Master Informatique Spécialités EID2
Apprentissage Statistique
TP2
Clustering à la base de Prototype

1 k-Moyennes

Le partitionnement en K-moyennes (ou K-means en anglais) est une méthode de partitionnement de données et un problème d'optimisation combinatoire. K-means travaille sur des objets représentés par des vecteurs. Soit $X = \{x^1, ..., x^N\}$ l'ensemble des vecteurs représentant les objets N et K clusters. Les prototypes (centroïdes des clusters) sont calculés de manière à minimiser la somme des distances carrées (SSD) entre chaque objet et son prototype le plus proche (Descente de Gradient) :

$$SSD = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu^k\|^2 \tag{1}$$

1.1 Algorithme

Algorithm 1 K-means

Input : Vectorial Data, K
Output : Prototype's vector μ^k.

1: Initialize cluster centroids $\mu^1, ..., \mu^k$
2: **while** the convergence is not attained **do**
3: **for** each object x^i **do**
4: $c^i = \arg\min_j \|x^i - \mu^j\|^2$
5: **for** for each cluster k **do**
6: $\mu^k = \frac{1}{|C_k|} \sum_{x \in C_k} x$

1.2 Questions

1. Implémentez l'algorithme K-Means sous Python
2. Initialiser les centroïdes (prototypes) avec la méthode de K-Means++
3. Initialisez les centroïdes de la façon suivante :
   Initialisez le premier centroïde au hasard. Sélectionnez les $k-1$ autres centroïdes parmi les $k-1$ plus proche du premier centroïde.
4. Calculez la qualité du clustering en utilisant différents indices (de votre choix).

5. Visualisez les clusters de données et leurs centroïdes.
6. Écrivez un protocole qui lance plusieurs fois l'algorithme K-Means (initialisation aléatoire) et calculer la qualité des résultats en utilisant l'indice de Silhouette. Que constatez vous ?
7. Écrivez un protocole qui lance plusieurs fois l'algorithme du K-Means (avec la même initialisation) avec différentes valeurs de K et calculer la qualité des résultats en utilisant l'indice de Silhouette. Que constatez vous ?

2 K-medoid

L'idée de K-medoid est de trouver les prototypes (médoïdes des clusters) parmi les objets o de l'ensemble de données $O = \{o^1, \ldots, o^N\}$ afin de minimiser une fonction de coût $\mathcal{J}$: somme des distances carrés entre chaque objet et son médoïde le plus proche.

$$\mathcal{J} = \sum_{k=1}^{K} \sum_{i|o^i \in C_k} d(o^i, \mu^k)^2 \tag{2}$$

avec K le nombre de clusters (c'est-à-dire le nombre de médoïdes), o un objet de l'ensemble de données, μ^k le médoïde pour le cluster k, $d(i, j)$ le distance carré entre i et j, et C_k l'ensemble des objets du cluster k.

2.1 Algorithme

Algorithm 2 K-medoids

Input : D, K. **Output :** The K medoids m^k.

```
Choose randomly K medoids.
Associate each data point to the closest medoid.
while the cost of the configuration decreases do
    for each pair of object o and medoid m do
        Swap m and o.
        Recompute the cost using equation (2).
        if the cost increased then
            undo the swap.
```

2.2 Questions

1. Implémentez l'algorithme K-Medoids sous Python.
2. Initialiser les medoïdes avec la méthode de K-means++.
3. Calculez la qualité du clustering en utilisant différents indices (de votre choix).
4. Visualisez les clusters et leurs medoïdes.
5. Lancez plusieurs fois l'algorithme du K-Medoids (initialisation au hasard) et comparez la qualité des résultats avec l'algorithme du K-Means. Quel est le plus stable ?

3 Discussion

Quelle méthode donne les meilleurs résultats ? Quels sont les avantages et inconvénients de chaque approche ? Justifiez bien vos réponses.