

Notation1: On note $\mathcal{P}$ le plan affine euclidien rapporté à un repère.

Soit N, p, q trois entiers naturels non nuls.

Un point de $\mathcal{P}$ sera représenté par une 2-liste de flottants correspondant à ses coordonnées.

Question 1: Écrire une fonction `creation_points` prenant entrée deux entiers naturels non nuls N et p et créant aléatoirement un tableau numpy de p points **deux à deux différents** de $\mathcal{P}$ dont les coordonnées sont à valeurs dans $\llbracket -N, N \rrbracket$. On pourra utiliser la fonction `rd.randint(-N, N)` retournant aléatoirement un entier $\llbracket -N, N \rrbracket$.

Question 2: Donner une inégalité (qu'on supposera ensuite acquise) reliant N et p pour que `creation_points( $N, p$ )` puisse permettre de créer effectivement un tableau et ajouter une clause `assert` à la fonction précédente.

Question 3: Écrire une fonction `creation_lst_caracteres` prenant entrée deux entiers naturels non nuls p, q et retournant une liste de p entiers à valeurs dans $[1, q]$

Question 4: Écrire une fonction `distance` prenant en entrée deux points A et B et retournant le **carré** de la norme euclidienne du vecteur $\overrightarrow{AB}$.

Apprentissage supervisé par la méthode des k plus proches voisins :

On suppose disposer d'un tableau numpy `tab_pts` formé de p points **deux à deux différents** de $\mathcal{P}$ dont les coordonnées sont à valeurs dans $\llbracket -N, N \rrbracket$ et d'une liste `lst_carac` de p entiers dans $[1, q]$. Pour tout $k \in \llbracket 0, p-1 \rrbracket$, `lst_carac[k]` représente la valeur du caractère associé au point `tab_pts[k]`. On dispose également d'un point `pt_nouv` de $\mathcal{P}$ dont les coordonnées sont aussi à valeurs dans $\llbracket -N, N \rrbracket$.

L'algorithme des k plus proches voisins consiste à déterminer les k points de `tab_pts` les plus proches de `pt_nouv`. Une fois, ces k voisins déterminés, on peut attribuer au nouveau point le caractère le plus fréquent parmi les k voisins.

Le caractère attribué à `pt_nouv` dépend de k dont le choix est décisif en pratique.

Cette méthode est une technique d' apprentissage supervisé.

Question 5: Écrire une fonction `creation_lst_dist` prenant en entrée un tableau de points `tab_points` de longueur p , un point `pt_nouv` et retournant la liste `lst_dist` telle que, pour tout $k \in \llbracket 0, p-1 \rrbracket$, `lst_dist[k]` est la distance `pt_nouv` à `tab_points[k]`.

Question 6: Étudier et comprendre la fonction `partage` et en déduire une fonction `tri_rap_ind` prenant en entrée deux listes `l_val`, `l_ind` de même longueur telle que `l_ind` est la liste des indices de la liste de valeur `l_val` et retournant, par méthode du tri rapide, la liste des indices de `l_val` rangés par ordre croissant des valeurs de `l_val`.

```
>>> tri_rap_ind([6, 8, 9, 12, 3, 5, 8, 4, 8], list(range(9)))
[4, 7, 5, 0, 8, 6, 1, 2, 3]
```

1

2

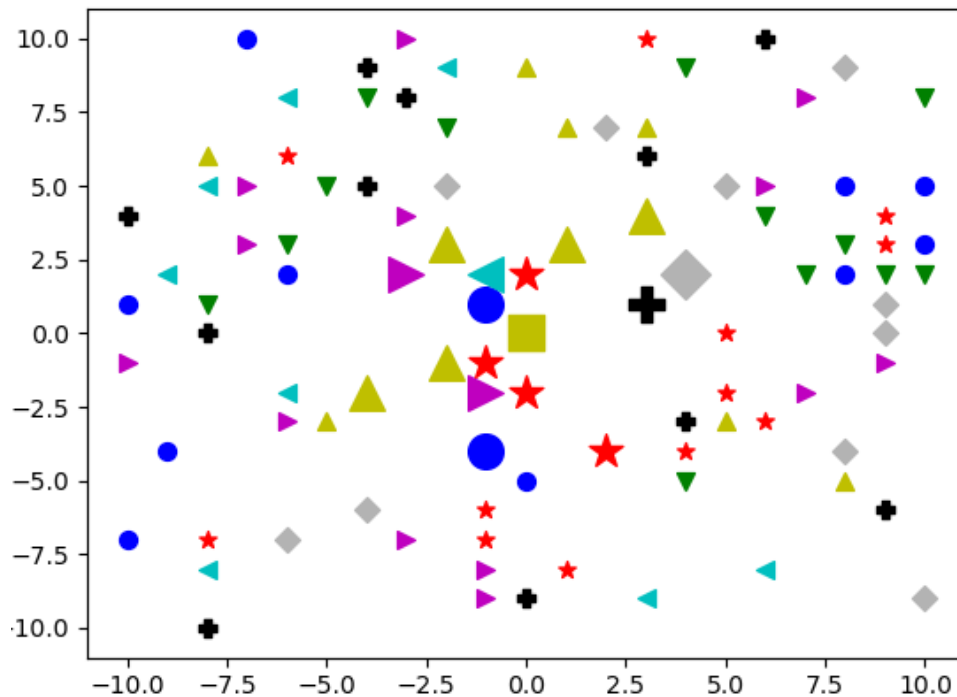
Question 7: Écrire une fonction `carac_dominant` prenant en entrée un entier q , une liste `l_carac`, composée d'entiers de $\llbracket 1, q \rrbracket$ et retournant l'entier (appartenant donc à $\llbracket 1, q \rrbracket$) le plus fréquemment présent dans `l_carac`.

Question 8: Écrire une fonction `methode_k_voisins` prenant en entrée un tableau de points `tab_pts`, sa liste `lst_carac` de caractères associés, l'entier k , le nombre q de caractères, un point nouveau `pt_nouv` et retournant la liste des k plus proches voisins et le caractère qu'il faut attribuer à `pt_nouv` en utilisant la méthode des k plus proches voisins.

Question 9: Compléter la fonction `visualisation` prenant en entrée en entrée un tableau de points `tab_pts`, sa liste `lst_carac` de caractères associés, l'entier k , le nombre q de caractères, un point nouveau `pt_nouv` et permettant d'afficher un graphique tel que :

- Chaque point est marqué et coloré en fonction de son caractère (disque bleu, croix noire, ...).
- Les k voisins sont identifiés par un marqueur "plus gros".
- Le nouveau point est marqué d'un carré et prend la couleur du caractère des voisins les plus nombreux.

Exemple Avec $N = 10$, $p = 100$ (100 points), $k = 16$ (16 voisins), $nb_car = 8$ (8 caractères). Le nouveau point, placé à l'origine $O(0,0)$, est marqué d'un carré et a pris la couleur du caractère des voisins les plus nombreux (ici 5 triangles dorés).



Définition 1: Soit $p \in \mathbb{N}$, $N \geq 2$, $q \in \mathbb{N}^*$. On considère un ensemble de p points tel qu'à chaque point est associé un unique entier compris entre 0 et $q - 1$ représentant sa classe. Pour tout entier naturel k non nul on associe à la méthode des k -voisins une matrice appelée matrice de confusion que l'on notera $M_k = (m_{i,j})_{1 \leq i \leq q, 1 \leq j \leq q} \in \mathcal{M}_p(\mathbb{R})$ tel que, pour tout $(i, j) \in \llbracket 1, q \rrbracket$, $m_{i,j}$ est le nombre d'éléments de la classe i que l'algorithme a étiqueté comme étant de la classe j . En particulier :

- Pour tout $i \in \llbracket 1, q \rrbracket$, $m_{i,i}$ est égal au nombre d'éléments de la classe i qui ont été reconnus par l'algorithme.
- $\sum_{i=1}^q \sum_{j=1, j \neq i}^q m_{i,j}$ est le nombre d'éléments mal classés par l'algorithme.

Question 10: Écrire une fonction `confusion` qui prend en entrée `tab_pts`, sa liste `lst_carac` de caractères associés, l'entier k , le nombre q de caractères et qui renvoie la matrice de confusion M_k précédemment définie.

Question 11: De quel type de matrice doit se rapprocher la matrice de confusion pour que la valeur de k puisse être considérée comme bien choisie.