TP ANREC k-means

I. L'algorithme des k-moyennes

k-means methode : *k* fixé = nombre de groupes

1. Initialization
On choisit k représentatants initiaux parmis les individus qui devront être regroupés.

2. Reallocation
On alloue chaque individu au groupe (cluster) auquel il est le plus similaire (fonction de similarité à définir).

3. Recentering
On redéfinit les représentants des groupes (par exemple, centre de gravité des groupes définis en 2)).

4. Repeat
On répète les étapes 2) et 3) jusqu'à stabilisation.

II. Commentaires sur le code

Pour exécuter le programme, on écrit la commande, avec les arguments suvants :

./a.out k dim_data seed type_similarité n_lines_skipped n_col_skipped < in_file > out_file

Avec :
k : nb de représentants
dim_data : dimension des données
seed : pour fixer la réalisation des nombres aléatoires tirés
type_similarité : on propose plusieurs types de fonctions de similarité (3 normes, cf. paragraphe suivant).
n_lines : dans le fichier de données, nombre de lignes à sauter pour accéder aux données
n_col : dans le fichier de données, nombre de lignes à sauter pour accéder aux données
in_file : fichier d'entrée
out_file : fichier de résultats, ils sont affichés sous la forme d'une liste des points pour chaque cluster, séparé par une ligne vide.

Exemple : si l'on souhaite faire :
- 3 groupes
- avec des données de dimension 2
- avec le seed 1
- en utilisant la fonction de similarité 2
- en sautant 0 lignes et 0 colonnes
- dans le fichier d'entrée "exemple1.txt"
- et en écrivant les résultats dans out.txt, cela donne :

./a.out 3 2 1 2 0 0 <exemple1.txt > out.txt

- On peut contrôler la réalisation du hasard à l'aide de l'argument seed que l'on passe à la fonction main.

- Pour la fonction de similarité, nous avons utilisé 3 normes :

- la norme 1 qui est donnée par la somme des modules des coefficients
- la norme euclidienne, ou norme 2 : qui est la somme des modules au carré des coefficients
- la norme infinie : qui est la limite de la somme des modules puissance p des coefficients, p tendant vers l'infini

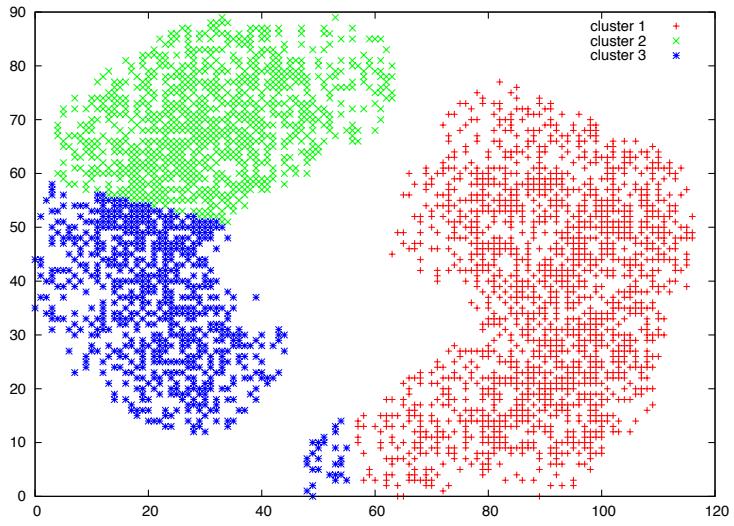
Nous pouvons ajouter et définir d'autres fonctions de similarité au début du fichier main.cpp, puis la passer en paramètre de la fonction get_means dans le main.

III. Graphes et commentaires

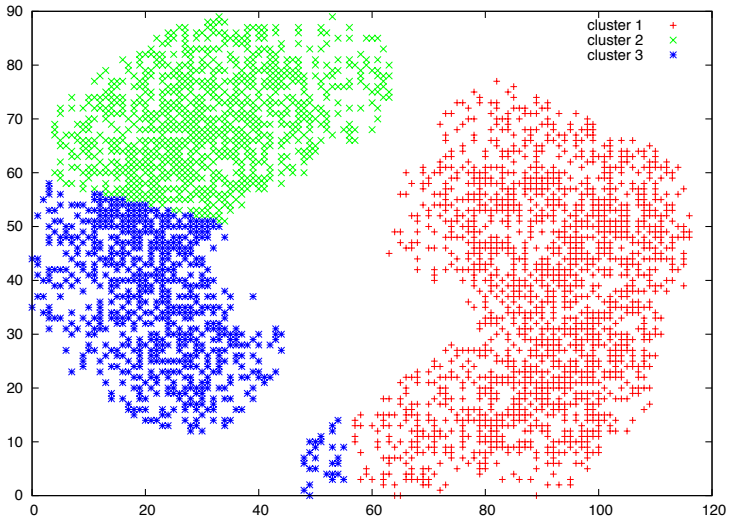
3.1. Influence de la norme

Nous remarquons que la formation des clusters dépend peu de la norme choisie :

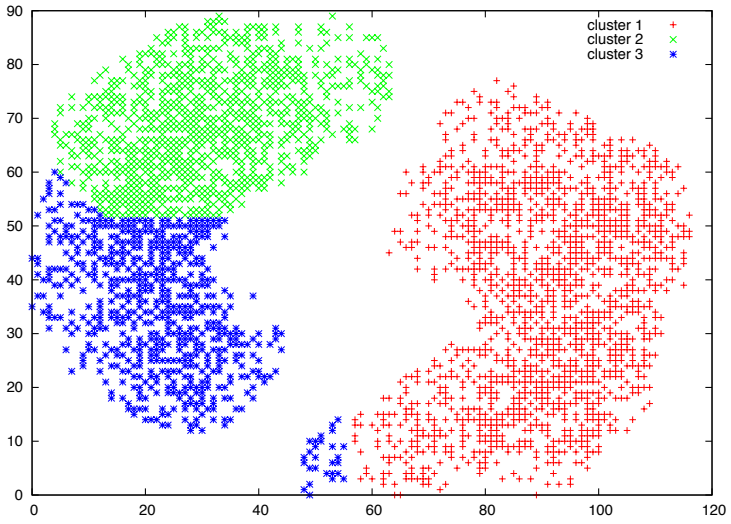
Par exemple, avec le fichier de données exemple1.txt :
En prenant la **norme 1**, nous avons les clusters suivants :



En choisissant la **norme 2**, avec le même choix de représentants initiaux (étape 1), nous avons les clusters suivants :



enfin, en prenant la **norme infinie**, et le choix des k représentants avec seed =1, nous avons les clusters suivants :

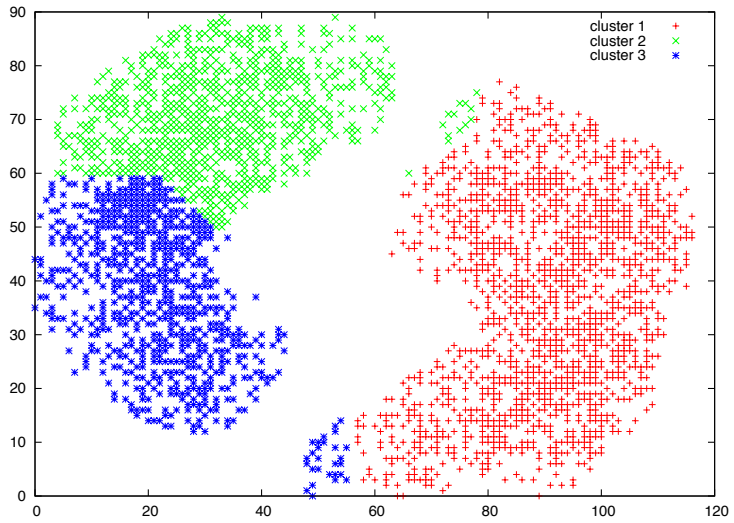


3.1. Influence du choix du hasard

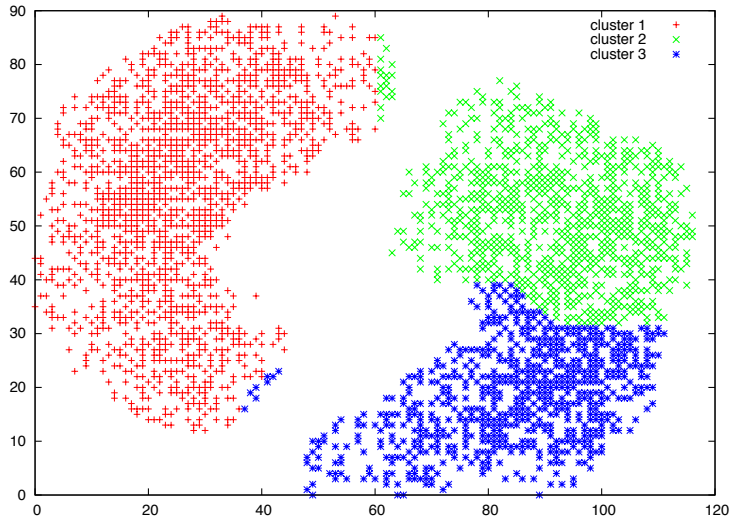
Mais la formation des clusters peut dépendre fortement du choix du hasard :

Par exemple, avec le fichier de données exemple1.txt et deux réalisations différentes du hasard :

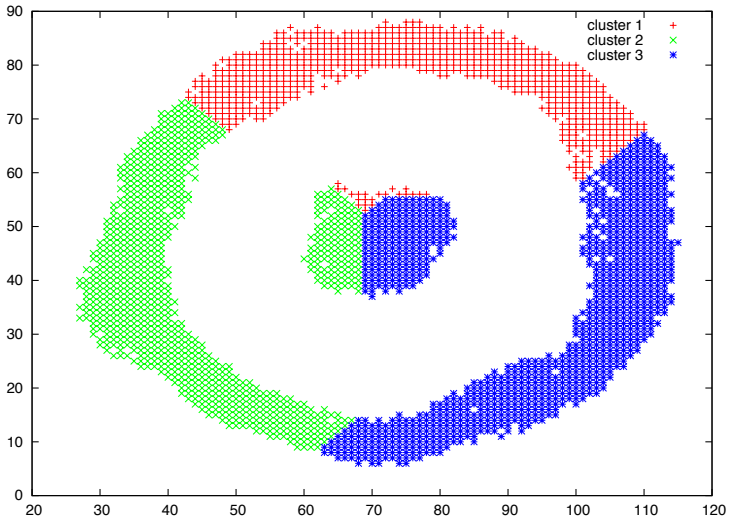
en prenant la norme 1, et avec une réalisation du hasard donnée, nous avons les clusters suivants :



Mais si, toutes choses étant égales par ailleurs, nous choisissons une autre réalisation du hasard, nous obtenons les clusters suivants :



Pour le fichier exemple2.txt, voici les résultats avec la norme infinie, par exemple :


cluster 1
cluster 2
cluster 3
90
80
70
60
50
40
30
20
10
0
20
30
40
50
60
70
80
90
100
110
120