Master 1 SD

Nom : Coelho Fernandes
Prénom : Rodrigo César

Analyse et visualisation de données

Séance de TP 7
Cartes propres Laplaciennes : Laplacian Eigenmaps

1- ISOMAP

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

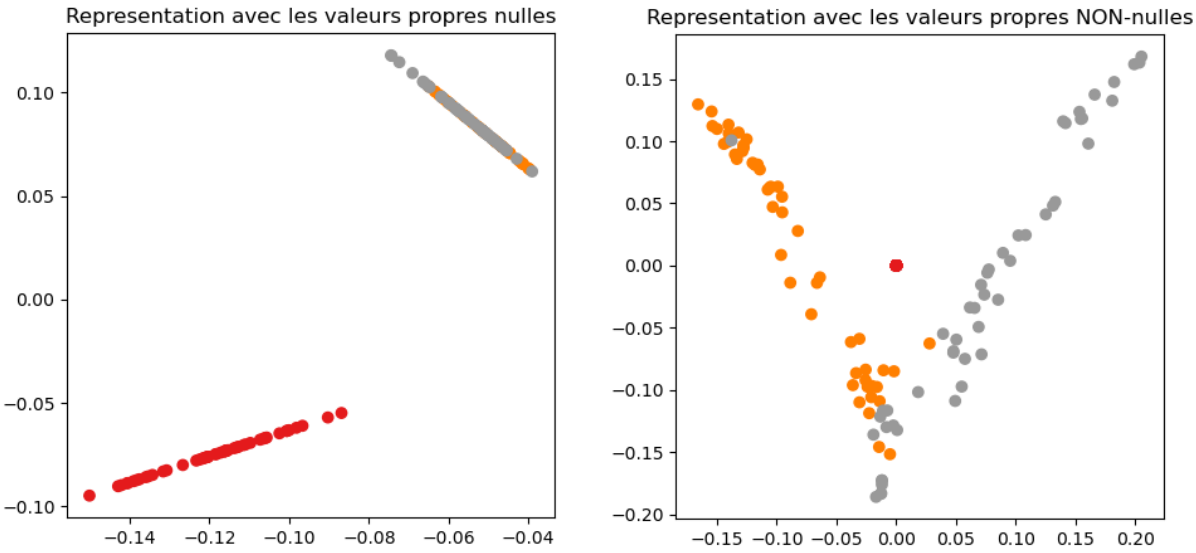
```
from sklearn.neighbors import kneighbors_graph
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
from scipy.linalg import eigh
from sklearn import manifold
from scipy.sparse import diags, linalg

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

N = X.shape[0]

"""
**Etape 1: Construire un graphe des voisins, pour chaque point Xi , déterminer
ses kPPV**
1. chaque nœud est connecté à ses kPPV
2. les arrêtes de poids non-nuls prennent comme valeurs la distance
euclidienne entre les 2 nœuds
3. symétriser le graphe pour obtenir la matrice d'adjacence W
"""
n_neighbors = 4
kng = kneighbors_graph(X, n_neighbors,mode='distance')
kng.nnz
# simetrizar (pode aumentar o numero de vizinhos para k+(k-1))
W = 0.5*(kng+kng.T)

plt.figure(figsize=(5,5))
plt.imshow(kng.todense())
plt.figure(figsize=(5,5))
plt.imshow(W.todense())
```

```
"""
**Etape 2: Calculer la matrice des degrés D**
"""
# matriz de degree ou matrice de adjacences (soma dos valores das colunas)
D = diags(np.asarray(W.sum(axis=0)).flatten())

"""
**Etape 3: Calculer la matrice Laplacienne**
"""
Laplacian = D-W

"""
**Etape 4: Déterminer les plus faibles valeurs propres non nulles de la
matrice Laplacienne**
"""
[yl, YL] = linalg.eigsh(Laplacian, n_neighbors, which='SM')
print("Val propres Laplacian non  normalisé =  ", yl)

fig,ax = plt.subplots(figsize=(5,5))
scatter = ax.scatter(YL[:,2],YL[:,3], c = y[0:N],cmap = plt.cm.Set1)

# Si on normalise le laplacian
Dinv = linalg.inv(D)
Dinv = Dinv.sqrt()
NormLaplacian  =  Dinv @ Laplacian @ Dinv

[yln, YLn] = linalg.eigsh(NormLaplacian, n_neighbors, which='SM')
print("Val propres Laplacian non  normalisé =  ", yln)

fig,ax = plt.subplots(figsize=(5,5))
scatter = ax.scatter(YLn[:,2],YLn[:,3], c = y[0:N],cmap = plt.cm.Set1)

"""
**Comparer à scikitlearn**
"""
C = y[0:N].astype(float)

X_iso = manifold.SpectralEmbedding(n_neighbors = n_neighbors , n_components =
3,
affinity='nearest_neighbors',random_state=0,eigen_solver="arpack").fit_transfo
rm(X)
fig,ax = plt.subplots(figsize=(5,5))
scatter = ax.scatter(X_iso[:,1],X_iso[:,2], 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()
```

b. *Déterminer l'influence des valeurs propres sur le résultat*

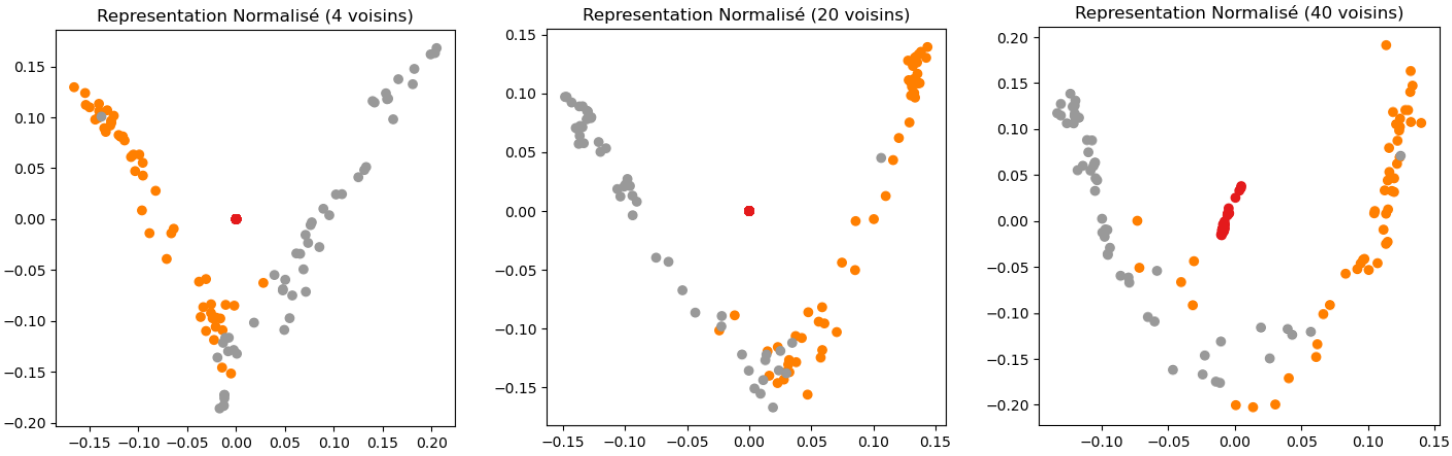
En analysant pour un nombre de 4 voisins, la matrice laplacienne possède 2 valeurs propres nulles et 2 valeurs propres non nulles.
Val propres Laplacian non normalisé = [2.89e-17, 3.82e-17, 1.21e-02, 3.16e-02]

Le choix des vecteurs propres associés aux valeurs propres est fondamental pour le succès de la méthode. Il est idéal de sélectionner ceux qui sont associés aux plus petites valeurs propres non nulles.



Comme on peut l'observer dans ces distributions des données normalisées, les vecteurs propres correspondant aux valeurs propres nulles ne fournissent pas une bonne séparation des données de deux classes qui se chevauchent. En revanche, lorsque nous utilisons les valeurs propres non nulles, la zone de chevauchement entre les classes est beaucoup plus réduite, ce qui permet une classification plus précise.

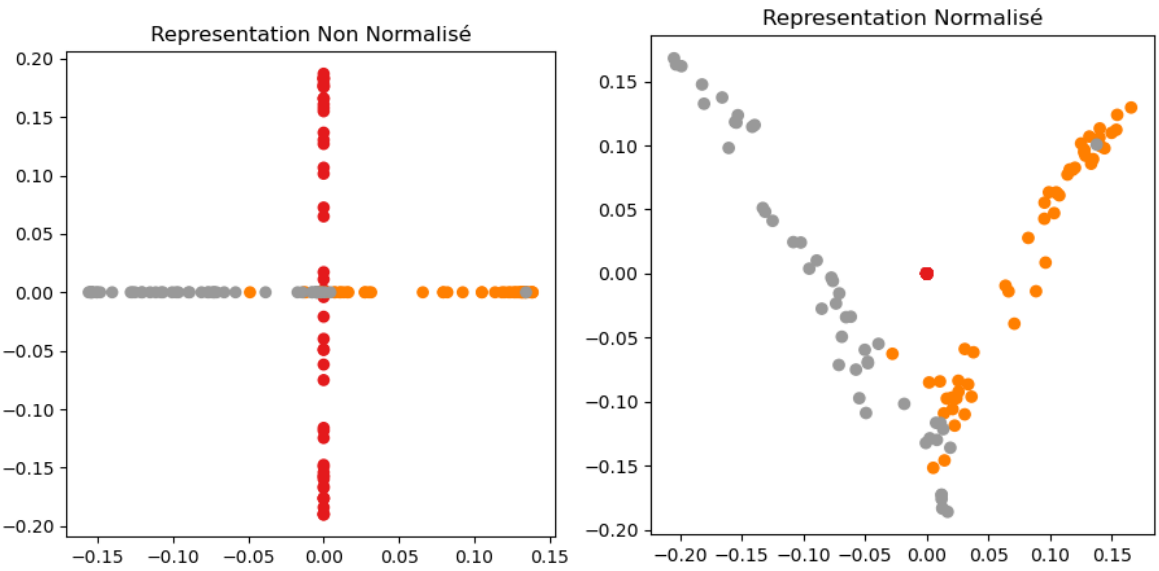
c. *Déterminer l'influence du nombre de voisins sur les résultats obtenus.*



Les graphiques ci-dessus montrent la projection des points du jeu de données IRIS sur les deux autovecteurs associés aux plus petits autovaleurs non nulles. Le nombre de voisins pris en compte pour établir les relations de similarité a été modifié.

Nous observons que l'augmentation du nombre de voisins provoque une plus grande dispersion des points dans l'espace. La prise en compte d'un plus grand nombre de voisins signifie que des points ayant une moindre similarité commencent à influencer les clusters. Par exemple, le cluster rouge perd sa cohésion avec une forte augmentation du nombre de voisins. La dispersion des autres clusters augmente également pour des valeurs plus faibles de voisins, car ce sont des clusters partageant un plus grand nombre de points similaires.

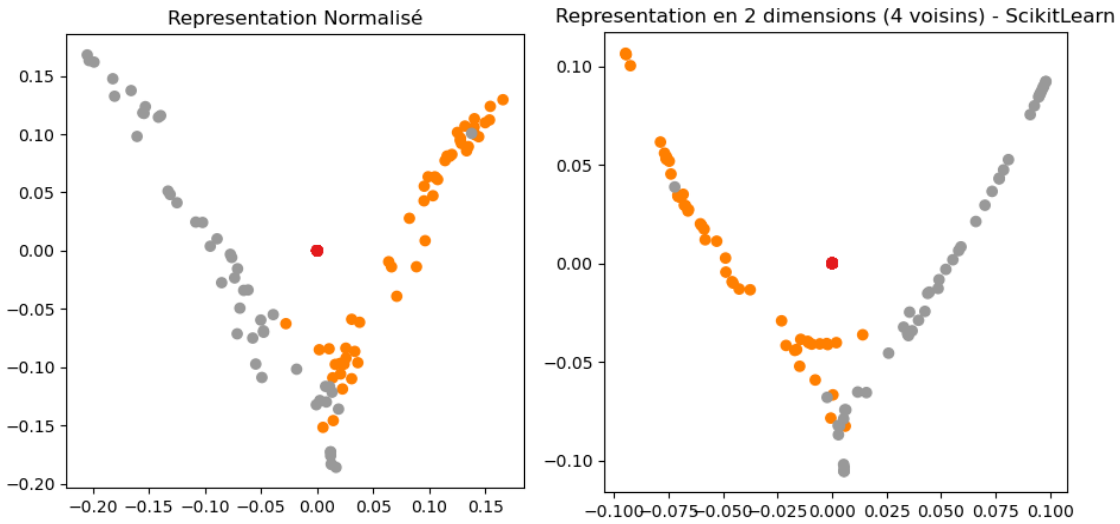
d. Déterminer l'influence de la normalisation du laplacian sur les résultats.



La normalisation de la matrice laplacienne a un effet fondamental pour améliorer la distribution des points. Les graphiques ci-dessus, générés pour 4 voisins, montrent qu'avant la normalisation, les points sont distribués de manière à ce qu'il y ait un chevauchement entre les trois classes, plus particulièrement entre les classes grise et orange.

Après la normalisation, l'une des classes s'est concentrée en un seul point isolé, ce qui facilite sa classification. Les deux autres classes présentent une zone de chevauchement plus réduite. On peut dire qu'après la normalisation, les points se répartissent mieux dans l'espace disponible.

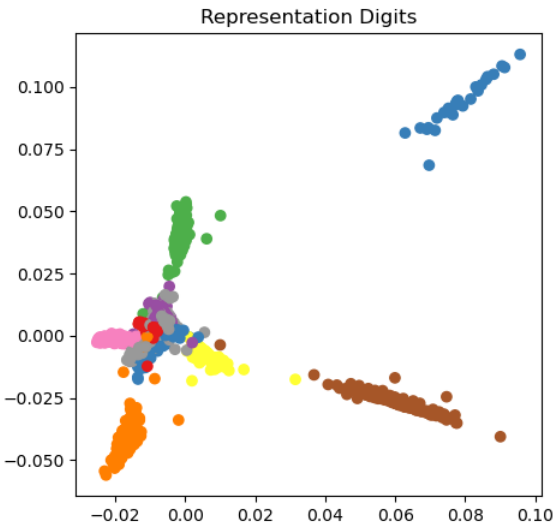
e. Comparer vos résultats à ceux obtenus avec scikitlearn pour les mêmes hyper-paramètres.



Les résultats obtenus avec scikit-learn sont proches de ceux obtenus après la normalisation, ce qui indique que cette étape est appliquée par la fonction *manifold.SpectralEmbedding* de cette bibliothèque.
Cette similarité indique également que des vecteurs propres similaires ont été identifiés. De la même manière, les plus petits vecteurs propres non nuls se sont révélés les plus efficaces pour la séparation des clusters.

Nous pouvons également noter que l'inversion de l'un des axes est un effet normal, car le signe de l'autovaleur associée n'a pas d'importance dans la normalisation et la centralisation du jeu de données.

f. ***Visualiser le résultat de la méthode Laplacian Eigenmaps sur le dataset DIGITS.***



Le jeu de données Digits présente une grande quantité de caractéristiques (64). Dans des représentations simplement bidimensionnelles de ses points, il existe un chevauchement important entre les clusters. Comme le montre l'image ci-dessus, la méthode des Cartes Propres Laplaciennes s'est révélée efficace pour isoler 4 de ses 10 clusters.