FISA4

TP4 : Apprentissage non-supervisé

- Clustering -

Khadidja OULD AMER (khadidja.ouldamer@isen-ouest.yncrea.fr)

Objectifs du TP

- Assimiler le principe de clustering
- L'interprétation des résultats (et non pas juste leur affichage)
- Prise en main des bibliothèques de la partie Liens utiles et savoir utiliser, idéalement, leurs documentations

Liens utiles

- Matplotlib
- Scikit-learn
- Numpy

I- Clustering des données synthétiques via K-means

1. Depuis votre Google Drive, créez un notebook, sur GoogleColab, nommé tp4_IA
2. Créez une section intitulée I- Clustering des données synthétiques via K-means
3. Générez des données synthétiques via le code suivant :

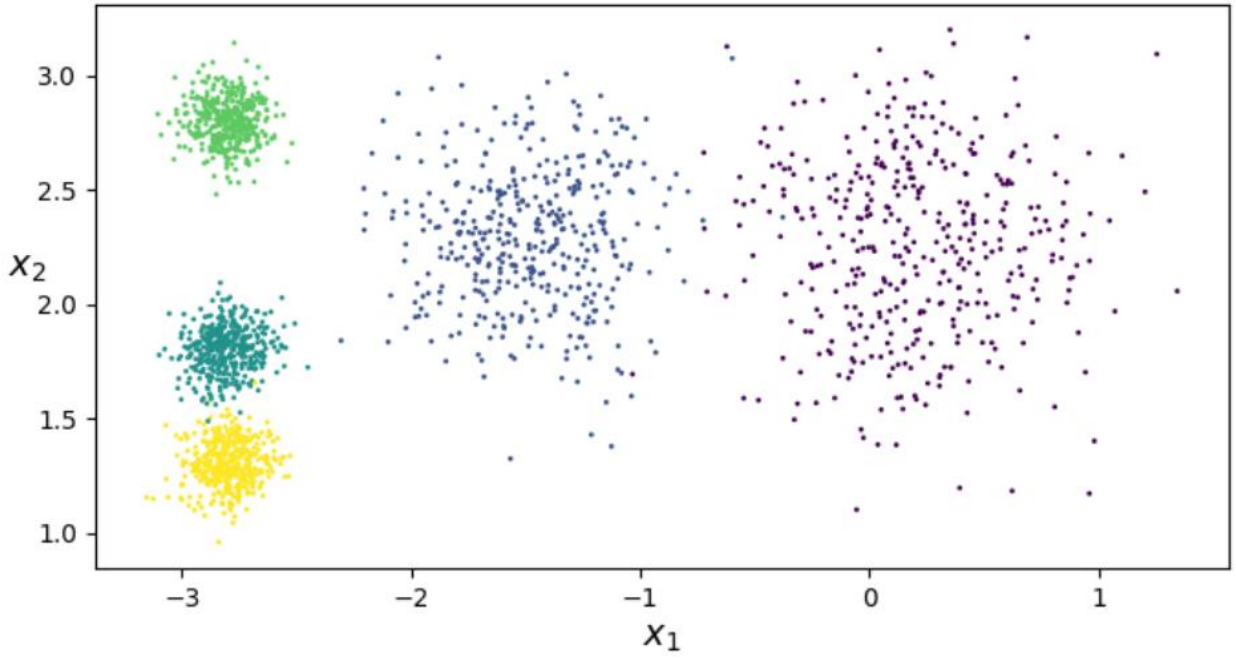
```
from sklearn.datasets import make_blobs
import numpy as np

blob_centers = np.array(
    [[ 0.2,  2.3],
     [-1.5 ,  2.3],
     [-2.8,  1.8],
     [-2.8,  2.8],
     [-2.8,  1.3]])
blob_std = np.array([0.4, 0.3, 0.1, 0.1, 0.1])
# "X" représente les données et "y" représente les indices de clusters réels
X, y = make_blobs(n_samples=2000, centers=blob_centers,
```

```
                    cluster_std=blob_std, random_state=7)
```

4. Affichez les données synthétiques générées via le code suivant :

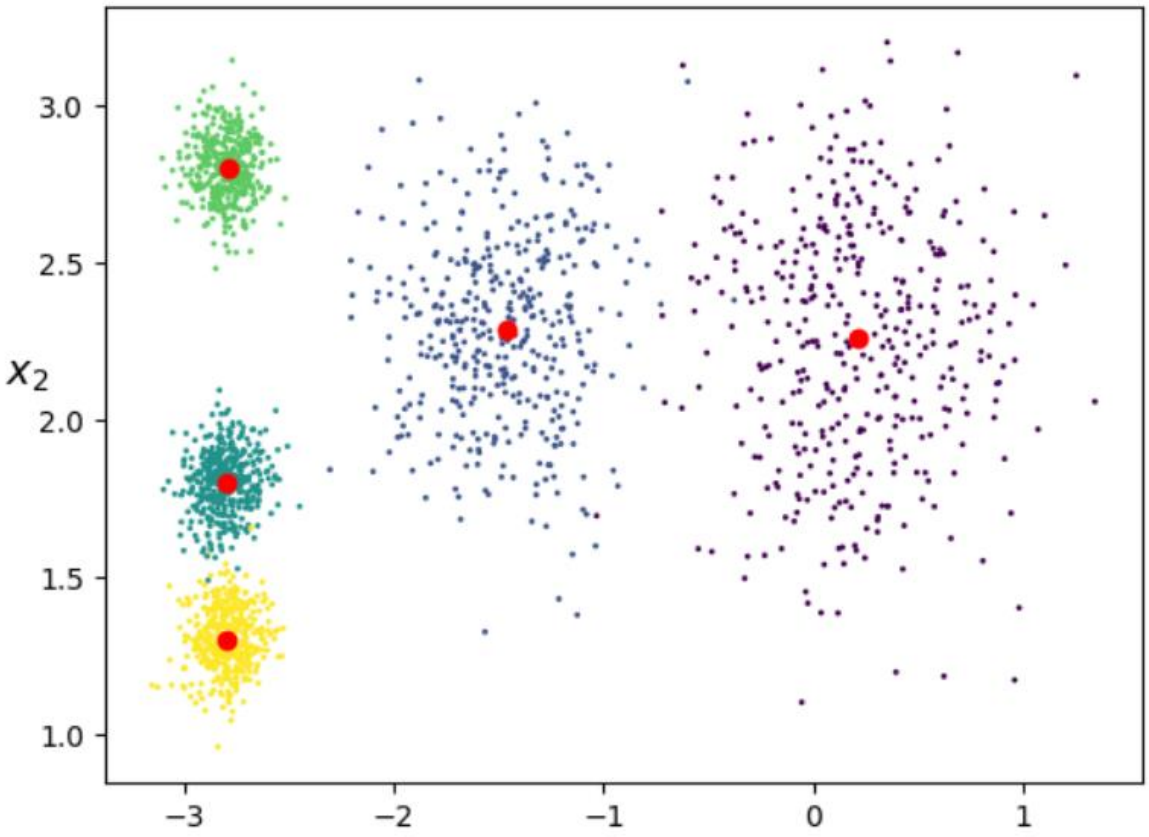
```
 import matplotlib.pyplot as plt

def plot_clusters(X, y=None):
    plt.scatter(X[:, 0], X[:, 1], c=y, s=1)
    plt.xlabel("$x_1$", fontsize=14)
    plt.ylabel("$x_2$", fontsize=14, rotation=0)

plt.figure(figsize=(8, 4))
plot_clusters(X, y)
plt.show()
```

L'affichage souhaité est :



5. Appliquez la méthode KMeans (avec k=5) sur les données synthétiques (X). Pour ce faire, instanciez un objet de la classe KMeans du sous-module cluster du module sklearn
6. En utilisant la méthode fit_predict, prédisez et affichez les indices des cluster de X. Stockez les indices de clusters prédits dans une variable
7. Pour évaluer le modèle, calculez son NMI (Normalized Mutual Index). Cet indice peut être obtenu via la fonction normalized_mutual_info_score du sous-module metrics du module sklearn

8. Intégrez, en rouge dans la figure ci-dessus, les centroïdes des cinq clusters via l'attribut "cluster_centers_" de l'objet créé dans la question 5



9.
   a. Créez la nouvelle instance suivante :

```
X_new = np.array([[-3, 2.5]])
```

   b. Calculez la distance de X_new aux centroïdes des cinqs clusters via la méthode transform de la classe KMeans. Quel cluster a le centroïde le plus proche de X_new ?
   c. Prédisez l'indice du cluster de X_new en utilisant la fonction predict. L'indice du cluster est conforme avec votre réponse à la question 9.b ?

II- Clustering des images faciales via K-means

10. Créez une section intitulée II- Clustering des images faciales via K-means
11. Importez la base de données fetch_olivetti_faces() depuis le sous-module datasets du module sklearn
12. Affichez la description de cette base de données via l'attribut DESCR
13. Affichez les indices de cluster de cette base de données via l'attribut target
14. Affichez les données de cette base de données via l'attribut data
15. Divisez cette base de données en bases d'apprentissage, de validation et de test. Pour ce faire, utilisez deux fois la fonction train_test_split du sous-module model_selection du module sklearn tout en optant la répartition suivante : 60% pour l'apprentissage 20% pour le test et 20% pour la validation

16. Appliquez KMeans sur les données d'apprentissage avec k=40
17. Prédisez les indices de cluster des données de validation tout en les affichant
18. Calculez le NMI (Normalized Mutual Index) du modèle. Cet indice peut être obtenu via l'appel de la fonction normalized_mutual_info_score du sous-module metrics du module sklearn
19. Proposez un code qui appelle la fonction ci-dessous et affiche les images faciales de validation et leurs cluster (voir le résultat ci-dessous) :

```
import numpy as np
import matplotlib.pyplot as plt

def plot_faces(faces, n_cols=5):
    n_rows = (len(faces) - 1) // n_cols + 1
    plt.figure(figsize=(n_cols, n_rows * 1.1))
    for index in range(0, len(faces)):
        plt.subplot(n_rows, n_cols, index + 1)
        plt.imshow(faces[index].reshape(64, 64), cmap="gray")
        plt.axis("off")
    plt.show()
```