Spectral Clustering

Luciano Scarpino
Salvatore Nocita

1 Spectral Clustering

Clustering is one of the most widely used and studied techniques in data analysis. Its application helps to uncover hidden patterns within a dataset's structure that might not be immediately visible through manual inspection. These connections reveal significant properties about the nature of the objects in a collection.

Clustering has applications in numerous fields, including *statistics, computer science, mathematics, biology, and social sciences.* Various algorithms and heuristics exist in the literature, each offering different advantages depending on the context.

This document describes the implementation of Spectral Clustering, a method that leverages graph theory to divide dataset instances into groups called clusters. It will outline the logical and instrumental steps involved in applying this technique and analyze the obtained results, commenting on its strengths and limitations.

2 Dataset

The datasets used to test the implementation of the algorithm are synthetic, meaning they do not correspond to real-world observations. Instead, the distribution of points within them, represented by feature values, is designed to form patterns that are easily recognizable by the human eye.

This characteristic allows for an intuitive visual evaluation of the clustering results, avoiding the need to rely exclusively on potentially misleading numerical metrics—though some of these metrics will be included at the end of the document.

The datasets considered are presented below:

- **Circle.csv**: As the name suggests, the alignment of data within this dataset, which can be easily visualized through a scatter plot, forms two concentric circles positioned in the upper-left corner of the figure. On the right, a cloud of points can be observed without any particular pattern. It is important to note that the points in the two clusters tend to be very close to each other near the boundary separating them, making classification more difficult.

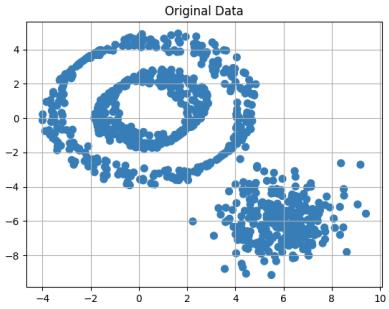

Figure 1: Circle Dataset Distribution

- **Spiral.csv**: This dataset contains point coordinates forming three concentric spirals. In the scatter plot (placeholder), it is evident that the three structures are particularly isolated, unlike the "Circle.csv" dataset. However, the intertwining nature of the spirals makes their separation more complex for traditional algorithms such as k-means, whose results will be analyzed later.

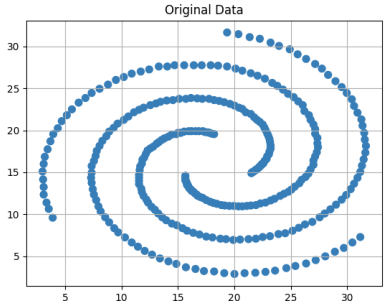

Figure 2: Spiral Dataset Distribution

- **3D Dataset.csv**: For additional model testing, we added a third dataset consisting of three-dimensional points. To keep things simple, it's a toy dataset that also includes labels; in fact, there are five clusters, with two of them being very close to each other. Visually, there isn't any specific shape—just five clumps in 3D space.

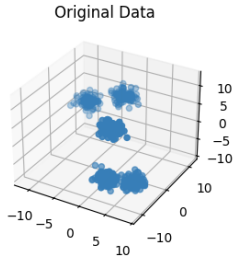

Figure 3: 3D Dataset Distribution

3 Graphs

A graph is a discrete mathematical structure consisting of two sets: nodes and edges. More formally, a graph is an ordered pair $G = (V, E)$ where V is the set of nodes and E is the set of edges. An edge represents a connection between a pair of vertices. The properties of the edges define the nature of the graph. If the edges represent symmetric relationships between vertices, the graph is called *undirected*. Conversely, if the edges have a direction, the graph is called *directed*. Furthermore, edges can have weights that determine their importance within the graph. A graph with weighted edges is called a *weighted graph*. Finally, a graph is said to be *connected* if, for every pair of vertices in V, there exists a path connecting the two nodes. Similarly, a *connected component* of a graph is a subgraph in which any two nodes are connected, and no additional nodes in the larger graph are connected to the subgraph.

The clustering problem can be formulated using a *similarity graph*. In this mathematical representation, each dataset point is represented as a graph node, while the edges connecting them are weighted using a similarity function. The similarity function is a scalar relation that assigns each pair of points x and y a value quantifying their similarity. There are multiple ways to define such a function, but in this study, the following definition is used:

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

where σ is set to 1.

Spectral Clustering utilizes this dataset representation to first identify neighborhoods consisting of the k closest points in terms of similarity for each instance in the dataset. Then, it derives the connected components of the graph and computes a projection of the data into a reference vector space where clusters are more easily visualized and grouped using algorithms such as k-means.

There are multiple variations of the Spectral Clustering algorithm, primarily differing in how the **Laplacian matrix** is defined. In the following discussion, two variants will be implemented: **unnormalized spectral clustering** and

normalized spectral clustering. The fundamental difference between these approaches lies in the form of the Laplacian matrix used during execution.

Below is the pseudocode for the both version:

```
1. Construct the similarity graph from the dataset.
2. Compute the unnormalized/normalized Laplacian matrix L.
3. Compute the eigenvalues and eigenvectors of L.
4. Select the k smallest eigenvectors to form a new feature space.
5. Apply k-means clustering to the transformed data.
6. Assign each data point to a cluster based on k-means output.
```

4 Definition of Laplacian Matrix

The first step in spectral clustering involves defining the Laplacian matrix, the Degree matrix, and the Weighted matrix. The Weighted matrix corresponds to the connectivity matrix B, but with weights assigned to each connection between vertices. The weight definition follows a chosen similarity criterion. In this study, similarity is defined as follows:

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

where σ is set to 1, and the distance between nodes is Euclidean.

Using this similarity metric, the similarity matrix allows us to determine which nodes are "similar" to each other. The greater the similarity (i.e., the weight assigned to each connection), the more similar two nodes are. This concept is used in K-Nearest Neighborhood (KNN), where k specifies the relevant values of interest. In this study, neighborhoods with $k = 10, 20, 40$ are explored.

An important aspect of defining the similarity matrix concerns its symmetry: in spectral clustering processes, similarity is fundamental as it ensures real eigenvalues and eigenvectors forming an orthonormal basis, which is essential for dimensionality reduction.

To ensure symmetry in the weighted matrix, we use a symmetrization technique by addiction: exploring the k-neighborhood of node i, if node j belongs to it, it has been checked whether node i also belongs to the k-neighborhood of node j. If this condition is not met, we set j in the neighborhood of i with value of i.

After replacing the main diagonal of the weight matrix (assigning a unitary n-dimensional vector), the Degree matrix is computed by summing the weights along the rows: the Degree matrix expresses the degree of a node (corresponding to the i-th row), i.e., the sum of the number of incoming and outgoing edges. The result is a diagonal matrix where each entry is the sum of the weights along the row.

At this point, the Laplacian matrix is obtained simply by subtracting the two matrices:

$$L = D - W$$

Taking a step back, we highlight properties of the incidence matrix: the Laplacian matrix can be seen as the difference between the Degree matrix and the unweighted Connectivity matrix, which corresponds to AA^T. Focusing on the matrix A^T, it describes the given graph: by studying this matrix through its decomposition (e.g., SVD), we can compute vectors belonging to the kernel of the matrix, which always has a dimension greater than 0. Since the incidence matrix consists of 1 and -1 symmetrically placed around the diagonal, the sum along its rows is zero. Consequently, if A is multiplied from the left by a compatible identity matrix, a null matrix is obtained.

Returning to $\ker(A^T)$, it is composed of eigenvectors corresponding to zero eigenvalues. Referring back to the graph representation, each nonzero entry in each eigenvector represents a directed edge connecting two vertices. From this perspective, the path described by an eigenvector forms a loop in the graph, meaning a connected subgraph of the graph.

Beyond the loops described by the eigenvectors in the matrix kernel, other loops exist within the same graph, but they are not linearly independent of the former.

Thus, we observe the relationship between eigenvectors belonging to the kernel of the matrix and the connected components of the graph. Additionally, the number of eigenvectors in the kernel represents the geometric multiplicity of the eigenvalue 0. As a direct implication, we conclude that the number of connected components in a graph is given by the geometric multiplicity of the zero eigenvalue.

4.1 Properties of the Laplacian Matrix

The following are fundamental properties of the Laplacian matrix L:

- L is symmetric and positive semi-definite (SPD) since W is symmetric and D is diagonal.
- The smallest eigenvalue of L is $\lambda = 0$, and its corresponding eigenvector is the unitary n-dimensional constant vector. This property confirms that the graph is at least fully connected.

4.2 Normalized Laplacian Matrix

The normalized Laplacian matrix is a variant of the previously described Laplacian matrix. It is often used in machine learning, especially in spectral graph analysis. This differs from the standard Laplacian because it is scaled to eliminate the influence of degree variations among the nodes of the graph. Below is the normalized and symmetric form:

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \leq 2$$

The eigenvalues of the normalized Laplacian matrix are always within the range $[0, 2]$ because:

- The matrix is symmetric and positive definite as it is derived from the Laplacian matrix. Consequently, it is known that its eigenvalues are real and greater than or equal to 0. - The matrix $S = D^{-1/2}WD^{-1/2}$ represents a transition matrix for a random walk on an undirected graph, in addition to being symmetric. It is demonstrable that its eigenvalues lie within $[-1, 1]$.

Thus, we can write $L_{\text{sym}} = I - S$ and applying this to the eigenvalues, we obtain:

$$\lambda = 1 - \mu$$
$$1 - 1 \leq \lambda \leq 1 - (-1)$$
$$0 \leq \lambda \leq 2$$

These values have specific meanings:

- $\lambda = 0$ implies that the graph has a connected component (corresponding to a constant eigenvector equal to one). - $\lambda = 2$ implies that the graph is bipartite, meaning there exist two disjoint groups of nodes where connections occur only within the respective groups.

A small clarification on why the eigenvalues of matrix S are within $[-1, 1]$: the transition matrix of a random walk is a stochastic matrix, as it is similar to the Markov matrix $P = D^{-1}A$, and therefore shares its eigenvalues. Additionally, matrix S is symmetric by definition, meaning its eigenvalues are real.

Each entry P_{ij} of the Markov matrix corresponds to the probability of transitioning from node i to node j in a single step since the degree matrix D distributes probability uniformly among its neighbors. The maximum eigenvalue of P, and consequently of S, is 1 because the sum of the values along the i-th row of P equals 1. Moreover, since S is both stochastic and symmetric, its norm is at most 1. Therefore, the eigenvalues of matrix S range between -1 and 1.

Inside of code, has been defined Normalized Laplacian matrix as definition, leading similar results as before.

5 Number of Connected Components

As stated earlier, the number of connected components in the graph corresponds to the geometric multiplicity of the zero eigenvalue. A good approach to evaluate this is to compute and sort the eigenvalues of the matrix in ascending order and count how many times the eigenvalue 0 appears. This is the approach taken in this study.

Using a greedy heuristic, we choose to compute 5 eigenvalues ($M = 5$): by graphically displaying the eigenvalues, i.e., plotting the eigenvalue index on the x-axis and its corresponding value on the y-axis, we determine for each case ($k = 10, 20, 40$) the number of connected components in the graph.

For the "Circle" dataset, the number of connected components for $k = 10, 20, 40$ are three, three, and two, respectively. For the "Spiral" dataset, the number of connected components in the three neighborhood cases are three, one, and one, respectively.

A crucial remark: due to numerical cancellation errors and machine precision issues, a connected component is identified when an eigenvalue is close to 0 within a given tolerance. The tolerance threshold could be optimized as a hyperparameter, but in this study, 0 is defined based on an interpretation of the described graph. The eigenvalue curve exhibits a significant "elbow point": before the elbow, the eigenvalues tend to be comparable (except for steeply increasing curves before the elbow), while after the elbow, a rapid increase is observed.

As a general rule (but not an absolute one), eigenvalues before the elbow are considered to be 0.

6 Methods for Eigenvalues

6.1 Rayleigh Quotient

The Rayleigh Quotient is a scalar function widely used to understand the relationship between a vector and a square matrix through eigenvectors and eigenvalues. Specifically, R_A is defined by a matrix and its eigenvectors, and its eigenvalues are the critical points. The definition is as follows:

$$R_A = \frac{v^T A v}{v^T v}$$

The numerator is a quadratic form, while the denominator is the squared norm of the vector v. If an eigenvector of the matrix A is passed to the function, the corresponding eigenvalue for that eigenvector is obtained.

The Rayleigh Quotient will be used by the following methods as a tool for calculating the eigenvalues of a matrix.

The methods explored below will focus only on those belonging to the family of eigenpair approximators, which target one eigenpair at a time. However, there are other methods, such as the QR method, that compute all eigenpairs.

6.2 Power Method

The Power Method is based on calculating the dominant eigenvalue (i.e., the eigenvalue with the largest magnitude in absolute value) of a matrix, assuming the eigenvalues can be ordered in descending magnitude and that the first is strictly greater than the second. Essentially, given any vector v, it can be expressed as a linear combination of the linearly independent eigenvectors of the matrix. Iteratively, a series of vectors can be written, each as a combination of the previous one, until the desired result is achieved. Based on this concept, the Power Method approximates the eigenvalue with the largest magnitude using

the Rayleigh Quotient: at each iteration, the method approximates the largest eigenvalue. When a vector v is applied to a matrix A, it is projected (i.e., stretched or compressed) by a factor λ. As v approaches an eigenvector of the matrix A, R_A converges to the eigenvalue which, when applied to v, approximates the eigenvalue λ. For this reason this is called *Couchy sequence*.

6.3 Inverse Power Method

Based on what has been said, it can be assumed that there exists an eigenvalue that is strictly smaller than the others. Thus, the inverse power method follows the logic that the eigenvalues of an inverse matrix are the reciprocals of the eigenvalues of the original matrix. There is nothing inconsistent in this observation, as if λ_n is the largest eigenvalue of the matrix, then $1/\lambda_n$ will be the smallest eigenvalue of the inverse matrix.

Reasoning in a similar way, the inverse power method uses the power method to compute the largest eigenvalue of the inverse matrix, which corresponds to the smallest eigenvalue of the non-inverted matrix.

For the actual computation, at each iteration of the algorithm (after the first initialization step with a random vector), the eigenvector is calculated by solving the linear system:

$$(A - \mu I)v^* = v,$$

where v is the vector from the previous iteration, I is the identity matrix, and μ is a shift factor necessary to ensure the matrix's invertibility. Once v^* is found, the Rayleigh Quotient is applied to determine the eigenvalue of interest.

The concept of the inverse power method is the principle upon which Shifting is based.

6.4 Shifting Method

In this study, the shifting method has been used for the computation of eigenvalues and eigenvectors of the matrix. The method starts from a calculated eigenpair (λ_1, x_1) and, iteratively, shifts the matrix by removing the contribution of the given eigenpair. This makes λ_2 the largest eigenvalue, which is then computed using the inverse power method (i.e., applying the power method to the inverse matrix).

Indeed, if a diagonal matrix D is multiplied from the left by a matrix X, the diagonal elements become the columns of matrix X ($AX = XD$). Therefore, we can assert that:

$$A = \sum_{j=1}^{n} \lambda_j x_j x_j^T = \lambda_1 x_1 x_1^T + \sum_{j=2}^{n} \lambda_j x_j x_j^T \rightarrow A_1 = A - \lambda_1 x_1 x_1^T = \sum_{j=2}^{n} \lambda_j x_j x_j^T.$$

In this way, we find a similar matrix A_1 where the contribution of λ_1 has been removed. As is well-known, similar matrices share the same eigenvalues.

Moreover, from the equation above, if both sides are multiplied by x_1, the result is zero. This demonstrates that shifting moves the eigenvector corresponding to the largest eigenvalue (i.e., the smallest of the non-inverted matrix) into the kernel of the matrix. Iteratively, this process determines the desired number of the smallest eigenvalues of the matrix (denoted as M in this study).

To shift the eigenvalues (and eigenvectors) in the spectrum of the matrix, a shift factor γ set to 100 with a greedy heuristic was used. Additionally, a maximum iteration limit of `maxIter` $= 1000$ was set, beyond which the method does not converge. Starting with a value of $\mu = 0$, if the method fails to converge to the solution of the linear system for the eigenvector computation, a shift factor of $\mu = 10^{-8}$ is applied.

6.5 Deflation Method

In this study, the Deflation Method was implemented for calculating eigenvalues and eigenvectors, following the literature on the subject. Starting from an initial eigenpair, an orthogonal matrix to the original matrix A was found using the *Householder Method* such that:

$$P_1 A P_1^T$$

is in Schur Canonical Form, meaning it is a matrix where the first element of the first column is λ_1 (with the remaining elements of the column being zero), and the rest of the matrix consists of a submatrix of size $(n-1, n-1)$ and a coefficient along the first row of size $(1, n-1)$. Essentially, P_1 performs a change of basis on A to produce a matrix B_1 similar to A, which shares the same eigenvalues but not eigenvectors.

The basic process for calculating eigenvalues is as follows: iteratively, the inverse power method is used to compute the largest eigenvalue of the inverse of matrix A_i, which is the submatrix of the Schur canonical form at the i-th iteration.

For the computation of eigenvectors, however, the method is more complex: each time the inverse power method is applied to matrix A_i, an eigenvector of size $n-i$ is obtained, which must then be completed computing $n-i$ coefficients.

To do this, it is necessary to solve a linear system given by:

$$B_{i-1} x_i^*$$

where x_i^* is the eigenvector calculated at the i-th iteration, completed by $n-1$ coefficients, which represent the unknowns of the problem.

In solving this system, certain critical issues of the method arise: each coefficient is calculated backward (i.e., starting from the coefficient $(n-i+1)$) as the product of the coefficient vector of the matrix B_{n-i} at the previous iteration with the eigenvector of the current iteration:

$$-b_{n-i}^T x_i^T$$

dividing everything by the difference between the eigenvalue found at the current iteration and the one found at the previous iteration. Thus, using backward substitution, the other coefficients are calculated.

The problem with this method lies in the subtraction of eigenvalues in the denominator: if the two eigenvalues are equal or very similar (which may lead to numerical cancellation), the method cannot calculate the coefficient and, consequently, the eigenvector.

In this study, two solutions were considered to address the above problem (where possible):

1. The first involves implementing a try-except control based on solving the linear system for calculating eigenvalues and eigenvectors using the inverse power method on the submatrix of size $(n-i, n-i)$. The try attempts to solve the linear system, but if it fails (e.g., if the matrix is singular), the except shifts it by a coefficient $\mu = 1e-8$ to slightly perturb its eigenvalues and make the matrix invertible.

In this way, the method performs well in clustering since it indirectly overcomes the problem of null eigenvalues.

2. The second solution addresses cases where, regardless of the eigenvalue magnitude, the eigenvector calculation fails because the two eigenvalues subtracted have the same absolute value or are so close (and small) that they cancel out due to numerical cancellation. In this case, deflation cannot be performed for the calculation of the specific eigenvector, so at each iteration where this occurs, a control calculates the eigenvector with shifting and skips the current iteration, moving to the next one (of the deflation).

Unfortunately, while this method works, it does not yield optimal results, but certain precautions could improve its performance.

7 Eigenvector Projection and Clustering

Computed the eigenvectors of the Laplacian matrix associated with the k smallest eigenvalues and arranged along the columns of a matrix, the rows of the same matrix contain the coordinates of the dataset points in a new vector space where the cluster separation appears clearer and more evident. In figure (placeholder), the results obtained for each of the three datasets can be observed. The dimension of the projection space depends on the number of connected components identified in the graph through the study of the eigenvalues of the Laplacian matrix. Thus, it is possible to note that most configurations, having identified a number of connected components equal to or greater than 3, present a three-dimensional scatter plot, whereas others are two-dimensional.

The new projection of points is much more suitable for clustering using simpler algorithms such as k-means. The latter partitions the data in the new reference system by creating k partitions and assigning each dataset point to the cluster whose centroid or midpoint is the closest. In brief, its functioning is as follows: through an iterative procedure, the algorithm initializes k centroids randomly or using some heuristic information and assigns each point to its clus-

ter according to the criterion mentioned earlier. At each iteration, each centroid is updated by computing the mean of the points within the cluster. With the newly obtained configuration, the clusters are reassigned to the dataset points, and the procedure is repeated until the method surpasses a certain convergence threshold.

Applying k-means to the points identified in the new reference system allows obtaining an optimal cluster division, which, when mapped back onto the original dataset, enables a meaningful partitioning of the objects in the collection.

8 Others Clustering Methods

To compare the performance of the spectral clustering performed in this work, other methods provided by specific libraries have been used. Below are the analyses and comparisons.

8.1 K-Means

The k-means algorithm, in its most generic form, is an algorithm that tends to create non-overlapping clusters based on the movement of centroids. The algorithm assigns space points to the nearest centroid, which is initially placed randomly in space and is then recalculated for each cluster it generates based on different possible metrics. The algorithm terminates when the cluster no longer moves significantly from its position.

Thus, one of the most important evaluation metrics for k-means is the Silhouette Score, which calculates how well the method has clustered the points based on the number of clusters provided. The Silhouette Score is computed for each point in a cluster, based on the difference between the average distance of point i from all other points in the same cluster and the average distance of point i to all points belonging to the nearest cluster to which i does not belong. To obtain a single global result, the individual point results are aggregated, and an average of the scores is computed.
In this work, k-means was entirely performed by passing the original dataset and the same number of clusters used to find the number of connected components during spectral clustering (Greedy approach) to the class. In this context, the Silhouette Score is not a reliable metric, as it is based on different matrix management techniques and subject to calculations limited by machine precision. Therefore, the observations are purely visual.

In Fig.4 and Fig.5, the plots of spectral clustering and K-Means with 3 clusters are shown. It is possible to observe that K-Means does not perform well, as the circular shape of the data challenges the method: the aforementioned algorithm is highly effective only in cases where the data has a circular distribution.

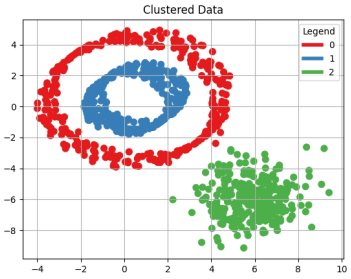

Figure 4: S.C.shifting-shifting k=10

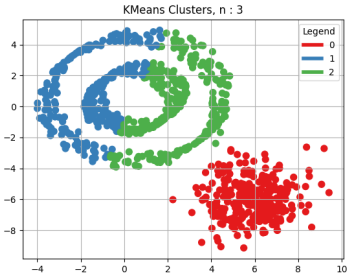

Figure 5: KMeans k=10

8.2 DBSCAN

DBSCAN performs clustering following a different logic: a cluster is a dense region in space. Density means a number of points within a given radius (eps). So, is talking about core points (i.e., points that have a number of neighboring points greater than a given threshold), border points (i.e., points that have fewer neighbors than the threshold but belong to the neighborhood of a core point), and noise points (i.e., points that have fewer neighbors than the threshold and do not belong to the neighborhood of a core point).

The DBSCAN algorithm starts with a current cluster label and assigns this label to all core points if they are not already assigned to any cluster. Then, for each core point, using an internal loop, it assigns the current cluster label to every point within its specified eps.

In this work, the following parameters were assigned (Greedy heuristic):

- For "Circle": eps = 0.75, minsamples = 10;
- For "Spiral": eps = 2, minsamples = 5.

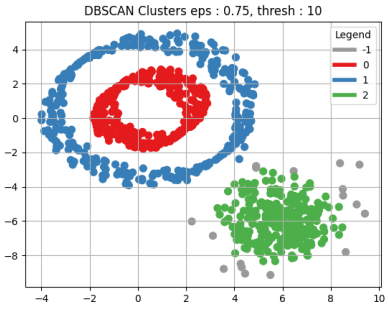

Figure 6: DBSCAN eps=0.75,minsamples=10

It's easy to see that when setting M=3, the three clusters are correctly formed, and the noise points are highlighted in gray. This result was expected since DBSCAN is very robust to noise and adapts well to all shapes.

8.3 Spectral Clustering

Here, we compare the spectral clustering implemented in this work with the built-in sklearn class, using the same neighborhood search parameters (in the similarity matrix) and the same number of clusters

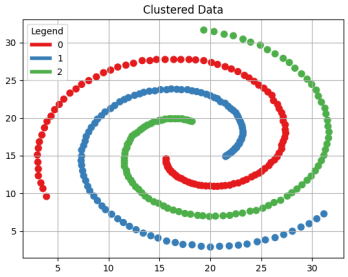

Figure 7: S.C.shifting-shifting k=10

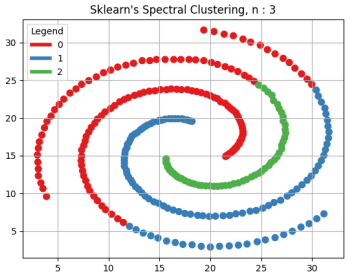

Figure 8: SKlearn S.C. k=10

In Fig.7 and Fig.8, it can be observed that the sklearn method does not perform clustering correctly. The reasons for this can be multiple, but they are not the focus of this study.

9 Results

Circles.csv

In the first case, considering the first 10 neighbors for each point in the dataset, 3 connected components were found when analyzing the eigenvalues of the Laplacian matrix. This result is consistent not only with the mathematical findings but also with what can be visually interpreted from the dataset's point distribution. When assigning each original point a color corresponding to the cluster found by k-means, the resulting grouping is clear. The two concentric circles were correctly divided into two distinct clusters, while the cloud of points on the right, which is evidently further away from the other points in the dataset, was entirely grouped into a third cluster. Points on the boundary between the cloud and the circles, on the other hand, were split evenly between the cluster of the outer circle and the cloud, which looks visually correct. Using such a small number of nearest neighbors allows the algorithm to take advantage of the dataset's density and prevent points that are too far apart from initially being placed in the same connected component and later in the same cluster. This is

evident in the previously mentioned boundary points, which were assigned to the clusters that are visually closest.

In the second case, considering the first 20 neighbors, the number of connected components identified was again 3, although the third eigenvalue was higher than in the previous case. The reasoning behind this is clear when considering that with this new configuration, points that are generally farther apart are more likely to be included in the same neighborhood. However, once again, thanks to the dataset's high density, the resulting partition is almost identical to the previous one, showing good performance from the algorithm even when changing the input attributes.

In the third case, with the first 40 neighbors, 2 connected components were identified based on the eigenvalue analysis. This result provides an interpretation that deviates more from the visual perception. Despite this, the outcome is still satisfactory, with the two concentric circles being assigned to the same cluster. This was expected, given that many points, especially those on the outer and inner edges of the smaller and larger circles, are quite close to each other. Choosing such a high number of nearest neighbors led to points that seem to belong to distinct groups being grouped into the same cluster. While the previous results might seem more intuitive, this still proves to be a valid interpretation of the dataset's internal structure.

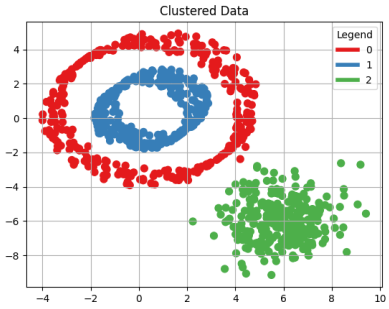

Figure 9: Spectral Clustering with 3 clusters and k=10

Spiral.csv

In the first test, where 10 nearest neighbors were selected and 3 connected components were chosen, the result obtained aligns with what could have been expected given the nature of the dataset. The spiral-like arrangement of the points already indicated a clear separation between the groups of points forming the spirals. The resulting partition groups the points on the same spiral curve within a single cluster. This assumes that the first 10 neighbors of each point are all within the same spiral, which is not obvious, especially when observing the tails of the spirals where the points are less densely packed.

In the second test, with 20 nearest neighbors, 3 connected components were again selected, assuming that the first 3 eigenvalues would be small enough to identify them. However, it can be seen from the plots that both the second and third eigenvalues have slightly larger magnitudes compared to the previous case. This is reasonable, considering that twice as many neighbors were selected compared to the previous test, and thus it is much more likely that pairs of points from different spirals will fall into the same cluster. Despite these components being less distinct from one another, the result obtained is very similar to the previous one, highlighting that the 20 nearest neighbors of each point are, on average, more likely to be found within the same spiral than outside it.

Unexpectedly, the final test with 40 nearest neighbors also produced a result similar to the two previous ones. Part of this outcome is likely due to the symmetrization of the adjacency matrix of the graph, which makes points connected in sequence more likely to be placed in the same cluster. While the central region of the plot showed this result to be partially predictable, it is surprising to see it also appear in the tails of the spirals.

Additionally, this dataset includes a column of values containing the correct labels, allowing for a comparison between the obtained results and those achievable using the correct labels. Results are showed in 10 and 11.

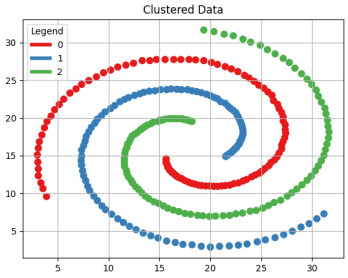

Figure 10: S.C.shifting-shifting k=10

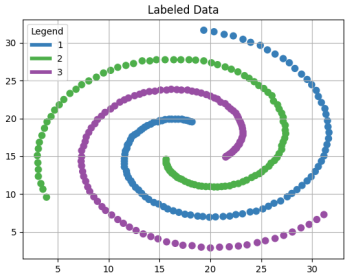

Figure 11: S.C with labels

3D Dataset.csv

The model proposed in this paper selects the number of clusters for the dataset based on a visual assessment. As mentioned earlier, eigenvalues are considered negligible if they appear before the "elbow" in the graphical representation. This approach usually leads to good results. However, in this case, the graphical method isn't very effective.

Following the usual logic, the chosen number of clusters would be 4 since two of the five actual clusters are very close to each other. Even though the model performs well when forced to cluster into five groups, for consistency

with the selection criterion, we decided to stick with four clusters. Below, it has been compared the obtained result with the expected result (using the provided labels)

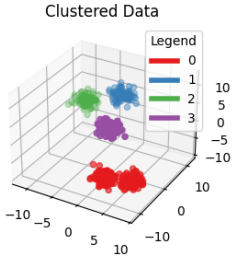

Figure 12: S.C. deflation-deflation k=10

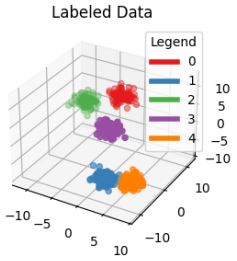

Figure 13: S.C. with labels