A Tutorial on Spectral Clustering

Ulrike von Luxburg
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tübingen, Germany
ulrike.luxburg@tuebingen.mpg.de

This article appears in Statistics and Computing, 17 (4), 2007.
The original publication is available at www.springer.com.

Abstract

In recent years, spectral clustering has become one of the most popular modern clustering algorithms. It is simple to implement, can be solved efficiently by standard linear algebra software, and very often outperforms traditional clustering algorithms such as the k-means algorithm. On the first glance spectral clustering appears slightly mysterious, and it is not obvious to see why it works at all and what it really does. The goal of this tutorial is to give some intuition on those questions. We describe different graph Laplacians and their basic properties, present the most common spectral clustering algorithms, and derive those algorithms from scratch by several different approaches. Advantages and disadvantages of the different spectral clustering algorithms are discussed.

Keywords: spectral clustering; graph Laplacian

1 Introduction

Clustering is one of the most widely used techniques for exploratory data analysis, with applications ranging from statistics, computer science, biology to social sciences or psychology. In virtually every scientific field dealing with empirical data, people attempt to get a first impression on their data by trying to identify groups of "similar behavior" in their data. In this article we would like to introduce the reader to the family of spectral clustering algorithms. Compared to the "traditional algorithms" such as k-means or single linkage, spectral clustering has many fundamental advantages. Results obtained by spectral clustering often outperform the traditional approaches, spectral clustering is very simple to implement and can be solved efficiently by standard linear algebra methods.

This tutorial is set up as a self-contained introduction to spectral clustering. We derive spectral clustering from scratch and present different points of view to why spectral clustering works. Apart from basic linear algebra, no particular mathematical background is required by the reader. However, we do not attempt to give a concise review of the whole literature on spectral clustering, which is impossible due to the overwhelming amount of literature on this subject. The first two sections are devoted to a step-by-step introduction to the mathematical objects used by spectral clustering: similarity graphs in Section 2, and graph Laplacians in Section 3. The spectral clustering algorithms themselves will be presented in Section 4. The next three sections are then devoted to explaining why those algorithms work. Each section corresponds to one explanation: Section 5 describes a graph partitioning approach, Section 6 a random walk perspective, and Section 7 a perturbation theory approach. In Section 8 we will study some practical issues related to spectral clustering, and discuss various extensions and literature related to spectral clustering in Section 9.

2 Similarity graphs

Given a set of data points $x_1, \dots x_n$ and some notion of similarity $s_{ij} \geq 0$ between all pairs of data points x_i and x_j, the intuitive goal of clustering is to divide the data points into several groups such that points in the same group are similar and points in different groups are dissimilar to each other. If we do not have more information than similarities between data points, a nice way of representing the data is in form of the *similarity graph* $G = (V, E)$. Each vertex v_i in this graph represents a data point x_i. Two vertices are connected if the similarity s_{ij} between the corresponding data points x_i and x_j is positive or larger than a certain threshold, and the edge is weighted by s_{ij}. The problem of clustering can now be reformulated using the similarity graph: we want to find a partition of the graph such that the edges between different groups have very low weights (which means that points in different clusters are dissimilar from each other) and the edges within a group have high weights (which means that points within the same cluster are similar to each other). To be able to formalize this intuition we first want to introduce some basic graph notation and briefly discuss the kind of graphs we are going to study.

2.1 Graph notation

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{v_1, \dots, v_n\}$. In the following we assume that the graph G is weighted, that is each edge between two vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$. The weighted *adjacency matrix* of the graph is the matrix $W = (w_{ij})_{i,j=1,\dots,n}$. If $w_{ij} = 0$ this means that the vertices v_i and v_j are not connected by an edge. As G is undirected we require $w_{ij} = w_{ji}$. The degree of a vertex $v_i \in V$ is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}.$$

Note that, in fact, this sum only runs over all vertices adjacent to v_i, as for all other vertices v_j the weight w_{ij} is 0. The *degree matrix* D is defined as the diagonal matrix with the degrees $d_1, \dots, d_n$ on the diagonal. Given a subset of vertices $A \subset V$, we denote its complement $V \setminus A$ by $\overline{A}$. We define the indicator vector $\mathbb{1}_A = (f_1, \dots, f_n)' \in \mathbb{R}^n$ as the vector with entries $f_i = 1$ if $v_i \in A$ and $f_i = 0$ otherwise. For convenience we introduce the shorthand notation $i \in A$ for the set of indices $\{i \mid v_i \in A\}$, in particular when dealing with a sum like $\sum_{i \in A} w_{ij}$. For two not necessarily disjoint sets $A, B \subset V$ we define

$$W(A, B) := \sum_{i \in A, j \in B} w_{ij}.$$

We consider two different ways of measuring the "size" of a subset $A \subset V$:

$$|A| := \text{ the number of vertices in } A$$
$$\text{vol}(A) := \sum_{i \in A} d_i.$$

Intuitively, $|A|$ measures the size of A by its number of vertices, while $\text{vol}(A)$ measures the size of A by summing over the weights of all edges attached to vertices in A. A subset $A \subset V$ of a graph is connected if any two vertices in A can be joined by a path such that all intermediate points also lie in A. A subset A is called a connected component if it is connected and if there are no connections between vertices in A and $\overline{A}$. The nonempty sets $A_1, \dots, A_k$ form a partition of the graph if $A_i \cap A_j = \emptyset$ and $A_1 \cup \dots \cup A_k = V$.

2.2 Different similarity graphs

There are several popular constructions to transform a given set $x_1, \ldots, x_n$ of data points with pairwise similarities s_{ij} or pairwise distances d_{ij} into a graph. When constructing similarity graphs the goal is to model the local neighborhood relationships between the data points.

The ε-neighborhood graph: Here we connect all points whose pairwise distances are smaller than ε. As the distances between all connected points are roughly of the same scale (at most ε), weighting the edges would not incorporate more information about the data to the graph. Hence, the ε-neighborhood graph is usually considered as an unweighted graph.

k-nearest neighbor graphs: Here the goal is to connect vertex v_i with vertex v_j if v_j is among the k-nearest neighbors of v_i. However, this definition leads to a directed graph, as the neighborhood relationship is not symmetric. There are two ways of making this graph undirected. The first way is to simply ignore the directions of the edges, that is we connect v_i and v_j with an undirected edge if v_i is among the k-nearest neighbors of v_j *or* if v_j is among the k-nearest neighbors of v_i. The resulting graph is what is usually called *the k-nearest neighbor graph*. The second choice is to connect vertices v_i and v_j if both v_i is among the k-nearest neighbors of v_j *and* v_j is among the k-nearest neighbors of v_i. The resulting graph is called the *mutual k-nearest neighbor graph*. In both cases, after connecting the appropriate vertices we weight the edges by the similarity of their endpoints.

The fully connected graph: Here we simply connect all points with positive similarity with each other, and we weight all edges by s_{ij}. As the graph should represent the local neighborhood relationships, this construction is only useful if the similarity function itself models local neighborhoods. An example for such a similarity function is the Gaussian similarity function $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$, where the parameter σ controls the width of the neighborhoods. This parameter plays a similar role as the parameter ε in case of the ε-neighborhood graph.

All graphs mentioned above are regularly used in spectral clustering. To our knowledge, theoretical results on the question how the choice of the similarity graph influences the spectral clustering result do not exist. For a discussion of the behavior of the different graphs we refer to Section 8.

3 Graph Laplacians and their basic properties

The main tools for spectral clustering are graph Laplacian matrices. There exists a whole field dedicated to the study of those matrices, called spectral graph theory (e.g., see Chung, 1997). In this section we want to define different graph Laplacians and point out their most important properties. We will carefully distinguish between different variants of graph Laplacians. Note that in the literature there is no unique convention which matrix exactly is called "graph Laplacian". Usually, every author just calls "his" matrix the graph Laplacian. Hence, a lot of care is needed when reading literature on graph Laplacians.

In the following we always assume that G is an undirected, weighted graph with weight matrix W, where $w_{ij} = w_{ji} \geq 0$. When using eigenvectors of a matrix, we will not necessarily assume that they are normalized. For example, the constant vector $\mathbb{1}$ and a multiple $a\mathbb{1}$ for some $a \neq 0$ will be considered as the same eigenvectors. Eigenvalues will always be ordered increasingly, respecting multiplicities. By "the first k eigenvectors" we refer to the eigenvectors corresponding to the k smallest eigenvalues.

3.1 The unnormalized graph Laplacian

The unnormalized graph Laplacian matrix is defined as

$$L = D - W.$$

An overview over many of its properties can be found in Mohar (1991, 1997). The following proposition summarizes the most important facts needed for spectral clustering.

Proposition 1 (Properties of L) *The matrix L satisfies the following properties:*

1. *For every vector $f \in \mathbb{R}^n$ we have*

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

2. *L is symmetric and positive semi-definite.*
3. *The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbb{1}$.*
4. *L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.*

Proof.
Part (1): By the definition of d_i,

$$f'Lf = f'Df - f'Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$
$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2\right) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Part (2): The symmetry of L follows directly from the symmetry of W and D. The positive semi-definiteness is a direct consequence of Part (1), which shows that $f'Lf \geq 0$ for all $f \in \mathbb{R}^n$.
Part (3): Obvious.
Part (4) is a direct consequence of Parts (1) - (3). □

Note that the unnormalized graph Laplacian does not depend on the diagonal elements of the adjacency matrix W. Each adjacency matrix which coincides with W on all off-diagonal positions leads to the same unnormalized graph Laplacian L. In particular, self-edges in a graph do not change the corresponding graph Laplacian.

The unnormalized graph Laplacian and its eigenvalues and eigenvectors can be used to describe many properties of graphs, see Mohar (1991, 1997). One example which will be important for spectral clustering is the following:

Proposition 2 (Number of connected components and the spectrum of L) *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, \ldots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of those components.*

Proof. We start with the case $k = 1$, that is the graph is connected. Assume that f is an eigenvector with eigenvalue 0. Then we know that

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

As the weights w_{ij} are non-negative, this sum can only vanish if all terms $w_{ij}(f_i - f_j)^2$ vanish. Thus, if two vertices v_i and v_j are connected (i.e., $w_{ij} > 0$), then f_i needs to equal f_j. With this argument we can see that f needs to be constant for all vertices which can be connected by a path in the graph. Moreover, as all vertices of a connected component in an undirected graph can be connected by a path, f needs to be constant on the whole connected component. In a graph consisting of only one connected component we thus only have the constant one vector $\mathbb{1}$ as eigenvector with eigenvalue 0, which obviously is the indicator vector of the connected component.

Now consider the case of k connected components. Without loss of generality we assume that the vertices are ordered according to the connected components they belong to. In this case, the adjacency matrix W has a block diagonal form, and the same is true for the matrix L:

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

Note that each of the blocks L_i is a proper graph Laplacian on its own, namely the Laplacian corresponding to the subgraph of the i-th connected component. As it is the case for all block diagonal matrices, we know that the spectrum of L is given by the union of the spectra of L_i, and the corresponding eigenvectors of L are the eigenvectors of L_i, filled with 0 at the positions of the other blocks. As each L_i is a graph Laplacian of a connected graph, we know that every L_i has eigenvalue 0 with multiplicity 1, and the corresponding eigenvector is the constant one vector on the i-th connected component. Thus, the matrix L has as many eigenvalues 0 as there are connected components, and the corresponding eigenvectors are the indicator vectors of the connected components. □

3.2 The normalized graph Laplacians

There are two matrices which are called normalized graph Laplacians in the literature. Both matrices are closely related to each other and are defined as

$$L_{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$
$$L_{\text{rw}} := D^{-1} L = I - D^{-1} W.$$

We denote the first matrix by L_{sym} as it is a symmetric matrix, and the second one by L_{rw} as it is closely related to a random walk. In the following we summarize several properties of L_{sym} and L_{rw}. The standard reference for normalized graph Laplacians is Chung (1997).

Proposition 3 (Properties of L_{sym} and L_{rw}) *The normalized Laplacians satisfy the following properties:*

1. *For every $f \in \mathbb{R}^n$ we have*

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

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

3. *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ and u solve the generalized eigenproblem $Lu = \lambda D u$.*

4. *0 is an eigenvalue of L_{rw} with the constant one vector $\mathbb{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbb{1}$.*

5. *L_{sym} and L_{rw} are positive semi-definite and have n non-negative real-valued eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_n$.*

Proof. Part (1) can be proved similarly to Part (1) of Proposition 1.
Part (2) can be seen immediately by multiplying the eigenvalue equation $L_{\text{sym}}w = \lambda w$ with $D^{-1/2}$ from the left and substituting $u = D^{-1/2}w$.
Part (3) follows directly by multiplying the eigenvalue equation $L_{\text{rw}}u = \lambda u$ with D from the left.
Part (4): The first statement is obvious as $L_{\text{rw}}\mathbb{1} = 0$, the second statement follows from (2).
Part (5): The statement about L_{sym} follows from (1), and then the statement about L_{rw} follows from (2). □

As it is the case for the unnormalized graph Laplacian, the multiplicity of the eigenvalue 0 of the normalized graph Laplacian is related to the number of connected components:

Proposition 4 (Number of connected components and spectra of L_{sym} and L_{rw}) *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of both L_{rw} and L_{sym} equals the number of connected components $A_1, \ldots, A_k$ in the graph. For L_{rw}, the eigenspace of 0 is spanned by the indicator vectors $\mathbb{1}_{A_i}$ of those components. For L_{sym}, the eigenspace of 0 is spanned by the vectors $D^{1/2}\mathbb{1}_{A_i}$.*

Proof. The proof is analogous to the one of Proposition 2, using Proposition 3. □

4 Spectral Clustering Algorithms

Now we would like to state the most common spectral clustering algorithms. For references and the history of spectral clustering we refer to Section 9. We assume that our data consists of n "points" $x_1, \ldots, x_n$ which can be arbitrary objects. We measure their pairwise similarities $s_{ij} = s(x_i, x_j)$ by some similarity function which is symmetric and non-negative, and we denote the corresponding similarity matrix by $S = (s_{ij})_{i,j=1\ldots n}$.

Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k eigenvectors $u_1, \ldots, u_k$ of L.**
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

There are two different versions of normalized spectral clustering, depending which of the normalized

graph Laplacians is used. We name both algorithms after two popular papers, for more references and history please see Section 9.

Normalized spectral clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k generalized eigenvectors $u_1, \ldots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.**
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

Note that this algorithm uses the generalized eigenvectors of L, which according to Proposition 3 correspond to the eigenvectors of the matrix L_{rw}. So in fact, the algorithm works with eigenvectors of the normalized Laplacian L_{rw}, and hence is called normalized spectral clustering. The next algorithm also uses a normalized Laplacian, but this time the matrix L_{sym} instead of L_{rw}. As we will see, this algorithm needs to introduce an additional row normalization step which is not needed in the other algorithms. The reasons will become clear in Section 7.

Normalized spectral clustering according to Ng, Jordan, and Weiss (2002)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the normalized Laplacian L_{sym}.
- **Compute the first k eigenvectors $u_1, \ldots, u_k$ of L_{sym}.**
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- **Form the matrix $T \in \mathbb{R}^{n \times k}$ from U by normalizing the rows to norm 1,** that is set $t_{ij} = u_{ij} / (\sum_k u_{ik}^2)^{1/2}$.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of T.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

All three algorithms stated above look rather similar, apart from the fact that they use three different graph Laplacians. In all three algorithms, the main trick is to change the representation of the abstract data points x_i to points $y_i \in \mathbb{R}^k$. It is due to the properties of the graph Laplacians that this change of representation is useful. We will see in the next sections that this change of representation enhances the cluster-properties in the data, so that clusters can be trivially detected in the new representation. In particular, the simple k-means clustering algorithm has no difficulties to detect the clusters in this new representation. Readers not familiar with k-means can read up on this algorithm in numerous

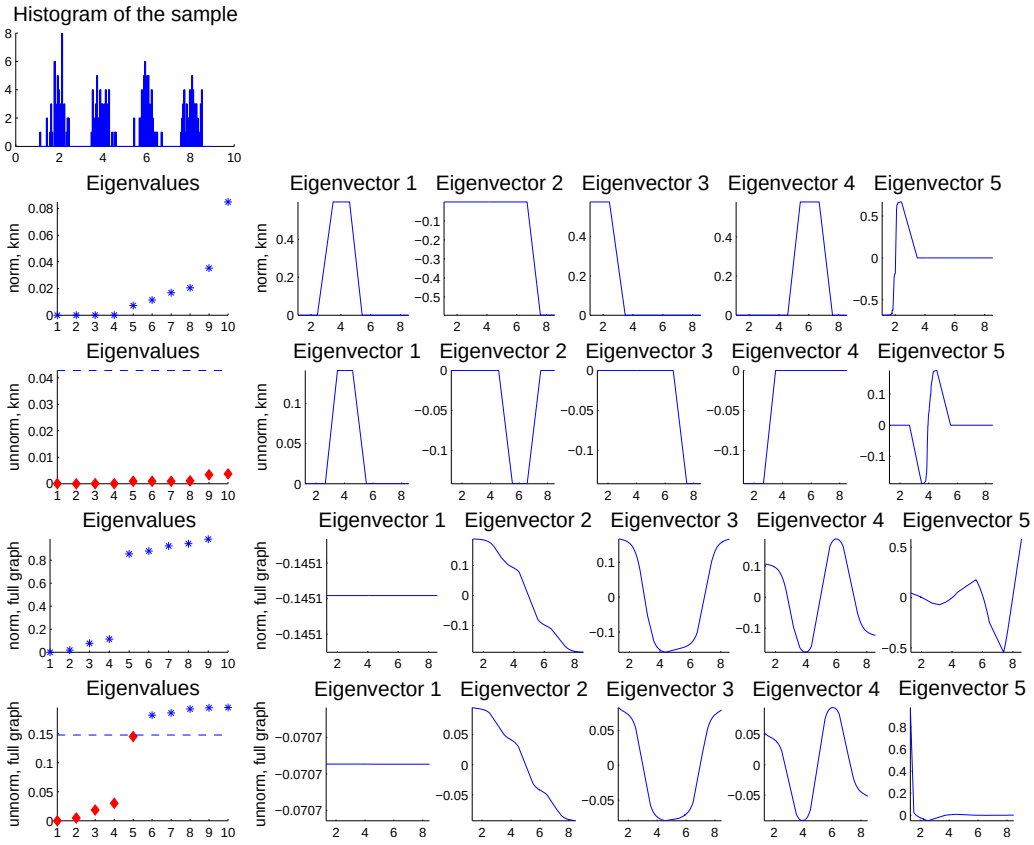

Figure 1: Toy example for spectral clustering where the data points have been drawn from a mixture of four Gaussians on $\mathbb{R}$. Left upper corner: histogram of the data. First and second row: eigenvalues and eigenvectors of L_{rw} and L based on the k-nearest neighbor graph. Third and fourth row: eigenvalues and eigenvectors of L_{rw} and L based on the fully connected graph. For all plots, we used the Gaussian kernel with $\sigma = 1$ as similarity function. See text for more details.

text books, for example in Hastie, Tibshirani, and Friedman (2001).

Before we dive into the theory of spectral clustering, we would like to illustrate its principle on a very simple toy example. This example will be used at several places in this tutorial, and we chose it because it is so simple that the relevant quantities can easily be plotted. This toy data set consists of a random sample of 200 points $x_1, \ldots, x_{200} \in \mathbb{R}$ drawn according to a mixture of four Gaussians. The first row of Figure 1 shows the histogram of a sample drawn from this distribution (the x-axis represents the one-dimensional data space). As similarity function on this data set we choose the Gaussian similarity function $s(x_i, x_j) = \exp(-|x_i - x_j|^2/(2\sigma^2))$ with $\sigma = 1$. As similarity graph we consider both the fully connected graph and the 10-nearest neighbor graph. In Figure 1 we show the first eigenvalues and eigenvectors of the unnormalized Laplacian L and the normalized Laplacian L_{rw}. That is, in the eigenvalue plot we plot i vs. λ_i (for the moment ignore the dashed line and the different shapes of the eigenvalues in the plots for the unnormalized case; their meaning will be discussed in Section 8.5). In the eigenvector plots of an eigenvector $u = (u_1, \ldots, u_{200})'$ we plot x_i vs. u_i (note that in the example chosen x_i is simply a real number, hence we can depict it on the x-axis). The first two rows of Figure 1 show the results based on the 10-nearest neighbor graph. We can see that the first four eigenvalues are 0, and the corresponding eigenvectors are cluster indicator vectors. The reason is that the clusters

form disconnected parts in the 10-nearest neighbor graph, in which case the eigenvectors are given as in Propositions 2 and 4. The next two rows show the results for the fully connected graph. As the Gaussian similarity function is always positive, this graph only consists of one connected component. Thus, eigenvalue 0 has multiplicity 1, and the first eigenvector is the constant vector. The following eigenvectors carry the information about the clusters. For example in the unnormalized case (last row), if we threshold the second eigenvector at 0, then the part below 0 corresponds to clusters 1 and 2, and the part above 0 to clusters 3 and 4. Similarly, thresholding the third eigenvector separates clusters 1 and 4 from clusters 2 and 3, and thresholding the fourth eigenvector separates clusters 1 and 3 from clusters 2 and 4. Altogether, the first four eigenvectors carry all the information about the four clusters. In all the cases illustrated in this figure, spectral clustering using k-means on the first four eigenvectors easily detects the correct four clusters.

5 Graph cut point of view

The intuition of clustering is to separate points in different groups according to their similarities. For data given in form of a similarity graph, this problem can be restated as follows: we want to find a partition of the graph such that the edges between different groups have a very low weight (which means that points in different clusters are dissimilar from each other) and the edges within a group have high weight (which means that points within the same cluster are similar to each other). In this section we will see how spectral clustering can be derived as an approximation to such graph partitioning problems.

Given a similarity graph with adjacency matrix W, the simplest and most direct way to construct a partition of the graph is to solve the mincut problem. To define it, please recall the notation $W(A, B) := \sum_{i \in A, j \in B} w_{ij}$ and $\overline{A}$ for the complement of A. For a given number k of subsets, the mincut approach simply consists in choosing a partition $A_1, \ldots, A_k$ which minimizes

$$\text{cut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A}_i).$$

Here we introduce the factor $1/2$ for notational consistency, otherwise we would count each edge twice in the cut. In particular for $k = 2$, mincut is a relatively easy problem and can be solved efficiently, see Stoer and Wagner (1997) and the discussion therein. However, in practice it often does not lead to satisfactory partitions. The problem is that in many cases, the solution of mincut simply separates one individual vertex from the rest of the graph. Of course this is not what we want to achieve in clustering, as clusters should be reasonably large groups of points. One way to circumvent this problem is to explicitly request that the sets $A_1, \ldots, A_k$ are "reasonably large". The two most common objective functions to encode this are RatioCut (Hagen and Kahng, 1992) and the normalized cut Ncut (Shi and Malik, 2000). In RatioCut, the size of a subset A of a graph is measured by its number of vertices $|A|$, while in Ncut the size is measured by the weights of its edges $\text{vol}(A)$. The definitions are:

$$\text{RatioCut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A}_i)}{|A_i|}$$

$$\text{Ncut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{\text{vol}(A_i)} = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A}_i)}{\text{vol}(A_i)}.$$

Note that both objective functions take a small value if the clusters A_i are not too small. In particular, the minimum of the function $\sum_{i=1}^{k} (1/|A_i|)$ is achieved if all $|A_i|$ coincide, and the minimum of $\sum_{i=1}^{k} (1/\text{vol}(A_i))$ is achieved if all $\text{vol}(A_i)$ coincide. So what both objective functions try to achieve is that the clusters are "balanced", as measured by the number of vertices or edge weights, respectively. Unfortunately, introducing balancing conditions makes the previously simple to solve mincut problem

become NP hard, see Wagner and Wagner (1993) for a discussion. Spectral clustering is a way to solve relaxed versions of those problems. We will see that relaxing Ncut leads to normalized spectral clustering, while relaxing RatioCut leads to unnormalized spectral clustering (see also the tutorial slides by Ding (2004)).

5.1 Approximating RatioCut for $k = 2$

Let us start with the case of RatioCut and $k = 2$, because the relaxation is easiest to understand in this setting. Our goal is to solve the optimization problem

$$\min_{A \subset V} \text{RatioCut}(A, \overline{A}). \tag{1}$$

We first rewrite the problem in a more convenient form. Given a subset $A \subset V$ we define the vector $f = (f_1, \ldots, f_n)' \in \mathbb{R}^n$ with entries

$$f_i = \begin{cases} \sqrt{|\overline{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\overline{A}|} & \text{if } v_i \in \overline{A}. \end{cases} \tag{2}$$

Now the RatioCut objective function can be conveniently rewritten using the unnormalized graph Laplacian. This is due to the following calculation:

$$\begin{aligned} f'Lf &= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \\ &= \frac{1}{2} \sum_{i \in A, j \in \overline{A}} w_{ij} \left(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}} \right)^2 + \frac{1}{2} \sum_{i \in \overline{A}, j \in A} w_{ij} \left(-\sqrt{\frac{|\overline{A}|}{|A|}} - \sqrt{\frac{|A|}{|\overline{A}|}} \right)^2 \\ &= \text{cut}(A, \overline{A}) \left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2 \right) \\ &= \text{cut}(A, \overline{A}) \left(\frac{|A| + |\overline{A}|}{|A|} + \frac{|A| + |\overline{A}|}{|\overline{A}|} \right) \\ &= |V| \cdot \text{RatioCut}(A, \overline{A}). \end{aligned}$$

Additionally, we have

$$\sum_{i=1}^{n} f_i = \sum_{i \in A} \sqrt{\frac{|\overline{A}|}{|A|}} - \sum_{i \in \overline{A}} \sqrt{\frac{|A|}{|\overline{A}|}} = |A| \sqrt{\frac{|\overline{A}|}{|A|}} - |\overline{A}| \sqrt{\frac{|A|}{|\overline{A}|}} = 0.$$

In other words, the vector f as defined in Equation (2) is orthogonal to the constant one vector $\mathbb{1}$. Finally, note that f satisfies

$$\|f\|^2 = \sum_{i=1}^{n} f_i^2 = |A| \frac{|\overline{A}|}{|A|} + |\overline{A}| \frac{|A|}{|\overline{A}|} = |\overline{A}| + |A| = n.$$

Altogether we can see that the problem of minimizing (1) can be equivalently rewritten as

$$\min_{A \subset V} f'Lf \text{ subject to } f \perp \mathbb{1},\ f_i \text{ as defined in Eq. (2)},\ \|f\| = \sqrt{n}. \tag{3}$$

This is a discrete optimization problem as the entries of the solution vector f are only allowed to take two particular values, and of course it is still NP hard. The most obvious relaxation in this setting is

to discard the discreteness condition and instead allow that f_i takes arbitrary values in $\mathbb{R}$. This leads to the relaxed optimization problem

$$\min_{f \in \mathbb{R}^n} f' L f \text{ subject to } f \perp \mathbb{1},\ \|f\| = \sqrt{n}. \tag{4}$$

By the Rayleigh-Ritz theorem (e.g., see Section 5.5.2. of Lütkepohl, 1997) it can be seen immediately that the solution of this problem is given by the vector f which is the eigenvector corresponding to the second smallest eigenvalue of L (recall that the smallest eigenvalue of L is 0 with eigenvector $\mathbb{1}$). So we can approximate a minimizer of RatioCut by the second eigenvector of L. However, in order to obtain a partition of the graph we need to re-transform the real-valued solution vector f of the relaxed problem into a discrete indicator vector. The simplest way to do this is to use the sign of f as indicator function, that is to choose

$$\begin{cases} v_i \in A & \text{if } f_i \geq 0 \\ v_i \in \overline{A} & \text{if } f_i < 0. \end{cases}$$

However, in particular in the case of $k > 2$ treated below, this heuristic is too simple. What most spectral clustering algorithms do instead is to consider the coordinates f_i as points in $\mathbb{R}$ and cluster them into two groups $C, \overline{C}$ by the k-means clustering algorithm. Then we carry over the resulting clustering to the underlying data points, that is we choose

$$\begin{cases} v_i \in A & \text{if } f_i \in C \\ v_i \in \overline{A} & \text{if } f_i \in \overline{C}. \end{cases}$$

This is exactly the *unnormalized spectral clustering* algorithm for the case of $k = 2$.

5.2 Approximating RatioCut for arbitrary k

The relaxation of the RatioCut minimization problem in the case of a general value k follows a similar principle as the one above. Given a partition of V into k sets $A_1, \ldots, A_k$, we define k indicator vectors $h_j = (h_{1,j}, \ldots, h_{n,j})'$ by

$$h_{i,j} = \begin{cases} 1/\sqrt{|A_j|} & \text{if } v_i \in A_j \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, \ldots, n;\ j = 1, \ldots, k). \tag{5}$$

Then we set the matrix $H \in \mathbb{R}^{n \times k}$ as the matrix containing those k indicator vectors as columns. Observe that the columns in H are orthonormal to each other, that is $H'H = I$. Similar to the calculations in the last section we can see that

$$h_i' L h_i = \frac{\text{cut}(A_i, \overline{A}_i)}{|A_i|}.$$

Moreover, one can check that

$$h_i' L h_i = (H'LH)_{ii}.$$

Combining those facts we get

$$\text{RatioCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} h_i' L h_i = \sum_{i=1}^{k} (H'LH)_{ii} = \text{Tr}(H'LH),$$

where Tr denotes the trace of a matrix. So the problem of minimizing $\mathrm{RatioCut}(A_1, \ldots, A_k)$ can be rewritten as

$$\min_{A_1,\ldots,A_k} \mathrm{Tr}(H'LH) \text{ subject to } H'H = I, \ H \text{ as defined in Eq. (5).}$$

Similar to above we now relax the problem by allowing the entries of the matrix H to take arbitrary real values. Then the relaxed problem becomes:

$$\min_{H \in \mathbb{R}^{n \times k}} \mathrm{Tr}(H'LH) \ \text{ subject to } H'H = I.$$

This is the standard form of a trace minimization problem, and again a version of the Rayleigh-Ritz theorem (e.g., see Section 5.2.2.(6) of Lütkepohl, 1997) tells us that the solution is given by choosing H as the matrix which contains the first k eigenvectors of L as columns. We can see that the matrix H is in fact the matrix U used in the unnormalized spectral clustering algorithm as described in Section 4. Again we need to re-convert the real valued solution matrix to a discrete partition. As above, the standard way is to use the k-means algorithms on the rows of U. This leads to the general unnormalized spectral clustering algorithm as presented in Section 4.

5.3 Approximating Ncut

Techniques very similar to the ones used for RatioCut can be used to derive normalized spectral clustering as relaxation of minimizing Ncut. In the case $k = 2$ we define the cluster indicator vector f by

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(\overline{A})}{\mathrm{vol}\, A}} & \text{if } v_i \in A \\ -\sqrt{\frac{\mathrm{vol}(A)}{\mathrm{vol}(\overline{A})}} & \text{if } v_i \in \overline{A}. \end{cases} \tag{6}$$

Similar to above one can check that $(Df)'\mathbb{1} = 0$, $f'Df = \mathrm{vol}(V)$, and $f'Lf = \mathrm{vol}(V)\,\mathrm{Ncut}(A, \overline{A})$. Thus we can rewrite the problem of minimizing Ncut by the equivalent problem

$$\min_A f'Lf \ \text{ subject to } \ f \text{ as in (6)}, \ Df \perp \mathbb{1}, \ f'Df = \mathrm{vol}(V). \tag{7}$$

Again we relax the problem by allowing f to take arbitrary real values:

$$\min_{f \in \mathbb{R}^n} f'Lf \ \text{ subject to } \ Df \perp \mathbb{1}, \ f'Df = \mathrm{vol}(V). \tag{8}$$

Now we substitute $g := D^{1/2}f$. After substitution, the problem is

$$\min_{g \in \mathbb{R}^n} g'D^{-1/2}LD^{-1/2}g \ \text{ subject to } \ g \perp D^{1/2}\mathbb{1}, \ \|g\|^2 = \mathrm{vol}(V). \tag{9}$$

Observe that $D^{-1/2}LD^{-1/2} = L_{\mathrm{sym}}$, $D^{1/2}\mathbb{1}$ is the first eigenvector of L_{sym}, and $\mathrm{vol}(V)$ is a constant. Hence, Problem (9) is in the form of the standard Rayleigh-Ritz theorem, and its solution g is given by the second eigenvector of L_{sym}. Re-substituting $f = D^{-1/2}g$ and using Proposition 3 we see that f is the second eigenvector of L_{rw}, or equivalently the generalized eigenvector of $Lu = \lambda Du$.

For the case of finding $k > 2$ clusters, we define the indicator vectors $h_j = (h_{1,j}, \ldots, h_{n,j})'$ by

$$h_{i,j} = \begin{cases} 1/\sqrt{\mathrm{vol}(A_j)} & \text{if } v_i \in A_j \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, \ldots, n; \ j = 1, \ldots, k). \tag{10}$$

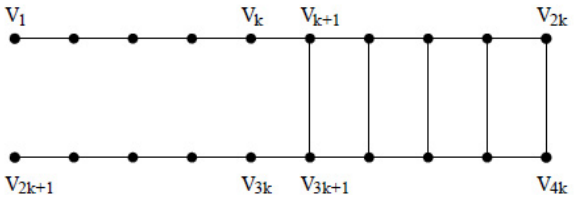

Figure 2: The cockroach graph from Guattery and Miller (1998).

Then we set the matrix H as the matrix containing those k indicator vectors as columns. Observe that $H'H = I$, $h_i' D h_i = 1$, and $h_i' L h_i = \mathrm{cut}(A_i, \overline{A}_i)/\mathrm{vol}(A_i)$. So we can write the problem of minimizing Ncut as

$$\min_{A_1,\ldots,A_k} \mathrm{Tr}(H'LH) \text{ subject to } H'DH = I, \quad \text{H as in (10)} .$$

Relaxing the discreteness condition and substituting $T = D^{1/2}H$ we obtain the relaxed problem

$$\min_{T \in \mathbb{R}^{n \times k}} \mathrm{Tr}(T' D^{-1/2} L D^{-1/2} T) \text{ subject to } T'T = I. \tag{11}$$

Again this is the standard trace minimization problem which is solved by the matrix T which contains the first k eigenvectors of L_{sym} as columns. Re-substituting $H = D^{-1/2}T$ and using Proposition 3 we see that the solution H consists of the first k eigenvectors of the matrix L_{rw}, or the first k generalized eigenvectors of $Lu = \lambda Du$. This yields the normalized spectral clustering algorithm according to Shi and Malik (2000).

5.4 Comments on the relaxation approach

There are several comments we should make about this derivation of spectral clustering. Most importantly, there is no guarantee whatsoever on the quality of the solution of the relaxed problem compared to the exact solution. That is, if $A_1, \ldots, A_k$ is the exact solution of minimizing RatioCut, and $B_1, \ldots, B_k$ is the solution constructed by unnormalized spectral clustering, then $\mathrm{RatioCut}(B_1, \ldots, B_k) - \mathrm{RatioCut}(A_1, \ldots, A_k)$ can be arbitrary large. Several examples for this can be found in Guattery and Miller (1998). For instance, the authors consider a very simple class of graphs called "cockroach graphs". Those graphs essentially look like a ladder, with a few rimes removed, see Figure 2. Obviously, the ideal RatioCut for $k = 2$ just cuts the ladder by a vertical cut such that $A = \{v_1, \ldots, v_k, v_{2k+1}, \ldots, v_{3k}\}$ and $\overline{A} = \{v_{k+1}, \ldots, v_{2k}, v_{3k+1}, \ldots, v_{4k}\}$. This cut is perfectly balanced with $|A| = |\overline{A}| = 2k$ and $\mathrm{cut}(A, \overline{A}) = 2$. However, by studying the properties of the second eigenvector of the unnormalized graph Laplacian of cockroach graphs the authors prove that unnormalized spectral clustering always cuts horizontally through the ladder, constructing the sets $B = \{v_1, \ldots, v_{2k}\}$ and $\overline{B} = \{v_{2k+1}, \ldots, v_{4k}\}$. This also results in a balanced cut, but now we cut k edges instead of just 2. So $\mathrm{RatioCut}(A, \overline{A}) = 2/k$, while $\mathrm{RatioCut}(B, \overline{B}) = 1$. This means that compared to the optimal cut, the RatioCut value obtained by spectral clustering is $k/2$ times worse, that is a factor in the order of n. Several other papers investigate the quality of the clustering constructed by spectral clustering, for example Spielman and Teng (1996) (for unnormalized spectral clustering) and Kannan, Vempala, and Vetta (2004) (for normalized spectral clustering). In general it is known that efficient algorithms to approximate balanced graph cuts up to a constant factor do not exist. To the contrary, this approximation problem can be NP hard itself (Bui and Jones, 1992).

Of course, the relaxation we discussed above is not unique. For example, a completely different relaxation which leads to a semi-definite program is derived in Bie and Cristianini (2006), and there might be many other useful relaxations. The reason why the spectral relaxation is so appealing is not that it leads to particularly good solutions. Its popularity is mainly due to the fact that it results in a standard linear algebra problem which is simple to solve.

6 Random walks point of view

Another line of argument to explain spectral clustering is based on random walks on the similarity graph. A random walk on a graph is a stochastic process which randomly jumps from vertex to vertex. We will see below that spectral clustering can be interpreted as trying to find a partition of the graph such that the random walk stays long within the same cluster and seldom jumps between clusters. Intuitively this makes sense, in particular together with the graph cut explanation of the last section: a balanced partition with a low cut will also have the property that the random walk does not have many opportunities to jump between clusters. For background reading on random walks in general we refer to Norris (1997) and Brémaud (1999), and for random walks on graphs we recommend Aldous and Fill (in preparation) and Lovász (1993). Formally, the transition probability of jumping in one step from vertex v_i to vertex v_j is proportional to the edge weight w_{ij} and is given by $p_{ij} := w_{ij}/d_i$. The transition matrix $P = (p_{ij})_{i,j=1,\dots,n}$ of the random walk is thus defined by

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

If the graph is connected and non-bipartite, then the random walk always possesses a unique stationary distribution $\pi = (\pi_1, \dots, \pi_n)'$, where $\pi_i = d_i/\operatorname{vol}(V)$. Obviously there is a tight relationship between L_{rw} and P, as $L_{\text{rw}} = I - P$. As a consequence, λ is an eigenvalue of L_{rw} with eigenvector u if and only if $1 - \lambda$ is an eigenvalue of P with eigenvector u. It is well known that many properties of a graph can be expressed in terms of the corresponding random walk transition matrix P, see Lovász (1993) for an overview. From this point of view it does not come as a surprise that the largest eigenvectors of P and the smallest eigenvectors of L_{rw} can be used to describe cluster properties of the graph.

Random walks and Ncut

A formal equivalence between Ncut and transition probabilities of the random walk has been observed in Meila and Shi (2001).

Proposition 5 (Ncut **via transition probabilities)** *Let G be connected and non bi-partite. Assume that we run the random walk $(X_t)_{t\in\mathbb{N}}$ starting with X_0 in the stationary distribution π. For disjoint subsets $A, B \subset V$, denote by $P(B|A) := P(X_1 \in B | X_0 \in A)$. Then:*

$$\operatorname{Ncut}(A, \overline{A}) = P(\overline{A}|A) + P(A|\overline{A}).$$

Proof. First of all observe that

$$P(X_0 \in A, X_1 \in B) = \sum_{i\in A, j\in B} P(X_0 = i, X_1 = j) = \sum_{i\in A, j\in B} \pi_i p_{ij}$$
$$= \sum_{i\in A, j\in B} \frac{d_i}{\operatorname{vol}(V)} \frac{w_{ij}}{d_i} = \frac{1}{\operatorname{vol}(V)} \sum_{i\in A, j\in B} w_{ij}.$$

Using this we obtain

$$P(X_1 \in B | X_0 \in A) = \frac{P(X_0 \in A, X_1 \in B)}{P(X_0 \in A)}$$
$$= \left(\frac{1}{\mathrm{vol}(V)} \sum_{i \in A, j \in B} w_{ij} \right) \left(\frac{\mathrm{vol}(A)}{\mathrm{vol}(V)} \right)^{-1} = \frac{\sum_{i \in A, j \in B} w_{ij}}{\mathrm{vol}(A)}.$$

Now the proposition follows directly with the definition of Ncut. □

This proposition leads to a nice interpretation of Ncut, and hence of normalized spectral clustering. It tells us that when minimizing Ncut, we actually look for a cut through the graph such that a random walk seldom transitions from A to $\overline{A}$ and vice versa.

The commute distance

A second connection between random walks and graph Laplacians can be made via the commute distance on the graph. The commute distance (also called resistance distance) c_{ij} between two vertices v_i and v_j is the expected time it takes the random walk to travel from vertex v_i to vertex v_j and back (Lovász, 1993; Aldous and Fill, in preparation). The commute distance has several nice properties which make it particularly appealing for machine learning. As opposed to the shortest path distance on a graph, the commute distance between two vertices decreases if there are many different short ways to get from vertex v_i to vertex v_j. So instead of just looking for the one shortest path, the commute distance looks at the set of short paths. Points which are connected by a short path in the graph and lie in the same high-density region of the graph are considered closer to each other than points which are connected by a short path but lie in different high-density regions of the graph. In this sense, the commute distance seems particularly well-suited to be used for clustering purposes.

Remarkably, the commute distance on a graph can be computed with the help of the generalized inverse (also called pseudo-inverse or Moore-Penrose inverse) $L^\dagger$ of the graph Laplacian L. In the following we denote $e_i = (0, \ldots 0, 1, 0, \ldots, 0)'$ as the i-th unit vector. To define the generalized inverse of L, recall that by Proposition 1 the matrix L can be decomposed as $L = U \Lambda U'$ where U is the matrix containing all eigenvectors as columns and Λ the diagonal matrix with the eigenvalues $\lambda_1, \ldots, \lambda_n$ on the diagonal. As at least one of the eigenvalues is 0, the matrix L is not invertible. Instead, we define its generalized inverse as $L^\dagger := U \Lambda^\dagger U'$ where the matrix $\Lambda^\dagger$ is the diagonal matrix with diagonal entries $1/\lambda_i$ if $\lambda_i \neq 0$ and 0 if $\lambda_i = 0$. The entries of $L^\dagger$ can be computed as $l^\dagger_{ij} = \sum_{k=2}^{n} \frac{1}{\lambda_k} u_{ik} u_{jk}$. The matrix $L^\dagger$ is positive semi-definite and symmetric. For further properties of $L^\dagger$ see Gutman and Xiao (2004).

Proposition 6 (Commute distance) *Let $G = (V, E)$ a connected, undirected graph. Denote by c_{ij} the commute distance between vertex v_i and vertex v_j, and by $L^\dagger = (l^\dagger_{ij})_{i,j=1,\ldots,n}$ the generalized inverse of L. Then we have:*

$$c_{ij} = \mathrm{vol}(V)(l^\dagger_{ii} - 2l^\dagger_{ij} + l^\dagger_{jj}) = \mathrm{vol}(V)(e_i - e_j)' L^\dagger (e_i - e_j).$$

This result has been published by Klein and Randic (1993), where it has been proved by methods of electrical network theory. For a proof using first step analysis for random walks see Fouss, Pirotte, Renders, and Saerens (2007). There also exist other ways to express the commute distance with the help of graph Laplacians. For example a method in terms of eigenvectors of the normalized Laplacian L_{sym} can be found as Corollary 3.2 in Lovász (1993), and a method computing the commute distance with the help of determinants of certain sub-matrices of L can be found in Bapat, Gutman, and Xiao (2003).

Proposition 6 has an important consequence. It shows that $\sqrt{c_{ij}}$ can be considered as a Euclidean distance function on the vertices of the graph. This means that we can construct an embedding which

maps the vertices v_i of the graph on points $z_i \in \mathbb{R}^n$ such that the Euclidean distances between the points z_i coincide with the commute distances on the graph. This works as follows. As the matrix $L^\dagger$ is positive semi-definite and symmetric, it induces an inner product on $\mathbb{R}^n$ (or to be more formal, it induces an inner product on the subspace of $\mathbb{R}^n$ which is perpendicular to the vector $\mathbb{1}$). Now choose z_i as the point in $\mathbb{R}^n$ corresponding to the i-th row of the matrix $U(\Lambda^\dagger)^{1/2}$. Then, by Proposition 6 and by the construction of $L^\dagger$ we have that $\langle z_i, z_j \rangle = e_i' L^\dagger e_j$ and $c_{ij} = \mathrm{vol}(V)||z_i - z_j||^2$.

The embedding used in unnormalized spectral clustering is related to the commute time embedding, but not identical. In spectral clustering, we map the vertices of the graph on the rows y_i of the matrix U, while the commute time embedding maps the vertices on the rows z_i of the matrix $(\Lambda^\dagger)^{1/2}U$. That is, compared to the entries of y_i, the entries of z_i are additionally scaled by the inverse eigenvalues of L. Moreover, in spectral clustering we only take the first k columns of the matrix, while the commute time embedding takes all columns. Several authors now try to justify why y_i and z_i are not so different after all and state a bit hand-waiving that the fact that spectral clustering constructs clusters based on the Euclidean distances between the y_i can be interpreted as building clusters of the vertices in the graph based on the commute distance. However, note that both approaches can differ considerably. For example, in the optimal case where the graph consists of k disconnected components, the first k eigenvalues of L are 0 according to Proposition 2, and the first k columns of U consist of the cluster indicator vectors. However, the first k columns of the matrix $(\Lambda^\dagger)^{1/2}U$ consist of zeros only, as the first k diagonal elements of $\Lambda^\dagger$ are 0. In this case, the information contained in the first k columns of U is completely ignored in the matrix $(\Lambda^\dagger)^{1/2}U$, and all the non-zero elements of the matrix $(\Lambda^\dagger)^{1/2}U$ which can be found in columns $k+1$ to n are not taken into account in spectral clustering, which discards all those columns. On the other hand, those problems do not occur if the underlying graph is connected. In this case, the only eigenvector with eigenvalue 0 is the constant one vector, which can be ignored in both cases. The eigenvectors corresponding to small eigenvalues λ_i of L are then stressed in the matrix $(\Lambda^\dagger)^{1/2}U$ as they are multiplied by $\lambda_i^\dagger = 1/\lambda_i$. In such a situation, it might be true that the commute time embedding and the spectral embedding do similar things.

All in all, it seems that the commute time distance can be a helpful intuition, but without making further assumptions there is only a rather loose relation between spectral clustering and the commute distance. It might be possible that those relations can be tightened, for example if the similarity function is strictly positive definite. However, we have not yet seen a precise mathematical statement about this.

7 Perturbation theory point of view

Perturbation theory studies the question of how eigenvalues and eigenvectors of a matrix A change if we add a small perturbation H, that is we consider the perturbed matrix $\tilde{A} := A + H$. Most perturbation theorems state that a certain distance between eigenvalues or eigenvectors of A and $\tilde{A}$ is bounded by a constant times a norm of H. The constant usually depends on which eigenvalue we are looking at, and how far this eigenvalue is separated from the rest of the spectrum (for a formal statement see below). The justification of spectral clustering is then the following: Let us first consider the "ideal case" where the between-cluster similarity is exactly 0. We have seen in Section 3 that then the first k eigenvectors of L or L_{rw} are the indicator vectors of the clusters. In this case, the points $y_i \in \mathbb{R}^k$ constructed in the spectral clustering algorithms have the form $(0, \ldots, 0, 1, 0, \ldots 0)'$ where the position of the 1 indicates the connected component this point belongs to. In particular, all y_i belonging to the same connected component coincide. The k-means algorithm will trivially find the correct partition by placing a center point on each of the points $(0, \ldots, 0, 1, 0, \ldots 0)' \in \mathbb{R}^k$. In a "nearly ideal case" where we still have distinct clusters, but the between-cluster similarity is not exactly 0, we consider the Laplacian matrices to be perturbed versions of the ones of the ideal case. Perturbation theory then tells us that the eigenvectors will be very close to the ideal indicator vectors. The points y_i might not

completely coincide with $(0, \ldots, 0, 1, 0, \ldots 0)'$, but do so up to some small error term. Hence, if the perturbations are not too large, then k-means algorithm will still separate the groups from each other.

7.1 The formal perturbation argument

The formal basis for the perturbation approach to spectral clustering is the Davis-Kahan theorem from matrix perturbation theory. This theorem bounds the difference between eigenspaces of symmetric matrices under perturbations. We state those results for completeness, but for background reading we refer to Section V of Stewart and Sun (1990) and Section VII.3 of Bhatia (1997). In perturbation theory, distances between subspaces are usually measured using "canonical angles" (also called "principal angles"). To define principal angles, let $\mathcal{V}_1$ and $\mathcal{V}_2$ be two p-dimensional subspaces of $\mathbb{R}^d$, and V_1 and V_2 two matrices such that their columns form orthonormal systems for $\mathcal{V}_1$ and $\mathcal{V}_2$, respectively. Then the cosines $\cos \Theta_i$ of the principal angles Θ_i are the singular values of $V_1' V_2$. For $p = 1$, the so defined canonical angles coincide with the normal definition of an angle. Canonical angles can also be defined if $\mathcal{V}_1$ and $\mathcal{V}_2$ do not have the same dimension, see Section V of Stewart and Sun (1990), Section VII.3 of Bhatia (1997), or Section 12.4.3 of Golub and Van Loan (1996). The matrix $\sin \mathbf{\Theta}(\mathcal{V}_1, \mathcal{V}_2)$ will denote the diagonal matrix with the sine of the canonical angles on the diagonal.

Theorem 7 (Davis-Kahan) *Let $A, H \in \mathbb{R}^{n \times n}$ be symmetric matrices, and let $\| \cdot \|$ be the Frobenius norm or the two-norm for matrices, respectively. Consider $\tilde{A} := A + H$ as a perturbed version of A. Let $S_1 \subset \mathbb{R}$ be an interval. Denote by $\sigma_{S_1}(A)$ the set of eigenvalues of A which are contained in S_1, and by V_1 the eigenspace corresponding to all those eigenvalues (more formally, V_1 is the image of the spectral projection induced by $\sigma_{S_1}(A)$). Denote by $\sigma_{S_1}(\tilde{A})$ and $\tilde{V}_1$ the analogous quantities for $\tilde{A}$. Define the distance between S_1 and the spectrum of A outside of S_1 as*

$$\delta = \min\{|\lambda - s|;\ \lambda \text{ eigenvalue of } A,\ \lambda \notin S_1,\ s \in S_1\}.$$

Then the distance $d(V_1, \tilde{V}_1) := \| \sin \mathbf{\Theta}(V_1, \tilde{V}_1) \|$ between the two subspaces V_1 and $\tilde{V}_1$ is bounded by

$$d(V_1, \tilde{V}_1) \leq \frac{\|H\|}{\delta}.$$

For a discussion and proofs of this theorem see for example Section V.3 of Stewart and Sun (1990). Let us try to decrypt this theorem, for simplicity in the case of the unnormalized Laplacian (for the normalized Laplacian it works analogously). The matrix A will correspond to the graph Laplacian L in the ideal case where the graph has k connected components. The matrix $\tilde{A}$ corresponds to a perturbed case, where due to noise the k components in the graph are no longer completely disconnected, but they are only connected by few edges with low weight. We denote the corresponding graph Laplacian of this case by $\tilde{L}$. For spectral clustering we need to consider the first k eigenvalues and eigenvectors of $\tilde{L}$. Denote the eigenvalues of L by $\lambda_1, \ldots \lambda_n$ and the ones of the perturbed Laplacian $\tilde{L}$ by $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$. Choosing the interval S_1 is now the crucial point. We want to choose it such that both the first k eigenvalues of $\tilde{L}$ and the first k eigenvalues of L are contained in S_1. This is easier the smaller the perturbation $H = L - \tilde{L}$ and the larger the eigengap $|\lambda_k - \lambda_{k+1}|$ is. If we manage to find such a set, then the Davis-Kahan theorem tells us that the eigenspaces corresponding to the first k eigenvalues of the ideal matrix L and the first k eigenvalues of the perturbed matrix $\tilde{L}$ are very close to each other, that is their distance is bounded by $\|H\|/\delta$. Then, as the eigenvectors in the ideal case are piecewise constant on the connected components, the same will approximately be true in the perturbed case. How good "approximately" is depends on the norm of the perturbation $\|H\|$ and the distance δ between S_1 and the $(k+1)$st eigenvector of L. If the set S_1 has been chosen as the interval $[0, \lambda_k]$, then δ coincides with the spectral gap $|\lambda_{k+1} - \lambda_k|$. We can see from the theorem that the larger this eigengap is, the closer the eigenvectors of the ideal case and the perturbed case are, and hence the better spectral clustering works. Below we will see that the size of the eigengap can also be used in a

different context as a quality criterion for spectral clustering, namely when choosing the number k of clusters to construct.

If the perturbation H is too large or the eigengap is too small, we might not find a set S_1 such that both the first k eigenvalues of L and $\tilde{L}$ are contained in S_1. In this case, we need to make a compromise by choosing the set S_1 to contain the first k eigenvalues of L, but maybe a few more or less eigenvalues of $\tilde{L}$. The statement of the theorem then becomes weaker in the sense that either we do not compare the eigenspaces corresponding to the first k eigenvectors of L and $\tilde{L}$, but the eigenspaces corresponding to the first k eigenvectors of L and the first $\tilde{k}$ eigenvectors of $\tilde{L}$ (where $\tilde{k}$ is the number of eigenvalues of $\tilde{L}$ contained in S_1). Or, it can happen that δ becomes so small that the bound on the distance between $d(V_1, \tilde{V}_1)$ blows up so much that it becomes useless.

7.2 Comments about the perturbation approach

A bit of caution is needed when using perturbation theory arguments to justify clustering algorithms based on eigenvectors of matrices. In general, *any* block diagonal symmetric matrix has the property that there exists a basis of eigenvectors which are zero outside the individual blocks and real-valued within the blocks. For example, based on this argument several authors use the eigenvectors of the similarity matrix S or adjacency matrix W to discover clusters. However, being block diagonal in the ideal case of completely separated clusters can be considered as a necessary condition for a successful use of eigenvectors, but not a sufficient one. At least two more properties should be satisfied:

First, we need to make sure that the *order* of the eigenvalues and eigenvectors is meaningful. In case of the Laplacians this is always true, as we know that any connected component possesses exactly one eigenvector which has eigenvalue 0. Hence, if the graph has k connected components and we take the first k eigenvectors of the Laplacian, then we know that we have exactly one eigenvector per component. However, this might not be the case for other matrices such as S or W. For example, it could be the case that the two largest eigenvalues of a block diagonal similarity matrix S come from the same block. In such a situation, if we take the first k eigenvectors of S, some blocks will be represented several times, while there are other blocks which we will miss completely (unless we take certain precautions). This is the reason why using the eigenvectors of S or W for clustering should be discouraged.

The second property is that in the ideal case, the entries of the eigenvectors on the components should be "safely bounded away" from 0. Assume that an eigenvector on the first connected component has an entry $u_{1,i} > 0$ at position i. In the ideal case, the fact that this entry is non-zero indicates that the corresponding point i belongs to the first cluster. The other way round, if a point j does not belong to cluster 1, then in the ideal case it should be the case that $u_{1,j} = 0$. Now consider the same situation, but with perturbed data. The perturbed eigenvector $\tilde{u}$ will usually not have any non-zero component any more; but if the noise is not too large, then perturbation theory tells us that the entries $\tilde{u}_{1,i}$ and $\tilde{u}_{1,j}$ are still "close" to their original values $u_{1,i}$ and $u_{1,j}$. So both entries $\tilde{u}_{1,i}$ and $\tilde{u}_{1,j}$ will take some small values, say ε_1 and ε_2. In practice, if those values are very small it is unclear how we should interpret this situation. Either we believe that small entries in $\tilde{u}$ indicate that the points do not belong to the first cluster (which then misclassifies the first data point i), or we think that the entries already indicate class membership and classify both points to the first cluster (which misclassifies point j).

For both matrices L and L_{rw}, the eigenvectors in the ideal situation are indicator vectors, so the second problem described above cannot occur. However, this is not true for the matrix L_{sym}, which is used in the normalized spectral clustering algorithm of Ng et al. (2002). Even in the ideal case, the eigenvectors of this matrix are given as $D^{1/2}\mathbb{1}_{A_i}$. If the degrees of the vertices differ a lot, and in particular if there are vertices which have a very low degree, the corresponding entries in the eigenvectors are very small. To counteract the problem described above, the row-normalization step in the algorithm of Ng et al. (2002) comes into play. In the ideal case, the matrix U in the algorithm has exactly one

non-zero entry per row. After row-normalization, the matrix T in the algorithm of Ng et al. (2002) then consists of the cluster indicator vectors. Note however, that this might not always work out correctly in practice. Assume that we have $\tilde{u}_{i,1} = \varepsilon_1$ and $\tilde{u}_{i,2} = \varepsilon_2$. If we now normalize the i-th row of U, both ε_1 and ε_2 will be multiplied by the factor of $1/\sqrt{\varepsilon_1^2 + \varepsilon_2^2}$ and become rather large. We now run into a similar problem as described above: both points are likely to be classified into the same cluster, even though they belong to different clusters. This argument shows that spectral clustering using the matrix L_{sym} can be problematic if the eigenvectors contain particularly small entries. On the other hand, note that such small entries in the eigenvectors only occur if some of the vertices have a particularly low degrees (as the eigenvectors of L_{sym} are given by $D^{1/2}\mathbb{1}_{A_i}$). One could argue that in such a case, the data point should be considered an outlier anyway, and then it does not really matter in which cluster the point will end up.

To summarize, the conclusion is that both unnormalized spectral clustering and normalized spectral clustering with L_{rw} are well justified by the perturbation theory approach. Normalized spectral clustering with L_{sym} can also be justified by perturbation theory, but it should be treated with more care if the graph contains vertices with very low degrees.

8 Practical details

In this section we will briefly discuss some of the issues which come up when actually implementing spectral clustering. There are several choices to be made and parameters to be set. However, the discussion in this section is mainly meant to raise awareness about the general problems which an occur. For thorough studies on the behavior of spectral clustering for various real world tasks we refer to the literature.

8.1 Constructing the similarity graph

Constructing the similarity graph for spectral clustering is not a trivial task, and little is known on theoretical implications of the various constructions.

The similarity function itself

Before we can even think about constructing a similarity graph, we need to define a similarity function on the data. As we are going to construct a neighborhood graph later on, we need to make sure that the local neighborhoods induced by this similarity function are "meaningful". This means that we need to be sure that points which are considered to be "very similar" by the similarity function are also closely related in the application the data comes from. For example, when constructing a similarity function between text documents it makes sense to check whether documents with a high similarity score indeed belong to the same text category. The global "long-range" behavior of the similarity function is not so important for spectral clustering — it does not really matter whether two data points have similarity score 0.01 or 0.001, say, as we will not connect those two points in the similarity graph anyway. In the common case where the data points live in the Euclidean space $\mathbb{R}^d$, a reasonable default candidate is the Gaussian similarity function $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$ (but of course we need to choose the parameter σ here, see below). Ultimately, the choice of the similarity function depends on the domain the data comes from, and no general advice can be given.

Which type of similarity graph

The next choice one has to make concerns the type of the graph one wants to use, such as the k-nearest neighbor or the ε-neighborhood graph. Let us illustrate the behavior of the different graphs using the toy example presented in Figure 3. As underlying distribution we choose a distribution on $\mathbb{R}^2$ with

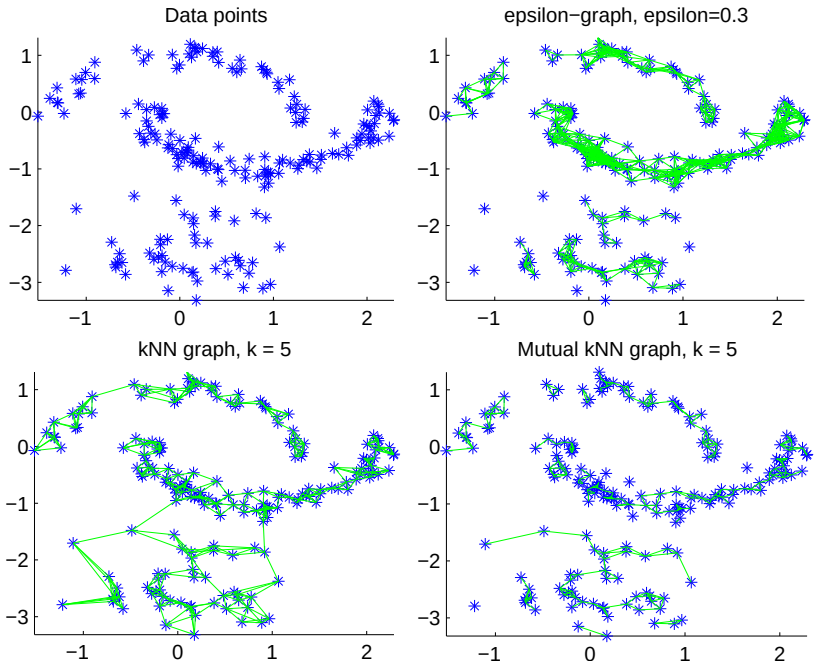

Figure 3: Different similarity graphs, see text for details.

three clusters: two "moons" and a Gaussian. The density of the bottom moon is chosen to be larger than the one of the top moon. The upper left panel in Figure 3 shows a sample drawn from this distribution. The next three panels show the different similarity graphs on this sample.

In the ε-neighborhood graph, we can see that it is difficult to choose a useful parameter ε. With $\varepsilon = 0.3$ as in the figure, the points on the middle moon are already very tightly connected, while the points in the Gaussian are barely connected. This problem always occurs if we have data "on different scales", that is the distances between data points are different in different regions of the space.

The k-nearest neighbor graph, on the other hand, can connect points "on different scales". We can see that points in the low-density Gaussian are connected with points in the high-density moon. This is a general property of k-nearest neighbor graphs which can be very useful. We can also see that the k-nearest neighbor graph can break into several disconnected components if there are high density regions which are reasonably far away from each other. This is the case for the two moons in this example.

The mutual k-nearest neighbor graph has the property that it tends to connect points within regions of constant density, but does not connect regions of different densities with each other. So the mutual k-nearest neighbor graph can be considered as being "in between" the ε-neighborhood graph and the k-nearest neighbor graph. It is able to act on different scales, but does not mix those scales with each other. Hence, the mutual k-nearest neighbor graph seems particularly well-suited if we want to detect clusters of different densities.

The fully connected graph is very often used in connection with the Gaussian similarity function $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$. Here the parameter σ plays a similar role as the parameter ε in the ε-neighborhood graph. Points in local neighborhoods are connected with relatively high weights, while edges between far away points have positive, but negligible weights. However, the resulting

similarity matrix is not a sparse matrix.

As a general recommendation we suggest to work with the k-nearest neighbor graph as the first choice. It is simple to work with, results in a sparse adjacency matrix W, and in our experience is less vulnerable to unsuitable choices of parameters than the other graphs.

The parameters of the similarity graph

Once one has decided for the type of the similarity graph, one has to choose its connectivity parameter k or ε, respectively. Unfortunately, barely any theoretical results are known to guide us in this task. In general, if the similarity graph contains more connected components than the number of clusters we ask the algorithm to detect, then spectral clustering will trivially return connected components as clusters. Unless one is perfectly sure that those connected components are the correct clusters, one should make sure that the similarity graph is connected, or only consists of "few" connected components and very few or no isolated vertices. There are many theoretical results on how connectivity of random graphs can be achieved, but all those results only hold in the limit for the sample size $n \to \infty$. For example, it is known that for n data points drawn i.i.d. from some underlying density with a connected support in $\mathbb{R}^d$, the k-nearest neighbor graph and the mutual k-nearest neighbor graph will be connected if we choose k on the order of $\log(n)$ (e.g., Brito, Chavez, Quiroz, and Yukich, 1997). Similar arguments show that the parameter ε in the ε-neighborhood graph has to be chosen as $(\log(n)/n)^d$ to guarantee connectivity in the limit (Penrose, 1999). While being of theoretical interest, all those results do not really help us for choosing k on a finite sample.

Now let us give some rules of thumb. When working with the k-nearest neighbor graph, then the connectivity parameter should be chosen such that the resulting graph is connected, or at least has significantly fewer connected components than clusters we want to detect. For small or medium-sized graphs this can be tried out "by foot". For very large graphs, a first approximation could be to choose k in the order of $\log(n)$, as suggested by the asymptotic connectivity results.

For the mutual k-nearest neighbor graph, we have to admit that we are a bit lost for rules of thumb. The advantage of the mutual k-nearest neighbor graph compared to the standard k-nearest neighbor graph is that it tends not to connect areas of different density. While this can be good if there are clear clusters induced by separate high-density areas, this can hurt in less obvious situations as disconnected parts in the graph will always be chosen to be clusters by spectral clustering. Very generally, one can observe that the mutual k-nearest neighbor graph has much fewer edges than the standard k-nearest neighbor graph for the same parameter k. This suggests to choose k significantly larger for the mutual k-nearest neighbor graph than one would do for the standard k-nearest neighbor graph. However, to take advantage of the property that the mutual k-nearest neighbor graph does not connect regions of different density, it would be necessary to allow for several "meaningful" disconnected parts of the graph. Unfortunately, we do not know of any general heuristic to choose the parameter k such that this can be achieved.

For the ε-neighborhood graph, we suggest to choose ε such that the resulting graph is safely connected. To determine the smallest value of ε where the graph is connected is very simple: one has to choose ε as the length of the longest edge in a minimal spanning tree of the fully connected graph on the data points. The latter can be determined easily by any minimal spanning tree algorithm. However, note that when the data contains outliers this heuristic will choose ε so large that even the outliers are connected to the rest of the data. A similar effect happens when the data contains several tight clusters which are very far apart from each other. In both cases, ε will be chosen too large to reflect the scale of the most important part of the data.

Finally, if one uses a fully connected graph together with a similarity function which can be scaled

itself, for example the Gaussian similarity function, then the scale of the similarity function should be chosen such that the resulting graph has similar properties as a corresponding k-nearest neighbor or ε-neighborhood graph would have. One needs to make sure that for most data points the set of neighbors with a similarity significantly larger than 0 is "not too small and not too large". In particular, for the Gaussian similarity function several rules of thumb are frequently used. For example, one can choose σ in the order of the mean distance of a point to its k-th nearest neighbor, where k is chosen similarly as above (e.g., $k \sim \log(n) + 1$). Another way is to determine ε by the minimal spanning tree heuristic described above, and then choose $\sigma = \varepsilon$. But note that all those rules of thumb are very ad-hoc, and depending on the given data at hand and its distribution of inter-point distances they might not work at all.

In general, experience shows that spectral clustering can be quite sensitive to changes in the similarity graph and to the choice of its parameters. Unfortunately, to our knowledge there has been no systematic study which investigates the effects of the similarity graph and its parameters on clustering and comes up with well-justified rules of thumb. None of the recommendations above is based on a firm theoretic ground. Finding rules which have a theoretical justification should be considered an interesting and important topic for future research.

8.2 Computing the eigenvectors

To implement spectral clustering in practice one has to compute the first k eigenvectors of a potentially large graph Laplace matrix. Luckily, if we use the k-nearest neighbor graph or the ε-neighborhood graph, then all those matrices are sparse. Efficient methods exist to compute the first eigenvectors of sparse matrices, the most popular ones being the power method or Krylov subspace methods such as the Lanczos method (Golub and Van Loan, 1996). The speed of convergence of those algorithms depends on the size of the eigengap (also called spectral gap) $\gamma_k = |\lambda_k - \lambda_{k+1}|$. The larger this eigengap is, the faster the algorithms computing the first k eigenvectors converge.

Note that a general problem occurs if one of the eigenvalues under consideration has multiplicity larger than one. For example, in the ideal situation of k disconnected clusters, the eigenvalue 0 has multiplicity k. As we have seen, in this case the eigenspace is spanned by the k cluster indicator vectors. But unfortunately, the vectors computed by the numerical eigensolvers do not necessarily converge to those particular vectors. Instead they just converge to some orthonormal basis of the eigenspace, and it usually depends on implementation details to which basis exactly the algorithm converges. But this is not so bad after all. Note that all vectors in the space spanned by the cluster indicator vectors $\mathbb{1}_{A_i}$ have the form $u = \sum_{i=1}^{k} a_i \mathbb{1}_{A_i}$ for some coefficients a_i, that is, they are piecewise constant on the clusters. So the vectors returned by the eigensolvers still encode the information about the clusters, which can then be used by the k-means algorithm to reconstruct the clusters.

8.3 The number of clusters

Choosing the number k of clusters is a general problem for all clustering algorithms, and a variety of more or less successful methods have been devised for this problem. In model-based clustering settings there exist well-justified criteria to choose the number of clusters from the data. Those criteria are usually based on the log-likelihood of the data, which can then be treated in a frequentist or Bayesian way, for examples see Fraley and Raftery (2002). In settings where no or few assumptions on the underlying model are made, a large variety of different indices can be used to pick the number of clusters. Examples range from ad-hoc measures such as the ratio of within-cluster and between-cluster similarities, over information-theoretic criteria (Still and Bialek, 2004), the gap statistic (Tibshirani, Walther, and Hastie, 2001), to stability approaches (Ben-Hur, Elisseeff, and Guyon, 2002; Lange, Roth,

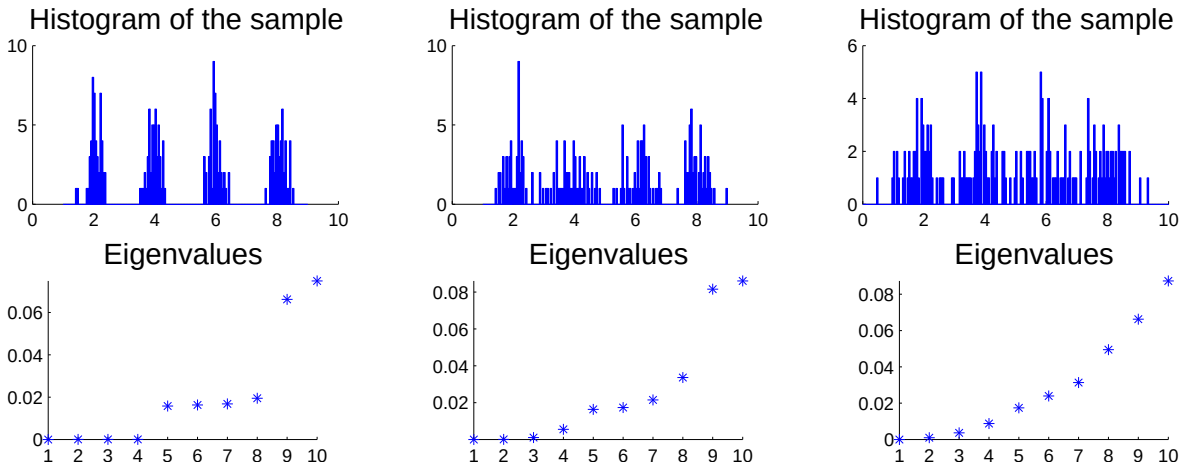

Figure 4: Three data sets, and the smallest 10 eigenvalues of L_{rw}. See text for more details.

Braun, and Buhmann, 2004; Ben-David, von Luxburg, and Pál, 2006). Of course all those methods can also be used for spectral clustering. Additionally, one tool which is particularly designed for spectral clustering is the eigengap heuristic, which can be used for all three graph Laplacians. Here the goal is to choose the number k such that all eigenvalues $\lambda_1, \ldots, \lambda_k$ are very small, but λ_{k+1} is relatively large. There are several justifications for this procedure. The first one is based on perturbation theory, where we observe that in the ideal case of k completely disconnected clusters, the eigenvalue 0 has multiplicity k, and then there is a gap to the $(k+1)$th eigenvalue $\lambda_{k+1} > 0$. Other explanations can be given by spectral graph theory. Here, many geometric invariants of the graph can be expressed or bounded with the help of the first eigenvalues of the graph Laplacian. In particular, the sizes of cuts are closely related to the size of the first eigenvalues. For more details on this topic we refer to Bolla (1991), Mohar (1997) and Chung (1997).

We would like to illustrate the eigengap heuristic on our toy example introduced in Section 4. For this purpose we consider similar data sets as in Section 4, but to vary the difficulty of clustering we consider the Gaussians with increasing variance. The first row of Figure 4 shows the histograms of the three samples. We construct the 10-nearest neighbor graph as described in Section 4, and plot the eigenvalues of the normalized Laplacian L_{rw} on the different samples (the results for the unnormalized Laplacian are similar). The first data set consists of four well separated clusters, and we can see that the first 4 eigenvalues are approximately 0. Then there is a gap between the 4th and 5th eigenvalue, that is $|\lambda_5 - \lambda_4|$ is relatively large. According to the eigengap heuristic, this gap indicates that the data set contains 4 clusters. The same behavior can also be observed for the results of the fully connected graph (already plotted in Figure 1). So we can see that the heuristic works well if the clusters in the data are very well pronounced. However, the more noisy or overlapping the clusters are, the less effective is this heuristic. We can see that for the second data set where the clusters are more "blurry", there is still a gap between the 4th and 5th eigenvalue, but it is not as clear to detect as in the case before. Finally, in the last data set, there is no well-defined gap, the differences between all eigenvalues are approximately the same. But on the other hand, the clusters in this data set overlap so much that many non-parametric algorithms will have difficulties to detect the clusters, unless they make strong assumptions on the underlying model. In this particular example, even for a human looking at the histogram it is not obvious what the correct number of clusters should be. This illustrates that, as most methods for choosing the number of clusters, the eigengap heuristic usually works well if the data contains very well pronounced clusters, but in ambiguous cases it also returns ambiguous results.

Finally, note that the choice of the number of clusters and the choice of the connectivity parameters of the neighborhood graph affect each other. For example, if the connectivity parameter of the neighborhood graph is so small that the graph breaks into, say, k_0 connected components, then choosing k_0 as the number of clusters is a valid choice. However, as soon as the neighborhood graph is connected, it is not clear how the number of clusters and the connectivity parameters of the neighborhood graph interact. Both the choice of the number of clusters and the choice of the connectivity parameters of the graph are difficult problems on their own, and to our knowledge nothing non-trivial is known on their interactions.

8.4 The k-means step

The three spectral clustering algorithms we presented in Section 4 use k-means as last step to extract the final partition from the real valued matrix of eigenvectors. First of all, note that there is nothing principled about using the k-means algorithm in this step. In fact, as we have seen from the various explanations of spectral clustering, this step should be very simple if the data contains well-expressed clusters. For example, in the ideal case if completely separated clusters we know that the eigenvectors of L and L_{rw} are piecewise constant. In this case, all points x_i which belong to the same cluster C_s are mapped to exactly the sample point y_i, namely to the unit vector $e_s \in \mathbb{R}^k$. In such a trivial case, any clustering algorithm applied to the points $y_i \in \mathbb{R}^k$ will be able to extract the correct clusters.

While it is somewhat arbitrary what clustering algorithm exactly one chooses in the final step of spectral clustering, one can argue that at least the Euclidean distance between the points y_i is a meaningful quantity to look at. We have seen that the Euclidean distance between the points y_i is related to the "commute distance" on the graph, and in Nadler, Lafon, Coifman, and Kevrekidis (2006) the authors show that the Euclidean distances between the y_i are also related to a more general "diffusion distance". Also, other uses of the spectral embeddings (e.g., Bolla (1991) or Belkin and Niyogi (2003)) show that the Euclidean distance in $\mathbb{R}^d$ is meaningful.

Instead of k-means, people also use other techniques to construct he final solution from the real-valued representation. For example, in Lang (2006) the authors use hyperplanes for this purpose. A more advanced post-processing of the eigenvectors is proposed in Bach and Jordan (2004). Here the authors study the subspace spanned by the first k eigenvectors, and try to approximate this subspace as good as possible using piecewise constant vectors. This also leads to minimizing certain Euclidean distances in the space $\mathbb{R}^k$, which can be done by some weighted k-means algorithm.

8.5 Which graph Laplacian should be used?

A fundamental question related to spectral clustering is the question which of the three graph Laplacians should be used to compute the eigenvectors. Before deciding this question, one should always look at the degree distribution of the similarity graph. If the graph is very regular and most vertices have approximately the same degree, then all the Laplacians are very similar to each other, and will work equally well for clustering. However, if the degrees in the graph are very broadly distributed, then the Laplacians differ considerably. In our opinion, there are several arguments which advocate for using normalized rather than unnormalized spectral clustering, and in the normalized case to use the eigenvectors of L_{rw} rather than those of L_{sym}.

Clustering objectives satisfied by the different algorithms

The first argument in favor of normalized spectral clustering comes from the graph partitioning point of view. For simplicity let us discuss the case $k = 2$. In general, clustering has two different objectives:

1. We want to find a partition such that points in different clusters are dissimilar to each other, that is we want to minimize the between-cluster similarity. In the graph setting, this means to minimize $\mathrm{cut}(A, \overline{A})$.

2. We want to find a partition such that points in the same cluster are similar to each other, that is we want to maximize the within-cluster similarities $W(A, A)$ and $W(\overline{A}, \overline{A})$.

Both RatioCut and Ncut directly implement the first objective by explicitly incorporating $\mathrm{cut}(A, \overline{A})$ in the objective function. However, concerning the second point, both algorithms behave differently. Note that

$$W(A, A) = W(A, V) - W(A, \overline{A}) = \mathrm{vol}(A) - \mathrm{cut}(A, \overline{A}).$$

Hence, the within-cluster similarity is maximized if $\mathrm{cut}(A, \overline{A})$ is small *and* if $\mathrm{vol}(A)$ is large. As this is exactly what we achieve by minimizing Ncut, the Ncut criterion implements the second objective. This can be seen even more explicitly by considering yet another graph cut objective function, namely the MinMaxCut criterion introduced by Ding, He, Zha, Gu, and Simon (2001):

$$\mathrm{MinMaxCut}(A_1, \ldots, A_k) := \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, \overline{A}_i)}{W(A_i, A_i)}.$$

Compared to Ncut, which has the terms $\mathrm{vol}(A) = \mathrm{cut}(A, \overline{A}) + W(A, A)$ in the denominator, the MinMaxCut criterion only has $W(A, A)$ in the denominator. In practice, Ncut and MinMaxCut are often minimized by similar cuts, as a good Ncut solution will have a small value of $\mathrm{cut}(A, \overline{A})$ anyway and hence the denominators are not so different after all. Moreover, relaxing MinMaxCut leads to exactly the same optimization problem as relaxing Ncut, namely to normalized spectral clustering with the eigenvectors of L_{rw}. So one can see by several ways that normalized spectral clustering incorporates both clustering objectives mentioned above.

Now consider the case of RatioCut. Here the objective is to maximize $|A|$ and$|\overline{A}|$ instead of $\mathrm{vol}(A)$ and $\mathrm{vol}(\overline{A})$. But $|A|$ and $|\overline{A}|$ are not necessarily related to the within-cluster similarity, as the within-cluster similarity depends on the edges and not on the number of vertices in A. For instance, just think of a set A which has very many vertices, all of which only have very low weighted edges to each other. Minimizing RatioCut does not attempt to maximize the within-cluster similarity, and the same is then true for its relaxation by unnormalized spectral clustering.

So this is our first important point to keep in mind: Normalized spectral clustering implements both clustering objectives mentioned above, while unnormalized spectral clustering only implements the first objective.

Consistency issues

A completely different argument for the superiority of normalized spectral clustering comes from a statistical analysis of both algorithms. In a statistical setting one assumes that the data points $x_1, \ldots, x_n$ have been sampled i.i.d. according to some probability distribution P on some underlying data space $\mathcal{X}$. The most fundamental question is then the question of consistency: if we draw more and more data points, do the clustering results of spectral clustering converge to a useful partition of the underlying space $\mathcal{X}$?

For both normalized spectral clustering algorithms, it can be proved that this is indeed the case (von Luxburg, Bousquet, and Belkin, 2004, 2005; von Luxburg, Belkin, and Bousquet, to appear). Mathematically, one proves that as we take the limit $n \to \infty$, the matrix L_{sym} converges in a strong sense

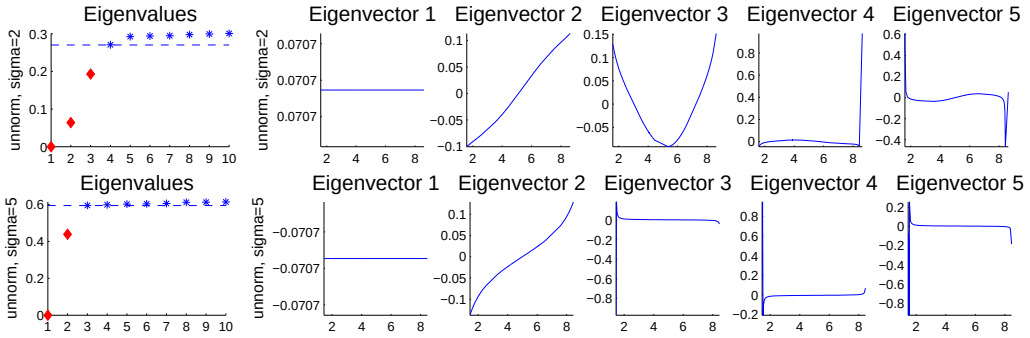

Figure 5: Consistency of unnormalized spectral clustering. Plotted are eigenvalues and eigenvectors of L, for parameter $\sigma = 2$ (first row) and $\sigma = 5$ (second row). The dashed line indicates $\min d_j$, the eigenvalues below $\min d_j$ are plotted as red diamonds, the eigenvalues above $\min d_j$ are plotted as blue stars. See text for more details.

to an operator U on the space $C(\mathcal{X})$ of continuous functions on $\mathcal{X}$. This convergence implies that the eigenvalues and eigenvectors of L_{sym} converge to those of U, which in turn can be transformed to a statement about the convergence of normalized spectral clustering. One can show that the partition which is induced on $\mathcal{X}$ by the eigenvectors of U can be interpreted similar to the random walks interpretation of spectral clustering. That is, if we consider a diffusion process on the data space $\mathcal{X}$, then the partition induced by the eigenvectors of U is such that the diffusion does not transition between the different clusters very often (von Luxburg et al., 2004). All consistency statements about normalized spectral clustering hold, for both L_{sym} and L_{rw}, under very mild conditions which are usually satisfied in real world applications. Unfortunately, explaining more details about those results goes beyond the scope of this tutorial, so we refer the interested reader to von Luxburg et al. (to appear).

In contrast to the clear convergence statements for normalized spectral clustering, the situation for unnormalized spectral clustering is much more unpleasant. It can be proved that unnormalized spectral clustering can fail to converge, or that it can converge to trivial solutions which construct clusters consisting of one single point of the data space (von Luxburg et al., 2005, to appear). Mathematically, even though one can prove that the matrix $(1/n)L$ itself converges to some limit operator T on $C(\mathcal{X})$ as $n \to \infty$, the spectral properties of this limit operator T can be so nasty that they prevent the convergence of spectral clustering. It is possible to construct examples which show that this is not only a problem for very large sample size, but that it can lead to completely unreliable results even for small sample size. At least it is possible to characterize the conditions when those problem do not occur: We have to make sure that the eigenvalues of L corresponding to the eigenvectors used in unnormalized spectral clustering are significantly smaller than the minimal degree in the graph. This means that if we use the first k eigenvectors for clustering, then $\lambda_i \ll \min_{j=1,\ldots,n} d_j$ should hold for all $i = 1, \ldots, k$. The mathematical reason for this condition is that eigenvectors corresponding to eigenvalues larger than $\min d_j$ approximate Dirac functions, that is they are approximately 0 in all but one coordinate. If those eigenvectors are used for clustering, then they separate the one vertex where the eigenvector is non-zero from all other vertices, and we clearly do not want to construct such a partition. Again we refer to the literature for precise statements and proofs.

For an illustration of this phenomenon, consider again our toy data set from Section 4. We consider the first eigenvalues and eigenvectors of the unnormalized graph Laplacian based on the fully connected graph, for different choices of the parameter σ of the Gaussian similarity function (see last row of Figure 1 and all rows of Figure 5). The eigenvalues above $\min d_j$ are plotted as blue stars, the eigenvalues below $\min d_j$ are plotted as red diamonds. The dashed line indicates $\min d_j$. In general, we can see

that the eigenvectors corresponding to eigenvalues which are much below the dashed lines are "useful" eigenvectors. In case $\sigma = 1$ (plotted already in the last row of Figure 1), Eigenvalues 2, 3 and 4 are significantly below $\min d_j$, and the corresponding Eigenvectors 2, 3, and 4 are meaningful (as already discussed in Section 4). If we increase the parameter σ, we can observe that the eigenvalues tend to move towards $\min d_j$. In case $\sigma = 2$, only the first three eigenvalues are below $\min d_j$ (first row in Figure 5), and in case $\sigma = 5$ only the first two eigenvalues are below $\min d_j$ (second row in Figure 5). We can see that as soon as an eigenvalue gets close to or above $\min d_j$, its corresponding eigenvector approximates a Dirac function. Of course, those eigenvectors are unsuitable for constructing a clustering. In the limit for $n \to \infty$, those eigenvectors would converge to perfect Dirac functions. Our illustration of the finite sample case shows that this behavior not only occurs for large sample size, but can be generated even on the small example in our toy data set.

It is very important to stress that those problems only concern the eigenvectors of the matrix L, and they do not occur for L_{rw} or L_{sym}. Thus, from a statistical point of view, it is preferable to avoid unnormalized spectral clustering and to use the normalized algorithms instead.

Which normalized Laplacian?

Looking at the differences between the two normalized spectral clustering algorithms using L_{rw} and L_{sym}, all three explanations of spectral clustering are in favor of L_{rw}. The reason is that the eigenvectors of L_{rw} are cluster indicator vectors $\mathbb{1}_{A_i}$, while the eigenvectors of L_{sym} are additionally multiplied with $D^{1/2}$, which might lead to undesired artifacts. As using L_{sym} also does not have any computational advantages, we thus advocate for using L_{rw}.

9 Outlook and further reading

Spectral clustering goes back to Donath and Hoffman (1973), who first suggested to construct graph partitions based on eigenvectors of the adjacency matrix. In the same year, Fiedler (1973) discovered that bi-partitions of a graph are closely connected with the second eigenvector of the graph Laplacian, and he suggested to use this eigenvector to partition a graph. Since then, spectral clustering has been discovered, re-discovered, and extended many times in different communities, see for example Pothen, Simon, and Liou (1990), Simon (1991), Bolla (1991), Hagen and Kahng (1992), Hendrickson and Leland (1995), Van Driessche and Roose (1995), Barnard, Pothen, and Simon (1995), Spielman and Teng (1996), Guattery and Miller (1998). A nice overview over the history of spectral clustering can be found in Spielman and Teng (1996).

In the machine learning community, spectral clustering has been made popular by the works of Shi and Malik (2000), Ng et al. (2002), Meila and Shi (2001), and Ding (2004). Subsequently, spectral clustering has been extended to many non-standard settings, for example spectral clustering applied to the co-clustering problem (Dhillon, 2001), spectral clustering with additional side information (Joachims, 2003) connections between spectral clustering and the weighted kernel-k-means algorithm (Dhillon, Guan, and Kulis, 2005), learning similarity functions based on spectral clustering (Bach and Jordan, 2004), or spectral clustering in a distributed environment (Kempe and McSherry, 2004). Also, new theoretical insights about the relation of spectral clustering to other algorithms have been found. A link between spectral clustering and the weighted kernel k-means algorithm is described in Dhillon et al. (2005). Relations between spectral clustering and (kernel) principal component analysis rely on the fact that the smallest eigenvectors of graph Laplacians can also be interpreted as the largest eigenvectors of kernel matrices (Gram matrices). Two different flavors of this interpretation exist: while Bengio et al. (2004) interpret the matrix $D^{-1/2}WD^{-1/2}$ as kernel matrix, other authors (Saerens,

Fouss, Yen, and Dupont, 2004) interpret the Moore-Penrose inverses of L or L_{sym} as kernel matrix. Both interpretations can be used to construct (different) out-of-sample extensions for spectral clustering. Concerning application cases of spectral clustering, in the last few years such a huge number of papers has been published in various scientific areas that it is impossible to cite all of them. We encourage the reader to query his favorite literature data base with the phrase "spectral clustering" to get an impression no the variety of applications.

The success of spectral clustering is mainly based on the fact that it does not make strong assumptions on the form of the clusters. As opposed to k-means, where the resulting clusters form convex sets (or, to be precise, lie in disjoint convex sets of the underlying space), spectral clustering can solve very general problems like intertwined spirals. Moreover, spectral clustering can be implemented efficiently even for large data sets, as long as we make sure that the similarity graph is sparse. Once the similarity graph is chosen, we just have to solve a linear problem, and there are no issues of getting stuck in local minima or restarting the algorithm for several times with different initializations. However, we have already mentioned that choosing a good similarity graph is not trivial, and spectral clustering can be quite unstable under different choices of the parameters for the neighborhood graphs. So spectral clustering cannot serve as a "black box algorithm" which automatically detects the correct clusters in any given data set. But it can be considered as a powerful tool which can produce good results if applied with care.

In the field of machine learning, graph Laplacians are not only used for clustering, but also emerge for many other tasks such as semi-supervised learning (e.g., Chapelle, Schölkopf, and Zien, 2006 for an overview) or manifold reconstruction (e.g., Belkin and Niyogi, 2003). In most applications, graph Laplacians are used to encode the assumption that data points which are "close" (i.e., w_{ij} is large) should have a "similar" label (i.e., $f_i \approx f_j$). A function f satisfies this assumption if $w_{ij}(f_i - f_j)^2$ is small for all i, j, that is $f'Lf$ is small. With this intuition one can use the quadratic form $f'Lf$ as a regularizer in a transductive classification problem. One other way to interpret the use of graph Laplacians is by the smoothness assumptions they encode. A function f which has a low value of $f'Lf$ has the property that it varies only "a little bit" in regions where the data points lie dense (i.e., the graph is tightly connected), whereas it is allowed to vary more (e.g., to change the sign) in regions of low data density. In this sense, a small value of $f'Lf$ encodes the so called "cluster assumption" in semi-supervised learning, which requests that the decision boundary of a classifier should lie in a region of low density.

An intuition often used is that graph Laplacians formally look like a continuous Laplace operator (and this is also where the name "graph Laplacian" comes from). To see this, transform a local similarity w_{ij} to a distance d_{ij} by the relationship $w_{ij} = 1/d_{ij}^2$ and observe that

$$w_{ij}(f_i - f_j)^2 \approx \left(\frac{f_i - f_j}{d_{ij}}\right)^2$$

looks like a difference quotient. As a consequence, the equation $f'Lf = \sum_{ij} w_{ij}(f_i - f_j)^2$ from Proposition 1 looks like a discrete version of the quadratic form associated to the standard Laplace operator $\mathcal{L}$ on $\mathbb{R}^n$, which satisfies

$$\langle g, \mathcal{L}g \rangle = \int |\nabla g|^2 dx.$$

This intuition has been made precise in the works of Belkin (2003), Lafon (2004), Hein, Audibert, and von Luxburg (2005); M., Audibert, and von Luxburg (2007), Belkin and Niyogi (2005), Hein (2006), Giné and Koltchinskii (2005). In general, it is proved that graph Laplacians are discrete versions of certain continuous Laplace operators, and that if the graph Laplacian is constructed on a similarity graph of randomly sampled data points, then it converges to some continuous Laplace operator (or

Laplace-Beltrami operator) on the underlying space. Belkin (2003) studied the first important step of the convergence proof, which deals with the convergence of a continuous operator related to discrete graph Laplacians to the Laplace-Beltrami operator. His results were generalized from uniform distributions to general distributions by Lafon (2004). Then in Belkin and Niyogi (2005), the authors prove pointwise convergence results for the unnormalized graph Laplacian using the Gaussian similarity function on manifolds with uniform distribution. At the same time, Hein et al. (2005) prove more general results, taking into account all different graph Laplacians L, L_{rw}, and L_{sym}, more general similarity functions, and manifolds with arbitrary distributions. In Giné and Koltchinskii (2005), distributional and uniform convergence results are proved on manifolds with uniform distribution. Hein (2006) studies the convergence of the smoothness functional induced by the graph Laplacians and shows uniform convergence results.

Apart from applications of graph Laplacians to partitioning problems in the widest sense, graph Laplacians can also be used for completely different purposes, for example for graph drawing (Koren, 2005). In fact, there are many more tight connections between the topology and properties of graphs and the graph Laplacian matrices than we have mentioned in this tutorial. Now equipped with an understanding for the most basic properties, the interested reader is invited to further explore and enjoy the huge literature in this field on his own.

References

Aldous, D. and Fill, J. (in preparation). *Reversible Markov Chains and Random Walks on Graphs.* online version available at http://www.stat.berkeley.edu/users/aldous/RWG/book.html.

Bach, F. and Jordan, M. (2004). Learning spectral clustering. In S. Thrun, L. Saul, and B. Schölkopf (Eds.), *Advances in Neural Information Processing Systems 16 (NIPS)* (pp. 305 – 312). Cambridge, MA: MIT Press.

Bapat, R., Gutman, I., and Xiao, W. (2003). A simple method for computing resistance distance. *Z. Naturforsch.*, *58*, 494 – 498.

Barnard, S., Pothen, A., and Simon, H. (1995). A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Applications*, *2*(4), 317 – 334.

Belkin, M. (2003). *Problems of Learning on Manifolds.* PhD Thesis, University of Chicago.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*(6), 1373 – 1396.

Belkin, M. and Niyogi, P. (2005). Towards a theoretical foundation for Laplacian-based manifold methods. In P. Auer and R. Meir (Eds.), *Proceedings of the 18th Annual Conference on Learning Theory (COLT)* (pp. 486 – 500). Springer, New York.

Ben-David, S., von Luxburg, U., and Pál, D. (2006). A sober look on clustering stability. In G. Lugosi and H. Simon (Eds.), *Proceedings of the 19th Annual Conference on Learning Theory (COLT)* (pp. 5 – 19). Springer, Berlin.

Bengio, Y., Delalleau, O., Roux, N., Paiement, J., Vincent, P., and Ouimet, M. (2004). Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, *16*, 2197 – 2219.

Ben-Hur, A., Elisseeff, A., and Guyon, I. (2002). A stability based method for discovering structure in clustered data. In *Pacific Symposium on Biocomputing* (pp. 6 – 17).

Bhatia, R. (1997). *Matrix Analysis.* Springer, New York.

Bie, T. D. and Cristianini, N. (2006). Fast SDP relaxations of graph cut clustering, transduction, and other combinatorial problems . *JMLR*, *7*, 1409 – 1436.

Bolla, M. (1991). *Relations between spectral and classification properties of multigraphs* (Technical Report No. DIMACS-91-27). Center for Discrete Mathematics and Theoretical Computer Science.

Brémaud, P. (1999). *Markov chains: Gibbs fields, Monte Carlo simulation, and queues.* New York: Springer-Verlag.

Brito, M., Chavez, E., Quiroz, A., and Yukich, J. (1997). Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics and Probability Letters*, *35*, 33 – 42.

Bui, T. N. and Jones, C. (1992). Finding good approximate vertex and edge partitions is NP-hard. *Inf. Process. Lett.*, *42*(3), 153 – 159.

Chapelle, O., Schölkopf, B., and Zien, A. (Eds.). (2006). *Semi-Supervised Learning*. MIT Press, Cambridge.

Chung, F. (1997). *Spectral graph theory* (Vol. 92 of the CBMS Regional Conference Series in Mathematics). Conference Board of the Mathematical Sciences, Washington.

Dhillon, I. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)* (pp. 269 – 274). New York: ACM Press.

Dhillon, I., Guan, Y., and Kulis, B. (2005). *A unified view of kernel k-means, spectral clustering, and graph partitioning* (Technical Report No. UTCS TR-04-25). University of Texas at Austin.

Ding, C. (2004). *A tutorial on spectral clustering.* Talk presented at ICML. (Slides available at `http://crd.lbl.gov/~cding/Spectral/`)

Ding, C., He, X., Zha, H., Gu, M., and Simon, H. (2001). A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the first IEEE International Conference on Data Mining (ICDM)* (pp. 107 – 114). Washington, DC, USA: IEEE Computer Society.

Donath, W. E. and Hoffman, A. J. (1973). Lower bounds for the partitioning of graphs. *IBM J. Res. Develop.*, *17*, 420 – 425.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Math. J.*, *23*, 298 – 305.

Fouss, F., Pirotte, A., Renders, J.-M., and Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. Knowl. Data Eng*, *19*(3), 355–369.

Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *JASA*, *97*, 611 – 631.

Giné, E. and Koltchinskii, V. (2005). Empirical graph Laplacian approximation of Laplace-Beltrami operators: large sample results. In *Proceedings of the 4th International Conference on High Dimensional Probability* (pp. 238 – 259).

Golub, G. and Van Loan, C. (1996). *Matrix computations*. Baltimore: Johns Hopkins University Press.

Guattery, S. and Miller, G. (1998). On the quality of spectral separators. *SIAM Journal of Matrix Anal. Appl.*, *19*(3), 701 – 719.

Gutman, I. and Xiao, W. (2004). Generalized inverse of the Laplacian matrix and some applications. *Bulletin de l'Academie Serbe des Sciences at des Arts (Cl. Math. Natur.)*, *129*, 15 – 23.

Hagen, L. and Kahng, A. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. Computer-Aided Design*, *11*(9), 1074 – 1085.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning*. New York: Springer.

Hein, M. (2006). Uniform convergence of adaptive graph-based regularization. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT)* (pp. 50 – 64). Springer, New York.

Hein, M., Audibert, J.-Y., and von Luxburg, U. (2005). From graphs to manifolds - weak and strong pointwise consistency of graph Laplacians. In P. Auer and R. Meir (Eds.), *Proceedings of the 18th Annual Conference on Learning Theory (COLT)* (pp. 470 – 485). Springer, New York.

Hendrickson, B. and Leland, R. (1995). An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. on Scientific Computing*, *16*, 452 – 469.

Joachims, T. (2003). Transductive Learning via Spectral Graph Partitioning. In T. Fawcett and N. Mishra (Eds.), *Proceedings of the 20th international conference on machine learning (ICML)* (pp. 290 – 297). AAAI Press.

Kannan, R., Vempala, S., and Vetta, A. (2004). On clusterings: Good, bad and spectral. *Journal of the ACM*, *51*(3), 497–515.

Kempe, D. and McSherry, F. (2004). A decentralized algorithm for spectral analysis. In *Proceedings*

of the 36th Annual ACM Symposium on Theory of Computing (STOC) (pp. 561 – 568). New York, NY, USA: ACM Press.

Klein, D. and Randic, M. (1993). Resistance distance. *Journal of Mathematical Chemistry*, *12*, 81 – 95.

Koren, Y. (2005). Drawing graphs by eigenvectors: theory and practice. *Computers and Mathematics with Applications*, *49*, 1867 – 1888.

Lafon, S. (2004). *Diffusion maps and geometric harmonics*. PhD Thesis, Yale University.

Lang, K. (2006). Fixing two weaknesses of the spectral method. In Y. Weiss, B. Schölkopf, and J. Platt (Eds.), *Advances in Neural Information Processing Systems 18* (pp. 715 – 722). Cambridge, MA: MIT Press.

Lange, T., Roth, V., Braun, M., and Buhmann, J. (2004). Stability-based validation of clustering solutions. *Neural Computation*, *16*(6), 1299 – 1323.

Lovász, L. (1993). Random walks on graphs: a survey. In *Combinatorics, Paul Erdös is eighty* (pp. 353 – 397). Budapest: János Bolyai Math. Soc.

Lütkepohl, H. (1997). *Handbook of Matrices*. Chichester: Wiley.

M., Audibert, J.-Y., and von Luxburg, U. (2007). Graph laplacians and their convergence on random neighborhood graphs. *JMLR*, *8*, 1325 – 1370.

Meila, M. and Shi, J. (2001). A random walks view of spectral segmentation. In *8th International Workshop on Artificial Intelligence and Statistics (AISTATS)*.

Mohar, B. (1991). The Laplacian spectrum of graphs. In *Graph theory, combinatorics, and applications. Vol. 2 (Kalamazoo, MI, 1988)* (pp. 871 – 898). New York: Wiley.

Mohar, B. (1997). Some applications of Laplace eigenvalues of graphs. In G. Hahn and G. Sabidussi (Eds.), *Graph Symmetry: Algebraic Methods and Applications* (Vol. NATO ASI Ser. C 497, pp. 225 – 275). Kluwer.

Nadler, B., Lafon, S., Coifman, R., and Kevrekidis, I. (2006). Diffusion maps, spectral clustering and eigenfunctions of Fokker-Planck operators. In Y. Weiss, B. Schölkopf, and J. Platt (Eds.), *Advances in Neural Information Processing Systems 18* (pp. 955 – 962). Cambridge, MA: MIT Press.

Ng, A., Jordan, M., and Weiss, Y. (2002). On spectral clustering: analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14* (pp. 849 – 856). MIT Press.

Norris, J. (1997). *Markov Chains*. Cambridge: Cambridge University Press.

Penrose, M. (1999). A strong law for the longest edge of the minimal spanning tree. *Ann. of Prob.*, *27*(1), 246 – 260.

Pothen, A., Simon, H. D., and Liou, K. P. (1990). Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Anal. Appl.*, *11*, 430 – 452.

Saerens, M., Fouss, F., Yen, L., and Dupont, P. (2004). The principal components analysis of a graph, and its relationships to spectral clustering. In *Proceedings of the 15th European Conference on Machine Learning (ECML)* (pp. 371 – 383). Springer, Berlin.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888 – 905.

Simon, H. (1991). Partitioning of unstructured problems for parallel processing. *Computing Systems Engineering*, *2*, 135 – 148.

Spielman, D. and Teng, S. (1996). Spectral partitioning works: planar graphs and finite element meshes. In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)* (pp. 96 – 105). Los Alamitos, CA: IEEE Comput. Soc. Press. (See also extended technical report.)

Stewart, G. and Sun, J. (1990). *Matrix Perturbation Theory*. New York: Academic Press.

Still, S. and Bialek, W. (2004). How many clusters? an information-theoretic perspective. *Neural Comput.*, *16*(12), 2483 – 2506.

Stoer, M. and Wagner, F. (1997). A simple min-cut algorithm. *J. ACM*, *44*(4), 585 – 591.

Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the number of clusters in a dataset via the gap statistic. *J. Royal. Statist. Soc. B*, *63*(2), 411 – 423.

Van Driessche, R. and Roose, D. (1995). An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel Comput.*, *21*(1), 29 – 48.

von Luxburg, U., Belkin, M., and Bousquet, O. (to appear). Consistency of spectral clustering. *Annals of Statistics*. (See also Technical Report 134, Max Planck Institute for Biological Cybernetics, 2004)

von Luxburg, U., Bousquet, O., and Belkin, M. (2004). On the convergence of spectral clustering on random samples: the normalized case. In J. Shawe-Taylor and Y. Singer (Eds.), *Proceedings of the 17th Annual Conference on Learning Theory (COLT)* (pp. 457 – 471). Springer, New York.

von Luxburg, U., Bousquet, O., and Belkin, M. (2005). Limits of spectral clustering. In L. Saul, Y. Weiss, and L. Bottou (Eds.), *Advances in Neural Information Processing Systems (NIPS) 17* (pp. 857 – 864). Cambridge, MA: MIT Press.

Wagner, D. and Wagner, F. (1993). Between min cut and graph bisection. In *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science (MFCS)* (pp. 744 – 750). London: Springer.