1 07/01/2020

1.1 Gaussian Similarity Function

The Gaussian similarity formula for the exponentially decaying weights ω_{ij} is defined as:

$$\omega_{ij} = \exp\Big(-\frac{||x_i - x_j||^2}{2\sigma^2}\Big)$$

What is the role of the scaling factor σ in the fully connected similarity graph with exponentially decaying weights ω_{ij} for $i, j = 1, ..., n$ computed using the Gaussian similarity formula?

If the separation of two points i, j is smaller than σ then their connection will have a significant weight ω_{ij} as both points will reside in the same neighborhood. However, if the distance was bigger than σ, then our weight will have a very small value, meaning that σ will somehow define the extend of our neighborhood.

The higher (lower) the value of σ, the stronger (weaker) connective weights there will be within the network. Figure (1) represents an example of spherical data in a 2-dimensional space, three observations from different regions in the data are marked with an x, representing the center of the circle, the local neighborhood of the data is defined by a radius of $\sigma = 2$. Hence, each marked observation will have a strong (weak) connective weight with the rest of data points lying inside (outside) the circle delimited by σ.

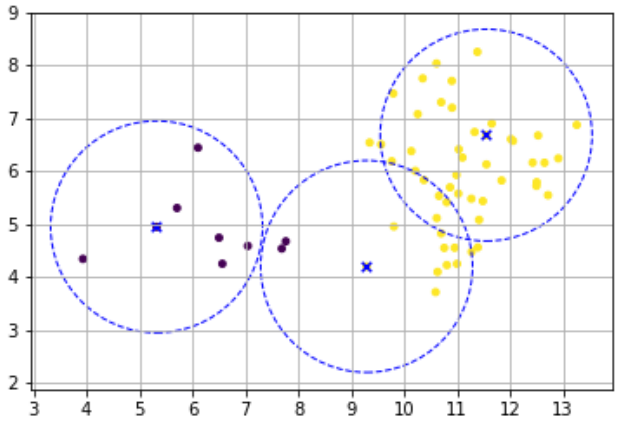


Figure 1: Spherical dataset with neighborhood extent of $\sigma = 2$

We want to choose a value of σ small enough that will connect only neighboring points. By too small we mean that if by one σ, a point is not able to reach it's nearest neighbor, that is a sign that σ is too small as the graph will be close to be fully disconnected. The choice of σ should be as small as possible, but still should be able to hold the network.

Furthermore, if σ is too big this will entail connections between inter-component points. Ideally we will like to get a natural boundary, that will allow us to easily separate the components in the graph.

A stopping criteria for choosing σ could be by selecting the value of σ that gives the biggest spectral gap. However, this method could be computationally expensive.

Another reasonable way of choosing σ could be by computing the distance from one data point to the rest of the network and selecting a value of σ that will be at least the median of the nearest neighbor distance multiplied by a factor of 2 or 3. We choose the median in order to avoid any outlier effect.

1.2 Gaussian Similarity Function: Imbalanced data

In general, **imbalanced data** refers to a classification problem in which the classes of a data-set are not equally distributed. For instance, if our data-set is composed by two clusters, it will be imbalanced if one of them has much more instances than the other. An example of imbalanced data could be a rare disease in a patient population.

On the other hand, when we are dealing with a data-set that has a different density from one region to another, this is known as **inhomogeneous data** and when the density of a data set is the same along all regions it is called **homogeneous data**. Hence, an imbalanced (balanced) data-set can be inhomogeneous or homogeneous.

In the case that we have an imbalanced and homogeneous data-set, we do not need to change/adapt the value of σ. However if we had an imbalanced and inhomogeneous data-set, we will need to adapt σ according to the density in each region.

1.3 Unnormalized graph Laplacian

The unnormalized graph Laplacian is defined as

$$L = D - W$$

The components of the graph Laplacian are:

- **Diagonalized degree matrix** D: n x n diagonal matrix. In the case of an unweighted graph the diagonal elements represent the number of connections to the rest of nodes in the network, and for a weighted graph each element will be the sum of connective weights per data points. Hence, each diagonal element will be:

$$D_{ii} = \sum_{j=1}^{n} \omega_{ij}$$

  This matrix will gives us an idea of how connected each data point is to the rest of the network.

- **Adjacency matrix** W: is the matrix representation of the graph, with dimension n x n. The elements of the matrix $W_{ij} = \omega_{ij}$ when $i \neq j$ and 0 otherwise.

*The graph Laplacian matrix is a **symmetric, positive semi-definite** matrix with precisely n real-valued, eigenvalues (if a matrix with real entries is symmetric then its eigenvalues are real) and n linearly independent non-zero eigenvectors.*

- **Symmetric** as $L^T = L$ and follows directly from the symmetry of D and W.

- **Positive semi-definite** A symmetric matrix is positive semi-definite if the quadratic form is at least

zero. The quadratic form of the Laplacian graph $L \in \mathbb{R}^{nxn}$ is:

$$
\begin{aligned}
f'Lf &= f'(D-W)f = f'Df - f'Wf \\
&= \begin{bmatrix} f_1, \ldots, f_n \end{bmatrix} \begin{bmatrix} \sum_{j=1}^{n} \omega_{1j} & & \\ & \ddots & \\ & & \sum_{j=1}^{n} \omega_{nj} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} - \begin{bmatrix} f_1, \ldots, f_n \end{bmatrix} \begin{bmatrix} 0 & & & & \\ \omega_{2,1} & 0 & & & \\ \omega_{3,1} & \omega_{3,2} & 0 & & \\ \vdots & \vdots & & \ddots & \\ \omega_{n,1} & \omega_{n,2} & \ldots & & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} \\
&= f_1^2 \sum_{j=1}^{n} \omega_{1j} + f_2^2 \sum_{j=1}^{n} \omega_{2j} + \ldots + f_n^2 \sum_{j=1}^{n} \omega_{nj} - \Big(f_1 \sum_{j=1}^{n} f_j \omega_{j1} + f_2 \sum_{j=1}^{n} f_j \omega_{j2} + \ldots + f_n \sum_{j=1}^{n} f_j \omega_{jn} \Big) \\
&= \sum_{i=1}^{n} f_i^2 \sum_{j=1}^{n} \omega_{ij} - \sum_{i=1}^{n} f_i \sum_{j=1}^{n} f_j \omega_{ji} = \frac{1}{2} \sum_{i,j=1}^{n} 2 f_i^2 \omega_{ij} - 2 f_i f_j \omega_{ji} = \frac{1}{2} \sum_{i,j=1}^{n} \omega_{ij} (f_i^2 + f_j^2 - 2 f_i f_j) \\
&= \frac{1}{2} \sum_{i,j=1}^{n} \omega_{ij} (f_i - f_j)^2 \geq 0
\end{aligned}
\tag{1}
$$

The Laplacian graph will be positive semi-definite for all vectors f, meaning that it will have non-negative eigenvalues. If f is an eigenvector of L, then we know that $f'Lf = f'\lambda f = f'f\lambda$, since $f'f$ and $f'Lf$ are non-negative values, then the eigenvalue λ must also be a non-negative value (this is not always true, it holds as the eigenvalues are real-valued).

Said that, we can notice that the smallest eigenvalue will be 0. The number of eigenvectors will correspond to the number of components in the graph. For instance, if we have a fully connected graph then the eigenvector of the eigenvalue 0, will be the constant ones-vector $\mathbb{1} =< 1, 1, ..., 1 >$, if we had k components we will have k non-trivial eigenvectors corresponding to the 0 eigenvalue, each eigenvector will indicate the connected nodes of the component, hence the eigenvectors will be indicator vectors.

If we take a look at (1), $f'Lf = \lambda = 0$ whenever $\omega_{ij}(f_i - f_j)^2 = 0 \; \forall i, j = 1, ..., n$. This condition will be satisfied if (i) $\omega_{ij} = 0$ and $f_i \neq f_j$, referring to self-connectivity cases or no connection between nodes, and (ii) the elements of the eigenvector $f_i = f_j > 0$, and $\omega_{ij} > 0$, which indicates that node i is connected to node j, and this is why the first (smallest) eigenvalue indicates the number of components of a graph. Furthermore k denotes the multiplicity of the eigenvector λ.

The unnormalized graph Laplacian matrix will look like:

$$
\begin{bmatrix} \sum_{j=1}^{n} \omega_{1j} & & & & \\ -\omega_{21} & \sum_{j=1}^{n} \omega_{2j} & & & \\ -\omega_{31} & -\omega_{32} & \sum_{j=1}^{n} \omega_{3j} & & \\ \vdots & \vdots & \vdots & \ddots & \\ -\omega_{n1} & -\omega_{n2} & \ldots & \sum_{j=1}^{n} \omega_{nj} \end{bmatrix} \quad L_{ij} = \begin{cases} D_{ii} & i = j \\ -\omega_{ij}, & i \neq j \end{cases}
$$

Notice that each row/column will sum to zero.

What does it mean by *unnormalized*?

If we re-scale the connectivity weights in our network, the eigenvectors will not be changed (the connectivity will remain the same), however the eigenvalues of the unnormalized Laplacian graph will be scaled by a constant. It is called *unnormalized* as the eigenvalue depends on ω_{ij}, we will need to fix a constant in order to normalize it.

Knowing that $f'Lf = f'f\lambda$. Eigenvectors can be normalize by requiring that $f'f = 1$, i.e we divide each element of the eigenvector f by $\sqrt{f_1^2 + f_2^2 + ... + f_n^2}$. By this procedure we can see that the eigenvector f does not depends on the weights, while the eigenvalue does, as it can be seen in (1), if we normalize the eigenvector f, $f'Lf = \lambda$, clearly λ depends on the value of ω_{ij}.

1.4 Normalized graph Laplacian

1.4.1 Symmetric Normalized graph Laplacian

$$\begin{aligned} L_{sym} &= D^{-1/2}LD^{-1/2} = D^{-1/2}(D-W)D^{-1/2} \\ &= D^{-1/2}DD^{-1/2} - D^{-1/2}WD^{-1/2} = I - \frac{1}{\sqrt{D}}W\frac{1}{\sqrt{D}} \end{aligned} \tag{2}$$

Where I denotes the identity matrix, L, the unnormalized Laplacian graph and W the adjacency matrix. The normalized symmetric Laplacian graph is an n x n positive semi-definite matrix. This property can be proved by computing the quadratic form such that $f'L_{sym}f \geq 0$ for all $f \in \mathbb{R}^n$

$$\begin{aligned} f'L_{sym}f &= f'If - f'\frac{1}{\sqrt{D}}W\frac{1}{\sqrt{D}}f \\ &= \begin{bmatrix} f_1, \dots, f_n \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} - \begin{bmatrix} \frac{f_1}{\sqrt{D_{11}}}, \dots, \frac{f_n}{\sqrt{D_{nn}}} \end{bmatrix} \begin{bmatrix} 0 & & & & \\ \omega_{21} & 0 & & & \\ \omega_{31} & \omega_{32} & 0 & & \\ \vdots & \vdots & & \ddots & \\ \omega_{n1} & \omega_{n2} & \dots & & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{D_{11}}} & & & \\ & \frac{1}{\sqrt{D_{22}}} & & \\ & & \ddots & \\ & & & \frac{1}{\sqrt{D_{nn}}} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} \\ &= \sum_{i=1}^{n} f_i^2 - \sum_{i=1}^{n} \frac{f_i}{\sqrt{D_{ii}}} \Big(\sum_{j=1}^{n} \frac{f_j}{\sqrt{D_{jj}}} \omega_{ji} \Big) = \sum_{i=1}^{n} f_i^2 \frac{\sum_{j=1}^{n} \omega_{ij}}{D_{ii}} - \sum_{i,j=1}^{n} f_i f_j \Big(\frac{\omega_{ij}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \Big) \\ &= \sum_{i,j=1}^{n} f_i^2 \frac{\omega_{ij}}{D_{ii}} - f_i f_j \Big(\frac{\omega_{ij}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \Big) = \frac{1}{2} \Big(\sum_{i,j=1}^{n} \omega_{ij} \Big(\frac{f_i^2}{D_{ii}} + \frac{f_j^2}{D_{jj}} - 2\frac{f_i f_j}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \Big) \Big) \\ &= \frac{1}{2} \sum_{i,j=1}^{n} \omega_{ij} \Big(\frac{f_i}{\sqrt{D_{ii}}} - \frac{f_j}{\sqrt{D_{jj}}} \Big)^2 \geq 0 \end{aligned} \tag{3}$$

The Normalized graph Laplacian will look like:

$$\begin{bmatrix} 1 - \omega_{11}/D_{11} & & & & \\ -\omega_{21}/\sqrt{D_{22}}\sqrt{D_{11}} & 1 - \omega_{22}/D_{22} & & & \\ -\omega_{31}/\sqrt{D_{33}}\sqrt{D_{11}} & -\omega_{32}/D_{33}\sqrt{D_{22}} & 1 - \omega_{33}/D_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ -\omega_{n1}/\sqrt{D_{nn}}\sqrt{D_{11}} & -\omega_{n2}/\sqrt{D_{nn}}\sqrt{D_{22}} & \dots & & 1 - \omega_{nn}/D_{nn} \end{bmatrix} L_{sym}^{ij} = \begin{cases} 1 & i = j \\ -\omega_{ij}/\sqrt{D_{ii}}\sqrt{D_{jj}}, & i \neq j \end{cases}$$

What does it mean by *normalized*?

If we now re-scale ω_{ij} by a constant factor, it will not have any effect on the Normalized graph Laplacian eigenvalues, given that the terms of the quadratic form are proportional to the elements of the diagonalized matrix (which depend on the weights). The constant factor multiplying ω_{ij} in (3) will cancel out with the scaling factor dividing the quadratic form terms.

1.4.2 Random walk normalized graph Laplacian

A discrete time random walk on a graph can be seen as a Markov chain where each node denotes an element of the state space. A random walk in a graph is a stochastic process which randomly jumps from one node to another. The Random walk normalized Laplacian matrix is a positive semi-definite non-symmetric matrix.

$$L_{rw} = D^{-1}L = D^{-1}(D - W) = I - D^{-1}W = I - P \tag{4}$$

The matrix $D^{-1}W$ is equivalent to the n x n transition matrix $P \in \mathbb{R}^{nxn}$ with elements given by $P_{ij} = P(j|i) = \frac{\omega_{ij}}{D_{ii}}$ and represents the conditional probability of transitioning to node j from node i in a time step. Notice that this probability is proportional to the connectivity weight between the two nodes, hence the stronger (weaker) the connection, the higher (lower) the transitioning probability.

By symmetry $\omega_{ij} = \omega_{ji}$, however, $D_{ii} \neq D_{jj}$. Hence, the transition probability matrix is not symmetric.

$$D^{-1}W = P = \begin{bmatrix} \omega_{11}/D_{11} & & & & \\ \omega_{21}/D_{22} & \omega_{22}/D_{22} & & & \\ \omega_{31}/D_{33} & \omega_{32}/_{33} & \omega_{33}/D_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \omega_{n1}/D_{nn} & \omega_{n2}/_{nn} & \dots & & \omega_{nn}/D_{nn} \end{bmatrix}, P_{ij} = \begin{cases} 0 & i = j \\ \omega_{ij}/D_{ii}, & i \neq j \end{cases}$$

P is a row stochastic matrix, meaning that $\sum_{j=1}^{n} w_{ij}/D_{ii} = 1$. The eigenspectrum of L_{rw} and P will differ by a constant factor of 1, meaning that λ is the eigenvalue of L_{rw} iff. $(1 - \lambda)$ is the eigenvalue of P, while their eigenvectors will be the same.

> λ is is an eigenvalue of L_{rw} with eigenvector v if and only if λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2}v$

$$\begin{aligned} L_{sym}\ w &= \lambda w \\ (I - D^{-1/2}WD^{-1/2})D^{1/2}v &= \lambda D^{1/2}v \\ (D^{1/2} - D^{-1/2}W)v &= \lambda D^{1/2}v \\ (I - \lambda)D^{1/2}v &= D^{-1/2}Wv \\ (I - \lambda)v &= D^{-1}Wv \\ (I - D^{-1}W)v &= \lambda v \\ L_{rw}\ v &= \lambda v \\ (D^{-1}L)v &= \lambda v \\ Lv &= \lambda Dv \end{aligned} \tag{5}$$

From the above we can also notice that the eigenvector v with the largest eigenvalue for P is the eigenvector v with the smallest eigenvalue λ for L_{rw}.

Generally it would be more convenient to work with a symmetric matrix such as L_{sym}, as it has nicer properties compared to L_{rw}. However, L_{sym} and L_{rw} offer two perspectives to the same problem.

> Why do we set $\omega_{ii} = 0$ as a condition?

The question is, why do we set the self-connectivity to zero and not any other number?. By convention we assign a value of zero to self-connectivity cases because when computing the weights we are only interested in knowing the pairwise similarity between points and not with itself.

If we were to assign a non-zero weight to self-connectivity cases we would be assigning a high connectivity weight to a case that does not contribute with relevant information in our similarity graph.

In terms of

- **The unnormalized Laplacian graph**, if we assign self-connectivity weights, it will not have any effect on the matrix diagonal as it will cancel out when subtracting the diagonal and adjacency matrix, meaning that the elements in the unnormalized Laplacian matrix will not change. In terms of the eigenspectrum, neither the eigenvectors nor eigenvalues will change.
- **Random walk normalized graph Laplacian** if we assign self-connectivity weights, the matrix elements will be changed. The transition probability matrix will be modified as we will now have non-zero self-transitioning probabilities, i.e the terms in the matrix $p_{ij} = w_{ij}/D_{ii} > 0$, for $i = j$.

  A side effect of assigning self-connectivity weights is that these weights will be higher than the rest of connectivity weights, meaning that the probability of transitioning to a different state will be lower. Under this scenario, both the eigenvalue and eigenvectors will change.
- **Symmetric normalized graph Laplacian** From (5) we know that the eigenvalues of L_{rw} and L_{sym} are the same if and only if their eigenvectors are related by a constant, i.e $w = D^{1/2}v$. Hence, if we include self-connectivity weights both normalized Laplacian matrices will be modified as well as their eigenvalues and eigenvectors.

There is a close relationship between the eigenspectrum of L_{sym} and L_{rw}, but there is not such a relationship between the eigenspectrum of the unnormalized graph Laplacian L and the normalized graph Laplacian L_{sym} and L_{rw}. This non-relationship also follows from the fact that the unnormalized graph Laplacian is invariant to self-transitioning weights, while the normalized Laplacian graphs are not invariant.

2 07/10/2020

We started by discussing about further reasons of why we set self-connectivity weights $\omega_{ij} = 0$ for $i = j$. Notice that as the unnormalized Laplacian graph is invariant to any transformation, and so setting self-connectivity weights to zero will only have an effect on the normalized Laplacian matrices, some further reasons can be:

- Setting the self-connectivity weights to zero can be computationally advantageous as it will make computations faster.
- Setting non-zero self-connectivity weights prevent exploring the manifold: For instance, if we have an isolated point with no neighbors in the radius defined by a Gaussian Kernel, the connective-weights to the rest of the nodes in the network will be very weak compared to the self-transition weight, which in terms of the self-transition conditional probability will dominate over the rest of transitioning probabilities i.e $p_{ij} \sim 1$ for $i = j$.

  The problem of this scenario is that although the probability of transitioning to this isolated node, could be very small or not, if we do transition, there is a high probability of staying trapped in the same isolated node for many time steps. Hence, if we want to study all possible paths of traveling from one node to another we will have to take into account the time the node will travel to itself before it jumps out, which will make the problem more complicated.

2.0.1 Graph Cut point of view

The spectral clustering problem can be derived as an approximation of a graph partitioning problem.

Let $A, \bar{A} \subset V$ where V is the set of all vertices $V = \{v_1, v_2, ..., v_n\}$ and $\bar{A}$ the complement of A, $\bar{A} = V/A$. We would like to find a partition $A_1, A_2, ..., A_k$ that minimizes the mincut problem:

$$\text{cut}(A_1, A_2, ..., A_k) = \sum_{i=1}^{k} \text{cut}(A_i \bar{A}_i) = \sum_{i=1}^{k} \sum_{j \in A_i, k \in \bar{A}_i} w_{jk} \tag{6}$$

A drawback when solving the mincut problem is that it is sensitive to outliers, meaning that if our graph contains isolated vertices, it will separate them from the rest of the data, resulting in a non-desirable partition, as one would expect clusters to have a reasonable large number of vertices.

RatioCut and Ncut are two objective functions that circumvent the mincut problem, they are defined as:

$$\begin{aligned} \text{RatioCut}(A_1, A_2, ..., A_k) &= \sum_{i=1}^{k} \frac{\text{cut}(A_i \bar{A}_i)}{|A_i|} \\ \text{Ncut}(A_1, A_2, ..., A_k) &= \sum_{i=1}^{k} \frac{\text{cut}(A_i \bar{A}_i)}{\text{vol}(A_i)} \end{aligned} \tag{7}$$

Where $|A_i|$ denotes the cardinality of the set A_i, while $\text{vol}(A_i)$ denotes the size of A_i by the sum of connectivity weights between the vertices of each ith component and the rest of the network, i.e $\text{vol}(A_i) = \sum_{i \in A} D_{ii} = \sum_{i \in A} \sum_{j=1}^{n} w_{ij}$. By scaling the mincut problem we now ensure components of not a small size, as the objective functions will have a small value as the size of A_i increases, i.e the components will be more "balanced".

By minimizing the objective functions, we look for weakly connection in between components and strongly connections within components. However, the complexity of solving this problem increases as the number of vertices and components increase. Hence, an alternative to this problem is by using spectral clustering to approximate a relaxed version of the RatioCut and Ncut objective functions.

2.0.2 Approximation of Ncut for k = 2

The goal is to solve the optimization problem:

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

We rewrite the problem in a more convenient form: given a set V divided in non-overlapping subsets A and $\bar{A}$, we define the indicator vector $f = (f_1, ..., f_n)' \in \mathbb{R}^n$ with entries:

$$f_i = \mathbb{1}_{\{i \in A\}} \sqrt{\frac{\text{vol}(\bar{A})}{\text{vol}(A)}} - \mathbb{1}_{\{i \in \bar{A}\}} \sqrt{\frac{\text{vol}(A)}{\text{vol}(\bar{A})}}$$

We can now rewrite the Ncut objective function in terms of the quadratic form of the unnormalized graph

Laplacian. From (1) we know that:

$$
\begin{aligned}
f'Lf &= \frac{1}{2}\sum_{i,j=1}^{n}\omega_{ij}(f_i - f_j)^2 \\
&= \frac{1}{2}\sum_{i\in A, j\in \bar{A}}^{n}\omega_{ij}\left(\sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} + \sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}}\right)^2 + \frac{1}{2}\sum_{i\in \bar{A}, j\in A}^{n}\omega_{ij}\left(-\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}} - \sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}}\right)^2 \\
&= \frac{1}{2}\sum_{i\in A, j\in \bar{A}}^{n}\omega_{ij}\left(\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)} + \frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})} + 2\right) + \frac{1}{2}\sum_{i\in \bar{A}, j\in A}^{n}\omega_{ij}\left(\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})} + \frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)} + 2\right) \\
&= \frac{1}{2}\sum_{i\in A, j\in \bar{A}}^{n}\omega_{ij}\left(\frac{\mathrm{vol}(\bar{A}) + \mathrm{vol}(A)}{\mathrm{vol}(A)} + \frac{\mathrm{vol}(A) + \mathrm{vol}(\bar{A})}{\mathrm{vol}(\bar{A})}\right) + \frac{1}{2}\sum_{i\in \bar{A}, j\in A}^{n}\omega_{ij}\left(\frac{\mathrm{vol}(A) + \mathrm{vol}(\bar{A})}{\mathrm{vol}(\bar{A})} + \frac{\mathrm{vol}(\bar{A}) + \mathrm{vol}(A)}{\mathrm{vol}(A)}\right) \\
&= \frac{1}{2}\sum_{i\in A, j\in \bar{A}}^{n}\omega_{ij}\mathrm{vol}(V)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(\bar{A})}\right) + \frac{1}{2}\sum_{i\in \bar{A}, j\in A}^{n}\omega_{ij}\mathrm{vol}(V)\left(\frac{1}{\mathrm{vol}(\bar{A})} + \frac{1}{\mathrm{vol}(A)}\right) \\
&= \frac{1}{2}\mathrm{vol}(V)\cdot\mathrm{cut}(A,\bar{A})\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(\bar{A})}\right) + \frac{1}{2}\mathrm{vol}(V)\cdot\mathrm{cut}(\bar{A},A)\left(\frac{1}{\mathrm{vol}(\bar{A})} + \frac{1}{\mathrm{vol}(A)}\right) \\
&= \mathrm{vol}(V)\ \mathrm{cut}(A,\bar{A})\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(\bar{A})}\right) = \mathrm{vol}(V)\cdot\mathrm{Ncut}(A,\bar{A})
\end{aligned}
\tag{8}
$$

Where the vector Df is orthogonal to the constant vector $\mathbb{1}$, and so their dot product is zero:

$$
\begin{aligned}
\langle Df, \mathbb{1}\rangle &= \sum_{i=1}^{n} D_{ii}f_i \\
&= \sum_{i\in A} D_{ii}\sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} - \sum_{i\in \bar{A}} D_{ii}\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}} \\
&= \mathrm{vol}(A)\cdot\sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} - \mathrm{vol}(\bar{A})\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}} = 0
\end{aligned}
\tag{9}
$$

$$
\begin{aligned}
fDf' &= \sum_{i=1}^{n} D_{ii}f_i^2 \\
&= \sum_{i\in A} D_{ii}\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)} - \sum_{i\in \bar{A}} D_{ii}\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})} \\
&= \mathrm{vol}(A)\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)} + \mathrm{vol}(\bar{A}) + \frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})} = \mathrm{vol}(V)
\end{aligned}
\tag{10}
$$

We can then re-write the minimization problem as:

$$
\begin{aligned}
\min_{A} \quad & f'Lf && \text{(11a)}\\
\text{subject to} \quad & Df \perp \mathbb{1}, && \text{(11b)}\\
& f_i = \mathbb{1}_{\{i\in A\}}\sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} - \mathbb{1}_{\{i\in \bar{A}\}}\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}}, && \text{(11c)}\\
& fDf' = \mathrm{vol}(V). && \text{(11d)}
\end{aligned}
$$

We relax the problem to:

$$\begin{aligned} \min_{f \in \mathbb{R}^n} \quad & f'Lf && \text{(12a)} \\ \text{subject to} \quad & Df \perp \mathbb{1}, && \text{(12b)} \\ & fDf' = \text{vol}(V). && \text{(12c)} \end{aligned}$$

We substitute $f = gD^{-1/2}$

$$\begin{aligned} g'D^{-1/2}LD^{-1/2}g = g'L_{sym}\ g = \frac{1}{2}\sum_{i,j=1}^{n} \omega_{ij}\Big(g_i - g_j\Big)^2 \geq 0 \\ \lambda_1^{sym} = 0 \\ g_1^{sym} = D^{1/2}\mathbb{1} \end{aligned} \tag{13}$$

$$\begin{aligned} \langle g, D^{1/2}\mathbb{1}\rangle = \sum_{i=1}^{n} D_{ii}f_i = 0 \\ ||g||^2 = \langle g, g\rangle = \sum_{i=1}^{n} g_i^2 = \sum_{i\in A}\left(\frac{D_{ii}}{\sqrt{D_{ii}}}\sqrt{\frac{\text{vol}(\bar{A})}{\text{vol}(A)}}\right)^2 + \sum_{i\in A}\left(-\frac{D_{ii}}{\sqrt{D_{ii}}}\sqrt{\frac{\text{vol}(A)}{\text{vol}(\bar{A})}}\right)^2 = \text{vol}(V) \end{aligned} \tag{14}$$

Let $G = \{g_1, .., g_n\}$ be a set of eigenvectors of L_{sym}, then we know that G is orthogonal iff.

$$(\forall i, g_i \neq 0) \cap (\forall i \neq j, \langle g_i, g_j\rangle = 0)$$

We know that for a symmetric matrix L, if g_1 and g_2 are eigenvectors of L with different eigenvalues λ_1, λ_2,

$$\lambda_1\langle g_1, g_2\rangle = \langle \lambda_1 g_1, g_2\rangle = \langle Lg_1, g_2\rangle = \langle g_1, L'g_2\rangle = \langle g_1, Lg_2\rangle = \langle g_1, \lambda_2 g_2\rangle = \lambda_2\langle g_1, g_2\rangle$$

$$\Rightarrow (\lambda_1 - \lambda_2)\langle g_1, g_2\rangle = 0$$

Then g_1 and g_2 are orthogonal $\langle g_1, g_2\rangle = 0$, as $(\lambda_1 - \lambda_2) \neq 0$. However, if the eigenvectors correspond to the same eigenvalue need not be orthogonal.

Furthermore, G is orthonormal iff G is orthonormal and $\forall i, \langle g_i, g_i\rangle = 1$, which will mean that all vectors have the same length.

We can rewrite the minimization problem as:

$$\begin{aligned} \min_{g \in \mathbb{R}^n} \quad & g'L_{sym}g && \text{(15a)} \\ \text{subject to} \quad & g \perp D^{1/2}\mathbb{1}, && \text{(15b)} \\ & ||g||^2 = \text{vol}(V). && \text{(15c)} \end{aligned}$$

The solution to the minimization problem subject to the constraints will be the second eigenvector g_2.

$$\begin{gathered} L_{sym}\ g_2 = \lambda_2 g_2 \\ L_{sym}\ f_2\ D^{1/2} = \lambda_2\ f_2\ D^{1/2} \\ L_{rw}\ f_2 = \lambda_2 f_2 \end{gathered}$$

2.0.3 Random walk point of view: Ncut via transition probabilities

Spectral clustering can be reformulated as a graph partitioning problem, where we will like to find a partition of the data such that the random walk stays long within the same data component.

Let $G = (V, E)$ be a connected and not bipartite graph (a graph where its vertices can be divided into two-disjoint and independent sets). Then:

$$\text{Ncut}(A, \bar{A}) = P(\bar{A}|A) + P(A|\bar{A})$$

Let the process $\{X_t, t \geq 0\}$ be a random walk, starting at X_0 with an initial state probability distribution $\boldsymbol{\pi} = (\pi_1, ..., \pi_n)'$ where

$$\pi_i = \frac{\sum_{j=1}^{n} \omega_{ij}}{\sum_{i \in V} D_{ii}} = \frac{D_{ii}}{\text{vol(V)}}$$

The joint probability is derived as:

$$\begin{aligned} P(X_0 \in A, X_1 \in \bar{A}) &= \sum_{i \in A, j \in \bar{A}} P(X_0 = i, X_1 = j) \\ &= \sum_{i \in A, j \in \bar{A}} P(X_0 = i) P(X_1 = j | X_0 = i) \\ &= \sum_{i \in A, j \in \bar{A}} \pi_i P_{ij} = \sum_{i \in A, j \in \bar{A}} \frac{d_i}{\text{vol(V)}} \frac{\omega_{ij}}{D_{ii}} = \frac{1}{\text{vol(V)}} \sum_{i \in A, j \in \bar{A}} \omega_{ij} \\ P(X_0 \in A) &= \sum_{i \in A} P(X_0 = i) = \frac{1}{\text{vol}(V)} \sum_{i \in A} D_{ii} = \frac{\text{vol}(A)}{\text{vol}(V)} \end{aligned} \tag{16}$$

The conditional probability is derived as:

$$P(X_1 \in \bar{A} | X_0 \in A) = \frac{P(X_0 \in A, X_1 \in \bar{A})}{P(X_0 \in A)} = \frac{\sum_{i \in A, j \in \bar{A}} \omega_{ij}}{\text{vol}(A)} = \frac{\text{cut}(A, \bar{A})}{\text{vol}(A)} \tag{17}$$

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

2.0.4 The commute time distance CTD

A connection between the random walk and the graph Laplacian can be made via the commute time distance in a graph. The CTD is defined as the expected time it takes a random walk to travel from vertex v_i to vertex v_j and back.

The CTD can be written in terms of the eigenspectrum and eigenvalue of either the unnormalized and normalized Laplacian graph.

- In terms of the **unnormalized graph Laplacian:**

$$c_{ij} = \text{vol}(V) \sum_{k=2}^{n} \frac{1}{\lambda_k} \Big(v_{kj} - v_{ki} \Big)^2$$

- In terms of the **normalized graph Laplacian:**

$$\begin{aligned} c_{ij} &= H(i,j) + H(j,i) \\ &= \mathrm{vol}(V)\sum_{k=2}^{n}\frac{1}{\lambda_k}\Big(\frac{v_{kj}^2}{d_j} - \frac{v_{ki}v_{kj}}{\sqrt{d_i D_{jj}}}\Big) + \mathrm{vol}(V)\sum_{k=2}^{n}\frac{1}{\lambda_k}\Big(\frac{v_{ki}^2}{D_{ii}} - \frac{v_{kj}v_{ki}}{\sqrt{D_{jj}D_{ii}}}\Big) \\ &= \mathrm{vol}(V)\sum_{k=2}^{n}\frac{1}{\lambda_k}\Big(\frac{v_{kj}^2}{D_{jj}} + \frac{v_{ki}^2}{D_{ii}} - 2\frac{v_{ki}v_{kj}}{\sqrt{D_{ii}D_{jj}}}\Big) \\ &= \mathrm{vol}(V)\sum_{k=2}^{n}\frac{1}{\lambda_k}\Big(\frac{v_{kj}}{\sqrt{D_{jj}}} - \frac{v_{ki}}{\sqrt{D_{ii}}}\Big)^2 \end{aligned} \tag{18}$$

Notice that (i) the CTD from vertex to vertex takes into account all eigenvectors of the graphs Laplacian, (ii) the scaling factor $1/\lambda_k$ makes the difference of elements in the eigenvectors corresponding to the smallest eigenvalues, to have more relevance, (iii) From the $\sqrt{c_{ij}}$ can be considered as a Euclidean distance function.

The eigendecomposition of the Laplacian matrix is:

$$\mathbf{Lv} = \lambda\mathbf{v} \Rightarrow \mathbf{LV} = \mathbf{V\Lambda} \Rightarrow \mathbf{L} = \mathbf{V}\lambda\mathbf{V}'$$

, where V is the $n \times n$ matrix containing n eigenvectors column-wise, while Λ is a diagonal matrix whose diagonal elements are the corresponding eigenvalues $\Lambda_{ii} = \lambda_i$ for $i = 1, ..., n$. We define the generalized inverse matrix $L^\dagger$ as:

$$L^\dagger = V\Lambda^\dagger V = \begin{bmatrix} \sum_{k=2}^{n}\frac{1}{\lambda_k}v_{1k}^2 & & & & \\ \sum_{k=2}^{n}\frac{1}{\lambda_k}v_{2k}v_{1k} & \sum_{k=2}^{n}\frac{1}{\lambda_k}v_{2k}^2 & & & \\ \sum_{k=2}^{n}\frac{1}{\lambda_k}v_{3k}v_{1k} & \sum_{k=2}^{n}\frac{1}{\lambda_k}v_{3k}v_{2k} & \sum_{k=2}^{n}\frac{1}{\lambda_k}v_{3k}^2 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \sum_{k=2}^{n}\frac{1}{\lambda_k}v_{nk}v_{1k} & \sum_{k=2}^{n}\frac{1}{\lambda_k}v_{nk}v_{2k} & \cdots & & \sum_{k=2}^{n}\frac{1}{\lambda_k}v_{nk}^2 \end{bmatrix},$$

Where $\Lambda^\dagger$ is the diagonal matrix with entries $1/\lambda_i$ if $\lambda_i \neq 0$ and 0 otherwise. From $L^\dagger$ we would like to find a function f such that $f(v_i) = z_i, \forall_{i\in 1,...,n}$ and $||z_i - z_j||^2 = c_{ij}$.

$$\begin{aligned} L^\dagger &= V(\Lambda^\dagger)^{1/2} \cdot ((\Lambda^\dagger)^{1/2}V)' \\ &= (\Lambda^\dagger)^{1/2}V\ V'(\Lambda^\dagger)^{1/2} \\ &= \Lambda^\dagger I = \Lambda^\dagger \end{aligned} \tag{19}$$

The i:th row of the matrix $(\Lambda^\dagger)^{1/2}V$ is $\mathbf{z_i'} = 1/\sqrt{\lambda_i} \cdot (v_{i1}, v_{i2}, \ldots, v_{in})$. Knowing that $V'V = I$ then $\langle z_i, z_i\rangle = 1/\lambda_i$, and $\langle z_i, z_j\rangle = 0$, for $i \neq j$. We can then write

$$V(\Lambda^{\dagger})^{1/2}\cdot((\Lambda^{\dagger})^{1/2}V)' = \begin{bmatrix}\mathbf{z_1'}\\ \mathbf{z_2'}\\ \vdots\\ \vdots\\ \mathbf{z_n'}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{z_1},\mathbf{z_2},\ldots,\ldots,\mathbf{z_n}\end{bmatrix} = \begin{bmatrix}\langle\mathbf{z_1},\mathbf{z_1}\rangle & \langle\mathbf{z_1},\mathbf{z_2}\rangle & \ldots & \langle\mathbf{z_1},\mathbf{z_n}\rangle\\ \langle\mathbf{z_2},\mathbf{z_1}\rangle & \langle\mathbf{z_2},\mathbf{z_2}\rangle & \ldots & \langle\mathbf{z_2},\mathbf{z_n}\rangle\\ \vdots & \vdots & \ddots & \vdots\\ \langle\mathbf{z_n},\mathbf{z_1}\rangle & \langle\mathbf{z_n},\mathbf{z_2}\rangle & \ldots & \langle\mathbf{z_n},\mathbf{z_n}\rangle\end{bmatrix} = \begin{bmatrix}1/\lambda_1 & 0 & \ldots & 0\\ 0 & 1/\lambda_2 & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & 1/\lambda_n\end{bmatrix} \tag{20}$$

Furthermore, the i:th column of the matrix V is $\mathbf{v_i'}$ and so

$$v_i'\ V = (\langle v_i', \mathbf{v_1}\rangle, \ldots, \langle v_i', \mathbf{v_i}\rangle, \ldots, \langle v_i', \mathbf{v_n}\rangle) = (0, \ldots, 1, \ldots, 0)$$

Which yields to $v_i' V\Lambda^{\dagger} = (0, \ldots, 1/\lambda_i, \ldots, 0)$, and:

$$(v_j\ V)' = \begin{bmatrix}\langle v_j, \mathbf{v_1}\rangle\\ \vdots\\ \langle v_j, \mathbf{v_j}\rangle\\ \vdots\\ \langle v_j, \mathbf{v_n}\rangle\end{bmatrix} = \begin{bmatrix}0\\ \vdots\\ 1\\ \vdots\\ 0\end{bmatrix}$$

$$\begin{aligned} v_i' L^{\dagger} v_j &= v_i' V\Lambda^{\dagger} V' v_j\\ &= v_i' V\Lambda^{\dagger}(v_j V)'\\ &= \langle z_i, z_j\rangle \end{aligned} \tag{21}$$

$$\begin{aligned} ||z_i - z_j||^2 &= \langle z_i - z_j, z_i - z_j\rangle\\ &= \langle z_i, z_i\rangle + \langle z_j, z_j\rangle - 2\langle z_i, z_j\rangle\\ &= v_i' L^{\dagger} v_i + v_j' L^{\dagger} v_j - 2 v_i' L^{\dagger} v_j\\ &= (v_i - v_j)' L^{\dagger}(v_i - v_j) = \frac{c_{ij}}{\mathrm{vol}(V)} \end{aligned} \tag{22}$$

Notice that the Euclidean norm between points $z_i \in \mathbb{R}^n$ coincides with c_{ij}. The commute time embedding maps the nodes from our graph to the rows z_i of our matrix $(\Lambda^{\dagger})^{1/2}V$, while spectral clustering maps the nodes from our graph to the rows y_i of our eigenvectors matrix V. Hence, building clusters based on the Euclidean distance of the coordinates y_i can be interpreted as building clusters of the vertices in the graph, based on the CTD.

Does the CTD c_{ij} will depends on non-zero self-connectivity weights?

We can write c_{ij} in terms of the eigenspectrum of the unnormalized and normalized Laplacian graph. This fact could lead to confusion as we know that if we set non-zero self-transitioning weights, the eigenspectrum of the unnormalized Laplacian graph will remain invariant while in the case of the eigenspectrum of the normalized Laplacian graph will not. Hence, it is not trivial to see that c_{ij} will also remain invariant to non-zero self-transitioning weights as it can be expressed in terms of the eigenspectrum of L_{sym} which is not invariant.

Notice that this will not be the case for other quantities computed by using the normalized graph Laplacian. This in-variance property will affect only the CTD.

From this formulation we cannot see that c_{ij} is not invariant to self-connectivity weights so given that:

$$c_{ij} \propto \sum_{k=2}^{n} \frac{1}{\lambda_k}\Big(v_{kj} - v_{ki}\Big)^2 \propto \sum_{k=2}^{n} \frac{1}{\lambda_k^{sym}}\Big(\frac{v_{kj}^{sym}}{\sqrt{D_{jj}}} - \frac{v_{ki}^{sym}}{\sqrt{D_{ii}}}\Big)^2$$

We know that c_{ij} is invariant to self-connectivity weights as it can be written in terms of L (unnormalized graph Laplacian), which will mean that the terms on the RHS of the equation above (although computed with non-invariant components - referring the the eigenvalues, eigenvectors and diagonal matrix components) will result in an invariant term.

2.1 Laplacian Eigenmaps

Let the set of points $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\} \in \mathbb{R}^{n\times k}$, represent our dataset, where $\mathbf{x_i} = (x_{i1}, x_{i2}, ..., x_{ik})$ and $i = 1, ..., n$. In order to address the dimensionality reduction problem, we would like to find a new set of points $\mathbf{Y} = \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_n}\} \in \mathbb{R}^{n\times m}$ where $\mathbf{y_j} = (y_{j1}, y_{j2}, ..., y_{jm})$ and $j = 1, ..., n$ such that $m \ll n$.

2.1.1 The algorithm

- **Step 1:**
- **Step 2:**
- **Step 3:**

The following pseudo-code implements the Laplacian eigenmap algorithm for an ϵ-neighborhood *similarity graph* with connectivity weights assigned by the heat kernel. The eigenvectors of the unnormalized Laplacian matrix $\mathbf{L}$ define a new coordinate of the data such that $\mathbf{x_i} \to (y_{i1}, \ldots, y_{im})$ determines the embedding map.

Algorithm 1: Laplacian eigenmap algorithm

Data: Set of data points $\mathbf{X} \in \mathbb{R}^{n\times k}$
Input: parameters $\epsilon, t \in \mathbb{R}$, constant ones-vector $\mathbb{1} \in \mathbb{R}^n$
Output: $\mathbf{V} \in \mathbb{R}^{n\times m}, \mathbf{\Lambda} \in \mathbb{R}^m$

Compute $\mathbf{W}$;
for $i = 1, ..., n$ **do**
  for $j = 1, ..., n$ **do**
    if $||\mathbf{x_i} - \mathbf{x_j}||^2 < \epsilon$ **then**
      $W_{ij} = e^{\frac{-||x_i - x_j||^2}{t}}$
    else
      $W_{ij} = 0$
Compute $\mathbf{D} = \text{diag}(\mathbf{W} \times \mathbb{1})$;
Compute $\mathbf{L} = \mathbf{D} - \mathbf{W}$;
Compute $[\mathbf{V}, \mathbf{\Lambda}] = \text{eigen}(\mathbf{L})$
for $i = 1, ..., n$ **do**
  sort$(\mathbf{v_i}, \lambda_\mathbf{i})$ by λ
Set $\mathbf{V} = (\mathbf{v_2}, \ldots, \mathbf{v_{m-1}})$ and $\mathbf{\Lambda} = (\lambda_2, \ldots, \lambda_{m-1})$

2.2 Justification of the algorithm

2.2.1 Optimal Embeddings

Denote $G = (V, E)$ to be the *similarity graph* representing our data with a set of *vertices* V and *edges* E. Each point of our data set, is represented by an element in the set of vertices, such that $\mathbf{x_i} \in \mathbf{X}$, corresponds to the node $\mathbf{v_i} \in \mathbf{V}$ for $i = 1, ..., n$.

We will now want to find a set of points $\mathbf{Y} = \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_n}\}$ such that, if we map G to $\mathbf{Y}$, the new set of points will provide a low-dimensional representation of the data that preserves local information "optimally".

Optimal embedding problem in terms of the unnormalized Laplacian (i)

Let

$$\sum_{ij}(y_i - y_j)^2 W_{ij} \tag{23}$$

define an objective function, where y_i and y_j represent the i:th and j:th element of $\mathbf{y}$ while W_{ij} denotes the connectivity weight between the pair of nodes i and j. Whenever the distance between $\mathbf{x_i}$ and $\mathbf{x_j}$ is large (small) the assigned connectivity weight W_{ij} takes a lower (higher) value which ensures minimizing the objective function (26).

From (1) we know that for any vector $\mathbf{y}$

$$\frac{1}{2}\sum_{ij}(y_i - y_j)^2 W_{ij} = \mathbf{y}'(D - W)\mathbf{y} = \mathbf{y}'L\mathbf{y}$$

The minimization problem can be reduced to finding a solution to:

$$\arg\min_{y \in \mathbb{R}^n} \quad \mathbf{y}'L\mathbf{y} \tag{24a}$$

$$\text{subject to} \qquad \mathbf{y}'D\mathbf{y} = 1 \tag{24b}$$

$$\mathbf{y}'D\mathbb{1} = 0 \tag{24c}$$

The vector $\mathbf{y}$ that minimizes the objective function (24a) is given by the smallest eigenvalue solution of the generalized eigenvalue problem:

$$L\mathbf{y} = \lambda D\mathbf{y}$$

Notice in this case, that although the optimal embedding problem is defined in terms of the unnormalized Laplacian L, the Laplacian eigenmaps are defined in terms of the normalized Laplacian graphs as the solution to the minimization problem is given by the the generalized eigenvalue problem, i.e $L\mathbf{y} = \lambda D\mathbf{y}$ instead of $L\mathbf{y} = \lambda\mathbf{y}$.

Referring to the weighted constraints for finding a solution to (24a) given the generalized eigenvalue problem, the vector $\mathbf{y}$ is required to be normalized and orthogonal to the constant vector $\mathbb{1}$, i.e ,

$$\mathbf{y}'D\mathbf{y} = \langle yD, y\rangle = \sum_{i=1}^{n} y_i^2 D_{ii} = 1$$

$$\mathbf{y}'D\mathbb{1} = \langle yD, \mathbb{1}\rangle = \sum_{i=1}^{n} D_{ii} y_i = 0$$

The normalizing constraint $\mathbf{y}'D\mathbf{y} = 1$ ensures unit length vectors in order to avoid arbitrary large y components (weighted unit norm), while the orthogonality constraint eliminates the trivial solution given

by the zero eigenvalue, (weighted orthogonality) i.e without this constraint the solution to the generalized eigenvalue problem $L\mathbf{y} = \lambda D\mathbf{y}$ (which follows from (5)) is given by the zero eigenvalue with constant eigenvector $\mathbb{1}$, as:

$$\begin{aligned} L\mathbf{v_1} = (D-W)\mathbb{1} &= \begin{bmatrix} \sum_{j=1}^{n} \omega_{1j} & & \\ & \ddots & \\ & & \sum_{j=1}^{n} \omega_{nj} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} - \begin{bmatrix} \omega_{11} & & & & \\ \omega_{21} & \omega_{22} & & & \\ \omega_{31} & \omega_{32} & \omega_{33} & & \\ \vdots & \vdots & & \ddots & \\ \omega_{n1} & \omega_{n2} & \dots & & \omega_{nn} \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \sum_{j=1}^{n} \omega_{1j} \\ \vdots \\ \sum_{j=1}^{n} \omega_{nj} \end{bmatrix} - \begin{bmatrix} \sum_{j=1}^{n} \omega_{1j} \\ \vdots \\ \sum_{j=1}^{n} \omega_{nj} \end{bmatrix} = 0 \ D\mathbb{1} \end{aligned} \tag{25}$$

m-Dimensional embedding problem:

Consider Y to be an $n \times m$ matrix, with rows determining the m-dimensional representation of the nodes in the graph. We re-write the objective function as:

$$\begin{aligned} \sum_{i,j} ||(y_i - y_j)||^2 \ W_{ij} &= \sum_{ij} W_{ij} \sum_{k=1}^{m} (y_{ik} - y_{jk})^2 \\ &= \sum_{ij} W_{ij} \Big(\sum_{k=1}^{m} y_{ik}^2 + y_{jk}^2 - 2 y_{ik} y_{jk} \Big) \\ &= \sum_{i,k} y_{ik}^2 \sum_{j} W_{ij} + \sum_{j,k} y_{jk}^2 \sum_{i} W_{ij} - 2 \sum_{i,j,k} W_{ij} y_{ik} y_{jk} \\ &= \sum_{i,k} y_{ik}^2 D_{ii} + \sum_{j,k} y_{jk}^2 D_{jj} - 2 \sum_{i,j,k} W_{ij} y_{ik} y_{jk} \\ &= 2 \sum_{i,j,k} y_{ik} D_{ij} y_{jk} - 2 \sum_{i,j,k} W_{ij} y_{ik} y_{jk} \\ &= 2 \sum_{i,j,k} y_{ik} (D_{ij} - W_{ij}) y_{jk} \\ &= 2 \sum_{i} \langle y_i L, y_i \rangle = 2 \ \mathrm{tr}(Y^T L Y) \end{aligned} \tag{26}$$

The minimization problem for an m-dimensional embedding is reduced to finding a solution to

$$\arg \min_{Y \in \mathbb{R}^{n \times m}} \ 2 \ \mathrm{tr}(Y^T L Y) \tag{27a}$$

$$\text{subject to} \qquad \mathbf{Y}' D \mathbf{Y} = I \tag{27b}$$

$$\tag{27c}$$

Where the constraint $\mathbf{Y}' D \mathbf{Y}$ requires $\mathbf{Y}$ to be an orthogonal matrix with normalized column vectors, such that

$$(\forall i \ v_i \neq 0) \cap (\forall \ i \neq j \ \langle v_i, v_j \rangle = 0) \quad \text{and} \quad \forall i \langle v_i, v_i \rangle = 1$$

Which yields to;

$$\begin{aligned}\mathbf{Y}'D\mathbf{Y} &= D\begin{bmatrix}\mathbf{v_1'}\\ \mathbf{v_2'}\\ \vdots\\ \vdots\\ \mathbf{v_n'}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{v_1},\mathbf{v_2},\ldots,\ldots,\mathbf{v_n}\end{bmatrix}\\ &= D\mathbb{1}\cdot\begin{bmatrix}\langle v_1,v_1\rangle & & & & \\ \langle v_2,v_1\rangle & \langle v_2,v_2\rangle & & & \\ \langle v_3,v_1\rangle & \langle v_3,v_2\rangle & \langle v_3,v_3\rangle & & \\ \vdots & \vdots & \vdots & \ddots & \\ \langle v_n,v_1\rangle & \langle v_n,v_2\rangle & \ldots & & \langle v_n,v_n\rangle\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & \cdots & 0\\ 0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1\end{bmatrix}\end{aligned} \tag{28}$$

Optimal embedding problem in terms of the unnormalized Laplacian (ii)

The objective is to minimize

$$\arg\min_{y\in\mathbb{R}^n} \quad \mathbf{y}'L\mathbf{y} \tag{29a}$$

$$\text{subject to} \qquad \mathbf{y}'\mathbf{y} = 1 \tag{29b}$$

$$\mathbf{y}'\mathbb{1} = 0 \tag{29c}$$

The solution to (29a) is associated to the standard eigenvalue problem:

$$L\mathbf{y} = \lambda\mathbf{y}$$

The constraint $\mathbf{y}'\mathbf{y}$ requires the vector $\mathbf{y}$ to have unit norm, while $\mathbf{y} \perp \mathbb{1}$ requires the vector to be orthogonal to the constant vector $\mathbb{1}$. Solving (29a) without constrains gives a trivial solution $\lambda_1 = 0$ and $v_1 = \mathbb{1}$.

Optimal embedding problem in terms of the normalized Laplacian

We could also reformulate the objective function on (26) such that it is related to the symmetric normalized Laplacian matrix:

$$\sum_{ij}(y_i - y_j)^2 W_{i,j} = \sum_{ij}\Big(\frac{u_i}{\sqrt{D_{ii}}} - \frac{u_j}{\sqrt{D_{jj}}}\Big)^2 W_{i,j} \tag{30}$$

Where $\mathbf{u} = D^{1/2}\mathbf{y}$, and so from (3) we know that for any vector $\mathbf{u}$ the following relation holds:

$$\frac{1}{2}\sum_{ij}\Big(\frac{u_i}{\sqrt{D_{ii}}} - \frac{u_j}{\sqrt{D_{jj}}}\Big)^2 W_{ij} = \mathbf{u}'L_{sym}\mathbf{u}$$

Which can be reduced to finding the solution of

$$\arg\min_{u\in\mathbb{R}^n} \quad \mathbf{u}'L_{sym}\mathbf{u} \tag{31a}$$

$$\text{subject to} \qquad \mathbf{u}'D\mathbf{u} = 1 \tag{31b}$$

$$\mathbf{u} \perp D^{1/2}\mathbb{1} \tag{31c}$$

The vector $\mathbf{u}$ that minimizes the objective function in (31a) is given by the smallest eigenvalue solution of the standard eigenvalue problem:

$$L_{sym}\mathbf{u} = \lambda\mathbf{u}$$

Notice that

$$\underset{\mathbf{y}'D\mathbf{y}=1;\;\; D\mathbf{y}\perp\mathbb{1}}{\arg\min}\; \mathbf{y}'L\mathbf{y} = \underset{\mathbf{u}'D\mathbf{u}=1;\;\; \mathbf{u}\perp D^{1/2}\mathbb{1}}{\arg\min}\; \mathbf{u}'L_{sym}\mathbf{u}$$

Furthermore, the solution to both minimization problems are the same iff they share the same eigenvalue λ with eigenvectors relating as $\mathbf{y} = D\mathbf{u}$, where $\mathbf{y}$ is the eigenvector solution to (24a) and $D\mathbf{u}$ the eigenvector solution to (31a). However, solving (31a) is simpler than solving (24a) as it reduces the generalized eigenvalue problem to an ordinary problem. The trivial solution of (31a) is given by $\lambda_1 = 0$ and $v_1 = D^{1/2}\mathbb{1}$ as:.

$$\begin{aligned}
L_{sym}\mathbf{u_1} &= (I - D^{-1/2}WD^{-1/2})\; D^{1/2} \\
&= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \sqrt{D_{11}} \\ \sqrt{D_{22}} \\ \sqrt{D_{33}} \\ \vdots \\ \sqrt{D_{nn}} \end{bmatrix} - \begin{bmatrix} \omega_{11}/D_{11} & & & & \\ \omega_{21}/D_{22} & \omega_{22}/D_{22} & & & \\ \omega_{31}/D_{33} & \omega_{32}/D_{33} & \omega_{33}/D_{33} & & \\ \vdots & \vdots & & \ddots & \\ \omega_{n1}/D_{nn} & \omega_{n2}/D_{nn} & \cdots & & \omega_{nn}/D_{nn} \end{bmatrix} \begin{bmatrix} \sqrt{D_{11}} \\ \sqrt{D_{22}} \\ \sqrt{D_{33}} \\ \vdots \\ \sqrt{D_{nn}} \end{bmatrix} \\
&= \begin{bmatrix} \sqrt{D_{11}} \\ \sqrt{D_{22}} \\ \sqrt{D_{33}} \\ \vdots \\ \sqrt{D_{nn}} \end{bmatrix} - \begin{bmatrix} \sqrt{D_{11}} \\ \sqrt{D_{22}} \\ \sqrt{D_{33}} \\ \vdots \\ \sqrt{D_{nn}} \end{bmatrix} = 0\; D^{1/2}
\end{aligned} \tag{32}$$

In the case of the m**-dimensional embedding problem**

$$\begin{aligned}
\sum_{i,j} \left|\left| (\frac{u_i}{\sqrt{D_{ii}}} - \frac{u_j}{\sqrt{D_{jj}}}) \right|\right|^2 W_{ij} &= \sum_{ij} W_{ij} \sum_{k=1}^{m} \Big(\frac{u_{ik}}{\sqrt{D_{ii}}} - \frac{u_{jk}}{\sqrt{D_{jj}}} \Big)^2 \\
&= \sum_{ij} W_{ij} \left(\sum_{k=1}^{m} \frac{u_{ik}^2}{D_{ii}} + \frac{u_{ik}^2}{D_{jj}} - 2\frac{u_{ik}u_{ik}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \right) \\
&= \sum_{i,k} \frac{u_{ik}^2}{D_{ii}} \sum_{j} W_{ij} + \sum_{j,k} \frac{u_{jk}^2}{D_{jj}} \sum_{i} W_{ij} - 2\sum_{i,j,k} W_{ij} \frac{u_{ik}\; u_{jk}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \\
&= \sum_{i,k} \frac{u_{ik}^2}{D_{ii}} D_{ii} + \sum_{j,k} \frac{u_{jk}^2}{D_{jj}} D_{jj} - 2\sum_{i,j,k} W_{ij} \frac{u_{ik}\; u_{jk}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} \\
&= 2\sum_{i,j,k} u_{ik}u_{jk} - 2\sum_{i,j,k} \frac{u_{ik}}{\sqrt{D_{ii}}} W_{ij} \frac{u_{jk}}{\sqrt{D_{jj}}} \\
&= 2\sum_{i,j,k} u_{ik} \Big(1 - \frac{1}{\sqrt{D_{ii}}} W_{ij} \frac{1}{\sqrt{D_{ii}}} \Big) u_{jk} \\
&= 2\sum_{i} \langle u_i L_{sym}, u_i \rangle = 2\; \mathrm{tr}(U^T\; L_{sym}\; U) = 2\; \mathrm{tr}(D^{1/2}Y\; L_{sym}\; D^{1/2}Y)
\end{aligned} \tag{33}$$

The minimization problem for an m-dimensional embedding is reduced to finding a solution to

$$\arg \min_{U \in \mathbb{R}^{n \times m}} \; 2 \; \mathrm{tr}(U^T L_{sym} U) \tag{34a}$$

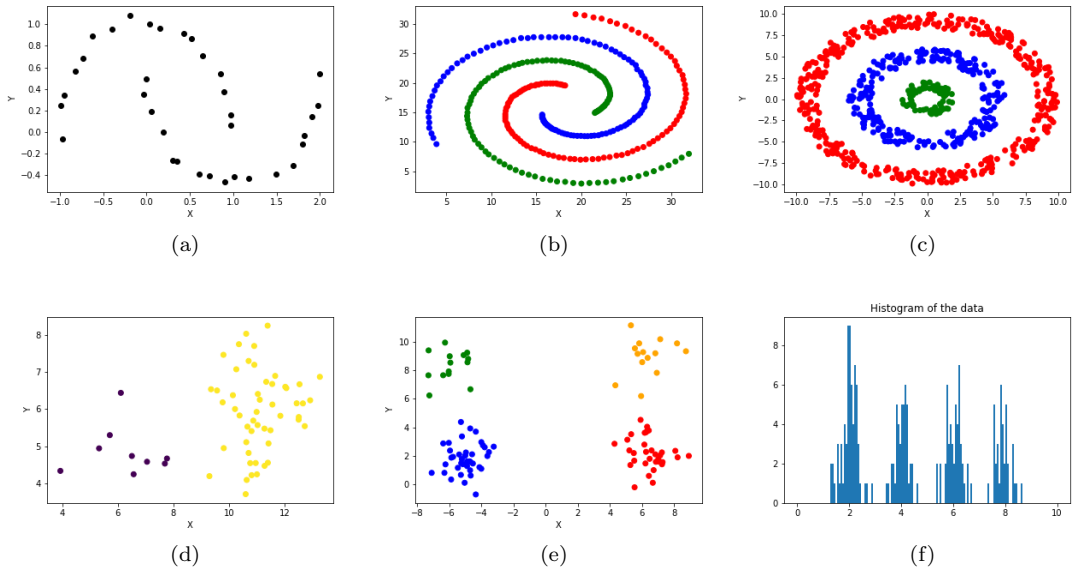
$$\text{subject to} \qquad \mathbf{U}' D \mathbf{U} = I \tag{34b}$$

$$\tag{34c}$$

3 Data Experiments: Spectral Clustering

We will now use simple toy data-set to visualize clustering. The data sets used are the following:

1. **Two Half-moon dataset**: 35 data points. The top moon contains 17 data points and the bottom moon 18.
2. **Spiral dataset**: 312 data points divided into 3 intertwined swirls with sizes 101, 105 and 106.
3. **Nested Circles dataset**: 720 data points divided into 3 nested circles with 400, 250 and 70 data points respectively.
4. **Imbalanced and inhomogeneous data set**: 60 data points divided into two clusters of different sizes, the smaller cluster contains 9 data points and the larger cluster contains 51 data points.
5. **Data set with a hierarchical organization**: 100 data points of two main clusters divided into two sub-clusters. Each main cluster is of size 53 and 47 respectively.
6. **Gaussian Mixture**: sample of 200 data points drawn according to a mixture of four Gaussians.



(a) (b) (c)

(d) (e) (f)

Figure 2: Toy data-sets: (a) Two Half-moon data set, (b) Spiral dataset, (c) Nested Circles data set, (d) Imbalanced and inhomogeneous dataset, (e) Data set with a hierarchical organization, (f) Gaussian Mixture

A fully connected graph with weights obtained by the Gaussian similarity function was constructed for all cases. Tests were performed with the three versions of the spectral clustering algorithm (as described in [Luxburg(2007)]).

In spectral graph theory, the eigenspectrum of the graph Laplacian gives us information about the connectivity of a graph and how strong these connections are, while the eigenvector related to each eigenvalue provides the information of the arrangement of the nodes into a particular graph component.

If our data had k components, we will then expect the smallest eigenvalue to have multiplicity k. Moreover, the spectral gap (smallest nonzero eigenvalue) will depend on how well pronounced the clusters are in our data. Meaning that the largest the eigengap($\gamma_k = |\lambda_k - \lambda_{k+1}|$) is, the better spectral clustering will work.

We know that for any matrix L

$$\mathrm{Tr}(D) = \sum_{i=1}^{n} \lambda_i$$

Where λ_i for $i = 1, ..., n$ are the eigenvalues, and D a square matrix. Hence we know that the sum of eigenvalues of the normalized and unnormalized graph Laplacian, equals to the sum of the diagonalized degree matrix where D_{ii} is the total sum of connectivity weights of the ith instance, i.e it represents how well connected the i'th data point is to the rest of the network, meaning that the highest the sum of eigenvalues is, the stronger connections the network will have.

3.1 Two Half Moons data set

From the two half-moons dataset it can be observed that (i) the number of data points is not very large, (ii) clusters are noisy and not very dense, (iii) the dataset is balanced.

From the Graphs Laplacian we can identify that the eigenspectrum of L_{rw} and L_{sym} are very alike, however it differs with the unnormalized graph Laplacian L after the 6th eigenvalue, as the difference between higher order eigenvalues seems to be the same. There are 4 main gaps in the first 10 smaller eigenvalues: between the (i) 2nd and 3rd eigenvalue, (ii) 4th and 5th eigenvalue, (iii) 6th and 7th eigenvalue, and (iv) 8th and 9th eigenvalue.

Referring to the eigenspectrum of the normalized random walk graph Laplacian $L_{_{rw}}$, the smallest eigenvalue has multiplicity 2, however, the spectral gap is not very large (0.014) compared to the rest of gaps in the eigenspectrum $(0.057, 0.08, 0.055)$, which suggests a hierarchical structure in the data, meaning that our data is close to have 4 weakly connected in between components and strongly connected within component. In terms of higher order eigenvalues, the number of eigenvalues between each eigengap is of 2, which coincides with the number of components.

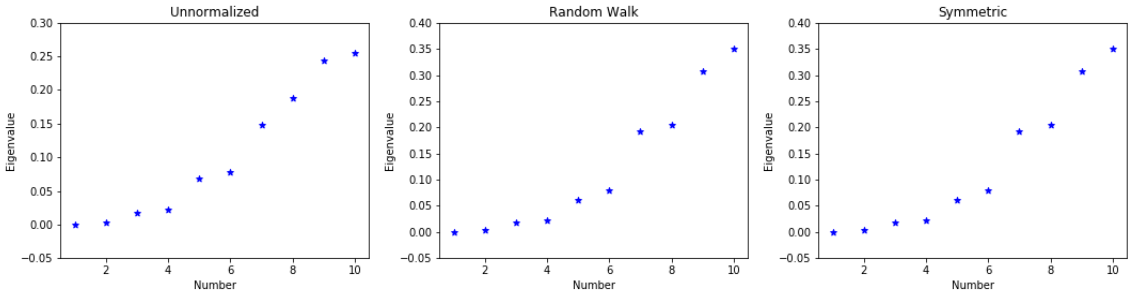


Figure 3: Eigenvalues of L, L_{rw}, L_{sym}

In Figure (4), the eigenvector of the first and smallest eigenvalue is the constant eigenvector and it does not provide any information about the connectivity of the data, while the eigenvector associated with the 2nd eigenvalue gives us a natural boundary where we can easily separate the data-points into two components.

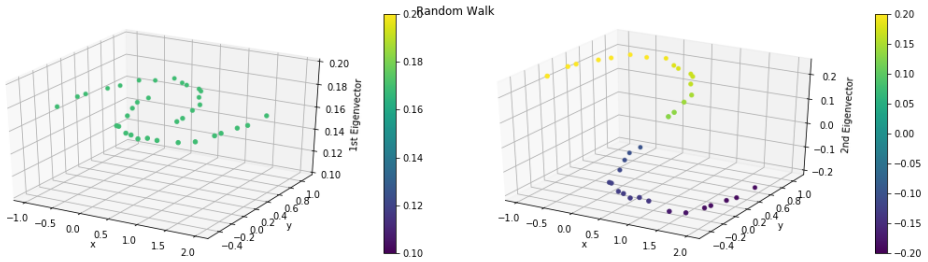


Figure 4: Eigenvectors of L_{rw} based on the fully connected graph with edges weighted by the Gaussian similarity function with $\sigma = 0.05$ vs. data

Figure (5) illustrates the eigenvectors corresponding to higher order eigenvalues, each row represents the eigenvectors of the eigenvalues that separates the 3rd, 4th and 5th eigengap. Notice that in all pairs of eigenvectors one of the two components is set to zero while the remaining data component is divided in sub-clusters.

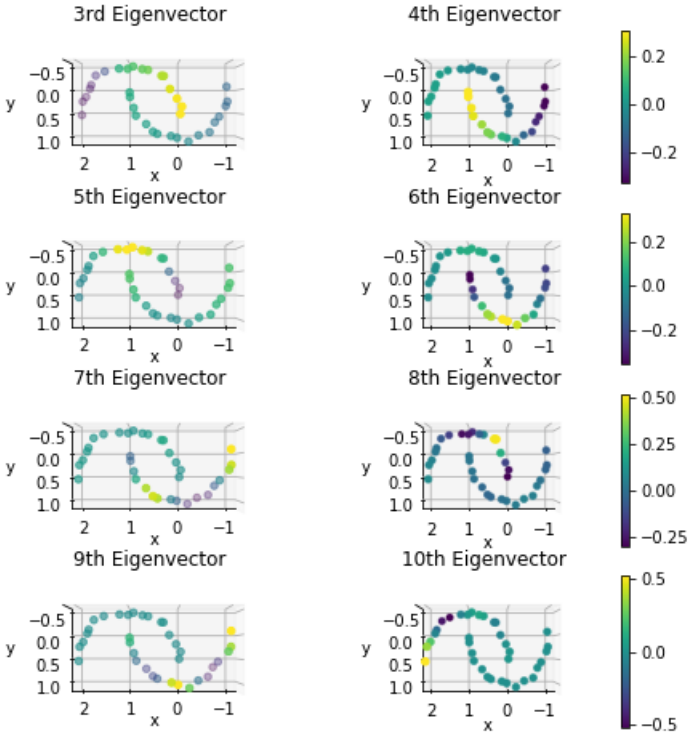


Figure 5: 3rd-10th eigenvectors of the Normalized random walk Graph Laplacian vs. Half-moon data seen from above

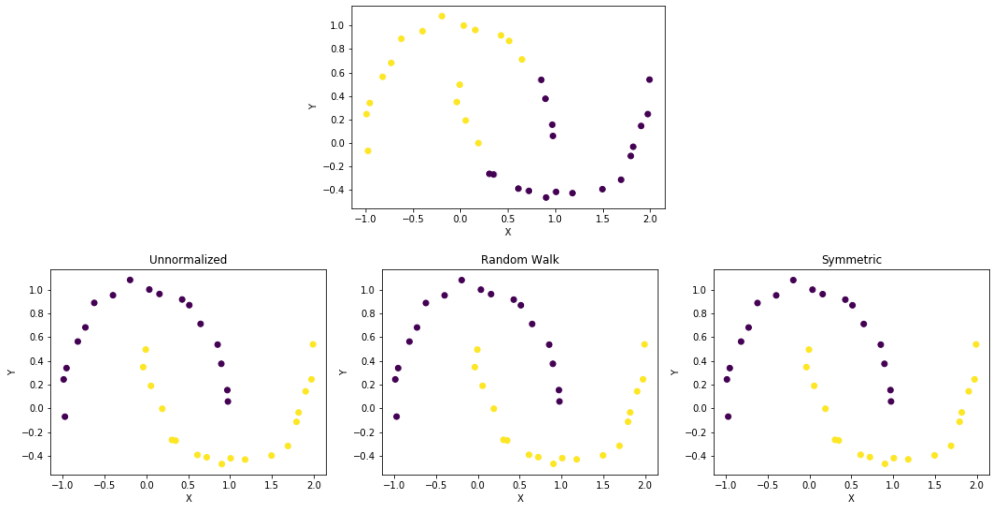


Figure 6: First row: K-means clustering algorithm performance, Second row: Spectral clustering algorithm performance based on L, L_{rw} and L_{sym}, respectively.

3.2 Spiral data set

The spiral dataset is a balanced data set with well pronounced clusters. From the Graphs Laplacian we can identify that the eigenspectrum of L, L_{rw} and L_{sym} are very alike and with well-defined eigen gaps.

The image below represents the first 12 smaller eigenvalues, all of them with multiplicity 3. The eigen gaps are between: the (i) 3rd and 4th eigenvalue, (ii) 6th and 7th eigenvalue, and (iii) 9th and 10th eigenvalue.

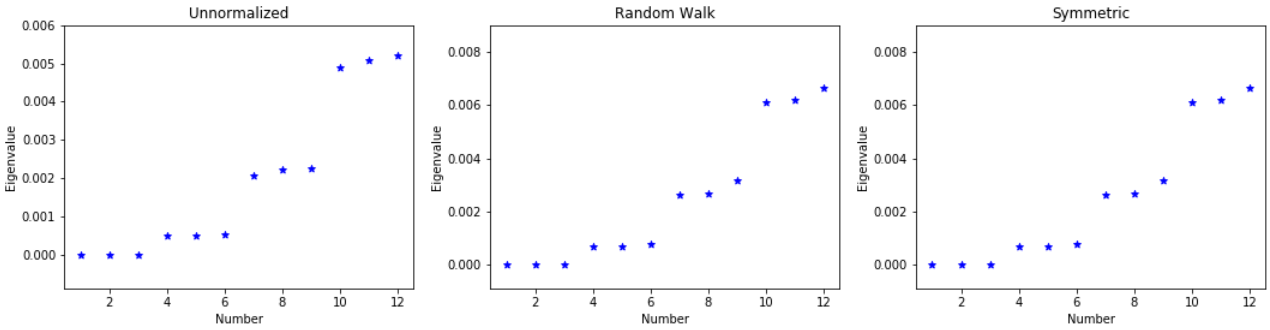


Figure 7: Eigenvalues of L, L_{rw}, L_{sym}

Referring to the normalized random walk graph Laplacian L_{rw}, we can notice that (i) the first eigenvalue is constant, i.e, all data points belong to the same cluster, (ii) the second eigenvector divides the data points into two components, and (iii) the 3rd eigenvector divides a component of the 2nd eigenvector into two, giving a clear boundary that divides the data into 3 strongly connected components.

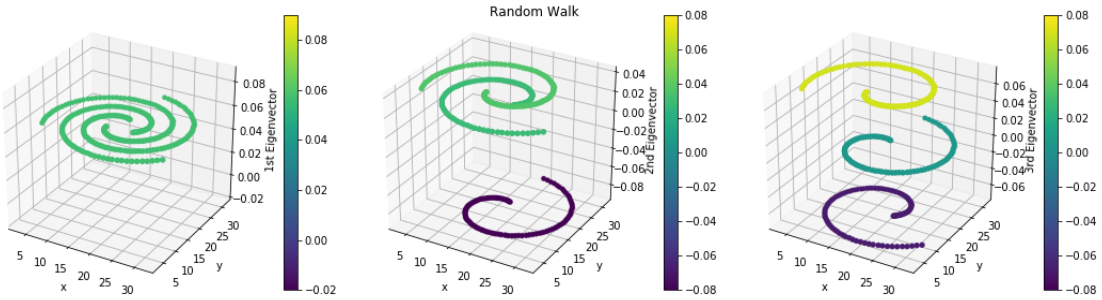


Figure 8: Eigenvectors of L_{rw} based on the fully connected graph with edges weighted by the Gaussian similarity function with $\sigma = 0.5$

Further analysis of higher order eigenvalues tells us: (i) the multiplicity of higher order eigenvalues coincide with the number of clusters, (ii) for higher order eigenvalues, the eigenvector sets two out of the three clusters at zero, and then divides the remaining component into sub-clusters (iii) the process in (ii) is performed for each data component.

For example, the first row in Figure (9) shows the 4th-6th eigenvector, notice that each of the three eigenvectors divides in sub-clusters a different spiral. The same figures can also be seen from above in Figure (23). In the spiral dataset, the eigengap for higher order eigenvalues identifies hierarchical structures in the data.

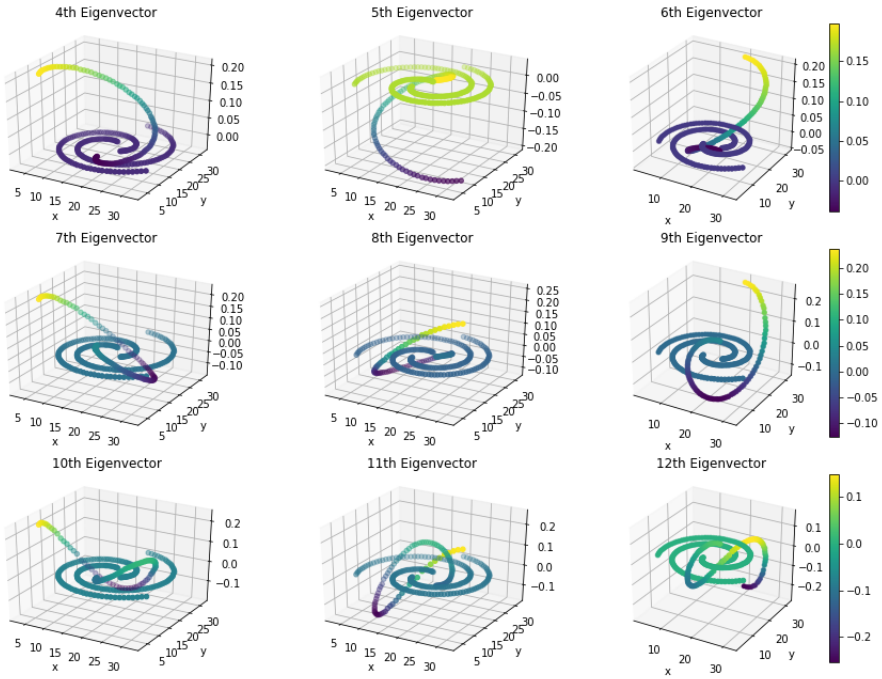


Figure 9: 4th-12th eigenvectors of the Normalized random walk Graph Laplacian vs. data

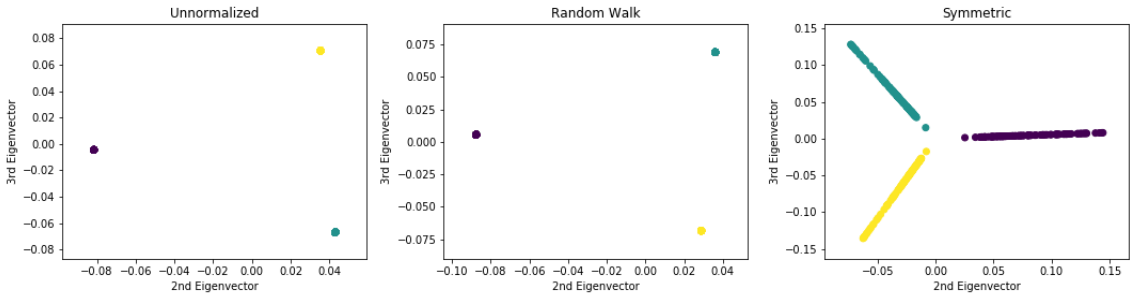


Figure 10: New representation of the data based on L, L_{rw} and L_{sym} respectively.

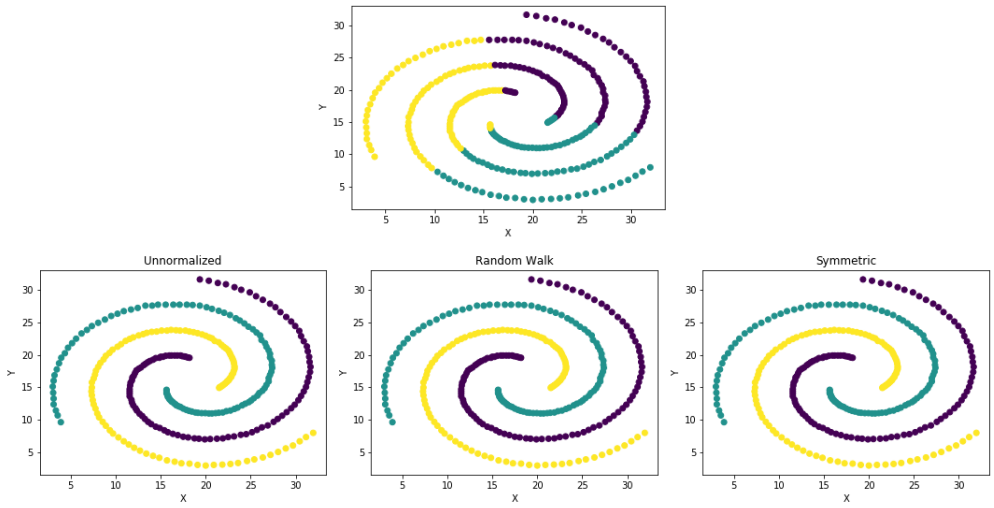


Figure 11: First row: K-means clustering algorithm performance, Second row: Spectral clustering algorithm performance based on L, L_{rw} and L_{sym} respectively.

3.3 Circles data set

As it can be seen in Figure (12) the Laplacians are very similar to each other as their eigenspectrum are very alike for the three cases.

If we consider the eigenspectrum of the normalized random walk Graph Laplacian we can notice that there are 3 main gaps in the first 12 smaller eigenvalues: between (i) the 3rd and 4th eigenvalue, (ii) 5th and 6th eigenvalue, and (iii) 9th and 10th eigenvalue.

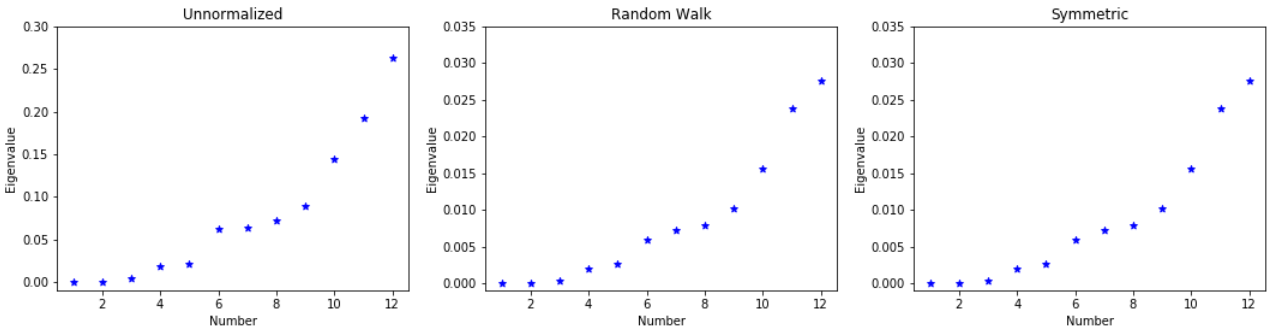


Figure 12: Eigenspectrum of L, L_{rw}, L_{sym}

From Figure (12) we can also identify that (i) the gaps distances increase as the eigenvalues increase, in this particular case, the distance gaps are 0.0016, 0.0034 and 0.0054 respectively, (ii) the second smallest eigenvalue has multiplicity of 2 , (iii) after the 9th eigenvalue the difference between eigenvalues are approximately the same.

As mentioned before the eigenspectrum of the Graphs Laplacian depend on how well pronounced the clusters are in our data. In the case of the nested circles dataset, the clusters are noisy and imbalanced which could be a reason why the gaps in the eigenspectrum are not well defined and the interpretation of the eigenvectors for higher order eigenvalues, becomes harder.

Figure (13) illustrates the eigenvectors corresponding to the smallest eigenvalue, notice that: (i) the 1st eigenvector is constant, hence it does not provide any information about the connectivity of the data, (ii) the 2nd eigenvector divides the data into two components, and (iii) the 3rd eigenvector divides a data component of the 2nd eigenvector into two, providing a clear boundary between the three components.

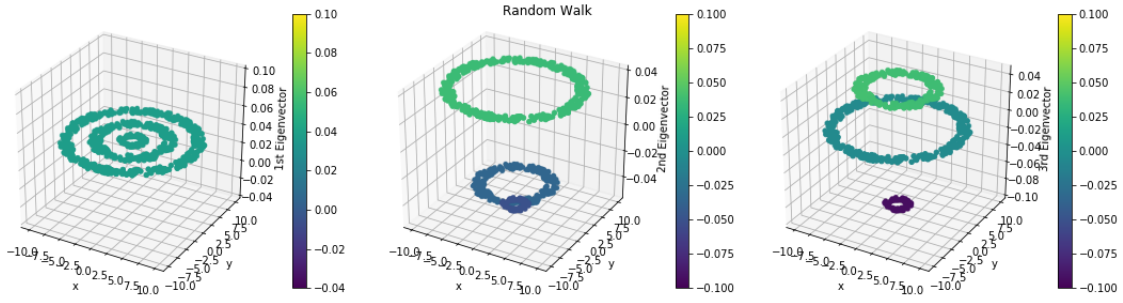


Figure 13: Eigenvectors of L_{rw} based on the fully connected graph with edges weighted by the Gaussian similarity function with $\sigma = 1$

Figure (14) illustrates the eigenvectors corresponding to higher order eigenvalues. From Figure (12) we

verified that the spectral gap is quite small, which tells us that given the multiplicity of the second smallest eigenvalue (2) our data is likely to have 5 clusters, instead of 2.

The 4th and 5th eigenvector sets the data of the two interior circles at zero, while divides into sub-clusters the data forming the external circle. This behavior is repeated in eigenvectors corresponding to higher order eigenvalues. However, notice that the components of all eigenvectors corresponding to the the data of the smaller/internal circle is set to zero.

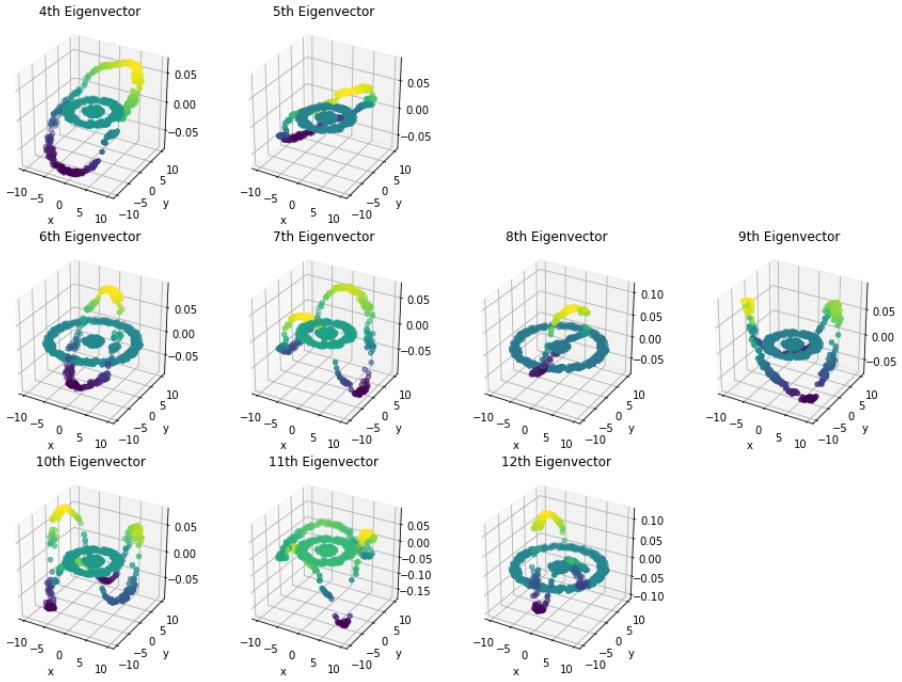


Figure 14: 4th-12th eigenvectors of the Normalized random walk Graph Laplacian vs. data

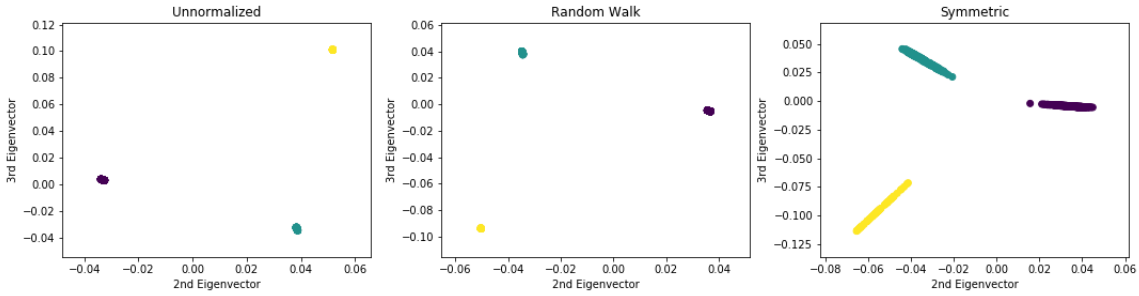


Figure 15: New representation of the data based on L, L_{rw} and L_{sym} respectively.

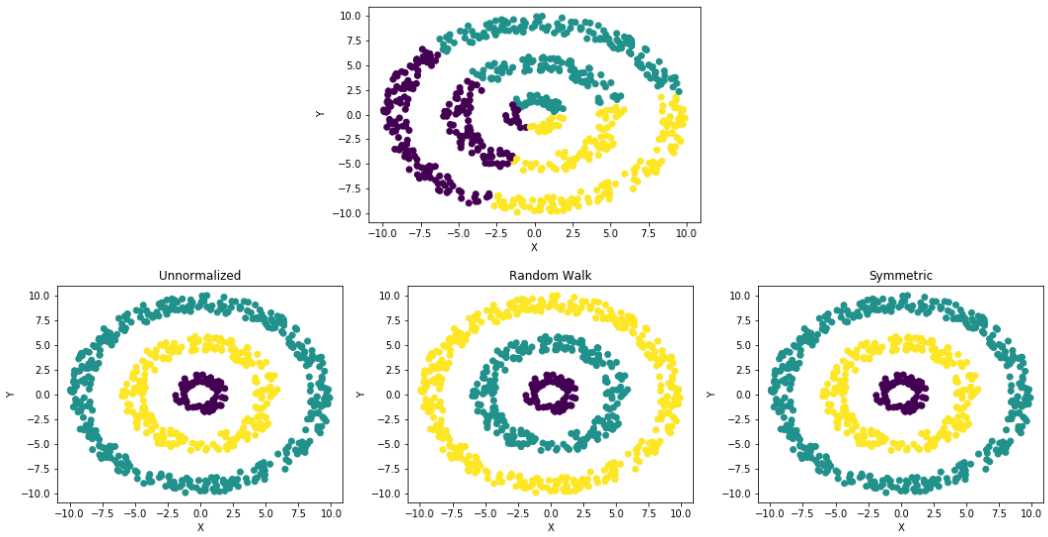


Figure 16: First row: K-means clustering algorithm performance, Second row: Spectral clustering algorithm performance based on L, L_{rw} and L_{sym} respectively.

3.4 Imbalanced data set

Figure (17) shows the eigenspectrum of L, L_{rw} and L_{sym}, respectively. As our dataset is composed by two clusters, we would expect to have a clear gap between the second and third eigenvalue, however, notice that in the case of the unnormalized graph Laplacian L there is not a clear gap between the second and third eigenvalue, if not, there is a bigger gap between the 5th and 6th eigenvalue, as opposed to the eigenspectrum of L_{rw} and L_{sym}.

The first gap on the eigenspectrum of the normalized graph Laplacian L is between the 4th and 5th eigenvalue. One of the reasons for this behaviour could be the inhomogeneous property of the data, as the data is less dense on the smaller cluster area and more dense on the bigger cluster area. Furthermore it can be noticed that there are two isolated points in the smallest cluster, presumably outliers.

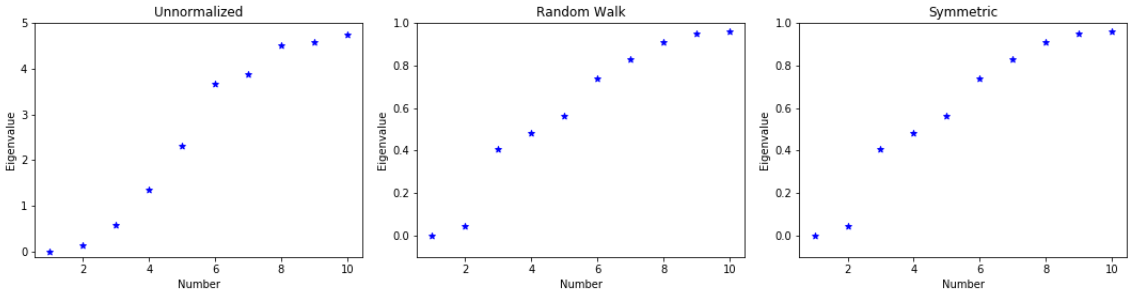


Figure 17: Eigenvalues of L, L_{rw}, L_{sym}

A reason why the unnormalized and normalized graph Laplacian differ considerably is because the sum of connectivity weights per data point are broadly distributed which can also be influenced by the small size of the dataset and the fact that the clusters are not well-pronounced. Hence, using the eigenvectors of the normalized graph Laplacian L_{rw} and L_{sym} seems to be a better option.

The figure below plots the first eigenvectors elements of normalized L_{rw} Graph Laplacian vs. the data points coordinates. Notice that for both graphs there is a clear gap in the elements of the second eigenvector which represent the weakest link between the two clusters.

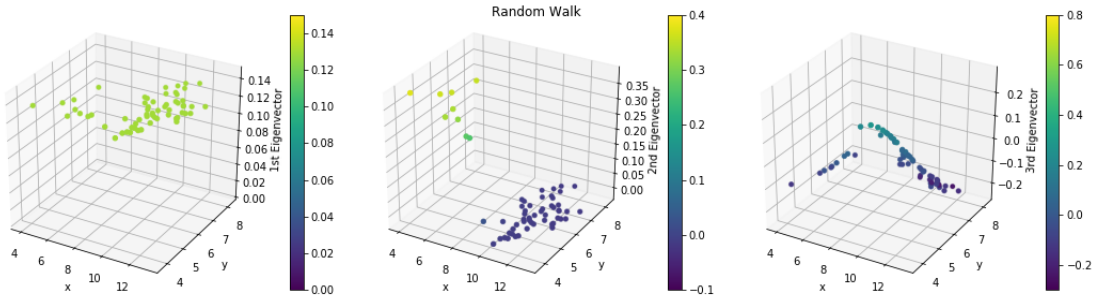


Figure 18: Eigenvectors of L_{rw} based on the fully connected graph with edges weighted by the Gaussian similarity function with $\sigma = 2$

3.5 Data set with a hierarchical organization

From the eigenspectrum of the data set with a hierarchical organization we can notice that there are 2 gaps in the first 10 smaller eigenvalues: between the (i) 1st and 2nd eigenvalue, and the (ii) 3th and 4th eigenvalue. After the 4th eigenvalue there are not well defined gaps, as the difference between eigenvalues are approximately the same. The 4th and 5th eigenvalues suggests that our data is close to have 4 clusters instead of 2.

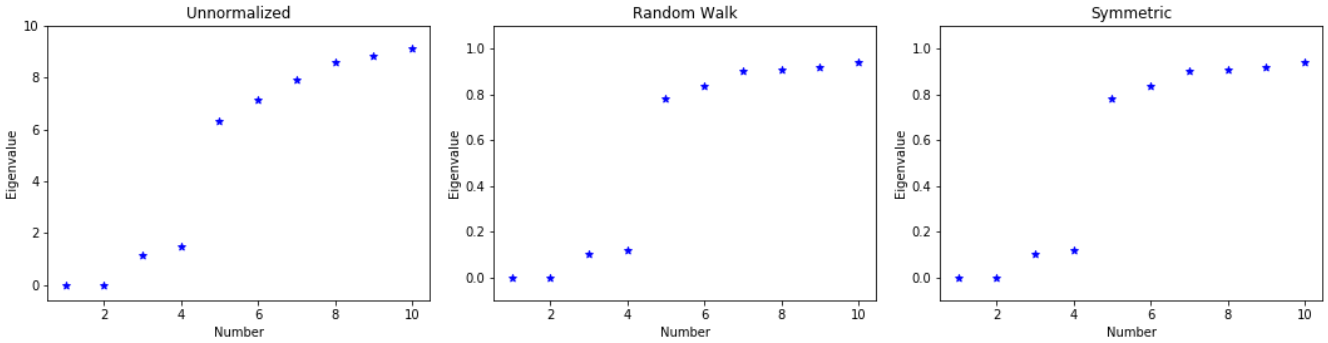


Figure 19: Eigenspectrum of L, L_{rw}, L_{sym}

Figure (20) illustrates the first two eigenvectors corresponding to the smallest eigenvalue, it can be noticed that (i) the components of the 1st eigenvector are all constant, (ii) the second eigenvector provides a clear boundary that divides the data into two components.

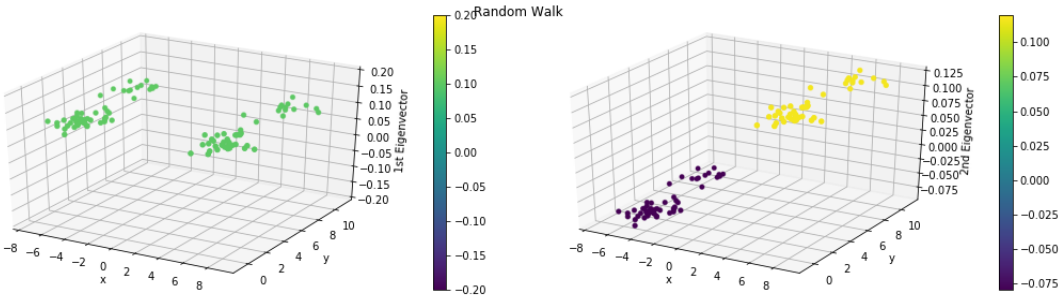


Figure 20: Eigenvectors of L_{rw} based on the fully connected graph with edges weighted by the Gaussian similarity function with $\sigma = 10$ vs. data

Figure (21) illustrates the eigenvectors corresponding to the second smallest eigenvalue which identifies a hierarchical structure in the data notice that from the two components identified by the second eigenvector, the 3rd eigenvector sets the first component to zero and divides the second component into 3 sub-clusters, (ii) the 4th eigenvector sets the second component to zero and divides the first component into 3 sub-clusters.

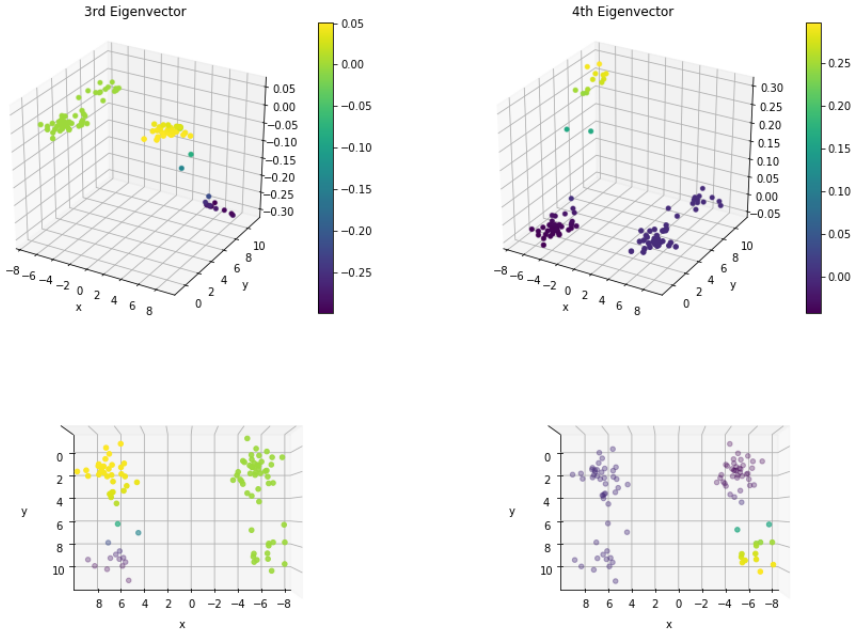
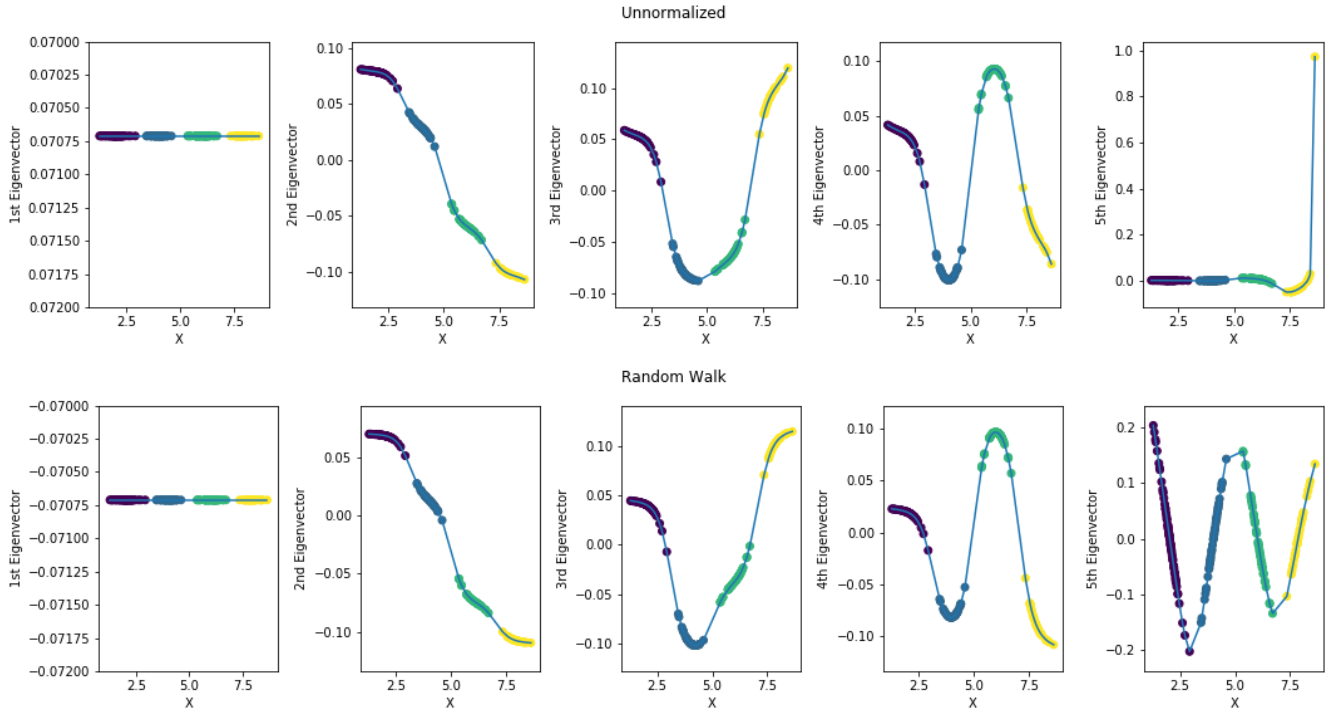


Figure 21: First row: 3rd and 4th eigenvectors of L_{rw} vs. data, second row: 3rd and 4th eigenvectors of L_{rw} vs, data seen from above

3.6 Mixture of Gaussians



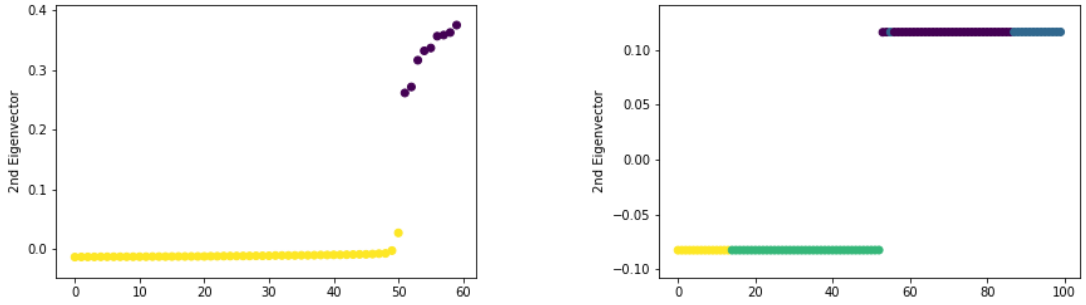
3.7 Graph cut point of view

In general, when we try to separate clusters in a dataset, either we look for the biggest gap between the elements of the eigenvector, or we consider the stiffness of the eigenvector. However, the stiffness of the eigenvector is not easy to estimate as the slope of the eigenvector could be more stiff in different areas, regardless of the size of the gap. Hence, looking for the biggest gap is a better option, as it is more reasonable to separate between the cluster's weakest link.

Another way to separate clusters in a graph could be by applying a graph cut criteria. In the case of balanced datasets such criteria could be to threshold the leading eigenvector at zero.

An illustrative example of this could be to consider the hierarchical structure data set and only look at the two main clusters, which form a balanced dataset. We can notice that the second eigenvector is about symmetric, beyond and below zero.

However, this is not necessarily the case for imbalanced data, if we look at our imbalanced dataset the second eigenvector is completely asymmetric below and above zero, so we can no long threshold at zero.



(a) (b)

Figure 22: 2nd Eigenvector for the balanced (a) and imbalanced (b) data sets of the Normalized Random Walk graph Laplacian L_{rw}.

.

A Appendices

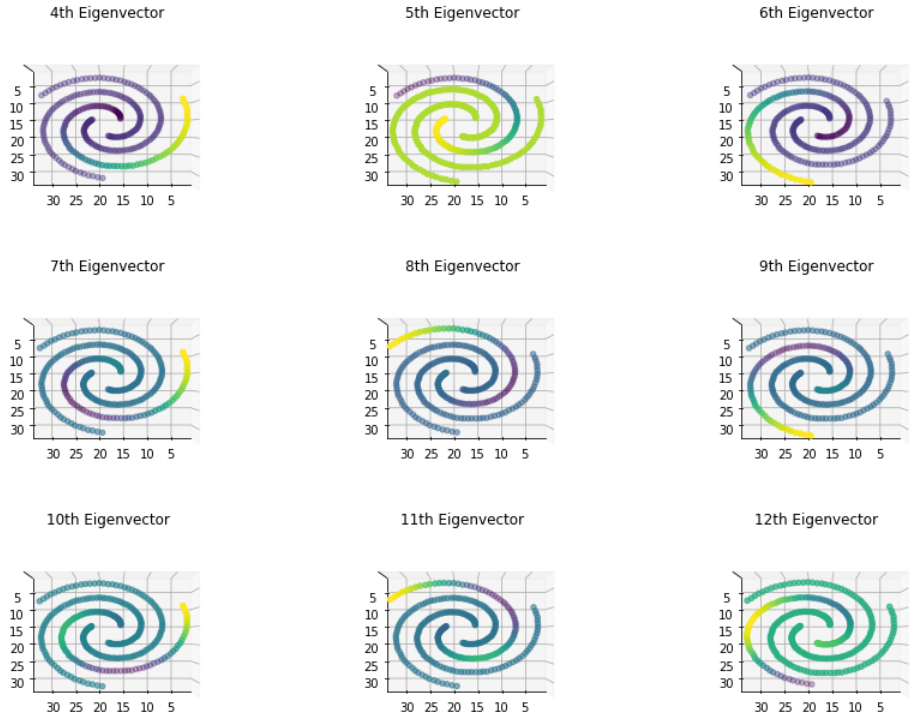


Figure 23: 4th-12th eigenvectors of the Normalized random walk Graph Laplacian vs. Spiral data seen from above

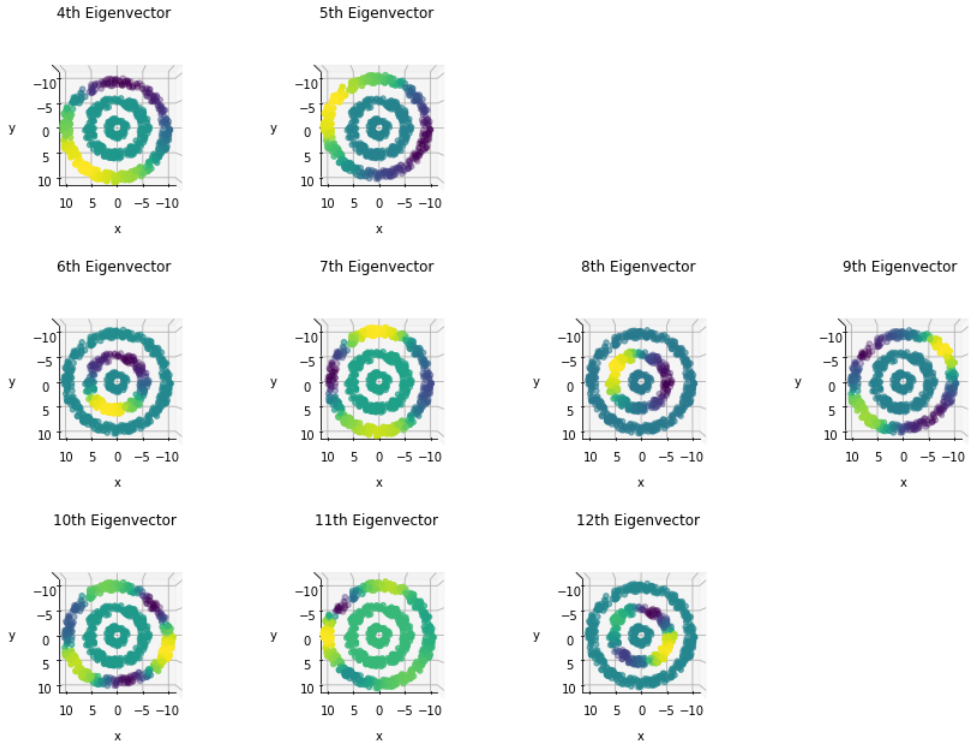


Figure 24: 4th-12th eigenvectors of the Normalized random walk Graph Laplacian vs. Nested Circles data seen from above

References

[Luxburg(2007)] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007. ISSN 0960-3174. doi: 10.1007/s11222-007-9033-z. URL `https://doi.org/10.1007/s11222-007-9033-z`.