TP n°1 spécialité informatique

January 9, 2023

Le but de ce TP est de commencer (ou continuer, pour ceux qui avait déjà commencé pendant le cours) à faire l'algorithme k-means en 2 dimensions. Puis d'utiliser une bibliothèque graphique de python (on vous propose matplolib, mais vous pouvez en prendre une autre) afin de visualiser votre algo.

Le langage choisit sera du python (seul choix que l'on vous impose). Pour les autres choix, nos propositions ne sont pas obligatoires et ont pour but de vous conseiller.

1 La représentation des données

Dans un premier temps, nous vous proposons de représenter le nombre de points N, le nombre de cluster K et le nombre de dimensions d dans des variables globales aux début du fichier.

Nous vous proposons de représenter les N points avec 3 tableaux de taille N. Les deux premiers donnant les coordonnées des points selon l'axe des abscisses et des ordonnées et le dernier le numéro du cluster auxquels ils sont assignés (il est possible d'initialiser les éléments du dernier tableau à -1 pour annoncer que les points n'ont pas encore de cluster).

Pour les K clusters, nous vous proposons de les représenter avec 2 tableaux de taille K, la i-ième case du 1er (resp. 2ème) contenant la coordonnées selon l'axe des x (resp. y) du milieu du $i$ème cluster.

2 K-moyennes (k-means)

Le but de cette partie est d'écrire l'algo k-means en 2 d uniquement. Voici des étapes intermédiaires que nous vous proposons pour faire l'algo. On ne se préoccupe pas, pour l'instant, de l'initialisation des points

1. Ecrire une fonction *distance* qui prend comme entrée 2 points (soit 2 couples, soit 4 entiers) et qui renvoie leur distance en 2 dimensions. (Pour ceux qui veulent, il est possible de déjà faire cette fonction en dimensions quelconques.) Cela vous permettra de plus facilement modifier votre algo pour de la dimension 3.

2. Ecrire une fonction *assigne* qui prend comme argument 1 point et les centres des clusters et lui assigne un cluster.

3. Ecrire une fonction *centres* qui prend comme arguments les coordonnées des points et leurs clusters et renvoie les nouveaux centres des clusters.

4. Utiliser ces fonctions pour écrire $k-means$ (en considérant l'initialisation déjà faite).

3 Initialisation général

1. Importer le module *random*.

2. Utiliser des fonctions du module random pour créer votre dataset de manière aléatoire. Nous vous proposons de créer une variable globale M au début de votre fichier et de l'utiliser pour créer des coordonées comprises entre 0 et M avec des fonctions de random vues dans le tronc commun (on pourra les rappeler à l'oral si nécessaire.)

3. Initialiser les clusters avec la méthode de Forgy-MacQueen (choix aléatoire avec probabilité uniforme des K points parmis ceux de notre dataset).

4. (Bonus: Nous vous conseillons de faire cela plus tard, après la dernière partie) Initialiser les clusters avec $k-means++$ ou Forgy-MacQueen itéré.

4 Visualisation

1. Importer *matplotlib.pyplot* (as plt, pour vous faciliter la vie).

2. (Facultatif)Vers le début de votre code, utiliser la commande $plt.ion()$ (mode interactif).

3. A chaque itération de $k-means$ utiliser $plt.scatter(argument1, argument2, etc..)$ avec les bons arguments puis $plt.show()$ pour afficher l'étape actuelle, puis $plt.clf$ pour effacer la fenêtre.

4. Conseil: utiliser $plt.time(.2)$ juste avant d'effacer la fenêtre.