Exploration and Implementation of Spectral Clustering

A Powerful Non-Parametric Clustering Technique

Thomas Keane

School of Mathematics and Statistics
University College Dublin
07/01/2019

Contents

Abstract

Spectral Clustering is a powerful non-parametric clustering algorithm. This paper explores some of the choices which must be made through out its implementation. We examine some the common choices for the Similarity, Adjacency, and Laplacian matrices. We also aim to explain the basic principles which provide a basis for Spectral Clustering. This is done in an effort to highlight the dependence of clustering results on the choices made during the algorithm. Currently, Spectral Clustering lacks documentation surrounding the process of choosing the optimal Similarity and Adjacency measures. This coupled with a lack of meaningful performance measure for parameter tuning, means the Spectral Clustering algorithm is unreliable despite the huge potential it possesses.

1 Introduction

Clustering involves grouping data points based on similarities. This gives rise to sub groupings and trends in data that may not have been obvious otherwise. Due of this clustering has become a popular method of data analysis and data exploration in recent years. Clustering aims to minimize differences within clusters and maximize differences between clusters. This ensures that members of the same cluster are similar while members of other clusters are different. This has numerous applications on a range of data types such as clustering of customers based on sales data, or clustering of congressmen based on voting data. However, for most clustering algorithms to perform optimally they require a high degree of separability between clusters. Spectral Clustering was discovered as a method to overcome a lack of separability between clusters in a dataset.

Spectral Clustering utilises a more commonly known clustering technique called K-means. K-means is a clustering algorithm that minimises differences within the Euclidean space. K-means aims to minimise the within cluster sum of squares. Therefore, to perform optimally this algorithm requires a high degree of separability within the raw data as the algorithm does not maximise between cluster differences. In high dimensional spaces, the required level of separability may not be available, and this leads to unstable clustering results. We use methods like Spectral Clustering to shift our data into lower dimensions where a higher level of separability may be found. Spectral Clustering utilises the matrix representation of a graph known as the Laplacian.

A graph $\boldsymbol{G}$ is a set of ordered pairs of vertices and edges. Vertices are points which edges connect to and are sometimes referred to as nodes. Edges are connections or links between nodes and can be either weighted or unweighted. If an edge is weighted it is represented by the value of its weight. If an edge is unweighted, it is represented by a 1 if a link is present and a 0 if a link is not present. Graphs are represented by an adjacency matrix $\boldsymbol{A}$ with $\boldsymbol{A_{ij}}$ being equal to the weight of the edge between node i and node j. For our purpose's graphs will be symmetric. This means that if node i is connected to node j then node j is connected to node i i.e. $\boldsymbol{A}$ will be a symmetric matrix and $\boldsymbol{A_{ij}} = \boldsymbol{A_{ji}}$. Symmetry is a key property required by Spectral Clustering and

will be discussed further in Section 2.3.2. The degree of a vertex is the sum of all edges connecting to it. All the necessary graph prerequisites can be found in the Preliminaries chapter of [1].

To demonstrate and explore Spectral Clustering as a method we will be using multiple datasets which more traditional methods perform poorly on. We will explain the poor performance of the K-means algorithm on this data and then show how Spectral Clustering may improve upon this.

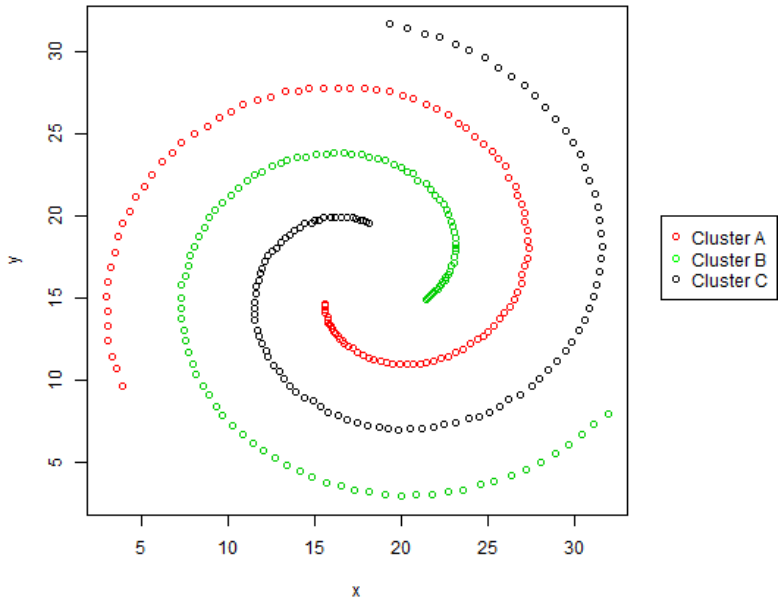


Figure 1: Spirals Data: True Clustering

The first test dataset is a set of 3 spirals interlocked around each other. This is a classic example of data which K-means will perform poorly on due to the shape of clusters that it can detect.

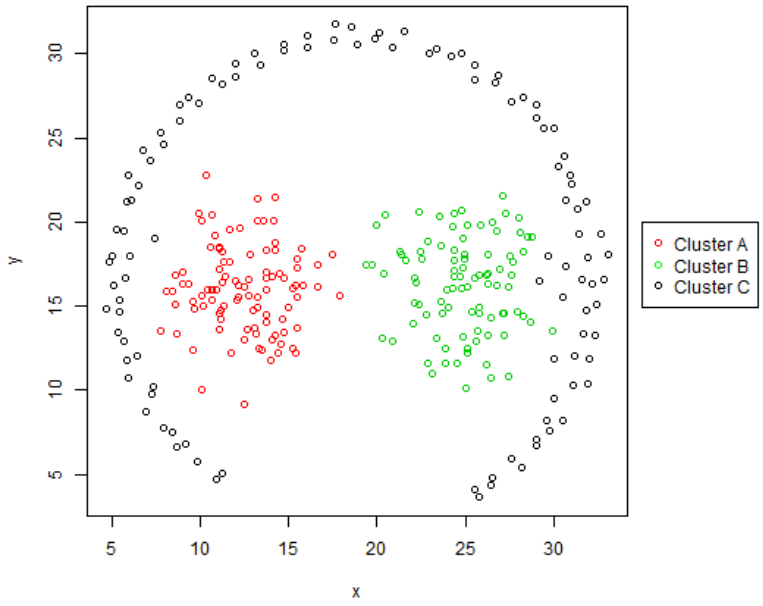


Figure 2: Pathbased Data: True Clustering

The second dataset is two smaller circular clusters, with a third cluster arching over them. K-means will also perform poorly on this data due to the shape of the third cluster.

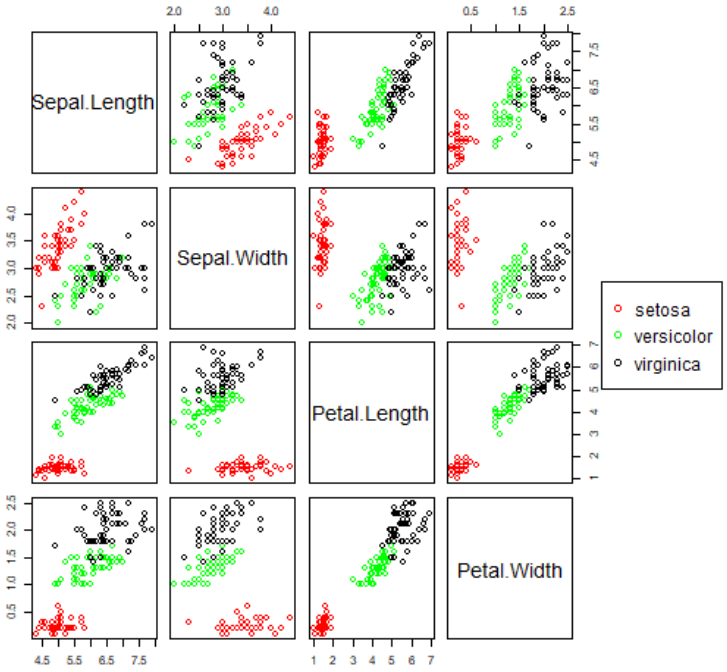


Figure 3: Iris Data: Known Species

The third dataset is the well-known iris dataset. K-means will perform poorly on this data due to the low separability between the versicolor and virginica species of iris.

2 Methodology

2.1 K-means Algorithm

The K-means clustering algorithm is a common form of clustering which aims to minimise the within cluster sum of squares according to:

$$\text{Total Within Sum of Squares } = \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik} \left\| x_i - \mu_k \right\|_2{}^2$$

Spectral Clustering may be viewed as K-means clustering with specific data pre-processing steps. As mentioned earlier these steps seek to transform the data into lower dimensions to achieve a higher degree of separability between clusters. To understand why this may be necessary we look to the K-means pseudo algorithm and the total within sum of squares.

1) Initialise K random centres and set equal to your initial μ_{k}.
2) Update z_{ik} such that within sum of squares is minimized.
3) Recalculate μ_{k} based on the new cluster Assignment.
4) Return to step 2 and repeat until convergence.

Spectral Clustering looks to address two of the largest issues with the K-means algorithm. Firstly, as the sum of squares is defined within the Euclidean space K-means can only detect clusters which are circular in shape. Secondly, the K-means algorithm focuses on minimising the within cluster differences. It does not allow for maximization of between cluster differences. Therefore, it is necessary to ensure that clusters have a high level of separability for optimal clustering.

2.2 Matrix Introduction

2.2.1 Similarity Matrix

The first step of the Spectral Clustering algorithm is the construction of a similarity matrix, $\boldsymbol{S}$. We first choose a similarity measure. There are many options available and the optimal choice is dependent on the nature of the data. A common choice for the similarity measure is the Gaussian kernel which is a popular measure within the machine learning community. However, when presented with time series data it may be possible that a correlation-based similarity measure would perform better. This will be discussed in greater detail in Section 2.3.2. A similarity measure returns a weighted value for the level of similarity between two points. We construct a similarity matrix by creating a N x N matrix and setting $\boldsymbol{S_{ij}}$ equal to the level of similarity between point i and point j. It should be noted that we will use the similarity matrix to construct the adjacency matrix, and so it is also required to be a symmetric matrix i.e. $\boldsymbol{S_{ij} = S_{ji}}$.

2.2.2 Adjacency Matrix

Now we can use the similarity matrix to construct the adjacency matrix, $\boldsymbol{A}$. The adjacency matrix is necessary to construct the Laplacian. It is quite common to simply set the adjacency matrix equal to the similarity matrix ensuring that the diagonal is equal to 0. This creates an undirected, weighted graph. However, it can be more optimal within certain datasets to perform binarization on the similarity matrix. This requires using a thresholding technique like $\varepsilon -$ neighbourhood or k-nearest neighbours. The advantages of using either of these approaches is the resulting adjacency matrix is sparse. This reduces computational time and problem complexity with a small penalisation in accuracy [2]. This can be achieved as the edges carrying the most information are preserved while less important edges are reduced to 0.

2.2.3 Laplacian Matrix

The graph Laplacian can come in many forms. The most basic is the un-normalized Laplacian which is defined in Section 2.3.4. For Spectral Clustering a normalised Laplacian can be used as it produces normalised eigenvalues. This allows for easier identification of the optimal number of clusters. The graph Laplacian has many interesting properties, some of which are utilised in Spectral Clustering. This area of study is known as spectral graph theory. One property that arises from the Laplacian matrix is that for every vector $\boldsymbol{x} \, \epsilon \, \mathbb{R}^N$ we have that:

$$\boldsymbol{x^T L x} \, = \, \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{A_{ij}} \left(\boldsymbol{x_i} - \boldsymbol{x_j} \right)^2$$

This result gives many interesting properties. First, we can conclude that $\boldsymbol{L}$ is a positive semi definite matrix. This means that all eigenvalues are non-negative. This can be extended further, and we can conclude that the smallest eigenvalue of $\boldsymbol{L}$, λ_0, is equal to 0 and has a corresponding constant one eigenvector. It is this result that gives the defining property of the Laplacian for its use in Spectral Clustering. It can be proven that the multiplicity of λ_0, m, is equal to the number connected components in $\boldsymbol{A}$. It also holds that the corresponding eigenvectors span the eigenspace of λ_0. Proof of these properties for the un-normalised and normalised case can be seen in [3]. To understand why this important, we must introduce the Spectral Clustering pseudo algorithm.

2.3 Spectral Clustering

2.3.1 Pseudo Algorithm

Spectral Clustering can involve a near in-exhaustive list of choices and the specific algorithm will depend heavily on these choices. The pseudo algorithm given here is sparse in specific detail but will work as a guideline for Spectral Clustering. Some examples of possible options will be discussed in the following sections and any nuanced changes required to the general algorithm will be noted there for each choice.

1) Choose valid similarity measure and construction similarity matrix, $\boldsymbol{S}$.
2) If applying an adjacency measure use it to construct the adjacency matrix, $\boldsymbol{A}$. If not, simply set $\boldsymbol{A} = \boldsymbol{S}$.
3) Construct a valid graph Laplacian $\boldsymbol{L}$. This can be either un-normalized or normalised.
4) Compute the first K eigenvectors of $\boldsymbol{L}$. Depending on your choice of Laplacian the first K eigenvectors will either correspond to:
   a) The eigenvectors corresponding to the K smallest eigenvalues.
   b) The eigenvectors corresponding to the K largest eigenvalues.
5) Column bind the eigenvectors to form a N x K matrix, $\boldsymbol{Q}$.
6) Using K-means fit K clusters on the rows of $\boldsymbol{Q}$.

The pseudo algorithm gives an indication why the properties of the Laplacian are important. To demonstrate this, we will consider a Laplacian generated on 3 distinct clusters. This means we can construct 3 block diagonal matrices $\boldsymbol{L_1}, \boldsymbol{L_2}, \boldsymbol{L_3}$ within $\boldsymbol{L}$.

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L_1} & & \\ & \boldsymbol{L_2} & \\ & & \boldsymbol{L_3} \end{bmatrix}$$

Taking each $\boldsymbol{L_i}$ individually and calculating the first eigenvector of each, we can generate 3 eigenvectors $\boldsymbol{z_1}, \boldsymbol{z_2}, \boldsymbol{z_3}$. These will be contained within the

first 3 eigenvectors of $\boldsymbol{L}$, $\boldsymbol{v_1}$, $\boldsymbol{v_2}$, $\boldsymbol{v_3}$ respectively.

$$\boldsymbol{v_1} = \begin{bmatrix} \boldsymbol{z_1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \boldsymbol{v_2} = \begin{bmatrix} 0 \\ \vdots \\ \boldsymbol{z_2} \\ \vdots \\ 0 \end{bmatrix} \quad \boldsymbol{v_3} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{z_3} \end{bmatrix}$$

Thus, when we combine these eigenvectors column wise to generate $\boldsymbol{Q}$, it is clear that the rows of this matrix will be separated into 3 clusters. These clusters will match the clusters within the data and those indicated by the Laplacian. While this is the ideal case with 3 distinct clusters, it gives an indication of how Spectral Clustering identifies clusters. It also highlights the importance of the non trivial task of choosing a similarity measure. Each choice within the spectral algorithm must be made carefully made for optimal results. We will now examine each set of choices and discuss some possibilities.

2.3.2 Choosing a Similarity measure

As was mentioned in Section 2.2.1, we must choose a similarity measure to construct our similarity matrix. This decision is dependent on the nature of the data and many functions can be used as a measure of similarity. To be considered a valid similarity measure for Spectral Clustering it must:

1) Produce non-negative values.
2) Produce symmetric values.

There is no structured framework to choose a similarity matrix and these next 3 merely serve as examples which commonly appear. One such similarity measure is the Gaussian kernel:

$$\boldsymbol{S_{ij}} \;=\; \exp(-\frac{|\boldsymbol{x_i} - \boldsymbol{x_j}|}{2\sigma^2})$$

This is a very common kernel in machine learning and is seen in many algorithms such as SVMs. It produces symmetric, non-negative values as required

and in fact returns values between 0 and 1. It also has great versatility due to the scaling factor seen in the denominator. The σ^2, referred to as the inverse kernel width, allows for scaling of similarity values such that points from the same cluster produce relatively larger values than points from different clusters. This is a tuning factor towards cluster density relative to separability. It is not possible to give a guideline on this value as it is dependent on the scale of the original data. Most software implementations rely on a grid search combined with a measure of cluster performance such as Total within cluster sum of square to identify an optimal value. However, generally a smaller value of sigma is best for more dense clusters.

One fault that arises with the Gaussian kernel and its inverse kernel width is when dealing with clusters of different densities. If clusters vary in density, then it can extremely difficult to tune this parameter. This lead to a method with a varying scaling factor called local scale [4]. Given a valid distance metric d, usually the euclidean, local scale is defined as follows:

$$\boldsymbol{S_{ij}} \;=\; \exp(-\frac{d(\boldsymbol{x_i}, \boldsymbol{x_j})^2}{d(\boldsymbol{x_i}, \boldsymbol{x_p})d(\boldsymbol{x_j}, \boldsymbol{x_p})})$$

where $\boldsymbol{x_p}$ is the pth neighbour of $\boldsymbol{x_i}$
This over comes the issue of varying cluster densities and the choice of p is from the $\mathbb{N}$ which is a reduction as the choice of σ is from $\mathbb{R}^+$. It also allows for good flexibility with various distance metrics depending on the nature of the data.

Both of these similarity measures can serve as good starting points. However, if we were to consider a time series dataset better results may be found elsewhere. This leads to another potential similarity measure in the standard correlation matrix.

$$\boldsymbol{S_{ij}} \;=\; \frac{\sum_{l=1}^{P}(\boldsymbol{x_{il}} - \bar{\boldsymbol{x}}_{\boldsymbol{i}})(\boldsymbol{x_{jl}} - \bar{\boldsymbol{x}}_{\boldsymbol{j}})}{\sqrt{\sum_{l=1}^{P}(\boldsymbol{x_{il}} - \bar{\boldsymbol{x}}_{\boldsymbol{i}})^2 \sum_{l=1}^{P}(\boldsymbol{x_{jl}} - \bar{\boldsymbol{x}}_{\boldsymbol{j}})^2}}$$

Where $\boldsymbol{x_{il}}$ refers to the lth feature of the ith data point and $\bar{\boldsymbol{x}}_{\boldsymbol{i}} = \frac{1}{P}\sum_{l=1}^{P}\boldsymbol{x_{il}}$

When examining time series the correlation matrix may serve as a valid measure of similarity as it will give a high level of similarity when points

have similar changes over time. It should be noted that the correlation matrix gives negative values. Therefore, depending on application it may be necessary to take either the absolute value of the correlation or translate the matrix from $[-1, 1]$ to $[0, 2]$. A correlation based algorithm known as nearest correlation [5] has been proposed as a measure of similarity however the proposed algorithm is complex and computationally intensive. Due to this local scale is preferred in most situations [6].

2.3.3 Choosing an Adjacency measure

Adjacency measures are most commonly utilised on large, high dimensional datasets. Our similarity matrix is an N x N matrix and so scales rapidly with dataset size. This can lead to complex and intensive calculations particularly on eigenvalues and eigenvectors. However, for sparse matrices these calculations have been optimised [2] and so we implement an adjacency measure to utilise these optimisations. An optimal adjacency measure reduces the similarity matrix to a sparse matrix while retaining as much information about cluster membership as possible. This can be a difficult balance to achieve and we have multiple options in the adjacency measures.

ε- neighbourhood is a simple adjacency measure. It requires choosing a value for ε and converting the similarity matrix according to the following formula

$$\boldsymbol{A_{ij}} = \begin{cases} 1, & \text{if } \boldsymbol{S_{ij}} \geq \varepsilon. \\ 0, & \text{otherwise.} \end{cases}$$

This adjacency measure can work well if cluster densities are consistent. If there are clusters of varying densities then it can be difficult to choose a value of ε such that the matrix is reduced to a sparse matrix and a high level of information is retained.

The k-Nearest Neighbours (k-NN) is another simple adjacency measure that creates a sparse matrix using the relative similarity of data points rather than absolute similarity. Let $\boldsymbol{S_i'}$ be the kth most similar point to $\boldsymbol{S_{ij}}$, we

could define an adjacency measure as follows:

$$\boldsymbol{A_{ij}} = \begin{cases} 1, & \text{if } \boldsymbol{S_{ij}} \geq \boldsymbol{S_{i'}}. \\ 0, & \text{otherwise.} \end{cases}$$

This will not generate a symmetric matrix as node i may be a nearest neighbour of node j but this does not imply node j is a nearest neighbour of node i. Therefore, there are two common suggestions to overcome this. The first is known as k-Nearest Neighbour graph.

$$\boldsymbol{A_{ij}} = \boldsymbol{A_{ji}} = \begin{cases} 1, & \text{if } \boldsymbol{S_{ij}} \geq \boldsymbol{S_{i'}}. \\ 1, & \text{if } \boldsymbol{S_{ji}} \geq \boldsymbol{S_{j'}}. \\ 0, & \text{otherwise.} \end{cases}$$

The second is known as the mutual k-Nearest Neighbour Graph.

$$\boldsymbol{A_{ij}} = \boldsymbol{A_{ji}} = \begin{cases} 1, & \text{if } \boldsymbol{S_{ij}} \geq \boldsymbol{S_{i'}} \text{ and } \boldsymbol{S_{ji}} \geq \boldsymbol{S_{j'}}. \\ 0, & \text{otherwise.} \end{cases}$$

It is clear that the mutual k-NN measure will produce a sparser matrix, though both act as a similar adjacency measure. Once again, this choice is dependent on form of the similarity matrix and general advice cannot be given. However, for smaller datasets there may be little to be gained from the optimisation using sparse matrices and relatively a lot of information may be lost in the process. It is also possible to implement a weighted version of both of these adjacency measures where $\boldsymbol{A_{ij}}$ is set equal to $\boldsymbol{S_{ij}}$ instead of 1.

2.3.4 Choosing the graph Laplacian

There are many versions of the graph Laplacian, we will consider 3 variations to give an example of the considerations required. The Laplacian, as mentioned earlier, has very interesting properties which allow for Spectral Clustering to perform as desired. The Laplacian is constructed from the adjacency matrix. First, we construct a diagonal matrix, $\boldsymbol{D}$, containing the degree of each vertex.

$$\boldsymbol{D_{ii}} = \sum_{j=1}^{N} \boldsymbol{A_{ij}}$$

The best-known Laplacian is the un-normalized Laplacian, given by:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$$

For now, it is necessary to know for the space spanned by the eigenvectors to be stable, it requires a relatively large eigengap. This is the gap between the Kth and $K+1$th first eigenvalues. A discussion on the stability of the space spanned by the eigenvectors can be seen in [4]. However, to examine this eigengap it may be desirable to have normalised eigenvalues. This leads to the normalised Laplacian.

$$\boldsymbol{L}_{norm} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$$

And a slight variation as proposed by Ng. et al [4]

$$\boldsymbol{L}_{NJW} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$$

It should be noted that subbing $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ into $\boldsymbol{L}_{norm}$ gives that

$$\boldsymbol{L}_{norm} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} = \boldsymbol{I} - \boldsymbol{L}_{NJW}$$

where $\boldsymbol{I}$ is the identity matrix

Both matrices produce normalised eigenvalues however the eigenvalues of interest do change. Given the eigenvalues $\lambda_0, \ldots, \lambda_N$ of $\boldsymbol{L}_{norm}$, the eigenvalues of $\boldsymbol{L}_{NJW}$ are equal to $1-\lambda_0, \ldots, 1-\lambda_N$. This means the first K eigenvalues for $\boldsymbol{L}_{norm}$ are the K smallest eigenvalues, while the first K eigenvalues for $\boldsymbol{L}_{NJW}$ now refers to the K largest eigenvalues. This change in Laplacian was only proposed to allow for easier proof of stability of the eigenspace [4]. Therefore, both can be used interchangeably provided a change is made to the eigenvalues that are chosen.

The difference between the unnormalized and normalized Laplacian is more complex than just the production of normalised eigenvalues. While the normalised eigenvalues pose some value from a practical implementation, the Laplacian graphs can perform very differently. The unnormalized and normalised Laplacian can be viewed as a relaxation of the Ratio and Normalised cut problems respectively. To choose a Laplacian we must first consider the degree of the vertices in the graph. The more similar the degree of the vertices are, the less important the choice of Laplacian is. However, if the degree

of the vertices varies the normalised Laplacian tends to be preferred. This is due to the normalised cut problem incorporating the maximisation of within cluster similarity as well as minimisation of between cluster similarity. The ratio cut problem only considers the minimisation of between cluster similarity. The comparison of the Laplacian choice and the respective cut problems is explained and the implications are discussed in great detail in [3].

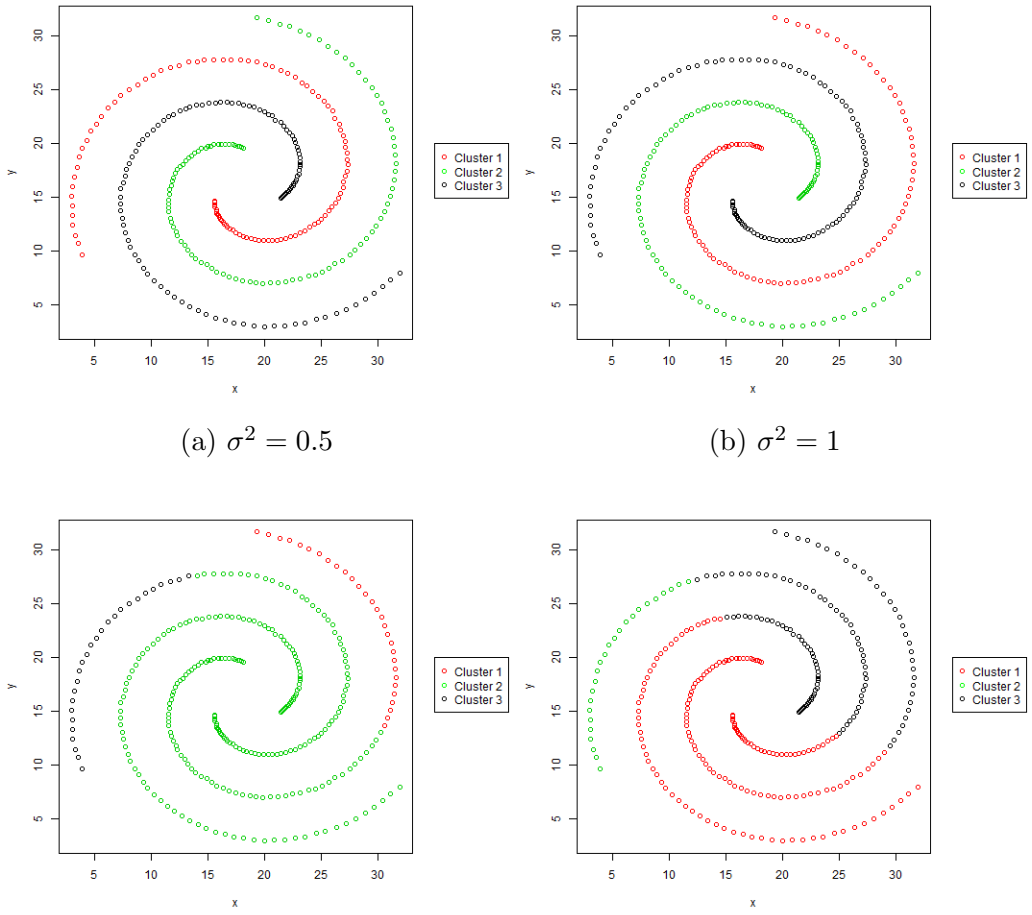
Now that we have examined some of the possible choices in the algorithm, we shall implement the algorithm on 3 different example datasets to show possible performance.

3 Results

3.1 Spirals Dataset

The first dataset we will examine is the spirals dataset seen in Figure 1. As mentioned earlier, K-means will preform poorly on the raw data due to the shape of the clusters. K-means will only detect circular clusters due to its dependence on the Euclidean distance. As the clusters are of similar density the Gaussian distance should be adequate as a similarity measure. The consistent density and separation means a constant scaling factor will perform well. A grid search of possible σ^2 values will be used to find an optimal value. As the data is in 2 dimensions and correct clustering can be easily identified visually, tuning can be carried out using a simple scatter plot colour by cluster membership as seen in Figure 4. Due to the size of data, N = 312, it is not necessary to implement an adjacency measure for computational reasons. It is also not necessary for density and separation reasons as the clusters in this example are consistent. The choice of Laplacian is also not of importance due to the regularity of the clusters, and therefore the similarity of the degree of each vertex. As the true number of clusters is known and the inspection of the eigengap is of lesser importance the unnormalised Laplacian is a suitable choice. Similar results would be seen when using either of the normalised Laplacian options, however a different optimal value of σ^2 may be found.

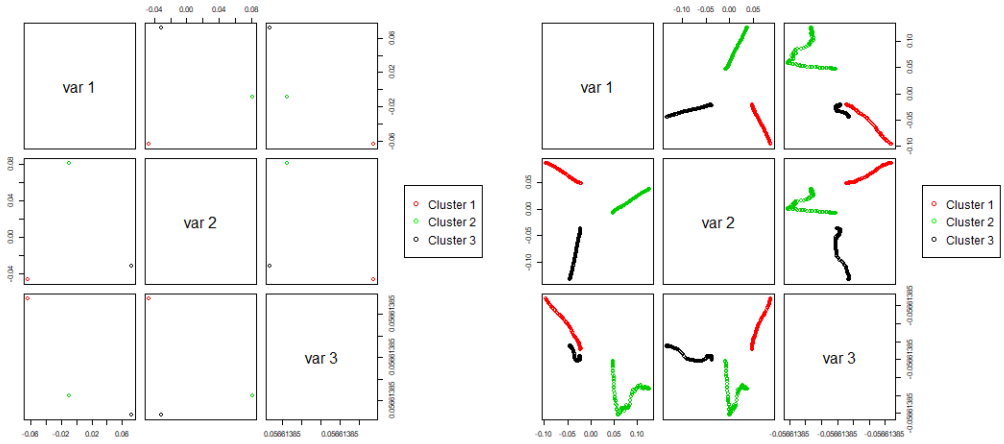
This example leads to two interesting questions. Our grid search will reveal that $\sigma^2 = 0.5$ and $\sigma^2 = 1$ both lead to the correct clustering. Therefore, is each clustering equal and how can we differentiate between them? To inspect the clustering in greater detail we look at a plot of eigenvectors. The Kernlab's implementation of Spectral Clustering, Specc [7], minimises the total within sum of squares of the K-means clustering find the optimal value of σ^2. This means it finds the most compact clusters as a measure of performance. In this case, we can see this occurs when $\sigma^2 = 0.5$



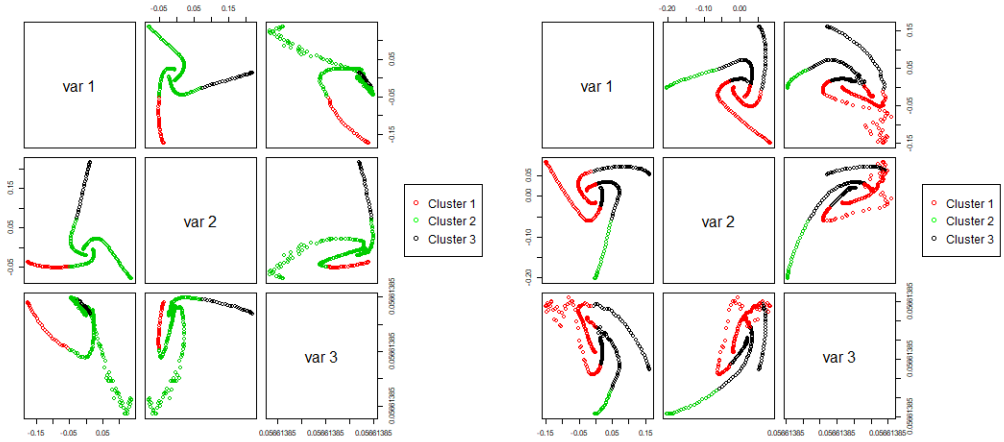
(a) $\sigma^2 = 0.5$ (b) $\sigma^2 = 1$

(c) $\sigma^2 = 1.5$ (d) $\sigma^2 = 2$

Figure 4: Spectral Clustering Results on Spiral Data with different values of σ^2



(a) $\sigma^2 = 0.5$ (b) $\sigma^2 = 1$



(c) $\sigma^2 = 1.5$ (d) $\sigma^2 = 2$

Figure 5: Eigenvector plots used in Spectral Clustering on Spiral Data with different values of σ^2

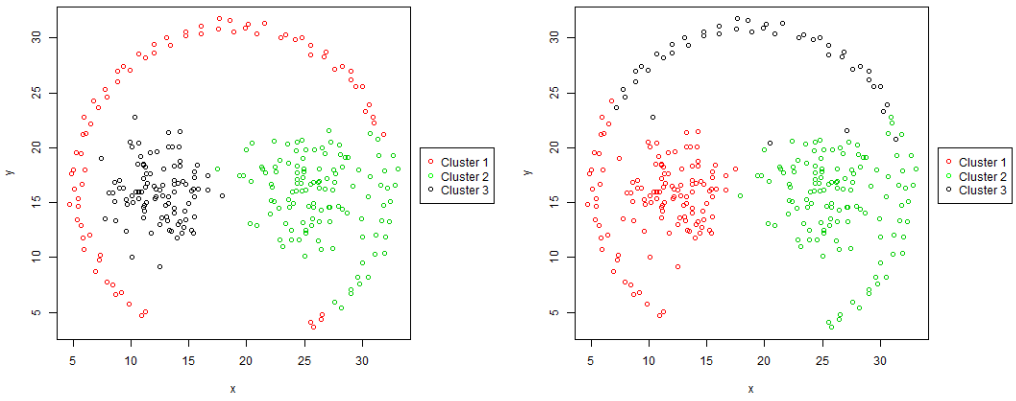
3.2 Pathbased Dataset

The second dataset is a little trickier to get correct results on. I was unable to find a combination of parameters which gave the fully correct clustering. The clustering I found mis-clustered 32 points out of 300. However, I will outline my reasoning for similarity and adjacency choices and the process of tuning the parameters. First, it should be noted that separability is low between cluster B and C as is seen in Figure 2. It should also be noted that while the middle two clusters, A and B, are of similar density and shape, the third cluster, C, is very different. To deal with this difference in density I chose the local scale similarity matrix. As explained earlier, this has a variable scale factor meaning the similarities should scale more appropriately across the clusters. To counteract the low separability, I decided to implement an unweighted k-NN adjacency measure. I chose to use unweighted edges which should highlight the most important cluster connections only. Using unweighted edges has an effect on the Laplacian choice and its relation to the respective cut problem. It means that the volume of our adjacency graph will be more similar to the cardinality and reduces the importance of the choice. It is also clear that the degree of each vertex will be constant and equal to the number of neighbours used in our k-NN. This will mean that the normalised and un-normalised Laplacian will perform similarly [3]. I decided to use the normalised Laplacian for the added benefit of normalised eigenvalues if I needed to compare the eigengap between clusterings.

This example required two parameters to be tuned. Which neighbour to be taken for local scale and the number of neighbours to be considered in k-NN adjacency measure. A grid search can be an effective but computationally intensive measure for parameter tuning. It also requires a measure of performance and while we used visually inspection initially in the previous example this option is not available here. I decided to use the adjusted rand index as a measure of performance but it should be noted this required the correct clustering to be known. This measure of performance is invariant to cluster permutations which is necessary due to the random nature of cluster name assignment in K-means. The grid search returned a adjusted rand index of X. This was achieved using the tenth neighbour in local scale and 110 neighbours in k-NN. These are both surprisingly high figures and shows the unpredictability of Spectral Clustering. The value of 110 neighbours in k-NN

means we do not get the added benefit of a spare matrix. I had implemented k-NN to counteract low separability and this was somewhat unsuccessful seen in Figure 6(a). However, it does out-perform Spectral Clustering with no adjacency measure.

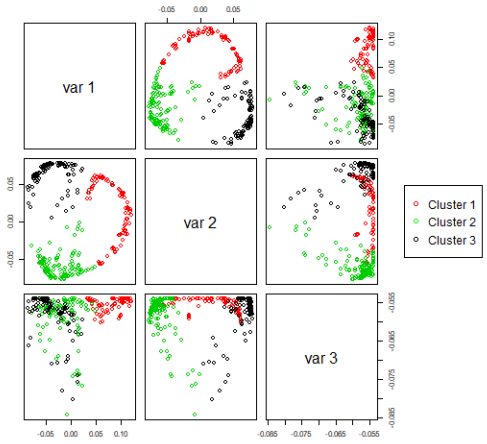
It a more realistic example the correct clustering would not have been known. This would mean that the adjusted rand index could not be used as a measure of performance. As mentioned in the previous example, Kernlab's Specc function minimises the total within sum of squares, TWSS. The result of using this as a measure of performance can been seen in Figure 6(b). This produced more compact clusters but does not perform as well as the parameters tuned with the adjusted rand index.



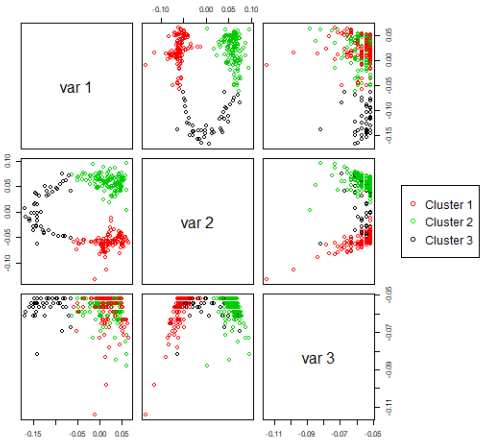
(a) Tuned with Adj. Rand Index
10th Neighbour for Local scale
110 Neighbours for k-NN

(b) Tuned with TWSS
3rd Neighbour for Local scale
10 Neighbours for k-NN

Figure 6: Comparison of Spectral Clustering Results on Pathbased data with different measures of performance



(a) Tuned with Adj. Rand Index
10th Neighbour for Local scale
110 Neighbours for k-NN

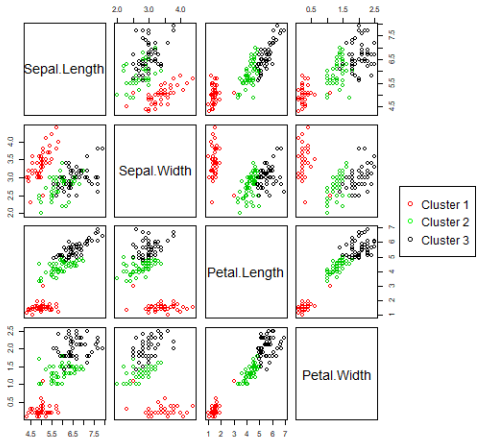


(b) Tuned with TWSS
3rd Neighbour for Local scale
10 Neighbours for k-NN

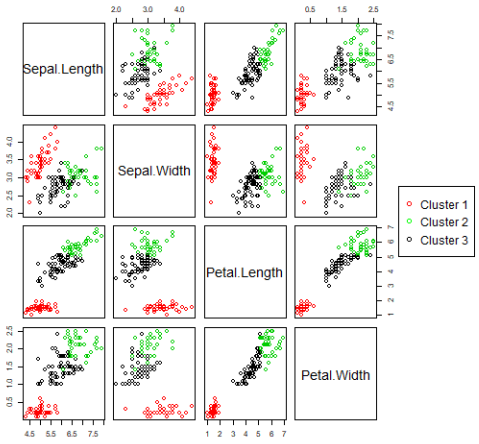
Figure 7: Comparison of Eigenvectors used in Spectral Clustering on Path-based data with different measures of performance

3.3 Iris Dataset

The final dataset is the Iris dataset seen in Figure 3. This data has a regular cluster shape but very low separability between the Versicolor and Virginica clusters. Similar to before, I will use local scale as the similarity measure due to its variable scaling factor. Due to the poor separation, I will use k-NN as my adjacency measure but with weighted edges. My Laplacian of choice will the NJW Laplacian as the degree of the vertices should vary due to the weighted k-NN adjacency. This will perform the same as the normalised Laplacian but the first 3 eigenvalues will now refer to the 3 largest rather than 3 smallest. As before, I will utilise a grid search to tune the two parameters. The best results using the adjusted rand index as a measure of performance was achieved with the third neighbour for local scale and 8 neighbours for k-NN. This mis-clustered 6 out of 150 points. When I repeated the grid search aiming to minimise the total sum of squares 14 points out the 150 are mis-clustered. It is also worth noting that the eigenvectors for this clustering appear to be better separated and this serves as an example that higher better cluster separation does not necessarily indicate more correct cluster assignment.

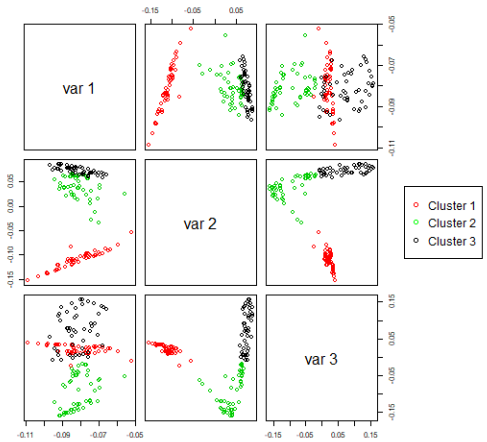


(a) Tuned with Adj. Rand Index
12th Neighbour for Local scale
12 Neighbours for k-NN

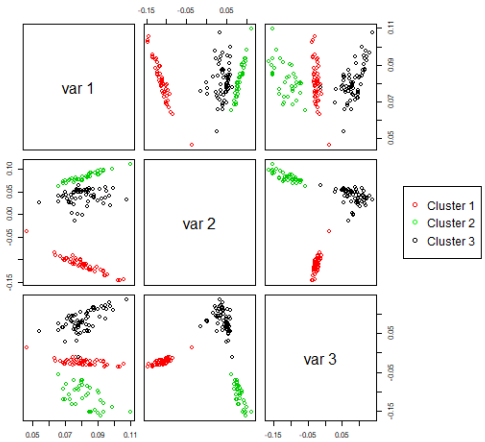


(b) Tuned with TWSS
5th Neighbour for Local scale
11 Neighbours for k-NN

Figure 8: Comparison of Spectral Clustering Results on Iris data with different measures of performance



(a) Tuned with Adj. Rand Index
12th Neighbour for Local scale
12 Neighbours for k-NN



(b) Tuned with TWSS
5th Neighbour for Local scale
11 Neighbours for k-NN

Figure 9: Comparison of Eigenvectors used in Spectral Clustering on Iris data with different measures of performance

4 Discussion

Spectral Clustering can be a very powerful non-parametric clustering technique. It is clear from the three examples above that Spectral Clustering can produce useful and correct clustering results where other algorithms can fail. The accuracy relies heavily on the quality of the choices made throughout. In our examples, we utilised a limited number of options for each decision and there are many valid similarity and adjacency measures not mentioned in this paper. These may have produced better results on our examples with the correct parameters. It is difficult to give general advice on choice of similarity and adjacency measures. Despite using a good understanding of each measure and their potential benefits throughout the process, there were some surprising results found during tuning. For example, in the Pathbased dataset, the best results were produced using the tenth neighbour for local scale and 110 neighbours for k-NN. This value of 110 is much larger than would have originally predicted and this clustering result would likely not have been found without utilising a grid search. I believe this example perfectly captures the problem with the current Spectral Clustering algorithm. Spectral Clustering is missing a key component before it can become a useful technique for more realistic clustering situations. That key component is a measure of cluster accuracy or a reliable optimisation problem for tuning parameters. The sum of squares approach as suggested by the Kernlab's Specc clearly under preformed when compared to the results generated by the adjusted rand index grid search. The example generated on the Iris data shows that better cluster separability does not necessarily guarantee correct clustering results. The eigenvectors are better separated in the example utilising total within sum of squares but 14 points are mis-clustered vs 6 points mis-clustered using the adjusted rand index.

Another way of viewing this problem that was not addressed during the examples in this paper, is rounding. The true number of clusters were known in each example, so we attempted to produce the best clustering based on this information. Rounding is the process of choosing the correct number of clusters based on your current clustering parameters. To choose the optimal number of clusters, K, based on our Laplacian we choose the number of clusters that maximises the eigengap. The eigengap as mentioned earlier is the absolute value of the difference between the Kth and $K+1$th eigenvalue.

Doing so maximises the stability that the K first eigenvectors span. This potentially provides another use case for Spectral Clustering and there has been some work to automate the choice of number of clusters [8].

However, this does not overcome the fact that despite Spectral Clustering being a powerful algorithm there is currently no comprehensive method of assessing the clusters it produces. The number of clusters found, and their composition is highly dependent on the choices made. This would be less of an issue if it were possible to have a stronger indication of which similarity, adjacency, and Laplacian options should be chosen. Therefore in the future, I believe work should be done in solidifying the literature around these decisions. This will help increase the confidence in the clusters found and therefore the usefulness of the Spectral Clustering algorithm. Until such a time where these decisions are more defined or there is a valid method of parameter tuning, Spectral Clustering has a limited number of use cases.

References

[1] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. Springer Publishing Company, Incorporated, 1st ed., 2009.

[2] W. Chen, Y. Song, H. Bai, C. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 568–586, March 2011.

[3] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395–416, Dec 2007.

[4] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14* (T. G. Dietterich, S. Becker, and Z. Ghahramani, eds.), pp. 849–856, MIT Press, 2002.

[5] K. Fujiwara, M. Kano, and S. Hasebe, "Development of correlation-based clustering method and its application to software sensing," *Chemometrics and Intelligent Laboratory Systems*, vol. 101, no. 2, pp. 130–138, 2010.

[6] X. Degang, Z. Panlei, G. Weihua, Y. Chunhua, and X. Yongfang, "Research on spectral clustering algorithms based on building different affinity matrix," in *2013 25th Chinese Control and Decision Conference (CCDC)*, pp. 3160–3165, May 2013.

[7] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis, "kernlab – an S4 package for kernel methods in R," *Journal of Statistical Software*, vol. 11, no. 9, pp. 1–20, 2004.

[8] L. Zelnik-manor and P. Perona, "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems 17* (L. K. Saul, Y. Weiss, and L. Bottou, eds.), pp. 1601–1608, MIT Press, 2005.

[9] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 191 – 203, 2008.

A Data

Spirals Dataset [9]: http://cs.joensuu.fi/sipu/datasets/spiral.txt
Pathbased Dataset [9]: http://cs.joensuu.fi/sipu/datasets/pathbased.txt

B Code

https://github.com/Thomas-keane/ResearchProject