Graph Clustering

Group 6
Diptanil Santra
Rajdeep Brahma
Arkaprava Sanki

October 8, 2022

Abstract

Clustering is one of the most widely used techniques for exploratory data analysis. It has applications in many fields including statistics, computer science, biology to social sciences etc. Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

1 Data Collected

We used the data given here.This is a data of sparse biological network.We will attach a rough summary of the data we used below.

Network Data Statistics	
Nodes	2.2K
Edges	53.7K
Density	0.021795
Maximum degree	242
Minimum degree	1
Average degree	48
Assortativity	0.0683706
Number of triangles	685.6K
Average number of triangles	308
Maximum number of triangles	3.4K
Average clustering coefficient	0.183877
Fraction of closed triangles	0.139016
Maximum k-core	49
Lower bound of Maximum Clique	16

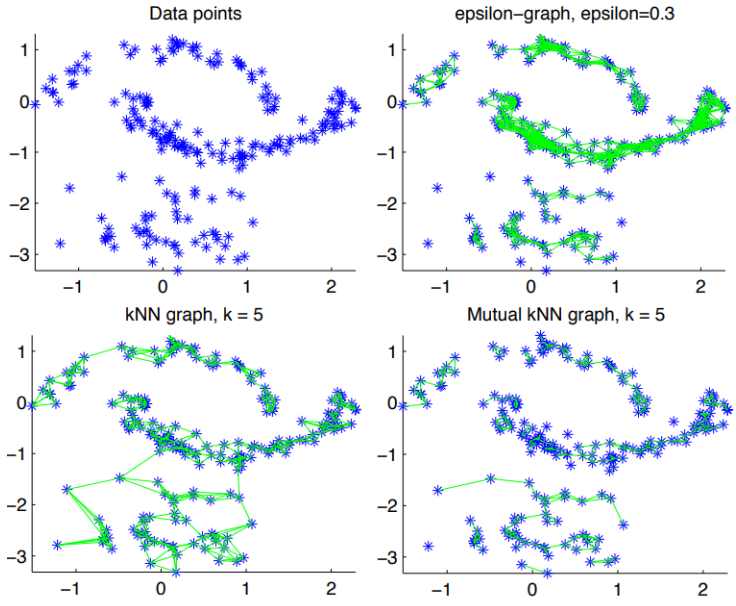
All the edges in the graph are weighted and undirected.Different graphs are given in the above mentioned site which can give a visual representation of the data.

2 Preliminaries

2.1 Similarity Graphs:

Given a set of data points $x_1, ...x_n$ and some notion of similarity $s_{ij} \geq 0$ between all pairs of data points x_i and x_j , the intuitive goal of clustering is to divide the data points into several groups such that points in the same group are similar and points in different groups are dissimilar to each other. If we do not have more information than similarities between data points, a nice way of representing the data is in form of the similarity graph $G = (V, E)$. Each vertex v_i in this graph represents a data point x_i. Two vertices are connected if the similarity sij between the corresponding data points x_i and x_j is positive or larger than a certain threshold, and the edge is weighted by s_{ij} . The problem of clustering can now be reformulated using the similarity graph: we want to find a partition of the graph such that the edges between different groups have very low weights (which means that points in different clusters are dissimilar from each other) and the edges within a group have high weights (which means that points within the same cluster are similar to each other).Now we have different types of similarity graphs like KNN graph,mutual KNN graph,fully connected graph and ϵ neighbourhood graph.
In this project we will be working with KNN graph.
The main advantages of using KNN over other graphs are :

- The k-nearest neighbour graph, on the other hand, can connect points "on different scales" and can break into several disconnected components if there are high density regions which are reasonably far away from each other.
- It is simple to work with, results in a sparse adjacency matrix W, and is less vulnerable to unsuitable choices of parameters than the other graphs.



Here is a picture that can explain this problem in details. In the fist picture we have data points. For ϵ-neighborhood graph it is hard to find a good choice of ϵ which we can work with. Here the data points are of different scales, data points moon shaped regions are very tightly situated, but in the lower part that is not the case.
For KNN graph it is better, in the bottom left picture, it is shown that, in KNN graph data points from moon shaped region are connected with the data points from the lower part (low density region). And also it breaks two moon regions into two different groups.
While in case of mutual KNN, this connection between two different density region is not present. Mutual KNN works on different scaled regions, but does not mix those scales with each other.

2.2 Graph Laplacian

The unnormalized graph Laplacian matrix is defined as

$$L = D - W$$

Normalised graph Laplacian matrix can be defined in 2 ways but here we will be using

$$L_{rw} = D^{-1}L = I - D^{-1}W$$

where D is the diagonal matrix containing all degrees of vertices in the diagonals.($L_{sym} := I - D^{-1/2}WD^{-1/2}$)$L_{rw}$ is so called because it is closely related to a random walk where as L_{sym} is a symmetric matrix and hence the name.
The theorem on graph Laplaciann that we will be using our algorithm is :
Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L_{rw}equals the number of connected components $A_1, ..., A_k$ in the graph.
We will use Spectral clustering algorithm based on Normalised Laplacian because it has two very important properties

- It finds a partition such that points in different clusters are dissimilar to each other,that is between-cluster similarity is minimised.
- It finds a partition such that points in the same cluster are similar to each other, that is it maximizes the within-cluster similarities.

Unnormalised Spectral clustering satisfies the first criteria but fails in the second.And we prefer L_{rw} over L_{sym} for the former is simpler to calculate and less prone to errors.

3 Algorithm for finding Spectral Clustering

We can use

1. unnormalized spectral clustering(unnormalized graph Laplacians)
2. normalized spectral clustering(can use L_{rw} or L_{sym})

Here two spectral clustering algorithms are described below.

Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
Steps

- Construct a similarity graph(popular choices are ϵ-neighbourhood graph, k-nearest neighbor graphs, Gaussian similarity function) . Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- Compute the first k eigen vectors $u_1, \cdots, u_k$ of L.(Basically we are finding the eigen vectors corresponding to the least k eigen values of L.)
- Let U be the $n \times k$ matrix containing the vectors $u_1, \cdots, u_k$ as columns.
- For $i = 1, \cdots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\cdots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \cdots, C_k$.

Output: Clusters $A_1, ..., A_k$ with $A_i = \{j | y_j \in C_i\}$.

Normalized spectral clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
Steps

- Construct a similarity graph(popular choices are ϵ-neighbourhood graph, k-nearest neighbor graphs, Gaussian similarity function) . Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- Compute the first k generalized eigen vectors $u_1, \cdots, u_k$ of the generalized eigen problem $Lu = \lambda Du$.
- Let U be the $n \times k$ matrix containing the vectors $u_1, \cdots, u_k$ as columns.

- For $i = 1, \cdots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\cdots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \cdots, C_k$.

Output: Clusters $A_1, ..., A_k$ with $A_i = \{j | y_j \in C_i\}$.
Difference between this two above mentioned method is that, in the first one we are working with the eigen vectors of L matrix, whereas in the second one we use eigen vectors of L_{rw}.

4 Different interpretation of Spectral Clustering

The algorithm of Spectral Clustering helps us to solve many problems,some of which are explained in this section.

4.1 Graph Cut point of view

There are two types of cuts: (i) Ratio cut and (ii) N cut.

RatioCut of k partitions$(A_1, A_2, ...A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{|A_i|}$. Where $W(A_i, \bar{A}_i)$ is the sum of all edge weights from A_i to $\bar{A}_i$ which can be considered as a measure of between cluster similarities. Since we are minimizing between cluster similarities, the problem is same as minimizing Ratio cut.

$|A_i|$ are there in denominator of the expression of Ratio cut because in many cases, the solution of mincut simply separates one individual vertex from the rest of the graph. But we want A_i's are "balanced". So if we minimise Ratio cut, the chance of getting a single node as a cluster is low.

First we talk about solving Ratio-cut problem for k=2. After some algebraic manipulation this problem is equivalent to

$$min_{f \in R^n} f'Lf$$

under the constraints $f \perp \mathbf{1}$ and $||f|| = \sqrt{n}$. Where L is the graph Laplacian.
It can be shown that the solution of this problem is given by the vector f which is the eigen vector corresponding to the second smallest eigenvalue of L. The set A is constructed as follows

$$\begin{cases} v_i \in A, & \text{if } f_i \geq 0 \\ v_i \notin A, & \text{if } f_i < 0 \end{cases}$$

What most spectral clustering algorithms do instead is to consider the coordinates f_i as points in $\mathbb{R}$ and cluster them into two groups $C, \bar{C}$ by the k-means clustering algorithm. Then we carry over the resulting clustering to the underlying data points, that is we choose

$$\begin{cases} v_i \in A, & \text{if } f_i \in C \\ v_i \notin A, & \text{if } f_i \notin C \end{cases}$$

So we can see that this algorithm for this problem is exactly the unnormalized spectral clustering algorithm for the case of k $= 2$.
For $k > 2$, the problem becomes

$$\min_{H \in \mathbb{R}^{n \times k}} Tr(H'LH) \text{ Subject to } H'H = I$$

The solution is given by choosing H as the matrix which contains the first k eigenvectors of L as columns.(Here first k eigenvectors means eigenvectors corresponding to the smallest k eigenvalues.) We can see that the matrix H is in fact the matrix U used in the unnormalized spectral clustering algorithm for general k.

Next we move on to N-Cut problem.
The problem (for k=2) may be stated as

$$\min_{A \subset V} NCut(A, \bar{A})$$

where,

$$NCut(A_1, A_2, ...A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{vol(A_i)}$$

Vol(A_i) is the sum of all the weights in A_i.
After some relaxation and manipulation it can be shown that this problem is equivalent to

$$min_{g \in R^n} g' D^{-\frac{1}{2}} L D^{-\frac{1}{2}} g$$

under the constraints $g \perp D^{\frac{1}{2}}\mathbf{1}$ and $||g||^2 = vol(V)$. Where L is the graph Laplacian. Re-substituting $f = D^{-1/2}g$ we see that f is the second eigen vector of L_{rw}. We construct A exactly in the same way described above and we see that this process is same as normalized spectral clustering for k=2.
This can be easily generalised for $k > 2$.
For $k > 2$, the problem becomes

$$\min_{T \in \mathbb{R}^{n \times k}} Tr(T' D^{-\frac{1}{2}} L D^{-\frac{1}{2}} T) \text{ Subject to } T'T = I$$

The solution consists of the first k eigenvectors of the matrix L_{rw}. This yields the normalized spectral clustering algorithm.

So we see that we can solve the approximate NCut and Ratio-Cut problem using Spectral Clustering Algorithm.

4.2 Random Walks point of view

We will see that spectral clustering can be interpreted as trying to find a partition of the graph such that the random walk stays long within the same cluster and seldom jumps between clusters. The transition probability of jumping in one step from vertex v_i to vertex v_j is proportional to the edge weight w_{ij} and is given by $p_{ij} := \frac{w_{ij}}{d_i}$. So the transition matrix

$$P = D^{-1}W$$

If the graph is connected and non-bipartite, then the random walk always possesses a unique stationary distribution $\pi = (\pi_1, \dots \pi_n)^\top$, where $\pi_i = \frac{d_i}{vol(V)}$.

It can be easily noticed that, $L_{rw} = I - P$. So, λ is an eigenvalue of L_{rw} with eigenvector u if and only if $1 - \lambda$ is an eigenvalue of P with eigenvector u. From this point of view the largest eigenvectors of P and the smallest eigenvectors of L_{rw} can be used to describe cluster properties of the graph.

Also there is an equivalence between Ncut and transition probabilities of the random walk:

$$Ncut(A, \bar{A}) = P(X_1 \in \bar{A} | X_0 \in A) + P(X_1 \in A | X_0 \in \bar{A})$$

So, when minimizing Ncut, we actually look for a cut through the graph such that a random walk seldom transitions from A to $\bar{A}$ and vice versa.

4.3 Perturbation Theory Point of view

Perturbation theory comprises methods for finding an approximate solution to a problem, by starting from the exact solution of a related, simpler problem. We first consider the "ideal case" where the between-cluster similarity is exactly 0 and then a "nearly ideal case" where the between-cluster similarity is not exactly 0 and the Laplacian matrices to be perturbed versions of the ones of the ideal case. We consider the perturbed Laplacian matrix $\tilde{L} := L + H$ where L is the Laplacian matrix in ideal case.
Let, $\sigma_{S_1}(L)$ be the set of eigenvalues of L which are contained in S_1 and and V_1 be the eigenspace corresponding to all those eigenvalues. $\sigma_{S_1}(\tilde{L})$ and $\tilde{V_1}$ be the analogous quantities for $\tilde{L}$. Then according to Davis-Kahan theorem,

$$d(V_1, \tilde{V_1}) \leq \frac{||H||}{\delta}$$

where,

$$\delta = \min\{|\lambda - s| : \lambda \text{ is eigenvalue of L}, \lambda \notin S_1, s \in S_1\}$$

d is some distance function measured using canonical angles.
We want to choose S_1 such that both the first k eigenvalues of $\tilde{L}$ and the first k eigenvalues of L are contained in it.

We can see from the theorem that the larger the eigengap $|\lambda_k - \lambda_{k+1}|$ is, the closer the eigenvectors of the ideal case and the perturbed case are,i.e, $H = \tilde{L} - L$ decreases. And hence the better spectral clustering works.

Now, in the ideal case, the entries of the eigenvectors on the components should be "safely bounded away" from 0. If we have two entries u_{1i}, u_{1j} which are very small, then it is unclear how we should interpret this situation. Either we believe that small entries indicate that the points do not belong to the first cluster (which then misclassifies the first data point i), or we think that the entries already indicate class membership and classify both points to the first cluster (which misclassifies point j). For L and L_{rw}, since the eigenvectors in the ideal situation are indicator vectors, this problem will not occur. So in this point of view, unnormalized spectral clustering and normalized spectral clustering with L_{rw} are well justified.

5 Choosing different parameters for our given problem

5.1 The parameters of the similarity graph

While using the similarity graph, one has to choose its connectivity parameter k or ϵ respectively. There are many theoretical results on how connectivity of random graphs can be achieved, but all those results only hold in the limit for the sample size $n \to \infty$. For example, it is known that for n data points drawn i.i.d. from some underlying density with a connected support in $\mathbb{R}^d$, the k-nearest neighbor graph and the mutual k-nearest neighbor graph will be connected if we choose k on the order of $\log(n)$ (e.g., Brito, Chavez, Quiroz, and Yukich, 1997).

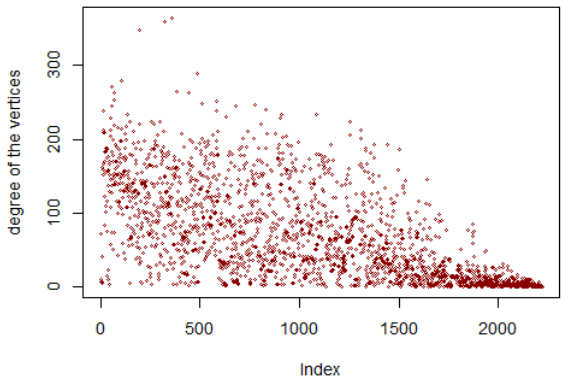
When working with the **k-nearest neighbor graph**, then the connectivity parameter should be chosen such that the resulting graph is connected, or at least has significantly fewer connected components than clusters we want to detect. For small or medium-sized graphs this can be tried out "by foot". For very large graphs, a first approximation could be to choose k in the order of $\log(n)$, as suggested by the asymptotic connectivity results.

But the mutual k-nearest neighbor graph has much fewer edges than the standard k-nearest neighbor graph for the same parameter k. This suggests to choose k significantly larger for the mutual k-nearest neighbor graph than one would do for the standard k-nearest neighbor graph.

For the ϵ-**neighborhood graph**, we suggest to choose ϵ such that the resulting graph is safely connected. To determine the smallest value of ϵ where the graph is connected one has to choose ϵ as the length of the longest edge in a minimal spanning tree of the fully connected graph on the data points.

5.2 Type of graph Laplacian to be used

At first one should always look at the degree distribution of the similarity graph. If the graph is very regular and most vertices have approximately the same degree, then all the Laplacians are very similar to each other, and will work equally well for clustering. However, if the degrees in the graph are very broadly distributed, then it is better to use normalized rather than unnormalized spectral clustering, and in the normalized case it is better to use the eigenvectors of L_{rw} rather than those of L_{sym}.



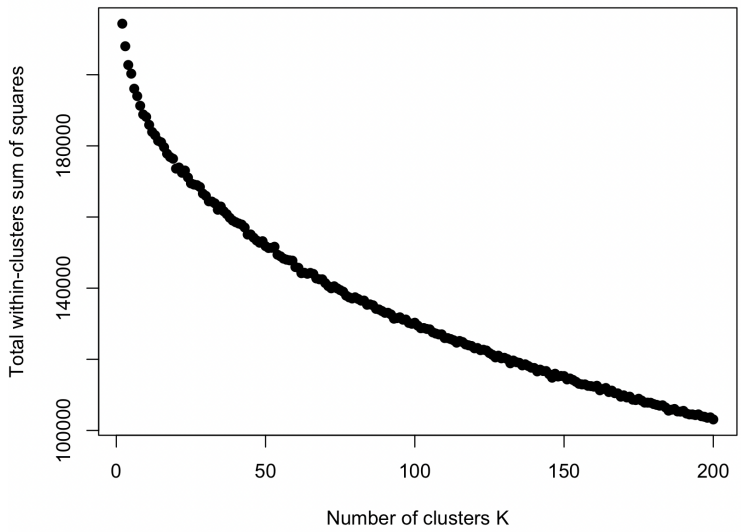
Here we can see degrees of the vertices are very much distributed, so will work with normalized graph clustering with L_{rw}.

5.3 The number of clusters

Choosing the number of clusters is the most important step in this process, and a variety of more or less successful methods have been devised for this problem. In this case one tool which is particularly designed for spectral clustering is the eigengap heuristic, which can be used for all three graph Laplacians. Here the goal is to choose the number k such that all eigenvalues $\lambda_1, \cdots, \lambda_k$ are very small, but λ_{k+1} is relatively large.

6 Output and Results

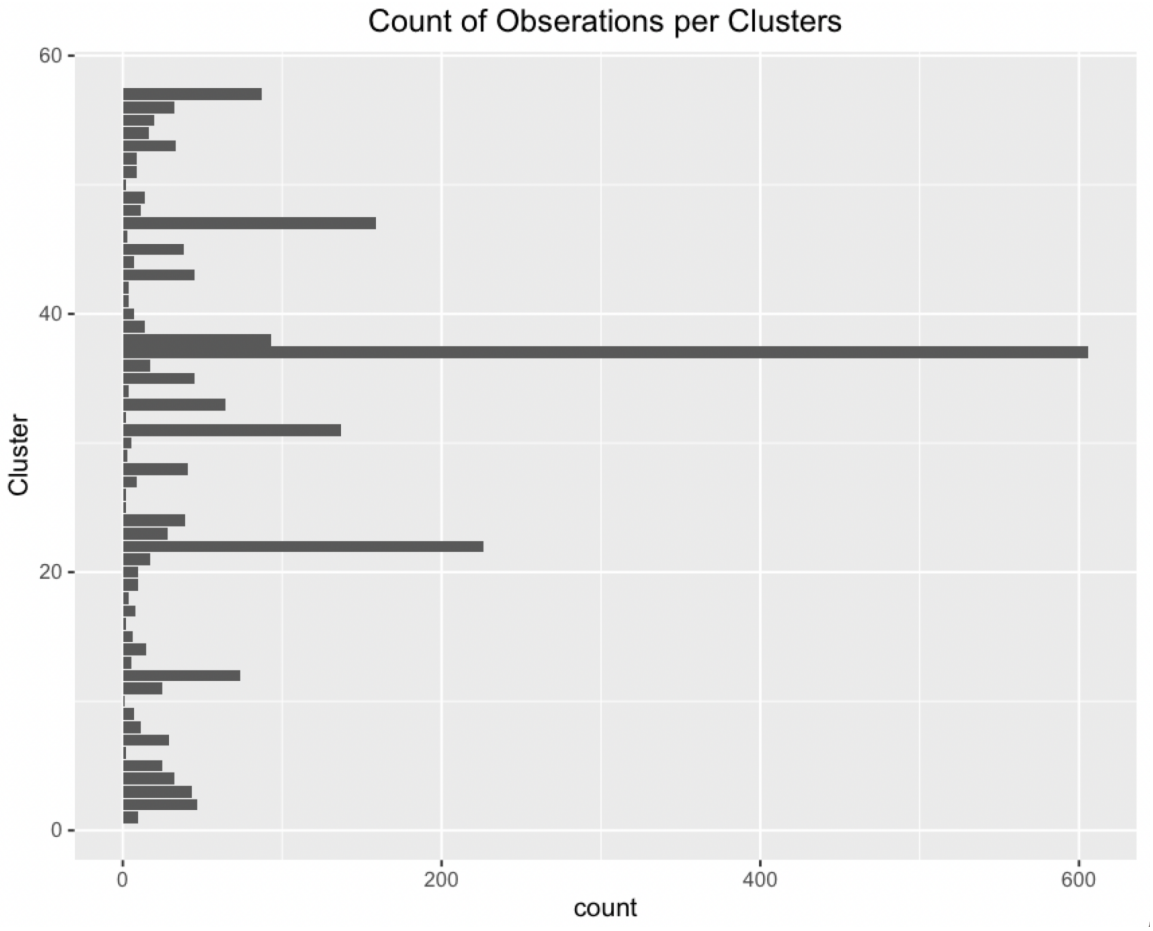
Since no prior information was available to us regarding the data,we had no idea about what should be an approximate number of clusters.Hence we used a simple general method.
The idea is to compute k-means clustering using different values of clusters k. Next, the WSS (within cluster sum of square) is drawn according to the number of clusters. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.The reason of such is because after the bend WSS do not decrease significantly by increasing number of clusters and hence we do not want to make some unnecessary clusters.



So from this image we can see that around $k = 50 - 60$ the bend exists.After that the graph becomes more or less linear. Now as stated before in section 2.2 number of components of a graph is same as the multiplicity of 0 eigen value of it's Laplacian.So given a KNN graph we first calculate the multiplicity of eigen value 0 of it's Laplacian and we select that k for which the multiplicity is around $50 - 60$.
To find the appropriate value of k we used trial and method approach and we created the following table

k	Components of Graph
7	592
10	474
30	183
50	90
70	57
90	42
150	10

So from this table $k = 70$ looks to be a decent choice and we use this value of k.
We performed the Normalised Spectral Clustering algorithm as explained above for a 70-NN graph.Below is given a picture of number of points per cluster.



Some important observations are

- one cluster $(\#37)$ has huge number of observations,as high as 606
- 3 other clusters$(\#22, \#31, \#47)$ have more than 100 observations.
- 23 clusters have less than 10 observations

Finally we give a breakdown of the sum of squares between and within the clusters.

- Within Cluster Sum of Squares: 9.31
- Between Cluster Sum of Squares: 46.40
- Total Sum of Squares: 55.71

Another important question is : if we draw more and more data points, do the clustering results of spectral clustering converge to a useful partition of the underlying space or in other words are our results consistent? We claim that our results are more or less consistent since our n is more than 2000 which is quite large and for normalized spectral clustering results are generally consistent under very mild conditions which are usually satisfied in real world applications.This gives us another reason to prefer normalised spectral clustering above the unnormalised one.

7 Acknowledgement

We give our thanks to Prof. S.S.Mukherjee for his kind advice and support in this project. We are very grateful to Prof. U.V. Luxburg for her "A Tutorial on Spectral Clustering" which helped us a lot.

8 References

- A Tutorial on Spectral Clustering, Ulrike von Luxburg.https://people.csail.mit.edu/dsontag/courses/ml14/notes/Luxburg07_tutorial_spectral_clustering.pdf
- Multiple online sites for K-mean Clustering like https://www.datanovia.com/en/lessons/k-means-clustering-in-r-alg and https://www.geeksforgeeks.org/k-means-clustering-in-r-programming/#:~:text=K%20Means%20Clustering%20in%20R,pre%2Dspecified%20number%20of%20clusters.