Clustering ou Regroupement des données

L'algorithme Kmeans

1. Présentation générale

Clustering est une tâche basique de Data Mining qui consiste à regrouper des données en *K* clusters ou groupes, tel que *K* est connu au préalable

Dans l'état de l'art, il y a plusieurs algorithmes qui permettent d'effectuer le clustering tels que kmeans, modèles de mélanges gaussiens (GMM), ….

Dans ce document, nous présentons l'algorithme kmeans et nous l'implémentons en Python.

2. Présentation de kmeans

Dans cette section, nous présentons le formulation mathématique de kmeans surtout le modèle mathématique des clusters et nous décrivons l'algorithme de kmeans qui estime les paramètres de ce modèle.

2.1.Modèle mathématique

Nous représentons les objets input et output de kmeans mathématiquement :

- **Modèle des données**
  Les données sont définies par $S = \{x_0, x_1, \dots x_n, \dots, x_{N-1}\}$
  - ➔ *S* est un ensemble de points de données $x_n \in \mathbb{R}^{dim}$, càd tous les points x_n ont la même dimension *dim* et appartiennent au même espace vectoriel $\mathbb{R}^{dim}$
  - ➔ Une représentation mathématique plus pratique consiste à représenter les données sous forme d'une matrice *X* de dimension $NXdim$ et dont les lignes sont les points x_n
- **Modèle des clusters**
  Chaque cluster est identifié par un entier $i \in \{0, \dots, K-1\}$.
  Chaque cluster est mathématiquement représenté par un point spécial de qui correspond à la moyenne arithmétique du cluster.
  - ➔ Les *K* moyennes sont définies par l'ensemble des points $Moy = \{m_0, m_1, \dots m_i, \dots, x_{K-1}\}$
  - ➔ *Moy* est un ensemble de points $m_i \in \mathbb{R}^{dim}$
  - ➔ m_i appartient au même espace vectoriel que x_n
  - ➔ Pour des raisons pratiques, on préfère représenter les moyennes ensemble sous forme d'une matrice *M* dont les lignes sont m_i

$$m_i = \frac{1}{N_i} \sum_{x_n \in S_i} x_n$$

Tels que :

$$S = \bigcup_{i=0}^{K-1} S_i$$

Et

$$\bigcap_{i=0}^{K-1} S_i = \emptyset$$

Et

$N_i = |S_i|$: nombre de points x_n qui appartiennent au cluster S_i

2.2.Algorithme

Etape 1 : Initialisation

K=2, *t* = 0, *Convergence* = *False* , $\tau = 0.001$

Choisir aléatoirement *K* points de *X* comme *K* moyennes initiales $m_i(0)$

Etape 2 : Estimer itérativement *K* moyennes

Tant que $(t < T)$ **and** (Convergence = False) **faire :**

Etape 2.1 : Déterminer les clusters (groupes) des points de données

Pour chaque x_n dans *X* **faire :**

Pour chaque $m_i(t)$ dans $M(t)$ **faire :**

Calculer la distance euclidienne entre x_n et $m_i(t)$:

$d_{ni}(t) = \|x_n - m_i(t)\| = distance\_euclidienne(x_n, m_i(t))$

Fin Pour

Déterminer l'identité du cluster $l_n(t)$ auquel x_n appartient ($l_n(t) \in \{0, \dots, K-1\}$)

$l_n(t) = \underset{i}{\operatorname{argmin}}\, d_{ni}(t)$ Ceci implique que $x_n \in S_{l_n}(t)$

Fin Pour

➔ Etape 2.1 donne une répartition des points dans les clusters
➔ On peut représenter les appartenances des *N* points de *X* aux clusters sous forme d'un vecteur $L(t)$

$L(t) = \{l_0(t), \dots, l_n(t), \dots, l_{N-1}(t)\}$

Etape 2.2 : Calculer les *K* moyennes des clusters obtenus

Pour chaque m_i dans *M* **faire :**

$m_i(t+1) = \frac{1}{N_i(t)} \sum_{[l_n(t)=i]} x_n$

Fin Pour

Etape 2.3 : Vérification de la convergence

$F(t) = \sum_{i=0}^{K-1} \sum_{[l_n(t)=i]} \|x_n - m_i(t)\|^2$

Si $(t > 0)$ **and** $(|F(t) - F(t-1)| < \tau)$ **Alors:**

Convergence = True

Sinon

t=t+1

Fin Si

Fin Tant que

3. Implémentation avec Python en utilisant Numpy et Matplotlib

3.1.Pré-requis de Numpy et Matplotlib

- Création d'une matrice

```
import numpy as np
```

```
X=np.array([[1,4],[4,1],[24,10],[22,12]])
X
```

```
array([[ 1,  4],
       [ 4,  1],
       [ 2,  3],
       [ 3,  2],
       [24, 10],
       [22, 12],
       [20, 10],
       [24, 14]])
```

➔ Shape d'une matrice

```
N,dim=X.shape
print("N=",N," ,dim=",dim)
```

```
N= 4  ,dim= 2
```

- Création d'une séquence

```
N=4
idx=np.arange(0,N,1)
idx
```

```
array([0, 1, 2, 3])
```

- Sélection des lignes d'une matrice étant donné leurs indices

```
X=np.array([[1,4],[4,1],[24,10],[22,12]])
idx=np.array([1,2])
M=X[idx,:]
M
```

```
array([[ 4,  1],
       [24, 10]])
```

- Sélection des lignes d'une matrice étant donné une condition

```
X=np.array([[1,4],[4,1],[24,10],[22,12]])
L=np.array([0,0,1,1])
X0=X[L==0]
X0
```

```
array([[1, 4],
       [4, 1]])
```

- Produit scalaire de 2 vecteurs

```
x=np.array([1,1])
m=np.array([1,4])
d2=np.dot(x-m,x-m)
d2
```

```
9
```

- Distance euclidienne de 2 vecteurs

```
x=np.array([1,1])
m=np.array([1,4])
d2=np.dot(x-m,x-m)
d=np.sqrt(d2)
d
```

```
3.0
```

- Création d'une matrice ou vecteur initialisé€ à 0

```
N=4
L=np.zeros(N)
L
```

```
array([0., 0., 0., 0.])
```

- Création d'une matrice ou vecteur initialiséۥ à 1

```
N=4
L=np.ones(N)
L
```

```
array([1., 1., 1., 1.])
```

- Calcul de la moyenne des lignes d'une matrice

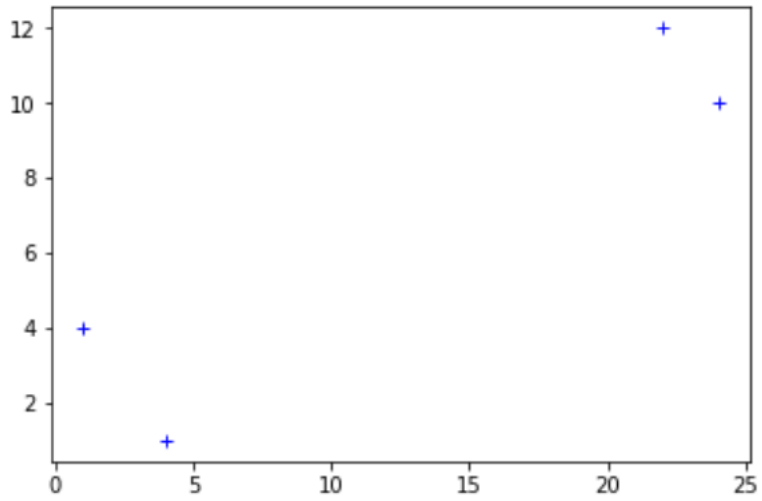
```
X=np.array([[1,2],[3,4]])
m=np.mean(X,axis=0)
m
```

```
array([2., 3.])
```

- **Visualisation des points**

```
import matplotlib.pyplot as plt
X=np.array([[1,4],[4,1],[24,10],[22,12]])
plt.plot(X[:,0],X[:,1],'+',color='b')
```

```
[<matplotlib.lines.Line2D at 0x2e9c0d2f860>]
```



➔ On peut combiner le symbole et la couleur dans le plot :

```
plt.plot(X[:,0],X[:,1],'+b')
```

3.2. Implémentation de kmeans

Etape 1 : Définir une fonction qui permet d'initialiser aléatoirement K moyennes à partir d'une matrice des points X

```
def init_moyennes(X,K):
```

Etape 2 : Définir une fonction qui permet de calculer la distance euclidienne entre deux points x et m

```
def distance_euclidienne(x,m):
```

Etape 3 : Définir une fonction qui permet de déterminer le cluster d'un point de données x à partir d'une matrice des moyennes M

```
def determiner_cluster(x,M):
```

➔ Utiliser la fonction distance_euclidienne()

Etape 4 : Définir une fonction qui permet de déterminer les clusters d'une matrice de points X étant donné une matrice des moyennes M

```
def determiner_clusters(X,M):
```

➔ Utiliser la fonction determiner_cluster()

Etape 5 : Définir une fonction qui permet de calculer les moyennes étant donné une matrice des points X, les identités des clusters des points L et le nombre de clusters K

```
def calculer_moyennes(X,L,K):
```

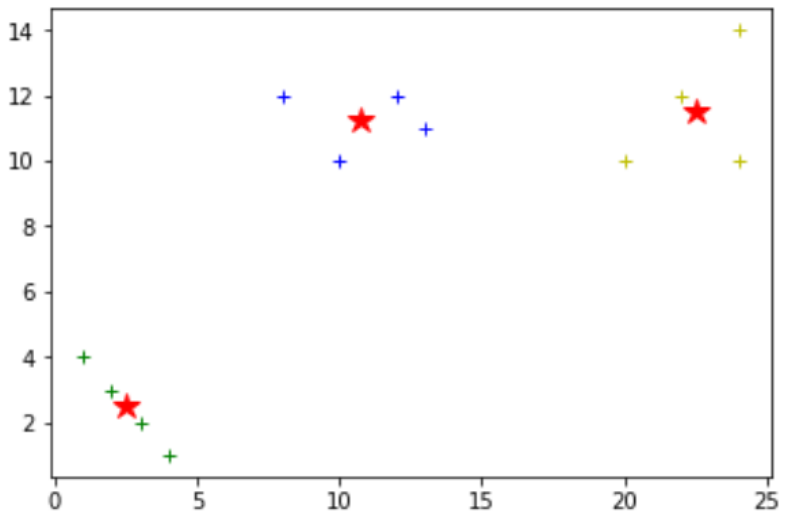
Etape 6 : Ecrire le code de kmeans en utilisant les fonctions ci-dessus

```
import numpy as np
X=np.array([[1,4],[4,1],[2,3],[3,2],[24,10],
            [22,12],[20,10],[24,14],[10,10],[8,12],[12,12],[13,11]])
K=3
M=init_moyennes(X,K)
for t in np.arange(0,10,1):
    L=determiner_clusters(X,M)
    M=calculer_moyennes(X,L,K)

print('M=',M)
print('L=',L)
```

```
M= [[ 2.5   2.5 ]
 [10.75 11.25]
 [22.5  11.5 ]]
L= [0. 0. 0. 0. 2. 2. 2. 2. 1. 1. 1. 1.]
```

Etape 7 : Visualiser les clusters obtenus par kmeans

```
import matplotlib.pyplot as plt
c=['g','b','y','k']
for i in np.arange(0,K,1):
    plt.plot(X[L==i,0],X[L==i,1],'+',color=c[i])

plt.plot(M[:,0],M[:,1],'r*',markersize=12)
plt.show()
```



Etape 8 : Compléter **l'étape 2.3.** « Vérification de la convergence » de l'algorithme de kmeans dans le code de **l'étape 7**