Stockholms universitet

When graph theory meets unsupervised learning: the statistical properties 'spectral clustering'

Fanny Bergström

Masteruppsats i matematisk statistik
Master Thesis in Mathematical Statistics

Masteruppsats 2020:5
Matematisk statistik
Juni 2020

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Stockholm University

Mathematical Statistics
Stockholm University
Master Thesis **2020:5**
http://www.math.su.se

When graph theory meets unsupervised learning: the statistical properties 'spectral clustering'

Fanny Bergström*

June 2020

Abstract

Spectral clustering treats clustering as a graph partitioning problem, where clusters are constructed based on the *commute time distance* (CTD) between the nodes of the graph. The CTD combines the local connections between nearby points to establish the global connections among remote points, allowing us to detect shapes and intrinsic manifold structures buried in high dimensional data. Furthermore, the CTD is simply the Euclidean distance in the space spanned by the eigenvectors of the graph Laplacian. This implies that clusters can be easily detected using a classical clustering algorithm (e.g., k-means) when data is represented in this space. This thesis aims to scrutinize the statistical principles of this nonparametric clustering method, which is robust and manages to capture both local and global geometrical structures in the data. Properties of the CTD and its relations with the clustering structures of the data are investigated extensively.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: fannybergstrom@gmail.com. Supervisor: Chun-Biu Li.

Acknowledgements

I would like to thank Prof. Chun-Biu Li, my supervisor from Stockholm University, for his support and guidance throughout this project. I would also like to thank Prof. Taras Bodnar, Stockholm University, for kindly lending me his office during the months of writing this thesis.

Contents

I. INTRODUCTION

Machine learning can be divided into *supervised* and *unsupervised* learning, where the latter is sometimes referred to as 'learning without a teacher' as it aims to detect unknown structures in data without pre-existing labels. Unsupervised learning is mainly focused on two types of problems; dimensionality reduction and clustering. Dimensionality reduction is the process of reducing the number of dimensions by feature selection or feature extraction, and clustering is the process of dividing a dataset into subsets. Spectral clustering methods have connections to both and can be used to study the structure of networks as it allows us to detect shapes and intrinsic manifold structures buried in high dimensional data. This thesis aims to scrutinize the statistical principles of this method, focusing on the problem of clustering.

Given a distance metric between any pair of data points, clustering is a problem where we want to group data points such that points within clusters are 'similar' and data points in different clusters are 'dissimilar'. The question now arises of how to define a similarity measure, what is the meaning of two points being 'similar'? It is well known that a simple metric, such as the Euclidean distance, fails to capture the underlying nonlinear geometrical structures on which the data points reside, especially in a high-dimensional space.

Spectral clustering treats clustering as a graph partitioning problem, where clusters are constructed based on the *commute time distance* (CTD) between the nodes of the graph. The CTD is the expected time for a random walk to travel from one node to the other and back again. Since all possible paths between the two nodes are considered, the metric is more robust to noises and outliers than if we would only use a single path. This distance metric combines the local connections between nearby points to establish global connections among remote points. This allows us to capture the global intrinsic geometrical structures buried in the high dimensional data. Furthermore, the CTD is simply the Euclidean distance in the space spanned by the eigenvectors of the graph Laplacian. This implies that clusters can be easily detected using a classical clustering algorithm (e.g. k-means) when data is represented in this space.

A. Objectives

This thesis aims to scrutinize the statistical principles of a nonparametric clustering method, called the spectral clustering. The main objectives are to

1. Investigate the properties of the graph Laplacian matrices related to spectral clustering, and how their eigenvalues and eigenvectors relate to the clustering structures of the data.

2. Describe how spectral clustering relates to graph cuts and random walks on graphs, with emphasis on its connection to the CTD between nodes on the graph.

3. Depict how the CTD can be used as a distance metric in cluster-validation methods.

4. Apply the method to various simulated datasets to investigate the abilities and possible drawbacks.

B. Structure of the thesis

The thesis is structured as follows; An introduction to clustering is found in Sec. II. Sec. III describes mathematical objects needed to perform the spectral clustering, the main tools being similarity graphs, and the graph Laplacian matrices. We also state some important properties of the graph Laplacians used in spectral clustering. The essential steps of the algorithm are found in Sec. IV, followed by an intuitive justification of the algorithm. Sec. V shows how spectral clustering can be derived as a graph partitioning problem, and Sec. VI describes how it relates to a random walk on graphs and the CTD. Demonstrative examples of the method using simulated data are found in Sec. VII. Validation in unsupervised learning methods and the use of the CTD as a distance metric in cluster-validation are discussed in Sec. VIII. General discussion and an outlook on further studies are found in Sec. IX.

II. CLUSTERING

Clustering has the purpose of grouping or segmenting a dataset into subsets or 'clusters' such that data points within the same cluster are more related than data points belonging

to different clusters. Clustering can also be used to arrange clusters in a hierarchy, such that clusters are successively grouped so that on each level of the hierarchy, clusters within groups are more similar than other groups. A data point can be described by its features or its relation to other data points. The extent to which a pair of data points are related can be measured by their pairwise *similarity* (or *distance*). Clustering methods attempt to group the objects based on the type of similarity supplied to them. The question now arises how to define an appropriate similarity measure.

A. The k-means clustering

Using a spatial metric, such as the Euclidean distance, is the most intuitive and natural way to describe the distance between two data points [1]. The Euclidean distance describes how the data points are separated in space without considering any other information, e.g. a manifold, beyond the coordinates of the two data points. The Euclidean distance is supplied to one of the most classical clustering methods; the k-means clustering [2]. This clustering method aims to minimize the Euclidean distance of data points to the cluster centers. Let x_i denote the data points in a dataset $X = (x_1, \ldots, x_n)$, and c_j denote cluster j in a set of k partitions of X, $C = (c_1, \ldots, c_k)$. The k-means algorithm constructs the clusters by minimizing the sum of squared distances simultaneously for all clusters

$$J(C) = \sum_{c_j \in C} \sum_{x_i \in c_j} ||x_i - \mu_j||^2, \tag{1}$$

where μ_j is the mean of the data points in c_j. For a given number of clusters k, the k-means algorithm starts with an initial guess of the cluster center. Then the following two steps are repeated until convergence; 1) for each data point the closest center is identified, and 2) the cluster centers are updated, such that they are centered by the data points belonging to the respective cluster.

Fig. 1 shows an example where the k-means algorithm is applied to two simulated datasets. From the figure, one can see that the k-means algorithm successfully manages to separate the spherical clusters found in the left panel. However, when the data points are formed into non-spherical clusters, such as the circles in the right panel, the k-means algorithm fails. The failure of k-means in this simple example originates from the fact that data points belonging to the same cluster, e.g., in the outmost circle in the right panel of Fig.

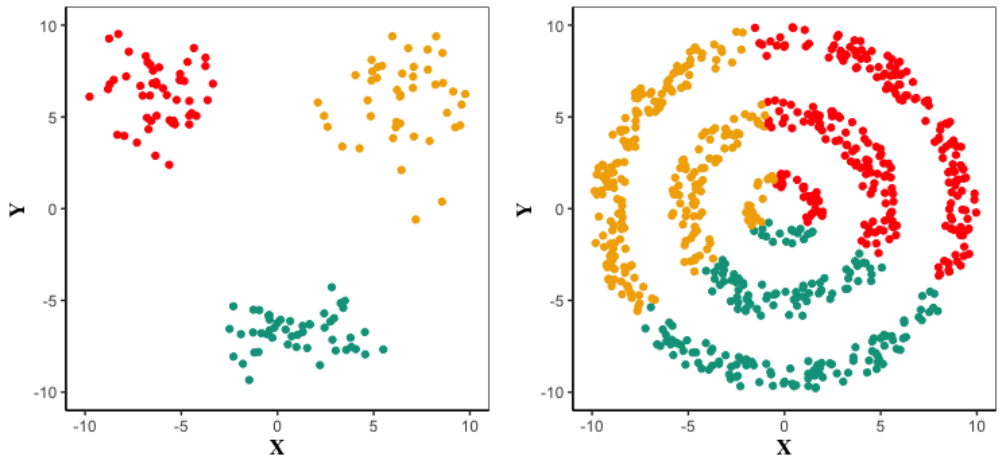

FIG. 1: Example of the k-means algorithm applied to two different datasets. The left panel consists of three spherical clusters generated from two-dimensional normal distributions, showing successful results of the clustering. The right panel has three clusters of circles where the k-means algorithm cannot classify the data points correctly.

1, are not necessarily close to each other in terms of the Euclidean distance. The example illustrates the requirement of an alternative objective function than to simply minimize the Euclidean distance to the cluster centers for the non-spherical clusters, as when clusters are non-convex sets in the feature space. This example is just given in a two-dimensional space, but it is well known that the Euclidean distance fails to capture the underlying nonlinear geometrical structures on which the data points reside, especially in a high-dimensional space [1].

Another drawback of the k-means clustering is its inability to handle clusters of significant imbalance in the number of data points [3]. As the cost function of the k-mean clustering, defined as in Eq. (1), does not aim to minimize the sum of squared distances of each cluster separately, the method tends to assign data points belonging to clusters of a larger number of data points to clusters of fewer points.

B. Clustering for high-dimensional data

As can be seen from Fig. 1, k-means clustering method is appropriate only when the clusters are convex sets in the feature space. This type of classical clustering method is based on the compactness of the clusters as they minimize the distance to the cluster center. If the

aim instead is to maximize the similarities between data points within clusters, we construct clusters based on the connectivity in data, rather than the compactness. Clustering methods based on the connectivity does not make any assumptions of the shape of the clusters, and generally perform better for high-dimensional data or if the clusters are non-convex sets in the feature space. Popular methods that construct clusters based on connectivity are density-based methods (e.g. the DBSCAN algorithm [4]), the ISOMAP [5], local linear embedding [6], spectral clustering, etc.

Spectral clustering dates back to 1973 when Donath and Hoffman [7] suggested to partition graphs based on the eigenvectors of the adjacency matrix. The same year Fiedler [8] connected the second smallest eigenvalue of the graph Laplacian to the connectivity of the graph and suggested partitioning the graph by its second smallest eigenvector.

By representing data in the form of a graph, spectral clustering reformulates the objective of clustering into finding a partition of the graph such that nodes within clusters are connected by strong edge weights, and nodes belonging to different clusters have low edge weights. The distance metric used on the graph to construct the clusters is based on the CTD (see Sec. VI A). The CTD between two nodes on a graph is the expected time it takes for a random walk to transition from one node to the other, and then back again. Furthermore, we will show in Sec. VI A that the CTD is simply the Euclidean distance in the space spanned by the eigenvectors of the graph Laplacian. This implies that clusters can be easily detected using a classical clustering algorithm (e.g. the k-means algorithm) when data is represented in this embedding.

III. SIMILARITY GRAPHS AND THE GRAPH LAPLACIAN

Spectral clustering is a method with roots in spectral graph theory, which can be described as the field that studies properties of graphs in terms of the eigenvalues and eigenvectors of matrices derived from the graphs, the main reference often being the book *Spectral Graph Theory* by Fan Chung [9]. In this section, we introduce how to represent data by a weighted network, also called the similarity graph, constructed from the pairwise similarities (or distances) between data points. In the following, we define matrices derived from the graph; the adjacency matrix, the degree matrix, and the graph Laplacian matrices. In addition, we state some properties of the graph Laplacian matrices associated with spectral clustering.

A. Similarity Graphs

Any set of data points, given their pairwise similarities, can be represented by a weighted network. In a network representation of data, we are only concerned with how the data points, represented by nodes in the network, are related to each other, and not the spatial location of the data point itself. A network representation of data captures the local neighborhood relationships between nodes and can help us detect groups of nodes that are strongly connected.

Given a dataset $X = (x_1, \ldots, x_n) \in \mathcal{R}^d$, and a distance $d_{ij} \geq 0$ with $d_{ij} = d_{ji}$, between all pairs of data points x_i and x_j, we can represent the data points in an undirected, possibly weighted, *similarity graph*. Denote the graph $G = (V, E, W)$, where V is the set of nodes and E is the set of edges, which are pairs of nodes, and W being the edge weights. The graph has nodes $V = (v_1, .., v_n)$, where each node $v_i \in V$ corresponds to a data point $x_i \in X$. The edges are weighted by the pairwise distance between the data points. It follows that the graph is undirected as $d_{ij} = d_{ji}$. In other words, the distances between data points are symmetric and as a direct consequence, the edges in the similarity graph are also symmetric. The nodes are connected if the edge weight is positive or exceeds some threshold. This threshold can be e.g. a distance ϵ between the data points or the k nearest neighbours. One can also consider a fully connected graph, where each node is connected to all the other nodes by weighted edges. The edge weights ω_{ij} are typically computed using the Gaussian similarity

$$\omega_{ij} = \exp\left(\frac{-||x_i - x_j||^2}{2\sigma^2}\right), \tag{2}$$

for $i \neq j$ and $\omega_{ii} = 0$. A justification of this formula was given by Belkin and Niyogi [10]. We note that in the similarity graph, we are only interested in capturing how data points are related given their pairwise similarity, not how similar data points are to themselves; hence the self-similarity is set to 0. The σ in the Gaussian similarity function is a scale parameter defining the extent of the local neighbourhood of a data point that will be discussed in section III A 1.

According to Hastie et al. [2], another popular similarity graph is the *mutual* k-nearest neighbour graph. In the mutual k-nearest neighbour graph node i and j are connected only if i is among the k nearest neighbours of j and vice versa. An overview of some other popular methods to construct similarity graphs is given by von Luxburg [11].

Fig. 2 illustrates three types of similarity graphs created from a dataset of two non-spherical clusters in the shape of two half-moons. The graph seen in b) is a ϵ-neighbourhood graph which only connects data points within a certain distance $\epsilon \geq 0$ from each-other. The graph in c) is a mutual k-nearest neighbour graph ($k = 3$) and the graph in d) is a fully connected graph where the edge weights are given by the Gaussian similarity function. Both the similarity graphs in c) and d) manage to capture the underlying geometrical structure of the moon-shaped clusters.

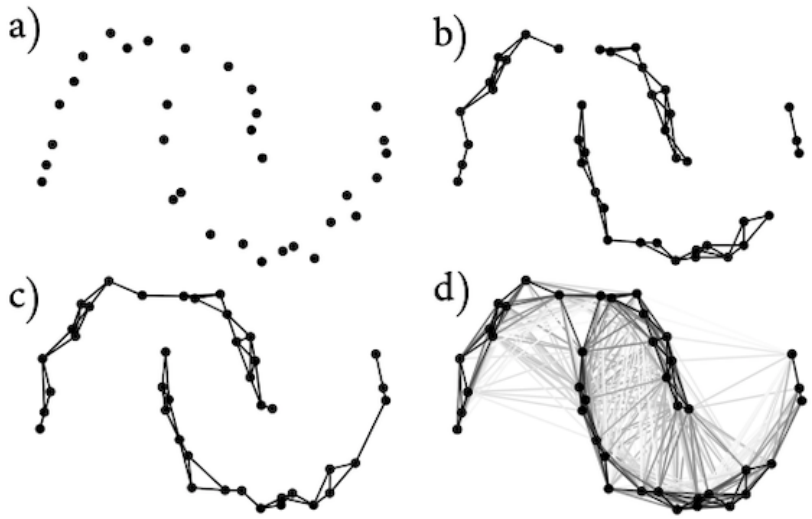

FIG. 2: Example of a dataset with two nonlinear clusters in the shape of half-moons seen in a) and three different similarity graphs. In b), an ϵ-neighbourhood is constructed, resulting in a graph with four connected components. The graph in c) is a mutual k-nearest neighbour graph ($k = 3$) and d) illustrates a fully connected graph with edges weighted by the Gaussian similarity function such that a dark shade illustrates a strong weight.

A graph is connected if there is a path of edges connecting all nodes, e.g. the fully connected graph seen in Fig. 2d) is a connected graph. A *connected component* of a graph is a subset of nodes $A \subset V$ such that there is a path of edges connecting all nodes in A, and no edges to nodes not belonging to A. The graphs in Fig. 2b) and c) have 4 and 2 connected components respectively. A connected graph has exactly one connected component. To use similarity graphs for clustering, we do not need to construct similarity graphs with the same number of connected components as clusters we are trying to identify. Instead, we can use the similarity graph to find partitions of nodes that are strongly connected within the

partition and weakly connected to nodes belonging to other partitions. On the other hand, creating a sparse graph whose nodes only connect to their local neighbours is advantageous as the simple Euclidean distance between data points is unlikely to be a suitable choice of distance metric for points with a large distance, especially in a high dimensional space, or if the dataset has non-spherical clusters. E.g., consider the data points in Figure 2. The data points follow the lines of two half-moons, and their pairwise distances should rather be measured along these lines. The Euclidean distance is not a good measure between data points belong to different lines or residing on different ends of the half-moons but is reasonable when only considering a within a few neighbouring data points as we can assume that the manifold is locally linear [1].

1. Practical considerations: Choice of similarity graph and respective parameters

Choosing the type of similarity graph, and the parameters related to it is not a trivial task. The aim of the similarity graph is to construct a sparse representation of data, capturing the underlying, intrinsic structures in data by the pairwise distances. It can be seen in Fig. 2 that the mutual k-nearest neighbour graph and the fully connected graph managed to capture the geometrical structures by the connections between the data points lying near each other on the manifolds consisting of two half-moons. From the figure, it could also be seen that the ϵ-neighbourhood graph did not capture the underlying structure of the clusters, as data points lying on the same manifold ended up in different partitions of the graph.

The parameters k, ϵ or σ, in the respective similarity graph needs to be chosen with caution such that the local neighbourhood structure in data is captured. Small k or ϵ will lead to a sparse similarity graph, limiting only to data points being within the first few neighbours, or a certain distance from each other, to be considered being similar. The σ in the Gaussian similarity function, and ϵ have a similar function as k. While k is related to the number of data points considered to be close, σ and ϵ are related to the scale of distance between the data points. Choosing the threshold too small will make the graph essentially disconnected. On the other hand, a large k, ϵ, or σ, will not be able to resolve the global nonlinear structures of the dataset as many remote points can be connected by a single edge that does not respect the underlying structures in the dataset. Again considering our

example of the two half-moons in Fig. 2, choosing k, ϵ or σ too large in any of the similarity graph would lead to connecting the two half-moons, such that it simply becomes one strongly connected component.

The ϵ in the ϵ-neighbourhood graph decides the scales of the local geometrical structures in data, as it controls the maximum distance between data points considered to belong to the same neighbourhood. Since there is only one value for ϵ chosen, we assume that the neighbourhoods are formed on the same scale. In other words, the ϵ-neighbourhood graph requires the underlying data points to be as spatially close to their neighbouring data points, in all local geometric structures in data. This is not the case for the mutual k-nearest neighbour graph. As k is not related to the spatial scale, it can connect communities of neighbouring data points on multiple scales. However, a drawback with the mutual k-nearest neighbour graph is that it cannot form connected regions of mixed density. If we consider a dataset with a high-density region with a nearby low-density region, then data points at the low-density region are likely to have nearest neighbours in the high-density region. However, since data points in the high-density region have more close neighbours also belonging to the same high-density region, the two regions will be disconnected. In the case of constructing a fully connected graph, it is proposed by [12] to use an adaptive σ_i for each data point x_i when neighbourhoods of data points have different densities. They suggested extending the Gaussian similarity function to the form

$$\omega_{ij} = \exp\left(\frac{-||x_i - x_j||^2}{2\sigma_i\sigma_j}\right), \tag{3}$$

where each σ is estimated locally. By locally estimating the appropriate scale parameter, one can connect local neighbourhoods on multiple scales and allow the within neighbourhood density to vary.

B. Matrices derived from graphs

There are several matrices associated with a graph, two of them being the edge weight matrix, also called the *adjacency matrix*, and the *degree matrix*. Let $G = (V, E, W)$ be an undirected and weighted graph with a set if n nodes.

Definition 1. *Given a graph G=(V, E, W), the adjacency matrix W is defined*

as

$$W_{ij} = \begin{cases} \omega_{ij} & \text{if } (i,j) \in E \\ 0 & \text{if else.} \end{cases}. \tag{4}$$

and the degree matrix D is defined as the diagonal matrix with diagonal elements

$$d_i = \sum_{k=1}^{n} \omega_{ik}.$$

For an unweighted network ω_{ij} is 1 if node i and j are connected, and 0 if they are not. If the edges are weighed, the adjacency matrix also specifies the extent of the connection. The degree matrix is the diagonal matrix whose elements d_i specifies the sum of edge weights connecting to node i, also called the degree of node i. Both the adjacency matrix and the degree matrix are symmetric and of size $n \times n$.

The *graph Laplacian*, is a matrix representation of a graph derived from D and W, which can be used to find various properties of the graph, extensively studied by Chung [9]. Using the graph Laplacian, we can e.g. construct a low dimensional embedding from high dimensional data (Sec. IV), identify and approximate the smallest cut of a graph by its second smallest eigenvector (Section V). In the following, we will define the *unnormalized* and *normalized* graph Laplacians, and state some of their properties connected to spectral clustering.

1. The unnormalized graph Laplacian

Definition 2. *Given a graph G=(V, E, W), the unnormalized graph Laplacian is defined as*

$$L = D - W, \tag{5}$$

where the adjacency matrix W and degree matrix D are defined as in Def. 1. For any real-valued vector f, L has the following property;

$$\begin{aligned} f'Lf &= f'Df - f'Wf \\ &= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j \omega_{ij} \\ &= \frac{1}{2} \sum_{i,j=1}^{n} \omega_{ij} (f_i - f_j)^2. \end{aligned} \tag{6}$$

Eq. (6) shows that $f'Lf$ have a small value if pairs f_i and f_j with large edge weight ω_{ij} have values that are close. It also shows that L is positive semi-definite, from which it follows that L has n non-negative real-valued eigenvalues $0 = \lambda_1 \leq \cdots \leq \lambda_n$. Since $\mathbb{1}'L\mathbb{1} = 0$, the constant one-vector $\mathbb{1}$ is a trivial eigenvector of L with corresponding eigenvalue 0. If the graph is connected, this is the only 0 eigenvalue. For a graph with k connected components, L has k eigenvectors of eigenvalue 0. These k eigenvectors can be arranged such that L is a block diagonal matrix, with one block for each connected component of the graph [2, p. 547].

2. The normalized graph Laplacian

Two matrices are generally referred to as the *normalized* graph Laplacian.

Definition 3. *Given a graph G=(V, E, W), the normalized graph Laplacian matrices are defined as*

$$\begin{aligned} L_{sym} &= D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2} \\ L_{rw} &= D^{-1}L = I - D^{-1}W, \end{aligned} \tag{7}$$

where L_{sym} is a symmetric matrix and L_{rw} is related to a random walk on graphs (see Sec. VI). For every real-valued vector f we have that

$$\begin{aligned} f'L_{sym}f &= f'f - f'D^{-1/2}WD^{-1/2}f \\ &= \sum_{i=1}^{n} f_i^2 - \sum_{i,j=1}^{n} f_i f_j \frac{\omega_{ij}}{\sqrt{d_i}\sqrt{d_j}} \\ &= \frac{1}{2}\sum_{i,j=1}^{n} \omega_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2. \end{aligned} \tag{8}$$

Eq. (8) shows that $f'L_{sym}f$ will have a small value if the difference between f_i and f_j are small for the pair of nodes i and j with large edge weight ω_{ij}. The same is also true for L_{rw}. The normalized graph Laplacians are positive semi-definite matrices with n non-negative real-valued eigenvalues $0 = \lambda_0 \leq \cdots \leq \lambda_{n-1}$. L_{rw} and L_{sym} share the same eigenvalues, such that λ is an eigenvalue of L_{rw} with corresponding eigenvector v if and only if λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2}v$. As for the unnormalized graph Laplacian, the multiplicity of the eigenvalue 0 of the normalized graph Laplacians corresponds to the number of connected components of the graph.

IV. THE ALGORITHM

There are various spectral clustering algorithms with different cost functions, yielding different results depending on whether one consider the normalized or unnormalized graph Laplacian. However, most algorithms follow the same essential steps. These can be summarized as follows;

- Input: $n \times n$ matrix of pairwise similarities, the number of clusters k.

1. Given the pairwise similarities, the adjacency matrix W and degree matrix D are constructed.

2. From W and D, the graph Laplacian matrix L is computed.

3. The first k eigenvectors of L are computed. The vectors are placed as columns in a $n \times k$ matrix V.

4. Rows of V are used as the coordinates in the new representation of data. Data points are clustered using the k-means algorithm.

5. The original data points are labeled based on the k-means clustering.

The trick of the algorithm is to change the representation of the original data points to the k-dimensional space spanned by the first k eigenvectors of L. This is a type of nonlinear dimensionality reduction, to a space where clusters are linearly separable such that they can be identified by the k-means clustering algorithm.

An illustrative example of the algorithm is shown in Figure 3. The dataset used in the example contains 300 data points separated into 3 clusters of intertwined swirls. A classic clustering algorithm, such as the k-means, would not be able to identify the clusters as they are non-spherical. A fully connected similarity graph was constructed from the data with edges weighted by the Gaussian similarity function with $\sigma = 0.5$. The first eigenvalue of the graph Laplacian is exactly 0 with the corresponding eigenvector the constant vector $\mathbb{1}$, as the graph is fully connected. The second and third following eigenvalues are close to 0, which indicates that three clusters are indeed a suitable number of clusters for this dataset. The right bottom panel of the figure illustrates the space spanned by the first and second eigenvectors, where clusters are now linearly separable and easily detected using e.g. the k-means clustering algorithm.

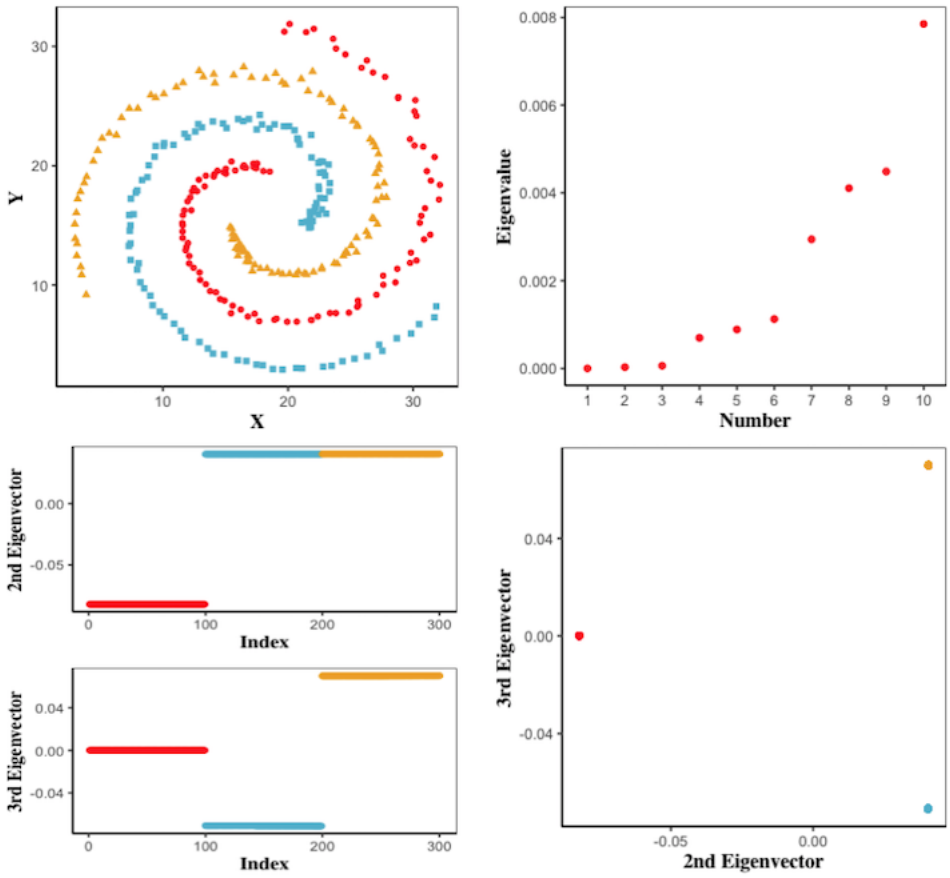

FIG. 3: Illustrative example of spectral clustering. The dataset contains three intertwined swirls seen in the top left panel. The similarity graph is constructed using a fully connected graph, with edge weights given by the Gaussian similarity function with $\sigma = 0.5$. The ten smallest eigenvalues of L can be seen in the top right panel. The first eigenvector corresponds to the constant one vector, as the graph is fully connected. Coordinates of the second and third smallest eigenvector of L is seen in the bottom left panel, colored the same way as in the first figure. The bottom right panel illustrates the new representation of data, given by the space spanned by the eigenvectors in which we can successfully apply the k-means algorithm for clustering.

A. Justification of the algorithm

This section is not aimed to mathematically justify the algorithm, but to provide some intuition. Considering a graph $G = (V, E, W)$ with k connected components. L is the graph Laplacian, and let $L_1, ..., L_k$ be the graph Laplacians of the connected components. Then,

L can be put into the following block form

$$L = \begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{bmatrix}.$$

As stated in Sec. III B 1, if L has k connected components, it also has k eigenvectors associated with the eigenvalue 0. Denote the connected components $A_1, \ldots, A_k$, then the k eigenvectors $v_1, \ldots, v_k$ with eigenvalue 0 will be arranged as following

$$\begin{array}{c} \\ A_1 \\ \\ \\ \\ \\ \\ \\ \\ \\ A_k \end{array} \overset{\begin{array}{c} v_1 \end{array}}{\left\{\begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right.} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \overset{\begin{array}{c} v_k \end{array}}{\left\{\begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right.} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

This illustrates that the eigenvectors form indicatorvectors labeling the sets of data points assigning them to the k clusters. It also shows that creating the $n \times k$ matrix V with the k smallest eigenvectors of L and clustering the rows of V into k clusters by minimizing the Euclidean distance between the rows, yields a clustering result of the k connected components.

In practice, the clusters are in general not exactly the connected components of the graph since the components are likely connected by edges with low edge weights. Then the multiplicity of the eigenvalue 0 will not equal the number of clusters k. However, by continuity, there will still be k eigenvalues close to 0. The rows of V can also be used in the same way; if two nodes belong to the same cluster, their corresponding rows of V should be close in terms of the Euclidean distance. This means that the k-means algorithm can be applied to successfully to identify the clusters when considering the rows of V as the new coordinates of the data points also when the numbers of clusters does not equal to the number of connected components of the graph.

V. GRAPH CUTS PERSPECTIVE

When data is represented in the form of a graph, clustering can be reformulated to a graph partitioning problem. The objective becomes finding a partition of the graph such that edges between nodes within a cluster have substantial weights and edges between nodes in different clusters have low weights. In this section, we will derive spectral clustering based on this graph partitioning problem. We have followed a similar discussion as in *A Tutorial on Spectral Clustering* by Ulrike von Luxburg [11].

For a subset of nodes $A \in V$, and its complement $\bar{A}$, we define the cut between A and $\bar{A}$

$$\text{cut}(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} \omega_{ij}.$$

The cut is simply the weights of the edges connecting nodes in different subsets. The graph cut problem can now be formulated as finding the non-overlapping partition $A_1, \ldots, A_k$, of k subsets of V, minimizing

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

However, in general, this approach does not work well as a clustering algorithm, as it tends to identify individual nodes weakly connected to others (outliers) as clusters. This follows from the fact that an outlier has low edge weights to the other nodes, meaning that isolating this single node would yield the smallest cut. In clustering, one generally does not want any of the clusters to be too small. An approach to overcome this problem is to introduce a balancing criterion on the clusters. Proposed by Shi and Malik [13] is the *normalized cut*, called the Ncut. The Ncut is defined as

$$\text{Ncut}(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)}, \tag{9}$$

where $\text{vol}(A_i) = \sum_{v_j \in A_i} d_j$ is the sum of edge weights of nodes in the subset A_i to all nodes in the graph. Minimizing Ncut means finding a partition of nodes in the graph of relatively small edge weights between different subsets and strong internal edge weights within the subsets. The Ncut avoids creating small clusters of isolated nodes by considering the total edge weights connecting the partition with the rest of the nodes in the graph.

Introducing this balancing criterion makes the problem of finding the minimum cut more complex to solve. In the following, it will be shown how spectral clustering is one way to solve a relaxed (or approximated) version of this problem. There are other objective functions, e.g. the RatioCut, introduced by Hagen and Kahng [14], favoring clusters of a comparable number of nodes. In this thesis, we have chosen to focus on the Ncut relaxation problem and how it relates to the normalized spectral clustering.

For the sake of simplicity, we start with a binary partition of nodes of a graph G into two subsets $A, \bar{A} \subset V$. The goal of the partition is to solve the optimization problem

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

As in von Luxburg [11], we define a cluster indicator vector $f = (f_1, ..., f_n)' \in \mathcal{R}^n$, having entries

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} & \text{if } i \in A \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}} & \text{if } i \in \bar{A}. \end{cases} \tag{10}$$

The Ncut objective function can now be rewritten using the unnormalized graph Laplacian L defined as in Eq. (5)

$$\begin{aligned} f'Lf &= \frac{1}{2}\sum_{i,j=1}^{n} \omega_{ij}(f_i - f_j)^2 \\ &= \frac{1}{2}\sum_{i\in A, j\in \bar{A}} \omega_{ij}\left(\sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} + \sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}}\right)^2 \\ &\qquad + \frac{1}{2}\sum_{i\in \bar{A}, j\in A} \omega_{ij}\left(-\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}} - \sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}}\right)^2 \\ &= \frac{1}{2}\mathrm{vol}(V)\mathrm{cut}(A, \bar{A})\left(\frac{1}{\mathrm{vol}(\bar{A})} + \frac{1}{\mathrm{vol}(A)}\right) \\ &\qquad + \frac{1}{2}\mathrm{vol}(V)\mathrm{cut}(\bar{A}, A)\left(\frac{1}{\mathrm{vol}(\bar{A})} + \frac{1}{\mathrm{vol}(A)}\right) \\ &= \mathrm{vol}(V)\mathrm{cut}(A, \bar{A})\left(\frac{1}{\mathrm{vol}(\bar{A})} + \frac{1}{\mathrm{vol}(A)}\right) \\ &= \mathrm{vol}(V)\mathrm{Ncut}(A, \bar{A}), \end{aligned} \tag{11}$$

where the first equality follows from Eq. (6). A more extensive derivation of Eq. (11) is found in the Appendix Sec. A. Furthermore, given the definition of f in Eq. (10), the vector

Df is orthogonal to the constant one-vector $\mathbb{1}$. This is showed showed by

$$\begin{aligned}(Df)'\mathbb{1} &= \sum_{i=1}^{n} d_i f_i \\ &= \sum_{i\in A} d_i \sqrt{\frac{\text{vol}(\bar{A})}{\text{vol}(A)}} - \sum_{i\in \bar{A}} d_i \sqrt{\frac{\text{vol}(A)}{\text{vol}(\bar{A})}} \\ &= \text{vol}(A)\sqrt{\frac{\text{vol}(\bar{A})}{\text{vol}(A)}} - \text{vol}(\bar{A})\sqrt{\frac{\text{vol}(A)}{\text{vol}(\bar{A})}} = 0.\end{aligned} \tag{12}$$

Also, the constant $f'Df$ equals the volume of the full graph G. This can be seen from

$$\begin{aligned}f'Df &= \sum_{i=1}^{n} d_i f_i^2 \\ &= \sum_{i\in A} d_i \frac{\text{vol}(\bar{A})}{\text{vol}(A)} + \sum_{i\in \bar{A}} d_i \frac{\text{vol}(A)}{\text{vol}(\bar{A})} \\ &= \text{vol}(A)\frac{\text{vol}(\bar{A})}{\text{vol}(A)} + \text{vol}(\bar{A})\frac{\text{vol}(A)}{\text{vol}(\bar{A})} = \text{vol}(V).\end{aligned} \tag{13}$$

Using Eq. (11)-(13), we can rewrite the problem of minimizing Ncut, given a binary partition, as

$$\min_{A\subset V} f'Lf \quad \text{s.t. } f \text{ as in (10)}, \; Df \perp \mathbb{1}, \; f'Df = \text{vol}(V).$$

However, as the elements in f are only allowed to take two particular values, this is a NP-hard optimization problem [15]. If we instead allow any $f \in \mathcal{R}^n$, we get the relaxed optimization problem

$$\min_{f\in\mathcal{R}^n} f'Lf \quad \text{s.t.} \quad Df \perp \mathbb{1}, \; f'Df = \text{vol}(V). \tag{14}$$

By the substitution $f = D^{-1/2}g$, the relaxed optimization problem in Eq. (14) is given by

$$\min_{g\in\mathcal{R}^n} g'D^{-1/2}LD^{-1/2}g \quad \text{s.t.} \quad g \perp D^{1/2}\mathbb{1}, \; ||g||^2 = \text{vol}(V). \tag{15}$$

Given the definition of L_{sym} in Eq. (7), we can further rewrite the minimization problem of Eq. (15) as

$$\min_{g\in\mathcal{R}^n} g'L_{sym}g \quad \text{s.t.} \quad g \perp D^{1/2}\mathbb{1}, \; ||g||^2 = \text{vol}(V), \tag{16}$$

since $g'D^{-1/2}LD^{-1/2}g = g'L_{sym}g$. Shown by Shi and Malik [13], this relaxation of Ncut transforms the mincut problem to an eigenvalue problem, where the solution is given by the second smallest eigenvalue λ from

$$L_{sym}g = D^{-1/2}LD^{-1/2}g = \lambda g.$$

It can easily be shown that $g_1 = D^{1/2}\mathbb{1}$ is the first eigenvector of L_{sym} with eigenvalue 0. For the minimization problem in Eq. (16), one seeks the smallest eigenvalue and corresponding eigenvector, given the constraints, since the problem is to minimize $g' L_{sym} g$.

The smallest eigenvector of L_{sym}, g_1, does not fulfill the first constraint $g \perp D^{1/2}\mathbb{1}$, since $g_1 = D^{1/2}\mathbb{1}$. It was seen in Sec. III B 2 that L_{sym} is positive semi-definite and symmetric and as a consequence its eigenvectors are orthogonal. Hence, g_2, the second eigenvector of L_{sym} , is orthogonal to g_1, and thus the solution to the minimization problem. The second constraint, $||g||^2$, can be seen as a normalization factor, making sure that the volume of the clusters equals the volume of the whole graph.

A simple approach to obtain the clusters is to use the sign of the values of the second eigenvector (also known as the Fiedler vector [8]) as an indicator vector. In other words, data point x_i is assigned to cluster A if $\text{sign}(g_i) > 0$, and $\bar{A}$ if else. Another approach, used in spectral clustering, is to separate the clusters given the position of the gap of the values of the elements in the vector. Fig. 4 shows an example of a graph of two strongly connected groups of nodes with few edges connecting the two groups (left panel), and its corresponding values of the second smallest eigenvector of the normalized (symmetric) graph Laplacian (right panel). The figure shows that the position of the gap in the second smallest eigenvector appears around the same indices where the elements of the vector change signs. This means that both the change in sign and the gap of the values of the second eigenvector of L_{sym} can be used to identify the two clusters in this example.

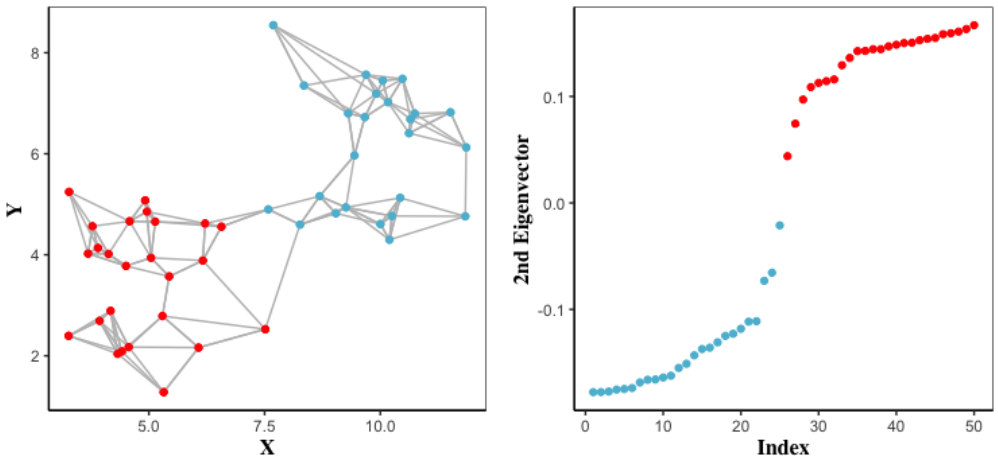

FIG. 4: A connected similarity graph of two Gaussian clusters (left) and the corresponding values of the second smallest eigenvector of the normalized (symmetric) graph Laplacian (right).

When the numbers of clusters are $k > 2$, the solution to the relaxation of Ncut is given by the first k eigenvectors of the graph Laplacian [11]. Using a similar argument as we used above, one can show that the eigenvector with the third smallest eigenvalue is the optimal real-valued solution to further partition the first partition of the graph [13]. As described in Section IV, most spectral algorithms use the values of the eigenvectors as coordinates for f_i as a low-dimensional representation of data, and cluster the data points with the k-means algorithm in this space spanned by the first k eigenvectors.

VI. RANDOM WALKS PERSPECTIVE

In the graph cut point of view of spectral clustering, the relaxation of the Ncut minimization is just an approximate solution. It is not guaranteed that the solution of the relaxed problem is always close to the true solution of the Ncut minimization, see an example in Guattery and Miller [16]. Another point of view to justify spectral clustering is that of a random walk on the similarity graph. In this section, the statistical foundation of spectral clustering is provided from the perspective of finding a partition of the graph such that the random walk stays within the same cluster for a long time and rarely jumps to another cluster.

By a normalisation of the rows in the adjacency matrix W defined as in Eq. (4), we obtain the transition matrix of a random walk

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

where all rows sum to 1 ($\sum_j p_{ij} = 1$). P has elements

$$p_{ij} = \frac{\omega_{ij}}{d_i} = \frac{\omega_{ij}}{\sum_k \omega_{ik}},$$

which means that we can interpret p_{ij} as the conditional probability of a random walk currently at node i transitioning to node j in one step. We note that the edge weights between node i and j are symmetric, but since in general $d_i \neq d_j$, the transition probabilities are not.

The relation of P to the normalized Laplacian L_{rw} defined as in Eq. (7) is $L_{rw} = I - P$. It follows immediately that λ is an eigenvalue L_{rw} if and only if $(1 - \lambda)$ is an eigenvalue of P. Also, L_{rw} and P share the same set of eigenvectors, where the smallest eigenvector of

L_{rw} corresponds to the largest eigenvector of P. In other words, the solution of the relaxed Ncut minimization problem as formulated in Eq. (14) can also be formulated in terms of the transition probabilities of the random walk, a formal equivalence between the two is provided by Maila and Shi [17].

The random walk defined on a similarity graph is a finite Markov chain, where each state of the Markov chain is a node in the similarity graph, and P its transition probability matrix. In a Markov chain, state i and j are said to be communicating if one can jump from i to j with a finite probability. If G is a connected graph, then the Markov chain corresponding to the random walk on G is *irreducible* as all states are communicating. The random walk defined on the graphs in Fig. 2b) and c) are not irreducible as the graphs have two or more connected components where states in different components are not communicating. A Markov chain is *aperiodic* if the largest common divisor of the number of steps in all possible paths from a state to itself is 1. See Serfozo [18, p. 23] for further discussion about aperiodicity. For an irreducible and aperiodic Markov chain (e.g. consider the random walk on the graph of Fig. 2d) there exists a unique stationary distribution $\pi' = (\pi_1, \ldots, \pi_n)$, satisfying $\pi' P = \pi'$. It can easily be shown that π have elements $\pi_i = d_i/\mathrm{vol}(V)$.

Let X_t be the state at which the random walk is at time $t \geq 0$, and $A, \bar{A} \subset V$ two disjoint subsets of nodes in a graph. Suppose one starts a random walk in $t = 0$ according to the stationary distribution π, one can then express the conditional probability of starting in A and transitioning to $\bar{A}$ in one step as

$$\begin{aligned} P(X_1 \in \bar{A} | X_0 \in A) &= \frac{P(X_0 \in A, X_1 \in \bar{A})}{P(X_0 \in A)} \\ &= \sum_{i \in A, j \in \bar{A}} \pi_i p_{ij} \Big/ \sum_{i \in A} \pi_i \\ &= \frac{\sum_{i \in A, j \in \bar{A}} \omega_{ij}}{\mathrm{vol}(V)} \Big/ \frac{\mathrm{vol}(A)}{\mathrm{vol}(V)} = \frac{\mathrm{cut}(A, \bar{A})}{\mathrm{vol}(A)}. \end{aligned}$$

It follows from the definition of Ncut, provided in Eq. (9), that

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

This leads to the interpretation of minimizing Ncut as finding the partition of nodes in a graph such that a random walk have a low probability of transitioning between the partitions.

A. The commute time distance

Another prominent statistical interpretation provided by the random walk viewpoint to the method of spectral clustering is the CTD. The CTD, denoted as c_{ij}, is the expected number of steps in a random walk starting at node i, and returning to the same node after visiting node j only once. In this section, we will establish the CTD connection to spectral clustering, and show the important property of the CTD that it corresponds to the Euclidean distance in the space spanned by the eigenvectors of the graph Laplacian. Therefore, this property justifies the use of the traditional unsupervised learning method, namely the k-means, in the space spanned by the eigenvectors of the graph Laplacian to discover clusters in the spectral clustering algorithm, as discussed in Sec. IV.

Mathematically, the CTD is defined in Lovász [19] as the sum

$$c_{ij} = H(i,j) + H(j,i),$$

where $H(i,j)$ is the expected number of steps it takes to reach node j from node i. We note that the CTD between two vertices decrease if there are many different short ways to get from node i to node j. The CTD have a strong connection to the spectrum of P, and it is showed in Theorem 3.1. in Lovasz [19] that we can express $H(i,j)$ as

$$H(i,j) = \text{vol(V)} \sum_{k=2}^{n} \frac{1}{\lambda_k} \left(\frac{v_{kj}^2}{d_j} - \frac{v_{ki} v_{kj}}{\sqrt{d_i d_j}} \right), \tag{17}$$

where λ_k is the k:th eigenvalue of L_{sym} and v_{ki} is the i:th element of the k:th eigenvector. As a direct consequence of Eq. (17), we obtain the formula of the CTD;

$$c_{ij} = \text{vol}(V) \sum_{k=2}^{n} \frac{1}{\lambda_k} \left(\frac{v_{kj}}{\sqrt{d_j}} - \frac{v_{ki}}{\sqrt{d_i}} \right)^2. \tag{18}$$

This shows that we can consider c_{ij} as the squared Euclidean distance between node i and j in the space spanned by the eigenvectors of L_{sym}. Therefore, k-means clustering using c_{ij} as a distance function is similar to the spectral clustering algorithm. The difference between the two is that spectral clustering only consider the first k eigenvectors while the CTD takes into account all of them, scaled by the inverse eigenvalue of the graph Laplacian. The scaling factor means that the most influential coordinates are those corresponding to the eigenvectors with small eigenvalues. This means that the CTD emphasise the difference of the values of the smallest eigenvectors. This may be used as a motivation why the algorithm

only considers the first few eigenvectors. The approximation of the CTD used in spectral clustering becomes better if there is a gap in the spectrum. This follows from that the dominance of the eigenvectors before the gap will increase, and at the same time, decrease for the eigenvectors after the gap. We further note that in Eq. (18) c_{ij} contains the volume of the graph, vol(V). This is a multiplicative constant; hence it will not affect the relative distances between the nodes and can be disregarded when constructing the clusters.

The CTD quantifies the random walk on the graph by finding a nonlinear transformation that embeds the nodes from the original feature space to the Euclidean space spanned by the eigenvectors of the graph Laplacian. We further note that the knowledge of the original (hidden) feature variables of the datasets is not needed since only the pairwise similarities of the data points are required as input for the whole spectral clustering algorithm. An important property of the CTD, making it a suitable distance metric used for clustering, is that it considers all possible round-trip paths between two nodes, implying that CTD decreases (increases) as the number of round-trip paths increases (decreases).

VII. DEMONSTRATIVE EXAMPLES

Using simulated datasets, this section provides some illustrative examples of the strengths and drawbacks of the spectral clustering. The software used for the implementation of the spectral clustering algorithm in these examples is R [20]. The eigenvectors were computed by the function EIGEN. In the space spanned by the first k eigenvectors, we performed the k-means clustering with the function KMEANS from the STATS package.

A. Imbalanced data

One promise of spectral clustering is its ability to handle clusters of various sizes. In this example, we have two two-dimensional Gaussian clusters with different numbers of data points, seen in the left panel of Fig. 5. The smaller cluster contains 10 data points, and the larger cluster 50 data points. From the spatial data, we created a fully connected similarity graph with edges weighted by the Gaussian similarity function with $\sigma = 2$. The right panel of Fig. 5, shows the values of the second smallest eigenvector of the normalized graph Laplacian. From the figure, it can be seen that the clusters can be identified by the gap in

the second eigenvector's values as discussed in Sec. V.

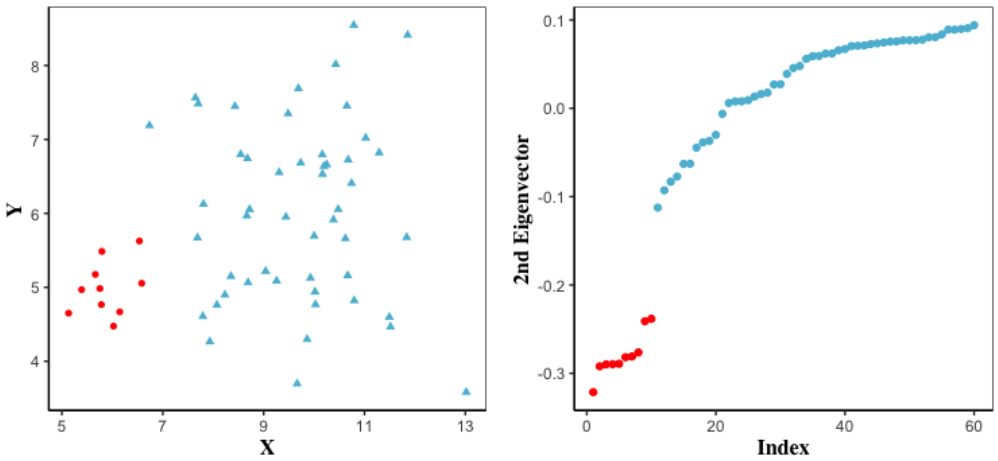

FIG. 5: A dataset of two imbalanced clusters (left) and the corresponding values of the second smallest eigenvector of the normalized graph Laplacian (right).

B. Detecting the number of clusters

As stated in Sec. III B and illustrated in Fig. 3, the spectrum of the graph Laplacian can be used as an indication of how many clusters would be appropriate for the dataset. In this section, we will show that it is also possible to detect hierarchical cluster structures in data from the gaps in the spectrum. We have used the simulated data of four two-dimensional Gaussian distributions, organized in a hierarchical structure, which can be seen in the left panel of Fig. 6. The structure in data is as follows; there are two well separated clusters. Within these two clusters, there are two subgroups that are not as well separated, as shown in Fig 6.

We created a fully connected graph, meaning that the only leading eigenvalue is exactly 0. Looking at the spectrum in the right panel of Fig. 6, we can see that the second eigenvalue is also close to 0. This indicates that one can identify the two main groups of highly connected nodes. After the first and second eigenvalues, there is a gap in the spectrum, followed by another gap after the fourth eigenvalue. This indicates that one can further identify two subgroups within the main groups of strongly connected nodes. The appearance of multiple gaps in the spectrum signifies the existence of a hierarchical cluster structure in the data. As illustrated in this example, we can use the spectrum of the graph Laplacian to detect

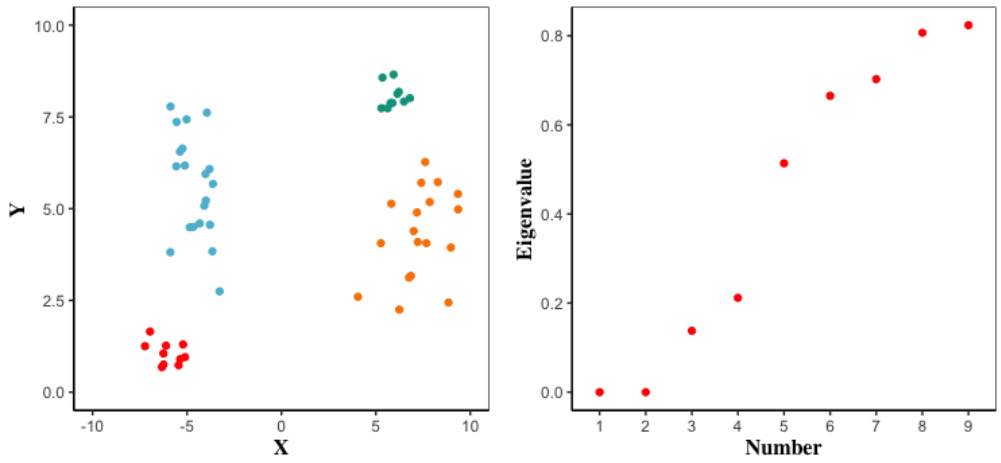

FIG. 6: An example of cluster detection by the graph spectrum. A dataset of two main clusters, divided into two sub-clusters, (left) and the corresponding 10 first eigenvalues of the graph Laplacian (right).

not only the numbers of clusters, but also possible hierarchical organizations in the dataset.

C. Sensitivity to parameter selection

A drawback with the spectral clustering, is the volatility in performance based on the construction of the similarity graph and choosing appropriate parameters related to the graph [11]. Using the dataset of two half-moons from Sec. III A, we will illustrate this sensitivity. From data, we created two similarity graphs, both fully connected but with different scale parameter σ in the Gaussian similarity function. Fig. 7 shows the values of the second eigenvector of the corresponding graph Laplacian, by choosing $\sigma = 0.1$ (left panel) and $\sigma = 0.03$ (right panel). From the figure, we can see that the values substantially changes when we vary σ. The larger σ (left panel), results in a constant increase of the values while choosing a smaller σ creates a big gap among values being below and above 0. This implies that in the first case, when $\sigma = 0.1$, we define the local neighbourhood as large enough to include both moon-shaped clusters. In the latter case, choosing $\sigma = 0.03$, we define a smaller neighbourhood and give stronger connections to fewer neighbouring data points, by which we manage to capture the underlying geometrical structures in data. The gap implies that we have two groups of nodes that are strongly connected within the groups. In our example, this shows that the smaller parameter value for σ is the more suitable choice.

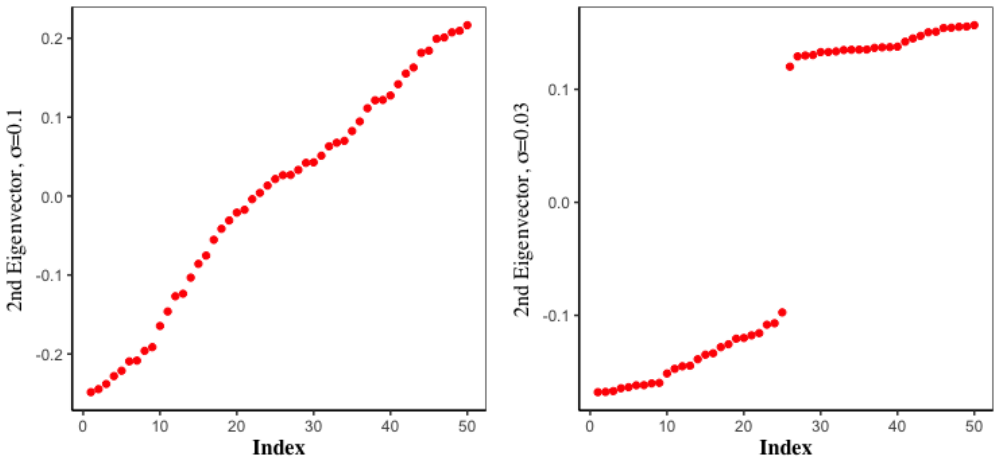

FIG. 7: An example of the sensitivity in parameter selection in spectral clustering. Shown are the values of the second largest eigenvector of the graph Laplacian of a fully connected similarity graph constructed from a dataset of two moon-shaped clusters. In the first case the Gaussian similarity has been used with $\sigma = 0.1$ (left), and in the second case $\sigma = 0.03$ (right).

Furthermore, this illustrates the use of the spectrum for the choice of σ, where an indication of a suitable σ would yield a spectrum of one, or multiple, gaps.

VIII. VALIDATION

In supervised learning, we have a 'teacher' who can tell us what the correct answer should be. It follows that there are clear evaluation measures for supervised methods. One can measure the success of a supervised learning method by e.g. a loss function or by cross-validation [2]. In unsupervised problems, like clustering, there is no direct answer to the correct results, so we need to use alternative methods to evaluate the outcome of these methods. According to Tan et al. [21], there are four main aspects of cluster validation; 1) distinguish whether there is a clustering tendency in data, 2) determine the number of clusters, 3) evaluate the clusters, and 4) compare sets of clusters.

Spectral methods provide a natural way to answer the aspects of clustering tendency and number of clusters. As described in Sec. III B, the spectrum of the graph Laplacian entails the connectivity of the graph, such that the multiplicity of the eigenvalue zero equals the number of connected components. By continuity, the number of clusters can be detected by the eigenvalues close to zero, even for a connected graph. This was shown in the illustrative

example of the algorithm in Fig. 3. As could be seen in Sec. VII B, we can also use the gaps in the spectrum to detect hierarchical, intrinsic cluster structures in data.

One approach to validate the clusters is by the correlation between the similarity matrix, and the *incident matrix*. The incident matrix is the $n \times n$ symmetric matrix with elements I_{ij} equal to 1 if data points i and j belong to the same cluster, and 0 if else. The Pearson correlation coefficient is computed for the two matrices. As the two matrices are symmetric, only $n * (n-1)/2$ elements are compared. If we consider the distance as the similarity matrix elements, a correlation coefficient of -1 is associated with data points belonging to the same cluster being similar. In other words, a correlation of -1 would indicate a successful clustering. If we would simply use the Euclidean distance as the elements in our similarity matrix, this validation approach would only be appropriate if the clusters were spherical symmetric. If we instead would replace the Euclidean distance in the similarity matrix with the CTD, we would also be able to capture possible geometrical structures data making it a suitable method for cluster validation of clusters of arbitrary shapes.

Spectral clustering was applied to a dataset consisting of three non-spherical clusters in the shape of three circles, seen in Fig. 8. As can be seen from the figure, the clusters

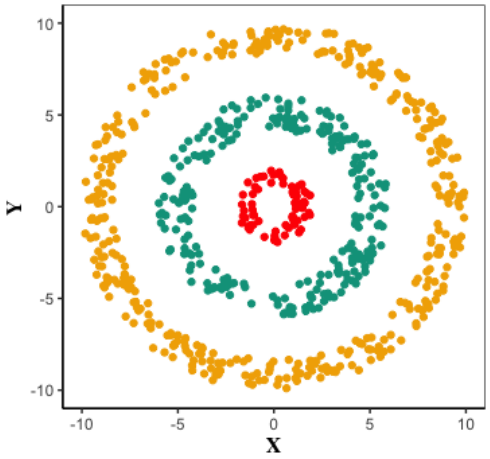

FIG. 8: A dataset of three non-spherical clusters. The spectral clustering algorithm successfully separated the clusters. In the algorithm, we used the normalized graph Laplacian, calculated from a fully connected graph with $\sigma = 1$ in the Gaussian similarity.

are well defined and detected using spectral clustering. By validating the clustering by the correlation with a similarity matrix defined by the Euclidean distance, the result is a Pearson correlation coefficient of 0.09. This indicates a poor clustering result as it indicates that there

is no correlation between data points being close in the Euclidean space and belonging to the same cluster. In other words, it does not entail the information that the three clusters were successfully identified. By replacing the Euclidean distance by the CTD, one would get a correlation coefficient of -0.72, showing that using the CTD in the correlation analysis would be a more appropriate measure in the validation of clusters that are non-convex sets in the feature space.

IX. DISCUSSION AND CONCLUSIONS

In this thesis, we give an introduction to the unsupervised learning method, the spectral clustering. By representing data points in the form of a weighted graph, spectral clustering reformulates the problem of clustering to a graph cut problem. This is a method especially well suited when the data is not linearly separable or forms spherical clusters. It is also a suitable method when the data points in a cluster reside on a low-dimensional manifold in a higher dimensional embedding.

Sec. II contained an introduction to the problem of clustering. The popular clustering algorithm, the k-means, was discussed and we illustrated how it is only a suitable method when clusters are linearly separable or spherical symmetric (Fig. 1). We further discussed the need for alternative methods more suitable for high-dimensional data or if the clusters are non-convex sets in the feature space, e.g. density-based methods [4], ISOMAP [6], and spectral clustering. In Sec. III, the main tools in spectral clustering, similarity graphs and the graph Laplacian matrices were presented. We described some of the spectral and algebraic properties of the graph Laplacians associated with spectral clustering. An extensive study of the graph Laplacians and their properties has been made by Chung [9].

The essential steps in spectral clustering algorithms were shown in Sec. IV, followed by an intuitive justification of the algorithm in Sec. IV A. There is not a single spectral clustering algorithm, and the outcome will differ depending on whether the normalized or unnormalized graph Laplacian is used. A discussion of the comparison of different spectral clustering algorithms can be found in von Luxburg [11]. In Sec. V, we derived spectral clustering as a relaxation of the problem of minimization of graph mincuts. We focused on the objective function Ncut [13] and how it relates to the normalized graph Laplacian. There are other objective functions in graph cuts that can also be linked to spectral clustering,

such as e.g. the RatioCut [14] and the MinMaxCut [22]. Then, we considered the statistical foundation of spectral clustering in terms of the random walk perspective in Sec. VI. We showed how spectral clustering finds a nonlinear transformation of data that embeds the nodes of the similarity graph in a Euclidean space, in which the associated Euclidean distance between the nodes corresponds to the CTD of a random walk on the similarity graph. This illustrated that the k-means algorithm applied to the CTD as a distance metric is equivalent to applying the k-means to the Euclidean distance in the space spanned by the eigenvectors of the graph Laplacians. In Sec. VI A, we concluded that the CTD is an appropriate distance metric in discovering clusters of arbitrary shapes since it considers all possible round-trip paths between two nodes, implying that CTD decreases (increases) as the number of round-trip paths increases (decreases). As a result, the CTD takes into account the underlying geometrical structures in data. This makes the CTD more robust to noises and outliers than if only a single path were considered, e.g. the shortest path between two nodes in the graph or the spatial distance (the Euclidean distance), that does not take into account the underlying geometrical structures of the data. In Sec. VII, we illustrated some strengths and drawbacks of spectral clustering by using some simulated data. The examples demonstrated the ability of clustering imbalanced data, how the spectrum of the graph Laplacian can be used to identify the number of clusters, and the sensitivity of parameter selection. Cluster validation and how the CTD can be used as a distance metric in one of these methods were discussed in section VIII.

An advantage of spectral clustering is its ability to handle clusters of arbitrary shape, and we have seen throughout the thesis examples of moon-shaped clusters, intertwined swirls, etc. We have assumed that the data points are represented in some d-dimensional Euclidean space. E.g., the Gaussian similarity function (defined as in Eq. (2)) is only applicable when this is the case. However, there are several situations where data is naturally represented as pairwise relationships rather than spatial data points (e.g. in social networks), where spectral clustering is highly applicable, see e.g. [23]. In other words, since spectral clustering only considers the pairwise relationships, it does not require a data representation of featured data points but only the pairwise distances.

One of the drawbacks of spectral clustering, which has not yet been discussed in this thesis, is that the method could be computationally costly if one were to calculate the eigenvectors of large matrices. This issue can be resolved by representing the data in a

sparse similarity graph (see Sec. III A) or only compute the first few leading eigenvectors (e.g. the Lanczos method proposed by Golub and Van Loan [24]). Another approach would be to use other methods to first detect communities in the similarity graph, e.g. using the k-clique method proposed by Farkas et al. [25]. Then the spectral clustering can be applied to the detected communities to either further detect sub-communities or compare and classify them based on their network structural difference, see e.g. Willis and Meyer [26] for metrics of graph comparison including metrics based on the graph Laplacian.

A. Outlook on further studies

In this thesis, mainly the clustering aspects of spectral clustering were investigated. However, as seen in the illustrative example of the algorithm (Fig. 3, Sec. IV) the main trick in spectral clustering is to consider the rows of the $n \times k$ matrix with the first k eigenvectors of the graph Laplacian as columns as the new coordinates of the data points. In other words, there is a nonlinear dimensionality reduction step before applying the k-means algorithm in this new k-dimensional space spanned by the first k eigenvectors of the graph Laplacian. A natural continuation of the study, after this thesis, would be the investigation of nonlinear dimensionality reduction techniques using the CTD, and more generally the random walk perspectives, as a distance preservation criterion for the new representation of data, in comparison with some other well-known distance preserving dimensionality reduction methods, such as the locally linear embedding (LLE) [6], diffusion maps [27], Laplacian Eigenmaps [28], etc.

Another interesting direction of further studies is the use of the graph Laplacian in the study of networks, not only to detect groups of nodes that are highly connected, but also to use the graph Laplacian for descriptive purposes, such as shape matching [29], shape segmentation [30], shape recognition [31], etc. In other words, the graph Laplacian can be used to study several properties of graphs; topology, connectivity, etc. Much beyond what we have been able to cover in the scope of this thesis.

[1] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*. Berlin, Heidelberg: Springer, 2007.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2 ed., 2008.

[3] H. Xiong, J. Wu, and J. Chen, "K-means Clustering Versus Validation Measures: A Data-Distribution Perspective," *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 318–331, 2009.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD, p. 226–231, AAAI Press, 1996.

[5] J. B. Tenenbaum, V. Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[6] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[7] W. Donath and A. Hoffman, "Lower Bounds for the Partitioning of Graphs," *IBM Journal of Research and Development*, vol. 17, no. 5, pp. 420–425, 1973.

[8] M. Fiedler, "Algebraic Connectivity of Graphs," *Czechoslovak Mathematical Journal*, vol. 23, pp. 298–305, 1973.

[9] F. Chung, *Spectral Graph Theory*. American Mathematical Soc., 1997.

[10] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.

[11] U. von Luxburg, "A Tutorial on Spectral Clustering," *Statistics and Computing*, vol. 17, 2007.

[12] L. Zelnik-Manor and P. Perona, "Self-Tuning Spectral Clustering," in *Advances in Neural Information Processing Systems 17* (L. K. Saul, Y. Weiss, and L. Bottou, eds.), MIT Press, 2005.

[13] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[14] L. Hagen and A. B. Kahng, "New Spectral Methods for Ratio Cut Partition and Clustering," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 9, pp. 1074–1085, 1992.

[15] D. Wagner and F. Wagner, "Between Min Cut and Graph Bisection," in *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, MFCS, pp. 774–750, MIT Press, 1993.

[16] S. Guattery and G. L. Miller, "On the Quality of Spectral Separators," *SIAM Journal on Matrix Analysis and Applications*, vol. 19, no. 3, pp. 701–719, 1998.

[17] M. Maila and J. Shi, "A Random Walks View of Spectral Segmentation," in *AISTATS*, 2001.

[18] R. Serfozo, *Basics of Applied Stochastic Processes*. Berlin Heidelberg: Springer, 2009.

[19] L. Lovász, "Random Walks on Graphs: A Survey," in *Combinatorics, Paul Erdős is Eighty*, vol. 2, pp. 1–46, János Bolyai Mathematical Society, 1993.

[20] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.

[21] P. N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*. Pearson, 2 ed., 2018.

[22] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon, "Spectral Relaxation for k-means Clustering," in *NIPS* (T. G. Dietterich, S. Becker, and Z. Ghahramani, eds.), pp. 1057–1064, MIT Press, 2001.

[23] P. Symeonidis and N. Mantas, "Spectral Clustering for Link Prediction in Social Networks with Positive and Negative Links," *Social Network Analysis and Mining*, vol. 3, pp. 1433–1447, 2013.

[24] G. H. Golub and C. F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, 3 ed., 1996.

[25] I. Farkas, D. Ábel, G. Palla, and T. Vicsek, "Weighted Network Modules," *New Journal of Physics*, vol. 9, no. 6 (180), 2007.

[26] P. Wills and F. G. Meyer, "Metrics for Graph Comparison: A Practitioner's Guide," *PLoS One*, vol. 15, no. 2 (e0228728), 2020.

[27] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric Diffusions as a Tool for Harmonic Analysis and Structure Definition of Data: Diffusion Maps," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 21, pp. 7426–7431, 2005.

[28] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," in *Advances in Neural Information Processing Systems 14* (T. G. Dietterich,

S. Becker, and Z. Ghahramani, eds.), pp. 585–591, MIT Press, 2002.

[29] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer, "Articulated Shape Matching Using Laplacian Eigenfunctions and Unsupervised Point Registration," in *Conference on Computer Vision and Pattern Recognition*, (Anchorage, United States), pp. 1–8, IEEE Computer Society, 2008.

[30] A. Sharma, E. von Lavante, and R. Horaud, "Learning Shape Segmentation Using Constrained Spectral Clustering and Probabilistic Label Transfer," in *Proceedings of the 11th European Conference on Computer Vision: Part V*, ECCV, (Berlin, Heidelberg), pp. 743–756, Springer, 2010.

[31] A. Tatsuma and M. Aono, "Multi-Fourier Spectra Descriptor and Augmentation with Spectral Clustering for 3D Shape Retrieval," *Vis. Comput.*, vol. 25, no. 8, pp. 785–804, 2009.

Appendix

A. Eq. (11) extended

Given the vector f, as defined in Eq. (10), and the unnormalized graph Laplacian L, defined in Eq. (5), we show how the objective function of Ncut, defined in Eq. (9), can be rewritten as $f'Lf$.

$$
\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 \omega_{ij} \\
&= \frac{1}{2}\Big(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j \omega_{ij} + \sum_{j=1}^{n} d_j f_j^2 \Big) \\
&= \frac{1}{2}\sum_{i,j=1}^{n} \omega_{ij}(f_i - f_j)^2 \\
&= \frac{1}{2}\sum_{i\in A, j\in \bar{A}} \omega_{ij}\left(\sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} + \sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}} \right)^2 + \frac{1}{2}\sum_{i\in \bar{A}, j\in A} \omega_{ij}\left(-\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})}} - \sqrt{\frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)}} \right)^2 \\
&= \frac{1}{2}\sum_{i\in A, j\in \bar{A}} \omega_{ij}\left(\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})} + \frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)} + 2 \right) + \frac{1}{2}\sum_{i\in \bar{A}, j\in A} \omega_{ij}\left(\frac{\mathrm{vol}(A)}{\mathrm{vol}(\bar{A})} + \frac{\mathrm{vol}(\bar{A})}{\mathrm{vol}(A)} + 2 \right) \\
&= \frac{1}{2}\sum_{i\in A, j\in \bar{A}} \omega_{ij}\left(\frac{\mathrm{vol}(A) + \mathrm{vol}(\bar{A})}{\mathrm{vol}(\bar{A})} + \frac{\mathrm{vol}(\bar{A}) + \mathrm{vol}(A)}{\mathrm{vol}(A)} \right) \\
&\qquad\qquad + \frac{1}{2}\sum_{i\in \bar{A}, j\in A} \omega_{ij}\left(\frac{\mathrm{vol}(A) + \mathrm{vol}(\bar{A})}{\mathrm{vol}(\bar{A})} + \frac{\mathrm{vol}(\bar{A}) + \mathrm{vol}(A)}{\mathrm{vol}(A)} \right) \\
&= \frac{1}{2}\sum_{i\in A, j\in \bar{A}} \omega_{ij}\left(\frac{\mathrm{vol}(V)}{\mathrm{vol}(\bar{A})} + \frac{\mathrm{vol}(V)}{\mathrm{vol}(A)} \right) + \frac{1}{2}\sum_{i\in \bar{A}, j\in A} \omega_{ij}\left(\frac{\mathrm{vol}(V)}{\mathrm{vol}(\bar{A})} + \frac{\mathrm{vol}(V)}{\mathrm{vol}(A)} \right) \\
&= \frac{1}{2}\mathrm{vol}(V)\mathrm{cut}(A,\bar{A})\left(\frac{1}{\mathrm{vol}(\bar{A})} + \frac{1}{\mathrm{vol}(A)} \right) + \frac{1}{2}\mathrm{vol}(V)\mathrm{cut}(\bar{A},A)\left(\frac{1}{\mathrm{vol}(\bar{A})} + \frac{1}{\mathrm{vol}(A)} \right) \\
&= \mathrm{vol}(V)\mathrm{cut}(A,\bar{A})\left(\frac{1}{\mathrm{vol}(\bar{A})} + \frac{1}{\mathrm{vol}(A)} \right) \\
&= \mathrm{vol}(V)\mathrm{Ncut}(A,\bar{A}).
\end{aligned}
$$