Spécifications du projet

Librairies non standard autorisées

➔ numpy, matplotlib, pyqtgraph

Installation

Pour Linux: *sudo -E apt-get install python-opengl python3-pyqt5 pyqt5-dev-tools qttools5-dev-tools*

Et dans tous les cas: *pip install numpy matplotlib pyqtgraph pyopengl PyQt5*

Lecture des données

Les données à traiter sont au format csv, et sont de la forme suivante (pour la 2D):

```
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. Les autres contiennent des données réelles en 2D et 3D (il y a une colonne 'z' en plus), et seront à utiliser afin de produire les fichiers de sortie faisant partie de votre évaluation.

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 10 clusters.

Il est aussi demandé que vous évaluiez la qualité de votre *k-means* en calculant la statistique $\frac{1}{n}\sum_{i=1}^{n} d(P_i, C(P_i))$, qui est la moyenne des distances entre chaque point P_i du jeu de données et le centroïde $C(P_i)$ que vous lui avez assigné. Plus ce score est faible, mieux c'est.

Attention à la phase d'initialisation des centroïdes, elle conditionne beaucoup le résultat final !

Fichiers de sortie

Outre le code que vous aurez produit afin d'implémenter l'algorithme *k-means*, il vous est demandé de sauvegarder les meilleurs résultats trouvés par l'algorithme dans des fichiers csv, ayant la forme suivante (pour la 2D):

```
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 n'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. Enfin il faudra rajouter une colonne 'z' et 'centroid_z' pour la 3D, cf. plus bas.

Les fichiers produits doivent être remis avec le reste de votre travail, au bon format, et compteront dans la notation.

Affichage des données

Vous pouvez afficher les données à l'aide d'un bout de code qui vous est fourni, présent dans *drawing.py* . Une ancienne version nommée *old_drawing.py* est aussi fournie, mais ne fonctionne que pour la 2D. Les fonctions d'affichage attendent en entrée des données sous la forme d'une matrice de taille n x d ou n x $2d$, où n est le nombre de points et d la dimension des données. Chaque ligne de la matrice doit contenir une liste de coordonnées au même format que dans les fichiers de sortie, i.e. de la forme [x, y] ou [x, y, centroid_x, centroid_y] pour la 2D, et [x, y, z] ou [x, y, z, centroid_x, centroid_y, centroid_z] pour la 3D, suivant que vous souhaitez afficher les données avant ou après application de l'algorithme *k-means*.

Bonus

1. Vous pouvez (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.
2. Vous pouvez aussi, si le nombre d'itération n'est pas trop grand, réaliser une animation montrant le découpage des données en clusters au cours du temps.