Machine learning for clustering

M. Mougeot

ENSIIE, November 2020

Contents

Introduction

Goal of the practical sessions

- To understand classification machine learning methods, from a methodological and practical point of view.
- To apply models and to tune the appropriate parameters on several data sets using the 'Python' language.
- To interprete 'Python' outputs.

Warnings and Advices

- The goal of this practical session is not "just to program with Python"" but more specifically to understand the framework of Modeling, to learn how to developp appropriate models for answering to a given operationnal question on a given data set. The MAL course belongs to the **Data Science courses** . For each MAL practical session, you should **first understand** the mathematical and statistical backgrounds, **then write your own program with 'Python'** to practically answer to the questions.

Instructions

- The practical work must be carried on with a 'group of two students'.
- The MAL project aims to developp a Jupyter notebook to solve a classification problem (the subject will be given soon). Your names have to be written in the first lines of the program file with comments. Please note that without this information, no grade will be attributed to the missing name project.

Python libraries

For this practical session, the following librairies need bo be uploaded in the python environment.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats
from sklearn import mixture
```

Mixture of models and the EM algorithms

The data

The data are stored in the irm_thorax.txt file and represent a human thorax image.

```
tab=pd.read_csv('irm_thorax.txt',sep=';',header=None);
tab=np.array(tab);
n=len(tab); X=tab.reshape(n*n);
```

Medical thorax image

Use the following instructions to display the Thorax image:

```
figure = plt.figure(figsize=(4,2));
plt.imshow(tab, cmap='gray');
plt.xticks([], []); plt.yticks([], []);
plt.show();
```

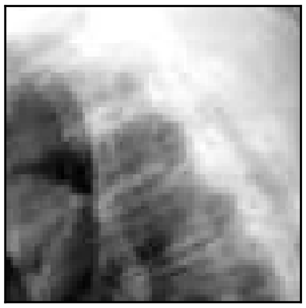

Figure 1: Thorax image

Image histogram

Display the histogram of the pixel values using the following instructions

```
n=len(tab); X=tab.reshape(n*n);
plt.hist(X, range = (0, 255), bins = 20, color = 'yellow', edgecolor = 'red');
```

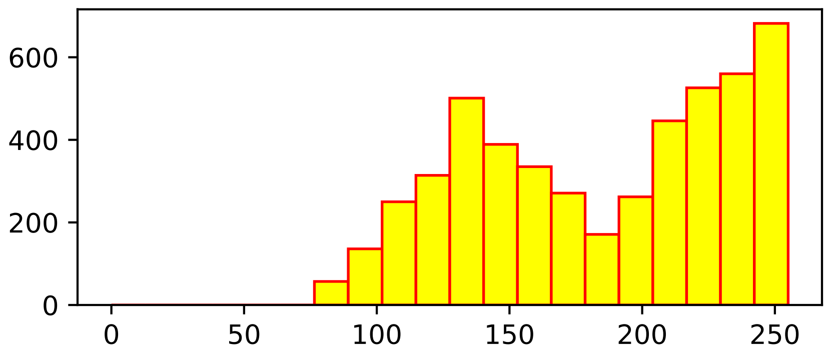

Figure 2: Histogram

What can we observe ?

Mixture of Gaussian writing your own Python instructions

The EM algorithm allows to realize a maximum likelihood type approach on mixture models.

- Write your own Python instructions to implement the Expectation-Maximization algorithm with the help of the ML lesson slides.
- Apply your EM code on this dataset to segment in two classes the thorax image based on the pixel values.

A bimodal distribution assumption

Display the parameters of the gaussian models (weight, mean and standard deviation) and draw the mixture of gaussian on the empirical density of the pixels (scipy.stats.norm.pdf function) as in the following graph:

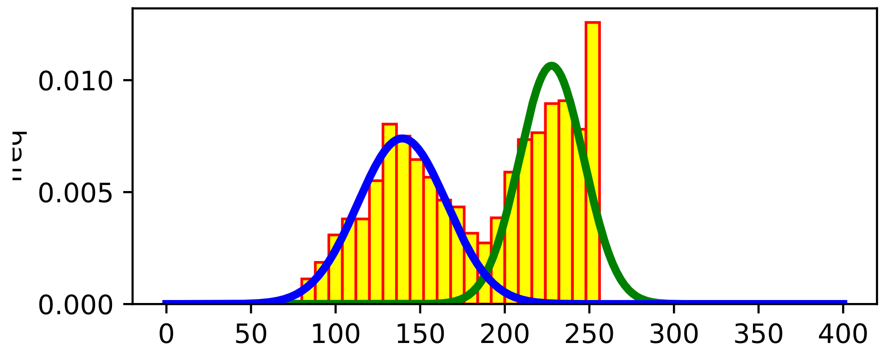

Figure 3: Mixture of gaussian densities. Assumption: K=2

Gaussian Mixture Models (GMM) using scikit learn

- Study the help of the `mixture.GaussianMixture()` function of the scikitlearn library.
- Use the following instructions to recover the previous segmentation with an assumption 2 classes.

```
from sklearn import mixture
X2 = X.reshape(-1, 1)
modgmm = mixture.GaussianMixture(n_components=2, covariance_type='full');
fitgmm=modgmm.fit(X);predX=fitgmm.fit_predict(X);imgGMMK2=predX.reshape(n,n);

#Display
figure = plt.figure(figsize=(5.5,2));
plt.xticks([], []); plt.yticks([], []);
ax = plt.subplot(1,2, 1); ax.imshow(tab, cmap='gray');
ax.set_title('Original Image')
ax = plt.subplot(1,2, 2); ax.imshow(imgGMMK2, cmap='Paired')
ax.set_title('GMM Image Segmentation')
plt.show();
```

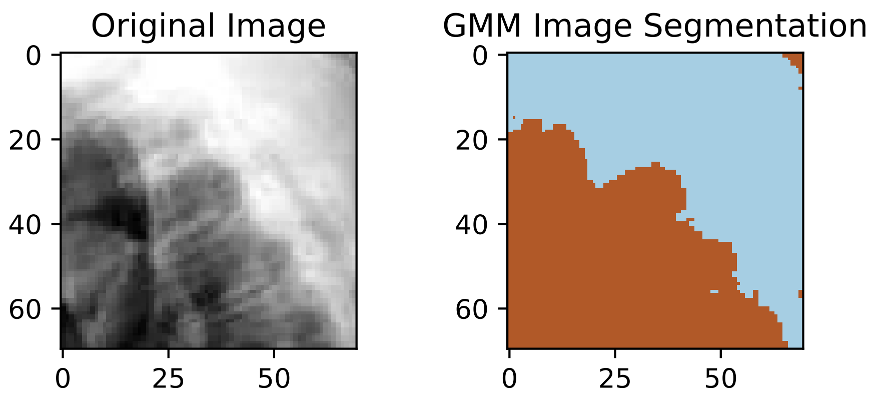

Figure 4: GMM segmentation. Assumption: K=2

Multimodal distribution assumption

- Study the help of the `mixture.GaussianMixture()` function of the scikitlearn library.
- Compute and dispay a segmentation of the thorax image with 3 and 5 classes.
- Display the parameters of the gaussian models (weight, mean and standard deviation).
- Do the segmentations seem relevant to you?

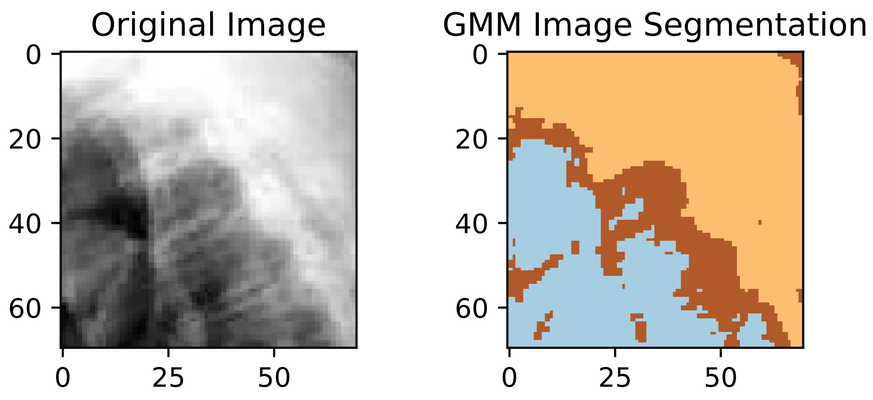

Figure 5: GMM segmentation. Assumption: K=3

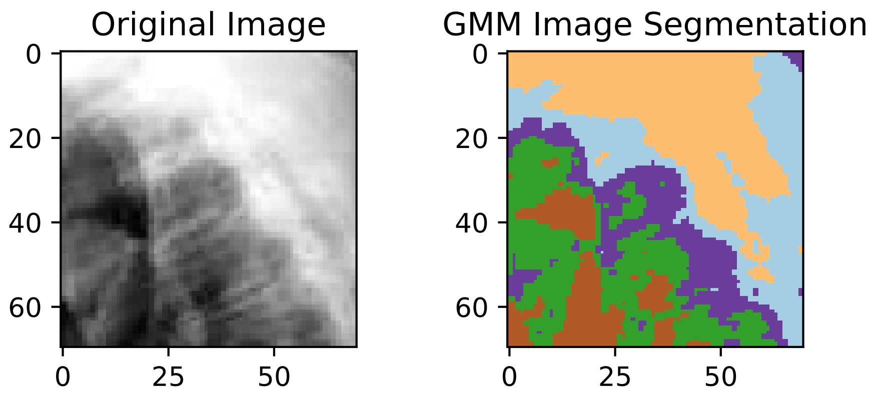

Figure 6: GMM segmentation. Assumption: K=5

Spectral clustering

This section presents the main steps to implement the Spectral Clustering (SC) algorithm and to evaluate the impact of the choices of the hyper parameters on the results provided by the SC algorithm of simulated data.

For this practical session, the following librairies need bo be uploaded in the python environment.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster, datasets
from sklearn.preprocessing import StandardScaler
from matplotlib.colors import ListedColormap
import pandas as pd
from scipy.spatial import distance_matrix
import math
```

The data

Use the following instructions to generate a sample of 2 circles with $nsub = 150$ points in each circle as illustrated in te following figure (left).

```
nsub = 150;
my2circles = datasets.make_circles(n_samples=2*nsub, factor=.5, noise=.025)
X,y=my2circles;
X = StandardScaler().fit_transform(X)
plt.scatter(X[:, 0], X[:, 1], s=10,c=y, cmap='Paired')
plt.xlabel("x1"); plt.xlabel("x2");
```

Imbricated circles

Add $nsub$ sample points drawn fromm a 2D Gaussian law with zero means, variance equaled to 0.001 and no correlation in the middle of the two circles as illustrated in the following figure (right).

The final data set used in the section, contains $n = 450$ points and $nsub = 150$ points in each circular area.

Spectral clustering algorithm writing your on Python instruction

Follow the presented steps to write your own Python instructions to implement the Spectral Clustreing Algorithm and to use your algorithm to cluster the observations of the data set.

- `Step 1.` Compute a matrix, called Z containing the computed values of the euclidian distances between all n observations

$$Z = (z_{ij}) \text{ with } 1 \leq i, j \leq n$$

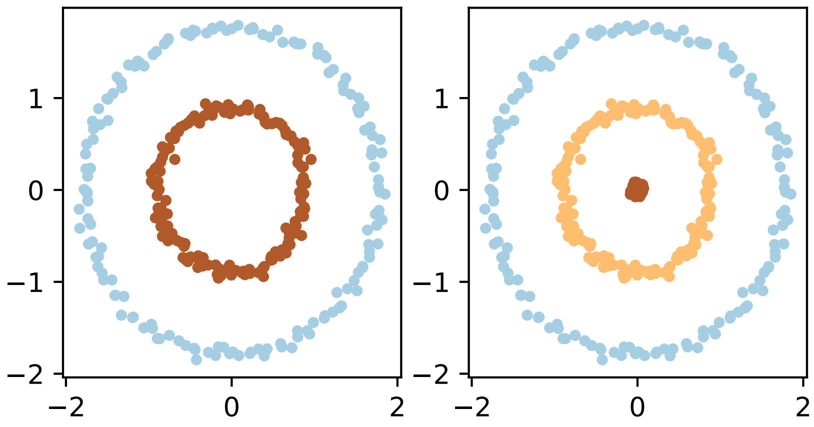

Figure 7: Imbricated circles

```
df = pd.DataFrame(X);
Z=distance_matrix(df.values, df.values);
```

- `Step 2.` Compute the affinity matrix, called W, between all the n observations: $W = (w_{ij})$, $1 \leq i, j \leq n$ such that:

  $$w_{ij} = exp(\frac{-z_{ij}^2}{2\sigma^2})$$

  In this application (for the example with the 3 circular area), the parameter σ is first chosen equal to $\sigma = 0.1$ (do not change this value at this step).

- `Step 3.` Compute the L matrix for the Laplacian graph with the appropriate normalization defined with:

  $$L = I_n - D^{-1/2} W D^{-1/2}$$

where I_n is the (n, n) identity matrix and D is the (n, n) diagonal matrix defined by : $D = (d_{ij})$ with $d_{ii} = \sum_j^n w_{ij}$, $1 \leq i \leq n$.

- `Step 4.` Implement a Singular Value Decomposition (SVD) of the L matrix such that:

$$L = U_n E_n U_n^T$$

where U_n is the n eigen vectors matrix of L and E_n corresponds to diagonal eigen values matrix

$E_n = diag(\lambda_1, .., \lambda_n)$.

$u_j \in \mathbb{R}^n$, is the j^h eigen vector associated with the eigen value λ_j.

$u_j = (u_{ij})\ with\ 1 \leq i \leq n$

The Singular Value Decomposition of the Laplacian can be computed using the following Python instructions:

```
# L denotes the laplacian matrix, as previously defined
U, E, V = np.linalg.svd(L, full_matrices=True)
```

The following graph visualizes the first K eigenvectors. The colors of the indices correpond to the colors used to represent the K circular area data set.

- Store and normalize the first K eigen vectors $u_{(1)}, u_{(2)}, ..., u_{(K)}$ corresponding to the smallest eigen values $0 \leq \lambda_{(1)} \leq \lambda_{(2)} \leq \ldots \leq \lambda_{(K)}$

where K correspond to the assumption of the number of groups in the data set:

$$u_{ij}^* = \frac{u_{ij}}{\sqrt{\sum_{j=1}^K (u_{ij})^2}}$$

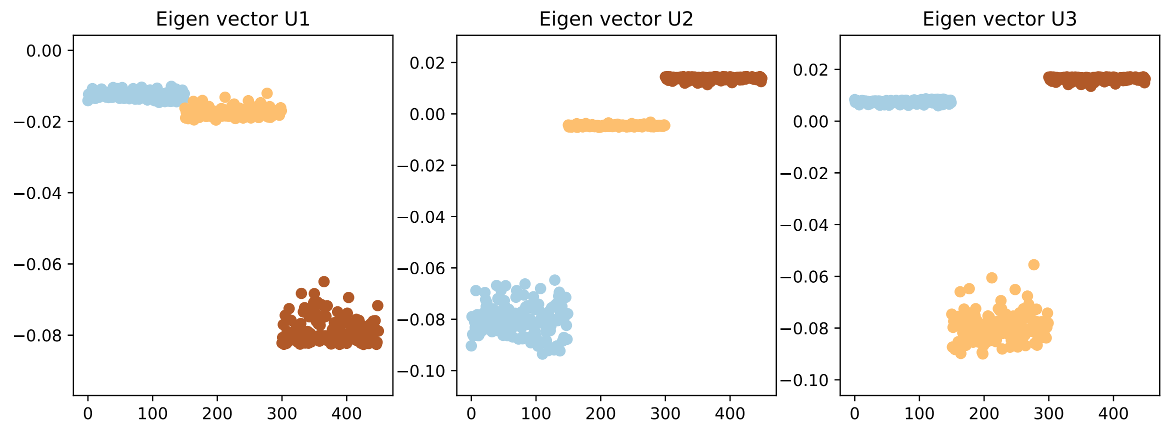

Figure 8: Representation of the first K eigenvectors

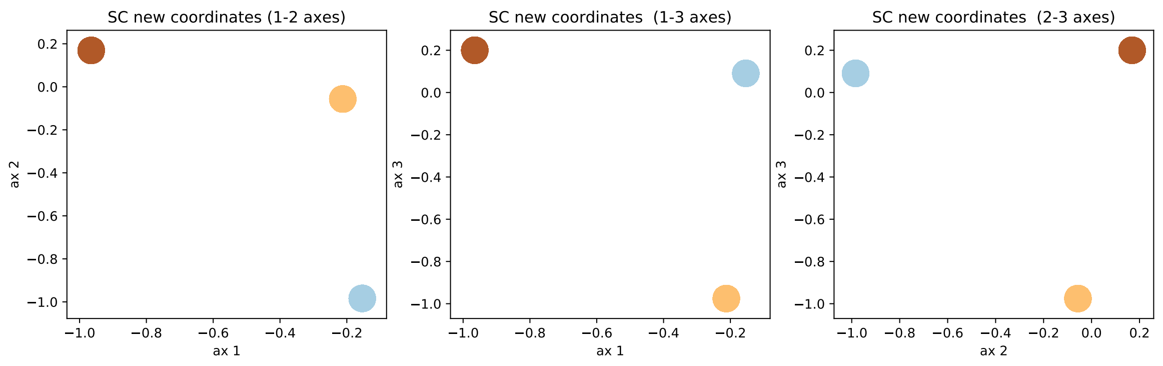

Figure 9: Visualisation of the new representations (coordinates) of the observations based on the new Spectral Clustering representation (1 large circle contains 150 observations)

- Use the *Kmeans* algorithm to cluster the n observations u_i^*, where each observations i is characterized with its K coordinates: $u_i^* = (u_{i1}^*, \ldots, u_{iK}^*)$.
- Using the labels of the K groups computed thanks to the *Kmeans* algorithm, visualize the initial observations and their corresponding labels as represented in following figure.

In order to compare the previous clustering result, apply the *Kmeans* algorithm on the initial raw data. Conclusion .

Impact of the SC hyperparameter σ.

- Apply then the Spectral clustering algorithm as in question **??** but with different values of σ $0.05; 0.5; 1$. Conclusion?

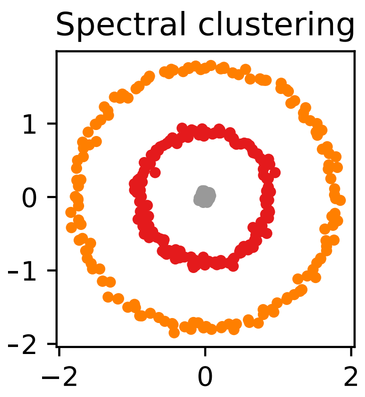

Figure 10: Spectral clustering of the data set: kmeans clustering based on the new coordinates of the data

Example 1

The W affinity matrix with $\sigma = 0.5$ is $W = (w_{ij})$, $1 \leq i, j \leq n$, $w_{ij} = exp(\frac{-z_{ij}^2}{2\sigma^2})$

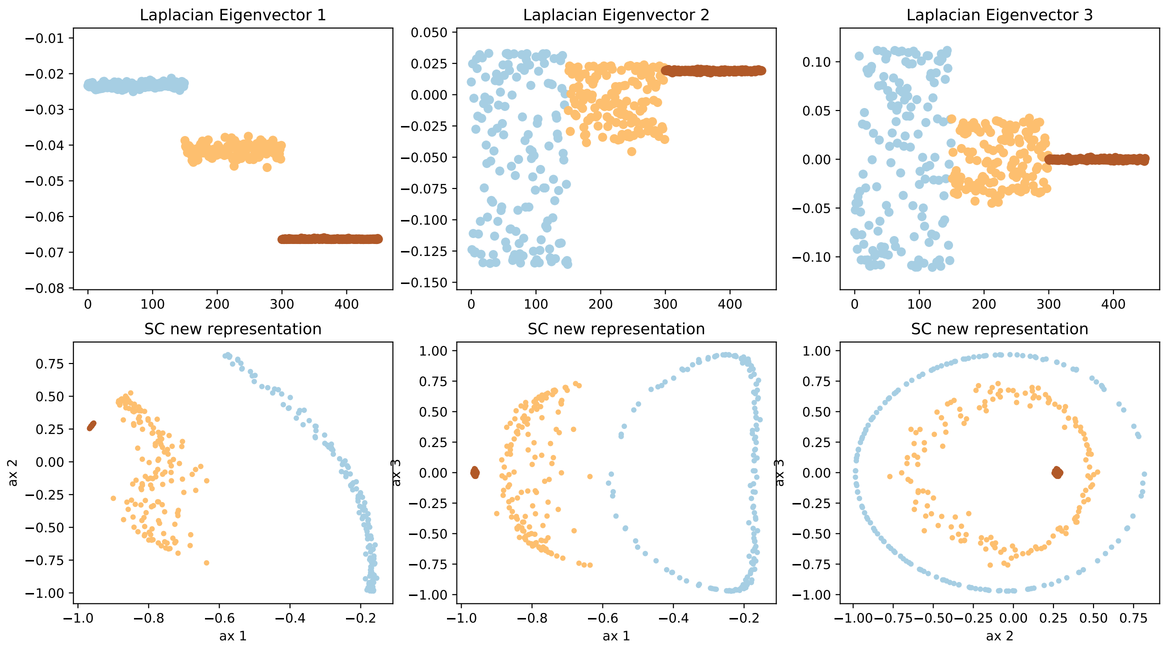

Figure 11: Visualisation of the detailed steps of the Spectral clustering algorithm (given sigma)

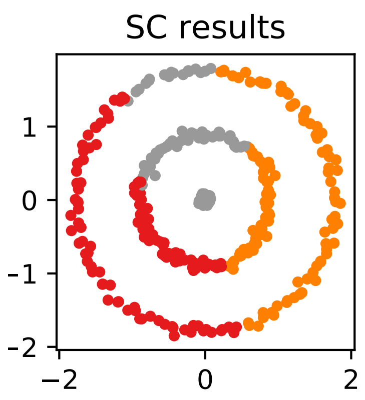

Figure 12: Results of the SC for a given sigma value

Example 2

The W affinity matrix with $\sigma = 0.25$ is $W = (w_{ij})$, $1 \leq i, j \leq n$, $w_{ij} = exp(\frac{-z_{ij}^2}{2\sigma^2})$

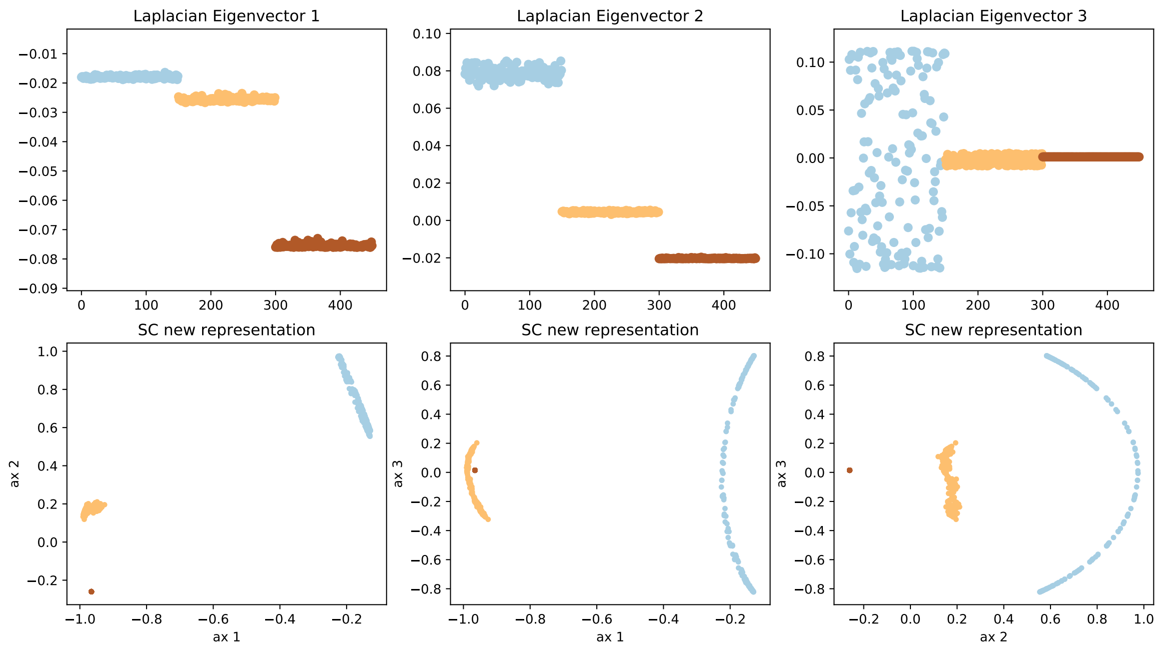

Figure 13: Visualisation of the detailed steps of the Spectral clustering algorithm (given sigma)

Spectral clustering using scikit learn

The Spectral clustering algorithm is available in the Scikit learn library.

```
from sklearn.cluster import SpectralClustering
```

Study the help of the `SpectralClusterin()`function and especially the àffinity‘parameter of the function. Apply this algorithm on the 3 circular imbricated dataset using a fine tuning of the parameters of the Scikitlearn function.

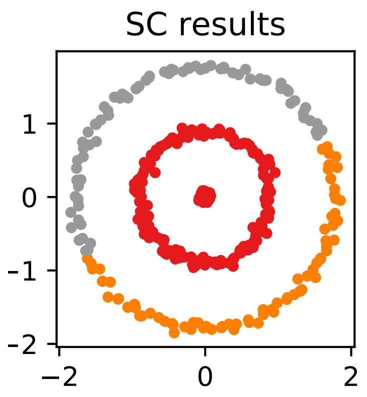

Figure 14: Results of the SC for a given sigma value

Kmeans

The data

The data are stored in the irm_thorax.txt file and represent a human thorax image.

Kmeans algorithm

- Study the help of the `Kmeans()` function of the sklearn.cluster library.

```
from sklearn.cluster import KMeans
modkmeans = KMeans(n_clusters=2, random_state=0)
fitkmeans=modkmeans.fit(X)
```

- Compute a segmentation of the thorax image using the Kmeans algorithm with an assumption of k=2 groups.
- Display the values of the K centroids.
- Compare, using your own instructions, the GMM and Kmeans segmentation of the thorax image as illustrated in the following figure.

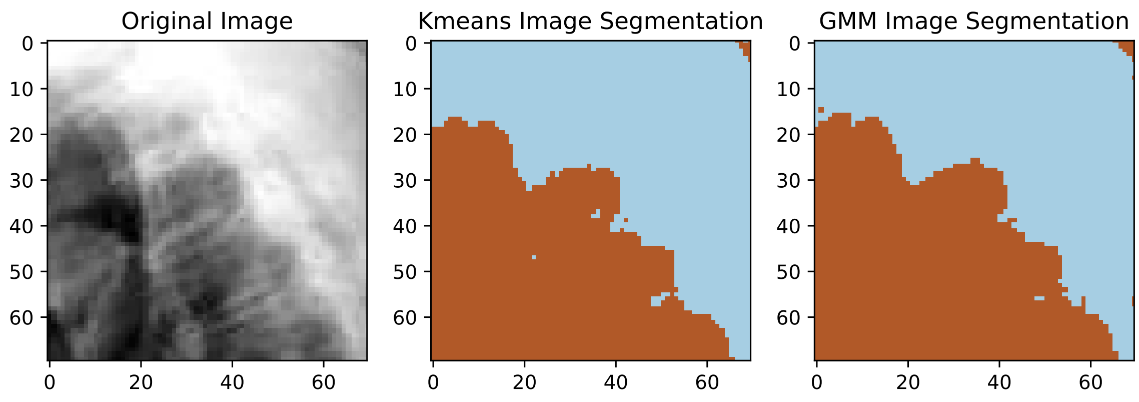

Figure 15: Comparison of Thorax image segmentation with Kmean and GMM algorithms. Assumption: K=2

Hierachical clustering

The data

The data are stored in the irm_thorax.txt file and represent a human thorax image.

HAC algorithm

- Study the help of the `AgglomerativeClustering()` function of the sklearn.cluster library.

```
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import AgglomerativeClustering
modHAC = AgglomerativeClustering(n_clusters=2).fit(X)
```

Between, Within and Total variance

Visualize the evolution of the Within, between and Toral variance for a range of clusters (1 to 10). What is your conclusion

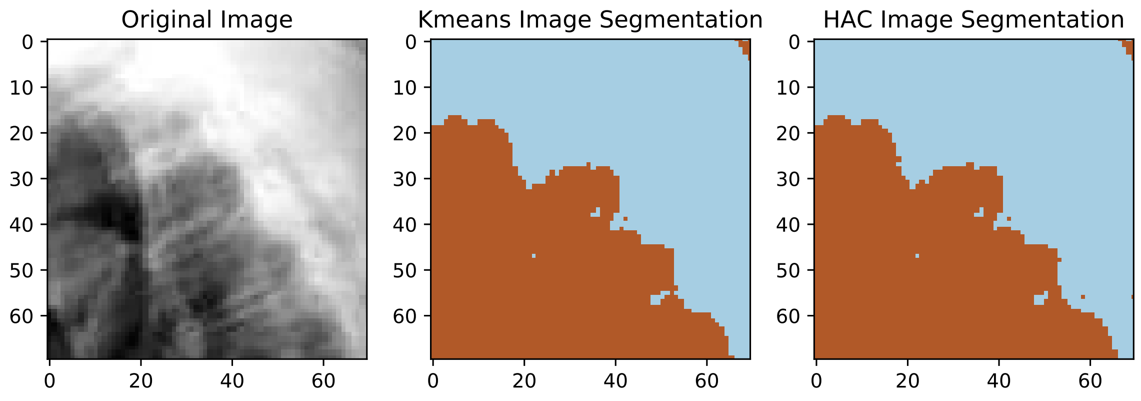

Figure 16: Comparison of Thorax image segmentation with Kmean and HAC algorithms. Assumption: K=2