Spectral Clustering

Marco Galano, s338658

Abstract
In this report we discuss the spectral clustering analysis performed on two datasets. The points in the two datasets are not well separated in convex clusters, so a traditional clustering method that works well with clusters with a convex shape, such as a circumference (e.g. *K-means*), is not the ideal approach for this problem. However, the points can be represented as a KNN graph, with the weight of the edges representing the similarities between them, so we will exploit these similarities to cluster them.

Contents

1 Introduction

The two datasets, both 2-dimensional, are the *Circle Dataset*, and the *Spiral Dataset*, respectively shown in the figures 1 and 2. The *Circle Dataset* contains 900 points, while the *Spiral Dataset* contains 312 points and one column with the correct label for each point. As can be seen, while in the circle dataset we could try to perform a *K-means* clustering, that would distinguish between the two concentric circumferences and the noise, it would not perform a good job with the spiral one. The proposed approach consists of the following:

1. Computing the similarities of the points in each dataset;
2. For each dataset, building a k-nearest neighbors graph for values of k=10,20,40;
3. For each graph, building the adjacency matrix **W**, the degree matrix **D** and the Laplacian matrix **L**;
4. Computing the number of connected components, and based on these and on the shape of the points in the dataset, computing the m smallest eigenvalues and the corresponding eigenvectors;
5. Create the matrix **U**, whose columns are the just computed eigenvectors, and that will be used to perform a *K-means* clustering, with k=m.

U represents a change of basis of the points of the dataset from the canonical basis to the one generated by the eigenvectors of **L**. **U** is still a representation of the original points, and it can be used to perform traditional clustering algorithms such as *K-Means*

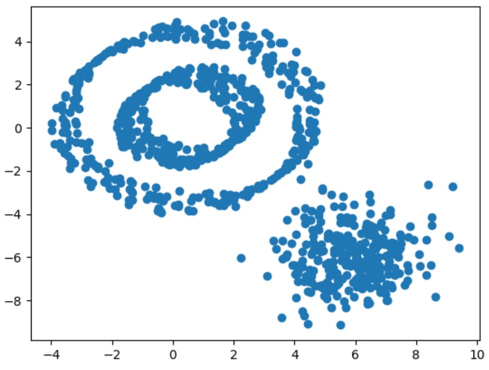

Figure 1: Circle dataset.

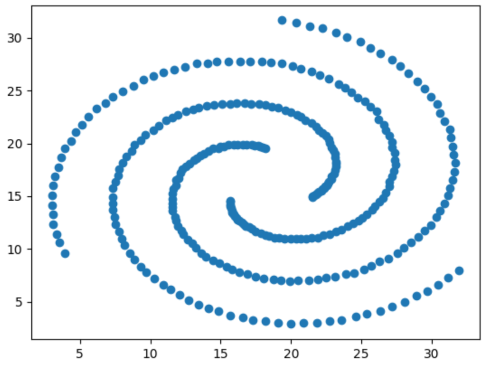

Figure 2: Spiral dataset.

2 Data Preparation

2.1 Dataframe cleansing operations

Both dataframes did not require any kind of cleansing operation, as the data provided were consistent and there were no missing data from any of the rows.

2.2 Data structures

The first operation performed with both dataframes was creating the similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, where n is the number of points in the dataset. The computation of this matrix is based on the *similarity function*, that is, given two points X_i and X_j;

$$s_{i,j} = \begin{cases} exp(-\frac{|X_i - X_j|^2}{2\sigma^2}) & \text{if } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

As it can be seen by the definition of the elements of the matrix, $s_{i,j} = s_{j,i}$, so the matrix just computed will be symmetric by construction. Moreover, it will have all zeros on the diagonal.
After this, we create a graph, whose vertices represent a point, and it is connected to its k

more similar points by weighted edges, representing their similarities. From this graph, we can build the *weighted adjacency matrix* **W**, defined as:

$$w_{i,j} = \begin{cases} s_{i,j} & \text{if } X_j \in KNN\{X_i\}, \\ 0 & \text{otherwise.} \end{cases}$$

With KNN$\{X_i\}$ which is the set of the k nearest neighbor of X_i.
Notice that, even though **S** is symmetric, if X_j is between the k more similar point to X_i, this does not mean that X_i will be between the most relevant point for X_j, this resulting in **W** not being symmetric. This would be an issue with the following usage of **D** and **L** for the spectral clustering. To solve this, we impose that if X_j is relevant to X_i, automatically X_i becomes relevant for X_j. After having done this, the matrix **W** will become symmetric, but the corresponding graph will have more than k nearest neighbors for some of its vertices.
The matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix, whose elements are defined as:

$$d_{i,i} = \sum_{j=1}^{n} W_{i,j}.$$

Finally, the matrix **L** is computed as:

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

As **W** and **D** are symmetric, also **L** is symmetric. Moreover, **L** is a positive semi-definite matrix (SPsD), this is an important property, as it tells us that the smallest eigenvalue of **L** is $\lambda = 0$, and all its eigenvalues are greater than or equal to 0. This explains why we built the graph in such a way that **W** would be symmetric.

3 Eigenvalues, eigenvectors and choice of number of clusters k

3.1 Connected components

First of all we count the number of connected components of the graph, knowing that this number is also the algebraic multiciplity of $\lambda = 0$ as an eigenvalue for the matrix **L**. To do so we apply a *Depth First Search* algorithm to the adjacency graph. Each connected component represents a well separated cluster from the others, while a small magnitude eigenvalue represents clusters with a very slight connection. By just computing the number of connected components, we could get a rough idea of how many clusters our model is going to identify. However, a better analysis can be done on the eigenvalues of **L**. For this purpose, we will compute and plot the smallest eigenvalues of the matrix, and decide how many clusters use for the *K-means* on the matrix **U**.

3.2 Eigenvalues computation

To compute the m smallest eigenvalues of **L** we use a combination of two numerical methods, which are the *Inverse Power Method*, used to compute an eigenpair λ_i and $\mathbf{x}_i$ from an initial guess p for the eigenvalue and **x** for the eigenvector, and the *Deflation Method*, to iteratively compute the smallest λ_i of **L**. These two methods can be properly used on **L** because of its property of being a SPsD matrix. In fact, the *Inverse Power Method*, with p = 0 as a guess for the computation of the eigenvalue, will always approximate the smallest eigenvalue of the matrix (i.e. the closest to 0) and its relative eigenvector. The *Deflation Method* is used to "*deflate*" the matrix, that means, given a matrix **A** with eigenvalues $\lambda_1 \leq ... \leq \lambda_n$, we can compute the eigenpair λ_1, $\mathbf{x}_1$, and from those we can obtain a matrix **B**, that has λ_1 as element in the position $\mathbf{B}_{1,1}$, and the submatrix $\mathbf{B}_{2:,2:}$ that will have the remaining eigenvalues from λ_2 to λ_n. By iteratively applying m times the *Deflation Method* to those submatrices $\mathbf{B}_{2:,2:}$, and using the *Inverse Power Method* to compute the smallest eigenvalue, we obtain the list of the m smallest eigenvalues of the matrix $\mathbf{L}$.

3.3 Eigenvalues analysis and eigenvectors computation

After having computed the m smallest eigenvalues, we can plot these, to identify a correct number of clusters that can be computed. In fact, in addition to computing a number of clusters equal to the algebraic multiciplity of 0, we can add a cluster for each eigenvalue that has a very small magnitude compared to the following ones. This can be done because the clusters represented from the smallest magnitudes eigenvalues have a very weak connection to others, so they can be considered separated. After having chosen a correct number of clusters k, we compute the eigenvectors corresponding to the k smallest eigenvalues, that will be the columns of the matrix $\mathbf{U}$. This matrix represents a change of basis of the original points in the dataset, and it will be used to perform the *K-means* clustering, and each of the k clusters computed will contain the indices of the points that belong to that cluster in the original dataset.

4 Results

Here we present the results obtained with the discussed model, and a comparison with what can be obtained using *K-means* clustering on the original datasets.

4.1 Circle Dataset

4.1.1 K=10 nearest neighbors

With a 10 nearest neighbors graph we obtain 2 connected components. As it can be seen in figure 3 that shows the 10 smallest eigenvalues of matrix $\mathbf{L}$, the third eigenvalue has a very small magnitude compared to those of the eigenvalues from 4 to 10, so the best choices for k would be k = 2 or k = 3. We can see that while with 2 clusters we do not get a meaningful result, with 3 clusters we perfectly identify the two concentric circumferences and the noise.

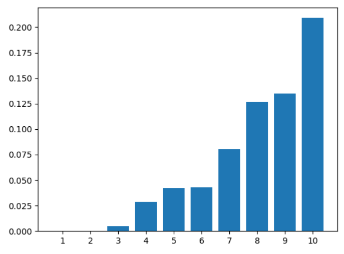

Figure 3: First 10 eigenvalues

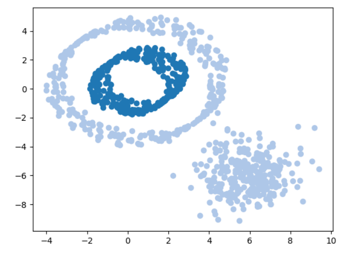

Figure 4: Spectral clustering with 2 clusters

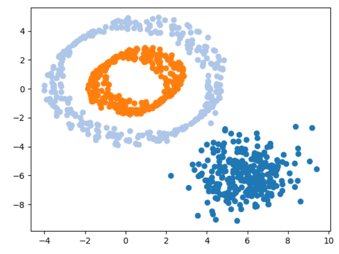

Figure 5: Spectral clustering with 3 clusters

4.1.2 K=20 nearest neighbors

With a 20 nearest neighbors graph we obtain 1 connected component. As it can be seen in figure 6 that shows the 10 smallest eigenvalues of matrix $\mathbf{L}$, the third eigenvalue still has a very small magnitude compared to those of the eigenvalues from 4 to 10, so, also for this graph, the best choices for k would be k = 2 or k = 3. Here with 2 clusters we get a much more meaningful result, that separates the two concentric circles from the noise, but it groups them in the same cluster. With 3 clusters we perfectly identify the two concentric circumferences and the noise.

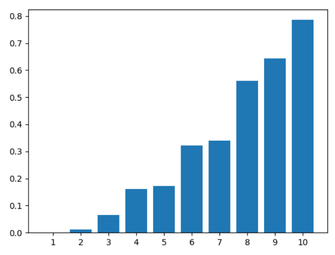

Figure 6: First 10 eigenvalues

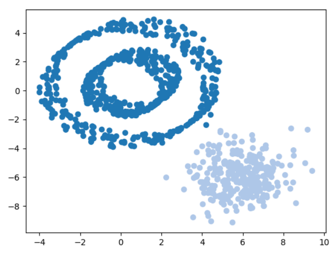

Figure 7: Spectral clustering with 2 clusters

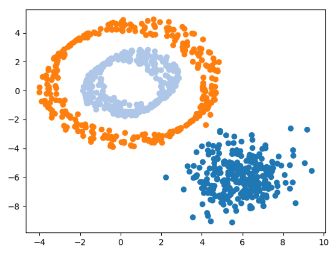

Figure 8: Spectral clustering with 3 clusters

4.1.3 K=40 nearest neighbors

With a 40 nearest neighbors graph we obtain 1 connected component. However, differently from the last case, the third eigenvalue has a very comparable magnitude to the following, so, the only correct choice for k is 2. With 2 clusters we get the same exact result of the previous case, while, looking at the eigenvalues, it would not be correct to identify 3 clusters, meaning that creating a 40 nearest neighbors graph is an overfit for this problem.

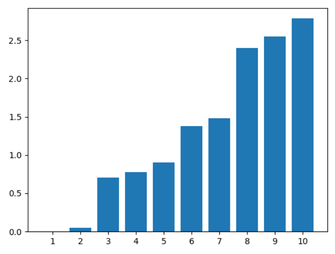

Figure 9: First 10 eigenvalues

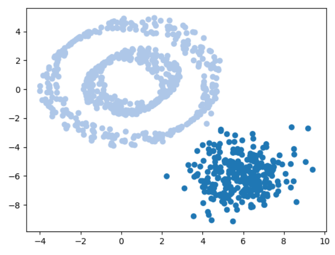

Figure 10: Spectral clustering with 2 clusters

Based on the previous considerations, the best fit for this problem would be creating a 20 nearest neighbors graph, because it could both identify 2 or 3 meaningful clusters correctly, while the other two graphs fail in one of the clustering.

4.1.4 Comparison with *K-means* clustering

The figure 11 shows a *K-means* clustering with 2 clusters. It gives us the same results obtained, with the 20 nearest neighbors and the 40 nearest neighbors graphs with 2-clustering. It is an expected behavior, as the two circumferences have a convex and almost circular shape, while the points in the noise part resemble a single cluster. The *K-means* clustering with 3 clusters, shown in figure 12, has a completely meaningless but predictable behavior, as the 3 main shapes in the datasets are not representable in terms of euclidean distance from 3 separated centroids, this resulting in a correct individuation of the noise, and in a completely misunderstanding of the circular part.

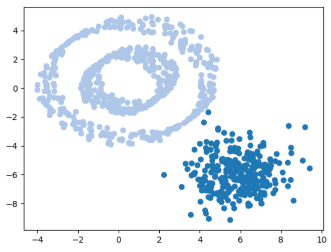
Figure 11: K-means clustering with 2 clusters

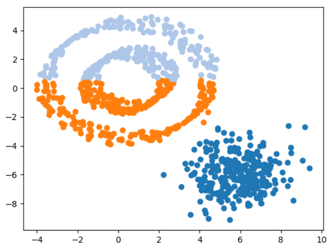
Figure 12: K-means clustering with 3 clusters

4.2 Spiral Dataset

4.2.1 K=10 nearest neighbors

With a 10 nearest neighbors graph we obtain 1 connected component. The eigenvalues shown in the figure 13 make clear that the correct choice of k is k = 3, as the following 3 eigenvalues have approximately the same magnitude. With 3 clusters our model correctly identifies each spiral in the dataset.

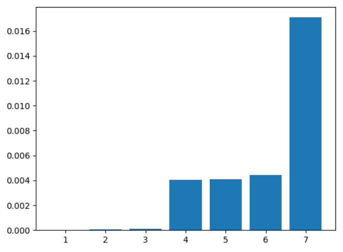
Figure 13: First 7 eigenvalues

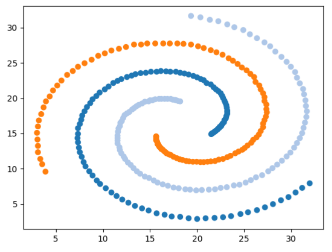
Figure 14: Spectral clustering with 3 clusters

4.2.2 K=20 nearest neighbors

With a 20 nearest neighbors graph we obtain 1 connected component. Also in this case, based on the eigenvalues shown in the figure 15, a good choice of k is k = 3. This model is still identifying correctly the 3 spirals.

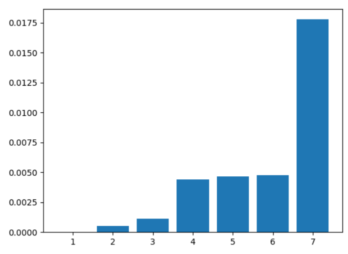

Figure 15: First 7 eigenvalues

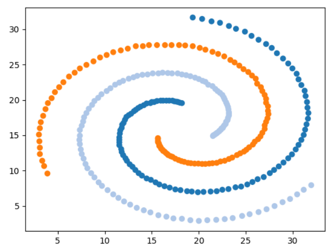

Figure 16: Spectral clustering with 3 clusters

4.2.3 K=40 nearest neighbors

With a 40 nearest neighbors graph we obtain 1 connected component. The eigenvalues in the figure 15, shows that even though the third eigenvalue is approaching the fourth one, it is still too small compared to it, so a good choice of k is still k = 3. With 3 clusters the result obtained is the same of the two previous graphs.

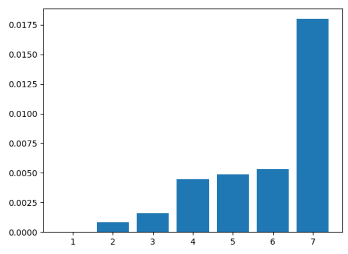

Figure 17: First 7 eigenvalues

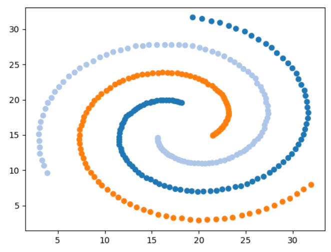

Figure 18: Spectral clustering with 3 clusters

All three of the graphs have one connected component, and their corresponding matrices **L**'s three smallest eigenvalues are way smaller than the ones from the fourth to the seventh, making for all three of them easy to identify 3 as the correct number of clusters to be computed. The spiral dataset provides also a third column for the labels that should be assigned to each point. With those we computed the correct clustering, shown in figure 19 shows that the three previous results were correct. As the result for all the models is the same of the one obtained with the 10 nearest neighbors graph, we could say that choosing to have only 10 neighbors for each point is a correct fit for this problem, and it is also shown in figure 13, as its second and third eigenvalues are very close to zero, meaning that the three spirals are very well disconnected.

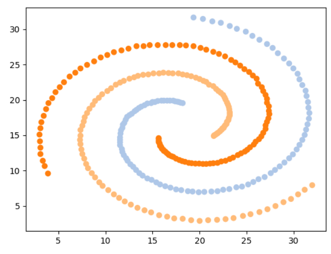
Figure 19: Correct clustering of the spiral dataset based on the provided labels

4.2.4 Comparison with *K-means* clustering

Figure 20 shows a *K-means* clustering with 3 clusters. As expected, the result obtained is awful, as there are no meaningful clusters represented by points near to a centroid.

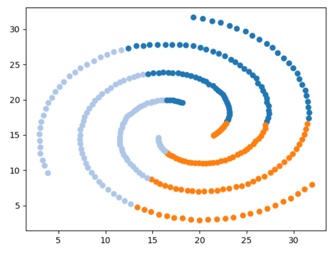
Figure 20: K-means clustering with 3 clusters

5 Analysis on a 3D dataset

To further test our model, we built a 3D dataset consisting of random line-shaped figures, shown in figure 21, and we applied the spectral clustering to it. For the scope of reproducibility, we used *numpy.random.seed(338658)* for the random functions. The number of lines, the number of points per line, and the noise (standard deviation of a point from the main line) can be customized in the source code. In our test, we used 10 lines, with 100 points per line, and a noise of 0.1. In this way, the whole dataset consists of 1000 points.

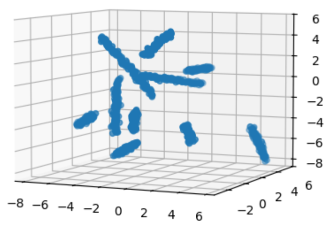
Figure 21: Random line-shaped dataset

5.1 Connected components

First of all we built the adjacency graph, the adjacency matrix **W**, the degree matrix **D** and the Laplacian matrix **L**, as it was described for the previous dataset. After some testing, we decided to use a 10 nearest neighbors graph for this dataset.
The number of connected components with these features is 10, meaning that all the lines are well separated.

5.2 Eigenvalues computation and analysis

For this dataset the number of connected components was higher than both the previous two datasets, so we decided to compute the smallest m = 20 eigenvalues, shown in the plot in figure 22. We can see that, obviously the first 10 eigenvalues are 0, while the eleventh one is not relatively small compared to the twelfth one, so the correct choice for k in the K-means clustering of the matrix **U** will be k = 10.

5.3 Comparison between Spectral clustering and K-means clustering

After having computed **U** with the eigenvectors corresponding to the 10 smallest eigenvalues, we used it to perform the spectral clustering. We also compared it to the K-means clustering on the same dataset. The figure 23 shows that our model perfectly recognize each line, which is an expected behavior, as in the previous step we checked that we had 10 connected components, so 10 perfectly separated clusters. Differently, the K-means clustering is having troubles with lines near to others. In fact, as it can be seen in figure 24 , the points of a line close to the points of another line are mis-clustered, and are given the same label. This is also an expected behavior, as our figures are not built around a single centroid, so the K-means struggles recognizing them.

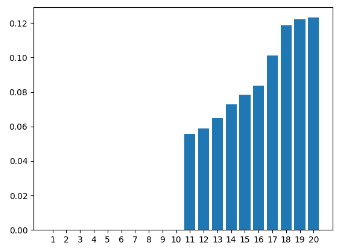

Figure 22: First 20 eigenvalues

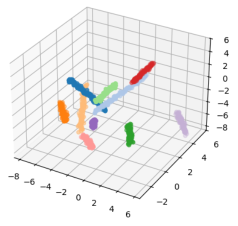
Figure 23: Spectral clustering with 10 clusters

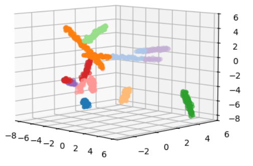
Figure 24: K-means clustering with 10 clusters

6 Normalized Laplacian matrix

To make an even more robust analysis, we could try to normalize the Laplacian matrix. The normalized Laplacian matrix is defined as:

$$\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$$

In all of our cases we already obtained a robust analysis, and the normalization of the Laplacian matrix did not get us any improvement. However, we will show the comparison between the eigenvalues computation to show some differences.

6.1 Circle Dataset

6.1.1 K=10 nearest neighbors

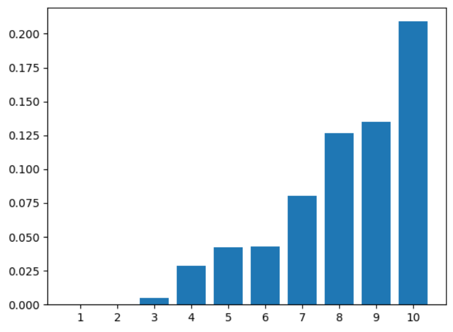

Figure 25: First 10 eigenvalues of $\mathbf{L}$

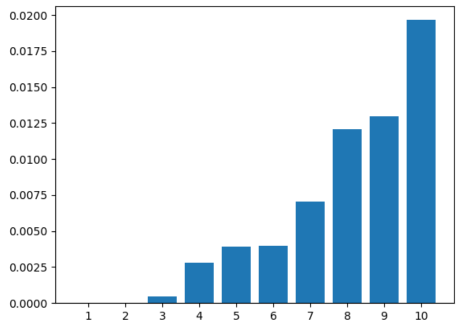

Figure 26: First 10 eigenvalues of $\mathbf{L}_{sym}$

6.1.2 K=20 nearest neighbors

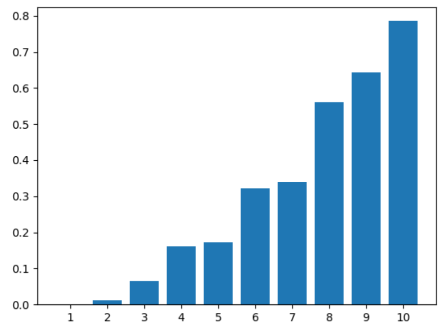

Figure 27: First 10 eigenvalues of $\mathbf{L}$

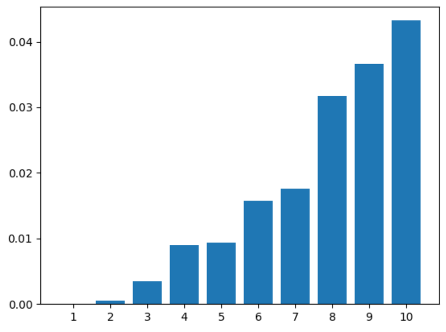

Figure 28: First 10 eigenvalues of $\mathbf{L}_{sym}$

6.1.3 K=40 nearest neighbors

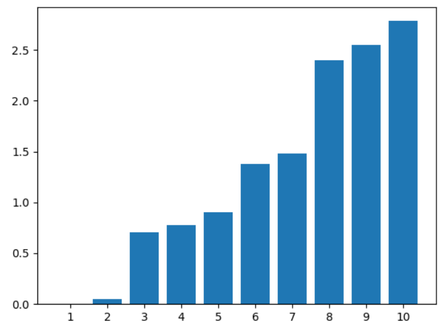

Figure 29: First 10 eigenvalues of $\mathbf{L}$

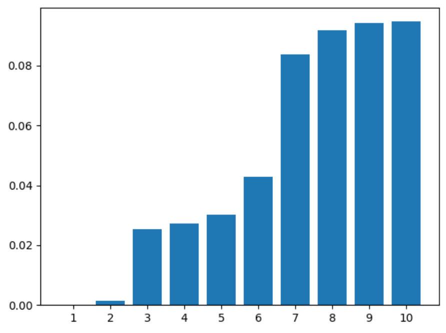

Figure 30: First 10 eigenvalues of $\mathbf{L}_{sym}$

6.2 Spiral Dataset

6.2.1 K=10 nearest neighbors

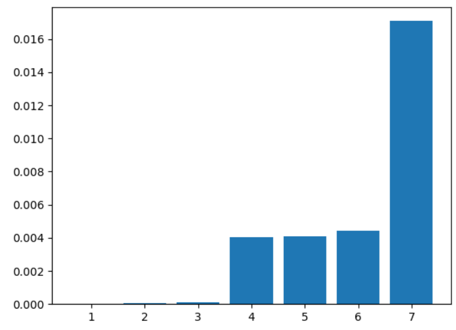

Figure 31: First 7 eigenvalues of $\mathbf{L}$

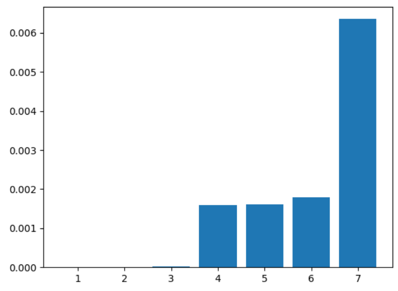

Figure 32: First 7 eigenvalues of $\mathbf{L}_{sym}$

6.2.2 K=20 nearest neighbors

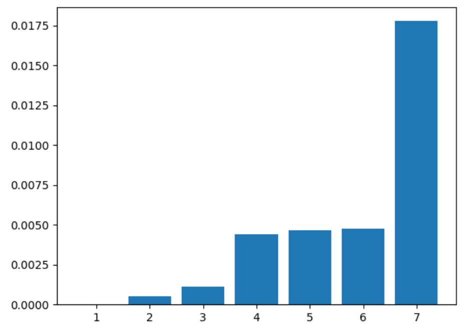

Figure 33: First 7 eigenvalues of $\mathbf{L}$

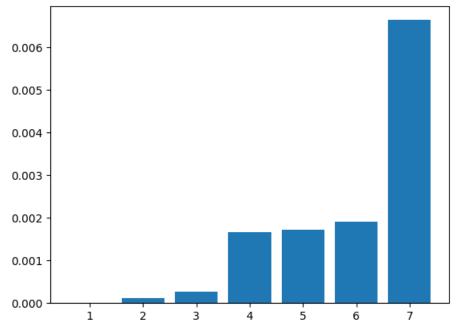

Figure 34: First 7 eigenvalues of $\mathbf{L}_{sym}$

6.2.3 K=40 nearest neighbors

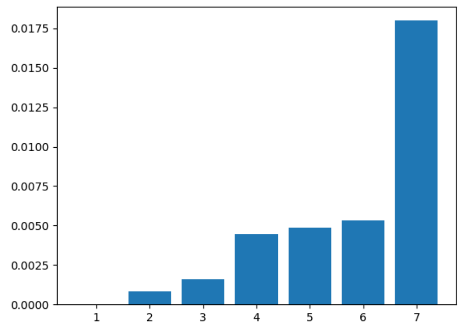

Figure 35: First 7 eigenvalues of $\mathbf{L}$

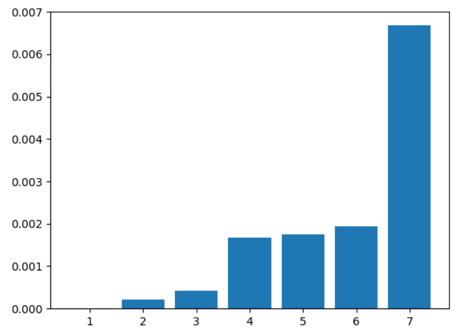

Figure 36: First 7 eigenvalues of $\mathbf{L}_{sym}$

6.3 3D Dataset

6.3.1 K=10 nearest neighbors

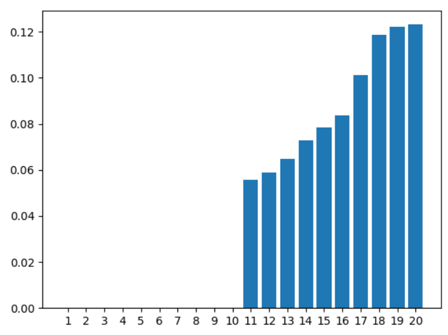

Figure 37: First 20 eigenvalues of $\mathbf{L}$

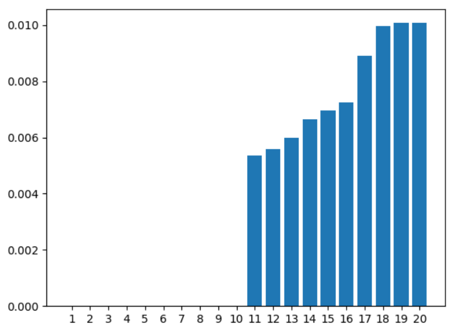

Figure 38: First 20 eigenvalues of $\mathbf{L}_{sym}$

While we can see an obvious shift of magnitudes of the eigenvalues, that are way smaller in the normalized Laplacian matrix compared to the normal one, the comparison between subsequent eigenvalues magnitudes is almost the same for the two matrices, so this normalization does not induce any further analysis.