus proches voisins du point P parmi ceux listés dans D

[22]:
```
"""
Les 3 plus proches, au sens de la distance euclidienne,
voisins de 0 dans les données d'apprentissage
"""
knn(Da,[0,0,0,0],d,6)
```

[22]:
```
[array([4.4, 3.2, 1.3, 0.2]),
 array([4.6, 3.1, 1.5, 0.2]),
 array([4.8, 3. , 1.4, 0.1]),
 array([4.8, 3. , 1.4, 0.3]),
 array([4.6, 3.4, 1.4, 0.3]),
 array([4.8, 3.1, 1.6, 0.2])]
```

[25]:
```
knn(Da,Dt[89],d,6)
```

[25]:
```
[array([6.4, 2.7, 5.3, 1.9]),
 array([6.3, 2.5, 4.9, 1.5]),
 array([6.7, 2.5, 5.8, 1.8]),
 array([6.3, 2.7, 4.9, 1.8]),
 array([6.4, 2.8, 5.6, 2.2]),
 array([6.5, 3. , 5.2, 2. ])]
```

Question 6

On détermine maintenant la classe de la donnée étudiée. On lui attribue la variété majoritaire parmi ses k plus proches voisins. Ecrire la fonction `classe_maj(voisins,dico)` qui prend en paramètre le dictionnaire des noms de classes, une liste de voisins d'un point et renvoie le nom d'étiquette majoritaire parmi ces voisins.

[53]:
```
classe_maj([Da[32],Da[43],Da[40],Da[38]],partition)
```

[53]: 2

Question 7

Implanter la fonction `classe(Da,dico,P,d)` qui prend en paramètres les données d'apprentissages, le dictionnaire des noms de classe, un point `P`, une distance `d` un nombre de voisins `k`. la fonction retourne la classe de `P` telle que calculée par l'algorithme des plus proches voisins.

[30]:
```
classe(Da,partition,Dt[89],d,6)
```

[30]: 2

Question 8

Ecrire la fonction `matrice(A,T,p,k,d)` qui prend en paramètre les données d'apprentissage, celles de tests, le dictionnaire du partitionnement, le nombre de voisins pour KNN et la distance utilisée.

La fonction retourne la matrice de confusion calculées d'après les données de tests.

[46]:
```
p=faire_partition(D,Y)
M = matrice(Da,Dt,partition,3,distance)
M
```

```
3
```

[46]: [[33, 0, 0], [0, 30, 3], [0, 2, 32]]

Question 9

Ecrire la fonction `taux(M)` qui prend en paramètre une matrice de confusion et retourne le taux d'ereur et la précision.

[49]:
```
M = matrice(Da,Dt,p,3,d)
taux(M)
```

[49]: (0.05, 0.95)

[ ]: