Chapitre 9

k plus proches voisins

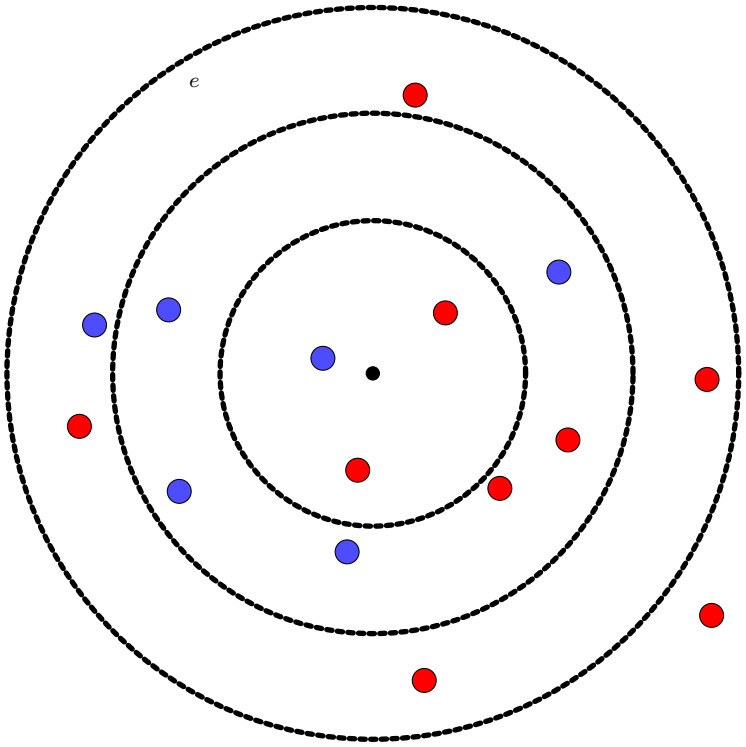
Table des matières

1 Principe

Définition 1 :

L'algorithme des k plus proches voisins (ou k-NN , soit k-*nearest neighbors*) fait partie des algorithmes de machine learning.
Nous disposons d'un ensemble d'éléments de classes différentes et d'une distance sur cet ensemble.
Le but d' algorithme est de trouve la classes d'un nouvel élément arrivant dans cet ensemble.

Exemple : Le point noir central est un point dont nous ne connaissons pas la couleur. L'idée est, à l'aide de l'algorithme du k-NN, de la déterminer :

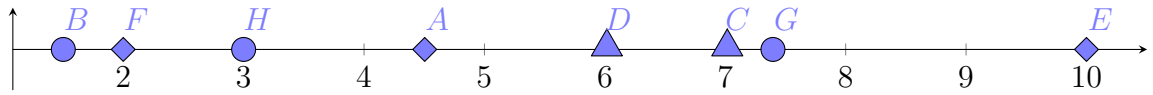


Valeur de k	Rouges	Bleus	Couleur choisie

2 Application

2.1 Sur une droite

Exercice 1 : Nous disposons d'une droite graduée, d'un ensemble de points ayant des formes différentes :



Points	Forme	Position
A	Losange	4,5
B	Rond	0,5
C	Triangle	7
D	Triangle	6
E	Losange	10
F	Losange	2
G	Rond	7,5
H	Rond	3

Nous pouvons définir la distance entre deux points par :

$$d(A,B) = |x_B - x_A|$$

1. A l'aide d'un dictionnaire, définir l'ensemble en utilisant comme clé le nom des points, et comme valeur le couple forme, position.

2. Écrire la fonction `distance` qui prend en argument une position (un flottant) et un point de l'ensemble et cet ensemble et retourne la distance entre les deux.

   Exemple :

   ```
   >>> distance(2,'A',ensemble)
   2.5
   ```

3. Écrire la fonction `distance_aux_points` qui prend en argument une position et un ensemble, et renvoie la liste des couples (point, distance).
   Exemple :

   ```
   >>> distance_aux_points(5,ensemble)
   [('A', 0.5), ('B', 4.5), ('C', 2), ('D', 1), ('E', 5), ('F', 3), ('G',
   2.5), ('H', 2)]
   ```

4. En s'inspirant du tri par sélection, écrire la fonction `tri_selection_distance` qui renvoie le tri d'une liste de couples (point, distance).
   Exemple :

   ```
   >>> l=distance_aux_points(5,ensemble)
   >>> tri_selection_distance(l)
   [('A', 0.5), ('D', 1), ('C', 2), ('H', 2), ('G', 2.5), ('F', 3), ('B',
   4.5), ('E', 5)]
   ```

5. Écrire la fonction *forme_max* qui prend une liste de couple et un ensemble et renvoie la forme la plus représentée dans la liste. On pourra s'aider d'un dictionnaire pour mémoriser le nombre de forme rencontrée.

6. En déduire la fonction `knn` qui prend en argument une position, un entier k et un ensemble et renvoie la forme détecté par le principe des k plus proches voisins.

   Exemple :

```
>>> print(knn(5,6,ensemble))
triangle
```

2.2 Dans le plan

Exercice 2 : On cherche à simuler une forêt dans laquelle se trouve différentes essences. Ces essences seront représenté par différentes couleurs. La forêt sera assimilée à un canevas de 600x600, et chaque arbre aura une position aléatoire dans ce canevas (rien n'empêche que deux arbres est la même position...).
La répartition des espèces est supposée équitable, et l'on suppose que l'on dispose de 5 espèces différentes :

- l'espèce rouge.
- l'espèce bleue.
- l'espèce verte.
- l'espèce jaune.
- l'espèce noire.

1. Écrire la fonction `construction_foret` qui prend en entrée un entier n est renvoie une liste de triplet : (essence, abscisse , ordonnée). Les coordonnées étant des entiers entre 0 et 600, choisis aléatoirement.
2. Écrire la fonction `dessin_foret` qui, à l'aide de `Tkinter`, propose une représentation graphique de la forêt.
3. Écrire la fonction `distance_carre` qui renvoie le carre de la distance entre deux points du plan (on donnera les quatre arguments).
4. En s'inspirant de l'exercice précédent, écrire les fonctions `distance_aux_points` et `tri_selection_distance`.
5. Écrire la fonction `couleur_max` qui renvoie la couleur la plus présente dans une liste composée de couples de types arbre, distance.
6. En déduire la fonction `knn` qui renvoie la couleur associée au k plus proches voisins. La fonction prendra comme arguments une abscisse, une ordonnée, une valeur k et une forêt.