machine learning : K-means

Ben Amira Rawia

23 novembre 2022

Université de Toulon, UFR Sciences et Techniques

M1 DID

2022-2023

Table des matières

1 Introduction

Ce rendu est elaboré dans le cadre du module I143 pour le TP K-means.

2 EXERCICE1

2.1 calcul distance

Pour calculer la distance entre un point et un centroide, plusieurs methodes existent. Dans mon code j'ai implementé deux methodes classiques la distance euclidienne ainsi que la distance de manhattan. Je n'ai pas remarqué de difference dans les resultats obtenues avec les deux methodes meme si que dans la litterature apparament la distance euclidienne et la mieux approprié pour des donnees sous forme de coordonnees spatiale.

2.2 initialisation des centroides

Ma premiere demarche pour calculer les centroides etait de generer des nombres aleatoires entre le maximum et le minimum des donnees, mais vite j'ai changer de methode car une exception est kevée suite à une division par zero si un cluster est vide. la deuxieme demarche etait de generer un entier x aleatoirement entre 0 et la longeur de la vx, et de considerer que vx[x] est un centroide tiré aleatoirement.

2.3 assign cluster

Cette fonction va renvoyer une matrice contenants les points de chaque cluster. par exemple pour k=3 la matrice contiendra dans matrice[0] les points les plus proches du premier centroide etc.

2.4 compute centroids

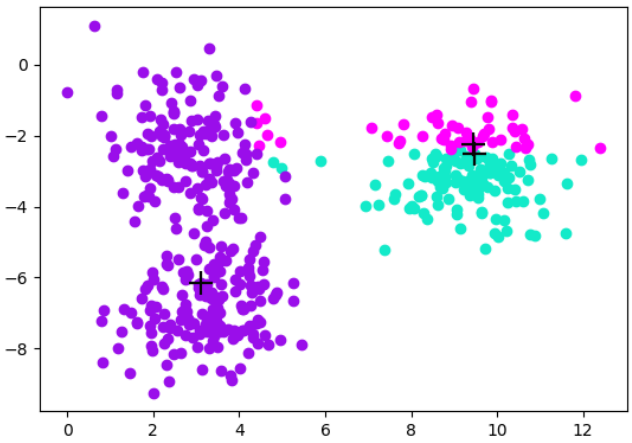
Cette fonction va contenir comme parametres k le nombre de clusters, vx les donnees, vc les centroides tirés aleatoirement et enfin un nouveau parametre a été ajouté qui est le nombre d'iterations. en effet ce dernier permet d'arreter le calcul des nouveaux centroides.

2.5 display

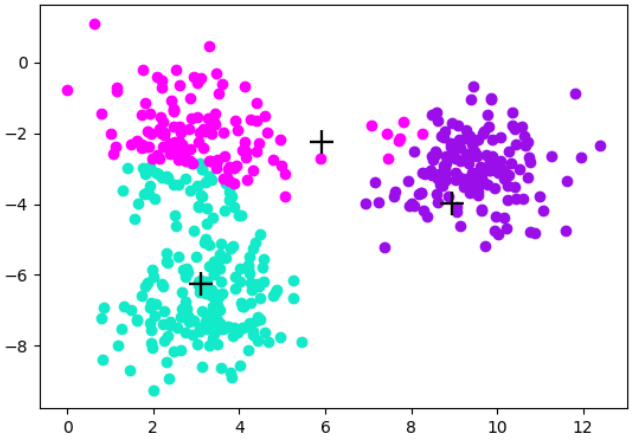
Cette fonction fait tous les affichages necessaires.La fonction display 2 data a ete supprimé et l'affichage se fait une fois pour toute dans cette fonction.

2.6 Resultats et analyse

Les premiers centroides generés sont :

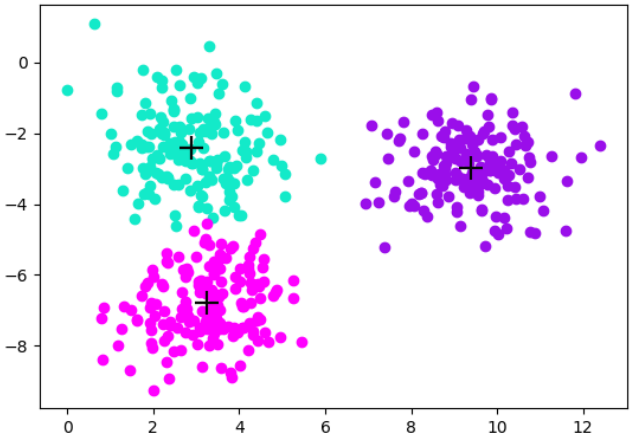


les croix representent les centroides tirés aleatoirement au debut de l algorithme ainsi que leurs clusters correspondant. Ensuite je decide de recalculer les centroides en deux autres iterations et j'obtient ces resultats :



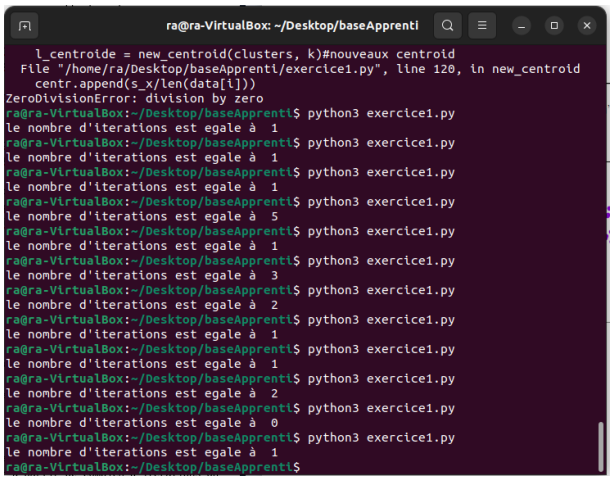
les donnees sont visiblement mieux repartis que dans le cas precedent. en effet j'ai fait le choix de programmer avec un nombre d'iterations qu'on peut fixer pour voir à partir de combien d'iterations on converge. Dans une ancienne version j'avais la version qui recalculer les centroides jusqu'a egalité des centroides entre deux iterations et puisque on est sur des flottants ceci prenait beaucoup de temps.

En testant avec nombre d'iterations egale à 5 j'obtient ce resultat final qui est bien parlant et qui represente le resultat attendu :



En reflechisant encore plus sur la convergence de l'algorithme j'ai ajouté une fonction qui qui arrete le recalcul des clusters jusqu'a obtention de difference entre les centroides egales a une valeur prise en parametres et pour pousser mes idées j'ai aussi ajouté une variable type compteur qui s'incremente à chaque nouveau calcul pour savoir apres combien de tour on obtient bien une convergence.

En lancant le programme à plusieurs reprises, j'obient ces nombres d'iterations :

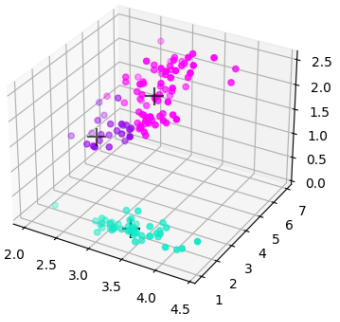


ensuite dans une derniere version j'ai ajouté le fait que cette fonction peut à la fois arreter le calcul jusqu'a convergence et en meme temps elle nous affiche le nombre d iterations pour arriver à convergence.

3 EXERCICE2

Dans l exercice 2 on reprend les memes concepts juste on ajoute une coordonnée z à x et y. Certaines fonctions implementées dans l exercice 1 ont été applées et d'autres ont été adaptées pour traiter des données à 3 features.

kmeans3d(k, conv) reprend le meme concept que kmeans(k, conv) juste les donnees sont à 3 features et l'affichage est ainsi :

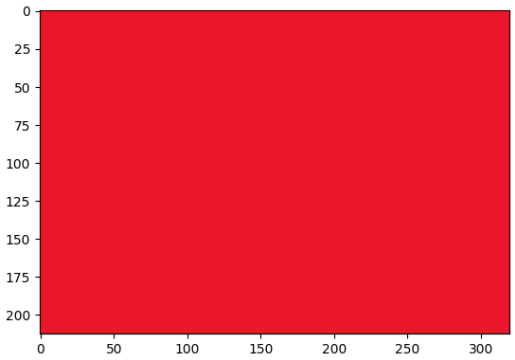


4 EXERCICE3

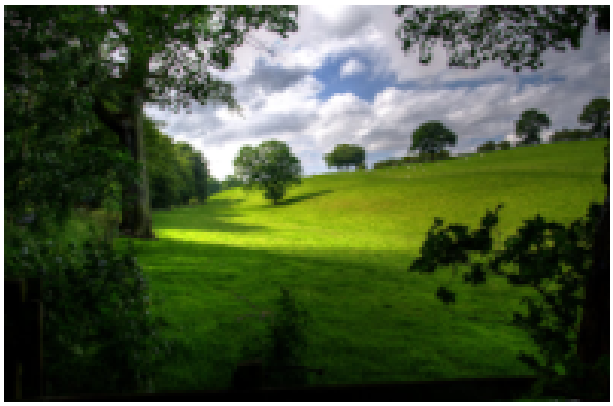
Dans cet exercice on modifie des images png et ceci en agissant sur les codes RGB. Des problemes de division par zero se sont posés lors du traitement par exemple j'ai tenté de changer l image du drapeau de la Tunisie (drapeau rouge et blanc) avec un clustering à 4 et j'obtient erreur de division par zero ce qui est totalement logique en fait tous les points seront forcement attribué au cluster 1(blanc) ou cluster 2 (rouge) et les deux autres seront à vide.

Drapeau de la tunisie :

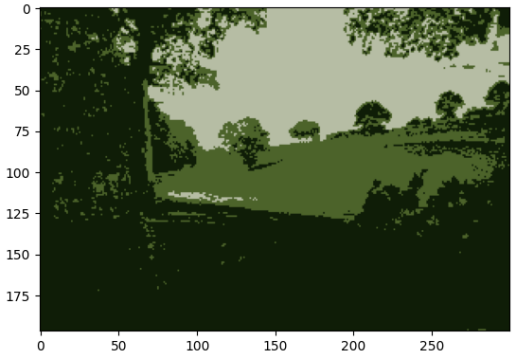
Pour k=1 on obtient cet affichage tout logique :



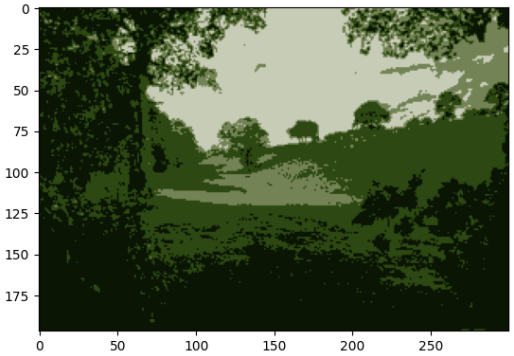
les images fournies avec le tp :
L'original :



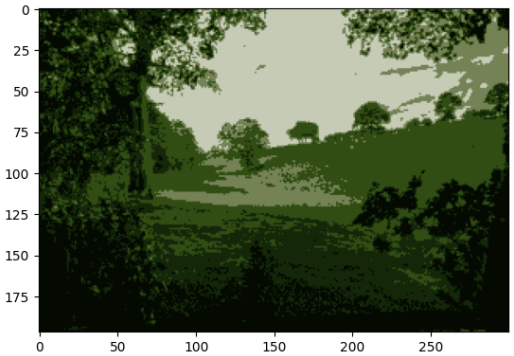
K=3



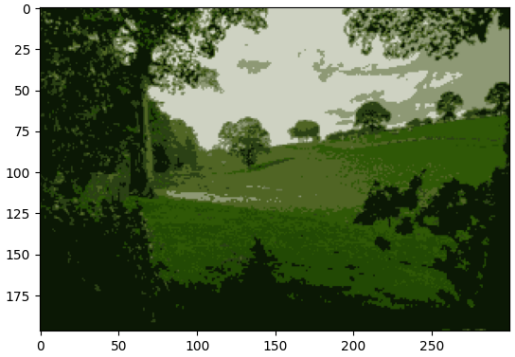
K=4


0
25
50
75
100
125
150
175
0
50
100
150
200
250

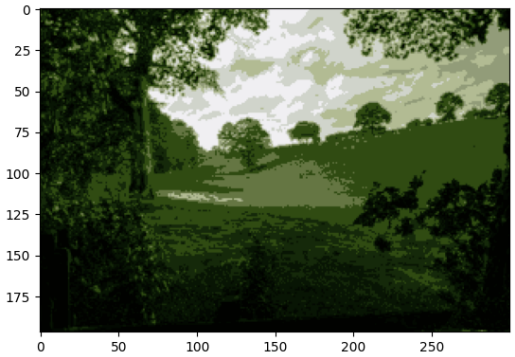
K=5


0
25
50
75
100
125
150
175
0
50
100
150
200
250

K=8



K=10



autre exemple

k=5

k=2

5 Conclusion

k-means est un algorithme de clustering tres repondue, son implementation sur python ne pose pas vraiment de soucis et ceci grace à numpy. Avoir les affichages dans le 3eme exercice aide beaucoup à la comprehension de l'algorrithme en detail.