241 Project - Spectral Clustering

Claire Chen, Charlotte Wang

December 4, 2021

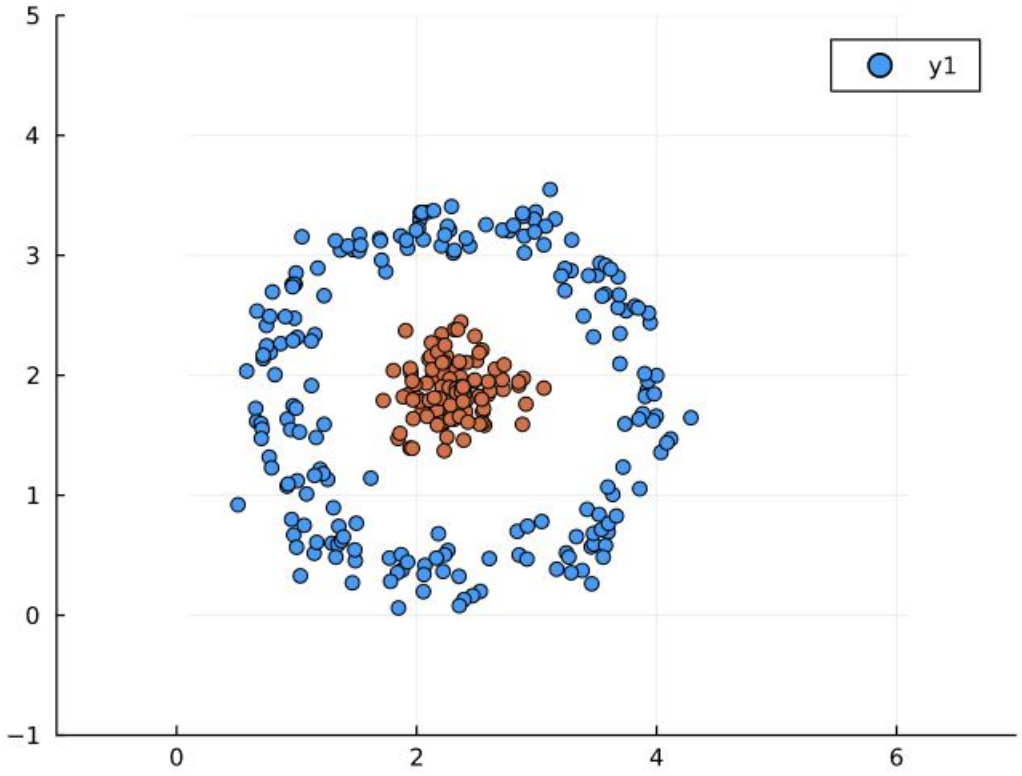


Figure 1: Caption

1 Introduction

In this project, we seek to group similar data points (of non-categorical data) into neighborhood clusters by applying spectral clustering. To achieve this, we create a similarity matrix and generate a Laplacian matrix by subtracting the similarity matrix from the degree matrix. Afterwards, we find the first k eigenvectors of the Laplacian matrix, to form a new $n \times k$ matrix. Then, we apply k-means to this new matrix to get the final clusters. Additionally, we applied this algorithm to several different data sets with different similarity matrices and parameters for the similarity functions. We found that the similarity matrix and its parameters has a large influence on whether spectral clustering perform well.

2 Mathematical Background

The following are the definitions for n, m and k we'll use for the rest of the paper.

- We perform clustering on a set of n data points, each with m numerical features, represented as a $n \times m$ matrix.
- We aim to group the data into k clusters.

2.1 Similarity Matrix

In this paper, the similarity matrix is denoted as W. It is a $n \times n$ matrix calculated from n points by:

$$W_{ij} = sim(i, j)$$

sim is a similarity function that calculates the similarity between point i and point j. It should output a non-negative answer, and it should be symmetric($sim(i, j) = sim(j, i)$). For a true adjacency matrix of a graph with n points, $w_{ij} = 1$ if point i and j are connected, and 0 if they are not. There are several different ways to compute similarity matrices based on n points and m numerical features. We'll perform the following in our code:

- Approach 1: **k-nearest neighbors**
  For each point in the data set, we connect this vertex v_i to its k-nearest neighbors v_j. Each point v_i would form directed edges to k other vertices; however, for the purpose of generating a similarity matrix, we ignore the directions of the digraph and use 1 and 0 to denote existence of a k-nearest neighborhood relationship between vertices. While this is not a symmetric matrix, the k-nearest neighbors graph is a well approximation for our purposes.
- Approach 2: **Fully-connected graph with Gaussian similarity function**
  We connect all points in the data set with mutually positive similarity and weight the edges using s_{ij}, which is the output of the Gaussian similarity function $s(x_i, x_j) = exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$. Note that, in essence, the function computes the inverse of the Euclidean distance and takes into account the parameter σ, the width of the neighborhoods. The larger the distance, the lower the output from this function.

2.2 Degree Matrix

In the context of graph theory, the degree matrix are denoted by D. Specifically it is defined as following:

$$d_i = \sum_{j=0}^{n} w_{ij}$$

The following properties are true

$$\begin{cases} D_{ii} = d_i \\ D_{ij} = 0 \quad \forall i \neq j \end{cases}$$

Note here that D is a diagonal matrix with non-negative numbers on the diagonal.

If W is an adjacency matrix, then d_i represents the number of points the ith point is connected to. If W is a weighted similarity matrix, then d_i would represent the sum of the edges to which the ith point is connected.

2.3 Laplacian Matrix

The unnormalized Laplacian Matrix can be calculated as :

$$L = D - W$$

Assuming that W is a symmetric matrix, and D is a diagonal matrix defined as both, the Laplacian matrix will have the following properties:

- **Positive semi-definite**. The proof is as follows:
  W.T.S. For every $f \in \mathbb{R}^n$, the Laplacian matrix L satisfies the following property:

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2 \geq 0$$

We proceed by invoking the definition of d_i

$$\begin{aligned}
f^T L f &= f^T (D - W) f && \text{(since } L = D - W\text{)} \\
&= f^T D f - f^T W f \\
&= \sum_{i,j=1}^{n} D_{ij} f_i f_j - \sum_{i,j=1}^{n} f_i w_{ij} f_j \\
&= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i w_{ij} f_j \\
&\text{(since } D \text{ is an } n \times n \text{ diagonal matrix, for } i, j \in [1, n],\ D_{ij} = 0 \text{ and } f_i^T D_{ij} f = 0 \text{ when } i \neq j\text{)} \\
&= 2 \times \frac{1}{2} \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i w_{ij} f_j \\
&= \frac{1}{2} \sum_{i=1}^{n} d_i f_i^2 + \frac{1}{2} \sum_{j=1}^{n} d_j f_j^2 - \sum_{i,j=1}^{n} f_i w_{ij} f_j \\
&= \frac{1}{2} (\sum_{i=1}^{n} d_i f_i^2 + \sum_{j=1}^{n} d_j f_j^2 - 2 \sum_{i,j=1}^{n} f_i w_{ij} f_j) \\
&= \frac{1}{2} (\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} f_i^2 + \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ij} f_j^2 - 2 \sum_{i,j=1}^{n} f_i w_{ij} f_j) \\
&\qquad \text{(by definition of the degree of a vertex } v_i \in V,\ d_i = \textstyle\sum_{j=1}^{n} w_{ij}\text{)} \\
&= \frac{1}{2} (\sum_{i,j=1}^{n} w_{ij} f_i^2 + w_{ij} f_j^2 - 2 w_{ij} f_i f_j) \\
&= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i^2 + f_j^2 - 2 f_i f_j) \\
&= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2 \\
&\geq 0 && \text{(since every } w_{ij} \geq 0 \text{ and } (f_i - f_j)^2 \geq 0\text{)}
\end{aligned}$$

  which is the definition of positive semi-definite matrices.

- **The eigenvalue 0 has at least 1 eigenvector:** $\mathbb{1}$. The ith row of $L\mathbb{1} = \sum_{j=1}^{n} L_{ij}$. By definition for L, this equals $d_i + \sum_{j=1}^{n} -W_{ij}$. Since d_i is the sum of the weights for each row, adding d_i to the negative of the weights in each row would equal 0. Thus, $L\mathbb{1} = \mathbb{0} = 0\mathbb{1}$.

2.4 Interpretation of Eigenvalues and Eigenvectors

Next, we take the k eigenvectors corresponding to the k lowest eigenvalues to form a $n \times k$ matrix. This is because of the following:
Taking the view of the W as an adjacency matrix, the multiplicity of 0 as an eigenvalue is equal to the number of connected components. A set of n points is connected if there is a path from each element to every other element in the set, and is not connected to any element outside of the set. Furthermore, suppose the multiplicity of 0 is k, then the first kth eigenvectors will have a constant for the points in

the kth cluster, and 0 everywhere else.
The proof is as below: Assume f is an eigenvector with eigenvalue 0 of L:

$$Lf = 0f \iff Lf = \mathbb{0} \iff f^T Lf = f^T \mathbb{0} \iff f^T Lf = 0$$

It follows that $f^T Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$. Since we want the sum to be 0 and all terms are positive, this means $f_i = f_j$ when w_{ij} is positive, or when 2 points are connected.

If the graph has 1 connected component, then 0 only has one eigenvector, f, and $f_i = constant$ for all $1 \leq i \leq n$.

If the graph has k connected components, then let's arrange the rows of L so that rows of connected points are together and the columns are arranged the same way

$$L = \begin{pmatrix} \text{rows for component 1} \\ \dots \\ \text{rows for component k} \end{pmatrix}$$

This will form a diagonal block matrix with k blocks, such as each diagonal block is an adjacency matrix for a graph with 1 connected component, each with one eigenvector($\mathbb{1}$), corresponding to an eigenvalue of 0. Thus, the multiplicity of the overall matrix for eigenvalue 0 is k. Also following, the eigenvectors of the overall matrix are the eigenvectors for each block filling in 0 for the rest of the places. This means that each of the k eigenvector f can be defined as:

$$f_i = \begin{cases} a \text{ for some } a \in \mathbb{R} & \text{if point } i \text{ is in the current connected component} \\ 0 & \text{else} \end{cases}$$

For example, suppose we have a graph where the following points are connected: {(1, 5, 6), (2, 3), (4)}, and $k = 3$. The eigenvector matrix is:

$$\begin{pmatrix} c_1 & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_3 \\ c_1 & 0 & 0 \\ c_1 & 0 & 0 \end{pmatrix}$$

for some $c_1, c_2, c_3 \in \mathbb{R}$

2.5 Spectral gap

As alluded to in the aforementioned sections, effective spectral clustering relies on choosing a value k such that all eigenvalues (in increasing order) $\lambda_1, \lambda_2, \cdots, \lambda_k$ are very small and λ_{k+1} is large. To employ this heuristic, we can compute $|\lambda_k - \lambda_{k+1}|$ referred to as the spectral gap, or eigengap, which has a large value when a given data set has k pronounced clusters.
Suppose f is an approximation of an eigenvector of L.

$$\begin{aligned} f^T Lf &= f^T(\lambda_c f) && \text{(by definition of eigenvalues that } A\mathbf{x} = \lambda\mathbf{x} \text{ for any } \mathbf{x} \in \mathbb{R}^n) \\ &= \lambda_c(f^T f) && \text{(by commutativity of scalar multiplication by } \lambda_c) \\ &= \lambda_c ||f||^2 && \text{(which is a scalar multiple of the dot product of } f) \end{aligned}$$

A good partition of points means that $\lambda_c \approx 0$. Thus, the goal is to make $f^T Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2$ as small as possible. When the weights w_{ij} are greater than 0, or when points are connected, we seek obtain $f_i \approx f_j$. In the ideal case, f needs to have constant values for all connected vertices.

There is one major motivation behind finding the eigengap-finding the number of clusters k. The larger the eigengap, the closer the model is to ideal case in which the eigenvectors are piecewise constant in the k neighborhoods, and thus the better the spectral clustering algorithm performs.

All else being equal, the spectral gap of a data set generated using the k-nearest neighbors algorithm with an adjacency matrix is greater than the spectral gap generated by the fully connected graph, which utilizes an similarity matrix.

As shown below in section 3.5, the k-nearest neighbors model outperforms the fully connected graph when maximizing the eigengap. This results from the fact that data points are connected to its k-nearest neighbors only, so solving the mincut problem to partition the graph into k clusters with perfect cuts is somewhat possible. However, when the graph is fully connected, a near-perfect solution no longer exists and thus the eigen-gap is larger.

2.6 Wrapping up with K-means

Taking the real-valued $n \times k$ matrix, we use the k-means algorithm to cluster the n points into k clusters. This is first less expensive than clustering on a $n \times m$ data set, as $m >> k$ for large data sets.

2.7 Summarize

To summarize, this is the following algorithm we performed:

- Take the $n \times m$ matrix and compute a $n \times n$ similarity matrix using k-neighbors or fully connected graph with Gaussian.
- Compute the degree matrix and the laplacian vis $L = D - W$
- Find the k eigenvectors of the laplace matrix corresponding to the lowest k eigenvalues. Put them as rows of a $n \times k$ matrix.
- Feed this $n \times k$ matrix into k-means to cluster the n points into k clusters. This is the result of the algorithm

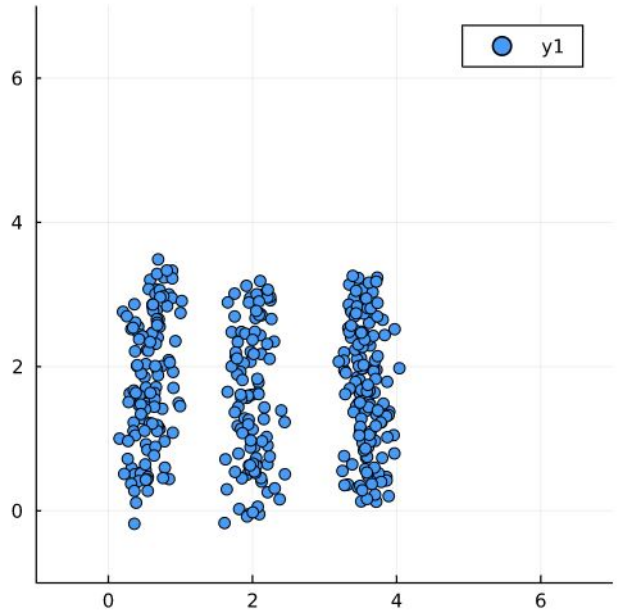
3 Results

3.1 Data

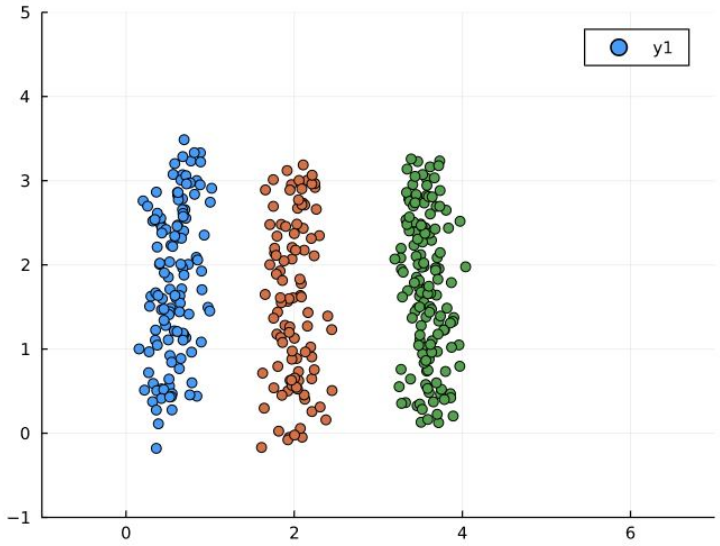
The data used for this project were generated manually using the Python library "Drawdata". As a result, all data are a series of 2-dimensional points.

3.2 Lines Data

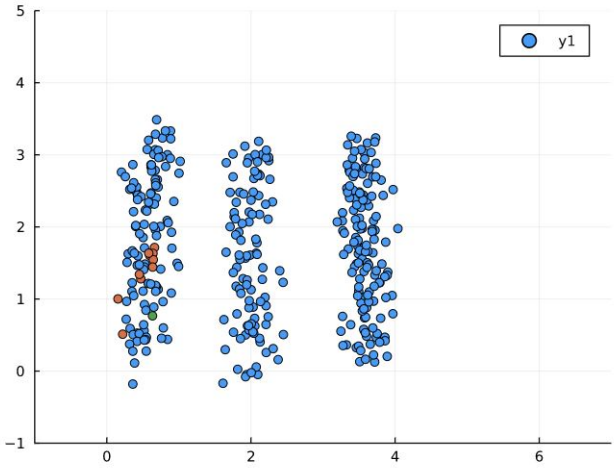
The original data plot is the following:

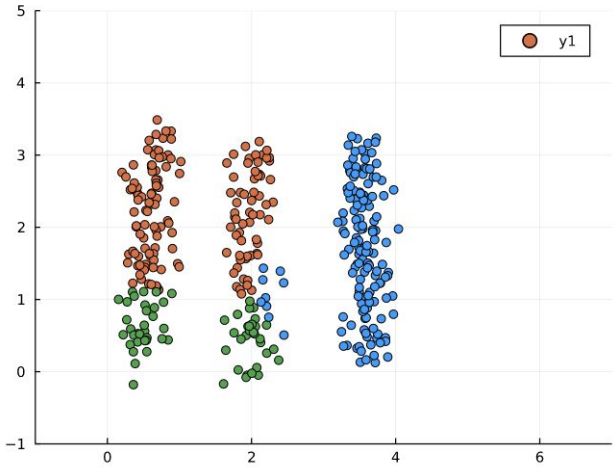


The following graphs are the result when we ran it with different similarity matrices with different parameters. They are colored according to the cluster they are assigned, and we used $k = 3$ for all.



This was the result of both k-neighbor clustering with $k = 20$, and fully-connected graph with $\sigma = 0.1$. These parameters were chosen after testing with a few ranges. They roughly match the general aim from the paper. For k-neighbors, it advised to make the number of neighborhoods much larger than k, but each neighborhood still large enough to represent mini-clusters of data. The paper recommended $log(n)$ as a starting number, and we had around 400 data points.





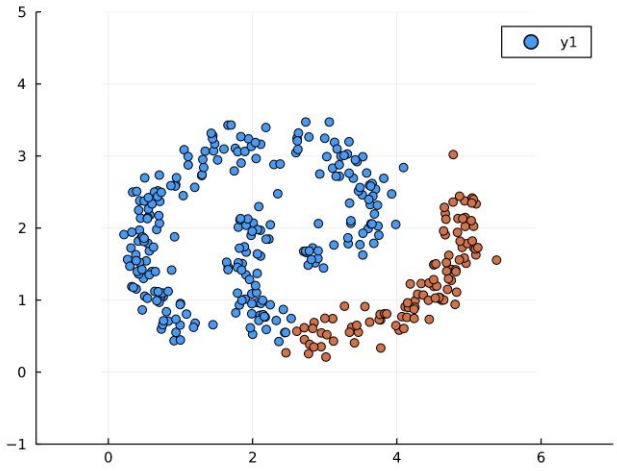
The above graphs were the result of problematic parameters where the sigma is too small($\sigma = 0.01$) and too large($\sigma = 1$) respectively.

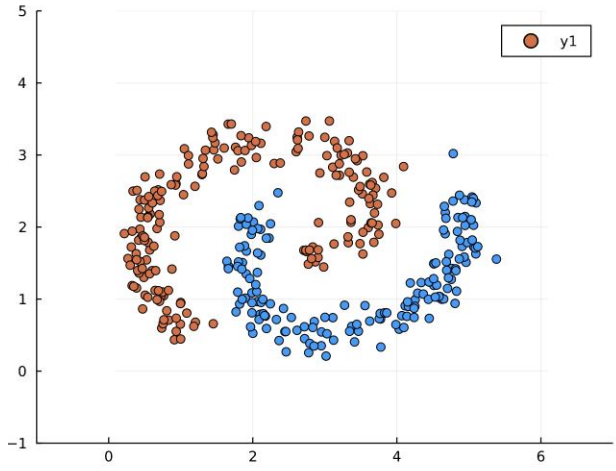
Here we see that too small of a value for σ leads to the case from the paper from a few points are chosen for 2 clusters and the majority of points are put into 1 cluster. This is because the small σ allowed most of the points to have high similarity. Thus, they were all placed into the same connected component, and k-means just picked a few points for the other two cluster since we specified that there must be 3.

In the case where σ is too large. It seem to yield similar results to running k-means on the raw data set. A very large σ resulted in the similarities values being very small, thus each point was its own connected component, and the transformation did not alter the data much, leading to the result being similar to run without transformation with k-means.

3.3 Moon Data

The following data set shows an example where the algorithm used for the similarity matrix affect the results of spectral clustering.



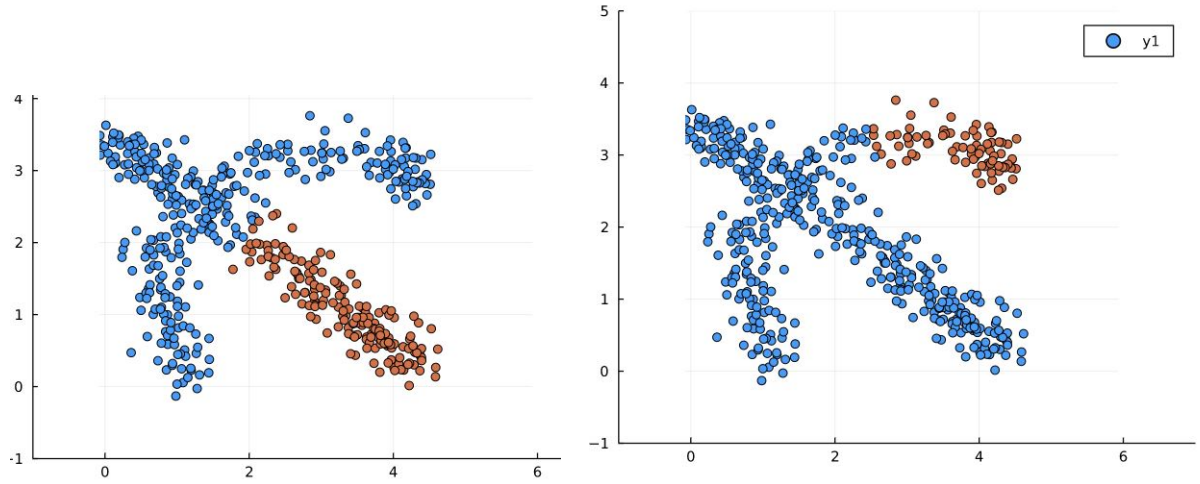


The left graph is the result of running with a fully connected similarity graph with $\sigma = 0.3$, and the right graph is with the k-neighbors graph with $k = 15$. Both parameters have been tuned by hand to the best they can be. By intention of how the graph was drawn, the right graph was the successful clustering.

This difference in clustering is a result of the fact that for this data set, cluster is not defined via distance(a point is close to points in its own cluster and far away from points that are not), but rather by connectedness(the clusters form a curved line). This allows k-neighbors, which is less sensitive to distance than the fully-connected graph, to work better.

3.4 Intersecting Data Points

The following is a data set that does not work well with spectral clustering with either similarity algorithms.

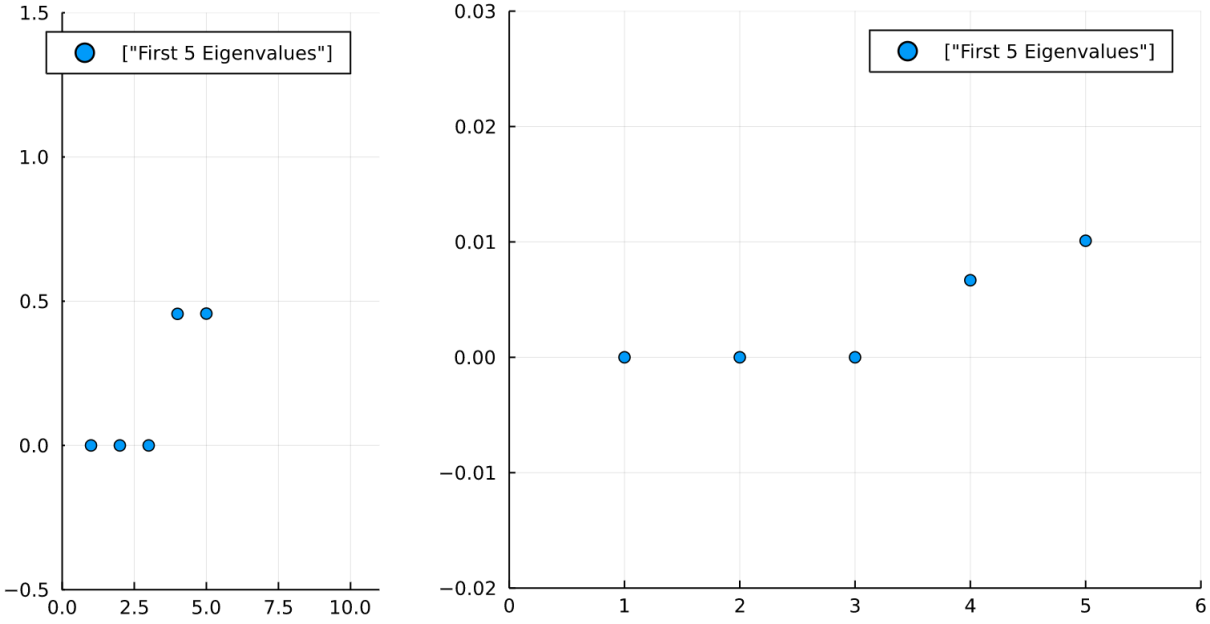


The data set was intended to be two overlapping clusters. However, both similarity matrix methods yielded clusters where one arm of the cross is identified as a cluster. This is because those clusters are not defined by distance, as the lines data set is, not is it defined by connectivity, as the two cluster overlap, and are thus connected together. Thus, neither similarity methods form the appropriate connected components.

3.5 Discussion of Eigenvalues and Eigenvectors

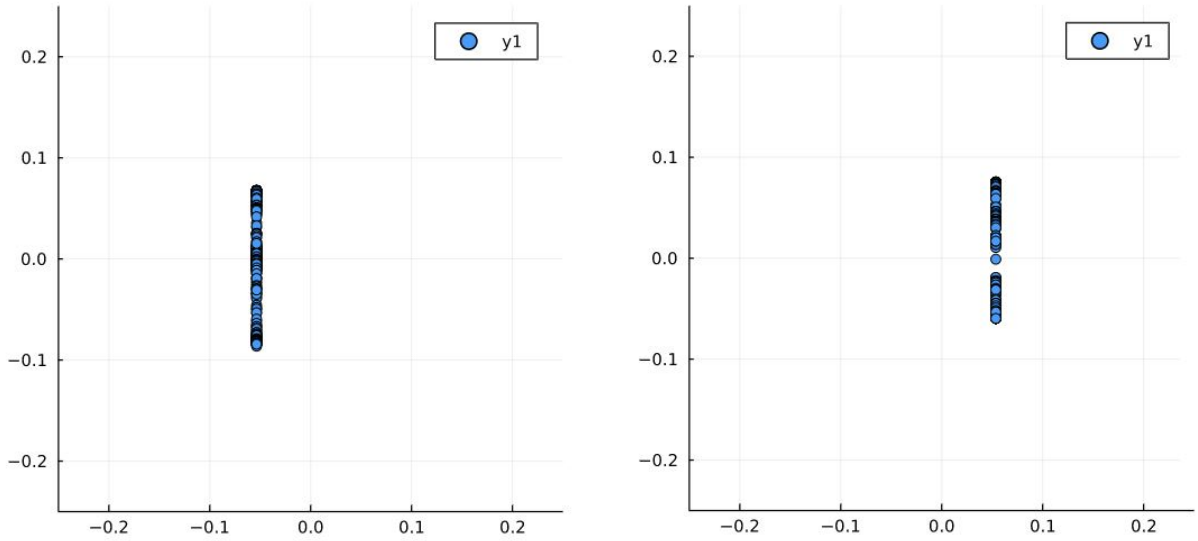
First, we'll validate the discussion about the multiplicity of eigenvalue 0 to the number of connected components and the eigen-gap heuristic. Here are the first 5 eigenvalues of the lines data set with k-nearest neighbors($k = 20$) and fully-connected graph($\sigma = 0.1$), respectively:

```
-7.771561172376096e-16    -3.8425454617099913e-16
 3.3306690738754696e-16    1.7148842268082412e-15
 1.3322676295501878e-15    2.7960913085249328e-11
 0.4561296045487133        0.006676326131240342
 0.4570995664803692        0.010100216357270382
```



As we can see, for both the first 3 eigenvalues are all extremely small compared to the next 2. This is essentially pronounced for the eigenvalues on the left. The first 3 eigenvalues for k-neighbors are essentially 0(they are only not because of how floating point arithmetic works). Under the context of eigenvectors cutting the set of points into different clusters, the first 3 eigenvectors thus make almost perfect cuts as their eigenvalues are almost 0. The first 3 eigenvalues for the fully connected graph are also small, whoever they are not as low as for the $k$neighbors graph. This is because all vertexes are connected in the fully connected graph, they only vary by their similarity. Thus, it is impossible to cut perfectly, and it's only possible to minimize similarity between clusters to a small value, but never 0.

Finally, some intuitive explanations of spectral clustering modeled it as a projection of the original matrix into points that are easily sorted with k-means. In the visual sense, points of a cluster are drawn closer together. Here, we plot the eigenvectors matrix for the moon data, where $k = 2$.



The left is a graph of the eigenvector matrix for an unsuccessfully fully connected graph, and the right is a graph of the eigenvector matrix for the successful k-neighbors graph. Shown clearly is that in the successful example, the data set are separated into 2 clusters. There is a clear break at around 0. Meanwhile, on the left graph, it is one unbroken lines of data, explaining why the k-means algorithm had been unable to separate the clusters.

4 Appendix: Code

The full results, including the eigenvalues, vectors and laplace matrix for each example, are in the code appendix.

The code used to calculate each step of the algorithm can also be found in the appendix.

The PDF of the notebook is appended after the paper. For a more human readable version, visit this link.

The first 5 pages are the code. Rest are just results from the data sets.

5 References

Below are the 3 references we used:

- Aoullay, A. (2018, June 03). Spectral Clustering for beginners. Retrieved from https://towardsdatascience.com/spect clustering-for-beginners-d08b7d25b4d8
- Fleshman, W. (2019, February 21). Spectral Clustering. Retrieved from https://towardsdatascience.com/spectral-clustering-aba2640c0d5b
- Luxburg, U. V. (2007). A tutorial on spectral clustering. Statistics and Computing, 17(4), 395-416. doi:10.1007/s11222-007-9033-z