Universidad Autónoma de Madrid
Escuela Politécnica Superior
Departamento de Ingeniería Informática

Advanced methods for dimensionality reduction and clustering: Laplacian Eigenmaps and Spectral Clustering

Master's thesis presented to apply for the Master
in Computer Engineering and Telecommunications degree

By
Ángela Fernández Pascual

under the direction of
Julia Díaz García

Madrid, December 13, 2010

Contents

Abstract

Dimensional reduction and clustering are important issues in machine learning. Classical methods do not always provide good results as they do not take into account the explicit form of the differential manifold structure in which our data probably lie. The objective of this work is to study new methods in these disciplines, specifically we study Laplacian Eigenmaps and Spectral Clustering algorithms. These techniques allow us to create an embedding based on the local geometric information of the original data. But these approaches have the inconvenient of not knowing how to treat new data points out-of-sample. With the purpose of clustering this new points we also present an expanded Spectral Clustering method based on the Nyström Formula.

Acknowledgements

The author is kindly supported by the FPI-UAM grant. She has been also partially supported by Spain's TIN 2007-66862 and Cátedra IIC Modelado y Predicción.

Chapter 1

Introduction

In many of the artificial intelligence disciplines, the information retrieval subjects or the data mining areas, we have to solve problems where the sample data sets are in a very high dimension space. Usually, this data is embedded in a differential manifold of low dimension. One of the main issues in machine learning and pattern recognition consists in the development of accurate representations for this type of data whose treatment is usually complex.

The dimensionality reduction general problem has a large history. One of the commonly used approximations for classification is Principal Component Analysis (PCA) [1]. The problem with this kind of methods is that they do not consider the explicit form of the differential manifold structure in which our data probably lie. The objective is to find an application that goes from our original n-dimensional space X to a k-dimensional space Y, with $k \ll n$, in which the local distances are conserved as much as possible.

If we based our study on graphs, we would be able to apply Laplacian Eigenmaps (LE) [2] to our data set. The Graph Laplacian is used to compute a representation on a lower dimension that preserves, in an optimal way, the sample local information. Following the pertinent algorithm, we obtain a representation map that could be considered as a discrete approximation to the continuous map that appears in a natural way from the differential manifold geometry where our set is embedded. But we could only apply this theory in the concrete case in which the manifold data is uniformly sampled.

To obtain the mentioned features, we can also study Locally Linear Embeddings (LLE) [3]. These non-supervised learning algorithms reduce the dimension of a non-linear structure, starting from a linear adjustment and taking advantage of the near neighbors approaches. The advantage of this method is its ability to take the entries to an unique coordinated global system of lower dimension. It is also able to optimize the results without involving local minimums.

If we talk about clustering techniques, we have to start our study with classic algorithms as k-means [4]. The goal to be reached is the identification of groups with similar properties in our data set. With this purpose, we study Spectral Clustering methods because they present implementation and efficiency advantages and also give us good results. The algorithms to be developed are based on the spectral theory, where the clusters are defined by the eigenvalues and eigenvectors of the Laplacian matrices of our data graph. ([5] [6])

Finally, when we have to lead with new points out of our training sample data, using the techniques previously mentioned, we must repeat the whole algorithms with all the points we want to work with. To avoid this, we can adapt our methods by learning eigenfunctions ([7] [8] [9]). We will present the Nyström Formula [10] to reach this objective.

As a summary, in this research work we shall present the theoretical fundaments and some results applying dimensionality reduction methods and clustering techniques. The work is distributed as follows. In chapter 2 we will present a review of classical dimensional reduction and clustering

methods, we mean, PCA and k-means. In chapter 3 we show the advanced dimensional reduction methods studied: LE and LLE. In chapter 4 we put forward the Spectral Clustering theory and their algorithms. In chapter 5 we explain how to adapt the previous Spectral Clustering methods to treat new out-of-sample data applying the Nyström formula. We will show the experimental results of LE and Spectral Clustering methods in chapter 6. Finally we will conclude in chapter 7 that the methods explained are really good approaches to solve dimensional reduction and clustering compared with classical methods. Additionally we include at the end some appendices presenting the notation used in this work and some relevant related theory results needed to understand the study we have carried out.

Chapter 2

Classical dimensional reduction and clustering methods

2.1 Principal Component Analysis (PCA)

2.1.1 Motivation

Principal Component Analysis (PCA) is used to compress the information contained in our data set. The main idea of this technique is to be able to compute the most relevant components or factors from the original features. In this way, we obtain an information summary and a data representation.

The objective is to simplify the data structure transforming the original features into others, named principal components, applying linear combinations of those features. For this purpose, PCA tries to reduce the dimensionality, preserving as much as possible the original randomness (variance) in the high dimensional space.

2.1.2 Algorithm

Let $\mathbf{x}$ be a N-dimensional vector and $\{\varphi_1, \varphi_2, \ldots, \varphi_N\}$ be and orthonormal vectors basis, i.e,

$$\varphi_i \cdot \varphi_j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

Then, we can express $\mathbf{x}$ as a linear combination of these vectors, as follows:

$$\mathbf{x} = \sum_{i=1}^{N} y_i \varphi_i$$

where y_i are the coeficients of the linear combination.

If we want to represent $\mathbf{x}$ only with k $(k < N)$ vectors in the basis, we ignore the components $[y_{k+1}, \ldots, y_N]^T$, obtaining the following expression:

$$\hat{\mathbf{x}}(k) = \sum_{i=1}^{k} y_i \varphi_i.$$

If we minimize y_i with respect to y_j, we obtain [4]

$$y_i = \mathbf{x}^T \varphi_j \quad \text{where } j = 1, \ldots, k.$$

The error made with this $\mathbf{x}$ representation can be calculated as follows:

$$\begin{aligned}\Delta\mathbf{x}(k) &= \mathbf{x} - \hat{\mathbf{x}}(k) \\ &= \sum_{i=1}^{N} y_i\varphi_i - \left(\sum_{i=1}^{k} y_i\varphi_i\right) \\ &= \sum_{i=k+1}^{N} y_i\varphi_i.\end{aligned}$$

The vector $\Delta\mathbf{x}(k)$ is a difference vector, whose module represents the error magnitude. We can quantify the mean error using the average of this quantity square, which represents the mean square error. It is given by the expression:

$$\begin{aligned}\bar{\epsilon}^2(k) &= \mathbb{E}\left[\Delta\mathbf{x}(k)^2\right] \\ &= \mathbb{E}\left[\sum_{i=k+1}^{N}\sum_{j=k+1}^{N}(y_i)(y_j)\varphi_i^T\varphi_j\right] \\ &= \sum_{i=k+1}^{N}\mathbb{E}\left[(y_i)^2\right] \\ &= \sum_{i=k+1}^{N}\varphi_i^T\mathbb{E}[x^Tx]\varphi_i \\ &= \sum_{i=k+1}^{N}\varphi_i^T\Sigma_x\varphi_i. \qquad (2.1.1)\end{aligned}$$

Where, Σ_x is the covariance matrix. In the target equation (2.1.1) we have taken into account that φ_i is an orthonormal basis.

So, our algorithm find the basis vectors φ_i that minimize this mean square error. So, we have to solve the minimization problem

$$\begin{aligned}\min_{\varphi,b} \quad & \bar{\epsilon}^2(k) \qquad (2.1.2)\\ s.t. \quad & \varphi \textit{ orthonormal basis}.\end{aligned}$$

Analitically, we can solve this problem (2.1.2) achiving a closed solution [4]: φ_i are the eigenvectors of the covariance matrix $\Sigma_{\mathbf{x}}$. This means that the mean square error is: $\bar{\epsilon}^2(k) = \sum_{i=k+1}^{N}\lambda_i$, where λ_i are the eigenvalues that correspond to the discarded eigenvectors. So we have to take into account the k eigenvectors that correspond to the highest eigenvalues.

We present a summary of this method in the algorithm (1).

Algorithm 1 PCA algorithm

1: *Input*: $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^n$.
2: Normalize X: $\hat{X} = \frac{X-\mu}{\sigma}$, where μ is X mean and σ its deviation.
3: Compute covariance matrix $\Sigma_{\hat{\mathbf{x}}}$.
4: Compute eigenvalues and eigenvectors of $\Sigma_{\hat{\mathbf{x}}}$.
5: Arrange eigenvalues in decreasing order $\Lambda = \{\lambda_i\}$ and also arrange their corresponding eigenvectors $\varphi = \{\varphi_i\}$.
6: Compute the m principal components PC_i as a linear combination of the original data:

$$PC_i = \varphi^T \hat{\mathbf{x}}_i.$$

7: **return** Principal Components computed.

2.1.3 Advantages and disadvantages

PCA is a classical dimensional reduction algorithm with many advantages:

- PCA studies the features relation, finding feature groups that are highly correlated.
- It is an useful technique for feature selection, for outlier detection or for clustering.
- It works properly when our features present a high correlation because few factors would explain a high part of the total variability.

But it presents an important disadvantage: PCA does not take into account the vector's classes, so it does not look at the classes' separability. This method only rotate the coordenates, changing the data axis in the maximum variance direction. So, there does not exist any guarantee of having good classifying information in the maximum variance axis.

2.2 K-means clustering

2.2.1 Motivation

The problem we must face with in clustering is the identification of groups of data in a multidimensional space. We denominate clusters to the groups that must present small intra-point distances compared with the inter-points distances between clusters. K-means algorithm is a non probabilistic technique for clustering. [4]

Let's formalize this technique. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a data set of N observations of a random n-dimensional euclidean variable $\mathbf{x}$. We suppose that the number k of clusters that we must search is given. To formalize our ideas, we must define some prototypes μ_m associated to each cluster m. We could think about them as the centered representation of the clusters.

The objective will be to look for the association between points and clusters, and also to search a set of vectors $\{\mu_m\}$ such that the squares sum of the distances from each data point to its closest vector μ_m is a minimum.

2.2.2 Algorithm

To explain the k-means algorithm, we are going to introduce a 1-of-k coding scheme. This new parameter will be denoted as $r_{im} \in \{0, 1\}$ with $m = 1, \ldots, k$, that are binary indicators that

describes the cluster to which the data point $\mathbf{x}_i$ is assigned to. I.e., if $\mathbf{x}_i$ is assigned to cluster k, then $\begin{cases} r_{ik} = 1 \\ r_{ij} = 0 \quad j \neq k. \end{cases}$

We must define also our objective function, called distortion measure, as follows:

$$J = \sum_{i=1}^{N} \sum_{m=1}^{k} r_{im} \|\mathbf{x}_i - \mu_m\|^2.$$

So the objective is to find values for $\{r_{im}\}$ and $\{\mu_m\}$ that minimize J. We could minimize them in an iterative procedure. Each iteration involves two successive steps corresponding to optimizations with respect to $\{r_{im}\}$ and $\{\mu_m\}$. We make this with algorithm (2).

Algorithm 2 K-means algorithm

1: *Input*: $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, k.
2: Initial values for $\mu_m, m = 1, \ldots, k$.
3: **while** no convergence **do**
4: **First step:** Minimize J with respect to r_{im}, with μ_m fixed.
5: **Second step:** Minimize J with respect to μ_m, with r_{im} fixed.
6: **end while**
7: **return** Clusters $C_1, \ldots, C_k$.

We can study each step of the algorithm, to simplify their computation:

- **Updating $\mathbf{r_{im}}$.** J is a linear function with respect to r_{im}, so the optimization can be performed easily. The terms that involves different "i" are independent, so we could optimize each one separately. We simply assign the i-th data point to the closest cluster centre. More formally,

$$r_{im} = \begin{cases} 1 & \text{if } m = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

- **Updating $\mu_\mathbf{m}$.** J is quadratic in μ_m, so it can be minimized by setting its derivative with respect to μ_m to zero. This gives us,

$$2 \sum_{i=1}^{N} r_{im}(\mathbf{x}_i - \mu_m) = 0.$$

And we obtain,

$$\mu_m = \frac{\sum_i r_{im} \mathbf{x}_i}{\sum_i r_{im}}.$$

The denominator represents the number of points assigned to the cluster m. So μ_m is the mean of all the points assigned to cluster m.

We repeat the two phases until there is no change in the assignments or the number of iterations is exceeded.

The convergence is assured because each phase reduces the value of J. But it can converge to a local minimum instead of to the global one.

It is usually useful to normalize the data. Normally we standardize the variable such that it has zero mean and unit standard deviation.

A good initialization of μ_m is to make it equal to a random subset of k data points.

So we could rewrite our algorithm as follows in algorithm (3).

Algorithm 3 K-means algorithm

Input: $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, k.
Normalize the sample data: $\hat{X} = \frac{X-\mu}{\sigma}$, where μ is X mean and σ its deviation.
Initial values for $\mu_m = random(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N), m = 1, \ldots, k$.
$iter = 0$
while assignments change && $iter < MAX\_ITER$ **do**
    First step: $r_{im} = \begin{cases} 1 & \text{if } m = \arg\min_j \|\hat{\mathbf{x}}_i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$ for $m = 1, \ldots, k$.
    Second step: $\mu_m = \frac{\sum_i r_{im}\hat{\mathbf{x}}_i}{\sum_i r_{im}}$ for $m = 1, \ldots, k$.
    $iter++$
end while
return Clusters $C_1, \ldots, C_k$.

Testing algorithm

When we recibe a new data point out-of-sample, we do not have to repeat the whole algorithm just explained. Once we have the cluster's centers calculated, we only have to find the nearest center to our new point. This means that we have to compute the distance between the new point and the different cluster's centers and we assign the new example to the cluster corresponding with the minimum distance.

2.2.3 Advantages and disadvantages

K-means is a classical algorithm with many advantages:

- With a large number of variables, K-means is computationally fast.
- K-means may produce tight clusters if they are globular.

But it also presents some disadvantages:

- We have to fix the number of clusters, so to predict what k should be used is difficult.
- This algorithm could be slow because in each updating of $\mathbf{r_{im}}$ it is necessary to compute the Euclidean distance between every prototype vector and every data point. We can modify the algorithm to arrange this problem. [4]
- The use of the Euclidean distance as dissimilarity measure limits the type of data variables that can be considered. Also, it can make the cluster determination non-robust to outliers. A more general method that solves this problem is shown in [4].
- The main problem is that it does not work well with non-globular clusters.

2.3 New approaches

The main problem with classic methods in both disciplines, dimensional reduction and clustering, is that they do not consider the explicit form of the differential manifold structure in which our data probably lie. In practice, this gives rise to some of the principal disadvantages of the exposed algorithms, like the missclasification of non-globular clusters applying k-means.

The objective of this work is to study different applications that go from our original n-dimensional space X to a k-dimensional space Y, $k \ll n$, in which the local distances are conserved as much as possible. And in the case of clustering, apply classical clustering methods in such new space.

For this purpose we are going to present LLE and LE algorithms for dimensionality reduction, and Spectral Clustering as an advanced clustering method.

Chapter 3

Advanced Dimensional Reduction Methods

3.1 Motivation

In machine learning we usually come up against problems where the sample data sets lie in a differential manifold of low dimension embedded in a space of very big dimension. Working in low dimension is always easier and has many computational advantages. So, dimension reduction is usually a very important issue we have to face in learning problems.

We describe the general dimensional reduction problem as follows: Our initial sample is $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, with $\mathbf{x}_i \in \mathbb{R}^n$, and we want to find another set $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, with $\mathbf{y}_i \in \mathbb{R}^k$, where $k \ll n$. In this way, $\mathbf{y}_i$ would represent the points $\mathbf{x}_i$ in our new data space. As just mentioned, we pay special attention to the case $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \in \mathcal{M}$ where $\mathcal{M} \in \mathbb{R}^n$ is a differential manifold of dimension k.

There exist different classical dimensionality reduction methods like Principal Component Analysis (PCA), Multidimensional Scaling (MDS) [11], Self-Organizing Maps (SOM) [12] or Artificial Neural Networks (ANR) [13] approaches. But all these algorithms do not attempt to use properly the manifold structure information.

Advanced dimensional reduction methods as Locally Linear Embedding (LLE) or Laplacian Eigenmaps (LE) try to obtain and use neighborhood information of each point by considering the data as a graph.

In the following sections we are going to present these two techniques. In section 3.2 we will show the algorithm, some possible justifications and advantages and disadvantages of LE. In section 3.3 we will present the algoritm of LLE. To finalize this chapter, we will relate LLE to the Laplacian point of view we applied in LE algorithm.

3.2 Laplacian Eigenmaps (LE)

The Laplacian Eigenmaps method is a very successful recently proposed procedure to reduce dimensionality for semi-supervised learning [2]. To preserve the local information, we build a weighted graph from a set of points $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^n$, with N nodes and links only between neighbors. The aim is to obtain an embedded map, computing the eigenvectors of a Laplacian graph.

Let's state the formal algorithm to apply this method.

3.2.1 Algorithm

Step 1: Constructing the adjacency graph. We are going to define our graph in the following way

$$G \quad = \quad (S = \{\mathbf{x}_i\}, E).$$

where E are the edges of the graph and S the vertices, that coincide with the points of our original sample. The edges will be established between two points when they are near; i.e. $(i, j) \in E$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are near. To decide when two points are neighbors we could use different methods as *ϵ-neighborhood graphs*, *k-nearest neighbor graphs* or *fully connected graphs*, all of them specified in the appendix B.

Step 2: Choosing the weights. To decide the value of w_{ij} we have two options:

- Simple-minded method, where the only possible values for our weights are 0 or 1.

$$w_{ij} \quad = \quad \begin{cases} 1 & \text{if } i, j \text{ are connected.} \\ 0 & \text{if } i, j \text{ are not connected.} \end{cases}$$

  The obvious advantage is that we do not have to fix any parameter, but it gives us less information about the local structure of our data.

- Heat Kernel with parameter $t \in \mathbb{R}$. In this case we will set up the weights in the following way

$$w_{ij} \quad = \quad \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}} & \text{if } i, j \text{ are connected.} \\ 0 & \text{if } i, j \text{ are not connected.} \end{cases}$$

Step 3: Computing eigenmaps. Once we have a connected graph G constructed following steps 1 and 2, we must define the map for the embedded coordinates. For this purpose, we work with the Laplacian L of G.

We define an Unnormalized Graph Laplacian L as follows

$$L = D - W, \tag{3.2.1}$$

where $\mathbf{D}$ is the degree matrix of the similarity matrix W, defined as the diagonal matrix with degrees

$$\mathbf{d_i} = \sum_{j=1}^{N} w_{ij}$$

as its diagonal elements.

We must compute the eigenvalues and eigenfunctions of the generalized eigenvector problem $Lf = \lambda Df$ to find the coordinates in our new space. In the appendix C we can see a Laplacian Graph classification and their properties. L has the special characteristic of being a positive semidefinite matrix (lemma 1), so its eigenvalues are always $\lambda_i \geqslant 0$.

Lemma 1 *The Graph Laplacian L is a positive semidefinite matrix.*

Proof To prove that L is positive semidefinite we check that $f^T L f \geqslant 0$.

$$\begin{aligned} f^T L f &= f^T D f - f^T W f \\ &= \sum_{i=1}^{N} d_i f_i^2 - \sum_{i,j=1}^{N} f_i f_j w_{ij} \\ &= \frac{1}{2}\left(\sum_{i=1}^{N} d_i f_i^2 - 2\sum_{i,j=1}^{N} f_i f_j w_{ij} + \sum_{j=1}^{N} d_j f_j^2\right) \\ &= \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2 \qquad (3.2.2) \\ &\geqslant 0. \end{aligned}$$

In the last equality (3.2.2) we have applied d_i definition (3.2.1) and the square of the subtraction expression. □

Under these conditions, we solve our problem and we obtain the solutions $f_0, f_1, \ldots, f_k, \ldots, f_{n-1}$ ordered by their eigenvalues in an increasing way

$$0 = \lambda_0 < \lambda_1 < \cdots < \lambda_{n-1}.$$

$\lambda = 0$ is always a trivial eigenvalue as, thanks to the normalization, our matrix satisfies:

$$\begin{pmatrix} d_1 - w_1 & \cdots & -w_{1n} \\ \vdots & & \\ -w_{n1} & \cdots & d_n - w_{nn} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}.$$

We are not going to consider the trivial solution $f_0 = \mathbf{1}$, associated to $\lambda_0 = 0$. This eigenfunction,

$$f_0 : \mathbf{x}_i \to (1, \ldots, 1),$$

collapses all the elements of each point onto the real number 1. This gives us, in a trivial way, the minimum distance between points (as it is zero) but we loose all the information.

The rest of eigenvectors selected will be orthogonal to f_0 as L is a positive semidefinite matrix. We take the first k eigenvectors in order to have an embedding from the original space to an euclidean k dimensional space:

$$F : \mathbf{x}_i \quad \to \quad (f_1(\mathbf{x}_i), \ldots, f_k(\mathbf{x}_i))$$

3.2.2 Justification

In this section, we are going to consider three different motivations to obtain this method [2].

Optimal Embeddings

A possible way to introduce the LE algorithm is by the construction of an embedding function that preserves the local information.

First, we have the information organized in a weighted graph $G = (S, E)$, with links between neighbor nodes, as we have explained before. We assume that our graph G is connected and we define an embedding that tries to link points that are close in the original space.

We present this justification with $k = 1$ to simplify notation and explanations. We have the embedding

$$Y : G \rightarrow \mathbb{R}$$

A good map will minimize the objective function defined by the error criterion:

$$J(Y) = \frac{1}{2}\sum_{i,j}(y_i - y_j)^2 w_{ij} \geqslant 0 \tag{3.2.3}$$

It is easy to see that applying the definition of D and L, the problem (3.2.3) can be written as follows:

$$\begin{aligned}
\frac{1}{2}\sum_{i,j}(y_i - y_j)^2 w_{ij} &= \frac{1}{2}\sum_{i,j}(y_i^2 + y_j^2 - 2y_i \cdot y_j) w_{ij} \\
&= \frac{1}{2}\left[\sum_i y_i^2 D_{ii} + \sum_j y_j^2 D_{jj} - 2\sum_{i,j} y_i \cdot y_j \cdot w_{ij}\right] \\
&= \frac{1}{2}\left[Y^T DY + Y^T DY - 2Y^T WY\right] \\
&= \frac{1}{2}\left[2Y^T DY - 2Y^T WY\right] \\
&= \frac{1}{2}\left[2Y^T (D - W)Y\right] \\
&= Y^T LY.
\end{aligned}$$

In other words, to solve (3.2.3) is equivalent to solve the matrix minimization problem

$$\arg\min_Y \quad Y^T LY. \tag{3.2.4}$$

$$s.t. \quad Y^T DY = 1. \tag{3.2.5}$$

We add the restriction (3.2.5) to remove any arbitrary scaling factor and then avoid degenerated solutions. Normally, we apply for this goal the restriction $\|Y\|^2 = 1$. In this case, we use matrix D because it is the natural measure on our graph, as it makes reference to the connections of each vertex in a way such that a big value of d_i means that the vertex x_i is strongly connected and, in consequence, it is more important.

To solve the problem (3.2.4) we can use the Lagrange multipliers. We rewrite the equations,

$$\begin{aligned}
\phi(Y) &= Y^T LY - \lambda(Y^T DY - 1) \\
\nabla\phi(Y) &= LY - \lambda DY = 0
\end{aligned}$$

so the solution is the eigenvector Y corresponding to the minimum eigenvalue λ of L that satisfies

$$LY = \lambda DY. \tag{3.2.6}$$

If we consider the vector $Y = \mathbf{1} = (1, \ldots, 1)$, we could easily see that it corresponds to an eigenvalue 0.

$$\begin{aligned}
L\mathbf{1}_i - \lambda D\mathbf{1}_i &= 0 - \lambda D\mathbf{1} \\
&= 0 \\
&\Leftrightarrow \lambda = 0.
\end{aligned}$$

But this is a trivial solution that collapses all vertices of G into the real number $\mathbf{1}$, as we have seen at the end of the previous section (3.2.1). To eliminate this possible solution we must add an additional constrain:

$$Y^T D\mathbf{1} \quad = \quad 0.$$

To find the optimal embedding we must solve the minimization problem:

$$\begin{aligned} \arg\min_{Y} \qquad & Y^T L Y. \\ s.t. \qquad & \begin{cases} Y^T D Y = 1 \\ Y^T D\mathbf{1} = 0. \end{cases} \end{aligned}$$

We have seen above that to solve this problem with the first restriction is equivalent to search the eigenvalues of the problem (3.2.6). With the second constrain we eliminate the trivial solution associated to the zero eigenvalue. So we have arrived at the same problem that we had to solve in LE.

Now let us generalize this explanation to a k dimensional embedding

$$Y : G \quad \rightarrow \quad \mathbb{R}^k.$$

In this case the embedding is given by the $N \times k$ matrix $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_k]$. We have to minimize the function

$$\sum_{ij} \left\| \mathbf{y}^{(i)} - \mathbf{y}^{(j)} \right\|^2 w_{ij}$$

where $\mathbf{y}^{(i)} = [\mathbf{y}_1(i), \ldots, \mathbf{y}_k(i)]^T$, which is equivalent to minimize the matrix expression

$$\begin{aligned} \arg\min_{Y} \qquad & tr(Y^T L Y) \\ s.t. \qquad & Y^T D Y = 1. \end{aligned}$$

We consider a restriction similar to that of the $k = 1$ case, but now, instead of preventing the collapse onto a point, it prevents the collapse onto a subspace of dimension less than $k - 1$.

We prove this equivalence as we did when $k = 1$.

$$\begin{aligned} \sum_{ij} \left\| \mathbf{y}^{(i)} - \mathbf{y}^{(j)} \right\|^2 w_{ij} &= \sum_{ij} \left\| \left((\mathbf{y}_1^{(i)} - \mathbf{y}_1^{(j)}), \ldots, (\mathbf{y}_k^{(i)} - \mathbf{y}_k^{(j)}) \right) \right\|^2 w_{ij} \\ &= \sum_{ij} \left((\mathbf{y}_1^{(i)} - \mathbf{y}_1^{(j)})^2 + \cdots + (\mathbf{y}_k^{(i)} - \mathbf{y}_k^{(j)})^2 \right) w_{ij} \\ &= \sum_{ij} (\mathbf{y}_1^{(i)} - \mathbf{y}_1^{(j)})^2 w_{ij} + \cdots + \sum_{ij} (\mathbf{y}_k^{(i)} - \mathbf{y}_k^{(j)})^2 w_{ij} \\ &= \mathbf{y}_1^T L \mathbf{y}_1 + \cdots + \mathbf{y}_k^T L \mathbf{y}_k \\ &= Tr(Y^T L Y). \end{aligned}$$

The solution will be given by the eigenvector matrix corresponding to the lowest eigenvalues of the generalized eigenvalue problem $LY = \lambda DY$. We arrive at this conclusion by solving the Lagrange multipliers problem, as we have just explained for $k = 1$.

The Laplace Beltrami Operator

We are going to change now our point of view. As we have explained this chapter motivation, we hope that our data lies in a smooth, compact, k-dimensional manifold $\mathcal{M}$ embedded in the original space $\mathcal{M} \subset \mathbb{R}^n$. In order to work in this context we have to introduce the Laplace Beltrami Operator which is the equivalent on a manifold to compute the Laplacian of a graph (used in the subsection of Optimal Embeddings above).

We can start studying a map from the manifold to a real line

$$f : \mathcal{M} \quad \rightarrow \quad \mathbb{R},$$

where f must be at least twice differentiable with the general goal of sending near points in $\mathcal{M}$ to near points in $\mathbb{R}$. To formalize it, we think about two neighbor points $\mathbf{x}, \mathbf{z} \in \mathcal{M}$ and we study each difference in the new space $|f(\mathbf{z}) - f(\mathbf{x})|$. For this purpose, we consider a geodesic curve C parametrized by length with origin in $\mathbf{x}$, i.e.,

$$\begin{aligned} r &= \text{dist}_{\mathcal{M}}(\mathbf{x}, \mathbf{z}). \\ \mathbf{z} &= C(r). \\ \mathbf{x} &= C(0). \\ f(C(t)) &= g(t). \end{aligned}$$

With this notation, and since $f(\mathbf{x}) = f(C(0)) = g(0)$ and $f(\mathbf{z}) = f(C(r)) = g(r)$, we can rewrite the difference that we want to study as

$$\begin{aligned} f(\mathbf{z}) - f(\mathbf{x}) &= g(r) - g(0) \\ &= \int_0^r g'(t)dt \\ &= \int_0^r \nabla f(C(t)) \cdot C'(t)dt. \end{aligned}$$

Taking absolute values in this difference and using the Schwarz inequality as we do in (3.2.7) we have

$$\begin{aligned} |f(\mathbf{z}) - f(\mathbf{x})| &\leqslant \int_0^r \|\nabla f(C(t))\|\|C'(t)\|dt && (3.2.7) \\ &= \int_0^r \|\nabla f(C(t))\|dt && (3.2.8) \\ &= \int_0^r \|\nabla f(\mathbf{x})\|dt + o(r) && (3.2.9) \\ &\leqslant r\|\nabla f(\mathbf{x})\| + o(r) && (3.2.10) \\ &= \|\mathbf{z} - \mathbf{x}\|\|\nabla f(\mathbf{x})\| + o(\|\mathbf{z} - \mathbf{x}\|). && (3.2.11) \end{aligned}$$

In the equality (3.2.8) we have used that the geodesic curve C is parametrized by length, so $\|C'(t)\| = 1$. In the second equality 3.2.9 we used Taylor's approximation that tells us that

$$\|\nabla f(C(t))\| \quad = \quad \|\nabla f(\mathbf{x})\| + O(t).$$

And for the last equality (3.2.11) we use the definition of distance

$$d_{\mathcal{M}}(\mathbf{x}, \mathbf{z}) \quad = \quad r = \|\mathbf{z} - \mathbf{x}\| + o(\|\mathbf{z} - \mathbf{x}\|).$$

We look now for the best map for preserving local information. In (3.2.10) we see that $\|\nabla f(\mathbf{x})\|$ gives us an approximation of the distance between neighbor points after applying the embedding f. Thus to reach this goal, we must solve the minimization problem

$$\begin{array}{ll} \min\limits_{f} & \int_{\mathcal{M}} \|\nabla f(\mathbf{x})\|^2 \\ s.t & \|f\|_{L^2(\mathcal{M})} = 1, \end{array} \tag{3.2.12}$$

where, in this case, the constrain helps us to remove scaling effects as we will work with unitary vectors.

To relate this problem with the Laplacian of our graph, we prove next that

$$\int_{\mathcal{M}} \|\nabla f\|^2 \;=\; \int_{\mathcal{M}} L(f) f.$$

In fact, by definition, $Lf = -div\nabla(f)$ and applying this equality and the Gauss's Divergence Theorem (see appendix D.1), we can see that

$$\begin{aligned} \int_{\mathcal{M}} < \nabla f, \nabla f > \;&=\; -\int_{\mathcal{M}} div(\nabla(f)) f \\ &=\; -\int_{\mathcal{M}} L(f) f \end{aligned}$$

So, to solve the problem (3.2.12) is equivalent to solve the minimization problem

$$\begin{array}{ll} \min\limits_{f} & -\int_{\mathcal{M}} L(f) f \\ s.t & \|f\|_{L^2(\mathcal{M})} = 1, \end{array} \tag{3.2.13}$$

L is semidefinite positive, so a minimum to this problem (3.2.13) has to be an eigenfunction of L. [2]

As we have made before, we search the optimal embedding that is formed by the first k eigenfunctions that correspond with the lowest eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_k$.

$$\mathbf{x} \;\rightarrow\; (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})).$$

The Heat Kernel and the Choice of Weight Matrix

Other important question that we must justify is the way of searching appropriate weights to construct our graph. We are going to explain a methodology using kernel functions, in this case the Heat Kernel.

The Heat equation is

$$\left(\frac{\partial}{\partial t} + L\right) u \;=\; 0,$$

where $u(\mathbf{x}, t)$ is the heat distribution in time t and $f(\mathbf{x}) = u(\mathbf{x}, 0)$ is the initial heat distribution with $f : \mathcal{M} \rightarrow \mathbb{R}$.

The solution to this equation is given in terms of the Heat kernel H_t as

$$u(\mathbf{x}, t) \;=\; \int_{\mathcal{M}} H_t(\mathbf{x}, \mathbf{y}) f(\mathbf{y}).$$

The $H_t(\mathbf{x}, \mathbf{y})$ function is equivalent to the Green's function (see appendix D.4) and tells us how much heat flows from $\mathbf{y}$ to $\mathbf{x}$ in time t.

Our final objective in this chapter is to find the solution of the LE problem. As we have just seen in the Laplace Beltrami Operator point of view, this problem is equivalent to minimize $Lf(\mathbf{x})$ (3.2.13). We could rewrite it using the Heat Kernel expression explained above.

$$\begin{aligned} Lf(\mathbf{x}) &= Lu(\mathbf{x},0) \\ &= -\frac{\partial}{\partial t}u(\mathbf{x},t)\Big|_{t=0} \\ &= -\frac{\partial}{\partial t}\left[\int_{\mathcal{M}} H_t(\mathbf{x},\mathbf{y})f(\mathbf{y})\right]\Big|_{t=0}, \end{aligned}$$

where we have used that f is the initial heat distribution and we have introduced the Heat Kernel equation and its solution.

In an exponential coordinate system (in which the first order coincides with the local coordinate system given by a tangent plane in $\mathbb{R}^n$), we can compute our Heat Kernel as a gaussian function plus an error term as

$$H_t(\mathbf{x},\mathbf{y}) = (4\pi t)^{-\frac{k}{2}} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{4t}}(\phi(\mathbf{x},\mathbf{y}) + O(t)),$$

where $\phi(\mathbf{x},\mathbf{y})$ is a smooth function with $\phi(\mathbf{x},\mathbf{x}) = 1$. [14]

In the special case when $\mathbf{x}$ and $\mathbf{y}$ are very close and t is small, we can simplify the expression of the Heat Kernel as

$$H_t(\mathbf{x},\mathbf{y}) \approx (4\pi t)^{-\frac{k}{2}} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{4t}}. \tag{3.2.14}$$

Using this concrete expression (3.2.14) for the Heat Kernel, we want to show how to look for the weights of our graph. We must search the weights to ensure that the approximation matrix of the Laplacian manifold is positive semidefinite. We could see [2] that when $t \to 0$ the Heat Kernel is more localized and tends to Dirac's δ-function:

$$\lim_{t\to 0}\int_{\mathcal{M}} H_t(\mathbf{x},\mathbf{y})f(\mathbf{y}) = f(\mathbf{x}).$$

So, if we apply the Heat Kernel equation, considering a small t, we obtain the following expression using the derivative definition in (3.2.15)

$$\begin{aligned} Lf(\mathbf{x}) &= Lu(\mathbf{x},t) \\ &= -\frac{\partial}{\partial s}u(\mathbf{x},s)\Big|_{s=t} \\ &\cong -\lim_{t\to 0}\frac{u(\mathbf{x},t)-u(\mathbf{x},0)}{t} \qquad (3.2.15) \\ &\cong \frac{1}{t}\left(f(\mathbf{x}) - \int_{\mathcal{M}} H(\mathbf{x},\mathbf{y})f(\mathbf{y})d\mathbf{y}\right) \\ &\cong \frac{1}{t}\left(f(\mathbf{x}) - (4\pi t)^{-\frac{k}{2}}\int_{\mathcal{M}} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{4t}} f(\mathbf{y})d\mathbf{y}\right). \qquad (3.2.16) \end{aligned}$$

And, if $\mathbf{x}_1,\ldots,\mathbf{x}_k$ are points in $\mathcal{M}$, we could rewrite the expression (3.2.16) in the following discrete form using only the information of our sample.

$$Lf(\mathbf{x}_i) \cong \frac{1}{t}\left(f(\mathbf{x}_i) - \frac{1}{k}(4\pi t)^{-\frac{k}{2}} \sum_{\substack{\mathbf{x}_j \\ 0<\|\mathbf{x}_i-\mathbf{x}_j\|<\epsilon}} e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{4t}} f(\mathbf{x}_j)\right). \tag{3.2.17}$$

The restriction that appears in the addition is due to the fact that points far away will give us a zero result in the exponential. We are only interested in near points.

If we write $\alpha = \frac{1}{k}(4\pi t)^{-\frac{k}{2}}$ and we use the above expression (3.2.17) with $f = 1$, then $Lf = 0$ and we have

$$\begin{aligned} \frac{1}{\alpha} &= \sum_{\substack{\mathbf{x}_j \\ 0<\|\mathbf{x}_i-\mathbf{x}_j\|<\epsilon}} e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{4t}} \\ \Rightarrow \alpha &= \left(\sum_{\substack{\mathbf{x}_j \\ 0<\|\mathbf{x}_i-\mathbf{x}_j\|<\epsilon}} e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{4t}} \right)^{-1}. \end{aligned}$$

This expression assure us that the weights are normalized and they sum $\frac{1}{t}$.

As our Laplacian expression is

$$Lf(\mathbf{x}_i) = d_i f(\mathbf{x}_i) - \sum_j w_{ij} f(\mathbf{x}_j),$$

comparing it with the Heat Kernel expression at which we have arrived in (3.2.17) and considering $\alpha = 1$, the weights of the Laplacian Graph must take the values

$$w_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{4t}} & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < \epsilon, \\ 0 & \text{otherwise.} \end{cases}$$

3.2.3 Discussion: advantages and disadvantages

LE is an algorithm with many advantages:

- This algorithm is easy to implement. We only have to solve an eigenvalue problem and have to do a few computations afterwards. The search of neighbor points could make our algorithm less efficient but there exist optimal methods as we could see, for example, in [15].
- Also, this algorithm lets us interpret our data in a geometric way. We have seen in the subsection (3.2.2) that we can change the dimension of our data to preserve its local geometry with this method.
- The locality-preserving character of the LE algorithm makes it insensitive to outliers and noise.
- It exhibits stability with respect to the embedding, because this approach is based on the intrinsic geometric structure of the manifold. This means that we will have the same embedding of the same underlying manifold into spaces of very different dimension.

But it is not a perfect algorithm, as it also presents some disadvantages:

- A problem of LE is that if we receive a new sample point, we must repeat the whole algorithm over the new complete sample to reduce its dimension.
- The approximation presented only handles manifolds from which data is sampled uniformly. The problem is that this rarely happens in real applications.
- It is difficult to select k, the reduced dimension, and t, the Heat Kernel parameter, and we do not know an efficient fixed form to choose them. The best way depends of our data.

3.3 Locally Linear Embedding (LLE)

Locally Linear Embedding is a method that preserves neighborhood relations [3]. With this new focus, we eliminate the requirement of distance computation between pairs of points that usually appears in classical approaches. The main idea of the algorithm is the reconstruction of the non-linear global structure from local linear information known in advance.

To formalize this method, assume we have a sample of N real vectors $\mathbf{x}_i$ with $dim(\mathbf{x}_i) = n$. We hope that the points $\mathbf{x}_i$ are located in or near to a smooth low-dimensional manifold. We are going to choose a reconstruction method of the graph knowing some local information. We briefly describe the algorithm below.

3.3.1 Algorithm

Step 1: Selecting the neighbors. First of all, we must obtain the underlying information of our data. We want to find for each $\mathbf{x}_i$ the k nearest neighbors in our dataset: $\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_k^{(i)}$ There exist different methods to reach this goal, like *k-nearest neighbor graph* o *ϵ-neighborhood graph* (Appendix B), that are based on the minimum distance to the main point. An example of this step is shown in figure 3.3.1.

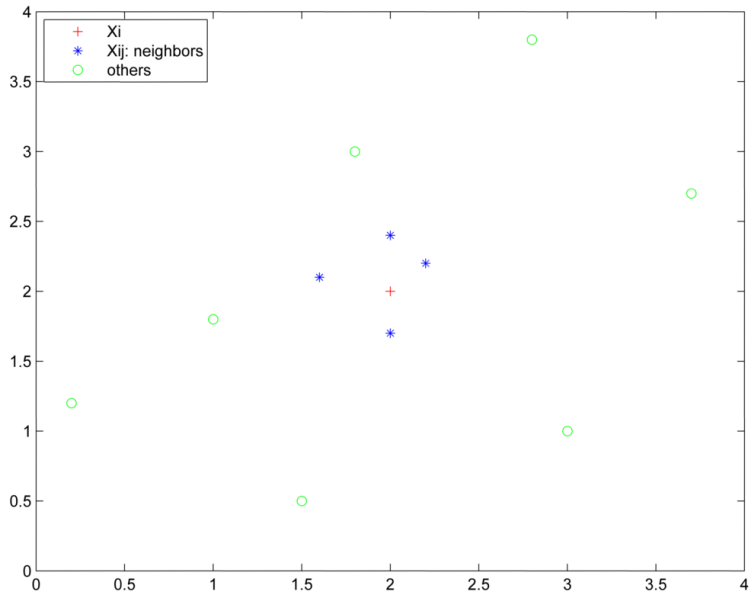

Figure 3.3.1: In the first step of LLE algorithm, we select the k neighbors of each data point. In this case we have used the *k-nearest neighbors* method with k=4.

Step 2: Reconstructing the graph with linear weights. We want to be able to reconstruct each point with the information of its neighbors. The idea is to make an orthogonal projection of the points $\mathbf{x}_i$ in the affine linear span of the nearest data. With this idea, we define the following cost function based on the reconstruction errors.

$$\varepsilon(W) \quad = \quad \sum_i \left| \mathbf{x}_i - \sum_j w_{ij} \mathbf{x}_j^{(i)} \right|^2 .$$

The weights w_{ij} symbolize the contribution of the neighbors $\mathbf{x}_j^{(i)}$ to the reconstruction of the point $\mathbf{x}_i$. Our aim is to find the values of the w_{ij} weights. So, we try to minimize the cost function defined above, subject to the following constrains:

1. $\mathbf{x}_i$ must be reconstructed only by neighbor points. When they are not neighbors, the weight between them must be equal to 0.
2. $\sum_i w_{ij} = 1 \quad \forall i$.
3. An additional constrain could be: $w_{ij} > 0$, but we are not going to impose it in this case.

In figure 3.3.2 we present a picture that represents this step.

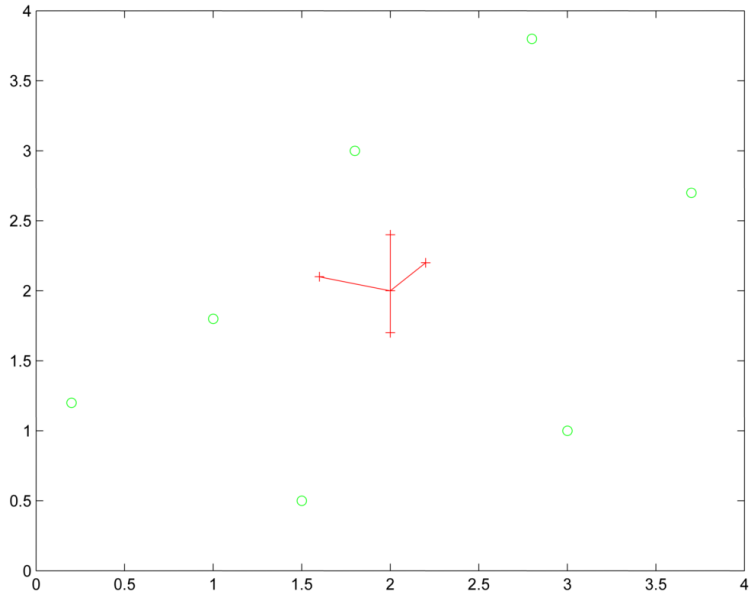

Figure 3.3.2: In the second step, we can see that neighbor points are linked. We assign to each pair a different weight.

Step 3: Mapping to embedded coordinates. The last step of this algorithm consists in the construction of the neighborhood of each point in a new k dimensional space. A good approximation is a linear mapping: a rotation, rescaling or translation of our dataset. The idea is that we have a neighborhood in a n dimensional space and we want to map it to global internal coordinates on the manifold of dimension k ($k \ll n$) that contains our subset. LLE builds a map with these objectives and considers that the reconstruction weights w_{ij} will help to the reconstruction of $\mathbf{x}_i$ in k dimensions. Our map will be:

$$\begin{array}{ll} \text{Mapping} & f : \mathbf{x}_i \to \mathbf{y}_i \\ \text{Cost Function} & \phi(Y) = \sum_i \left| \mathbf{y}_i - \sum_j w_{ij} \mathbf{y}_j^{(i)} \right|^2 \end{array}$$

Finally, we want to find the best reconstructed embeddings Y using the W matrix obtained from the second step. So, we minimize the cost function solving the eigenvalue problem with constrains:

1. $\sum_i \mathbf{y}_i = \vec{0}$, to guarantee that our coordinates are centered at the origin.

2. $\frac{1}{N}\sum_i \mathbf{y}_i \bigotimes \mathbf{y}_i = I$, where $\mathbf{y}_i \bigotimes \mathbf{y}_i$ is the outer product (see appendix A) between embedding points and I is the $N \times N$ identity matrix. We fix this constrain to avoid degenerate solutions, because with this restriction we impose the embedding points to have unit covariance.

We could rewrite our cost functions:

$$\phi(Y) = \sum_{ij} M_{ij}(\mathbf{y}_i \mathbf{y}_j^{(i)})$$

where $M_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_l w_{li}w_{lj}$ and δ_{ij} symbolizes the Kronecker's delta. We can easily prove this statement:

$$\begin{aligned}
\sum_i \left| \mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j^{(i)} \right|^2 &= \sum_i \mathbf{y}_i^2 + \sum_i \left(\sum_j w_{ij}\mathbf{y}_j^{(i)} \right)^2 - 2\sum_i \mathbf{y}_i \sum_j w_{ij}\mathbf{y}_j^{(i)} \\
&= \sum_i \mathbf{y}_i^2 + \sum_i \sum_{l,j} w_{il}w_{ij}\mathbf{y}_l^{(i)}\mathbf{y}_j^{(i)} - 2\sum_i \mathbf{y}_i \sum_j w_{ij}\mathbf{y}_j^{(i)} \\
&= \sum_{i,j} \delta_{ij}\mathbf{y}_i\mathbf{y}_j^{(i)} + \sum_{i,j,l} w_{il}w_{ij}\mathbf{y}_l^{(i)}\mathbf{y}_j^{(i)} - \sum_{i,j} w_{ij}\mathbf{y}_i\mathbf{y}_j^{(i)} - \sum_{i,j} w_{ji}\mathbf{y}_i\mathbf{y}_j^{(i)} \\
&= \sum_{i,j} (\delta_{ij} - w_{ij} - w_{ji} + \sum_l w_{li}w_{lj})(\mathbf{y}_i\mathbf{y}_j^{(i)}) \\
&= \sum_{i,j} M_{ij}(\mathbf{y}_i\mathbf{y}_j^{(i)}).
\end{aligned}$$

If we use matrix notation, $M = E = (I - W)^T(I - W)$, as

$$\begin{aligned}
E &= (I - W)^T(I - W) \\
&= (I - W^T)(I - W) \\
&= I - W - W^T + W^TW
\end{aligned}$$

And, studying each component of this matrix, we aconclude the M_{ij} definition:

$$\begin{aligned}
E_{ij} &= \delta_{ij} - w_{ij} - w_{ji} + \sum_l w_{li}w_{lj} \\
&= M_{ij}.
\end{aligned}$$

With this definition we see that to minimize the cost function $\phi(Y)$ is equivalent to minimize Y^TEY. And this is equivalent to find the eigenvectors of the sparse matrix $E = (I-W)^T(I-W)$, which has the property of being a symmetric positive semidefinite matrix. The optimal embedding is determined by the $k+1$ minor eigenvalues of this matrix. We discard the eigenvector $f_0 \cdot \mathbf{1}$ corresponding to the eigenvalue zero. As we have done in LE, we discard this value because it is a trivial solution. An illustration of the third step is shown in figure 3.3.3.

3.4 LLE Laplacian point of view

The LLE algorithm (explained in section 3.3) can be seen in the same Laplacian terms as the LE algorithm (section 3.2). This point of view only affects to the last step: the embedding of our original coordinates. The matrix

$$E = (I - W)^T(I - W)$$

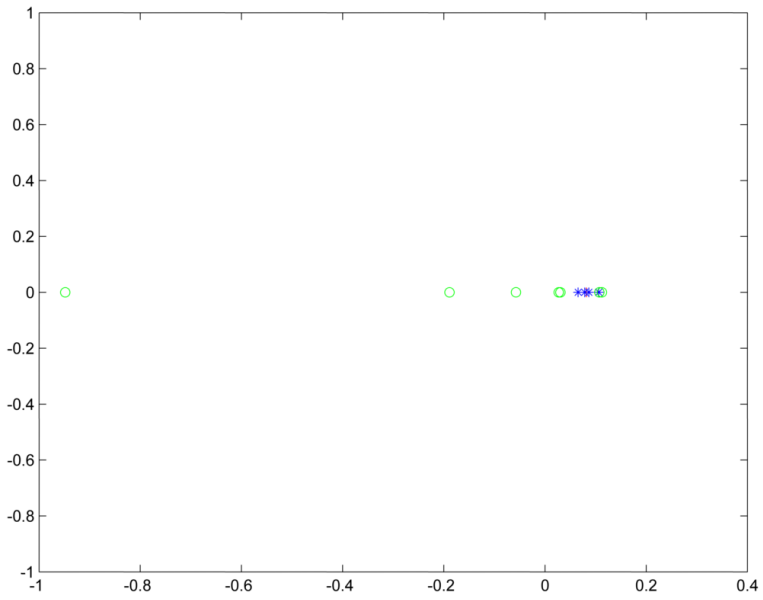

Figure 3.3.3: In the third step, after embedded our coordinates, we can see that the neighbor points in the original space continue being neighbors (we preserve the local information) and the weights of each link are the same computed in step 2.

can be approximately rewritten as

$$Ef \approx \frac{1}{2}L^2 f.$$

We will see now how to justify this affirmation.

Proof 1. We fix a point $\mathbf{x}_i$. We want to prove first the expression

$$[(I-W)f]_i \approx -\frac{1}{2}\sum_j w_{ij}(\mathbf{x}_i - \mathbf{x}_j^{(i)})^T H^{(i)}(\mathbf{x}_i - \mathbf{x}_j^{(i)}),$$

where

- f is a function in our manifold $\mathcal{M}$
- $H^{(i)}$ is the Hessian of f at $\mathbf{x}_i$, $H^{(i)}_{kl} = \frac{\partial^2 f}{\partial \mathbf{x}_l \partial \mathbf{x}_l}$
- $\mathbf{x}_j^{(i)}$ are the neighbors of $\mathbf{x}_i$
- w_{ij} are the weights of each edge (i,j) of our graph.

We consider a coordinate system in the tangent plane $\mathcal{TM}$ centered at $\mathbf{o} = \mathbf{x}_i$. Let's introduce the following notation:

- $\mathbf{v}_j = \mathbf{x}_j^{(i)} - \mathbf{x}_i$
- $\alpha_i = w_{ij}$
- $\mathbf{o} = \mathbf{x}_i = \sum_j \alpha_j \cdot \mathbf{v}_j$ with $\sum_j \alpha_j = 1$.

If f is a smooth function, we can use the second-order Taylor approximation to rewrite the equation

$$f(\mathbf{v}) = f(\mathbf{o}) + \mathbf{v}^T \cdot \nabla f + \frac{1}{2}(\mathbf{v}^T H \mathbf{v}) + o(\|\mathbf{v}\|^2).$$

On the other hand,

$$\begin{aligned}
[(I-W)f]_i &= \left[\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ & & \ddots & \\ 0 & \cdots & 0 & 1 \end{pmatrix} - \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ & & \ddots & \\ w_{n1} & \cdots & & w_{nn} \end{pmatrix} \begin{pmatrix} f(\mathbf{v}_1) \\ \vdots \\ f(\mathbf{v}_n) \end{pmatrix} \right]_i \\
&= \left[\begin{pmatrix} 1-w_{11} & -w_{12} & \cdots & -w_{1n} \\ -w_{21} & 1-w_{22} & \cdots & -w_{2n} \\ & & \ddots & \\ -w_{n1} & \cdots & & 1-w_{nn} \end{pmatrix} \begin{pmatrix} f(\mathbf{v}_1) \\ \vdots \\ f(\mathbf{v}_n) \end{pmatrix} \right]_i \\
&= \left[\begin{pmatrix} f(\mathbf{v}_1) - w_{11}f(\mathbf{v}_1) - \cdots - w_{1n}f(\mathbf{v}_n) \\ \vdots \\ f(\mathbf{v}_i) - w_{i1}f(\mathbf{v}_1) \cdots - w_{in}f(\mathbf{v}_n) \\ \vdots \\ f(\mathbf{v}_n) - w_{n1}f(\mathbf{v}_1) - \cdots - w_{nn}f(\mathbf{v}_n) \end{pmatrix} \right]_i \\
&= f(\mathbf{v}_i) - w_{i1}f(\mathbf{v}_1) \cdots - w_{in}f(\mathbf{v}_n) = f(\mathbf{o}) - \sum_j \alpha_j f(\mathbf{v}_j).
\end{aligned}$$

Using Taylor approximation for $f(\mathbf{v}_j)$ we have

$$\begin{aligned}
f(\mathbf{o}) - \sum_j \alpha_j f(\mathbf{v}_j) &\approx f(\mathbf{o}) - \sum_j \alpha_j (f(\mathbf{o}) + \mathbf{v}_j^T \cdot \nabla f(o) + \frac{1}{2}(\mathbf{v}_j^T H \mathbf{v}_j)) \\
&\approx f(\mathbf{o}) - \sum_j \alpha_j f(\mathbf{o}) - \sum_j \alpha_j \mathbf{v}_j^T \nabla f - \frac{1}{2}\sum_j \alpha_j \mathbf{v}_j^T H \mathbf{v}_j \\
&\approx -\frac{1}{2}\sum_j \alpha_j \mathbf{v}_j^T H \mathbf{v}_j \qquad (3.4.1)
\end{aligned}$$

To arrive to the final expression we have used in the step (3.4.1) that $\sum_j \alpha_j = 1$ and

$$\sum_j \alpha_j \mathbf{v}_j = \mathbf{o}.$$

2. We assume now that $\mathbf{u}_i = \sqrt{\alpha_i} \cdot \mathbf{v}_i$ form an orthonormal basis, i.e., $< \sqrt{\alpha_i} \cdot \mathbf{v}_i | \sqrt{\alpha_i} \cdot \nabla_j >= \delta_{ij}$, and let U be the matrix with $\mathbf{u}_i$ as columns. We have

$$\begin{aligned}
\sum_j w_{ij} \cdot \mathbf{v}_j^T H \mathbf{v}_j &= \sum_j \sqrt{\alpha_j} \cdot \mathbf{v}_j^T H \sqrt{\alpha_j} \cdot \mathbf{v}_j \\
&= \sum_j \mathbf{u}_j^T H \mathbf{u}_j \\
&= U^T H U \\
&= tr(H) \qquad (3.4.2) \\
&= Lf. \qquad (3.4.3)
\end{aligned}$$

The equality (3.4.2) is true because $\mathbf{u}_i$ form an orthonormal basis. The last equality (3.4.3) derives from the usual Laplacian definition:

$$Lf(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}(\mathbf{x}_1, \ldots, \mathbf{x}_n) = tr(H),$$

where H is the Hessian matrix, i.e, $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$.

If $\mathbf{x}$ is a random vector with uniform distribution on every sphere centered at $\mathbf{x}_i$, then the trace of the H matrix is proportional to $\mathbb{E}(\mathbf{v}^T H \mathbf{v})$.

If $(e_1, \ldots, e_n)$ is an orthonormal basis for H with the eigenvalues $\lambda_1, \ldots, \lambda_n$, then using the spectral theorem, we obtain the expression

$$\begin{aligned} \mathbb{E}(\mathbf{v}^T H \mathbf{v}) &= \mathbb{E}\left(\sum_i \lambda_i < \mathbf{v}_i, e_i >^2\right). \qquad (3.4.4) \\ &= \sum_i \lambda_i \mathbb{E}\left(< \mathbf{v}_i, e_i >^2\right) \qquad (3.4.5) \end{aligned}$$

The expectation $\mathbb{E}(< \mathbf{v}, e_i >^2)$ is independent of i so we call it r and we rewrite the expression (3.4.4) in the form

$$\begin{aligned} \mathbb{E}(\mathbf{v}^T H \mathbf{v}) &= r \sum_i \lambda_i \\ &= r \cdot tr(H) \\ &= r \cdot Lf. \end{aligned}$$

3. If we put the step 1 and 2 together, we observe that

$$(I - W)^T (I - W) f \approx \frac{1}{2} L^2 f.$$

LLE tries to minimize the function $f^T (I - W)^T (I - W) f$. We know that to minimize this function is equivalent to look for the eigenvalues of the matrix $(I - W)^T (I - W)$. We have just proved that matrix $(I - W)^T (I - W)$ is equivalent to matrix $\frac{1}{2} L^2$. So we can just look for the eigenvalues of L^2, that coincide with those of L. □

Chapter 4

Spectral Clustering

4.1 Motivation

Clustering is one of the most widely used techniques for data analysis. In recent years, Spectral Clustering has become one of the most popular modern clustering algorithms [5].This new method is easy to implement and can be solved it efficiently. It could seem to be a little bit hard to understand how and why it works. In the end, this method is able to extract the geometry and local information from the data set we are working with in order to, first, reduce dimension and, later, apply any of the existing clustering algorithms that have a good performance in easier subspaces. We have seen this idea in chapter 3 when we have explained LE algorithm.

In the following sections we are going to analyze this method in depth. Firstly, we present the different possible algorithms based on the Laplacian Graph used. In section 4.3 we justify, from different points of view, the use of these methods. In section 4.4 we explain some practical questions on how to apply these algorithms and to chose the different parameters involved. Next, we show some advantages and disadvantages of this method. And finally, in section 4.6, we present a short comparison between Spectral Clustering and LE even though the main difference is the use of k-means in the Spectral Clustering algorithm to classify the data.

4.2 Algorithms

Let us describe some different algorithms to implement Spectral Clustering. The main difference between them is the Laplacian Graph they use to implement the embedding (see appendix C for a classification of these special graphs). All these algorithms present the same structure. They receive as input an arbitrary subset $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, with $\mathbf{x}_i \in \mathbb{R}^n$. All the algorithms use the weights $W = (s_{ij})_{i,j=1,\ldots,N} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1,\ldots,N}$ according to some symmetric non-negative kernel similarity function. The result are always the clusters in the original space. The main trick in all the algorithms presented below is the change of the data $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ from the original space to the points $y_i \in \mathbb{R}^k$ in an euclidean space of smaller dimension. This representation change is very useful, because is easier to work in euclidean spaces, and the embedding does not change the cluster properties of our data.

4.2.1 Unnormalized Spectral Clustering

The first algorithm (algorithm 4) we present is based on the Unnormalized Laplacian Graph to reduce the dimension of the original space. We have defined the Unnormalized Laplacian Graph L in chapter 3 (3.2.1) as follows

$$L = D - W.$$

Algorithm 4 Unnormalized Spectral Clustering

Input: $\mathbf{x}_1, \ldots, \mathbf{x}_n$, k which is the number of clusters we want to obtain.
Construct the similarity graph G, determining the weights by the adjacency matrix W.
Compute the Unnormalized Laplacian Graph L.
Obtain the first k eigenvectors $v_1, \ldots, v_k$ of L.
Let $V \in \mathbb{R}^{N \times k}$ be the matrix with the selected eigenvectors. To obtain the embedding:
for $i = 1$ to N **do**
    Compute $y_i \in \mathbb{R}^k$ as the i-th row of V.
end for
To obtain the clusters $C_1, \ldots, C_k$ of our new points y_i, apply k-means algorithm.
return Clusters $A_1, \ldots, A_k$ where $A_i = \{j | y_j \in C_i\}$.

4.2.2 Normalized Spectral Clustering

In this case, we present two different versions of the Spectral Clustering Algorithm, according to the two different Normalized Laplacian Graphs considered. The first one, the **Symmetric Laplacian**, is given by the expression

$$\begin{aligned} L_{sym} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \\ &= I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}. \end{aligned} \tag{4.2.1}$$

The second one is the called **Random Walk Laplacian**, as it is closely related with random walks.

$$\begin{aligned} L_{rw} &= D^{-1} L \\ &= I - D^{-1} W. \end{aligned}$$

See appendix C for a further explanation.

In the first algorithm (algorithm 5), we realize that the eigenvalues obtained as solution of $Lv = \lambda Dv$ are equivalent to the eigenvalues of the Normalized Random Walk Laplacian Graph.

Lemma 2 *λ is an eigenvalue of L_{rw} with $\mathbf{v}$ its corresponding eigenvector $\Leftrightarrow$ λ and v solves the equation $Lv = \lambda Dv$.*

Proof We assume that λ is an eigenvalue of L_{rw} and v its corresponding eigenvector. Then we have

$$\begin{aligned} \lambda v &= L_{rw} v \\ &= D^{-1} L v \end{aligned}$$

and, therefore, we conclude

$$L v = \lambda D v.$$

□

As we will see in section 4.6, the algorithm (5) is essentially the LE algorithm plus k-means.

Algorithm 5 Normalized Spectral Clustering: Random Walk Laplacian

Input: $\mathbf{x}_1, \ldots, \mathbf{x}_n$, k which is the number of clusters we want to obtain.
Construct the similarity graph G, determining the weights by the adjacency matrix W.
Compute the Unnormalized Laplacian Graph L.
Solve the problem $Lv = \lambda Dv$ and obtain the first k eigenvectors $v_1, \ldots, v_k$.
Let $V \in \mathbb{R}^{N \times k}$ be the matrix with the selected eigenvectors. To obtain the embedding:
for $i = 1$ to N **do**
    Compute $y_i \in \mathbb{R}^k$ as the i-th row of V.
end for
To obtain the clusters $C_1, \ldots, C_k$ of our new points y_i, apply k-means algorithm.
return Clusters $A_1, \ldots, A_k$ where $A_i = \{j | y_j \in C_i\}$.

The second normalized algorithm (algorithm 6) is based on the Normalized Symmetric Laplacian Graph and it includes a normalization step not necessary in the previous versions of our algorithm. The reason will become clear in the justification of this method from a Graph Cut point of view, in the next section (section 4.3).

Algorithm 6 Normalized Spectral Clustering: Symmetric Laplacian

Input: $\mathbf{x}_1, \ldots, \mathbf{x}_n$, k which is the number of clusters we want to obtain.
Construct the similarity graph G, determining the weights by the adjacency matrix W.
Compute the Normalized Symmetric Laplacian Graph $L_{sym} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$.
Obtain the first k eigenvectors $v_1, \ldots, v_k$ of L_{sym}.
Let $V \in \mathbb{R}^{N \times k}$ be the matrix with the selected eigenvectors. Compute a new matrix U, such that each element is equivalent to a normalized element of V, i.e.,

$$u_{ij} = \frac{v_{ij}}{(\sum_l v_{il}^2)^{\frac{1}{2}}}$$

The embedding is obtained from this matrix.
for $i = 1$ to N **do**
    Compute $y_i \in \mathbb{R}^k$ as the i-th row of U.
end for
We obtain the clusters $C_1, \ldots, C_k$ of our new points y_i by applying k-means algorithm.
return Clusters $A_1, \ldots, A_k$ where $A_i = \{j | y_j \in C_i\}$.

4.3 Justification

The main idea of clustering is to group our data in sets with similar properties, that is the purpose of the algorithms previously implemented. In this new section we justify this statement from different points of view.

Graph Cut point of view

A way to justify Spectral Clustering as a data clustering technique is by applying a graph cut point of view. The idea is to construct a partition of the graph such that the edges between groups have low weights but, inside the group, the edges have high weights. To study the possible partitions in our graph, taking into account this definition, we introduce a new measure: the cut of the graph.

Definition 1 *Let $A, B \in S$, where $A \bigcap B = \emptyset$. Then, we define*

$$\mathbf{cut}(\mathbf{A}, \mathbf{B}) = \sum_{\substack{i \in A \\ j \in B}} w_{ij}$$

Looking on the clustering objectives, we want to look for disjoint subsets, so this measure gives us an estimate about the edges that we must eliminate from A in order to have an isolated group according to B.

If G is our similarity graph and W is our adjacency matrix, to find a partition of the graph is the same than to solve the minimization problem of the cut. More generally, we have to find the partition $A_1, \ldots, A_k$ that minimizes

$$cut(A_1, \ldots, A_k) = \sum_{i=1}^{k} cut(A_i, \bar{A}_i).$$

In practice, minimizing this quantity does not give us a good partition of the data. To sort out this problem we could require that the sets of the partition $A_1, \ldots, A_k$ were reasonably large. For this purpose, we define the following objective functions:

$$RatioCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{|A_i|} \tag{4.3.1}$$

$$NCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}. \tag{4.3.2}$$

where $|A|$ is the number of nodes in A and $vol(A) = \sum_{i \in A} d_i$, that represents the size of A.

The $RatioCut$ (4.3.1) is related with the Unnormalized Spectral Clustering. It measures the partition size by the number of vertices on the graph. The $NCut$ (4.3.2) is related with the Normalized Spectral Clustering. It measures the partition size by the weights of its edges.
In both cases, if A_i is very small, the objective function ((4.3.1) or (4.3.2)) has a high value. When we solve the minimization problem of the objective functions, we see that the only difference lie in the expressions

$$\min \left(\sum_{i=1}^{k} \frac{1}{|A_i|} \right) \tag{4.3.3}$$

$$\min \left(\sum_{i=1}^{k} \frac{1}{vol(A_i)} \right). \tag{4.3.4}$$

Both of them achieve the minimum when all clusters have the same $|A_i|$ (4.3.3) or $vol(A_i)$ (4.3.4). This means that we search for balanced clusters solving the problem of non-optimal solutions, but we have now a NP-Hard problem to solve [16]. Spectral Clustering lets us solve relaxed versions of these problems. Let us describe them.

Approximation to RatioCut

First, we are going to study the approximation to the $RatioCut$ with $k = 2$. The main objective is to solve

$$\min_{A \subset S} RatioCut(A, \bar{A})$$

For this purpose, we rewrite the problem. Let us assume $A \subset S$ and consider

$$f = (f_1, \ldots, f_N)^T \in \mathbb{R}^N \text{ with entries } f_i = \begin{cases} \sqrt{\frac{|\bar{A}|}{|A|}} & v_i \in A \\ -\sqrt{\frac{|A|}{|\bar{A}|}} & v_i \in \bar{A}. \end{cases}$$

The objective function can be rewritten using the Normalized Laplacian Graph. We prove that minimize $f^T L f$ is equivalent to minimize our original $RatioCut$ problem. To do so, recall that we proved in chapter 3 (3.2.2) that $f^T L f = \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2$. Then we have,

$$\begin{aligned}
f^T L f &= \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2 \\
&= \sum_{\substack{i\in A \\ j\in \bar{A}}} w_{ij} \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + \sum_{\substack{i\in \bar{A} \\ j\in A}} w_{ij} \left(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 \\
&\quad + \sum_{\substack{i\in A \\ j\in A}} w_{ij} \left(\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|\bar{A}|}{|A|}} \right)^2 + \sum_{\substack{i\in \bar{A} \\ j\in \bar{A}}} w_{ij} \left(-\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|\bar{A}|}{|A|}} \right)^2 \\
&= \sum_{\substack{i\in A \\ j\in \bar{A}}} w_{ij} \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 + \sum_{\substack{i\in \bar{A} \\ j\in A}} w_{ij} \left(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}} \right)^2 \\
&= 2cut(A, \bar{A}) \left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) \\
&= 2cut(A, \bar{A}) \left(\frac{|\bar{A}| + |A|}{|A|} + \frac{|A| + |\bar{A}|}{|\bar{A}|} \right) \\
&= 2(|\bar{A}| + |A|) \left(\frac{cut(A, \bar{A})}{|A|} + \frac{cut(A, \bar{A})}{|\bar{A}|} \right) \\
&= 2|S| RatioCut(A, \bar{A}).
\end{aligned}$$

We can now rewrite the minimization problem that we must solve as

$$\begin{aligned}
\min_{A\subset S} \quad & f^T L f \\
s.t. \quad & f_i = \begin{cases} \sqrt{\frac{|\bar{A}|}{|A|}} & v_i \in A \\ -\sqrt{\frac{|A|}{|\bar{A}|}} & v_i \in \bar{A}. \end{cases}
\end{aligned} \tag{4.3.5}$$

We have discretized the $RatioCut$ equation but it continues being NP-Hard. The problem is that we minimize over A. In order to solve it, we can study some characteristic of the function f

we have just defined. We first notice that $f^T\mathbf{1} = 0$. In fact we have

$$\begin{aligned} f^T\mathbf{1} &= \sum_{i=1}^{N} f_i \\ &= \sum_{i\in A}\sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i\in\bar{A}}\sqrt{\frac{|A|}{|\bar{A}|}} \\ &= |A|\sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}|\sqrt{\frac{|A|}{|\bar{A}|}} \\ &= \sqrt{|A||\bar{A}|} - \sqrt{|\bar{A}||A|} \\ &= 0. \end{aligned}$$

We observe next that $\|f\|^2 = |S|$. So,

$$\begin{aligned} \|f\|^2 &= \sum_{i=1}^{N} f_i^2 = |A|\frac{|\bar{A}|}{|A|} + |\bar{A}|\frac{|A|}{|\bar{A}|} \\ &= |\bar{A}| + |A| = N = |S|. \end{aligned}$$

Problem (4.3.5) is now equivalent to,

$$\begin{aligned} \min_{A\subset S} \quad & f^T L f \\ s.t. \quad & \begin{cases} f \perp \mathbf{1} \\ f_i = \begin{cases} \sqrt{\frac{|\bar{A}|}{|A|}} & v_i \in A \\ -\sqrt{\frac{|A|}{|\bar{A}|}} & v_i \in \bar{A} \end{cases} \\ \|f\| = \sqrt{N}. \end{cases} \end{aligned}$$

We are going to apply a relaxation consisting on the elimination of the restriction over the discrete values of f_i, allowing $f_i \in \mathbb{R}$. So we can rewrite the problem as

$$\begin{aligned} \min_{f\in\mathbb{R}^N} \quad & f^T L f \\ s.t. \quad & \begin{cases} f \perp \mathbf{1} \\ \|f\| = \sqrt{N}. \end{cases} \end{aligned} \tag{4.3.6}$$

Applying the Rayleigh-Ritz theorem (see appendix D.2), the solution is given by the eigenvector f_1, which corresponds to the lowest eigenvalue λ_1 of L not equal to 0. To obtain a partition of the graph, we transform f in a discrete indicator

$$f = \begin{cases} v_i \in A & f_i \geqslant 0 \\ v_i \in \bar{A} & f_i < 0. \end{cases}$$

We have seen in chapter 3 that to solve the problem (4.3.6) is equivalent to find the eigenvectors of the generalized eigenvalue problem $Lf = \lambda Df$. So this is an Spectral Clustering algorithm, as we must solve the same minimization problem. Thus, we obtain a solution resulting of the spectral theory and we discretize the result in the same way.

We can extend this explanation to an arbitrary k in the following way. Let $A_1, \ldots, A_k$ be a partition of S. We define k indicator vectors $h_i = (h_{1i}, \ldots, h_{Ni})^T$ such that

$$h_{ji} = \begin{cases} \frac{1}{\sqrt{|A_i|}} & j \in A_i \\ 0 & otherwise. \end{cases}$$

$H \in \mathbb{R}^{N \times k}$ will be the matrix that contains these indicator vectors h_i on their columns. It is easy to see that this matrix has the property of being an orthonormal matrix ($H^T H = I$).

In this case, our objective function can be rewritten in the following way, remembering that W is a symmetric matrix,

$$\begin{aligned}
h_i^T L h_i &= \sum_{l,m=1}^{N} w_{lm}(h_{li} - h_{mi})^2 \\
&= \sum_{l\in A_i}\sum_{m\in A_i} w_{lm}(h_{li} - h_{mi})^2 + \sum_{l\in A_i}\sum_{m\in \bar{A}_i} w_{lm}(h_{li} - h_{mi})^2 + \sum_{l\in \bar{A}_i}\sum_{m\in A_i} w_{lm}(h_{li} - h_{mi})^2 \\
&\quad + \sum_{l\in \bar{A}_i}\sum_{m\in \bar{A}_i} w_{lm}(h_{li} - h_{mi})^2 \\
&= \sum_{l\in A_i}\sum_{m\in A_i} w_{lm}\left(\frac{1}{\sqrt{|A_i|}} - \frac{1}{\sqrt{|A_i|}}\right)^2 + \sum_{l\in A_i}\sum_{m\in \bar{A}_i} w_{lm}\left(\frac{1}{\sqrt{|A_i|}}\right)^2 + \sum_{l\in \bar{A}_i}\sum_{m\in A_i} w_{lm}\left(-\frac{1}{\sqrt{|A_i|}}\right)^2 \\
&\quad + \sum_{l\in \bar{A}_i}\sum_{m\in \bar{A}_i} w_{lm} \cdot 0 \\
&= 2\sum_{l\in A_i}\sum_{m\in \bar{A}_i} w_{lm}\left(\frac{1}{\sqrt{|A_i|}}\right)^2 \\
&= 2cut(A_i, \bar{A}_i)\frac{1}{|A_i|}.
\end{aligned}$$

In matrix notation,

$$h_i^T L h_i = (H^T L H)_{ii}.$$

So, the $RatioCut$ can be rewritten as

$$\begin{aligned}
RatioCut(A_1, \ldots, A_k) &= \frac{1}{2}\sum_{i=1}^{k} h_i^T L h_i \\
&= \frac{1}{2}\sum_{i=1}^{k} (H^T L H)_{ii} \\
&= \frac{1}{2} Tr(H^T L H).
\end{aligned}$$

The minimization problem to solve now will be

$$\min_{A_1,\cdots,A_k} \quad Tr(H^T L H)$$

$$s.t. \quad \begin{cases} H^T H = I \\ h_{ji} = \begin{cases} \frac{1}{\sqrt{|A_i|}} & j \in A_i \\ 0 & otherwise. \end{cases} \end{cases}$$

In this case, we relax the problem allowing any real entry in matrix H, and we arrive at the new problem

$$\begin{aligned} \min_{H \in \mathbb{R}^{N \times k}} \quad & Tr(H^T L H) \\ s.t. \quad & H^T H = I. \end{aligned}$$

And thanks again to the Rayleigh-Ritz theorem (see appendix D.2), we know that a solution will be a matrix H^* with the first k eigenvectors of L as columns.

This result is equivalent to the matrix V computed in the Unnormalized Spectral Clustering method. Now we can apply k-means over H's rows and we will obtain the same results with both methods.

Approximation to NCut

The approach to this approximation is really similar to the one just explained for the $RatioCut$. We are going to start studying the easiest case, when $k = 2$. First of all we define a cluster indicator vector

$$f_i = \begin{cases} \sqrt{\frac{vol(\bar{A})}{vol(A)}} & v_i \in A \\ -\sqrt{\frac{vol(A)}{vol(\bar{A})}} & v_i \in \bar{A}. \end{cases}$$

This indicator is related to the $NCut$ defined before in (4.3.2). So, we have

$$\begin{aligned} f^T L f &= \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2 \\ &= \sum_{\substack{i \in A \\ j \in \bar{A}}} w_{ij} \left(\sqrt{\frac{vol(\bar{A})}{vol(A)}} + \sqrt{\frac{vol(A)}{vol(\bar{A})}} \right)^2 + \sum_{\substack{i \in \bar{A} \\ j \in A}} w_{ij} \left(-\sqrt{\frac{vol(A)}{vol(\bar{A})}} - \sqrt{\frac{vol(\bar{A})}{vol(A)}} \right)^2 \\ &\quad + \sum_{\substack{i \in A \\ j \in A}} w_{ij} \left(\sqrt{\frac{vol(\bar{A})}{vol(A)}} - \sqrt{\frac{vol(\bar{A})}{vol(A)}} \right)^2 + \sum_{\substack{i \in \bar{A} \\ j \in \bar{A}}} w_{ij} \left(-\sqrt{\frac{vol(A)}{vol(\bar{A})}} + \sqrt{\frac{vol(A)}{vol(\bar{A})}} \right)^2 \\ &= \sum_{\substack{i \in A \\ j \in \bar{A}}} w_{ij} \left(\sqrt{\frac{vol(\bar{A})}{vol(A)}} + \sqrt{\frac{vol(A)}{vol(\bar{A})}} \right)^2 + \sum_{\substack{i \in \bar{A} \\ j \in A}} w_{ij} \left(-\sqrt{\frac{vol(A)}{vol(\bar{A})}} - \sqrt{\frac{vol(\bar{A})}{vol(A)}} \right)^2 \\ &= 2cut(A, \bar{A}) \left(\frac{vol(\bar{A})}{vol(A)} + \frac{vol(A)}{vol(\bar{A})} + 2 \right) \\ &= 2cut(A, \bar{A}) \left(\frac{vol(\bar{A}) + vol(A)}{vol(A)} + \frac{vol(A) + vol(\bar{A})}{vol(\bar{A})} \right) \\ &= 2vol(S) NCut(A, \bar{A}). \end{aligned}$$

We are going to take into account some restrictions associated with f. First we have that

$(Df)^T\mathbf{1} = 0$ as shown below:

$$\begin{aligned}
(Df)^T\mathbf{1} &= \sum_i d_i f_i \\
&= \sum_{i\in A} d_i f_i + \sum_{i\in \bar{A}} d_i f_i \\
&= \sqrt{\frac{vol(\bar{A})}{vol(A)}} \sum_{i\in A} d_i - \sqrt{\frac{vol(A)}{vol(\bar{A})}} \sum_{i\in \bar{A}} d_i \\
&= \sqrt{\frac{vol(\bar{A})}{vol(A)}} \sum_{i\in A}\sum_{j=1}^{N} w_{ij} - \sqrt{\frac{vol(A)}{vol(\bar{A})}} \sum_{i\in \bar{A}}\sum_{j=1}^{N} w_{ij} \\
&= \sqrt{\frac{vol(\bar{A})}{vol(A)}} vol(A) - \sqrt{\frac{vol(A)}{vol(\bar{A})}} vol(\bar{A}) \\
&= \sqrt{vol(\bar{A})vol(A)} - \sqrt{vol(A)vol(\bar{A})} \\
&= 0.
\end{aligned}$$

Then we observe that $f^T Df = vol(S)$, following the next reasoning:

$$\begin{aligned}
f^T Df &= \sum_{i\in A} d_i f_i^2 + \sum_{i\in \bar{A}} d_i f_i^2 \\
&= \sum_{i\in A} d_i \frac{vol(\bar{A})}{vol(A)} + \sum_{i\in \bar{A}} d_i \frac{vol(A)}{vol(\bar{A})} \\
&= \frac{vol(\bar{A})}{vol(A)} \sum_{i\in A}\sum_{j=1}^{N} w_{ij} + \frac{vol(A)}{vol(\bar{A})} \sum_{i\in \bar{A}}\sum_{j=1}^{N} w_{ij} \\
&= vol(\bar{A}) + vol(A) \\
&= vol(S).
\end{aligned}$$

Taking these expressions into consideration, we can now rewrite the minimization $NCut$ problem (4.3.2) as

$$\begin{aligned}
&\min_{A\subset S} && f^T Lf \\
&s.t. && \begin{cases} Df \perp \mathbf{1} \\ f_i = \begin{cases} \sqrt{\frac{vol(\bar{A}}{vol(A)}} & v_i \in A \\ -\sqrt{\frac{vol(A)}{vol(\bar{A})}} & v_i \in \bar{A} \end{cases} \\ f^T Df = vol(S). \end{cases}
\end{aligned}$$

Now we relax the problem, allowing the functions f_i to have any real value. The problem we have to solve in this case is

$$\begin{aligned}
&\min_{f\in\mathbb{R}^N} && f^T Lf \\
&s.t. && \begin{cases} Df \perp \mathbf{1} \\ f^T Df = vol(S). \end{cases}
\end{aligned}$$

Then we substitute next our function f by $g = D^{\frac{1}{2}} f$, and we have

$$\begin{array}{ll} \min\limits_{g \in \mathbb{R}^N} & g^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} g \\ s.t. & \begin{cases} g \perp D^{\frac{1}{2}} \mathbf{1} \\ \|g\|^2 = vol(S). \end{cases} \end{array}$$

We must observe that $D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ corresponds with L_{sym} (4.2.1). Applying as in previous reasonings the Rayleigh-Ritz theorem (see appendix D.2), the solution to this problem is $g = f_1$, that is the first eigenvector of L_{sym} corresponding to the lowest non-zero eigenvalue. If we consider $f = D^{-\frac{1}{2}} g$, we obtain the eigenvectors of L_{rw}, as there exists a direct relation between both Laplacian graphs (symmetric and random walks) (see appendix C). The eigenvectors of L_{rw} are the same that solve the generalized problem $Lv = \lambda Dv$. So, in this case, we also obtain the same results we have achieved with the Normalized Spectral Clustering methods.

We can extend this explanation to an arbitrary k, as we have made before in the $RatioCut$ approximation. So, we define $h_i = (h_{1i}, \ldots, h_{Ni})$ such that

$$h_{ji} = \begin{cases} \frac{1}{\sqrt{vol(A_i)}} & j \in A_i \\ 0 & otherwise. \end{cases}$$

We can see that this function h is related with $NCut$. More precisely, we have

$$\begin{aligned} h_i^T L h_i &= \sum_{l,m=1}^{N} w_{lm}(h_{li} - h_{mi})^2 \\ &= \sum_{\substack{l \in A_i \\ m \in \bar{A}_i}} w_{lm} \left(\frac{1}{\sqrt{vol(A_i)}} \right)^2 + \sum_{\substack{l \in \bar{A}_i \\ m \in A_i}} w_{lm} \left(-\frac{1}{\sqrt{vol(A_i)}} \right)^2 \\ &= \frac{1}{vol(A_i)} \left(\sum_{\substack{l \in A_i \\ m \in \bar{A}_i}} w_{lm} + \sum_{\substack{l \in \bar{A}_i \\ m \in A_i}} w_{lm} \right) \\ &= \frac{2cut(A_i, \bar{A}_i)}{vol(A_i)}. \end{aligned}$$

where the symmetry of W have been used in the last equality.

In matrix notation,

$$\begin{aligned} NCut(A_1, \ldots, A_l) &= \frac{1}{2} \sum_{i=1}^{k} h_i^T L h_i \\ &= \frac{1}{2} \sum_{i=1}^{k} (H^T L H)_{ii} \\ &= \frac{1}{2} Tr(H^T L H). \end{aligned}$$

We show next that $H^T D H = I$. In fact, we have

$$\begin{aligned} h_i^T D h_i &= \sum_l d_l h_{li}^2 \\ &= \sum_{l \in A_i} d_l \frac{1}{vol(A_i)} \\ &= \frac{1}{vol(A_i)} \sum_{l \in A_i} \sum_{m=1}^{N} w_{lm} \\ &= 1. \end{aligned}$$

As a consequence, now we can rewrite our minimization problem in the following way,

$$\begin{aligned} \min_{A_1,\dots,A_k} \quad & Tr(H^T L H) \\ s.t. \quad & \begin{cases} H^T D H = I \\ h_{ji} = \begin{cases} \frac{1}{\sqrt{vol(A_i)}} & j \in A_i \\ 0 & otherwise. \end{cases} \end{cases} \end{aligned}$$

Now we relax the problem allowing any real entry in matrix H. Also, we define the new normalized matrix U as $U = D^{\frac{1}{2}} H$.

$$\begin{aligned} \min_{U \in \mathbb{R}^{N \times k}} \quad & Tr(U^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} U) \\ s.t. \quad & U^T U = I. \end{aligned}$$

And thanks to the Rayleigh-Ritz theorem (see appendix D.2), we know that a solution will be a matrix U^* which has the first k eigenvectors of L_{sym} as columns. We re-substitute $H = D^{-\frac{1}{2}} U$, so H has the first k eigenvectors of L_{rw} that coincide with the ones that solve the generalized problem $Lv = \lambda Dv$.

This result is equivalent to the matrix V computed in the Normalized Spectral Clustering method (eigenvectors of matrix U^* will give us V of a Symmetric Laplacian Graph and eigenvectors of matrix H will give us the matrix V of a Random walk Laplacian Graph). Then, we can apply k-means over H's rows and we will obtain the same results in both cases.

Random Walks point of view

In this subsection we present another way to justify the Spectral Clustering methods based, in this case, on random walks on a graph. When we talk about random walks on a graph, we are talking about a stochastic process that jumps in a random way from a vertex to another. But we can also see a random walk on a graph as a way to search a partition on the graph such that

- The random walk spend more time inside the cluster.
- The random walk rarely jumps to another cluster.

We define the jump probability $P = \{p_{ij}\}$ as

$$\begin{aligned} p_{ij} &= \frac{w_{ij}}{d_i} \\ P &= (p_{ij})_{i,j=1,\dots,N} = D^{-1} W. \end{aligned}$$

If we assume that our graph G is connected and non-bipartite, i.e. its vertices can not be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V (it contains at least an odd-length cycle), then the associated random walk always has a unique stationary distribution, that is the limit distribution of a stochastic process, $\pi = (\pi_1, \ldots, \pi_N)^T$ such that

$$\pi_i = \frac{d_i}{vol(G)}$$

With the previous definition, we show that a graph with a low cut will also have few possibilities to jump between clusters. So there exists a relation between random walks and $NCuts$,

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

So, in this case, λ is an eigenvalue of L_{rw} with eigenvector v only if $(1 - \lambda)$ is an eigenvalue of P with eigenvector v. To describe the properties of our graph, we can use the largest eigenvalues of P and the smallest eigenvalues of L_{rw}.

Proposition 1 NCuts *via transition probabilities*
Let G be a connected and non-bipartite graph. Consider the random walk $(X_t)_{t\geqslant 0}$ that starts in X_0 and has associated the stationary distribution π. For disjoint subsets, $A, B \subset S$, consider

$$P(B|A) = P(X_1 \in B | X_0 \in A).$$

Then,

$$NCut(A, \bar{A}) = P(\bar{A}|A) + P(A|\bar{A}).$$

Proof We start studying the probability of being in A at time 0 and in B at time 1,

$$\begin{aligned} P(X_0 \in A, X_1 \in B) &= \sum_{\substack{i\in A \\ j\in B}} P(X_0 = i, X_1 = j) \\ &= \sum_{\substack{i\in A \\ j\in B}} \pi_i p_{ij} \\ &= \sum_{\substack{i\in A \\ j\in B}} \frac{d_i}{vol(G)} \frac{w_{ij}}{d_i} \\ &= \frac{1}{vol(G)} \sum_{\substack{i\in A \\ j\in B}} w_{ij}. \end{aligned}$$

Using this expressions and the fact that $P(X_0 \in A) = \frac{\text{favourable cases}}{\text{possible cases}}$, we can compute the conditional probability $P(X_1 \in B | X_0 \in A)$.

$$\begin{aligned} P(X_1 \in B | X_0 \in A) &= \frac{P(X_0 \in A, X_1 \in B)}{P(X_0 \in A)} \\ &= \frac{\left(\frac{1}{vol(G)} \sum_{\substack{i\in A \\ j\in B}} w_{ij}\right)}{\left(\frac{vol(A)}{vol(G)}\right)} \\ &= \frac{\sum_{\substack{i\in A \\ j\in B}} w_{ij}}{vol(A)}. \end{aligned}$$

With this result and using the assumption of the proposition, we conclude

$$\begin{cases} P(\bar{A}|A) = P(X_1 \in \bar{A}|X_0 \in A) = \frac{\sum_{i\in A} \sum_{j\in\bar{A}} w_{ij}}{vol(A)} \\ P(A|\bar{A}) = P(X_1 \in A|X_0 \in \bar{A}) = \frac{\sum_{i\in\bar{A}} \sum_{j\in A} w_{ij}}{vol(\bar{A})} \end{cases}$$

And if we add both expressions, we obtain the $NCut$ definition:

$$\begin{aligned} P(\bar{A}|A) + P(A|\bar{A}) &= \sum_{\substack{i\in A \\ j\in\bar{A}}} w_{ij} \left(\frac{1}{vol(A)} + \frac{1}{vol(\bar{A})} \right) \\ &= NCut(A, \bar{A}). \end{aligned}$$

□

This proposition demonstrates that, when we minimize $NCut$, we are looking for a cut graph such that the random walk on it has few transitions from A to $\bar{A}$ and vice versa. Another way to relate random walks and Laplacians Graph is via the commute distance, as follows.

Definition 2 *We define the commute distance or resistance distance* $\mathbf{c_{ij}}$ *as the expected time to travel in the random walk from a vertex* i *to a vertex* j *and coming back.*

This measure decreases if there exist many ways to connect i and j. It is based on the set of shortest paths instead of looking only for the shortest one.

The commute distance is equivalent to the pseudo-inverse of the Laplacian Graph $L^\dagger$. We compute this quantity describing our Laplacian Graph and its pseudo-inverse as

$$\begin{aligned} L &= V\Lambda V^T \\ L^\dagger &= [L^T L]^{-1} L^T = [(V\Lambda^T V^T)(V\Lambda V^T)]^{-1}(V\Lambda^T V^T) \\ &= [V\Lambda^T \Lambda V^T]^{-1}(V\Lambda^T V^T) = V[\Lambda^T\Lambda]^{-1} V^{-1} V\Lambda^T V^T \\ &= V[\Lambda^T\Lambda]^{-1}\Lambda^T V^T = V\Lambda^\dagger V^T \end{aligned}$$

where V is the matrix that contains the eigenvectors of L.

$\Lambda^\dagger$ is a diagonal matrix with entries $\begin{cases} \frac{1}{\lambda_i} & \lambda_i \neq 0 \\ 0 & \lambda_i = 0 \end{cases}$

then

$$l_{ij}^\dagger = \sum_{m=1}^{N} \frac{1}{\lambda_m} v_{im} v_{jm}.$$

Proposition 2 *Let* G *be a connected and undirected graph,* c_{ij} *the commute distance between* i *and* j*, and* $L^\dagger = (l_{ij}^\dagger)_{i,j=1,\dots,N}$ *the pseudo-inverse of* L*. Then,*

$$\begin{aligned} c_{ij} &= vol(G)(l_{ii}^\dagger - 2l_{ij}^\dagger + l_{jj}^\dagger) \\ &= vol(G)(\mathbf{v}_i - \mathbf{v}_j)^T L^\dagger (\mathbf{v}_i - \mathbf{v}_j). \end{aligned}$$

We can find the proof of this proposition in [17].

From this proposition we can conclude that $\sqrt{c_{ij}}$ could be considered an euclidean distance of the graph's vertices, as it is a multiple of the vertices difference. This means that we can construct an embedding $\mathbf{x}_i \in S \to \mathbf{y}_i \in \mathbb{R}^N$ such that the euclidean distance between the points $\mathbf{y}_i$ matched up with the commute distance between the corresponding points on the graph. We make the embedding in the following way:

1. $L^\dagger$ is a positive semidefinite matrix, so it induces an inner product in a subspace of $\mathbb{R}^N$ orthogonal to $\mathbf{1}$, as its eigenvectors are orthogonal.
2. We take now the points $\mathbf{y}_i \in \mathbb{R}^N$ as the rows of $(\Lambda^\dagger)^{\frac{1}{2}}V$.
3. At the end, applying the proposition 2 and constructing the matrix $L^\dagger$we obtain

$$\begin{aligned} < \mathbf{y}_i, \mathbf{y}_j > &= e_i^T L^\dagger e_j \\ \|\mathbf{y}_i - \mathbf{y}_j\|^2 &= c_{ij}. \end{aligned}$$

We can observe that Spectral Clustering does not work exactly in the same way, there exist some important differences:

- In Spectral Clustering the points $\mathbf{x}_i$ of the graph are embedded to the points $\mathbf{y}_i$ formed as the rows of V. In the commute distance, they are embedded to the points $\mathbf{y}_i$ formed as rows of $(\Lambda^\dagger)^{\frac{1}{2}}V$, where V is the matrix formed by the eigenvectors of L.
- With the commute distance, the entries of $\mathbf{y}_i$ are scaled by $\frac{1}{\lambda_i}$.
- To make the embedding, we take the first k columns of V. But in the case of the commute distance we use all the columns of the matrix.

The main idea of this justification is that both methods are similar because they try to construct clusters based on an euclidean distance between $\mathbf{y}_i$ that is equivalent to construct them based on $\mathbf{x}_i$ and their commute distance on the graph.

4.4 Practical questions

In this section we want to synthesize the different choices we must make to apply Spectral Clustering methods.

Constructing the similarity graph

The first step when we apply a Spectral Clustering method is to construct the similarity graph. This involves two different parts.

1. We start selecting our similarity function K_{ij}, a kernel function that will determine the edge weights. We try to have a function that gives us an idea of significant local neighbors. The problem is that there does not exist any rule to decided the best similarity function because of our data, it depends on domain.
2. We construct our graph, computing W. We choose the algorithm to construct our similarity graph based on the input data set. (See appendix B for the different methods explanation). We will summarize here some considerations for this decision:
   - The ϵ-neighborhood graphs will not work well if our data have different scales, i.e., if the distances between the points are different in different regions.
   - The k-nearest neighbor graphs connect points in different scales. But if we are not in the ideal case, it will connect points of different clusters when they are close.
   - The mutual k-nearest neighbors graph connect regions with the same density but they do not connect regions with different densities on it, so it is perfect to detect clusters of different density. When we have data with different scales, they work well, and they do not mix scales, so this algorithm is between the two previous methods.

- The fully connected graphs, usually with a gaussian function, will give us a non sparse similarity matrix.

Computing the eigenvectors

Once we have constructed the graph, we must compute the eigenvectors of the matrix. The k-nearest neighbor and ϵ-neighborhood graphs give us sparse matrices. With this kind of matrix there exist efficient methods to solve eigenvectors in an easy way. These methods have the property of converging faster when the eigengap $\gamma_k = |\lambda_k - \lambda_{k+1}|$ becomes bigger.

Selecting the number of clusters

We are going to present a special technique designed for Spectral Clustering: the eigengap heuristic. This simple method choose a number k of clusters such that the eigenvalues $\lambda_1, \ldots, \lambda_k$ are small but the eigenvalue λ_{k+1} is big.

This method is based on the spectral theory, because the cuts size is related with the first eigenvalues size.

This method will work worst when we have noisy or overlapped clusters. This means that in an ambiguous context, the result of our new technique will give ambiguous results.

Selecting Laplacian Graph type

To choose the best Laplacian Graph to compute Spectral Clustering methods, we have to observe the degree distribution of the similarity graph. If we have a regular graph and all their vertices are more or less big, all the Laplacians present a similar behavior. But if they have different distribution, the different graphs do not work in the same way.

It is better to work with a Normalized Laplacian Graph than with an Unnormalized one. Between L_{rw} and L_{sym} there do not exist significant differences. We explain the reasons of this preference in the following subsections.

Objectives satisfied by each method

Let us present some arguments in favor of the Normalized Spectral Clustering via a partition graph point of view. Let be $k = 2$. We have two objectives when we make clustering:

1. We want that points in different clusters are not similar. So we must minimize the similarity between clusters:
$$\min \sum_{\substack{i \in A \\ j \in \bar{A}}} w_{ij}.$$
2. We also want that points inside the same clusters are similar. So we want to maximize the similarity inside the cluster:
$$\max \sum_{i,j \in A} w_{ij}, \quad \max \sum_{i,j \in \bar{A}} w_{ij}.$$

$RatioCut$ and $NCut$ incorporate the first goal because they take into account the $cut(A, \bar{A})$. If we analyze the second objective, both measures work in a different way. We see that

$$\begin{aligned} \sum_{i,j \in A} w_{ij} &= \sum_{\substack{i \in A \\ j \in A \bigcup \bar{A}}} w_{ij} - \sum_{\substack{i \in A \\ j \in \bar{A}}} w_{ij} \\ &= vol(A) - cut(A, \bar{A}). \end{aligned}$$

So, the $NCut$ implements the second objective for clustering. But the $RatioCut$ does not take into account the similarity intra-class, as it maximizes the $|A_i|$ and $|\bar{A}_i|$. We must remember that the $RatioCut$ is related to Unnormalized Spectral Clustering while $NCut$ is related to Normalized algorithms.

We can conclude that Normalized Spectral Clustering give us better results than the Unnormalized one.

Consistency reasons

We can also justify the preference for Normalized Spectral Clustering algorithms via convergence reasons.

When we work in smooth conditions, both normalized algorithms are consistent from a statistical point of view [18] [19]. This means that if we assume random data following any distribution, with a number of elements which tend to ∞, the result converge and the limit partition is a sensible partition of the subjacent space.

This does not work with unnormalized algorithms. In this case, the algorithm could not converge or it does to trivial solutions. If we want to avoid trivial solutions, we must impose that the eigenvalues λ_i of L are significant under the minimum degree of the graph. Mathematically,

$$\lambda_i \ll \min_{j=1,\ldots,N} d_j \quad i = 1, \ldots, k.$$

This works because $\lambda \geqslant \min d_j$ gives an approach to Dirac's functions [18], i.e., they will be 0 in all coordinates except in one, so we only isolate the point with non-zero eigenvectors for all the vertices.

4.5 Advantages and Disadvantages

Spectral Clustering algorithms present many advantages:

- They are simple algorithms, that can be implemented in an efficient form using standard linear algebra methods. It does no matter the data sets size.

- They do not make any assumption about the cluster's form, it can solve very general problems as intertwined spirals.

- They take into account the geometric information of local data.

These algorithms can be a very powerful tool if we apply them with care.

But they also presents some disadvantages that we must take into account:

- We have to consider several parameters as the selection of a good similarity graph or a convenient number of clusters. We must select them carefully if we want to obtain good results.

- If we want to use these methods with a classification purpose, we must repeat the whole algorithm each time we have a new point in our data set. It does not make a difference between training and testing processes. In next chapter 5 we present a modified algorithm based on the Nyström Formula that lets us learn out-of-sample examples without repeating the process.

4.6 Relation between LE and Spectral Clustering

Although Laplacian Eigenmaps are oriented to dimensionality reduction algorithm and Spectral Clustering is focused on cluster data, they are very related.

Both methods are approaches provided by the eigenvectors of Laplacian Graphs. Specifically, if we want to compare them, we must focus our attention on Normalized Spectral Clustering based on Random Walks (algorithm 5). If we compare both algorithms (see comparative table 4.6.1), we will see that the only difference is that Spectral Clustering makes one step more to cluster the information in the new dimension.

LE algorithm

1: *Input*: $\mathbf{x}_1, \ldots, \mathbf{x}_n$, k is the dimension of our new space.
2: Construct the similarity graph G, determining the weights by the adjacency matrix W.
3: Compute the Unnormalized Laplacian Graph L.
4: Obtain the first k eigenvectors $v_1, \ldots, v_k$ resulting from the solution of the problem $Lv = \lambda Dv$.
5: Let $V \in \mathbb{R}^{N \times k}$ be the matrix with the selected eigenvectors. We obtain the embedding in the following way:
6: **for** $i = 1$ to N **do**
7: Compute $y_i \in \mathbb{R}^k$ as the i-th row of V.
8: **end for**
9: **return** Embedding $\{y_i\}$.

Spectral Clustering algorithm

1: *Input*: $\mathbf{x}_1, \ldots, \mathbf{x}_n$, k is the number of clusters we want to obtain.
2: Construct the similarity graph G, determining the weights by the adjacency matrix W.
3: Compute the Unnormalized Laplacian Graph L.
4: Obtain the first k eigenvectors $v_1, \ldots, v_k$ resulting from the solution of the problem $Lv = \lambda Dv$.
5: Let $V \in \mathbb{R}^{N \times k}$ be the matrix with the selected eigenvectors. We obtain the embedding in the following way:
6: **for** $i = 1$ to N **do**
7: Compute $y_i \in \mathbb{R}^k$ as the i-th row of V.
8: **end for**
9: Obtain the clusters $C_1, \ldots, C_k$ of our new points y_i by applying k-means algorithm.
10: **return** Clusters $A_1, \ldots, A_k$ where $A_i = \{j | y_j \in C_i\}$.

Table 4.6.1: Comparative table between LE and Spectral Clustering

Chapter 5

Out-of-Sample Spectral Clustering: Nyström Formula

5.1 Motivation

Machine learning methods normally have the advantage that they can be applied on new data points without repeating the whole training algorithms. Such characterization is missing in LE dimensional reduction and in Spectral Clustering algorithms. These methods do not provide a function that can be applied to new points.

In this last chapter, the search of a way to extend our algorithm to new out-of-sample points is discussed. Until now, we have presented methods which aim to provide coordinates for the training points. In the following sections we are going to present an algorithm that extends the ones studied to out-of-sample examples. In section 5.2 we explain the Nyström Formula, the way to improve Spectral Clustering that we are going to apply. We shall show the motivation and justification of this formula, that allows us to generalize any adequate kernel eigenfunction. We will also present an adapted algorithm for Spectral Clustering using this technique in section 5.3.

5.2 The Nyström Formula: generalizing kernel eigenfunctions

5.2.1 Introduction

To learn Spectral Clustering, the first problem we face with is the metric selection. The choice of the metric is one of the key issues of our algorithm. A first attempt to develop Spectral Clustering methods, robust to irrelevant features, could be the learning of the similarity matrix [20]. The problem of this kind of methods is that, even though we save the computation of W for each new point and we eliminate the embedding part, they are less intuitive than the algortihms presented in the previous chapter and they have the disadvantage that we must know a good classification in clusters of the sample data in advance.

Because of this, we look for another method that, working over the algorithms explained in chapter 4, can learn the embedding function and classify a new point on each corresponding cluster without repeating the whole process. This algorithm is based on the Nyström Formula.

The Nyström Formula [7] is a general method for learning kernel eigenfunctions, based on the prediction of the eigenvector value of a new data point and on the convergence of the eigenvalues of a matrix when the number of examples of our data set grows.

The purpose of this section is to addapt this general theory to our Spectral Clustering method, such that we can add new examples without computing again the eigenvalues and eigenvectors of our graph.

We are going to work in a Hilbert's space $\mathcal{H}_p$, i.e, we work in a space of continuous functions f with an associated inner product $< f, g >$ defined as

$$< f, g >= \int f(x)g(x)p(x)dx,$$

where $p(x)$ is the probability density in the input space.

The norm defined by an inner product is the real-valued function

$$\|f\| = \sqrt{< f, f >},$$

and in a Hilbert space $\mathcal{H}_p$, the functions $f$are square integrables, i.e.,

$$\int f^2(x)p(x)dx < \infty.$$

The distance between two functions f,g in $\mathcal{H}_p$ is also defined in terms of the norm by

$$d(f, g) = \|f - g\| = \sqrt{< f - g, f - g >}.$$

and according to this distance concept we define equivalence classes in the Hilbert space,

$$f, g \text{ in the same class } \Leftrightarrow \int (f(x) - g(x))^2 p(x)dx = 0.$$

As we have seen before (subsection 3.2.2), we can express the weight matrix W of our graph using a kernel function [21] [22]. Associated to this kernel we have a linear operator on functions defined by the integral

$$[Gf](\mathbf{x}) = \int K(\mathbf{x}, \mathbf{y})f(\mathbf{y})p(\mathbf{y})d\mathbf{y}.$$

Since G is a linear operator, we can talk about its eigenvalues λ_k and eigenfunctions ϕ_k. $K(\mathbf{x}, \mathbf{y})$ is a positive semidefinite Kernel function, i.e, $\int f(z)Gf(z)p(z)dz \geqslant 0 \quad \forall f$, so $\lambda_k \geqslant 0$ and ϕ_k is an orthonormal basis of the $Im(G)$. Then, we can apply the Mercer's theorem (see appendix D.5) and rewrite our kernel with respect to an orthonormal basis of eigenfunctions in the following way:

$$K(\mathbf{x}, \mathbf{y}) = \sum_{j=i}^{\infty} \lambda_j \phi_j(\mathbf{x})\phi_j(\mathbf{y}). \tag{5.2.1}$$

We see that ϕ_k is an eigenvector of G, with eigenvalue λ_k ($G\phi_k = \lambda_k\phi_k$) using the expression (5.2.1) in the following way

$$\begin{aligned} [G\phi_k](\mathbf{x}) &= \int K(\mathbf{x}, \mathbf{y})\phi_k(\mathbf{y})p(\mathbf{y})d\mathbf{y} \\ &= \sum_{j=i}^{\infty} \lambda_j \phi_j(\mathbf{x}) \int \phi_j(\mathbf{y})\phi_k(\mathbf{y})p(\mathbf{y})d\mathbf{y} \\ &= \lambda_k\phi_k(\mathbf{x}). \end{aligned} \tag{5.2.2}$$

In the third equality (5.2.2) we apply that the eigenfunctions ϕ_i form an orthonormal basis, so $\int \phi_j(\mathbf{y})\phi_k(\mathbf{y})p(\mathbf{y})d\mathbf{y} = \delta_{jk}$, and the summatory only takes a non-zero value when $j = k$.

But we have many unknown variables in this expression: we do not know neither the data-generating density function $p(\mathbf{x})$ to determine G nor the eigenfunctions ϕ_k with their corresponding eigenvalues λ_k.

To estimate them, we discretize our linear operators, using an approximation by a lower-dimensional linear model G^n. A good approximation is given by the eigenfunctions of the kernel correponding to n eigenfunctions of G which form a basis. To approximate this eigenfunction equation (5.2.2), given an iid sample $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ from $p(x)$, we replace the integral over $p(x)$ by an empirical average to obtain

$$.[G^n \phi_k^n](\mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} K^n(\mathbf{y}, \mathbf{x}_i) \phi_k^n(\mathbf{x}_i) \approx \lambda_k \phi_k(\mathbf{y}). \tag{5.2.3}$$

Our new data-dependent bounded kernel K^n are also semidefinite positive in the limit:

$$\begin{aligned} \int f(\mathbf{x}) G^n f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} &= \frac{1}{n} \sum_{i=1}^{n} \int f(\mathbf{x}) K(\mathbf{x}, \mathbf{x}_i) f(\mathbf{x}_i) p(\mathbf{x}) d\mathbf{x} \\ &= \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i) G f(\mathbf{x}_i) \\ &\to \int f(\mathbf{x}) G f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \geqslant 0. \end{aligned}$$

Notice that, if we denote ϕ_j^n as the eigenfunctions of G^n,

$$\begin{aligned} G^n \phi_j^n(\mathbf{x}) &= \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{n} \lambda_k^n \phi_k^n(\mathbf{x}) \phi_k^n(\mathbf{x}_i) \phi_j^n(\mathbf{x}_i) \\ &= \sum_{k=1}^{n} \lambda_k^n \phi_k^n(\mathbf{x}) \left(\frac{1}{n} \sum_{i=1}^{n} \phi_k^n(\mathbf{x}_i) \phi_j^n(\mathbf{x}_i) \right) \\ &= \lambda_j^n \phi_j^n(\mathbf{x}). \end{aligned} \tag{5.2.4}$$

where we have applied in the last equality that ϕ_i^n are eigenfunctions of G^n, so $\phi_k^n(\mathbf{x}_i)\phi_j^n(\mathbf{x}_i) = \delta_{kj}$.

So, G^n has eigenfunctions ϕ_k^n associated to eigenvalues λ_k^n, i.e, G^n satisfy

$$G^n \phi_k^n(\mathbf{x}) = \lambda_k^n \phi_k^n(\mathbf{x}). \tag{5.2.5}$$

To compute the eigenfunctions of G^n is not an easy task. This motivates to study the matrix eigenproblem

$$K^n V^n = V^n \Lambda^n, \tag{5.2.6}$$

where,

- K^n is the $n \times n$ Gram matrix, i.e. its entries are given by $K_{ij}^n = K(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j = 1, \ldots, n$.
- $V^n \in \mathbb{R}^{n \times n}$ is an eigenvectors v_{ij}^n matrix of K^n.
- Λ^n is a diagonal eigenvalues matrix with entries $\ell_1, \ldots, \ell_n$ of K^n.

We will see in the justification below (subsection 5.2.2) that we obtain the following expressions for some eigenfunctions of G^n:

$$\begin{aligned} \phi_i^n(\mathbf{x}_j) &= \sqrt{n} v_{ji}^n \\ \lambda_i^n &= \frac{\ell_i}{n}. \end{aligned}$$

With this approximation we define the Nyström Formula, that let us obtain the ith eigenfunction of G^n

$$\phi_i^n(\mathbf{x}) = \frac{\sqrt{n}}{\ell_i} \sum_{j=1}^{n} v_{ji}^n K^n(\mathbf{x}, \mathbf{x}_j) \quad i = 1, \cdots, k$$

Our hypothesis is that the value of the eigenfunctions of G^n in the train examples will convergence to the eigenfunctions of G and they will correspond with the spectral embeddings.

5.2.2 Justification

In this subsection, we are going to formalize the ideas presented above.

Our first purpose is to demonstrate that the Nyström Formula really gives us an expression of the eigenfunctions of G^n in terms of the eigenvalues and eigenvectors of K^n.

Proposition 3 *Let λ_k^n, ϕ_k^n be the eigenvalues and eigenfunctions of G^n, and ℓ_k, v_k^n be the eigenvalues and eigenvectors of K^n. If we define a function $\varphi_k^n = \varphi_k^n(\mathbf{x}) = \frac{\sqrt{n}}{\ell_k} \sum_{j=1}^{n} K^n(\mathbf{x}, \mathbf{x}_j) v_{kj}^n$, then $\lambda_k^n = \frac{\ell_k}{n}$ and $\phi_k^n(\mathbf{x}_i) = \sqrt{n} v_{ki}^n$.*

Proof If ϕ_k^n is an eigenfunction of G^n, it satisfies, according to the eigenproblem (5.2.5), that

$$\begin{aligned} \phi_k^n(\mathbf{x}) &= \frac{1}{\lambda_k^n} G^n \phi_k^n(\mathbf{x}) \\ &= \frac{1}{\lambda_k^n} \frac{1}{n} \sum_{j=1}^{n} K(\mathbf{x}, \mathbf{x}_j) \phi_k^n(\mathbf{x}_j). \end{aligned}$$

The hypothesis tell us that

$$\varphi_k^n(\mathbf{x}) = \frac{\sqrt{n}}{\ell_k} \sum_{j=1}^{n} K(\mathbf{x}, \mathbf{x}_j) v_{kj}^n,$$

with $\varphi_k^n(\mathbf{x}_i) = \sqrt{n} v_{ki}^n$. This is true because

$$\begin{aligned} \varphi_k^n(\mathbf{x}_i) &= \frac{\sqrt{n}}{\ell_k} \sum_{j=1}^{n} K^n(\mathbf{x}_i, \mathbf{x}_j) v_{kj}^n \\ &= \frac{\sqrt{n}\ell_k}{\ell_k} v_{ki}^n \\ &= \sqrt{n} v_{ki}^n. \end{aligned}$$

In fact,

$$\begin{aligned} G^n \varphi_k^n(\mathbf{x}) &= \frac{1}{n} \sum_{j=1}^{n} K(\mathbf{x}, \mathbf{x}_j) \varphi_k^n(\mathbf{x}_j) \\ &= \frac{1}{n} \sum_{j=1}^{n} K(\mathbf{x}, \mathbf{x}_j) \sqrt{n} v_{kj}^n \\ &= \frac{1}{\sqrt{n}} \sum_{j=1}^{n} K(\mathbf{x}, \mathbf{x}_j) v_{kj}^n \\ &= \frac{\ell_k}{n} \frac{\sqrt{n}}{\ell_k} \sum_{j} K(\mathbf{x}, \mathbf{x}_j) v_{kj}^n \\ &= \frac{\ell_k}{n} \varphi_k^n(\mathbf{x}). \end{aligned}$$

So, $\varphi_k^n(\mathbf{x})$ is an eigenfunction of G^n with eigenvalue $\frac{\ell_k}{n}$, i.e

$$\begin{aligned} \lambda_k^n &= \frac{\ell_k}{n} \\ \phi_k^n(\mathbf{x}_i) &= \sqrt{n} v_{ki}^n. \end{aligned}$$

□

We conclude from this proposition that the Nyström Formula generalizes the spectral embedding out of the training set, because we can associate a new point with its corresponding eigenvalue and eigenvector. We choose this generalization instead of any other because it guarantees convergence as n grows and we have interest in being near the asymptotic embedding.

Lemma 3 *Assume that K^n is bounded $|K^n(\mathbf{x},\mathbf{y})| < c$, independently of n. Then, $\phi_m^n(\mathbf{x})$ is bounded by the quantity $\frac{c}{|\lambda_m^n|}$.*

Proof Taking into account the assumption, we have

$$\begin{aligned} |\phi_m^n(\mathbf{x})| &= \left| \frac{1}{n\lambda_m^n} \sum_{i=1}^n \phi_m^n(\mathbf{x}_i) K^n(\mathbf{x},\mathbf{x}_i) \right| \\ &\leqslant \frac{1}{n|\lambda_m^n|} \sum_{i=1}^n |\phi_m^n(\mathbf{x}_i)||K^n(\mathbf{x},\mathbf{x}_i)| \\ &\leqslant \frac{c}{n|\lambda_m^n|} \sum_{i=1}^n |\phi_m^n(\mathbf{x}_i)| \qquad (5.2.7) \\ &\leqslant \frac{c}{n|\lambda_m^n|} \sum_{i=1}^n \sqrt{n}|v_{im}^n| \qquad (5.2.8) \\ &\leqslant \frac{c}{\sqrt{n}|\lambda_m^n|}\sqrt{n} \qquad (5.2.9) \\ &\leqslant \frac{c}{|\lambda_m^n|}. \end{aligned}$$

We have applied the bound on K^n in the inequality (5.2.7). In (5.2.8) we have applied the proposition 3. And for inequality (5.2.9), we have used that v_{im}^n are orthonormal vectors and by the Cauchy-Schwarz inequality,

$$\sum |v_{im}^n| \leq \left(\sum 1\right)^{\frac{1}{2}} \left(\sum (v_{im}^n)^2\right)^{\frac{1}{2}} \leq \sqrt{n}.$$

□

Definition 3 *We say that a function F_n converge uniformly in probability to another function F when*

$$\forall \epsilon > 0 \quad \lim_{n\to\infty} P(\sup_{x,y} |F_n(x,y) - F(x,y)| \geqslant \epsilon) = 0$$

Proposition 4 *Let us assume that*

1. *The data-dependent bounded kernel K^n converges uniformly in probability toward K.*
2. *The eigenvalues and eigenvectors of the Gram matrix K^n converge.*

3. *The eigenfunctions ϕ_m^n, with $m = 1, \ldots, k$ of G^n, associated to non-zero eigenvalues, converge uniformly in probability to a function ϕ_m^∞.*

Then, ϕ_m^∞ are the eigenfunctions of G.

Proof 1. First of all we take into account some assumptions.
Let ϕ_m^n converge in probability toward ϕ_m^∞. In fact,

$$\forall \epsilon > 0 \quad \lim_{n\to\infty} P(\sup_{\mathbf{x}} |\phi_m^n(\mathbf{x}) - \phi_m^\infty(\mathbf{x})| \geqslant \epsilon) = 0, \tag{5.2.10}$$

Let K^n converge in probability toward K, which is a non-random kernel. Then,

$$\forall \epsilon > 0 \quad \lim_{n\to\infty} P(\sup_{\mathbf{x},\mathbf{y}} |K^n(\mathbf{x},\mathbf{y}) - K(\mathbf{x},\mathbf{y})| \geqslant \epsilon) = 0. \tag{5.2.11}$$

2. Applying lemma 3, we know that $\phi_m^n(\mathbf{x})$ is bounded by $\frac{c}{|\lambda_m^n|}$. We can apply the Koltchinskii and Giné theorem [23], that tells us that the eigenvalues of G^n converge to the ones of G ($\lambda_m^n \to \lambda_m$).
With these results plus the assumption of $\phi_m^\infty(\mathbf{x})$ convergence (5.2.10), we arrive to

$$|\phi_m^\infty| \leqslant \frac{c}{|\lambda_m|}.$$

3. We introduce now ϕ_m^∞ in the Nyström Formula:

$$\begin{aligned}
\phi_m^n(\mathbf{x}) &= \frac{1}{n\lambda_m^n}\sum_{i=1}^n \phi_m^n(\mathbf{x}_i)K^n(\mathbf{x},\mathbf{x}_i) \\
&= \frac{1}{n\lambda_m}\sum_{i=1}^n \phi_m^\infty(\mathbf{x}_i)K(\mathbf{x},\mathbf{x}_i) \\
&\quad + \frac{\lambda_m - \lambda_m^n}{n\lambda_m^n\lambda_m}\sum_{i=1}^n \phi_m^\infty(\mathbf{x}_i)K(\mathbf{x},\mathbf{x}_i) \\
&\quad + \frac{1}{n\lambda_m^n}\sum_{i=1}^n \phi_m^\infty(\mathbf{x}_i)[K^n(\mathbf{x},\mathbf{x}_i) - K(\mathbf{x},\mathbf{x}_i)] \\
&\quad + \frac{1}{n\lambda_m^n}\sum_{i=1}^n K^n(\mathbf{x},\mathbf{x}_i)[\phi_m^n(\mathbf{x}_i) - \phi_m^\infty(\mathbf{x}_i)].
\end{aligned} \tag{5.2.12}$$

We can substract $\frac{1}{\lambda_m}\int \phi_m^\infty(\mathbf{y})K(\mathbf{x},\mathbf{y})p(\mathbf{y})d\mathbf{y}$ in the equation (5.2.12) obtaining

$$\begin{aligned}
&\left|\phi_m^n(\mathbf{x}) - \frac{1}{\lambda_m}\int \phi_m^\infty(\mathbf{y})K(\mathbf{x},\mathbf{y})p(\mathbf{y})d\mathbf{y}\right| \\
\leqslant\ &\left|\frac{1}{n\lambda_m}\sum_{i=1}^n \phi_m^\infty(\mathbf{x}_i)K(\mathbf{x},\mathbf{x}_i) - \frac{1}{\lambda_m}\int \phi_m^\infty(\mathbf{y})K(\mathbf{x},\mathbf{y})p(\mathbf{y})d\mathbf{y}\right| \\
&+ \left|\frac{\lambda_m - \lambda_m^n}{n\lambda_m^n\lambda_m}\sum_{i=1}^n \phi_m^\infty(\mathbf{x}_i)K(\mathbf{x},\mathbf{x}_i)\right| \\
&+ \left|\frac{1}{n\lambda_m^n}\sum_{i=1}^n \phi_m^\infty(\mathbf{x}_i)[K^n(\mathbf{x},\mathbf{x}_i) - K(\mathbf{x},\mathbf{x}_i)]\right| \\
&+ \left|\frac{1}{n\lambda_m^n}\sum_{i=1}^n K^n(\mathbf{x},\mathbf{x}_i)[\phi_m^n(\mathbf{x}_i) - \phi_m^\infty(\mathbf{x}_i)]\right| \\
\leqslant\ &A_n + B_n + C_n + D_n.
\end{aligned}$$

Analyzing these four therms separately we can see that they all tend toward 0.

- $B_n \to 0$. This is true because of the convergence of λ_m^n and the bound of $\phi_m^\infty \leqslant C_1$ and $K \leqslant C_2$.

$$\left|\frac{\lambda_m - \lambda_m^n}{n\lambda_m^n\lambda_m}\sum_{i=1}^{n}\phi_m^\infty(\mathbf{x}_i)K(\mathbf{x},\mathbf{x}_i)\right| \leqslant \left|\frac{\lambda_m - \lambda_m^n}{n\lambda_m^n\lambda_m}\sum_{i=1}^{n}C_1C_2\right|$$
$$\to \left|\frac{\lambda_m - \lambda_m}{n\lambda_m^2}\sum_{i=1}^{n}C_1C_2\right| \to 0.$$

- $C_n \to 0$ in probability. Considering the statement (5.2.11) and the bound of $\phi_m^\infty \leqslant C_1$, we arrive at

$$\left|\frac{1}{n\lambda_m^n}\sum_{i=1}^{n}\phi_m^\infty(\mathbf{x}_i)[K^n(\mathbf{x},\mathbf{x}_i) - K(\mathbf{x},\mathbf{x}_i)]\right| \leqslant \left|\frac{1}{n\lambda_m^n}\sum_{i=1}^{n}C_1[K^n(\mathbf{x},\mathbf{x}_i) - K(\mathbf{x},\mathbf{x}_i)]\right|$$
$$\to \left|\frac{1}{n\lambda_m^n}\sum_{i=1}^{n}C_1[K(\mathbf{x},\mathbf{x}_i) - K(\mathbf{x},\mathbf{x}_i)]\right| \to 0.$$

- $D_n \to 0$ in probability as, thanks to the statement (5.2.10) and the bound of $K^n \leqslant C_3$, we obtain:

$$\left|\frac{1}{n\lambda_m^n}\sum_{i=1}^{n}K^n(\mathbf{x},\mathbf{x}_i)[\phi_m^n(\mathbf{x}_i) - \phi_m^\infty(\mathbf{x}_i)]\right| \leqslant \left|\frac{1}{n\lambda_m^n}\sum_{i=1}^{n}C_3[\phi_m^n(\mathbf{x}_i) - \phi_m^\infty(\mathbf{x}_i)]\right|$$
$$\to \left|\frac{1}{n\lambda_m^n}\sum_{i=1}^{n}C_3[\phi_m^\infty(\mathbf{x}_i) - \phi_m^\infty(\mathbf{x}_i)]\right| \to 0.$$

- If we apply the law of large numbers, A_n also converges to 0 in probability.

Then,

$$\phi_m^n(\mathbf{x}) \to \frac{1}{\lambda_m}\int \phi_m^\infty(\mathbf{y})K(\mathbf{x},\mathbf{y})p(\mathbf{y})d\mathbf{y} \quad \text{in probability}$$

and remembering that

$$\phi_m^n(\mathbf{x}) \to \phi_m^\infty(\mathbf{x}),$$

we obtain,

$$\lambda_m\phi_m^\infty(\mathbf{x}) \;=\; \int \phi_m^\infty(\mathbf{y})K(\mathbf{x},\mathbf{y})p(\mathbf{y})d\mathbf{y} \quad \forall\mathbf{x}$$

In this way, we observe that

$$\begin{array}{ll}\phi_m^\infty & \text{eigenfunction of } G \text{ with eigenvalue } \lambda_m \\ \Rightarrow & \phi_m^\infty = \phi_m,\end{array}$$

as ϕ_m^∞ satisfies the Nyström function of G.

□

5.3 Algorithm for Spectral Clustering

We have just justified the use of the Nyström Formula, and how to use it to learn eigenfunctions. We are going to present now the algorithms with the Nyström formula for Spectral Clustering: the training algorithm (algorithm 7) and the testing one (algorithm 8) [10].

The normalization step let us determine the Laplacian Graph we use to construct the data graph. The algorithm (7) is based on the Symmetric Laplacian Graph, because it is the only one of the Laplacian Graphs studied that defines a symmetric kernel.

Algorithm 7 Training algorithm

1: *Input*: $D = \mathbf{x}_1, \ldots, \mathbf{x}_N$ with $\mathbf{x}_i \in \mathbb{R}^n$, $k < n$ that is the number of clusters.
2: We compute the similarity matrix W, starting from $K_D(\cdot, \cdot)$, that is a gaussian kernel such that $w_{ij} = K_D(\mathbf{x}_i, \mathbf{x}_j)$.
3: We normalize the weight matrix $\tilde{W}$ having $\tilde{w}_{ij} = \frac{w_{ij}}{\sqrt{d_i d_j}}$.
4: We search the k highest eigenvalues λ_i of $\tilde{W}$ and their associated eigenvectors v_k.
5: We make the embedding of each point in the sample $\mathbf{x}_i \rightarrow \mathbf{y}_i$, such that $y_{ik} = v_{ki}$ is the i-th element of the k-th eigenvector v_k of $\tilde{W}$.
6: We obtain the clusters $C_1, \ldots, C_k$ of our new points y_i by applying k-means algorithm.
7: **return** Clusters $A_1, \ldots, A_k$ where $A_i = \{j | y_j \in C_i\}$.

Algorithm 8 Testing algorithm

1: *Input*: $\mathbf{x} \in \mathbb{R}^n$, new point.
2: We compute our new kernel function $\tilde{K}(a, b) = \frac{1}{N} \frac{K(a,b)}{\sqrt{\mathbb{E}_{\mathbf{x}}[K(a,\mathbf{x})]\mathbb{E}_{\mathbf{x}'}[K(b,\mathbf{x}')]}}$, where the expectations are taken over the empirical data X.
3: Using the Nyström Formula we obtain the eigenfunction that we can apply to the new point

$$\phi_k(\mathbf{x}) = \frac{\sqrt{N}}{\lambda_k} \sum_{i=1}^{N} v_{ki} \tilde{K}(\mathbf{x}, \mathbf{x}_i).$$

4: From this eigenfunctions we obtain the embedded point $\mathbf{y}_k = \frac{\phi_k(\mathbf{x})}{\sqrt{N}} = \frac{1}{\lambda_k} \sum_{i=1}^{N} v_{ki} \tilde{K}(\mathbf{x}, \mathbf{x}_i)$.
5: We can apply the testing k-means algorithm in order to know the cluster that contains our new point.

Chapter 6

Numerical Experiments

In this chapter we shall put under test some of the explained methods. As we have seen in previous sections, the objective of this work is to study LE and Spectral Clustering techniques. We have implemented them and we have proved their accuracy.

To explain how we have made this analysis, we will first detail the implementations made of these methods, next we shall show which datasets were used for the tests, and finally the results of the experiments will be presented.

6.1 Methodology

We have implemented an LE algorithm (chapter 3.2) for dimensional reduction and Spectral Clustering methods (chapter 4) for clustering different data sets. Both techniques are very similar, and in both of them we have to:

1. Read the entries of our data file.

2. Compute the weights of our graph using a Gaussian Kernel. In this way we obtain a non-sparse weight matrix. We meta-model the variance parameter t varying this parameter in a logaritmic scale between $0,001$ and 100, to find the one that best represent the local information of our data.

3. Normalize the obtained weights according to the Laplacian graph we want to compute. In the case of LE we compute always a Random Walk Laplacian.

4. Compute the eigenvalues and eigenvectors of such matrix and to embed our points to a k dimensional space corresponding to the elements of the k eigenvectors with highest eigenvalues.

If we are computing one of the Spectral Clustering methods, the next steps must be:

1. Cluster the embedded data using k-means.

2. And, at the end, come back to the original space, where we have the information clustered.

All these algorithms have been implemented in C code.

To analyze the accuracy of our methods, in the one hand, we are going to study the results in a visual way, plotting the reduced or clustered data, as we have examples in a low dimension. On the other hand, to measure the accuracy of the Spectral Clustering methods we also use the percentage

of patterns correctly clustered. For this purpose, we have data sets with a target column. We compute this measure as:

$$Percentage\_accuracy = \frac{\textit{good targeted data}}{\textit{total data}}.$$

To evaluate the efficacy of our methods we will compare this percentage with the one obtained if we apply k-means to the data sets.

6.2 Experimental scenarios and datasets used

We have different data sets to test dimensional reduction techniques and Spectral Clustering methods. For LE, we have two easy synthetic samples:

1. **Spiral**: We have 2500 data points of dimension 2 that we want to reduce into a lower dimensional space. The spiral is an embedded 1-dimensional manifold in a 2-dimensional space.

2. **Helix**: We have 2500 3-dimensional examples. The helix is also an embedded 1-dimensional manifold.

For Spectral Clustering we are going to use some data sets obtained from the Graph Demo `http://www.ml.uni-saarland.de/GraphDemo/GraphDemo.html` based on article [5]. They are also easy synthetic samples, that let us verify we have implemented correctly these algorithms and that they obtain competitive results.

1. **Two moons balanced**: the first two dimensions are two half-circles of two moons shape. In all dimensions, Gaussian noise with mean 0 and variance $0,01$ has been added. Both moons have the same weight (probability mass), that is in average they will contain the same number of data points.

2. **Two moons unbalanced**: as "Two moons balanced", but the two classes have unequal weights of $0,2$ and $0,8$, respectively. That is, in expectation only 20% of the points come from the first moon, while 80% of the points come from the second moon.

3. **Two Gaussians balanced**: The data are the two points, $(-1.1, -1.1)$ and $(1, 1)$, in two-dimensional space. Both points are disturbed by Gaussian noise of variance $0,36$. Both classes have equal weight.

4. **Two Gaussians unbalanced**: As "Two Gaussians Balanced", but unbalanced class weights of $0,2$ and $0,8$.

5. **Two Gaussians, different variance**: As "Two Gaussians Balanced", but each point is disturbed by a Gaussian noise of different variance, one of $0,36$ and thge other one of $0,16$.

6. **Three Gaussians**: The data are the three points $(1, 1)$, $(-1.1, -1.1)$ and $(2, -2)$, in two-dimensional space. All three points are disturbed by Gaussian noise of variance $0,36$. The weights of the Gaussian are $0,3$, $0,3$ and $0,4$.

Finally, we will also prove Spectral Clustering methods with a classic classification example of higher dimension: the **Ringnorm** problem. It has as inputs the points from two Gaussian distributions. Class 1 is multivariate normal with mean 0 and covariance 4 times the identity matrix. Class 2 has unit covariance and mean $(a, a, ..., a)$, $a = dim^{-0,5}$.

6.3 LE Dimensionality Reduction Results

To prove the dimension reduction methods, we have executed the LE algorithm with two data sets.

Helix The Helix problem is a set of 2500 examples in a 3-dimensional space. But an helix is a 1-dimensional manifold, so we want to reduce our points to a space with dimension 1. Then, to apply LE we must use $k = 1$ and the best weight matrix parameter is fixed in $t = 0, 1$.

This easy example shows us that LE is really able to reduce dimension. The helix is perfectly unrolled in dimension 1, as we can see because the colours follows the same order than in the 3-dimensional space.

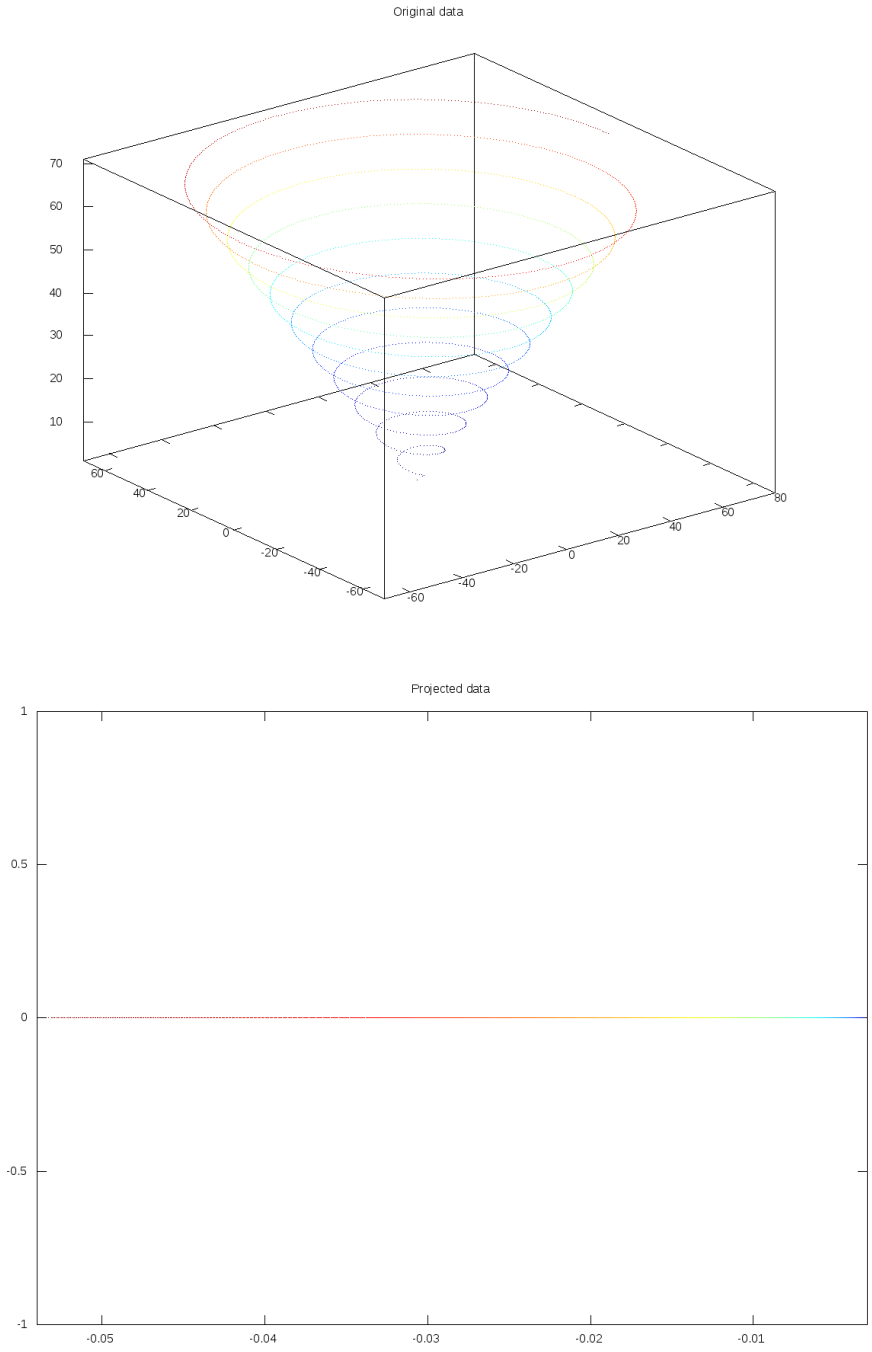

Figure 6.3.1: Helix before and after the embedding applying LE algorithm .

Spiral The Spiral problem is a data set with 2500 sample points in a 2-dimensional space. As a spiral is an embedded 1-dimensional manifold, we apply LE with $k = 1$. The weight matrix parameter in this case should be $t = 0,01$.

The result of this method is shown in figure 6.3.2. The idea is to unroll the spiral in $k = 1$, so the local information is conserved if and only if the colours in the reduced space follows the same sequence than in the original one. Whereas PCA is not able to keep up local distance in this case as it makes a linear projection of the data, with LE we unroll the spiral in a more satisfactory way.

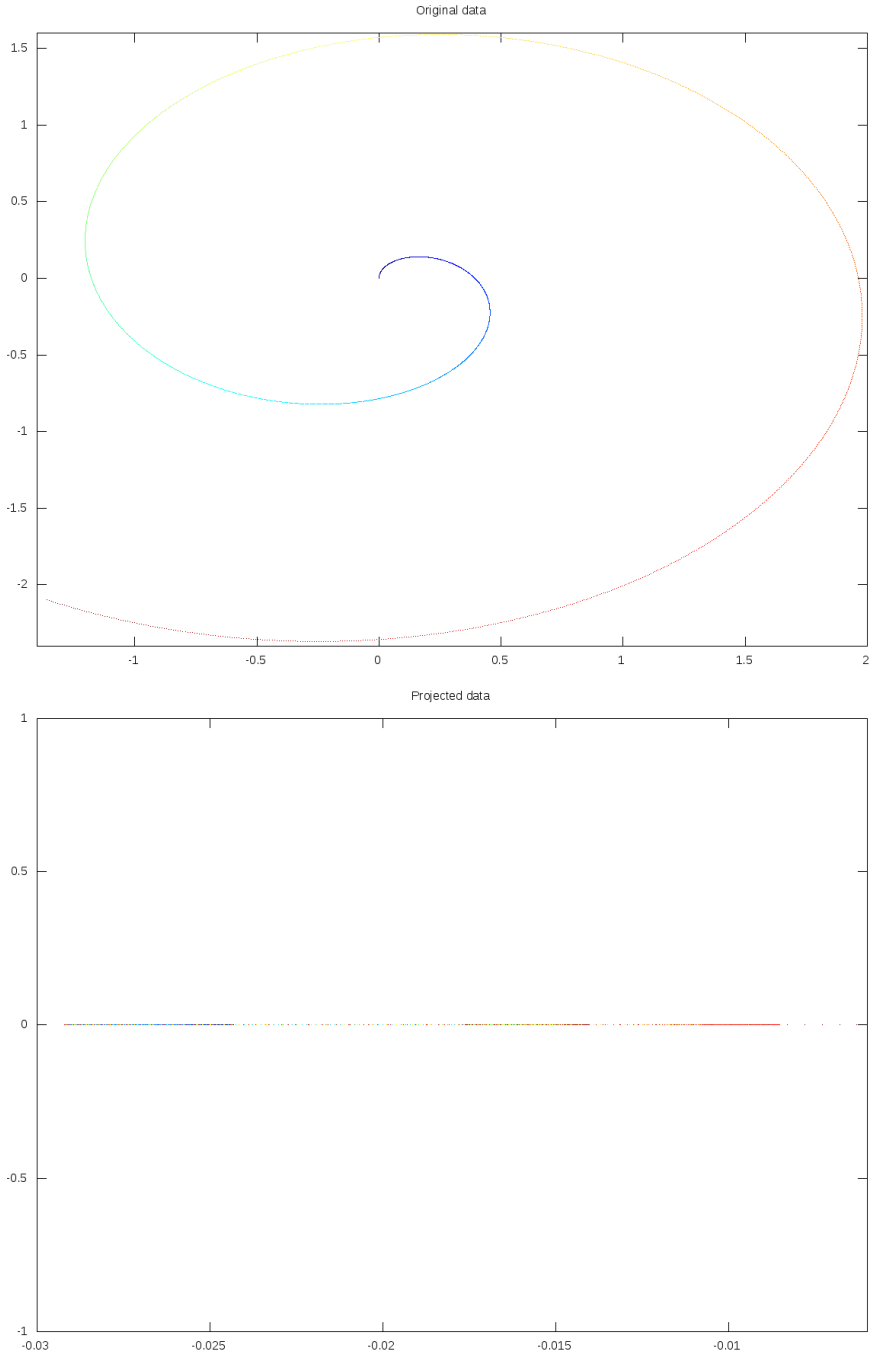

Figure 6.3.2: Spiral before and after the embedding applying LE algortihm.

6.4 Spectral Clustering Results

To cluster data, we have implemented Spectral Clustering with the three different Laplacian Graphs studied before (chapter 4). We are going to see in the following results that, as we have anticipated, the unnormalized laplacians do not work as well as the normalized one. Between Symmetric and Random Walk Laplacians there do not exist a big difference, the best method depend on the data set.

We are going to present the best results obtained with the different Spectral Clustering implementations, comparing the results with the ones that we obtain if we apply k-means over the original data. At the end of this section we present a table (6.4.1) with all the results obtained with different parameters.

Case 1: Two moons balanced This data set is a 500 patterns sample of dimension 3. We reduce it to a 2-dimensional space where we must find $k = 2$ clusters. The best Gaussian kernel parameter is $t = 0,01$, obtaining the same result for Symmetric and for Random Walk Laplacians.

The percentage accuracy in this case has been of $99,60\%$ better than k-means, that obtained a $78,80\%$.

We show in figures (6.4.1), (6.4.2), (6.4.3) the results obtained with the Symmetric Laplacian Graph. We observe that Spectral Clustering classifies almost every point in each corresponding clusters while k-means always fail in the moons' extremes.

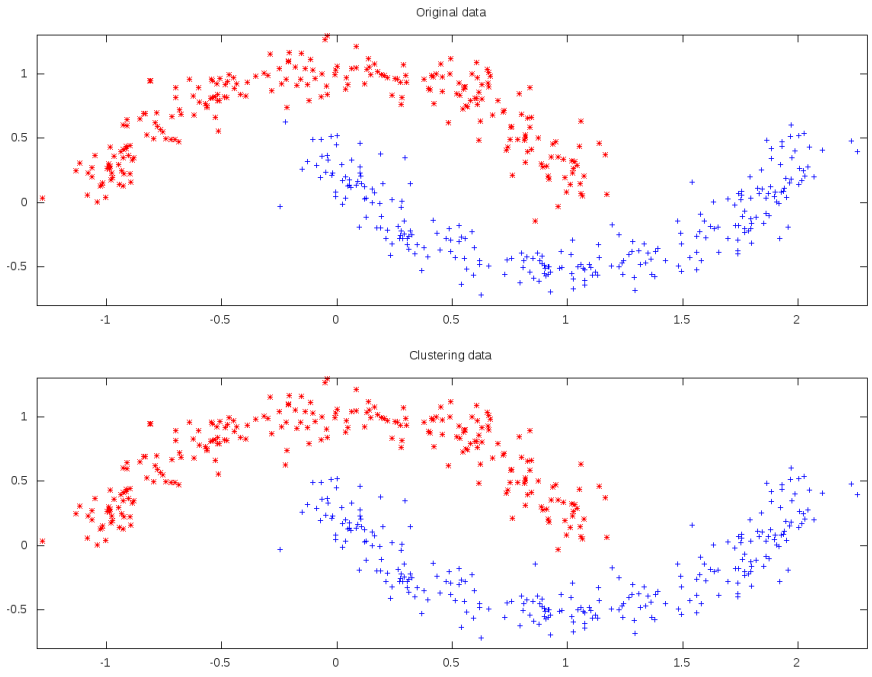

Figure 6.4.1: Original clusters and predicted ones using Spectral Clustering in the initial space.

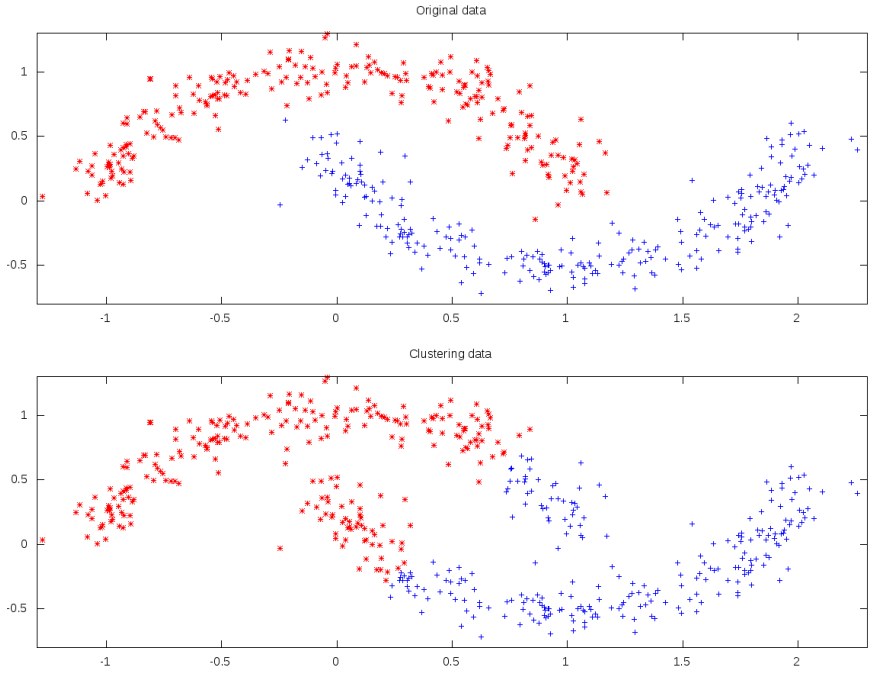

Figure 6.4.2: Original clusters and predicted ones using k-means in the initial space.

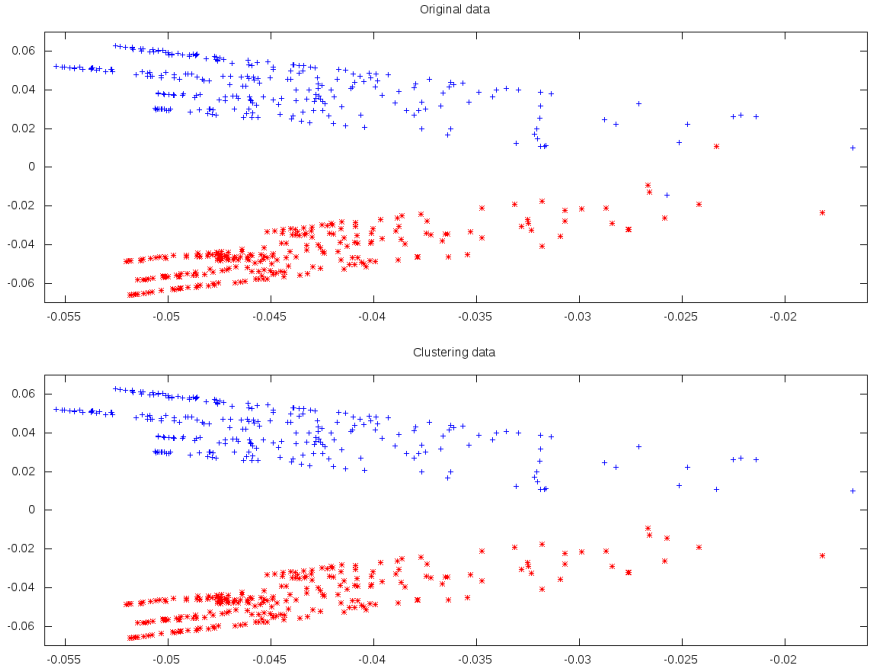

Figure 6.4.3: Original clusters and predicted ones using Spectral Clustering in the embedded space.

Case 2: Two moons unbalanced We have now a 500 points data set of dimension 3. We reduced it to a 2-dimensional space where we must find $k = 2$ clusters. The best results are obtained with a Gaussian kernel parameter $t = 0,001$ for Symmetric Laplacian Graph.

The percentage accuracy in this case has been of 100% better than k-means, that obtained a $62,20\%$.

We show in figures (6.4.4), (6.4.5), (6.4.6) the results obtained for this example. We observe that Spectral Clustering classifies well the whole data set while k-means always fail again in the moons' extremes.

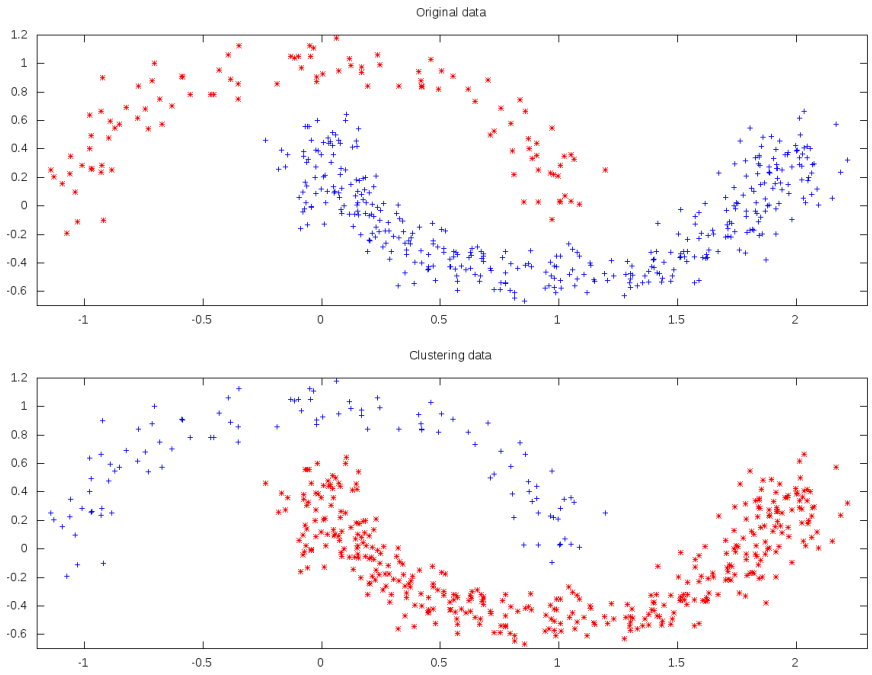

Figure 6.4.4: Original clusters and predicted ones using Spectral Clustering in the initial space.

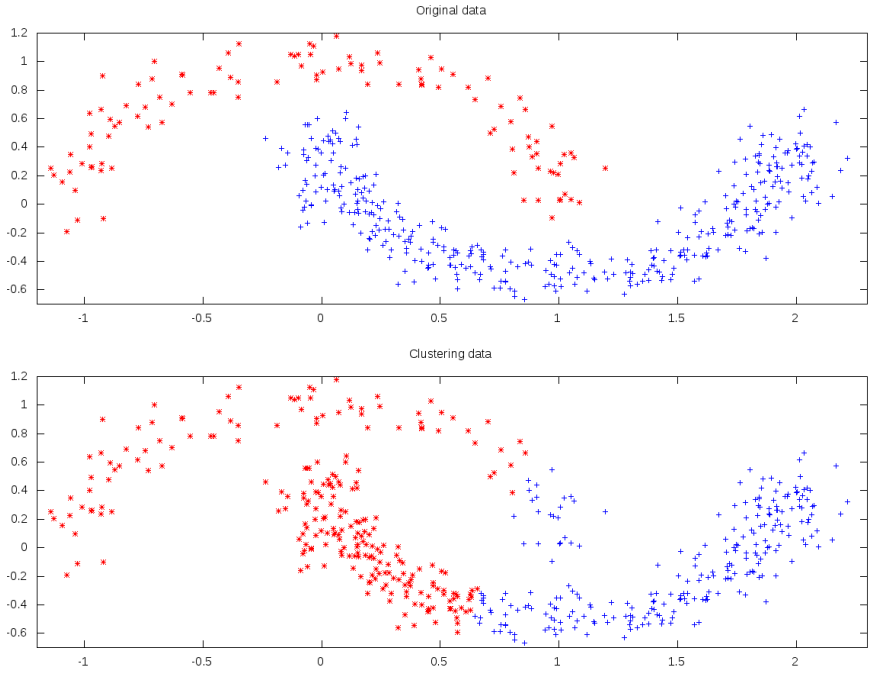

Figure 6.4.5: Original clusters and predicted ones using k-means in the initial space.

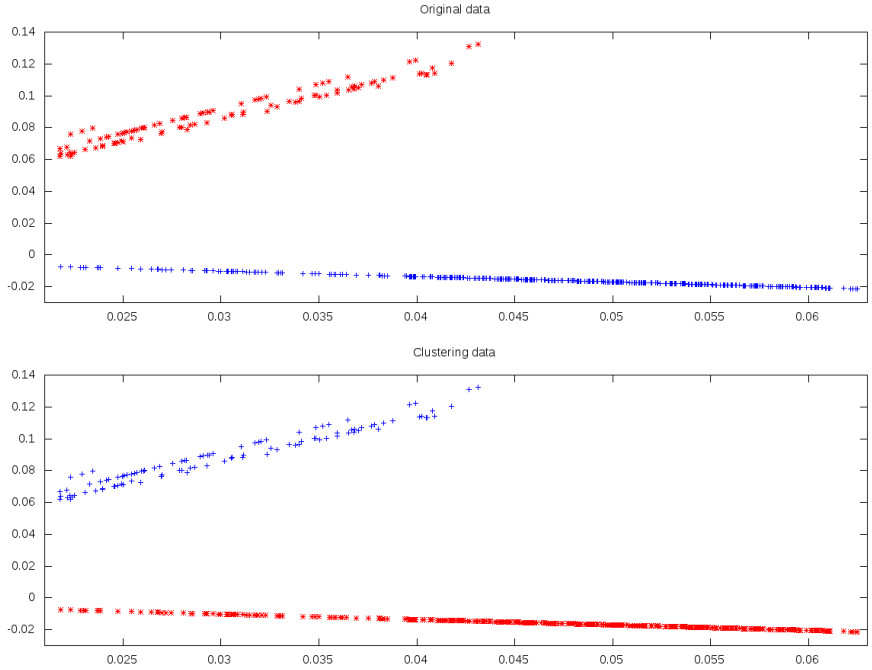

Figure 6.4.6: Original clusters and predicted ones using Spectral Clustering in the embedded space.

Case 3: Two Gaussians balanced This data set is a 500 patterns sample of dimension 3. We reduced it to a 2-dimensional space where we must find $k = 2$ clusters. The best Gaussian kernel parameter is $t = 0,01$, obtaining sligthly better results with a Random Walk Laplacian Graph method. This two Gaussians are sufficiently separated, so any clustering method offers a good prediction.

The percentage accuracy in this case has been of $99,20\%$, that is really similar to the $99,00\%$ obtained with k-means.

We show in figures (6.4.7), (6.4.8), (6.4.9) the results obtained with the Random Walk Laplacian Graph.

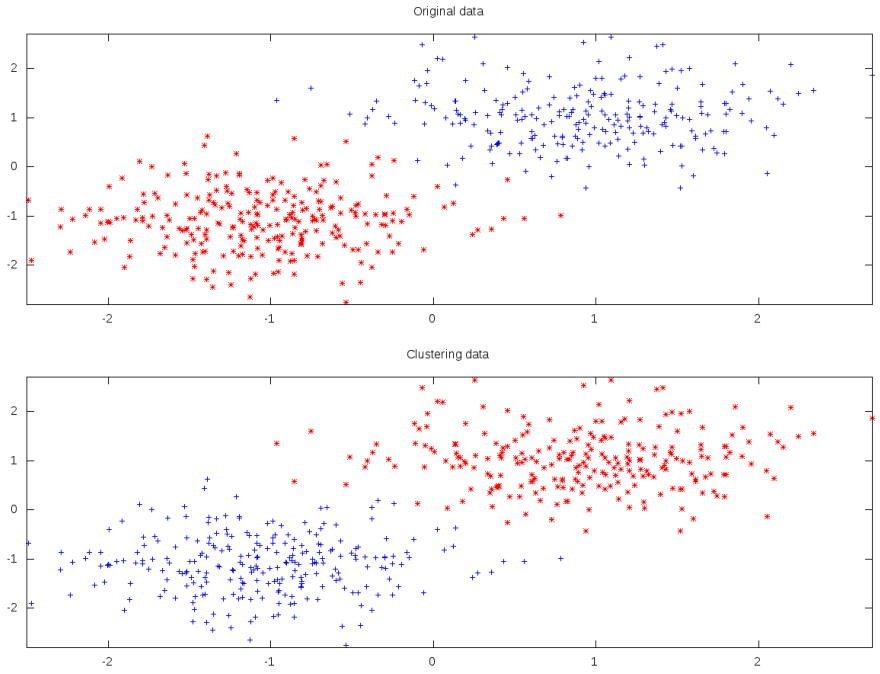

Figure 6.4.7: Original clusters and predicted ones using Spectral Clustering in the initial space.

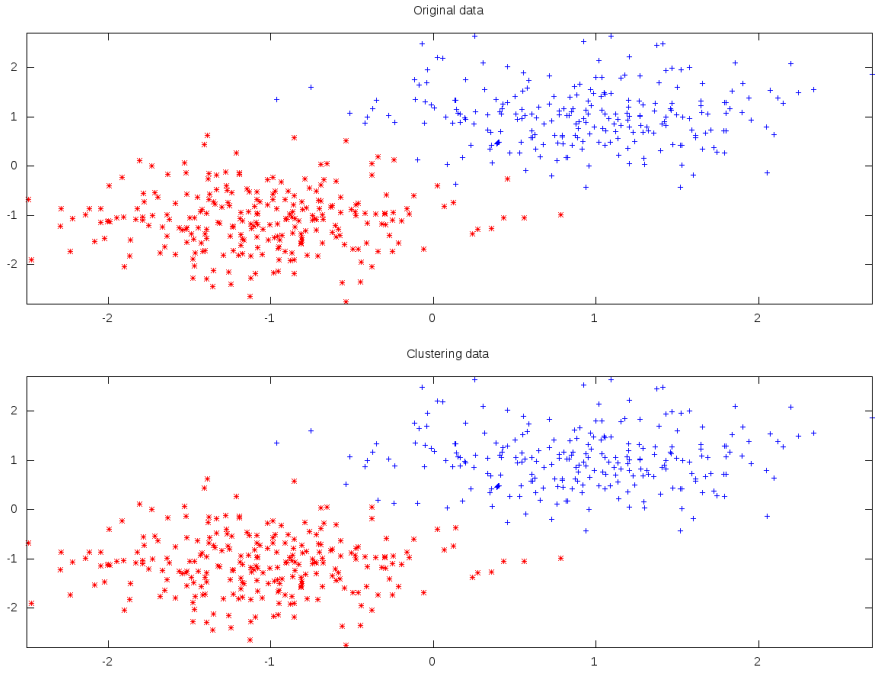

Figure 6.4.8: Original clusters and predicted ones using k-means in the initial space.

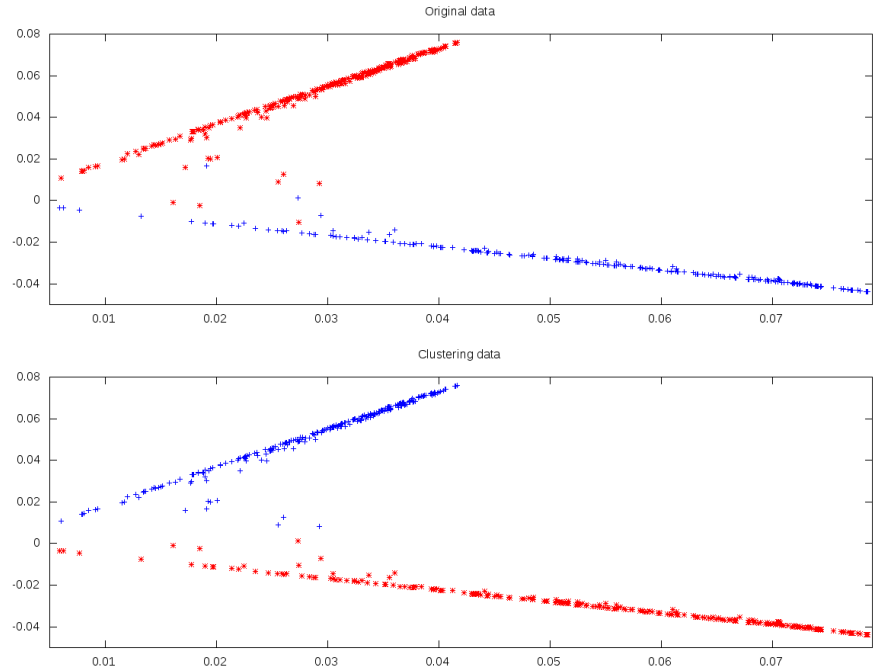

Figure 6.4.9: Original clusters and predicted ones using Spectral Clustering in the embedded space.

Case 4: Two Gaussians unbalanced We cluster now a 500 patterns sample of dimension 3. We reduced it to a 2-dimensional space where we must find $k = 2$ clusters. The best Gaussian kernel parameter is $t = 0,01$, obtaining the same result for Symmetric and Random Walk Laplacians. This example is also very simple to solve, and results with Spectral Clustering and k-means are both very good.

The percentage accuracy in this case has been of $98,80\%$, very near to k-means results, that obtained a $96,60\%$.

We show in figures (6.4.10), (6.4.11), (6.4.12) the results obtained with the Symmetric Laplacian Graph.

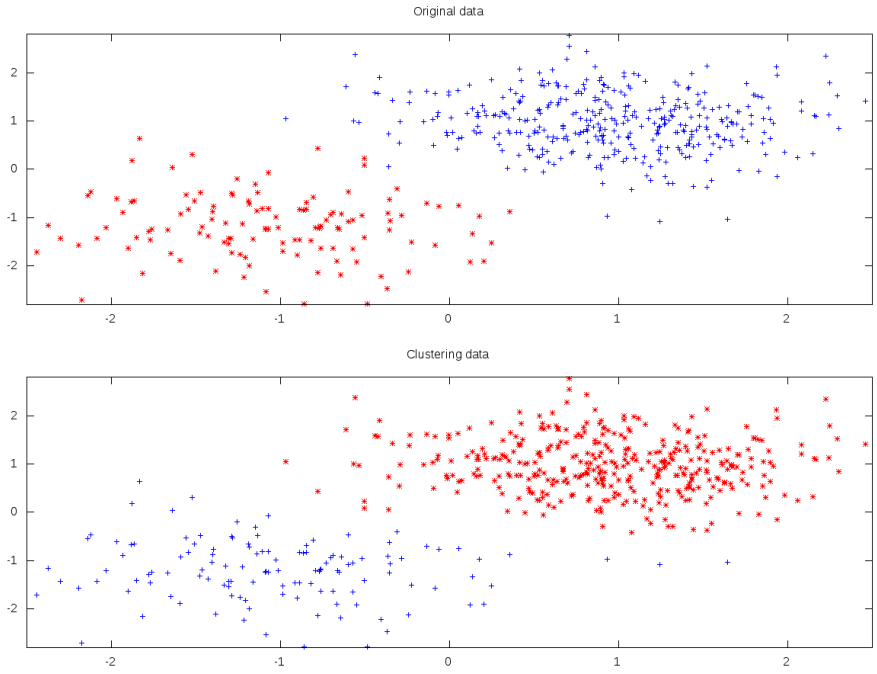

Figure 6.4.10: Original clusters and predicted ones using Spectral Clustering in the initial space.

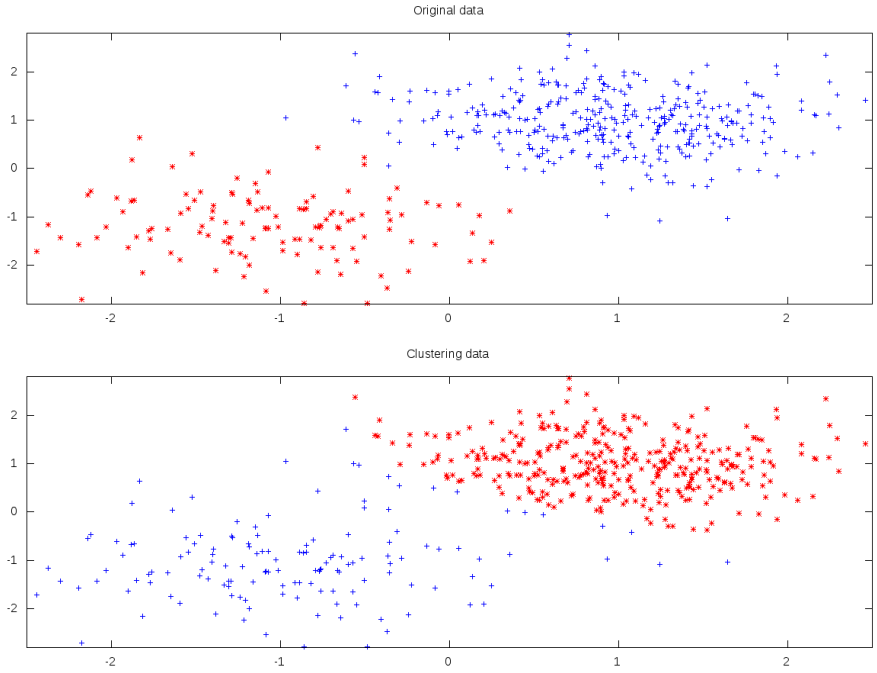

Figure 6.4.11: Original clusters and predicted ones using k-means in the initial space.

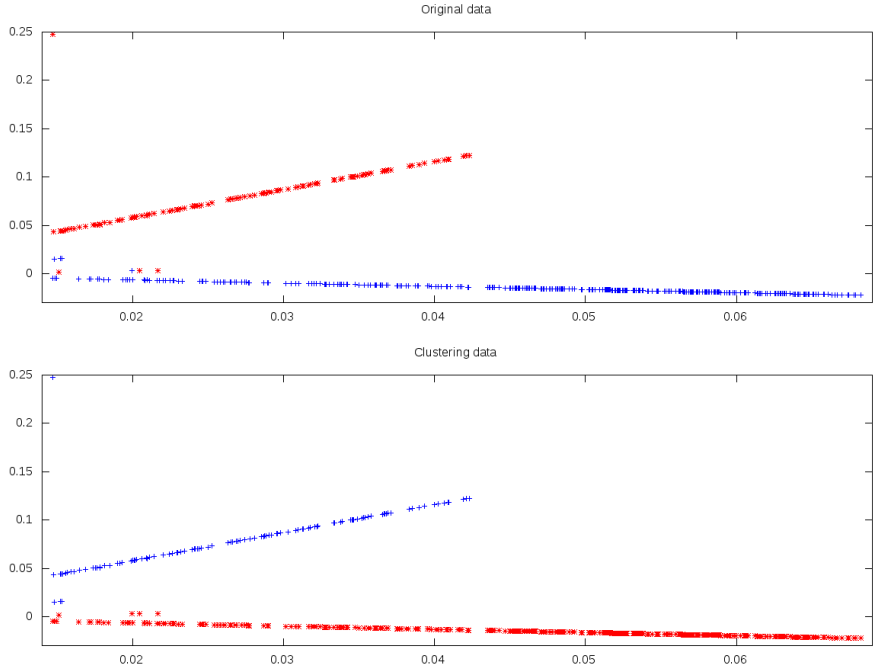

Figure 6.4.12: Original clusters and predicted ones using Spectral Clustering in the embedded space.

Case 5: Two Gaussians different variance This data set is a 500 patterns sample of dimension 3. We reduced it to a 2-dimensional space where we must find $k = 2$ clusters, corresponding to the two Gaussians. The best Gaussian kernel parameter is $t = 0,1$, obtaining a sligthly better result with Random Walk Laplacians. Again, clustering two Gaussians that do not overlap in many points is easy, so the results are very good.

The percentage accuracy in this case has been of 100% more or lees as k-means, that obtained a $99,80\%$.

We show in figures (6.4.13), (6.4.14), (6.4.15) the results obtained with the Random Walk Laplacian Graph.

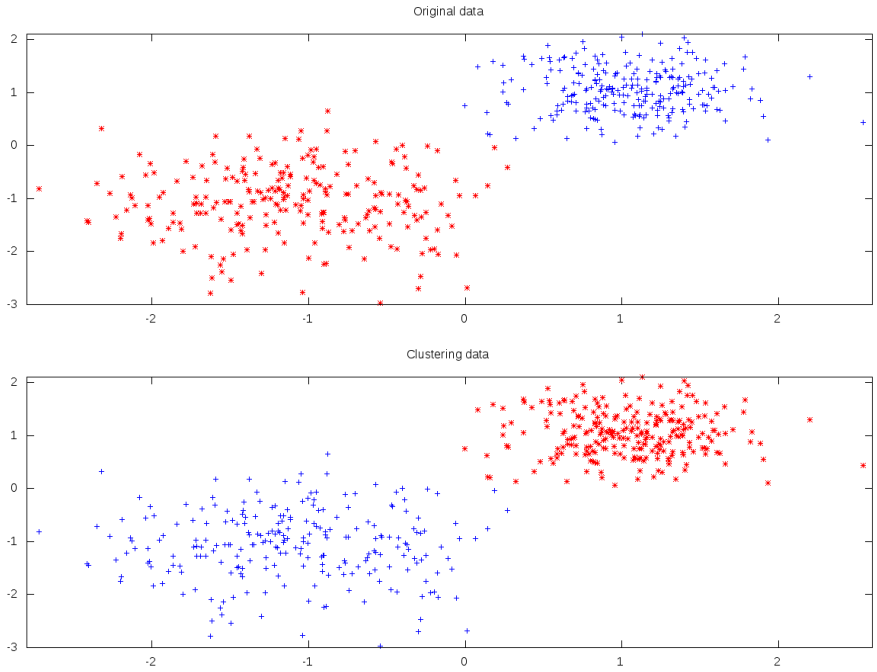

Figure 6.4.13: Original clusters and predicted ones using Spectral Clustering in the initial space.

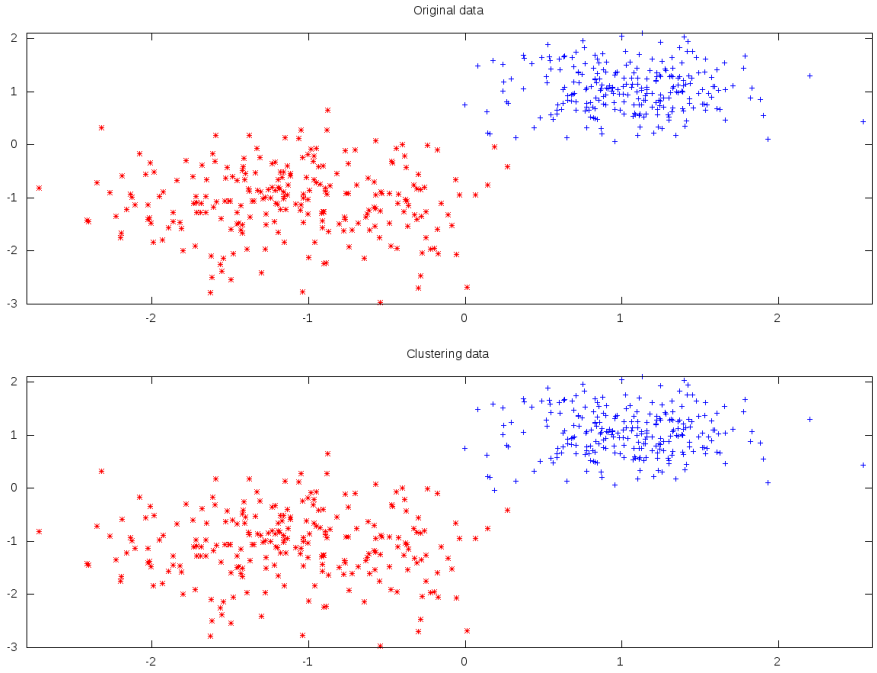

Figure 6.4.14: Original clusters and predicted ones using k-means in the initial space.

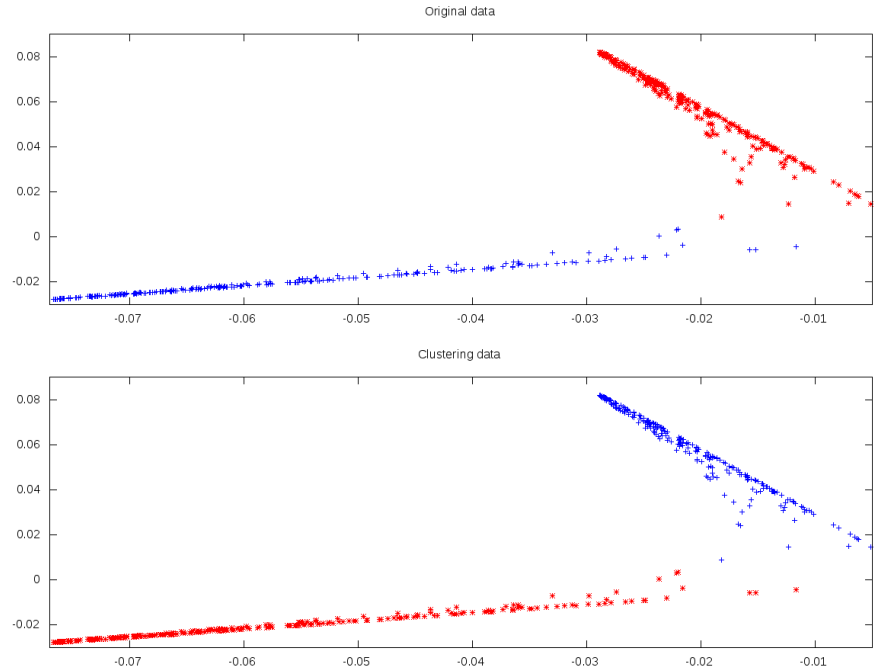

Figure 6.4.15: Original clusters and predicted ones using Spectral Clustering in the embedded space.

Case 6: Three Gaussians This data set is a 500 patterns sample of dimension 3. We reduced it to a 2-dimensional space where we must find $k = 3$ clusters. The percentage accuracy in this case is always lower than k-means, that obtained a $98,60\%$ independently of the parameters. The best result is a $95,20\%$ obtained with a Random Walk Laplacian Graph and $t = 0,01$.

If we observe the embedding (figure 6.4.18), for example for $t = 0,1$ with a Random Walk Laplacian, we see that it has made a good dimension reduction, as it conserves the geometric distance between neighbors points. We see that the different clusters appears separately, and the red and blue Gaussians, that in the original data (figure 6.4.16) are closer, maintain their distances. The bad results are due to the k-means limitations in this situation. In the original space we face with an easy problem so k-means gives excellent results (figure 6.4.17).

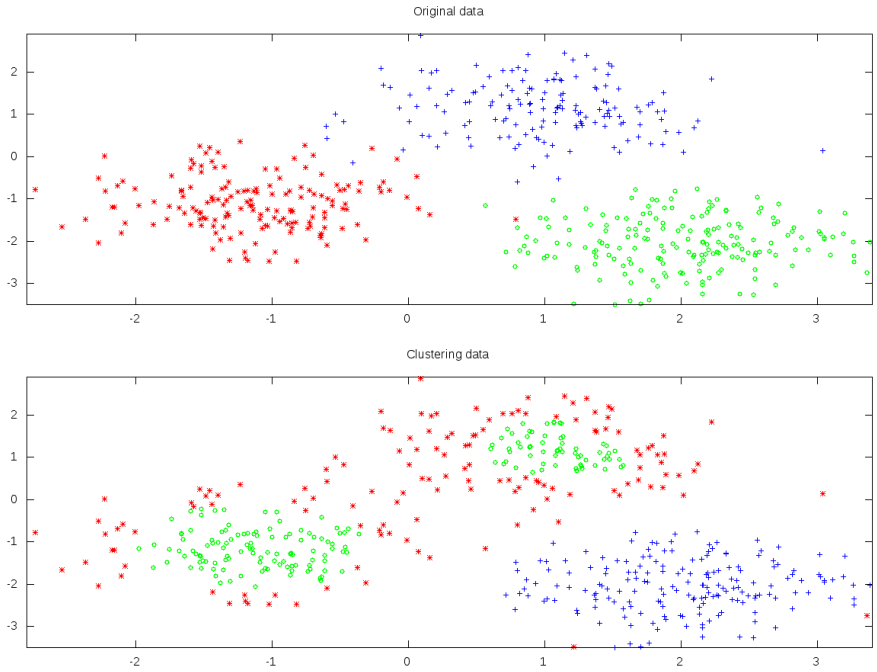

Figure 6.4.16: Original clusters and predicted ones using Spectral Clustering in the initial space.

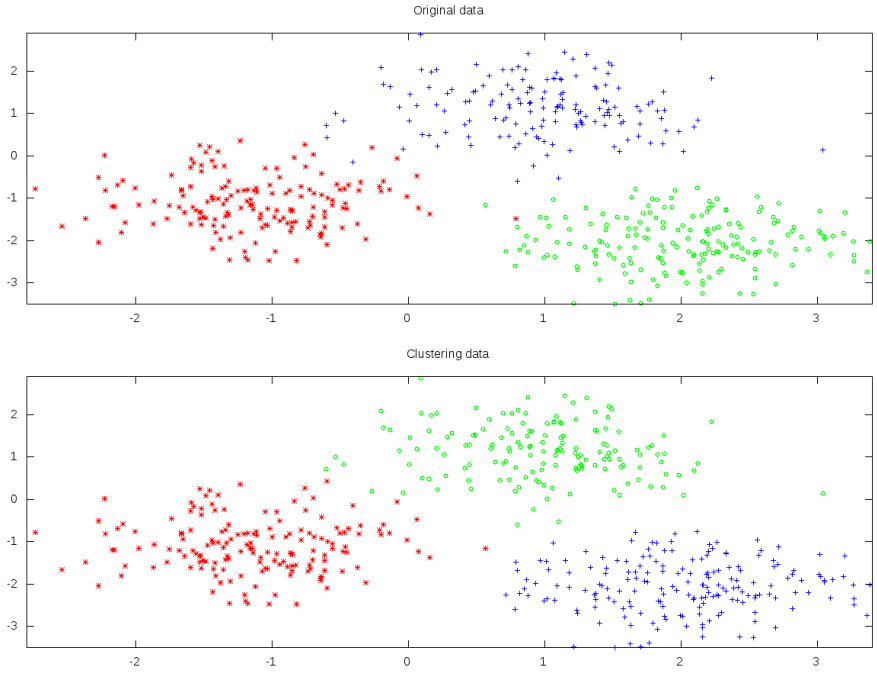

Figure 6.4.17: Original clusters and predicted ones using k-means in the initial space.

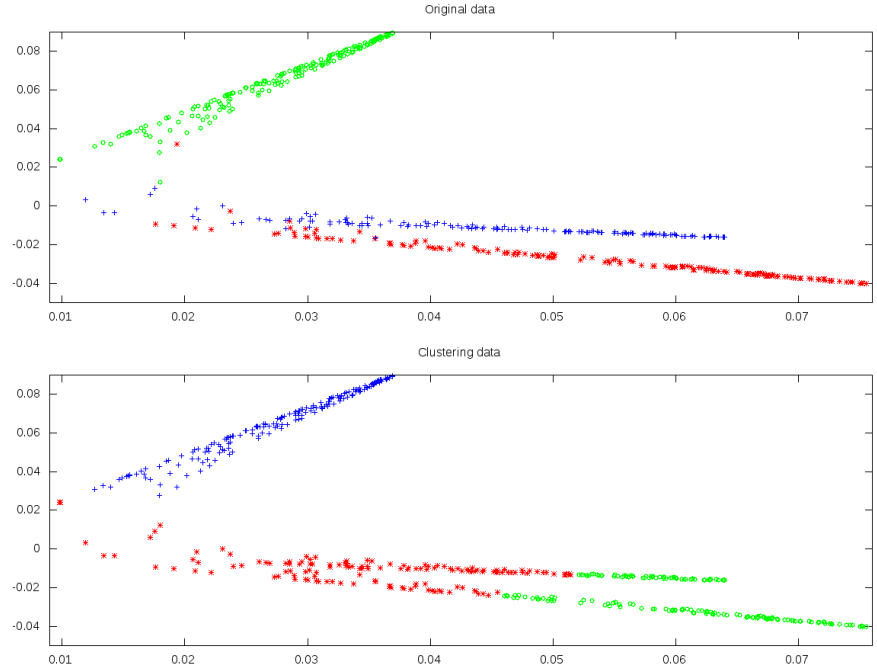

Figure 6.4.18: Original clusters and predicted ones using Spectral Clustering in the embedded space.

Case 7: Ringnorm With this data set of 400 examples, we want to prove a higher dimensional case. The original data is embedded in a 20-dimensional space. We reduced it to a 2-dimensional space where we must find $k = 2$ clusters. The best weight matrix parameter is $t = 0,1$, obtained with a Random Walk Laplacian Graph. This example represents a central circle with an external annulus. For k-means is very difficult to cluster this kind of data, as we can see in figure (6.4.20), whereas Spectral Clustering obtained a good result (figure 6.4.19).

The percentage accuracy in this case has been of $96,00\%$ better than k-means, that obtained a $59,50\%$.

We show in figures (6.4.19), (6.4.20), (6.4.21) the results obtained with the Random Walk Laplacian Graph method.

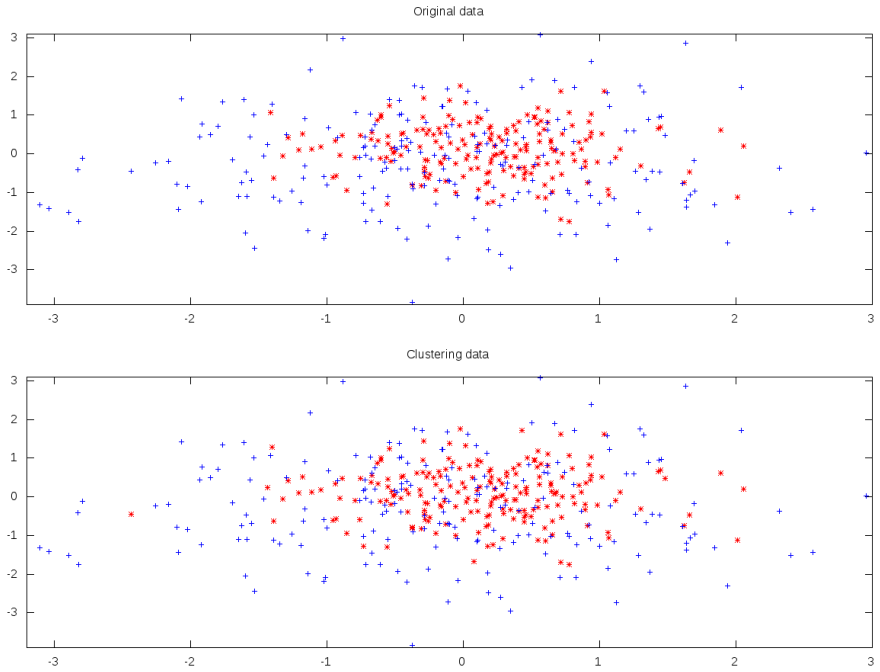

Figure 6.4.19: Original clusters and predicted ones using Spectral Clustering in the initial space.

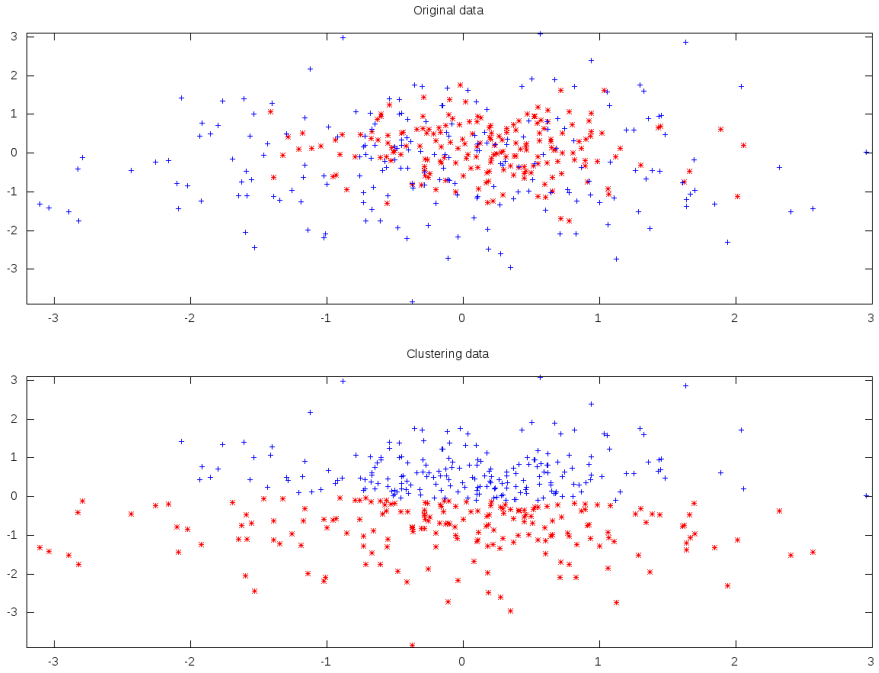

Figure 6.4.20: Original clusters and predicted ones using k-means in the initial space.

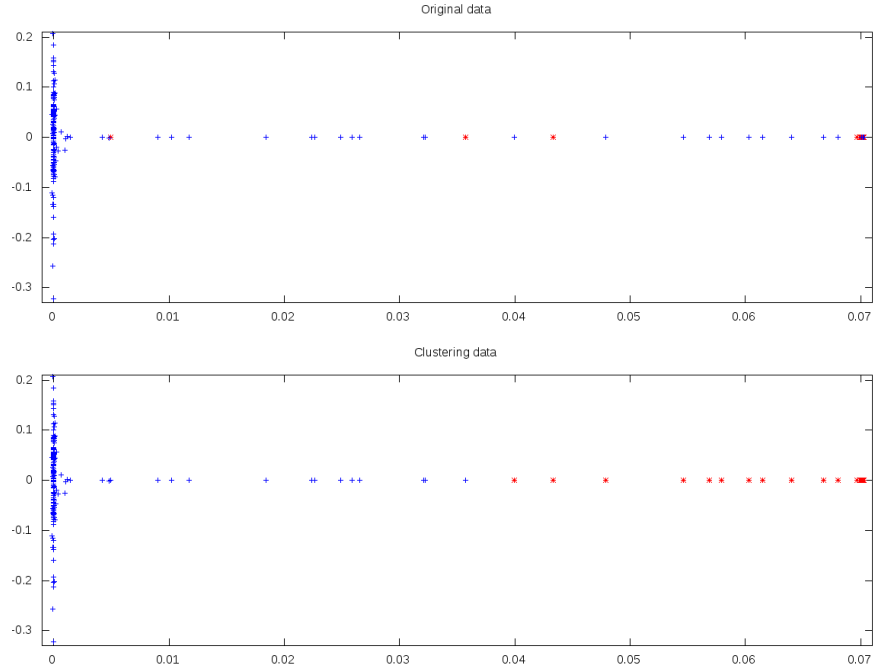

Figure 6.4.21: Original clusters and predicted ones using Spectral Clustering in the embedded space.

	Case 1		Case 2		Case 3		Case 4		Case 5		Case 6		Case 7	
	SC	kM	SC	kM	SC	kM	SC	kM	SC	kM	SC	kM	SC	kM
$\mathbf{L_{sym}}$														
0,001	51,40%	78,80%	100,00%	62,20%	50,20%	99,00%	98,60%	96,60%	67,00%	99,80%	95,00%	98,60%	52,25%	59,50%
0,010	99,60%	78,80%	99,40%	61,40%	99,00%	99,00%	98,80%	96,60%	51,00%	99,80%	18,40%	98,60%	53,25%	59,50%
0,100	83,00%	78,80%	59,40%	62,20%	99,00%	99,00%	99,40%	96,60%	99,80%	99,80%	71,20%	98,60%	87,75%	59,50%
1,000	76,40%	78,80%	62,00%	61,40%	99,00%	99,00%	99,60%	96,60%	99,80%	99,80%	68,40%	98,60%	51,50%	59,75%
10,000	75,60%	78,80%	62,20%	62,20%	99,20%	99,00%	98,80%	96,60%	99,80%	99,80%	71,40%	98,60%	73,50%	59,50%
100,000	75,20%	78,80%	61,00%	62,00%	99,20%	99,00%	98,20%	96,60%	99,80%	99,80%	71,00%	98,60%	71,25%	60,50%
$\mathbf{L_{rw}}$														
0,001	53,80%	79,20%	50,00%	62,20%	62,60%	99,00%	67,60%	96,60%	79,00%	99,80%	33,40%	98,60%	52,25%	59,25%
0,010	99,60%	78,80%	56,60%	62,00%	99,20%	99,00%	98,80%	96,60%	99,80%	99,80%	95,20%	98,60%	53,50%	59,50%
0,100	82,60%	78,80%	59,80%	61,20%	99,00%	99,00%	99,60%	96,80%	100,00%	99,80%	76,80%	98,60%	96,00%	59,50%
1,000	75,80%	78,80%	62,00%	62,20%	99,20%	99,20%	99,60%	96,60%	99,80%	99,80%	68,80%	98,60%	51,50%	59,50%
10,000	75,40%	78,80%	61,00%	61,40%	99,20%	99,20%	97,80%	96,60%	99,80%	99,80%	71,20%	98,60%	72,00%	59,50%
100,000	75,20%	78,80%	61,20%	62,00%	99,00%	99,00%	98,20%	96,60%	99,80%	99,80%	71,00%	98,60%	70,50%	58,50%
$\mathbf{L}$														
0,001	54,80%	78,80%	70,60%	62,00%	53,00%	99,20%	71,40%	96,60%	51,60%	99,80%	29,00%	98,60%	52,25%	59,25%
0,010	58,40%	79,20%	72,00%	62,00%	58,60%	99,00%	63,40%	96,60%	62,60%	99,80%	34,60%	98,60%	52,25%	61,75%
0,100	51,60%	78,80%	55,20%	62,00%	53,20%	99,00%	69,40%	96,60%	63,20%	99,80%	40,60%	98,60%	53,50%	59,75%
1,000	51,40%	78,80%	78,80%	62,00%	53,80%	99,00%	69,40%	96,80%	57,60%	99,80%	29,00%	98,60%	68,75%	59,50%
10,000	51,80%	78,80%	79,00%	62,00%	51,60%	99,00%	74,00%	96,60%	51,20%	99,80%	27,40%	98,60%	58,75%	60,50%
100,000	51,80%	78,80%	79,00%	61,40%	51,60%	99,00%	72,40%	96,60%	51,80%	99,80%	27,40%	98,60%	52,75%	58,50%

Table 6.4.1: Completed results for Spectral Clustering methods

Chapter 7

Conclusions and further work

7.1 Conclusions

In this work we have made a revision of classical methods for dimensionality reduction and clustering. After seen the disadvantages of these approaches, we conclude that the main problem is that they do not consider the explicit form of the differential manifold structure in which our data probably lie. With the purpose of solving this problem we have focused this work in the study of advanced methods for dimensionality reduction and clustering, specifically Laplacian Eigenmaps (LE) and Spectral Clustering algorithms. The main advantages of these techniques is that they reduced the dimension of our original data, preserving the local geometric information of the points. The embedding of the points corresponds with the subjacent manifold to which the original data belongs. For Spectral Clustering methods, we apply classical clustering methods over the embedded data, problem that is easier to be solved. In this way, Spectral Clustering does not make any assumption on the form of the clusters.

After a theoretical study, we have implemented and proved these methods. Whereas the experiments have been made over synthetical simple examples, we have seen that the results are as good as we have expected. In the one hand, LE reduce the dimension of our original data, constructing an embedding that keeps close neighbor points. It is able to reduce in a good way examples as spirals that PCA is not able to unroll. The problem of LE algorithm is that only works properly if the data is uniformly sampled in the original space.

On the other hand, we have further proved that Spectral Clustering with the right parameters gives very good results in easy examples. It also improves the results obtained with k-means in examples where this classical algorithm is not able to find a proper solution. The main problem of Spectral Clustering techniques is the selection of the parameter values and the Laplacian Graph more adequate to each case. We have seen in the experimental work performed that depending on the value of these variables, the results can be excelent or a complete disaster due to an inappropriate embedding.

In conclussion, we have presented some methods for dimensional reduction and clustering that, if they are correctly parametrized, improves the classical approaches studied until now.

7.2 Further work

The experiments presented in this work have been performed with synthetical simple data sets that have allowed us to study the behaviour and accuracy of the different algorithms. In future work, we will apply these methods to real high dimensional data sets. The objective will be to cluster the data and try to classify it taking advantages of the information that gives us those clusters.

As the main problem of Spectral Clustering is the search of appropiate parameters for the method, it will be useful to define a methodology to select these values depending on the properties of the data sets we are working with.

Although we have carried out a deep theoretical study of the out-of-sample Spectral Clustering, we have not implemented it in this work. In future works we want to prove this approximation to Spectral Clustering and check that the results in test data are similar to the ones obtained with a training set.

Also, we could improve these studies working on Diffusion Maps [24], [25]. These methods are based on the searching of significant geometric properties of a data set and they work even with non-uniformly distributed data. They are very related with Spectral Clustering and we would like to compare the behaviour and accuracy of both approaches.

Appendix A

Notation glossary

A.1 Graph notation

$\mathbf{1}$ is the unity vector, i.e., $\mathbf{1} = (1, \ldots, 1)$.

$\mathbf{G} = (S, E)$, is the similarity graph, formed by the nodes V and the edges E.

$\mathbf{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is our set of nodes formed by the points of the dataset in the original n dimensional space.

$\mathbf{W} = (w_{ij})$, with $w_{ij} \geqslant 0$, is the adjacency matrix of the graph.

We work with undirected graphs, so $w_{ij} = w_{ji}$.

$\mathbf{D}$ is the degree matrix, defined as the diagonal matrix with the degrees $\mathbf{d_i} = \sum_{j=1}^{N} w_{ij}$ as its elements.

$\bar{\mathbf{A}} = S/A$ where $A \subset V$ is a subset of the vertices set. We define the indicator vector

$$\mathbf{1}_A = (f_1, \ldots, f_m)'$$

where $f_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \in A. \\ 0 & \text{if } \mathbf{x}_i \notin A. \end{cases}$

To simplify notation, we use the compact expression $\{i|\mathbf{x}_i \in A\}$.

$|\mathbf{A}|$ represents the number of nodes in A. It is a mesure based on the number of vertices of our graph.

$\mathbf{vol}(\mathbf{A}) = \sum_{i \in A} d_i$ represents the size of A. It is a mesure based on the weights of the edges of our graph.

We say that our subset A is **connected** if any two vertices in A are connected and all the intermediate points belong to A.

A is a **connected component** if it is connected and it no has connections with $\bar{A}$.

The sets $A_1, \ldots, A_m$ form a partition of the graph if $A_i \bigcap A_j = \emptyset$ and $A_1 \bigcup \cdots \bigcup A_k = V$, where k is the dimension of the reduced space.

We call **Frobenius norm** to the two-norm for matrices. Mathematically, if A is a $m \times n$ matrix and A^* its conjugated matrix

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{Tr(A^* \cdot A)} = \sqrt{\sum_{i=1}^{\min\{m,n\}}\sigma_i^2}.$$

The **Gram matrix** G of a set of vectors $v_1, \ldots, v_n$ in an inner product space is the Hermitian matrix of inner products, whose entries are given by $G_{ij} =< v_i, v_j >$.

In linear algebra, the **outer product** typically refers to the tensor product of two vectors. It is equivalent to a matrix multiplication $\mathbf{uv}^T$.

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{uv}^T = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \\ u_4v_1 & u_4v_2 & u_4v_3 \end{bmatrix}.$$

Appendix B

Constructing similarity graphs

To construct similarity graphs from a given set $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, we establish the nodes as the points of the set. To create the links between nodes we must determine the proximity between them and the weights of these edges. The goal we want to reach by the construction of a similarity graph is to model the local neighborhood relationships between the data points. We are going to present here some of these techniques, that appear explained in [2] and [5].

- **The ϵ-neighborhood graphs:** We connect all points with distance smaller than ϵ,

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2 \quad < \quad \epsilon.$$

  When the distances between all connected points are of the same scale the weights will not incorporate more information, so we usually construct no weighted graphs. The principal advantage of this method is that it is geometrically motivated, as it is the natural measure between points. But in graphs very connected is more difficult to choose an appropriate ϵ.

- **The k-nearest neighbor graphs:** We connect $\mathbf{x}_i$ with its k nearest neighbors. If we construct our graph in this way, i.e, $(\mathbf{x}_i, \mathbf{x}_j) \in E$ if $\mathbf{x}_j \in k$ nearest neighbors of $\mathbf{x}_i$, we obtain a symmetric graph. If we prefer to work with an asymmetric graph, we could modify slightly this method in one of the following ways:

  - Option 1: The usually called *k-nearest neighbors graph* is constructed without considering the direction, i.e., $(\mathbf{x}_i, \mathbf{x}_j) \in E$ if $\mathbf{x}_j \in k$ nearest neighbors of $\mathbf{x}_i$ or $\mathbf{x}_i \in k$ nearest neighbors of $\mathbf{x}_j$.
  - Option 2: The graph called *mutual k-nearest neighbors graph* is constructed connecting $\mathbf{x}_i$ and $\mathbf{x}_j$ only if $\mathbf{x}_j \in k$ nearest neighbors of $\mathbf{x}_i$ and $\mathbf{x}_i \in k$ nearest neighbors of $\mathbf{x}_j$.

  We weight the edges by the similarity between adjacent points. This relation is given by the similarity matrix S (see appendix A). The advantage of this method is that the neighbors of each point are easier to choose and the resulting graph is always connected. But we obtain a graph geometrically less intuitive.

- **The fully connected graphs:** We connect each point with all the other nodes of our graph. In this case, we weight the connected nodes with s_{ij} (see appendix A) for each edge. We only use the similarity function with this purpose if it contributes with local information. An example of this kind of functions is the exponential decay function:

$$s(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right),$$

  where σ controls the change of our neighborhood.

Appendix C

Laplacian Graphs

Laplacian Graphs are one of the main tools for Laplacian Eigenmaps and Spectral Clustering. In this appendix we are going to explain the different types of Laplacian Graphs and their most important properties following the classification given at [5]. For this purpose, we assume that our graph G is an undirected weighted graph and the weighted matrix W has positive entries ($w_{ij} = w_{ji} \geqslant 0$). We classify the Laplacian Graphs in unnormalized and normalized Laplacian Graphs.

- **Unnormalized Laplacian Graph**

  We define an Unnormalized Laplacian Graph L as follows

  $$L = D - W$$

  This kind of graphs are characterized by a series of properties describe in the following proposition.

  Proposition 5 ***Properties of*** **L**.

  1. $\forall f \in \mathbb{R}^N$, *where* f *is any vector, it satisfies*

  $$f'Lf = \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2. \tag{C.0.1}$$

  2. L *is a symmetric and positive semidefinite matrix.*
  3. *The smaller eigenvalue of* L *is* 0 *and it corresponds to the eigenvector* $\mathbf{1}$.
  4. L *has* N *non-negative real-valued eigenvalues* $0 = \lambda_0 \leqslant \lambda_1 \leqslant \cdots \leqslant \lambda_{N-1}$.

  Proof We prove each property separately.

  1. We proof that the expression C.0.1 is true.

  $$\begin{aligned} f'Lf &= f'Df - f'Wf \\ &= \sum_{i=1}^{N} d_i f_i^2 - \sum_{i,j=1}^{N} f_i f_j w_{ij} \\ &= \frac{1}{2}\left(\sum_{i=1}^{N} d_i f_i^2 - 2\sum_{i,j=1}^{N} f_i f_j w_{ij} + \sum_{j=1}^{N} d_j f_j^2\right) \qquad \text{(C.0.2)} \\ &= \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2. \qquad \text{(C.0.3)} \end{aligned}$$

In the equality C.0.2 we have applied that $\sum_{i=1}^{N} d_i f_i^2 = \sum_{j=1}^{N} d_j f_j^2$. Finally, in the last equality(C.0.3) we have applied d_i definition (see appendix A) and the square of the subtraction expression.

2. To show that L is a symmetric matrix, we only have to realize that the matrices D and W are symmetric by definition (see appendix A). As $L = D - W$, we can affirm immediately that L is symmetric.
   We could also prove in an easy way that L is positive semidefinite, i.e., that $f'Lf \geqslant 0$.

$$f'Lf = \frac{1}{2}\sum w_{ij}(f_i - f_j)^2 \geqslant 0,$$

as long as

$$\begin{aligned} w_{ij} &\geqslant 0, \text{ by definition.} \\ (f_i - f_j)^2 &\geqslant 0, \text{ for being a square.} \end{aligned}$$

3. To look for the eigenvalues of L we must compute $|L - \lambda I| = 0$.
   If $\lambda = 0$, then

$$\begin{aligned} |L| &= 0 \\ |D - W| &= 0 \\ \left| \begin{pmatrix} d_1 - w_{11} & w_{12} & \cdots \\ & \ddots & \\ & \cdots & d_N - w_{NN} \end{pmatrix} \right| &= 0. \end{aligned}$$

   And this is always true because each element in the diagonal is a linear combination of the rest of the elements that form its row. If we apply this result to our equation to find the corresponding eigenvector, we arrive to the vector $\mathbf{1}$,

$$\begin{aligned} (L - \lambda I)f &= 0. \\ Lf &= 0 \\ (D - W)f &= 0 \\ \begin{pmatrix} d_1 - w_{11} & w_{12} & \cdots & w_{1n} \\ & \ddots & & \\ & & \cdots & d_N - w_{NN} \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ \\ f_N \end{pmatrix} &= 0 \end{aligned}$$

   For each row we have

$$\begin{aligned} d_i w_{ii} f_i + \sum_{j \neq i} w_{ij} f_j &= 0 \\ d_i w_{ii} f_i &= \sum_{j \neq i} w_{ij} f_j \\ \sum_{j \neq i} w_{ij} f_i &= \sum_{j \neq i} w_{ij} f_j \end{aligned}$$

$$f_i = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

4. We have seen before that L is positive semidefinite, so, by definition, their eigenvalues are positive.
   L has always N eigenvalues because $f = (f_1, \ldots, f_N)$, so the dimension of the weight matrix W is $N \times N$. $\square$

We have to notice that the Laplacian Graph does not depend on the elements on the diagonal of W. If we take $U = W\ d_i$ we obtain the same Laplacian matrix L.

Proposition 6 ***Number of connected components in an Unnormalized Laplacian Graph.*** *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue λ_0 of L is the number of connected components $A_1, \ldots, A_k$ in our graph. The eigenspace of the eigenvalue λ_0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}$ of these components.*

Proof First, we assume that $k = 1$, that means that G is a connected graph. Let f be an eigenvector with eigenvalue 0. In this case,

$$0 = f'Lf = \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2$$

As the weights are always non-negative valued,

$$\begin{aligned} \sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2 &= 0 \\ w_{ij}(f_i - f_j)^2 &= 0 \quad \forall i, j. \end{aligned}$$

So, we see that

$$\begin{cases} w_{ij} = 0 & \Rightarrow \mathbf{x}_i \text{ and } \mathbf{x}_j \textit{ are not connected} \\ w_{ij} > 0 & \Rightarrow f_i = f_j \Rightarrow f \textit{ is constant on the whole connected graph.} \end{cases}$$

We study know the case with k connected components. We assume that the vertices are ordered according to the connected components they belong to. In this case, our matrix L will be formed by block diagonal components,

$$L = \begin{pmatrix} L_1 & & & 0 \\ & L_2 & & \\ & & \ddots & \\ 0 & & & L_k \end{pmatrix},$$

where L_i are Laplacian Graphs, each one corresponding to the i-th connected component of G. As L is a block diagonal matrix, its spectrum is given by the union of the spectra of L_i. As each L_i is a Laplacian connected graph, each one have an eigenvalue $\lambda_{i0} = 0$ and its corresponding eigenvector $f_{i0} = \mathbf{1}$. There exist k eigenvalues 0 valued in L. This tell us that L has multiplicity k. $\square$

- **Normalized Laplacian Graph**

We are going to study two types of Normalized Laplacian Graphs. The first one, the **Symmetric Laplacian** is given by the expression

$$\begin{aligned} L_{sym} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \\ &= I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}. \end{aligned}$$

L_{sym} has the property of being a symmetric matrix.

The second one is the called **Random Walk Laplacian**, as it is closely related with random walks.

$$\begin{aligned} L_{rw} &= D^{-1} L \\ &= I - D^{-1} W. \end{aligned}$$

As we have made before with Unnormalized Laplacian Graphs, we present a proposition with the most important properties of these types of graphs.

Proposition 7 ***Properties of*** $\mathbf{L_{sym}}$ ***and*** $\mathbf{L_{rw}}$.

1. *$\forall f \in \mathbb{R}^N$, where f is any vector, it satisfies*

$$f' L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2. \tag{C.0.4}$$

2. *λ is an eigenvalue of L_{rw} with $\mathbf{v}$ its corresponding eigenvector $\Leftrightarrow$ λ is an eigenvalue of L_{sym} with associated eigenvector $\mathbf{w} = D^{\frac{1}{2}} v$.*

3. *λ is an eigenvalue of L_{rw} with $\mathbf{v}$ its corresponding eigenvector $\Leftrightarrow$ λ and $\mathbf{v}$ solves the equation $Lv = \lambda Dv$.*

4. *0 is an eigenvalue of L_{rw} with eigenvector $v = \mathbf{1}$. 0 is also eigenvalue of L_{sym} associated to the eigenvector $w = D^{\frac{1}{2}} \mathbf{1}$.*

5. *L_{sym} and L_{rw} are positive semidefinite matrices and they have N non-negative real-valued eigenvalues $0 = \lambda_0 \leqslant \lambda_1 \leqslant \cdots \leqslant \lambda_N$.*

Proof We are going to prove each property separately.

1. In an analogous form to the one for Unnormalized Laplacian Graph, we proof that the

expression C.0.4 is true.

$$
\begin{aligned}
f'L_{sym}f &= f'(I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}})f \\
&= f^2 - f'D^{-\frac{1}{2}}WD^{-\frac{1}{2}}f \\
&= f^2 - \begin{pmatrix} f_1 & \cdots & f_N \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{d_1}\sqrt{d_1}}w_{11} & \frac{1}{\sqrt{d_1}\sqrt{d_2}}w_{12} & \cdots & \frac{1}{\sqrt{d_1}\sqrt{d_N}}w_{1N} \\ & \ddots & & \\ \frac{1}{\sqrt{d_N}\sqrt{d_1}}w_{N1} & & \cdots & \frac{1}{\sqrt{d_N}\sqrt{d_N}}w_{NN} \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix} \\
&= f^2 - \sum_{i,j=1}^{N} \frac{1}{\sqrt{d_i}\sqrt{d_j}} w_{ij} f_i f_j \\
&= \sum_{i,j=1}^{N} f_i f_i - \frac{1}{\sqrt{d_i}\sqrt{d_j}} w_{ij} f_i f_j \\
&= \frac{1}{2}\left(f_i^2 + f_j^2 - 2\sum_{i,j=1}^{N} \frac{1}{\sqrt{d_i}\sqrt{d_j}} w_{ij} f_i f_j \right) \\
&= \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 .
\end{aligned}
$$

2. We assume that λ is an eigenvalue of L_{sym} and $w = D^{\frac{1}{2}}v$ its corresponding eigenvector.

$$
\begin{aligned}
L_{sym}w &= \lambda w \\
D^{-\frac{1}{2}}L_{sym}w &= D^{-\frac{1}{2}}\lambda w \\
D^{-\frac{1}{2}}D^{-\frac{1}{2}}LD^{-\frac{1}{2}}w &= D^{-\frac{1}{2}}\lambda w \\
D^{-1}LD^{-\frac{1}{2}}D^{\frac{1}{2}}v &= D^{-\frac{1}{2}}\lambda D^{\frac{1}{2}}v \\
D^{-1}Lv &= \lambda v \\
L_{rw}v &= \lambda v.
\end{aligned}
$$

So, λ is an eigenvalue of L_{rw} and v is its corresponding eigenvector.

3. We assume now that λ is an eigenvalue of L_{rw} and v its corresponding eigenvector.

$$
\begin{aligned}
L_{rw}v &= \lambda v \\
DL_{rw}v &= D\lambda v \\
DD^{-1}Lv &= D\lambda v \\
Lv &= \lambda Dv.
\end{aligned}
$$

4. We could easily see that $L_{rw}\mathbf{1} = 0$, because we know that $L\mathbf{1} = 0$ (see proposition 5), and $L_{rw} = D^{-1}L$. We want to see also that $L_{sym}\mathbf{1} = 0$:
   - If 0 is an eigenvalue of L_{rw}, then 0 is an eigenvalue of L_{sym} as we have seen in the previous properties.
   - In the same way, we could say that if $\mathbf{1}$ is an eigenvector of L_{rw} then $D^{\frac{1}{2}}\mathbf{1}$ is an eigenvector of L_{sym}.

5. L_{sym} is positive semidefinite, as $f'L_{sym}f \geqslant 0$. This is true because we have seen that

$$\begin{aligned} f'L_{sym}f &= \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2 \\ w_{ij} &\geqslant 0 \\ \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2 &\geqslant 0. \end{aligned}$$

Let's prove know that L_{rw} is also positive semidefinite. Let be $f = D^{\frac{1}{2}}v$,

$$\begin{aligned} D^{\frac{1}{2}}v'D^{-\frac{1}{2}}LD^{-\frac{1}{2}}D^{\frac{1}{2}}v &\geqslant 0 \\ v'D^{-\frac{1}{2}}D^{-\frac{1}{2}}Lv &\geqslant 0 \\ v'D^{-1}Lv &\geqslant 0 \\ v'L_{rw}v &\geqslant 0. \end{aligned}$$

□

Proposition 8 ***Number of connected components in a Normalized Laplacian Graph.*** *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue λ_0 of L_{sym} and L_{rw} is the number of connected components $A_1, \ldots, A_k$ in our graph. The eigenspace of the eigenvalue λ_0 is spanned by the indicator vectors $\mathbf{1}_{A_i}$ of these components in the case of L_{rw}, and it is spanned by $D^{\frac{1}{2}}\mathbf{1}_{A_i}$ in the case of L_{sym}.*

The proof in this case is analogous to the one for the proposition 6, so we are not going to present it here.

Appendix D

Auxiliar Theorems and Lemmas

D.1 Gauss's Divergence Theorem

Theorem 1 *Let V be a simple solid in $\mathbb{R}^3$ and $S = \partial V$ its boundary, oriented by the outward pointing unit normal field of the boundary S. Let $F : V \to \mathbb{R}^3$ be a vector field C^1 . Then*

$$\int_V divF = \int_S F \cdot n \quad dS.$$

D.2 Rayleigh-Ritz Theorem

Theorem 2 *Let $A \in \mathbb{C}^{n\times n}$ be an hermitic matrix and $R : \mathbb{C}^n \{0\} \to \mathbb{R}$ such that,*

$$R(x) = \frac{x^H Ax}{x^H x}, \text{ with } \|x\| \neq 0.$$

This expression is called the Rayleigh quotien and it defines the eigenvectors as critical points in $\mathbb{R}$. Then

$$\lambda_{max} = \max_{\|x\|\neq 0} R(x)$$
$$\lambda_{min} = \min_{\|x\|\neq 0} R(x).$$

D.3 Parseval's Identity

This theorem is a generalization of the Pythagorean Theorem in separable Hilbert's spaces.

Theorem 3 *Let B be an orthogonal basis in a vectorial space. Then,*

$$\|x\|^2 =< x, x >= \sum_{v\in B} |< x, v >|^2 .$$

D.4 Green's Function

The Green's Function is the kernel of an integral operator. It is useful to solve non-homogeneus boundary value problems.

Let L be the Sturm–Liouville differential operator, that presents the form

$$L = \frac{d}{dx}\left[p(x)\frac{d}{dx}\right] + q(x)$$

and let D be the boundary conditions operator

$$Du = \begin{cases} \alpha_1 u'(0) + \beta_1 u(0) \\ \alpha_2 u'(l) + \beta_2 u(l) \end{cases}$$

Let $f(x)$ be a continuous function in $[0, l]$. We shall also suppose that the problem

$$\begin{aligned} Lu &= f \\ Du &= 0 \end{aligned}$$

is regular (i.e., only the trivial solution exists for the homogeneous problem).

Theorem 4 *There is one and only one solution $u(x)$ which satisfies*

$$\begin{aligned} Lu &= f \\ Du &= 0 \end{aligned}$$

and it is given by

$$u(x) = \int_0^l f(s)G(x,s)ds$$

where $G(x, s)$ is a Green's function satisfying the following conditions:

1. *$G(x, s)$ is continuous in x and s.*
2. *For $x \neq s$, $LG(x, s) = 0$.*
3. *For $s \neq 0$, $DG(x, s) = 0$.*
4. *Derivative "jumps": $G'(s_{+0}, s) - G'(s_{-0}, s) = \frac{1}{p(s)}$.*
5. *Symmetry: $G(x, s) = G(s, x)$.*

D.5 Mercer's theorem

Theorem 5 *Suppose K is a continuous symmetric non-negative definite kernel. Then there is an orthonormal basis $\{e_i\}_i$ of $L^2[a, b]$ consisting of eigenfunctions of TK, the linear operator associated to K, such that the corresponding sequence of eigenvalues $\{\lambda_i\}_i$ is nonnegative. The eigenfunctions corresponding to non-zero eigenvalues are continuous on $[a, b]$ and K has the representation*

$$K(s,t) = \sum_{j=1}^{\infty} \lambda_j e_j(s) e_j(t)$$

where the convergence is absolute and uniform.

Bibliography

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2002.

[3] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

[5] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[6] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2001.

[7] Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux, Jean-François Paiement, Pascal Vincent, Marie Ouimet, Département D'informatique Et Recherche Opérationnelle, and Centre De Recherches Mathématiques. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation*, 16:2004, 2004.

[8] Yoshua Bengio, Pascal Vincent, Jean-François Paiement, Olivier Delalleau, Marie Ouimet, and Nicolas Le Roux. Spectral clustering and kernel PCA are learning eigenfunctions. Technical Report 1239, Département d'informatique et recherche opérationnelle, Université de Montréal, 2003.

[9] M. Belkin. *Problems of Learning on Manifolds*. PhD thesis, The University of Chicago, 2003.

[10] Yoshua Bengio, Jean-Francois Paiement, and Pascal Vincent. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *In Advances in Neural Information Processing Systems*, pages 177–184. MIT Press, 2003.

[11] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 1994.

[12] T. Kohonen. *Self-organization and associative memory*. Springer-Verlag New York, Inc. New York, NY, USA, 1989.

[13] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[14] Rosenberg S. *The Laplacian on a Riemannian manifold*. Cambridge University Press, 1997.

[15] Piotr Indyk. Dimensionality reduction techniques for proximity problems. In *In Proc. 9th SODA*, pages 371–378, 2000.

[16] Dorothea Wagner and Frank Wagner. Between min cut and graph bisection. In *Mathematical Foundations of Computer Science 1993*, volume 711 of *Lecture Notes in Computer Science*, pages 744–750. Springer Berlin / Heidelberg, 1993.

[17] A novel way of computing dissimilarities between nodes of a graph, with application to collaborative ltering and subspace projection of the graph nodes. *Information Systems Research*, 2005.

[18] Ulrike von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. pages 857–864. MIT Press, 2004.

[19] Ulrike Von Luxburg, Olivier Bousquet, and Mikhail Belkin. Limits of spectral clustering. In *In Advances in Neural Information Processing Systems*. MIT Press, 2005.

[20] Francis Bach and Michael I. Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7:1963–2001, 2006.

[21] Christopher Williams and Matthias Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1159–1166. Morgan Kaufmann, 2000.

[22] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[23] V. Koltchinskii and E. Giné. Random matrix approximation of spectra of integral operators. *Bernouilli*, 6:113–167, 2000.

[24] Richard Socher and Supervisor Prof. Matthias Hein. *Manifold Learning and Dimensionality Reduction with Diffusion Maps*. PhD thesis, 2008.

[25] R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, July 2006.