Master 1 SD

Nom : Coelho Fernandes
Prénom : Rodrigo César

Analyse et visualisation de données

Séance de TP 6
ISOMAP

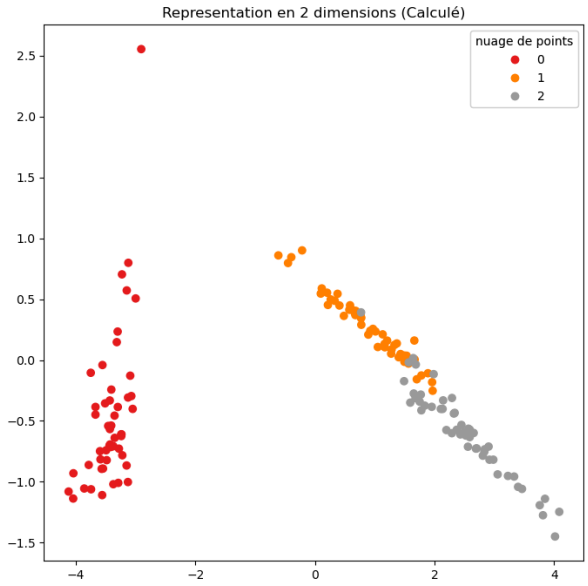
1- ISOMAP

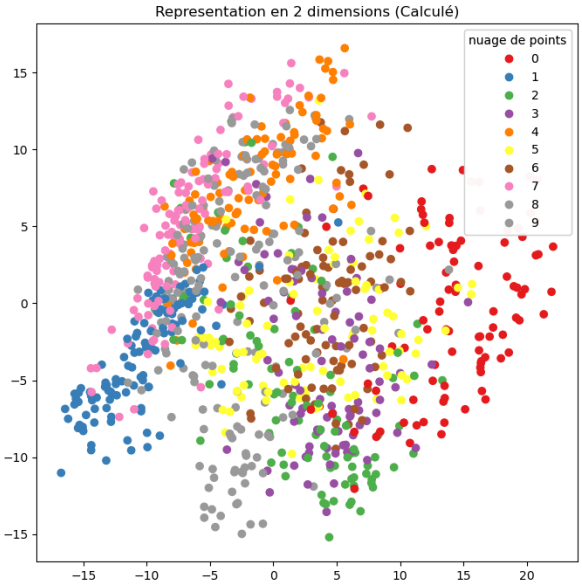
a. *Programmer la méthode ISOMAP en vous conformant au modèle proposé.*

```
from sklearn.neighbors import kneighbors_graph
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
from scipy.sparse.csgraph import shortest_path
from scipy.linalg import eigh
from sklearn import manifold

# Etape 1: construire un graphe des voisins
def Etape1(X, n_neighbors):
    kng = kneighbors_graph(X, n_neighbors,mode='distance')

    # kng = 0.5*(kng+kng.T) # podemos parametrizar para tornar kng simetrico
    print('Non zero values:',kng.nnz)

    # Matrice des plus proches voisins
    plt.figure(figsize=(8,8))
    plt.imshow(kng.todense())
    plt.title("Matrice des plus proches voisins")
    plt.show()

    # "Zoom" sur les 50 premiers
    plt.figure(figsize=(8,8))
    plt.imshow(kng.todense()[:50,:50])
    plt.title("Zoom sur la Matrice des plus proches voisins (50 premiers
points)")
    plt.show()

    return kng

# Etape 2: calcule des distances géodésiques entre les points
def Etape2(kng):
    D = shortest_path(kng,directed=True)
```

```
    plt.figure(figsize=(8,8))
    plt.title("Matrice des Distances Geodesiques")
    plt.imshow(D)
    plt.show()

    return D


# Etape 3: construire l'embedding de basse dimension à partir des distances
géodésiques
def Etape3(D):
    m = np.shape(X)[0]
    Id = np.eye(m)
    Ones = np.ones((m,m))

    centrage = Id - (Ones/m)
    B = -0.5 * (centrage) @ (D**2) @ (centrage)

    plt.figure(figsize=(8,8))
    plt.imshow(B)
    plt.title("Matrice des Similarités")
    plt.show()

    [v,V] = eigh(B) # sao criados em ordem crescente

    Y_iso = np.fliplr(V[:,-2:] @ np.diag(v[-2:]**0.5))
    fig,ax = plt.subplots(figsize=(8,8))
    scatter = ax.scatter(Y_iso[:,0],Y_iso[:,1], c = y[0:m],cmap = plt.cm.Set1)
    plt.title("Representation en 2 dimensions (Calculé)")
    legend1 = ax.legend(*scatter.legend_elements(),loc = "upper right",title=
"nuage de points")
    ax.add_artist(legend1)

    plt.show()
```

b. ***Visualiser le résultat d'ISOMAP sur les deux dataset IRIS et MNIST.***

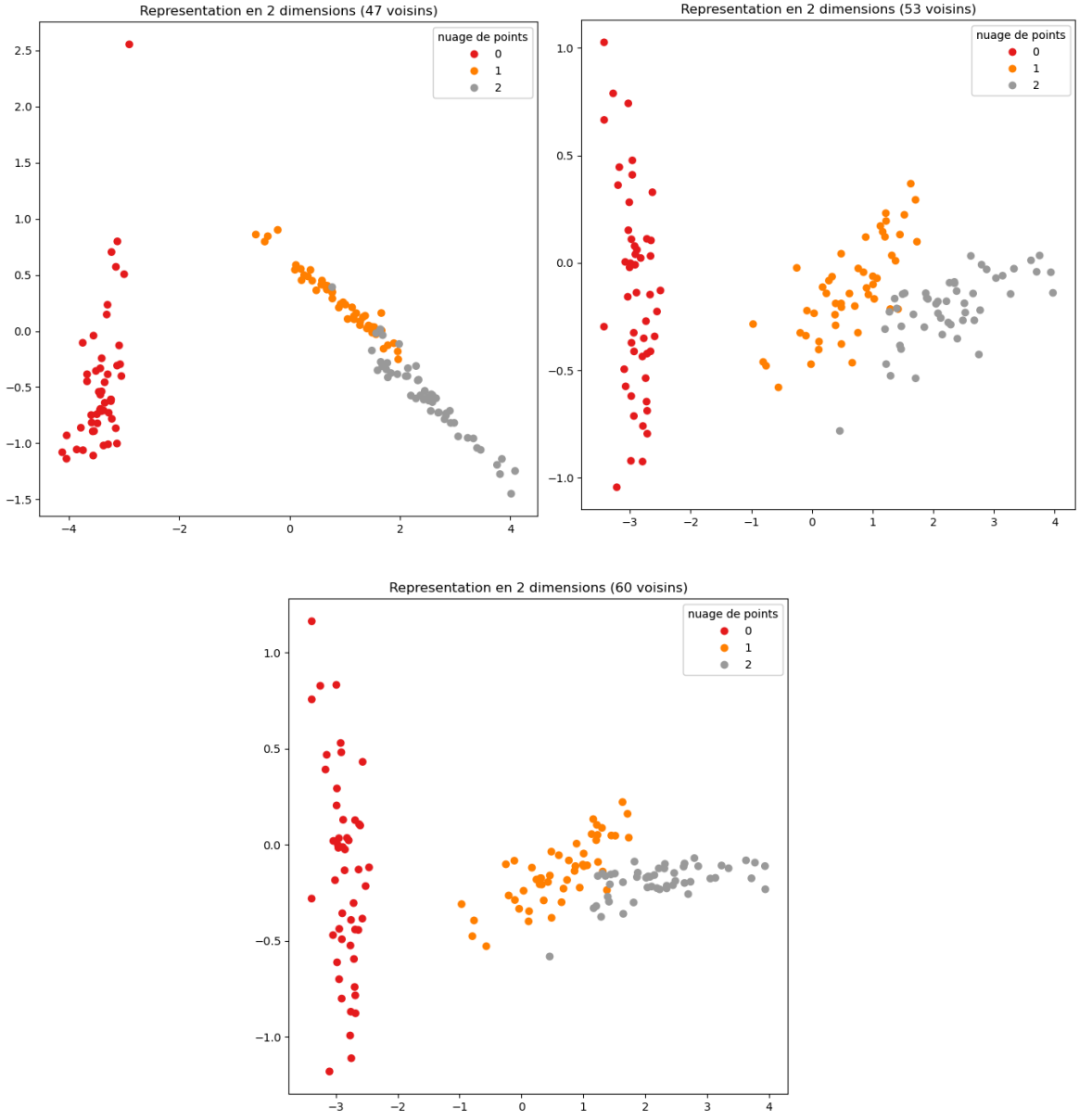




La méthode ISOMAP s'est révélée efficace pour réduire les quatre dimensions du jeu de données IRIS à seulement deux, car les clusters sont représentés de manière plus dense.

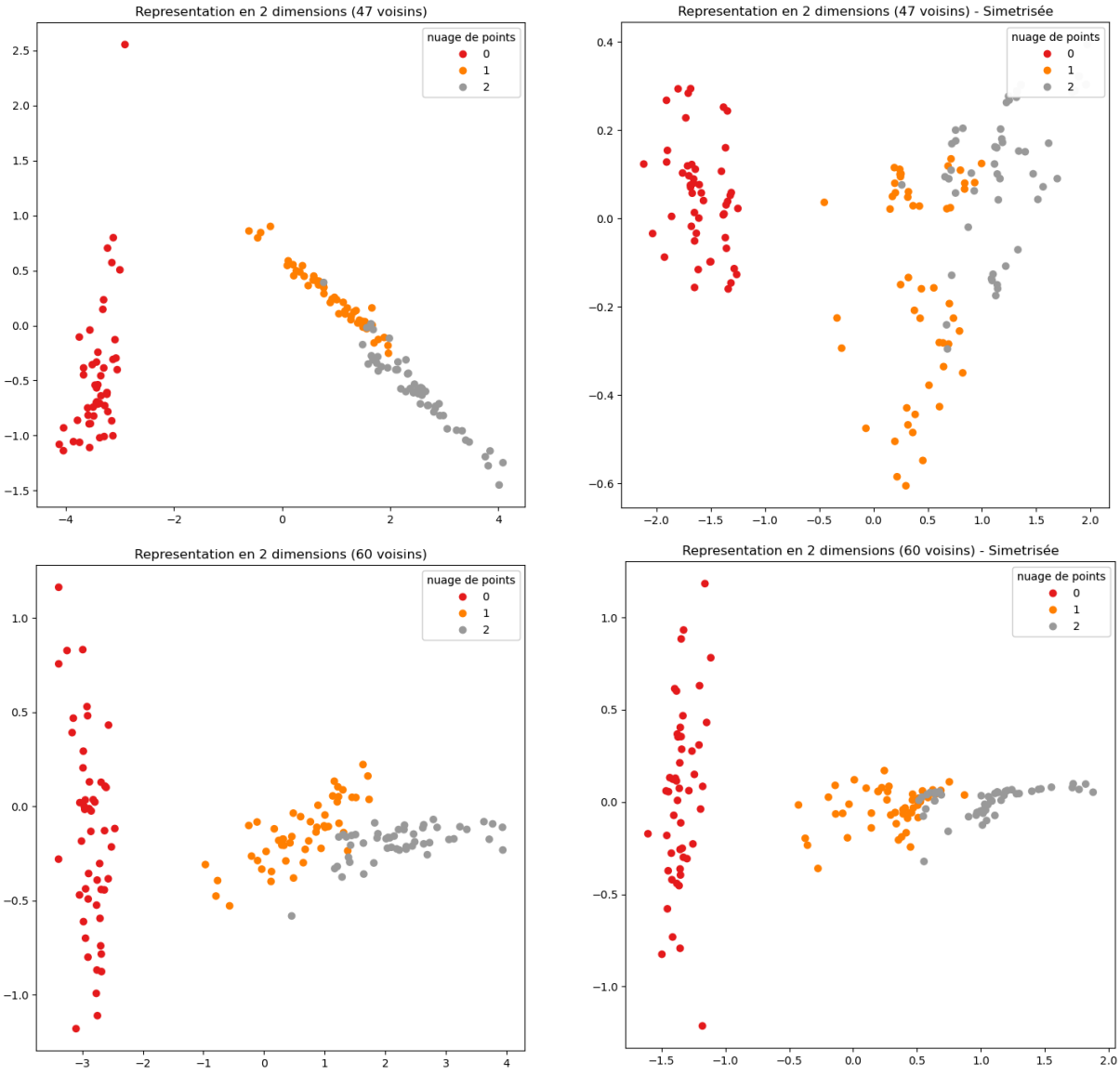
En revanche, pour le jeu de données MNIST, l'ISOMAP ne semble pas capable de le réduire à seulement deux dimensions, car les clusters apparaissent extrêmement superposés dans l'espace. Étant donné que ce jeu de données contient initialement 784 caractéristiques, la réduction à seulement deux dimensions n'est pas efficace.

c. ***Déterminer l'influence du nombre de voisins sur les résultats obtenus.***



Il a été observé, pour le jeu de données IRIS, que l'augmentation du nombre de voisins a significativement influencé les résultats. Avec une valeur de 47 voisins, la distribution est consolidée, ce qui indique que les points présentent une forte similarité avec les points du même cluster. Cependant, en augmentant la valeur de n_neighbors, la distribution devient moins dense, ce qui suggère que la similarité avec des points d'autres clusters perturbe alors le résultat final.

d. *Examiner la possibilité de symétriser le graphe des plus proches voisins et l'influence de cette symétrisation sur les résultats.*

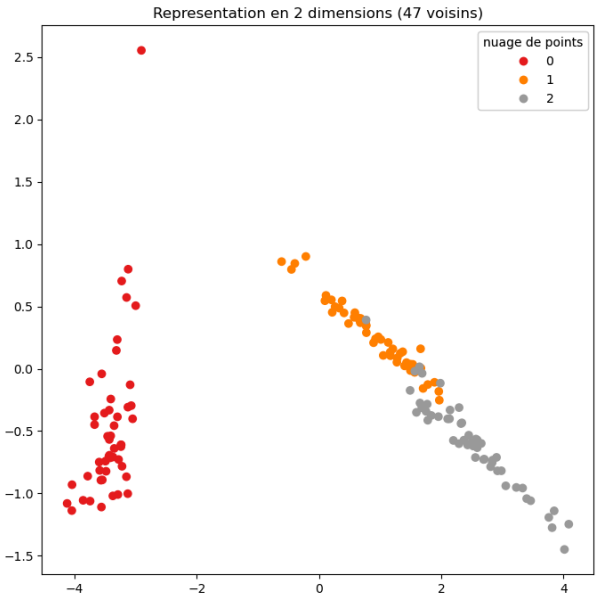


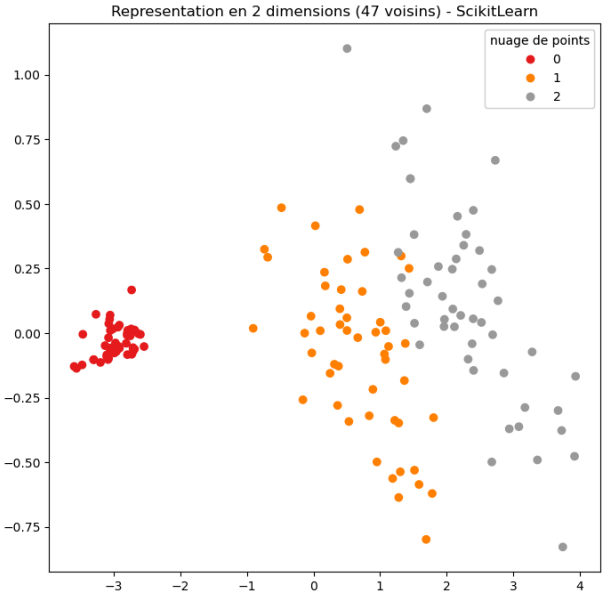
Les images ci-dessus montrent que le résultat final de l'ISOMAP est extrêmement modifié en symétrisant la matrice des Plus Proches Voisins. La symétrisation vise à apporter une plus grande cohérence aux relations de similarité entre les points, en garantissant que les relations de voisinage soient bidirectionnelles.

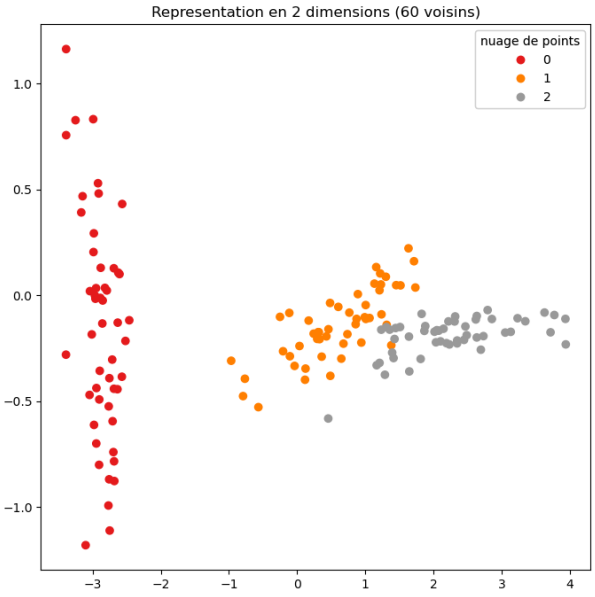
Pour le jeu de données IRIS, il a été observé que la symétrisation a, en réalité, rendu la représentation en deux dimensions plus dispersée lorsque 47 voisins sont considérés. Cependant, pour 60 voisins, la représentation est plus condensée après la symétrisation.

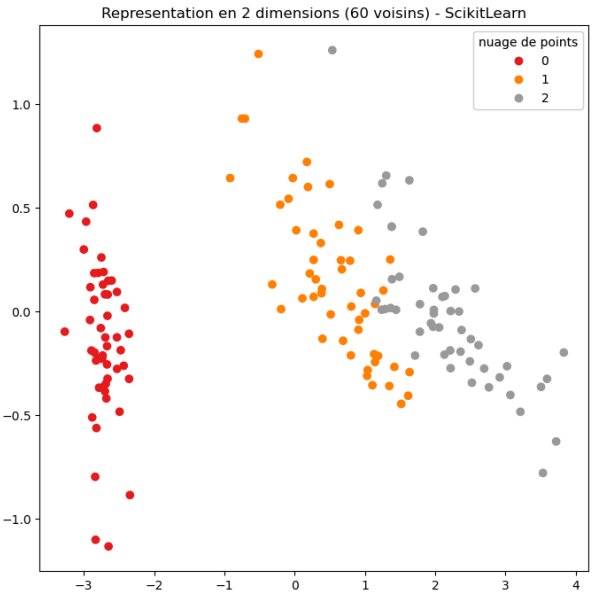
2- Comparaison avec Scikitlearn

a. Comparer vos résultats à ceux obtenus avec scikitlearn pour les mêmes hyper-paramètres.









En évaluant les résultats obtenus avec l'ISOMAP construit manuellement et celui de ScikitLearn, il est perceptible que les clusters 1 et 2 sont plus denses dans les représentations générées par le premier. Cependant, ScikitLearn a réussi à offrir une densité plus élevée pour le cluster 0. Cette tendance a été observée pour différentes valeurs de voisins.

```python
from sklearn.neighbors import kneighbors_graph
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
from scipy.sparse.csgraph import shortest_path
from scipy.linalg import eigh
from sklearn import manifold
from sklearn.datasets import fetch_openml


# Etape 1: construire un graphe des voisins
def Etape1(X, n_neighbors):
    kng = kneighbors_graph(X, n_neighbors,mode='distance')

    kng = 0.5*(kng+kng.T) # podemos parametrizar para tornar kng simetrico
    # print('Non zero values:',kng.nnz)

    # Matrice des plus proches voisins
    plt.figure(figsize=(8,8))
    plt.imshow(kng.todense())
    plt.title("Matrice des plus proches voisins")
    plt.show()

    # "Zoom" sur les 50 premiers
    plt.figure(figsize=(8,8))
    plt.imshow(kng.todense()[:50,:50])
    plt.title("Zoom sur la Matrice des plus proches voisins (50 premiers
points)")
    plt.show()

    return kng

# Etape 2: calcule des distances géodésiques entre les points
def Etape2(kng):
    D = shortest_path(kng,directed=True)

    plt.figure(figsize=(8,8))
    plt.title("Matrice des Distances Geodesiques")
    plt.imshow(D)
    plt.show()

    return D


# Etape 3: construire l'embedding de basse dimension à partir des distances
géodésiques
def Etape3(D, n_neighbors):
    m = np.shape(X)[0]
    Id = np.eye(m)
```

```
    Ones = np.ones((m,m))
    C = y[0:m].astype(float)

    centrage = Id - (Ones/m)
    B = -0.5 * (centrage) @ (D**2) @ (centrage)

    plt.figure(figsize=(8,8))
    plt.imshow(B)
    plt.title("Matrice des Similarités")
    plt.show()

    [v,V] = eigh(B) # sao criados em ordem crescente

    Y_iso = np.fliplr(V[:,-2:] @ np.diag(v[-2:]**0.5))
    fig,ax = plt.subplots(figsize=(8,8))
    scatter = ax.scatter(Y_iso[:,0],Y_iso[:,1], c = C,cmap = plt.cm.Set1)
    plt.title("Representation en 2 dimensions ({} voisins) -
Simetrisée".format(n_neighbors))
    legend1 = ax.legend(*scatter.legend_elements(),loc = "upper right",title=
"nuage de points")
    ax.add_artist(legend1)

    plt.show()

# Comparer les résultats à ceux obtenus avec scikitlearn
def CompScikt(X, y, n_neighbors):
    n_components = 2
    m = np.shape(X)[0]
    C = y[0:m].astype(float)

    X_iso = manifold.Isomap(n_neighbors = n_neighbors , n_components =
n_components,path_method='D').fit_transform(X)
    fig,ax = plt.subplots(figsize=(8,8))
    scatter = ax.scatter(X_iso[:,0],X_iso[:,1], c = C,cmap = plt.cm.Set1)
    plt.title("Representation en 2 dimensions ({} voisins) -
ScikitLearn".format(n_neighbors))
    legend1 = ax.legend(*scatter.legend_elements(),loc = "upper right",title=
"nuage de points")
    ax.add_artist(legend1)
    plt.show()

# Visualiser le résultat d'ISOMAP sur IRIS
# Déterminer l'influence du nombre de voisins sur les résultats obtenus.
n_neighbors_list = np.linspace(47, 60, 3, dtype=int)

iris = datasets.load_iris()
X = iris.data
y = iris.target
```

```
for n_neighbors in n_neighbors_list:
    print(f'\nProcessando para n_neighbors = {n_neighbors}')
    kng = Etape1(X, n_neighbors)
    D = Etape2(kng)
    Etape3(D, n_neighbors)
    CompScikt(X, y, n_neighbors)


# Visualiser le résultat d'ISOMAP sur MNIST
n_neighbors = 47

dMNIST = fetch_openml("mnist_784", version=1, as_frame=False)
X = dMNIST.data[:1000]
y = dMNIST.target[:1000]
X = (X - np.min(X, axis=0)) / (np.ptp(X, axis=0) + 1e-6)
print(np.shape(X))

kng = Etape1(X, n_neighbors)
D = Etape2(kng)
Etape3(D, n_neighbors)
```