A Tutorial on Spectral Clustering

Ulrike von Luxburg

2 Similarity graphs

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**. However, we connect 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.

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*

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

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,

$$f' L f = f' D f - f' W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$
$$= \frac{1}{2}\Big(\sum_{i=1}^{n} d_i f_i^2 - 2 \sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2 \Big) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

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 $f' L f \geq 0$ 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

As the weights w_{ij} are non-negative, this sum can only vanish if all terms $w_{ij}(f_i - f_j$

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 es2.s

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 $A_i = \{j \mid y_j \in C_i\}$.

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 $A_i = \{j \mid y_j \in C_i\}$.

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

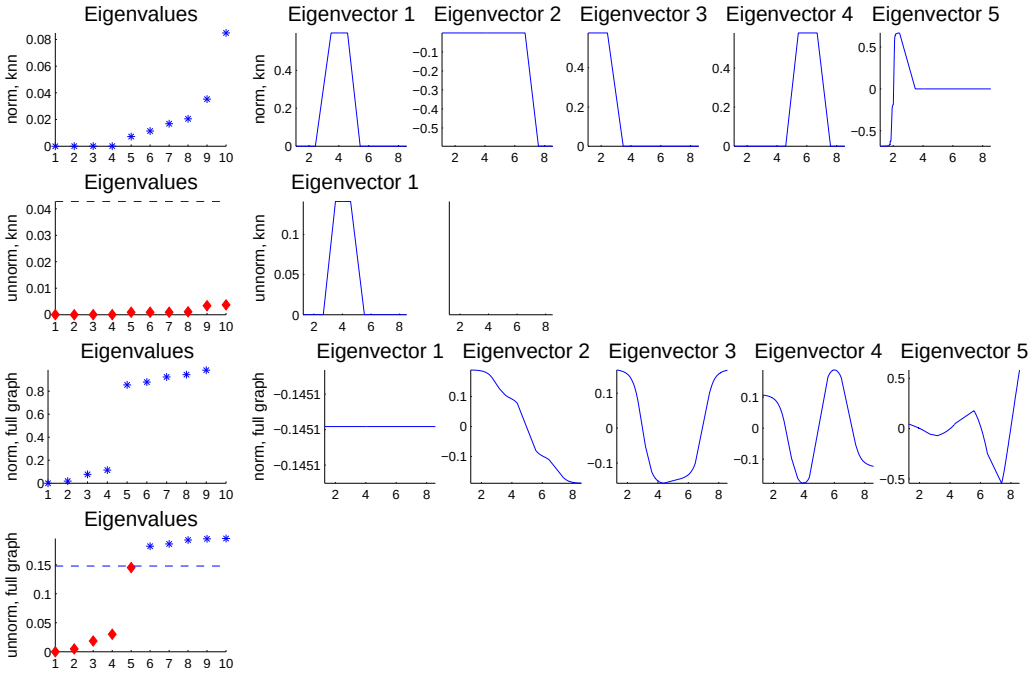

Figure 1: Toy example for spectral clustering where the data points have been drawn from a mixture of four Gaussians on $\mathbb{R}$. Left upper corner: histogram of the data. First and second row: eigenvalues and eigenvectors of L_{rw} and L based on the k-nearest neighbor graph. Third and fourth row: 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 $x_1, \dots, x_{200} \in \mathbb{R}$ 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 $u = (u_1, \dots, u_{200})$ 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

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

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

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 $H^T H = I$. Similar to the calculations in the last section we can see that

$$h_i^T L h_i = \frac{\mathrm{cut}(A_i, \overline{A}_i)}{|A_i|}.$$

Moreover, one can check that

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

Combining those facts we get

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

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

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

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:

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

This is the standard form of a trace minimization problem, and again a version of the Rayleigh-Ritz

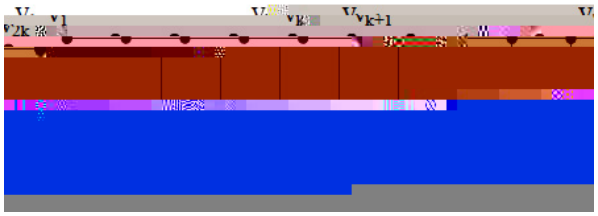

Figure 2: The cockroach graph from Guattery and Miller (1998).

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 $T = D$

Of course, the relaxation we discussed above is not unique. For example, a completely di□erent relax-

Using this we obtain

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

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 $\lambda_i = 0$ 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

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

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

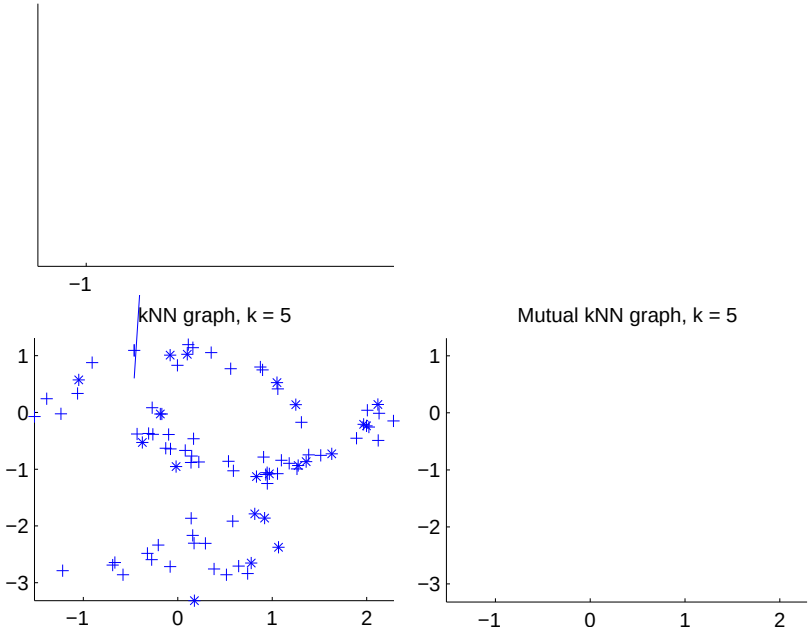

Figure 3: Different similarity graphs, see text for details.

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

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., $k \approx \log(n) + 1$). 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.

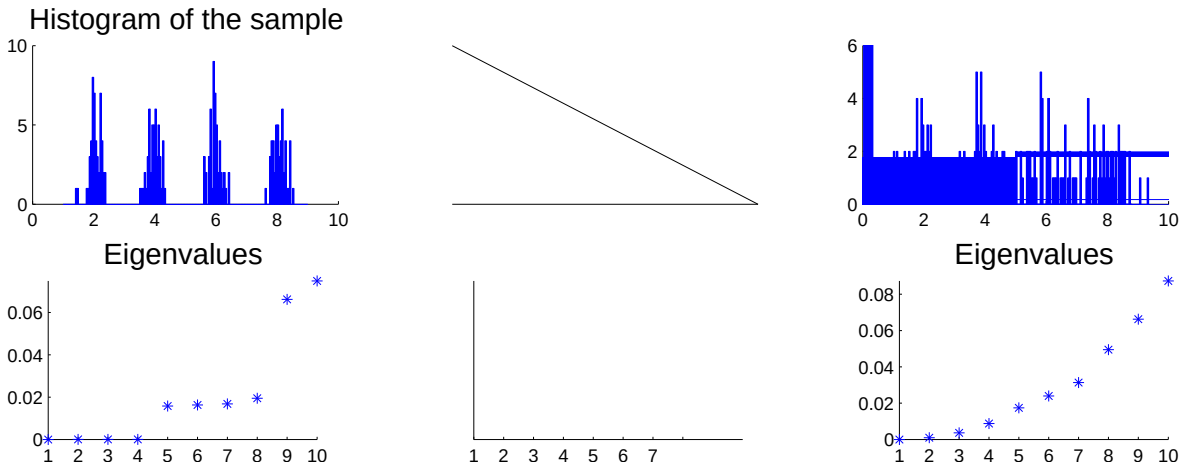

Figure 4: Three data sets, and the smallest 10 eigenvalues of L_{rw}. See text for more details.

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 a□ect 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 di□cult 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

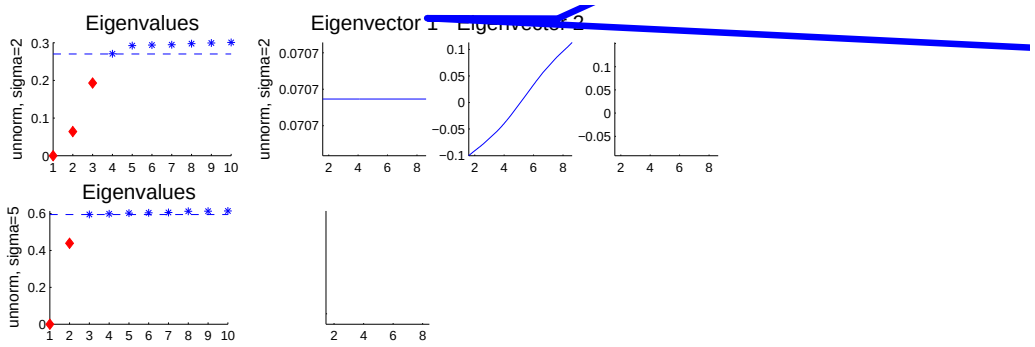

Figure 5: Consistency of unnormalized spectral clustering. Plotted are eigenvalues and eigenvectors of L, for parameter $\sigma = 2$ (first row) and $\sigma = 5$ (second row). The dashed line indicates $\min d_j$, the eigenvalues below $\min d_j$ are plotted as red diamonds, the eigenvalues above $\min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

to an operator U on the space $C(X)$ of continuous functions on 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 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 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 $C(X)$ 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

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-

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 $f\,L\,f$ is small. With this intuition one can use the quadratic form $f\,L\,f$ 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 $f\,L\,f$ 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 $f\,L\,f$ encodes the so called "cluster assumption"

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. Schë2¨ andow functi051.E6(w

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). *Di usion maps and geometric harmonics*. PhD Thesis, Yale University.

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*