Atelier d'optimisation :

K-means Clustering

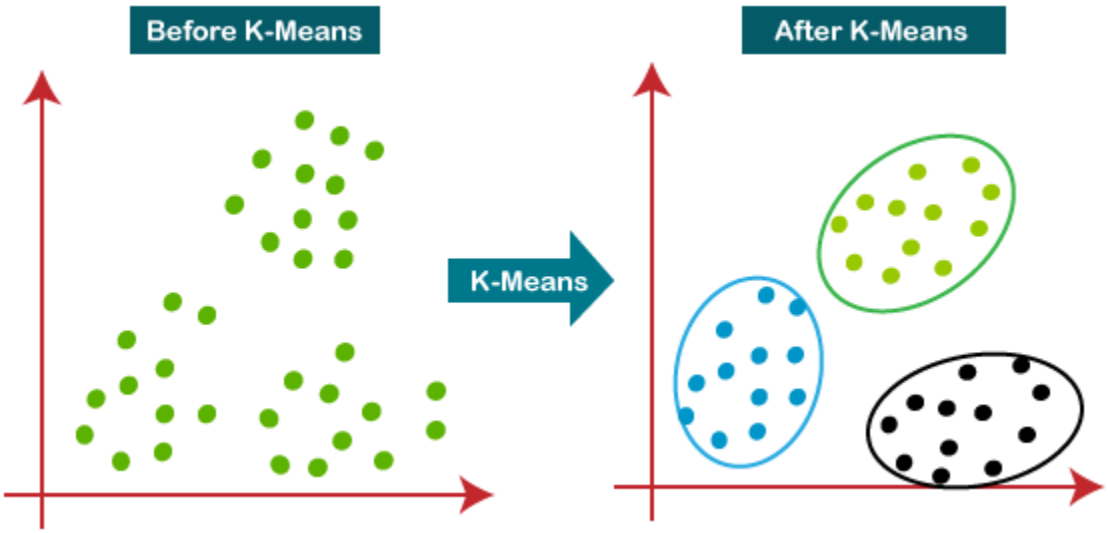
Réalisé par :

Abdelkader MILADI

INDP2 F

Qu'est ce que K-means

K-means est un algorithme non supervisé de clustering **non hiérarchique**. Il permet de regrouper en clusters distincts les observations du data set. Ainsi les données similaires se retrouveront dans un même cluster.



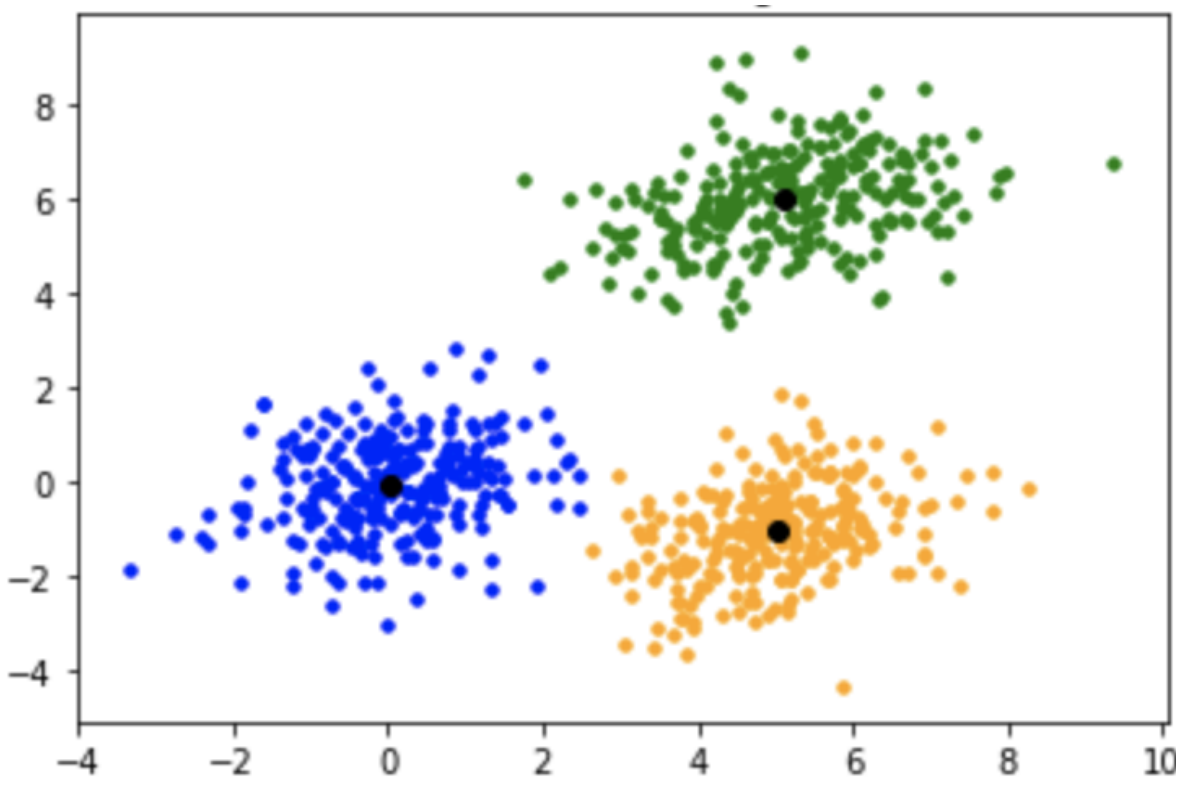
Principe algorithmique :

Algorithme K-means
Entrée :

- K le nombre de cluster à former
- Le Training Set (matrice de données)

DEBUT
Choisir aléatoirement K points (une ligne de la matrice de données). Ces points sont les centres des clusters (nommé centroïd).

REPETER
Affecter chaque point (élément de la matrice de donnée) au groupe dont il est le plus proche au son centre
Recalculer le centre de chaque cluster et modifier le centroïde
JUSQU'A CONVERGENCE
OU (stabilisation de l'**inertie totale** de la population)
FIN ALGORITHME



Description

- Choisir *k* points qui représentent la position moyenne des partitions $\mathbf{m}_1^{(1)}$, …, $\mathbf{m}_k^{(1)}$ initiales (au hasard par exemple)
- Répéter jusqu'à ce qu'il y ait convergence :
  – affecter chaque observation à la partition la plus proche :

$$S_i^{(t)} = \left\{ \mathbf{x}_j : \left\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \right\| \leq \left\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \right\| \forall\, i^* = 1, \ldots, k \right\}$$

– mettre à jour la moyenne de chaque cluster :

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

Avantages de l'algorithme :

1) L'algorithme de k-means est très populaire du fait qu'il est très facile à comprendre et à mettre en œuvre.

2) Sa simplicité conceptuelle et sa rapidité

3) Applicable à des données de grandes tailles, et aussi à tout type de données (mêmes textuelles), en choisissant une bonne notion de distance.

Inconvénients de l'algorithme :

1) Le nombre de classe doit être fixé au départ,

2) Le résultat dépend de tirage initial des centres des classes,

3) Les clusters sont construits par rapports à des objets inexistants (les milieux)

Cas d'utilisation K-means

K-Means en particulier et les algorithmes de clustering de façon générale ont tous un objectif commun : Regrouper des éléments similaires dans des clusters. Ces éléments peuvent être tous et n'importe quoi, du moment qu'ils sont encodés dans une matrice de données.

Les champs d'application de K-Means sont nombreux, il est notamment utilisé en :

- la segmentation de la clientèle en fonction d'un certain critère (démographique, habitude d'achat etc….)
- Utilisation du clustering en Data Mining lors de l'exploration de données pour déceler des individus similaires. Généralement, une fois ces populations détectées, d'autres techniques peuvent être employées en fonction du besoin.
- Clustering de documents (regroupement de documents en fonction de leurs contenus. Pensez à comment Google Actualités regroupe des documents par thématiques.)

Programmation

```
import math

#Calculate the distance between two points
def distance(p1,p2):
    return math.sqrt((p2[0]-p1[0])**2+(p2[1]-p1[1])**2)
```

```
def getClusters(points,centroid):
    clusters={}
    for p in points:
        d_old=-1
        for c in centroid:
            d_new=distance(p,centroid[c])
            if d_new<d_old or d_old==-1:
                clusters[p]=c
            d_old=d_new
    return clusters
```

```
#Initiation
points=[];clusters={};centroid={};result={}
n_p=int(input("Enter number of points in the data set : "))

for i in range (0,n_p):
    points.append(tuple(float(i) for i in input("Enter coordinates of point "+str(i+1)+" : ").split()))
n_c=len(points)+1

while(n_c>len(points)):
    n_c=int(input("Enter number of clusters to be created (<"+str(len(points))+"): "))
    if n_c > len(points):
        print("Sorry.The maximum number of clusters that can be made is ",len(points))

for i in range(0,n_c):
    centroid[i]=points[i]
clusters=getClusters(points,centroid)
```

```
c_old={}
while c_old != clusters:
    c_old=clusters
    for i in range(0,n_c):
        x=0;y=0;count=0
        for p in points:
            if clusters[p]==i:
                x+=p[0];y+=p[1];count+=1
        x/=count;y/=count
        centroid[i]=(x,y)
    clusters=getClusters(points,centroid)

for key, value in sorted(clusters.items()):
    result.setdefault(value, []).append(key)

for i in result:
    print("\nCluster ",i+1," -> ",result[i], "with Centroid = ",centroid[i])
```