Data Science et Machine Learning

TP sur la classification automatique non-supervisée (Catégorisation ou Clustering)

A réaliser dans l'environnement Python avec les bibliothèques Numpy, Pandas et Matplotlib

Le jeu de données de ce TP est constitué de 60 points du plan à catégoriser en 3 classes. Le tableau de données X est donc de dimension (n, p) = (60, 2). Les données ont été délibérément choisies en deux dimensions, p=2, pour faciliter la visualisation du fonctionnement des algorithmes de catégorisation. L'objectif du TP est donc purement pédagogique, car l'algorithme K-means, développé dans ce TP, fonctionne bien avec des données de dimension quelconque p.

1. Utilisez la bibliothèque Pandas pour lire le data frame qui se trouve dans un fichier Excel points.xlsx.
2. Construisez le tableau Numpy des données X.
3. Générez aléatoirement 3 centres initiaux en ajoutant des réalisations de la loi Normale centrée réduite aux composantes du centre de gravité du nuage des points.

   ```
   G = np.mean(X,axis=0)
   centres = np.zeros((m,p))
   for k in range(m):
      centres[k,:] = G + np.random.randn(1,p)
   ```
4. Visualisez le nuage de points et les centres initiaux avec Matplotlib.
5. Calculez l'inertie totale de ce nuage de points. Remarquez qu'elle est égale à l'inertie intra avant la catégorisation des individus dans des classes différentes.
6. Construisez une catégorisation en 3 classes avec la méthode K-means en réalisation à chaque tour de la boucle tant que :
   6.1. Une partition des individus en 3 classes.
   6.2. Une actualisation des centres de gravité.
   6.3. Un calcul de la nouvelle inertie intra.
   6.4. Une visualisation de la catégorisation en cours en affectant une couleur à chaque classe.

```
Y = np.zeros(n).reshape(n,1)
Inertie_totale = np.sum([np.power(X[i,:]-G,2) for i in range(n)])/n
new_I_intra = Inertie_totale
amelioration = True
while amelioration :
   # Création d'une partition
   for i in range(n):
      dists = [np.linalg.norm(X[i,:]-...) for k in range(m)]
      Y[i] = np.argmin(...)
   # Calcul des centres de gravités
   XY = np.concatenate((X,Y),axis=...)
   for k in range(m):
      Selection = XY[np.where(XY[:,p] == k)]
      Groupe_k = Selection[:,0:p]
      if Groupe_k.shape[0] > 0 :
         centres[k,:] = np.mean(...,axis=...)
   # Calcul de l'inertie intra-classes
   old_I_intra = new_I_intra
   new_I_intra = ...
   amelioration = (old_I_intra > new_I_intra)
```

7. Affichez la partition finale et les centres de gravité après la convergence de l'algorithme.