Homework 2 - Spectral clustering

Computational linear algebra for large scale problems

Gaudino Andrea s346119
Grivet Talocia Lorenzo s346559

A.A. 2024/25

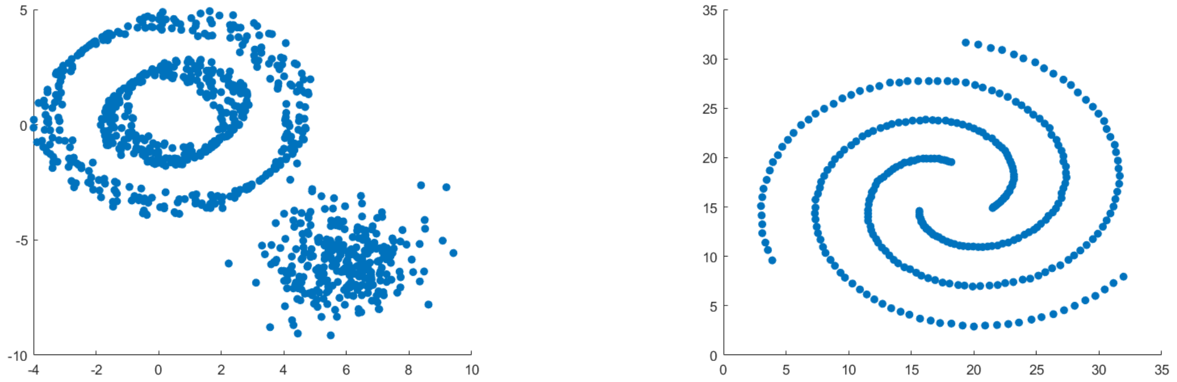
1.1 Introduction

Abstract

The goal of our project is to implement the spectral clustering algorithm on two different data sets, by constructing similarity graphs, computing Laplacian matrices, and analyzing eigenvalues in order to cluster data points effectively, followed by a comparative analysis with other clustering methods.

1.1.1 Data sets description

We perform our analysis on two data sets: *Circle* and *Spiral*. They both have two columns, representing the points in a bi-dimensional space. In the figure 1.1 we display the scatter plot of both data sets.



(a) Circle data set

(b) Spiral data set

Figure 1.1: Scatter plots of the Circle and Spiral data sets.

1.2 Construction of similarity matrix

For each data set, to compute the similarity matrix S that we used the similarity function below, where $s_{i,j}$ represents each entry of the matrix and X_i, X_j are two points in the data set.

$$s_{i,j} = exp\left(-\frac{\|X_i - X_j\|^2}{2\sigma^2}\right),$$

using $\sigma = 1$.

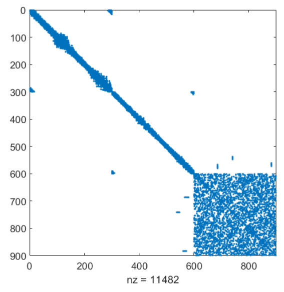
Then, we applied the *k-nearest neighborhood* in order to create the adjacency matrix W: for each row of S we took the k highest values, ensuring that the most similar nodes to the one under consideration were included.

Moreover, the matrix W must be symmetric, so every time we added an element to the position (i,j), we also added the same value to the position (j,i). To ensure that if node i is considered a neighbour of j, then j is also considered a neighbour of i, we could have a number of non-zero elements greater than k in some rows.

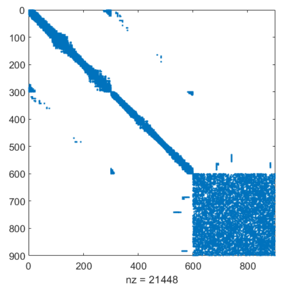
1.3 Degree and Laplacian matrices

In this section we compute D, which is a diagonal matrix where each element (i,i) on the main diagonal is equal to the sum of the values of the *i-th* row of the adjacency matrix W.

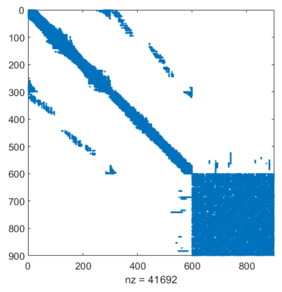
Finally, the Laplacian matrix L is computed as $L = D - W$ In order to improve storage efficiency for these matrices, we used the sparse format. Hereunder we plot the Laplacian matrix.



(a) k = 10

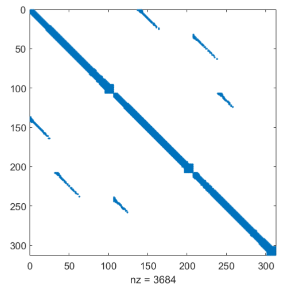


(b) k = 20

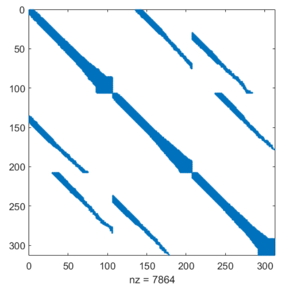


(c) k = 40

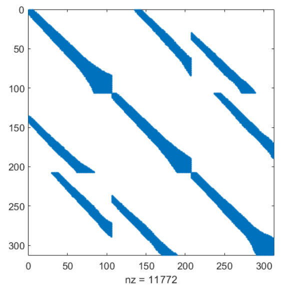
Figure 1.2: Circle data set Laplacian matrix plot for different values of k.



(a) k = 10



(b) k = 20



(c) k = 40

Figure 1.3: Spiral data set Laplacian matrix plot for different values of k.

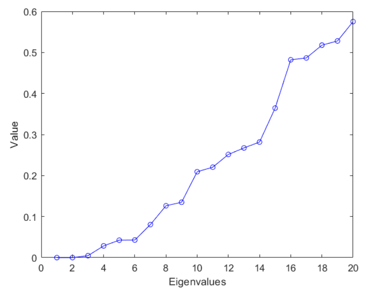
The graphs in figure 1.2 and 1.3 show the correlation between the points in the L matrix. As we expected, most of the values are distributed near the main diagonal.

As we increase the value of k the number of points taken into account rises and this is due to the *knn algorithm* implementation. In the Laplacian matrix each rows has a number of non-zero elements equal to k so that is why in the graphs there is an increasing number of elements (i.e. blue points).

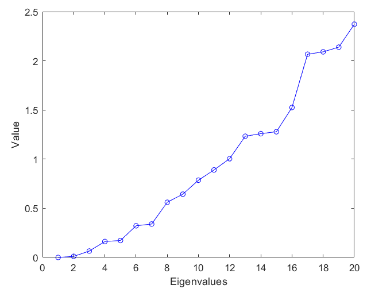
1.4 Eigenvalues analysis

Here, we focus on the clusters' analysis by computing the eigenvalues of L.

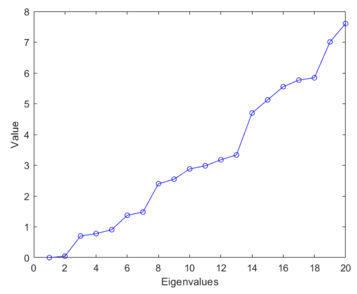
The eigenvalues computation aims to the analysis of the connected components in the graph. Since the elements of D are equal to the sum of the row of W, and L is computed as $D - W$, we know, for sure, that the sum of the values of each row is equal to 0. Proven that, if we multiply L by a column-array of ones, we obtain an array with all entries equal to zero, so at least one eigenvalue must be equal to 0 and its corresponding eigenvector is $[1, ..., 1]^T$.



(a) k = 10

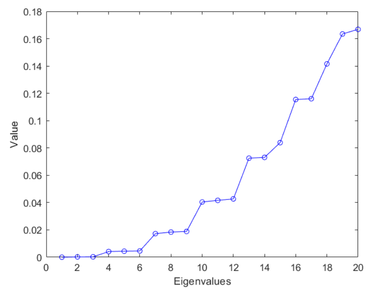


(b) k = 20

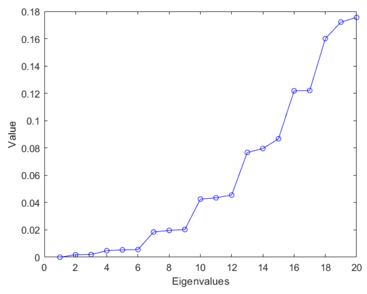


(c) k = 40

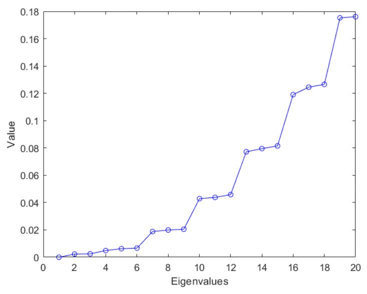
Figure 1.4: Circle data set first 20 eigenvalues plot for different values of k.



(a) k = 10



(b) k = 20



(c) k = 40

Figure 1.5: Spiral data set first 20 eigenvalues plot for different values of k.

Each eigenvalue equal to zero indicates a connected component in the graph. A very low eigenvalue denotes that a group of nodes is weakly connected with the other ones. This is evident in our graphs: the first value is equal to zero, as demonstrated before, the next few ones are very low and close to zero.

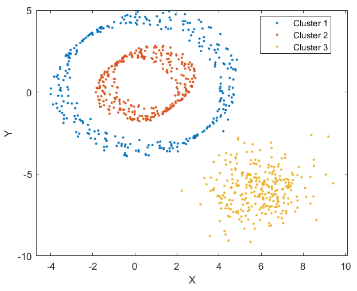
Regarding the circle data set, while conducting the test using $k = 10, 20$ we decided to take into account the first three eigenvalues; increasing k to 40 the eigenvalues we selected were only the first two because the third one was too high with respect of them.

Observing the Spiral data set, the three spirals are equally distant, so we expect that, apart from the first eigenvalue (equal to zero), only the following two should be very low because the points within each spiral are very near to each other. We could not have only one low value because the spirals are equally distant, so it would be difficult to identify only two clusters. However, in the case with $k = 40$ the second and third ones are slightly higher with respect to the previous graph, but it is still insufficient for us to consider only the first eigenvalue for clustering.

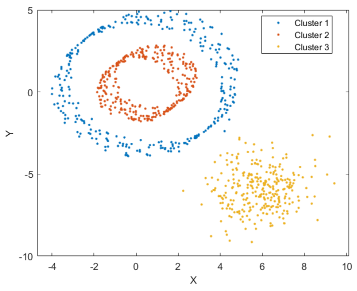
It is evident that selecting an appropriate threshold is essential for the interpretation of clusters. Consequently, for each graph, we identified a threshold in correspondence of the first notable increment between two consecutive eigenvalues.

1.5 Eigenvectors' basis

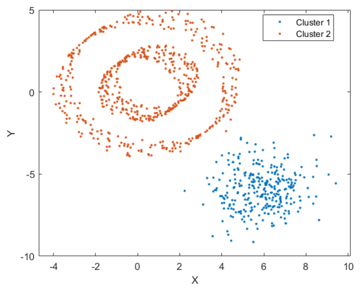
We selected the corresponding eigenvectors to the first n eigenvalues, as said before, for each graph. We stored them in a matrix U with these vectors as columns. Each row of this matrix corresponds to the coordinates of a point with respect to the eigenvectors' basis. We performed the *K-means* algorithm on the rows of U creating n clusters. Below we plot the result of our clustering method.



(a) k = 10

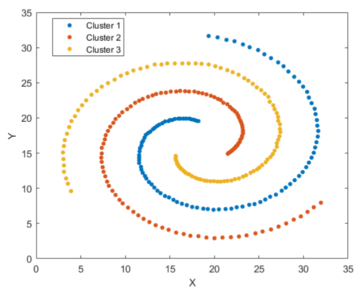


(b) k = 20

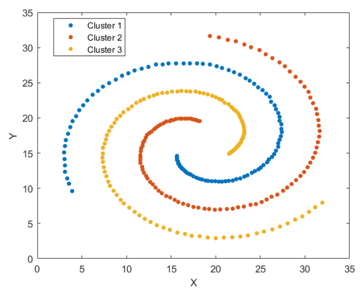


(c) k = 40

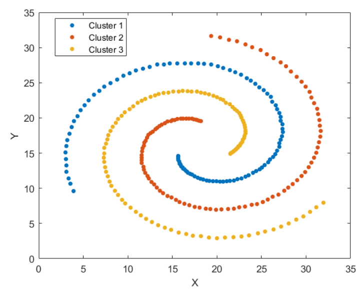
Figure 1.6: Circle data set clusters plot for different values of k.



(a) k = 10



(b) k = 20



(c) k = 40

Figure 1.7: Spiral data set clusters plot for different values of k.

When using k = *10* or k = *20*, the results are almost identical due to the eigenvalue distribution, as the number of values close to zero remains nearly the same.

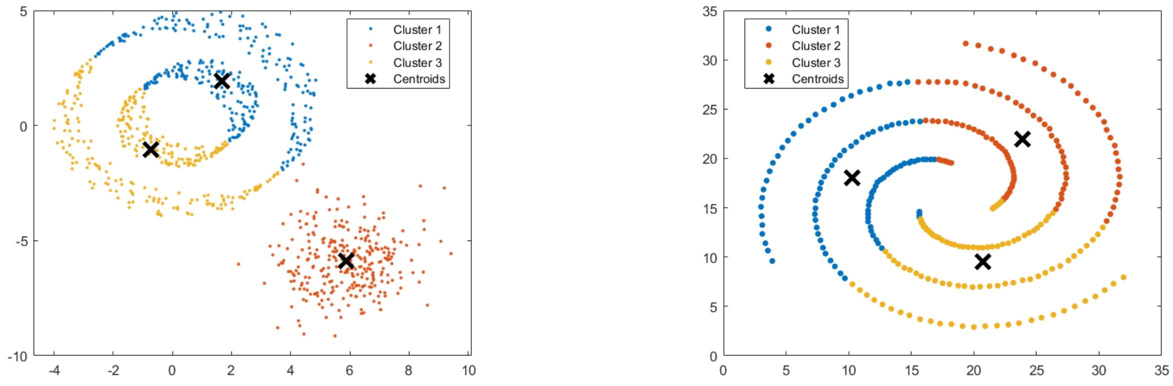
However, there is a slight difference when analyzing the first 40 *k-nearest neighbours*. In the plot of the circle dataset, the two concentric circles are assessed as a single cluster because their points are considered close to one another in the adjacency matrix.

1.6 Other clustering methods

In this section we compute the clusters with other methods.

The main difference is that, here, we do not perform the whole spectral clustering algorithm but we implemented these new methods directly in the initial data sets in order to compare their effectiveness with the outcome of the preceding analysis.

Here, we first present the results of the *K-means algorithm*. We opted for three clusters, as this method requires specifying the number of centroids in advance.



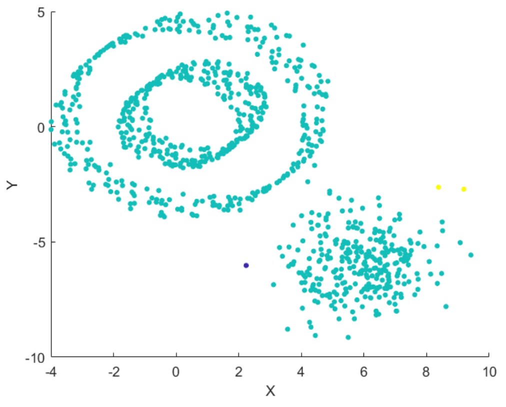
(a) Circle data set (b) Spirl data set

Figure 1.8: K-means algorithm.

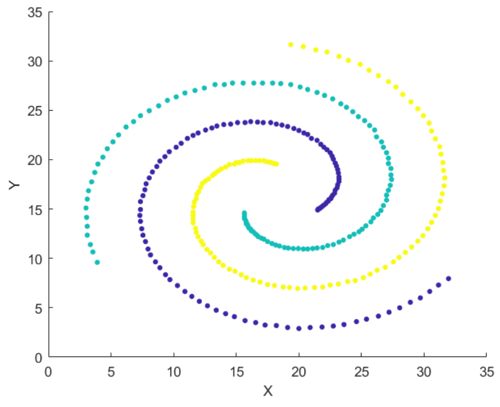
The *K-means algorithm* starts from a random choice of k centroids, then each point is associated to the nearest centroid in order to minimize the variance within each cluster. After that the centroids are computed as the mean of the points of each cluster. This process is repeated iteratively and it stops when between two consecutive iterations the points do not change cluster.

The outcome of this analysis depends on the initial random choice of the centroids so it can lead to a sub-optimal result. In fact, in our analysis the points are arranged in proximity to a centroid and the cluster distribution does not resemble the one computed using the *Spectral clustering algorithm*.

We now display the graphs from the implementation of the *Hierarchical clustering method*.



(a) Circle data set

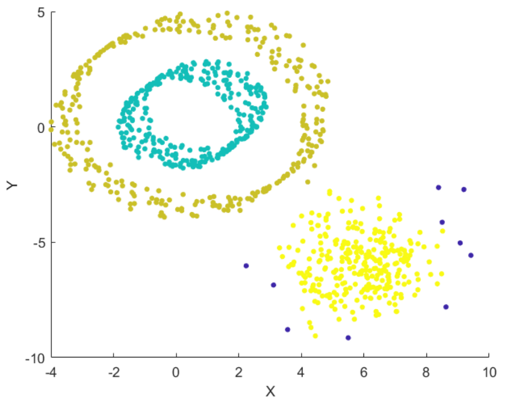


(b) Spiral data set

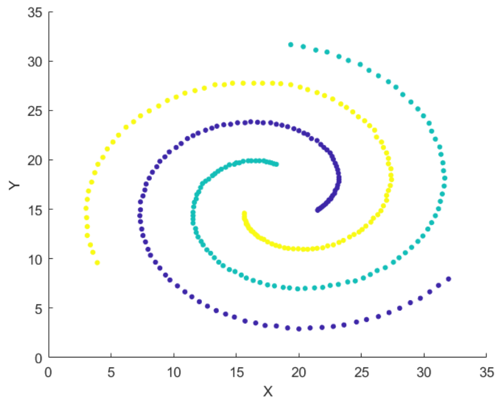
Figure 1.9: Hierarchical algorithm.

Our implementation of the *Hierarchical algorithm* is based on the single linkage method. At each iterative step, the algorithm merges the two clusters with the smallest minimum distance between their points. The process continues until it reaches the number of clusters specified by the user. In our case, we chose three clusters, as this matches the outcome of most of our spectral clustering implementations. The *Spiral* data set shows a cluster distribution identical to the one provided by the *Spectral clustering* method, while the *Circle* one is totally different. This is due to the points of the graph: there are several outliers (i.e. isolated points) so the algorithm tends to consider these points as connection between clusters, so the algorithm merge them erroneously.

Finally, we show the *DBSCAN algorithm*.



(a) Circle data set



(b) Spiral data set

Figure 1.10: DBSCAN algorithm.

The *DBSCAN algorithm* is based on two parameters: maximum distance and minimum number of points. The first specifies the distance within which two points can considered close to each other, while the second defines the smallest number of points required to form a cluster.

The *Circle* data set's outcome closely resemble the one computed with the *Spectral clustering*, this is due to the high density of points is some part of the graph. In fact, the *DBSCAN* is a valuable method for grouping points which are close each other. In the graph there are a few points colored in dark blue: those are the ones that the algorithm could not arrange in any cluster, this is possible because the threshold for the distance is decided in advance so the outliers remain isolated and they do not belong to any cluster.

In the *Spiral* data set the points are more sparse and distant so we had to increase the value of the maximum distance in order to obtain a valid result. In fact, the plot is identical to the one computed when using the *Spectral clustering*.

1.7 Conclusion

In this project, we successfully applied the spectral clustering algorithm to two distinct data sets (*Circle* and *Spiral*). We also compared this method with other common clustering techniques such as K-means, hierarchical clustering, and DBSCAN, finding that spectral clustering provided a more accurate representation of the clusters, particularly in the Spiral data set.

The performance of other algorithms, such as *K-means*, was affected by the initial random centroids, while *DBSCAN* struggled to correctly group the sparsely distributed points in the Spiral data set.

Overall, the spectral clustering algorithm proved to be an effective method for clustering data, especially in cases with complex geometries like the *Spiral* data set.