Spectral Clustering: Implementation and Performance Analysis

July 2023

Lorenzo Bergadano
Politecnico di Torino
lorenzo.bergadano@studenti.polito.it

Giovanni Cadau
Politecnico di Torino
giovanni.cadau@studenti.polito.it

I. Introduction

This report discusses how to complete the Spectral Clustering Homework for the "Computational Linear Algebra for Large Scale Problems" final exam, which was held in the academic year 2022-2023. In this assignment, we'll use the Python programming language to develop the Spectral Clustering approach, an Unsupervised Learning method to categorize unlabeled data.

Our study's empirical goals are to compare the performance of the Spectral Clustering algorithm to that of other clustering algorithms (such as k-Means) in grouping an unlabeled set of points. We also want to determine what factors affect the Spectral Clustering method's performance.

We give a thorough overview of the work in this report: In II, we outline our approach to various tasks. In the III, we present our experimental setup, which includes two different datasets and a third clustering method (k-Means) for comparison. In IV, we examine how the Spectral Clustering technique behaves among the various datasets and how it stacks up against the k-Means algorithm. Finally, we summarize our findings in V.

II. Methodology

A method called Spectral Clustering (SC) was developed initially for graphs. A *connected components* is a group of nodes in a graph where every node has a path to every other node within the group. SC enables us to divide the nodes of a graph into what are known as its connected components. If all nodes of a graph are connected, then there only is a single, big connected component. In order to approximate the graph by separating those two "not-perfectly connected components" but rather "approximated connected components," SC seeks to identify related components that only have a "weak" relationship between them. SC does this by making use of a basic fact, namely, that the graph may be broken into linked components by the lowest eigenvectors of the Laplacian matrix related to the graph. If there is a graph representation of the points, the SC may also be applied to a collection of n-dimensional points.

We move on to explaining how we implemented the full SC algorithm in the programming language MATLAB by presenting the steps we followed, according to the instructions provided in the Homework assignment.

1) The first step consists in, starting from the set of points, constructing the k-nearest neighborhood similarity graph associated with the points. To do this, we define the nodes of the graph as the points in the set, and their edges (i.e., their connections) as numbers based on the value of the similarity between the points, calculated as follows:

$$s_{i,j} = \exp\left(-\frac{||X_i - X_j||^2}{2\sigma^2}\right), \text{with } \sigma = 1 \qquad (1)$$

where X_i, X_j are two points, and $s_{i,j}$ is the weight of the edge between node i and node j. Note that values of $s_{i,j}$ close to 1 indicate higher similarity between the points, while values closer to 0 indicate higher dissimilarity. Actually, we would like to work with sparse matrices, as datasets of points can be arbitrarily large, and for this reason we introduce a simplification: we do not take all the $s_{i,j}$, but rather the more "significant" values. To accomplish this, we only take, for each point i, its k-nearest neighbours based on the specified similarity function, filtering as follows:

$$s_{i,j} = \begin{cases} s_{i,j} & \text{, if } s_{i,j} \in \text{k-highest values} \\ & \text{among } \{s_{i,1}, s_{i,2}, \dots s_{i,n}\} \\ 0 & \text{, otherwise} \end{cases}$$

A further simplification is introduced:

$$s_{i,j} = s_{j,i}, \forall i, j$$

i.e. the graph is undirected.
To conclude this point, we must compute the adjacency matrix, which contains all zeros along the diagonal, whereas outside it contains the $s_{i,j}$. To accomplish this task, we implemented the following function:

- **similarity(X, K, sigma)**: This function calculates the adjacency matrix based on pairwise distances between points in a matrix X, considering only the k-nearest neighbors for each point. It takes the following arguments:

- X (nd-array): matrix containing points' coordinates. Rows represent points, and columns represent coordinates
- K (int): Number of nearest neighbors to consider.
- sigma (float): Value of sigma.

The function first create an empty sparse matrix W which is the adjacency matrix. Then, it finds the K-points with the highest similarity (based on Euclidean distance) for each point by sorting the similarities and selecting the k highest indices. Next, it transforms the distances using the kernel given by the problem (1). Finally, it constructs the adjacency matrix as a sparse matrix where the (i, j)-th element is set to the corresponding pairwise distance if j is one of the k-nearest neighbors of i; otherwise, it is set to 0. The function returns the adjacency matrix.

2) Next, we calculated the Degree Matrix and the Laplacian Matrix through the following functions:
   - **D=diag(sum(W,2))**: This function calculates the degree matrix D using the adjacency matrix W. The function first calculates the sum of weights (or similarities) for each point by summing the elements in each row of the adjacency matrix. Then, it constructs the diagonal matrix D as a sparse matrix where the (i, i)-th element is set to the sum of weights for the i-th point. Finally, it returns the diagonal matrix D.
   - **L = D - W**: This function calculates the Laplacian matrix L by subtracting the adjacency matrix W from the diagonal matrix D. It takes the following arguments:
     - A (sparse matrix): Adjacency matrix.
     - D (sparse matrix): Diagonal matrix.

     The function simply subtracts the adjacency matrix W from the diagonal matrix D. It then returns the Laplacian matrix L.
3) To compute the number of the connected components, we simply calculated the first smallest eigenvalues (10) of the Laplacian matrix. From the theory we know in fact that the number of connected components is approximately equal to the number of eigenvalues of the Laplacian matrix having a value near zero. Specific results regarding the number of connected components of the different datasets are presented in Section III.
4) To compute the smallest eigenvalues of the Laplacian matrix, we used the *eigs()* [1] function using the 'smallestabs' as a type of returned eigenvalues ($\sigma = 0$) and a sorting function based on the values of the diagonal matrix builded with the smallest eigenvalues as diagonal. Then, the value of M for the Spectral Clustering algorithm has then been chosen: as we know from the theory, the eigenvalues of the Laplacian that are 0 or very close to 0 indicate that the graph can be split in connected components, whose number is given exactly by the number of near to zero eigenvalues of the Laplacian. So, what we did is to define a function that calculates first the average of the 10 smallest eigenvalues, and then sets a threshold at the value of the average divided by a constant value, to be empirically determined, which we set at 2.5:
$$ths = \frac{\hat{\lambda_1} + \cdots + \hat{\lambda_{10}}}{10} * \frac{1}{2.5}$$
Then, each eigenvalue under the threshold is considered zero.
5) Next step is to calculate the corresponding eigenvectors of the Laplacian: they will indicate us the "cuts" that have to be done in the initial graph in order to separate it in its M connected components. We calculate the eigenvectors associated with the smallest eigenvalues again both with the *eigs()* [1] function. In this step, we also construct the matrix U, whose columns are given by the M found eigenvectors. Note that the matrix U has N rows, with N number of points in the initial dataset and M rows, where M is the number of found connected components.
6) We can think of the process we did up to now as a pre-processing of our initial data, as we transformed the starting dataset, which was of dimension NxD, with $D = 2$ number of dimensions of the data, to a dataset of dimension NxM, with M number of connected components of the graph associated to it. Here, we simply apply any clustering algorithm to the rows of this new matrix (we apply the KMeans function of the "Statistics and Machine Learning" library for MATLAB).
7) Then, we take the column containing the labels obtained from this clustering and simply copy-paste it into the initial dataset, to find clusters of the original points.
8) We plot the results of the clustering, along with other diagnostics figures in IV.
9) Finally, we apply the k-Means algorithm directly to the original set of points to compare and inspect the differences with the results obtained with Spectral Clustering.

III. Experimental Setup

We present here the components of our experimental setup:

1) We have two different datasets to cluster: "Circle" and "Spiral". Both are 2-D datasets, so we can easily visualize them in Fig. 1. The first is made of 900 points, the second 312. The first does not have the correct clusters indicated, while the second does.
2) We start by constructing the corresponding k-nearest neighborhood similarity graph for our two datasets, trying out different values for k (number of considered neighbours): 10, 20 and 40. We can visualize the results in the first columns of the graphs in Fig. 2 for the Circle dataset and in Fig. 3 for the Spiral. We can immediately get a sense of what constructing the k-

nearest neighborhood similarity graph means: we are drawing edges between the points, based on whether they are similar to each other, according to Equation (1). We can also see how increasing the value of k leads to having more connections in the graph.

3) Then, we calculate the Laplacian matrix and plot its 10 smallest eigenvalues in the second columns of Figures 2 and 3. We also plot, in these graphs, a vertical line corresponding to the last eigenvalue that is comprised within the threshold defined. In other words, the vertical line indicates the number of connected components in the graph.
4) Continuing in our experimental setup, we plot, in the last columns of Figures 2 and 3, the clusters obtained by Spectral Clustering.
5) To conclude our experimental setup, we perform the clustering of the initial dataset using a k-Means method directly to the set of points, and plot the results for comparison. The results are contained in the graphs of Figure 4. This result is obtained by applying an evaluation of clusters based on the Calinski-Harabasz criteria [2].

IV. Discussion of results

We summarize in this section our findings and interpretation of the results, by answering the following questions.

- **What are the effects on Spectral Clustering of using different values of k-nearest neighbours?** Each row in Fig.2 and 3 is referenced by a specific value of k, where k is the number of neighbors taken into account. Let's examine the clustering shown in Fig.2 starting with the Circle dataset. Let's humanely identify the "inner circumference," "outer circumference," and "small circle on bottom right" three groupings of points in this dataset. The clustering that is obtained by choosing $k <= 20$ neighbors appears to be the most visually significant, while the other choices of "k" above result in too few clusters. Let's look at the graph that corresponds to it. We can see that there are connections in the region around the points on the outside circle that were designated as cluster 1; in fact, they become more obvious when $k = 40$. The graph for $k = 10$ confirms that the two halves of the outer circle do not appear to be "heavily connected." These behaviors may have a small but discernible impact on the graph's Laplacian matrix, as we can see in the accompanying eigenvalues plot where the blue line begins to rise from zero around the values of λ_3 and λ_4. The previously defined threshold has detected the point of λ_2 as the first eigenvalue significantly different from 0. In that case, the $\approx 20\%$ of the points of the outer circumference may be incorporated by the inner circumference, and the two halves of the outer circumference may be connected into one single cluster, as their light connection is still heavier that the connection there is between the outer circumference and the small circle in the bottom right of the graph.
- **What are the effects on Spectral Clustering of initial dataset distribution?** Here, we'd like to talk about potential Spectral Clustering algorithm best practices for a certain dataset. Figures 2 and 3 show the findings, and when a sufficient value of k is chosen, we observe that the algorithm appears to accurately detect clusters in both scenarios. In reality, for unsuitable values of k, both approaches have a tendency to separate the points incorrectly, but we point out that the value of k is chosen as part of the algorithm and is sometimes referred to as a "hyperparameter" in the field of data science. As a result, we do not explicitly state that the approach is best for a particular dataset, indicating that it may be appropriate for a variety of datasets.
- **How does Spectral Clustering compare to k-Means Clustering?** Finally, we compare the results with those of a k-Means algorithm applied directly to the initial dataset. We can visualise the results in Figure 4, and immediately see that, once the right number of clusters has been visually selected, the clustering performed succeeds in correctly extracting the point patterns for both the Spiral and Circle datasets; however, the algorithm fails to obtain the separation between the inner and outer circle, which the Spectral Clustering managed to find instead. It is important to note that the selection of the best model was done by applying the Calinski-Harabasz criterion [1], which is based on intra and inter clusters variance.

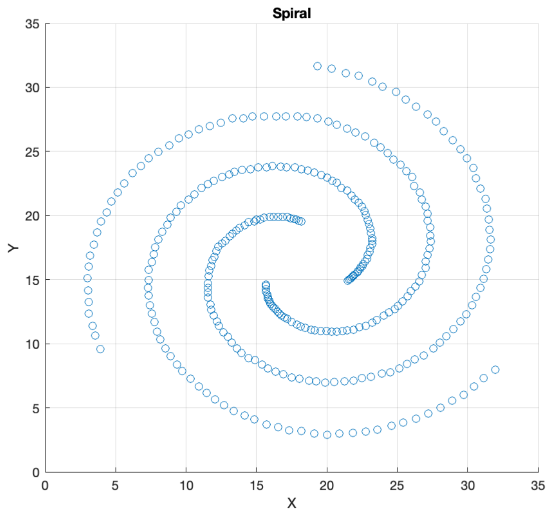
V. Conclusion

We constructed the Spectral Clustering method, evaluated its effectiveness, and identified the variables that have the greatest influence on it. We discovered that the final clustering certainly depends on the number of neighbors used to build the associated graph. The distribution of the datasets doesn't appear to have an impact on the algorithm's performance, although more testing with different datasets might be done. Finally, we evaluated the Spectral Clustering's superior performance over the k-Means clustering technique, at least in these two datasets.

References

[1] Eigs function, MATLAB Documentation Subset of eigenvalues and eigenvectors https://it.mathworks.com/help/matlab/ref/eigs.html.

[2] Calinski-Harabasz Criterion, MATLAB Documentation Calinski-Harabasz Criterion for clusters evalutation https://it.mathworks.com/help/stats/clustering.evaluation.calinskiharabaszevaluation.html#bt07j9e.



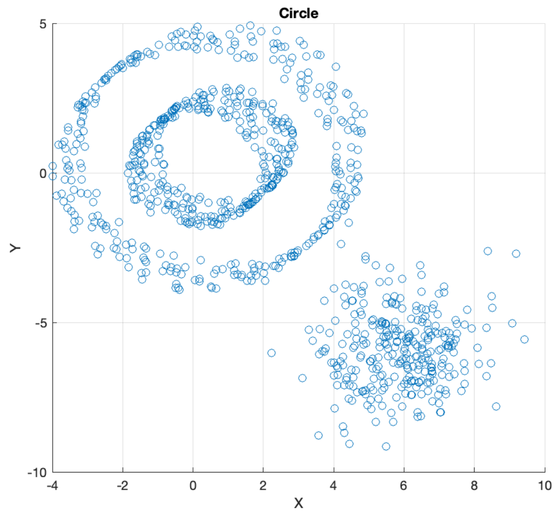


Fig. 1. Visualization of the two datasets: Circle and Spiral.

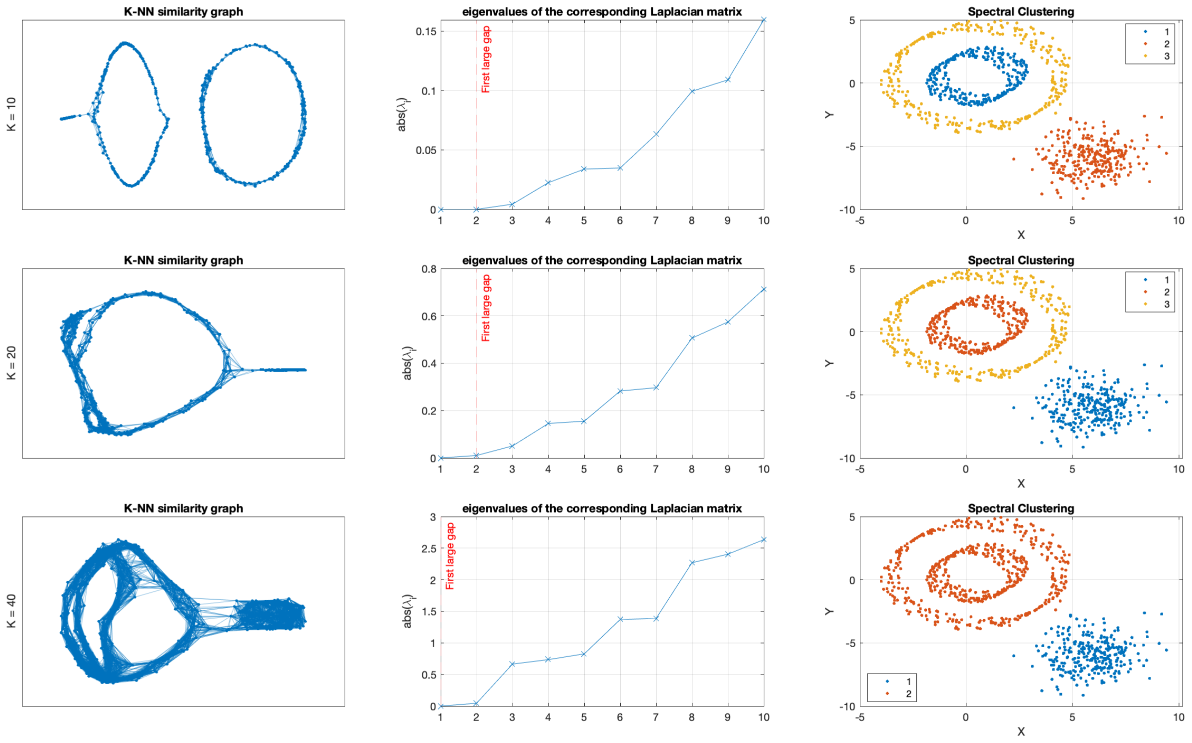


Fig. 2. Spectral Clustering workflow for the Circle dataset: k-nearest neighborhood similarity graph, plot of the eigenvalues of the Laplacian matrix and final clustering of the points.

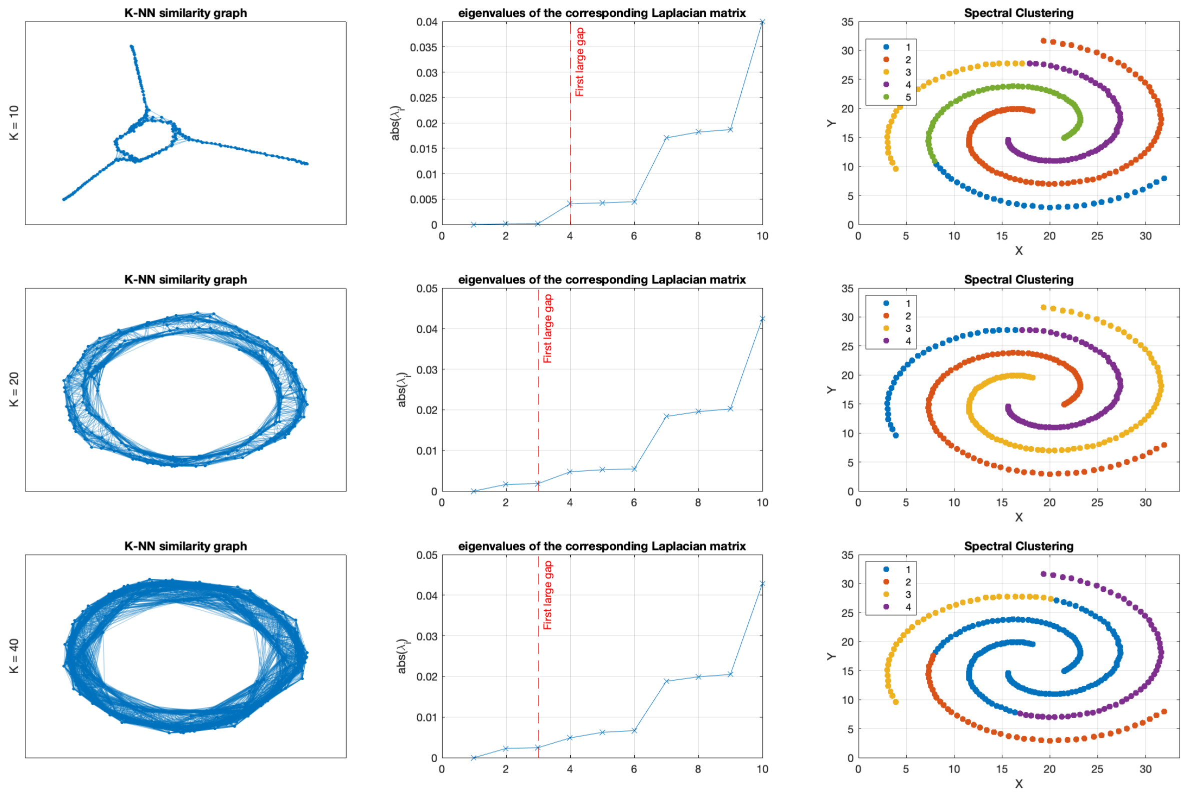


Fig. 3. Spectral Clustering workflow for the Spiral dataset: k-nearest neighborhood similarity graph, plot of the eigenvalues of the Laplacian matrix and final clustering of the points.

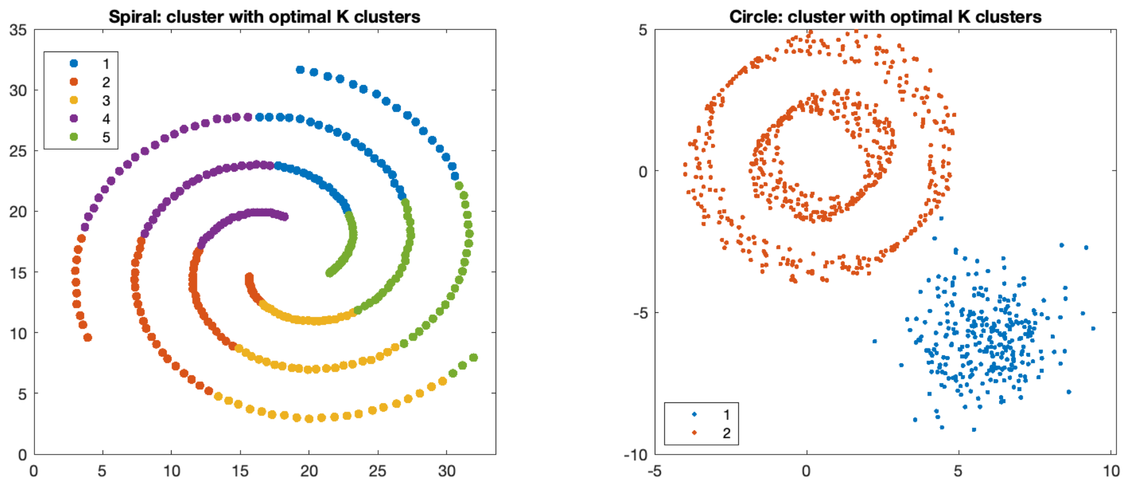


Fig. 4. kMeans Clustering of the two datasets.