Spécifications du projet

Librairies autorisées

- numpy
- matplotlib

Installation: *pip install numpy matplotlib*

Lecture des données

Les données à traiter sont au format csv, et sont de la forme suivante:

```
x,y
34.938,-1.013
50.385,-5.659
-38.268,-29.443
-36.296,21.152
26.568,41.072
41.244,41.389
```

Les deux colonnes sont les coordonnées en x et y des points. Il y a autant de lignes qu'il y a de points. Il vous est fourni 3 fichiers de données:

- mock_2d_data.csv
- 2d_data.csv
- 3d_data.csv

Le premier contient un faible nombre de données factices, le but étant d'utiliser ce fichier lorsque vous mettez en place l'algorithme. Le deuxième contient des données réelles, et sera le fichier à utiliser afin de produire le fichier de sortie faisant partie de votre évaluation. Le dernier fichier est en bonus, et contient des données réelles en 3 dimensions (il y a une colonne 'z' en plus).

Algorithme

Le nombre de clusters est à choisir lorsque l'on lance l'algorithme *k-means*, pour les données factices vous pourrez tester avec 4 clusters, pour les données réelles il vous sera imposé de travailler avec 6 clusters.

Il sera apprécié que vous évaluiez la qualité de votre *k-means* en calculant la moyenne des distances entre chaque point du jeu de données et le centroïde que vous lui avez assigné. Plus cette moyenne est faible, mieux c'est. Attention à la phase d'initialisation des centroïdes, elle conditionne beaucoup le résultat final !

Fichier de sortie

Outre le code que vous aurez produit afin d'implémenter l'algorithme *k-means*, il vous est demandé de sauvegarder les résultats de l'algorithme dans un fichier csv, ayant la forme suivante:

```
x,y,centroid_x,centroid_y
34.938,-1.013,35.41422,-3.2503
50.385,-5.659,35.41422,-3.2503
-38.268,-29.443,-43.61103,-41.05197
-36.296,21.152,-31.60671,7.05349
26.568,41.072,38.35683,47.0298
41.244,41.389,38.35683,47.0298
```

On a ainsi rajouté deux colonnes contenant les coordonnées en x et en y des centroïdes associés à chaque point du fichier source. Notez qu'il ne vous est pas imposé de conserver l'ordre des points du fichier source, le nouveau fichier peut très bien contenir ces points dans un ordre différent, tant qu'ils sont tous présents et que chacun a bien été assigné à un unique centroïde.

Affichage des données

Vous pouvez afficher les données à l'aide d'un bout de code qui vous est fourni, présent dans *drawing.py* . La fonction *draw2D()* attend en entrée des données sous la forme d'une matrice de taille *n* x 2 ou *n* x 4, où *n* est le nombre de points. Chaque ligne de la matrice doit contenir une liste de coordonnées de la forme [x, y] ou [x, y, centroid_x, centroid_y], suivant que vous souhaitez afficher les données avant ou après application de l'algorithme *k-means*. Enfin, dans le dernier cas vous pouvez jouer avec l'option *drawLinks* pour afficher les liens entre les points et leur centroïde.

Bonus

1. Vous pouvez, en plus du *k-means* en 2 dimensions, le réaliser en 3D sur les données présentes dans *3d_data.csv*
2. Si votre k-means fonctionne en 3D, vous pouvez tenter d'afficher le découpage en clusters en 3D à l'aide de matplotlib.
3. Vous pouvez aussi (pour les données en 2D ou 3D) tracer le graphe de la distance moyenne après découpage en clusters évoquée plus haut, en fonction du nombre de clusters que vous ferez varier.