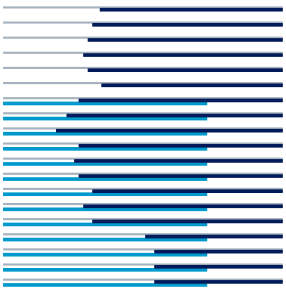
New Methods for Spectral Clustering

Igor Fischer
School of Computer Science and Engineering
Hebrew University, Jerusalem 91904, Israel
fischer@cs.huji.ac.il

Jan Poland
IDSIA, Galleria 2,
6928 Manno-Lugano, Switzerland
jan@idsia.ch



Technical Report No. IDSIA-12-04
June 21, 2004

IDSIA / USI-SUPSI
Dalle Molle Institute for Artificial Intelligence
Galleria 2, 6928 Manno, Switzerland

Technical Report IDSIA-12-04 June 21, 2004

New Methods for Spectral Clustering

Igor Fischer
School of Computer Science and Engineering
Hebrew University, Jerusalem 91904, Israel
fischer@cs.huji.ac.il

Jan Poland*
IDSIA, Galleria 2,
6928 Manno-Lugano, Switzerland
jan@idsia.ch

Abstract

Analyzing the affinity matrix spectrum is an increasingly popular data clustering method. We propose three new algorithmic components which are appropriate for improving performance of spectral clustering. First, observing the eigenvectors suggests to use a K-lines algorithm instead of the commonly applied K-means. Second, the clustering works best if the affinity matrix has a clear block structure, which can be achieved by computing a conductivity matrix. Third, many clustering problems are inhomogeneous or asymmetric in the sense that some clusters are concentrated while others are dispersed. In this case, a context-dependent calculation of the affinity matrix helps. This method also turns out to allow a robust automatic determination of the kernel radius σ.

Keywords: Spectral Clustering, Affinity Matrix, Conductivity, Hierarchy

1 Introduction

Spectral clustering is is appealingly simple: Given some data, you build an affinity (or kernel) matrix, analyze its spectrum, and often get a perfect clustering from the dominant eigenvectors for free. This simple algorithm or its slightly more complex variants which yield so good results are widely appreciated for applications. As a result, a whole family of algorithms and variants has been published recently, see e.g. papers by Scott and Longuet-Higgins (14); Weiss (19); Meilă and Shi (8); Ng et al. (10); Shi and Malik (15); Verma and Meilă (18). Although spectral clustering methods are still incompletely understood and no closed theory for their functioning and limitations is yet available, there is progress towards this goal, e.g. approaches using matrix perturbation theory (see 10).

Spectral methods are computationally rather expensive (typically cubic in the data set size), but advantageous in situations where the shape of the cluster is other than approximately Gaussian, like examples in Figures 4 and 6. Although there are other possible clustering algorithms to tackle these problems (e.g. probability estimation), we will fully concentrate on the spectral method in the sequel.

We shall try to find some insights in the way spectral clustering works. These observations will motivate the use of our "K-lines" algorithm instead of the usual K-means for finally identifying the clusters. This is subject of the next section 2. In section 3, we suggest a new

*Supported by SNF grant 2100-67712.02.

method for reinforcing the block structure of the affinity matrix. It may improve performance considerably in case of clusters which are far from spherical. This is the most important case to prefer spectral methods over simple ones such as K-means. Section 4 deals with the problem of finding good radii for the kernel used for building the affinity matrix. We suggest a selection criterion based on the distance diagram of the data. Then we develop a new context-dependent way of choosing the radius, which results in a different choice for each data point. The resulting affinity matrix thus becomes asymmetric. This method provides a surprisingly easy solution for the generally difficult problem of automatically determining the radius. Moreover, it will turn out to be valuable particularly if the clusters are very inhomogeneous in the sense that some clusters are concentrated and others are widespread. The final algorithm and its variants will be stated section 5. In section 6, we discuss alternative methods. Simulation results and comparison with known spectral clustering algorithms are presented in section 7).

Before we start, we shall outline the basic algorithm on which all spectral clustering methods rely.

Algorithm 1: Basic spectral clustering
1. Build the affinity matrix
2. Determine the dominant eigenvalues and eigenvectors of the matrix
3. Use these to compute the clustering

In the third step of this general procedure, most often a simple metric clustering algorithm is applied, in particular K-means. Instead, we will propose the K-lines algorithm for this task.

It is our main goal to propose new methods that are intuitively sensible and show a convincing performance on benchmark problems. Thus, the following analysis will remain on an informal level. We still hope that also theorists will be interested in our methods, since they are exhibit a certain simplicity and elegance. On the other hand, a formal analysis of our methods would be desirable and will be subject of future research. However, there is an inherent problem with the quantitative analysis of clustering algorithms, namely the fact that the clustering task itself is not well-defined in general. (But compare e.g. (7) for this issue.) This is supposably the main reason for the difficulty of developing closed theories of clustering and spectral clustering in particular.

2 Spectral Analysis and K-Lines

We believe that spectral clustering methods can be best interpreted as tools for analysis of the *block structure* of the affinity matrix. The affinity matrix is a weighted adjacency matrix of the data. All data can be considered to form vertices of a weighted graph, larger weights implying higher similarity between the points. In this paper we consider only non-negative weight functions, leading to non-negative affinity matrices.

For illustrative purposes we shall start with idealized, un-weighted adjacency matrices:

$$\boldsymbol{A}(i,j) = \boldsymbol{A}(j,i) = \begin{cases} 1 & \text{if } \mathrm{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\mathrm{sim}(\cdot,\cdot)$ denotes a similarity function and θ some fixed threshold. Let us now imagine that the data form K clear, disjunct clusters $\mathcal{C}_k$, so that $\mathrm{sim}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \theta \Leftrightarrow \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{C}_k$. Then, assuming an appropriate enumeration of the points, the matrix will be block-diagonal, with blocks of sizes $n_k \times n_k$.

¿From the defining equation $\boldsymbol{A}\boldsymbol{e} = \lambda\boldsymbol{e}$ it is easy to see that $\lambda_k = n_k$ are the nonzero eigenvalues of $\boldsymbol{A}$ and that the associated eigenvectors $\boldsymbol{e}_k$ can be composed as follows:

$$\boldsymbol{e}_k(i) = \begin{cases} 1 & \text{for all } i : \sum_{l=1}^{k-1} n_l < i \leq \sum_{l=1}^{k} n_l \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The eigenvectors can, of course, be scaled by an arbitrary nonzero factor. Recalling how the affinity matrix was constructed, it is obvious that the following holds:

$$\boldsymbol{x}_i \in \mathcal{C}_k \Leftrightarrow \boldsymbol{e}_k(i) \neq 0 \tag{3}$$

Different enumeration of the points does not change the veracity of the above statement, since then the eigenvectors become permuted accordingly. Bearing this in mind, we can restrict our discussion to nicely permuted matrices.

Departing from the above simple case, we now allow different weights of graph edges. For numerical data, a convenient weighting function is the Gaussian kernel

$$\boldsymbol{A}(i,j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \tag{4}$$

As noted by Perona and Freeman (11), there is nothing magical with this function. Any symmetrical, non-negative function monotonously falling with increasing distance can be applied. A slight advantage of the Gaussian function is that it results in a positive definite affinity matrix – a kernel matrix, – somewhat simplifying the analysis of eigenvalues. The parameter σ is a user-defined value, referred to as the kernel width. The choice of a correct width is critical for the performance of the algorithm. We shall return later, in section 4, to the question how to choose a good σ and assume for the moment that it has a sensible value. Then the points belonging to the same cluster will result in affinity matrix entries close to one, whereas for the points from different clusters, the entries will be close to zero. Thus, for nicely enumerated data the matrix still resembles a block-diagonal matrix (see examples in Figures 1 and 2).

Contrary to the idealized case with strictly block-diagonal affinity matrix, now many eigenvalues and eigenvector components will be nonzero. The simple rule (3) cannot be applied any more, but can serve as motivation. As the real-valued affinity matrix approaches binary, few clearly dominant eigenvalues emerge. Also, eigenvector components diminish for non-cluster points and approach some clearly defined nonzero value for points belonging to the cluster associated with the eigenvector. To cluster data we can therefore observe the eigenvectors associated with the dominant eigenvalues. This idea is illustrated in Figure 1. There are three dominant eigenvalues, and the correspondence between the eigenvector components and cluster membership is obvious.

Figure 2 shows a slightly more complicated setting, where the two right clusters are so close that they partially overlap. In the affinity matrix, two blocks are visible, with the larger containing two smaller sub-blocks. This structure is reflected in the dominant eigenvectors: the second has large, positive components for points from both right clusters, whereas the third has mostly negative entries for the points from the second cluster and positive for the third.

We propose the following interpretation: all points form the two right clusters form a super-cluster, as reflected by the second eigenvector. At the same time, they belong to

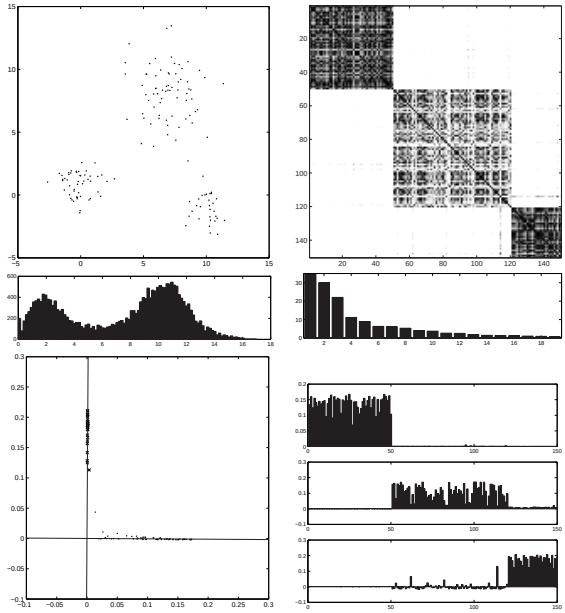


Figure 1: Three well separated clusters. **Left column:** scatter plot of the data (top); distance histogram (middle); spectral plot along the 2nd and 3rd eigenvector (bottom). **Right column:** Affinity matrix $\boldsymbol{A}$ of the data (top); 20 largest eigenvalues of $\boldsymbol{A}$ (middle); components of the first three eigenvectors of $\boldsymbol{A}$ (bottom).

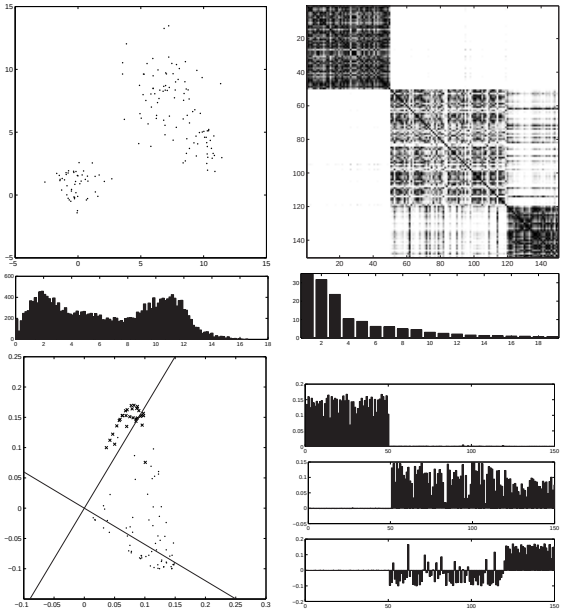


Figure 2: Three less separated clusters. Visualized are the same values as in Figure 1. The rightmost, lower cluster from Figure 1 is shifted upwards towards the central one. Note how a weaker super-block in the affinity matrix appears, encompassing the lower two blocks, and how the spectral plot is rotated.

two different sub-clusters, as reflected in the positive and negative components of the third eigenvector. In other words, the third eigenvector splits the cluster into its two constitutive subclusters. We will use this observation later for exploring the hierarchical data structure.

This example also illustrates another reason why the rule (3) cannot be applied directly. The space spanned by the dominant eigenvectors can be invariant under rotation for the given data set. For example, both $\boldsymbol{e}_1 = (1,0)^\intercal, \boldsymbol{e}_2 = (0,1)^\intercal$ and $\boldsymbol{e}_1 = (1/\sqrt{2}, 1/\sqrt{2})^\intercal, \boldsymbol{e}_2 = (1/\sqrt{2}, -1/\sqrt{2})^\intercal$ are equally valid decompositions for the identity matrix $\boldsymbol{A} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$. For real-world data, slight perturbations in the data can lead to unpredictable rotation of the eigenvectors. Still, cluster membership of a point $\boldsymbol{x_i}$ is determined only by the i-th components of the eigenvectors (otherwise a different enumeration of the points would influence the clustering). Therefore, we can try to cluster the points by observing all dominant eigenvectors simultaneously. The meanwhile standard approach, which we shall also follow, is to observe the spectral images of the data, i.e. the D-dimensional vectors $\boldsymbol{y}_i$, whose components are the i-th components of the D dominant eigenvectors:

$$\boldsymbol{y}_i(1) = \boldsymbol{e}_1(i), \quad \boldsymbol{y}_i(2) = \boldsymbol{e}_2(i), \quad \ldots, \quad \boldsymbol{y}_i(D) = \boldsymbol{e}_D(i) \tag{5}$$

For the moment we will agree tacitly that the number of clusters K is simultaneously the number D of dominant eigenvectors to use. We will return to this issue shortly.

We shall state one empirical observation concerning the $\boldsymbol{y}$-vectors. For clearly distinct, convex clusters, the spectral images are nicely distributed along straight lines passing through

the origin (Figure 1). As clusters begin to overlap, the points disperse angularly around the lines. The lines, in addition, get rotated proportional to the degree the clusters form a common super-cluster (Figure 2). Thus the problem of clustering original points $\boldsymbol{x}_i$ can be transformed into clustering their spectral images. The latter problem is often easier to solve: we only need to find the typical vector, the one lying on the line, for each cluster.

2.1 K-lines

These observations immediately motivate the following algorithm for transforming the spectral images $\boldsymbol{y}_i$ into clustering information. We distinguish between the dimension of the vectors $\boldsymbol{y}_i$, which is D, and the number of clusters, which is K. Recall that up to now we agreed on $D = K$, this point will be treated in the next subsection.

Algorithm 2: K-lines

1. Initialize vectors $\boldsymbol{m}_1 ... \boldsymbol{m}_K$ (e.g. randomly, or as the first K eigenvectors of the spectral data $\boldsymbol{y}_i$)
2. **for** j=1 ... K:
   Define $\mathcal{P}_j$ as the set of indices of all points $\boldsymbol{y}_i$ that are closest to the line defined by $\boldsymbol{m}_j$, and create the matrix $\boldsymbol{M}_j = [\boldsymbol{y}_i]_{i \in \mathcal{P}_i}$ whose columns are the corresponding vectors $\boldsymbol{y}_i$
3. Compute the new value of every $\boldsymbol{m}_j$ as the first eigenvector of $\boldsymbol{M}_j \boldsymbol{M}_j^\intercal$
4. Repeat from 2 until $\boldsymbol{m}_j$'s do not change

The mean vectors $\boldsymbol{m}_j$ are prototype vectors for each cluster scaled to the unit length. Each $\boldsymbol{m}_j$ defines a line through the origin. By computing $\boldsymbol{m}_j$ as principal eigenvector of $\boldsymbol{M}_j \boldsymbol{M}_j^\intercal$, one asserts that the sum of square distances of the points $\boldsymbol{y}_i$ to the respective line defined by $\boldsymbol{m}_j$ is minimal:

$$\sum_{i \in \mathcal{P}_j} \|\boldsymbol{y}_i - \langle \boldsymbol{y}_i, \boldsymbol{m}_j \rangle \boldsymbol{m}_j\|^2 = \mathbf{min}! \tag{6}$$

To see this, observe that

$$\sum_{i \in \mathcal{P}_j} \|\boldsymbol{y}_i - \langle \boldsymbol{y}_i, \boldsymbol{m}_j \rangle \boldsymbol{m}_j\|^2 = \sum_{i \in \mathcal{P}_j} \left(\|\boldsymbol{y}_i\|^2 - \langle \boldsymbol{y}_i, \boldsymbol{m}_j \rangle^2 \right) = \sum_{i \in \mathcal{P}_j} \|\boldsymbol{y}_i\|^2 - \boldsymbol{m}_j^\intercal \boldsymbol{M}_j \boldsymbol{M}_j^\intercal \boldsymbol{m}_j. \tag{7}$$

So in order to minimize the above term, one has to maximize $\boldsymbol{m}_j^\intercal \boldsymbol{M}_j \boldsymbol{M}_j^\intercal \boldsymbol{m}_j$ subject to $\|\boldsymbol{m}_j\| = 1$, which is done by choosing the principal eigenvector.

Clustering of the original data $\boldsymbol{x}_i$ is then performed according to the rule:

Assign $\boldsymbol{x}_i$ to the j-th cluster if the line determined by $\boldsymbol{m}_j$ is the nearest line to $\boldsymbol{y}_i$. (8)

At this point the reader might wonder why we introduce the K-lines algorithm instead of using the K-means algorithm which is commonly applied for the final clustering. The answer is that the above analysis indicates that the spectral images are aligned to lines rather than spherical. Simulations also support this conjecture, see sections 6 and 7. Then the transformation of K-means to K-lines is quite natural. (Similar transformations are known for other types of clustering problems, e.g. the K-planes algorithm by Bradley and Mangasarian.)

2.2 How many clusters and eigenvectors?

We have motivated above that the dominant eigenvectors are those carrying the most information about the clustering. Moreover, in our examples with Kclusters so far, always the $D = K$ dominant eigenvectors have been sufficient. This needs not be always the case. Namely, among the K largest eigenvectors may be at least one that does not help for the clustering. Consider the following example: The data set contains two clusters. One is very concentrated and well separated from the second, but contains only few points. The other cluster is irrelevantly divided in two overlapping sub-clusters both of which contain even more points than the first cluster. Then the first two dominant eigenvectors will both "describe" the second cluster, ignoring the first one. Only the third eigenvector points to the first cluster. However, if K-lines is applied to the image of the first *three* eigenvectors, the result is likely to be the correct clustering, since the overlapping sub-clusters are less well separated.

Other reasons for the necessity to consider more than K dominant eigenvectors may be overlapping clusters. Of course the situation is still relatively simple as long as we know the "true" number of clusters K. If K is unknown, then one can try to recover that number from the spectral analysis. Several according methods are possible in principle. Here we will argue that those techniques are likely to fail in other than easy cases.

Again the situation is most simple for data sets forming clearly separated, convex and not too elongated clusters. Then there is a significant drop between dominant and non-dominant values (see Figure 1). For more complex data sets, the choice can be harder, because the eigenvalues decrease smoothly. A method proposed by Girolami (6) relies on dominant terms in

$$\sum_{i=1}^{N} \lambda_i \{\mathbf{1}_N^\top \boldsymbol{e}_i\}^2 \tag{9}$$

where $\mathbf{1}_N$ is a shorthand notation for an N-dimensional vector with all components equal to $1/N$, N being the number of points. It was claimed that if there are K distinct clusters within the data, there are K dominant terms $\lambda_i\{\mathbf{1}_N^\top \boldsymbol{e}_i\}^2$ in the above summation. The statement was illustrated by several examples, but counterexamples are also easy to find. Consider the identity matrix from the previous section: depending on the eigenvalue decomposition we choose, we obtain either two dominant terms, both equal to one, or only one term equal to two. Generally it can be said that the method is likely to fail when clusters overlap, but it may worth trying if no obvious dominant eigenvalues exist.

In the remainder of the paper, we avoid this problem by assuming that the number of clusters K is always known. Although we might want to consider more than D dominant eigenvectors even then as observed above, we restrict to $K = D$. The reason is twofold. Considering more eigenvectors may improve the clustering and slows down the computation definitely, but it introduces another degree of freedom in the first place. So we keep things simple by setting $K = D$, since the simulations do not strongly refute this restriction. Second, setting $D > K$ (e.g. $D = 2K$) in order to improve performance for an application is an easy and straightforward modification which needs no further discussion.

Of course we could also keep things simple by taking *all* $D = N$ eigenvectors. In fact, in the context of graph partitioning there are arguments that it is best to do so, see e.g. (1). Since our goal also includes performance in terms of computation time, we still prefer $D = K$. Since normally $N \gg K$ holds and the K-lines algorithm involves a repeated eigenvector computation of a $D \times D$ matrix, $D = N$ results in a significantly slower algorithm. In all examples we consider in the sequel, $D = K$ yields optimal clustering, and no improvement

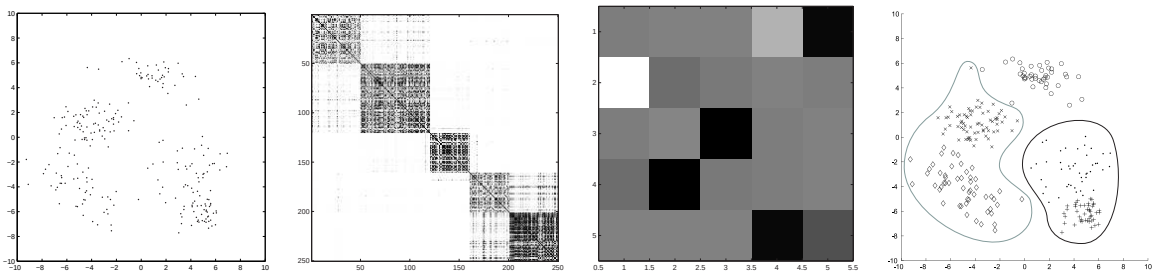


Figure 3: Five clusters with partial overlap. **From left-to-right:** scatter plot of the data; Affinity matrix $\boldsymbol{A}$; Matrix of prototypical vectors $\boldsymbol{M}$ (black $\equiv 1$, white $\equiv -1$); Hierarchically clustered data.

can be achieved for $D > K$.

2.3 Hierarchical structure

The idea of exploring cluster hierarchy by observing the affinity matrix spectrum was proposed already in (14). We develop it further, relying only on the vectors determining the K lines around which the spectral images $\boldsymbol{y}_i$ are distributed. Recall the above observation that their rotation depends on the amount of cluster overlap, unless the cluster sizes are almost equal. For fully disjunctive clusters, provided all eigenvalues are different and the restricted affinity matrix therefore is not rotation invariant, these axes are close to the coordinate axes. For overlapping clusters, where both clusters are expressed to a same extent in the super-cluster, the spectral axes are rotated for 45 degrees. In an intermediate case, the axes are rotated for a smaller amount (see Figures 1 and 2). The axes' rotation stems from the way the point membership to clusters is represented in eigenvectors. In the eigenvector describing the super-cluster, the components corresponding to the points of both sub-clusters have same sign, thus stretching the spectral images $\boldsymbol{y}_i$ along a same axis. In the eigenvectors describing the sub-clusters, the components corresponding to the points of different sub-clusters have different sign, distributing $\boldsymbol{y}_i$'s accordingly on the positive and negative side of another coordinate axis.

The axes passing through points $\boldsymbol{y}_i$ are determined by the vectors $\boldsymbol{m}_j$, which are prototypical spectral vectors for different clusters. So by examining their components, we can obtain information about the hierarchical data structure. Let us construct a matrix $\boldsymbol{M}$ whose columns are the vectors $\mathbf{m}_j$. Now, if any row in the matrix contains large values of a same sign – i.e. components of two prototypical vectors are comparable – this is an indication for a clusters overlap. Clusters described by the columns in which the large values are located form a super-cluster. A complementary row also exists, in which the entries of the columns are also large, but with opposite signs. This latter row indicates the splitting of the super-cluster into sub-clusters.

We illustrate this using an example with five clusters, as shown in Figure 3. Second from right is a graphical representation of a matrix $\boldsymbol{M}$, with dark blocks representing large positive values and white large negative. For demonstration purposes we have ordered the columns of $\boldsymbol{M}$ to reflect our enumeration of the points, so that the first vector in $\boldsymbol{M}$ describes the first cluster, the second vector the second cluster and so on. In the fifth row we see large, positive values at the positions four and five, indicating that the fourth and the fifth cluster form a super-cluster. The first row, with large positive and negative values at the same positions

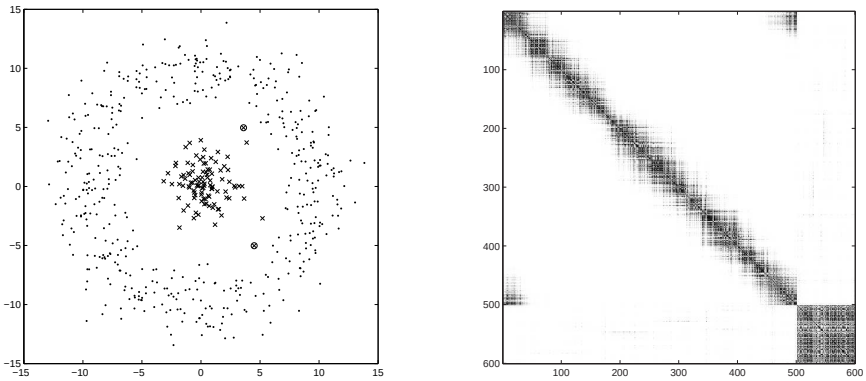


Figure 4: Ring and spherical cluster. **Left:** scatter plot of the data (the circles are incorrectly clustered points explained later). **Right:** The affinity matrix $\boldsymbol{A}$.

provides for splitting the super-cluster. We also notice a less expressed overlap of clusters 1 and 2, indicated by the second and fourth row. Based on these observations, we are able to draw a hierarchical data structure, as shown in the same figure on the right.

We can summarize that a hierarchy information about the clustering may be quite easily obtainable from spectral methods if desired, and is easy to visualize through the $\boldsymbol{M}$ matrix. In the remaining part of the paper however, we will ignore the hierarchy aspect and restrict to standard clustering tasks.

3 Block Structure and Conductivity

We have shown that clustering by spectral analysis works well for block-diagonal-like affinity matrices. We have also argued that the affinity matrix is approximately block-diagonal if we apply the Gaussian kernel on data forming clear clusters. The implicit assumption is that all points in a cluster are relatively close to each other and that the clusters are far away. In other words, the clusters are assumed to be convex, compact and well separated. This description applies to many natural data sets, but other configurations corresponding with the intuitive understanding of clusters are also possible.

An easy example is a ring-like cluster encircling a convex, circular one in the ring center (Figure 4). The ring is obviously not convex: the points at the opposite sides of the ring are far from each other, further than from points from the central cluster. Nevertheless, according to our common-sense understanding, we would say that points in the ring form one and the points in the center another cluster. If we compute the affinity matrix of the data, we see that only the central cluster results in a block, whereas the ring cluster produces a diagonal band in the matrix. With a suitable kernel width and sufficient separation of the clusters, the diagonal band is wide. If we are lucky, it approximates a block closely enough for our algorithm to perform correct clustering. This is, however, not the expected data configuration for the algorithm, so it cannot be expected to work well in general.

In order to handle cases where data do not form compact and convex clusters, we have to extend our definition of a cluster. We note that we intuitively regard clusters as continuous concentration of data points, a notion which was applied by Ben-Hur et al. (2). Two points belong to a cluster if they are close to each other, or if they are well connected by paths of short "hops" over other points. The more such paths exist, the higher the chances are that

the points belong to a cluster.

To quantify cluster membership, we introduce a new affinity measure. Recall the original weighted graph, where edges are assigned larger weights for closer points. Instead of considering two points similar if they are connected by a high-weight edge, we assign them a high affinity if the overall graph conductivity between them is high. These considerations exhibit a striking analogy to electrical networks: the conductivity between two nodes depends not only on the conductivity of the direct path between them, but also on all other indirect paths. This conductivity measure should not be confused with the graph "conductance" by Sinclair and Jerrum (16), which is – up to a constant factor – equivalent to the normalized cuts applied by Shi and Malik (15). In the normalized cuts approach, the overall flow between two disjunct graph parts is considered. Our approach is, in a sense, complementary: we consider the flow between any two points, and construct a new graph based on it.

The conductivity for any two points $\boldsymbol{x}_p$ and $\boldsymbol{x}_q$ is easily computed and can be found in textbooks of electrical engineering. We first solve the system of linear equations:

$$\boldsymbol{G}\boldsymbol{\varphi} = \boldsymbol{\eta} \tag{10}$$

where $\boldsymbol{G}$ is a matrix constructed from the original affinity matrix $\boldsymbol{A}$:

$$\boldsymbol{G}(p,q) \quad = \quad \begin{cases} \text{for } p = 1: & \begin{cases} 1 & \text{for } q = 1 \\ 0 & \text{otherwise} \end{cases} \\ \text{otherwise}: & \begin{cases} \sum_{k \neq p} \boldsymbol{A}(p,k) & \text{for } p = q \\ -\boldsymbol{A}(p,q) & \text{otherwise} \end{cases} \end{cases} \tag{11}$$

and $\boldsymbol{\eta}$ is the vector representing points for which the conductivity is computed:

$$\boldsymbol{\eta}(k) = \begin{cases} -1 & \text{for } k = p \text{ and } p > 1 \\ 1 & \text{for } k = q \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Then the conductivity between $\boldsymbol{x}_p$ and $\boldsymbol{x}_q$, $p < q$ is given by

$$\boldsymbol{C}(p,q) = \frac{1}{\boldsymbol{\varphi}(q) - \boldsymbol{\varphi}(p)} \tag{13}$$

which, due to the way $\boldsymbol{\eta}$ is constructed, can be simplified to

$$\boldsymbol{C}(p,q) = \frac{1}{\boldsymbol{G}^{-1}(p,p) + \boldsymbol{G}^{-1}(q,q) - \boldsymbol{G}^{-1}(p,q) - \boldsymbol{G}^{-1}(q,p)} \tag{14}$$

Due to the symmetry, $\boldsymbol{C}(p,q) = \boldsymbol{C}(q,p)$. It therefore suffices to compute $\boldsymbol{G}^{-1}$ only once, in $O(N^3)$ time, and to compute the conductivity matrix $\boldsymbol{C}$ in $O(N^2)$ time.

In electrical engineering, the method above is known as node analysis. To compute the overall conductivity between two nodes i and j in a resistor network, we measure the voltage U_{ij} between them when we let a known current I enter the network at one and leave it at the other node. The overall conductivity is then given by Ohm's law: $G_{ij} = I/U_{ij}$. The voltage is defined as the potential difference between the nodes: $U_{ij} = \varphi_j - \varphi_i$, and the potentials can be computed from Kirchhoff's law, stating that all currents entering a node i must also leave it: $\sum_{j \neq i} I_{ij} = 0$. Applying Ohm's law again, the currents can be expressed over voltages and

conductivities, so that this Equation becomes: $\sum_{j \neq i} G_{ij} U_{ij} = G_{ij}(\varphi_j - \varphi_i) = 0$. Grouping the direct conductivities by the corresponding potentials and formulating the Equation for all nodes, we obtain the matrix Equation (10). The vector $\boldsymbol{\eta}$ represents the known current I, which we have transferred to the right side of the Equation.

It can be easily seen that in a system of N nodes only $N-1$ are linearly independent. If we would compose $\boldsymbol{G}$ relying only on Kirchhoff's and Ohm's law, its rows would sum to zero, i.e. the system would be undetermined. In a physical sense, currents entering and leaving $N-1$ nodes determine also the currents in the N-th node, since they have nowhere else to go. In order to obtain a determined system, we have to choose a node and fix it to a known potential, so it becomes the reference node. In our method we set the potential of the first node to zero ($\varphi_1 = 0$), which is reflected by the way the first rows of $\boldsymbol{G}$ and $\boldsymbol{\eta}$ are defined in Equations (11) and (12).

The method here seems to require solving the Equations anew for every pair of nodes – the computational analogy of connecting the current source between all pairs of nodes and measuring the voltage. This is, fortunately, not the case: First, since direct conductivities between nodes do not change, it suffices to invert the matrix $\boldsymbol{G}$ only once. And second, for computing the overall conductivity between two nodes, we do not need all voltages in the network, the voltage between these nodes suffices. This allows us to observe only two rows in the $\boldsymbol{G}^{-1}$ matrix. Further, due to the fact that all except two components of vector $\boldsymbol{\eta}$ are zeros (i.e. the external current source is attached only to two nodes), we only need to consider two columns in $\boldsymbol{G}^{-1}$. Consequently, the conductivity between any two nodes can be computed from only four elements of the matrix $\boldsymbol{G}^{-1}$, as the Equation (14) shows.

We have left the diagonal elements here undefined. Consequently applying the method above would lead to infinite values, because the denominator in (14) is zero for $p = q$. In practical applications, it is a good choice to set them to $\max_{p,q} \boldsymbol{C}(p,q)$. The matrix $\boldsymbol{C}$ then resembles a block-diagonal matrix not only for data forming compact clusters, but also for data which clusters are best described by high connectivity. We may thus replace the affinity matrix $\boldsymbol{A}$ used for spectral analysis by the conductivity matrix $\boldsymbol{C}$.

The reader may have noted that now the role of the kernel radius σ governing the single affinities fundamentally changes. Previously for the affinity matrix σ should be set in order to give high intra-cluster affinities and simultaneously low inter-cluster dependencies. Now for the conductivity matrix the value of σ can and must be much lower: Even relatively moderate inter-cluster affinities lead to high conductivities and thus seriously affect the clustering. On the other hand, low intra-cluster affinities are no problem since the conductivity "boosts" the matrix. The next section will treat the problem of identifying σ in detail.

4 Choice of σ

Clearly the Gaussian kernel is actually the most common kernel function. Its only parameter is the kernel radius σ. The performance of spectral clustering heavily depends on this σ, as one can easily see and immediately check with simple simulations. So determining σ is an important issue. Even by hand this is a nontrivial task. A common method is to try different σ and use the best. This can be also automated as suggested by Ng et al. (10), since concentration of the spectral images may be an indicator for the quality of the choice of σ. Here we propose two new selection methods, one manual and one automatic. The first one relies on the distance histogram and helps finding a good global σ, the second sets σ automatically to an *individually different* value for each point, thus resulting in an asymmetric

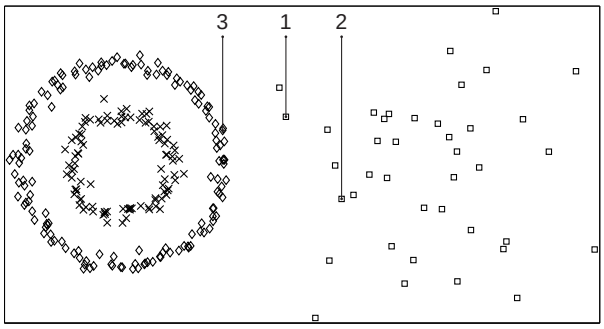


Figure 5: Two rings with low dispersion and a Gaussian cluster with large dispersion

affinity matrix. This method was originally motivated by inhomogeneously dispersed clusters, but it provides also a very robust way for selecting σ in homogeneous cases.

4.1 The Distance Histogram

In a common sense, a cluster is a set of points sharing some higher level of proximity. So if the data form clusters, the histogram of their distances is multi-modal, the first mode corresponding to the average intra-cluster distance and others to between-cluster distances see e.g. Figures 1 and 2). By choosing σ around the first mode, the affinity values of points forming a cluster can be expected to be significantly larger than others. Consequently, if everything goes well, the affinity matrix resembles a block-diagonal matrix or its permutation. This works particularly well if the clusters are spherical.

If our goal is computing a conductivity matrix $\boldsymbol{C}$, we must choose a smaller σ as already observed above. Our experiments have shown that the best choice for σ lies at about the half of the value one would take for analyzing $\boldsymbol{A}$, i.e. about a half of the position of the first peak in the distance histogram, or somewhat below. Otherwise, the algorithm tends to create an over-connected network, thus merging the clusters.

Although this recipe gives good recommendations in many cases, we point out that it is quite tough to automate it. Finding the first peak of the histogram requires an approximation of the histogram, since the histogram can be noisy or highly multi-modal. But even finding the first peak of this approximation is highly error-prone, since there may be artificial peaks caused by overfitting.

4.2 Context-Dependent Similarity

For some data sets, one σ equally good for all points might not even exist. Consider the two rings and a Gaussian cluster in Figure 5. The points in the third, Gaussian cluster have a significantly larger dispersion than the points in the two rings. So each of the points in the third cluster will be connected to almost no other point than itself if σ is chosen correctly for the rings. However, a human observer is likely to judge that point 1 and 3 are much less similar than point 1 and 2, although they are closer! This is because a (s)he takes into account not only the pure distance, but also the *context* of the points, i.e. the neighborhood. Thus point 3, lying in a densely populated region, has a low affinity to point 1. Conversely, point 1 is located in a sparsely populated region and therefore has a high affinity to point 3. In other words, a human would consider a context dependent and thus *asymmetric* similarity relation. Clearly, the distance $d(\cdot, \cdot)$ is still symmetric.

This motivates following way to proceed. First, we will choose a *different* σ_i for each point x_i. This results in an asymmetric similarity matrix, which is denoted by

$$\boldsymbol{A} = \Big(A_{i,j}\Big)_{i,j=1}^{N} = \left(\exp\left(-\frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{2\sigma_i^2}\right)\right)_{i,j=1}^{N}. \tag{15}$$

In order to determine each σ_i, we establish the following condition:

$$\sum_{j=1}^{n} \exp\left(-\frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{2\sigma_i^2}\right) = \tau \tag{16}$$

for each $1 \leq i \leq n$, where $\tau > 0$ is a constant. That is, we choose σ_i such that the row sum of $\boldsymbol{A}$ assumes the fixed value τ. If we think of the binary case where close points have similarity 1 and distant points have similarity 0, this procedure asserts that each point has a constant number of neighbors. If neighborhood is understood in a fuzzy sense, then this interpretation works for other similarity measures as well, e.g. the Gaussian kernel. Thus we call τ the *neighborhood size*.

Clearly, we have only shifted the problem of choosing σ_i, since we now have to choose τ. But there are three advantages with this method: First, there is only *one* parameter τ to fix, which provides values for *all* σ_i. Thus the benefits of asymmetric similarity are exploited, which are considerable as we shall see below, while things are kept simple. Second, the parameter τ is scaling-independent and the neighborhood size interpretation is more intuitive for a human expert who has to tune the algorithm. And third, simulations show that the algorithm is not sensitive to the choice of τ. In fact, all but one of the below comparisons have been carried out with $\tau = 1 + 2d$, where d is the dimension of the data. This seems to be a somewhat sensible choice: For each dimension there should be two neighbors, plus one because the point is neighbor of itself. Another good choice for the neighborhood size which mostly yields the same results in the simulations below is the constant (!) choice $\tau = 10$.

Determining σ_i in order to fulfil (16) is easy, e.g. an iterative bisection does the job reliably. Few iterations are sufficient, since there is no great precision required. After this, we have come up with an asymmetric matrix $\boldsymbol{A}$. In order to allow an efficient spectral analysis and in particular the conductivity approach discussed in the next section, it would be desirable to have a symmetric matrix. A simple observation helps here: If point $\boldsymbol{x}_i$ has affinity $A_{i,j}$ to $\boldsymbol{x}_j$ and $\boldsymbol{x}_j$ has affinity $A_{j,i}$ to $\boldsymbol{x}_i$, then the symmetric affinity should be the *smaller* of these values. This can be motivated by the example in fig. 5. Point 1 lies in a relatively sparse area, thus having a large neighborhood size. In this relation, point 3 is close to point 1, leading to a high affinity $A_{1,3}$. On the other hand, point 3 has a small neighborhood size, since it lies on a relatively densely populated ring. In terms of this neighborhood, point 1 is far from point 3, resulting in a low affinity $A_{3,1}$. This latter value separates the two points and thus determines their overall similarity. So we obtain a symmetric similarity matrix by

$$\boldsymbol{S} = \Big(S_{i,j}\Big)_{i,j=1}^{n} = \Big(\min(A_{i,j}, A_{j,i})\Big)_{i,j=1}^{n}. \tag{17}$$

A very interesting parallel approach to considering a context-dependent neighborhood is given by Rosales and Frey (12). While they suggest and motivate an EM-like repeated update of the neighborhood size, our method is rather ad hoc and computationally cheaper.

5 Algorithm Summary

We summarize both algorithms used for the simulations below. There are two variants for the symmetric one, since conductivity can be used or not. In the latter case, the Laplacian will be used, see section 6.1 below.

Algorithm 3: Symmetric spectral clustering
Parameter: *Preprocessing* $\in \{$*Conductivity*, *Laplacian*$\}$

1. Determine σ using the distance histogram dependent on *Conductivity*.
2. Calculate the affinity matrix $\boldsymbol{A}$.
3. If *Preprocessing* == *Conductivity*, construct the conductivity matrix according to (14).
4. If *Preprocessing* == *Laplacian*, construct the Laplacian according to (18) (see section 6.1).
5. Determine K principal eigenvalues and eigenvectors, e.g. by Lanczo's method.
6. Use these to create K-dimensional spectral images of the original data.
7. Cluster the images by K-lines (Algorithm 2).

Algorithm 4: Asymmetric spectral clustering

1. Determine σ_i for each $1 \leq i \leq N$ according to (16).
2. Calculate the asymmetric affinity matrix $\boldsymbol{A}$ according to (15).
3. Build the symmetric affinity matrix $\boldsymbol{S}$ according to (17).
4. Construct the conductivity matrix according to (14).
5. Determine K principal eigenvalues and eigenvectors, e.g. by Lanczo's method.
6. Use these to create K-dimensional spectral images of the original data.
7. Cluster the images by K-lines (Algorithm 2).

6 Discussion

Before presenting and comparing experimental results, we shall review and discuss some of the many existing variants and methods for spectral clustering. In particular we will try to motivate why we have used the so far presented ingredients and no other. In 6.6 and 6.7 we briefly sketch two approaches which to our knowledge are new and have not been reported so far, but unfortunately they are not very successful. Some of the points discussed below will be revisited in section 7 from the viewpoint of experimental results.

6.1 The Laplacian

Instead of performing a spectral analysis of the affinity matrix $\boldsymbol{A}$, most authors favor the Laplacian of the affinity matrix

$$\boldsymbol{L} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} \tag{18}$$

(the standard Laplacian known to graph theorists is $\mathbf{1} - \boldsymbol{L}$, but this behaves identically for clustering). Here $\boldsymbol{D}$ is a diagonal matrix with the elements $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$. In particular, the algorithm proposed by Ng et al. makes use of the Laplacian and is thus very similar to Algorithm 3 with *Preprocessing* == *Laplace*. There are theoretical results in graph partitioning that depend on the use of the Laplacian ((17)), and experimental results in clustering are encouraging. The Laplacian seems to behave like a method to emphasize the block structure of the matrix, and can be seen as an alternative to our conductivity approach. However, we do

not recommend combining the two: when the affinity matrix is boosted with the conductivity method, simulations suggest that an additional use of the Laplacian is rather questionable.

6.2 K-means

It has become quite common to use the K-means algorithm after having performed the spectral analysis. The algorithm by Ng et al. does so and applies it to the eigenvector components projected onto the unit ball. This projection step is indeed sensible since, as pointed out earlier, the points are distributed along straight lines. Nevertheless the projection discards information, and in fact K-lines yields comparable or even better empirical results (see section 7). Applying K-means directly to the eigenvector components yields a significantly worse performance.

6.3 Feature space

The K-means algorithm can also be used in a feature space induced by the affinity matrix, if this matrix is a kernel matrix. A sophisticated variant of this approach is pursued by Girolami: he considers an expectation-maximization (EM) algorithm with simulated annealing in a high-dimensional feature space. No explicit description of the feature space is necessary, since all calculations can be performed using the kernel matrix. The clustering problem reduces to the maximization of

$$\sum_{j=1}^{k} \frac{1}{n_k} \boldsymbol{z}_j^\top \boldsymbol{A} \boldsymbol{z}_j, \tag{19}$$

where $\boldsymbol{z}_j$ is the binary indicator vector of cluster j. It is interesting to note that the resulting algorithm is independent of its feature space interpretation. If $\boldsymbol{A}$ is an arbitrary affinity matrix, maximizing the above expression means maximizing the in-cluster similarities and is thus still sensible. Our experiments, however, show that this algorithm has difficulties with complex-shaped clusters. This might be due to the high dimension of the feature space and a resulting strongly multi-modal clustering problem.

6.4 Kernel PCA

The first steps of the proposed clustering algorithm can be considered as a kernel PCA, i.e. a PCA in a feature space (see 13). Since the principal eigenvectors can be interpreted as projections of the points in feature space onto the principal components, the spectral analysis reduces the dimension. In fact, it is also possible to apply the K-lines algorithm to the points formed by *all* eigenvectors of $\boldsymbol{A}$, i.e. to the feature space image of the points. This leads to almost the same clustering quality but considerably increases computational costs (see discussion in subsection 2.2).

6.5 Centering

Another question to be posed under this viewpoint is the question of centering the data in the feature space, so that the feature space images sum to zero. This is easily performed by manipulating the kernel matrix only (13). The manipulated, "centered" matrix has a rank one lower than the original kernel matrix. This implies that already the first eigenvector separates two clusters, and in general one eigenvector less has to be considered. Thus the dimension of the final task is reduced by one, and one could expect that centering the kernel matrix should simplify and therefore improve the clustering. However, simulations are against this hope. In fact, the K-lines algorithm becomes slightly more complicated after this modification, because

now the direction of the lines must be considered, too. It seems that actually by centering $\boldsymbol{A}$ some information is lost. Also this might be an issue that deserves further analysis. In contrast to K-lines, the K-means algorithm in feature space is not at all affected by centering, which is not surprising and easily proven.

6.6 Linear separation

Simulations refute also another straightforward idea that is derived from support vector machines. Considering again the affinity values as dot products in a feature space, one could try to find an optimal linear separation in this feature space. To this end, a separation criterion has to be designed. Discussing this topic in depth is beyond the scope of this work, however our experiments in this direction have not been satisfactory at all. Another problem with linear separation arises when dealing with more than two clusters. In contrast, other approaches using separation in the feature space do work, see e.g. Ben-Hur et al. (2).

6.7 Sequential separation

A less sophisticated separation approach sometimes may work quite reasonably. We start with a centered kernel matrix and calculate its eigenvalues and eigenvectors. The principal eigenvector divides the data into two clusters. Then the next eigenvector can be used to establish a third cluster by removing those points from the previous clusters which have significant components in the actual eigenvector. Of course, the corresponding eigenvalue should also be used in order to determine the significance. This step is repeated until the desired number of clusters has been constructed. The sequential algorithm is clearly outperformed by many other spectral clustering algorithms, including our suggested algorithm as well as the algorithm proposed by Ng et al.

7 Simulations

Beside toy examples presented so far, we have tested our algorithms on a number of hard artificial data sets and on three standard benchmark data sets containing real-world data. We have compared the performance of our proposed algorithms to three other clustering algorithms.

Our algorithm using a symmetric similarity function and the conductivity matrix will be referred to as "Conductivity". It is precisely described in Algorithm 3 where *Preprocessing == Conductivity*. The variant having *Preprocessing == Laplace* is called "Laplace". The algorithm stated in (10) is very similar, but it skips the conductivity matrix and uses K-means after a projection of the eigenvector components to the unit ball. It will be referred to as "Ng & al.". The algorithm with a context-dependent σ, Algorithm 4, is abbreviated by "Asymmetric". The method suggested by Girolami (6) uses neither any reinforcement of the matrix block structure nor spectral analysis. It directly works with the kernel matrix. In fact, it can be interpreted as a sophisticated variant of K-means, (actually an expectation-maximization method) in the kernel-induced feature space, and is referred here as "KEM". Finally, we included the basic K-means algorithm in our list.

In order to evaluate the robustness of the algorithms, all except "Asymmetric" (which produces deterministic results) were run 100 times on each data set. Except for K-means, which does not depend on the kernel width, a different value of the parameter σ was used in each run. The values were randomly sampled from a $\pm 20\%$ range around the experimentally found best σ for each algorithm. As our measurements show, the best σ can differ significantly for different algorithms. The automatic computation of σ suggested by Ng et al. (10) (briefly

Data set	Abbr.	Dim.	Size	σ_{best}	σ_{hist}	Figure
1 Ring + Gaussian	1RG	2	600	0.78 ... 2.73	1.50 ... 2.00	4
2 3D rings, $\sigma_D = 0.1$	2R3D.1	3	600	0.11 ... 0.34	0.17 ... 0.63	6
2 3D rings, $\sigma_D = 0.2$	2R3D.2	3	600	0.10 ... 0.40	0.17 ... 0.33	7
2 Rings + Gaussian	2RG	2	290	0.26 ... 1.76	0.25 ... 1.10	5
2 S	2S	2	220	0.02 ... 0.35	0.01 ... 0.25	9
4 Gaussians	4G	3	200	0.21 ... 1.08	0.20 ... 0.90	8
5 Gaussians	5G	4	250	0.24 ... 2.31	0.30 ... 1.10	
2 Spirals	2 Spi	2	386	0.08 ... 0.34	0.25 ... 1.10	10
Iris	Iris	4	150	0.30 ... 0.87	0.38 ... 0.75	
Wine	Wine	13	178	0.98 ... 5.09	1.25 ... 3.00	
Breast Cancer	BC	9	683	5.03 ... 12.95	2.50 ... 4.00	

Table 1: The data sets used in experiments. We refer to each set by its abbreviation, listed in the second column. Beside some basic specification of the data, the range of kernel widths used in experiments is given. σ_{best} denotes the best widths found automatically, relying on class labels, and σ_{hist} values found from the distance histogram.

mentioned in section 4) was also tested. It did not yield stronger results and gave misleading indications in some of our experiments. For comparison, we also tested the algorithms using σ's manually selected from the distance histogram. As mentioned in the motivation for the asymmetric approach, recognizing peaks in a noisy histogram is not an unambiguous task, so for many data sets, several kernel widths were used.

All algorithms have been implemented in MATLAB. In the experiments concerning the running time, the implementation of Lanczo's algorithm coming with MATLAB (eigs) was used for the spectral decomposition. However, this algorithm can be unstable for matrices resulting from an unsuitable σ. Such σ's can appear when being randomly sampled, as in our experiments concerning the clustering accuracy. So in these experiments, the whole spectrum was computed, using the more stable, but considerably slower MATLAB "eig" function. The source code of our algorithms and our artificial test data sets are available at `http://www.cs.huji.ac.il/~fischer/papers/spectClust03.html`.

Our test data sets are outlined in Table 1. There, we also state the size and the dimension of each data set as well as the ranges of σ. The entry σ_{best} refers to the best kernel width found in the experiments. Since it varies for different algorithms, only the range is given. Similarly, due to histogram ambiguities, several different values can be guessed from observing the it, so the range of the best values is given in the column σ_{hist}.

Our main results are summarized in Table 2. Here, we show the *best* results for each algorithm and each data set, respectively. If these results are competitive (i.e. comparable to the best) and additionally *robust* in terms of mean value and variance, they are printed in **bold**. The careful reader should look up mean value and variance in Table 3 and give his own judgement of robustness. Entries which differ much from the mean value, or have a large variance, are printed in *italic*, thus indicating a lack of robustness.

The data set from Figure 4 was already presented in section 3. It consists of a spherical cluster of 100 normally distributed points, encircled by a cluster of 500 points distributed along a ring. It is impossible to cluster this set using K-means, because clusters are not

	K-means	KEM	Ng & al.	Conduct.	Laplace	Asymm.
1RG	265	*0*	*0*	**2**	**1**	**2**
2R3D.1	197	*0*	*0*	**0**	**0**	**0**
2R3D.2	195	*68*	4	**6**	*4*	93
2RG	110	*66*	101	*2*	101	**0**
2S	26	*25*	0	97	**0**	**0**
4G	*24*	**2**	18	**2**	19	**1**
5G	*41*	13	33	*13*	40	**11**
2 Spi	191	151	*0*	*0*	*0*	193[1]
Iris	*16*	**8**	**14**	**10**	**14**	**7**
Wine	*5*	**3**	**3**	12	**3**	65[2]
BC	**26**	**20**	**22**	**21**	**22**	**20**

Table 2: Empirical comparison of the algorithms: Minimum number of incorrectly clustered points for the data sets from Table 1. Values that are good and at the same time robust are marked **bold**, and very volatile values *italic*. A value is considered robust if it does not differ much from the mean over all 100 runs and if the standard deviation is not too large. Careful readers may compare Table 3 and obtain their own judgement of robustness.

	K-means	KEM	Ng & al.	Conduct.	Laplace
1RG	275.3 ± 9.4	24.4 ± 43.2	61.1 ± 93.8	2.0 ± 0.0	1.2 ± 0.4
2R3D.1	197.9 ± 1.0	162.2 ± 86.4	17.7 ± 70.4	0.0 ± 0.0	0.0 ± 0.0
2R3D.2	195.1 ± 0.2	202.9 ± 54.6	7.5 ± 28.9	8.2 ± 1.9	51.1 ± 106.5
2RG	125.3 ± 2.4	121.7 ± 26.5	102.2 ± 3.8	16.8 ± 38.0	101.5 ± 0.5
2S	34.5 ± 6.4	49.9 ± 26.6	9.3 ± 10.8	101.8 ± 4.8	5.5 ± 10.1
4G	40.0 ± 24.2	2.6 ± 5.8	35.1 ± 22.2	3.0 ± 7.0	26.3 ± 18.5
5G	62.0 ± 27.8	21.2 ± 14.1	40.1 ± 14.6	34.5 ± 14.6	45.3 ± 10.9
2 Spi	192.3 ± 0.5	178.5 ± 10.7	79.1 ± 95.4	39.4 ± 55.5	0.0 ± 0.0
Iris	27.8 ± 22.2	10.3 ± 2.2	14.8 ± 3.6	12.3 ± 5.3	15.6 ± 3.4
Wine	8.7 ± 10.5	3.9 ± 0.3	3.3 ± 0.5	18.9 ± 6.6	3.3 ± 0.5
BC	26.5 ± 0.5	21.5 ± 1.0	22.8 ± 0.6	25.1 ± 2.0	22.7 ± 0.8

Table 3: Empirical comparison of the algorithms: average number and the standard deviation of incorrectly clustered points in 100 runs. Except for K-means, a different σ, sampled ±20% around the best one, was used in each run.

linearly separable and their centers coincide. But for the other algorithms we tested, this data set proved to be rather simple, reaching stably as low as two wrong classifications.

For the two interlacing rings in 3D with data dispersion of 0.1 (Figure 6), both KEM and spectral approaches work well, and K-means fails, as expected. Increasing the dispersion of the points to 0.2 (Figure 7) leads to a somewhat surprising result: although most spectral algorithms still perform well, the asymmetric one does not. The ring segments passing through the center of the other ring are incorrectly assigned to that ring. Our interpretation is that the context-sensitive kernel width acts as a drawback in this situation, since it actually abolishes the already weak gap between the clusters.

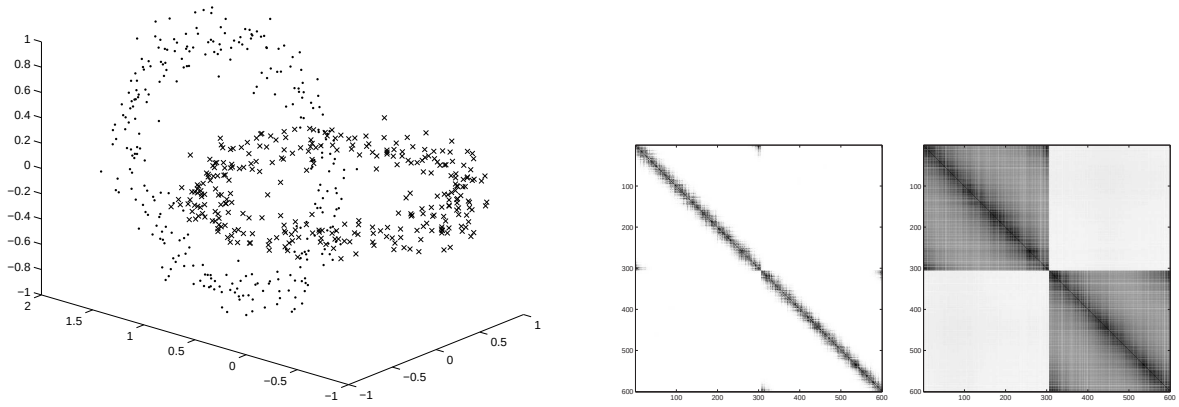


Figure 6: Two intersecting ring clusters with data dispersion $\sigma_D = 0.1$. **Left:** scatter plot of the data. **Middle:** Affinity matrix $\boldsymbol{A}$ computed with the Gaussian kernel. **Right:** Conductivity matrix $\boldsymbol{C}$.

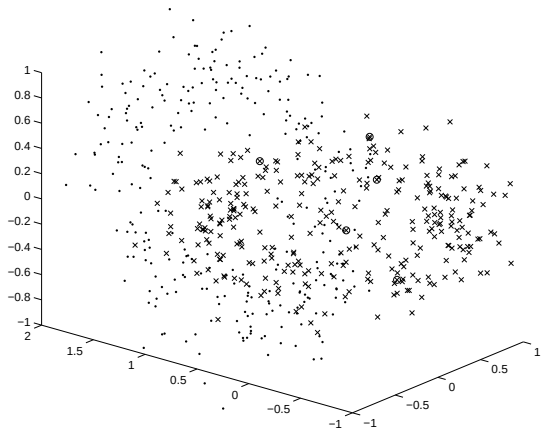


Figure 7: Scatter plot of the intersecting ring clusters with dispersion $\sigma_D = 0.2$. Circles mark incorrectly clustered points.

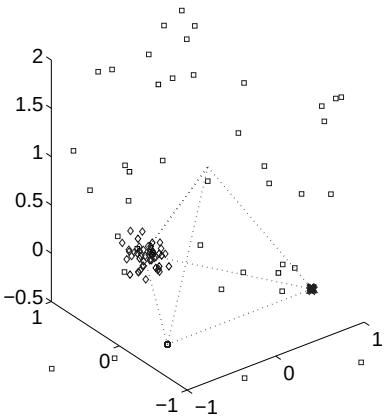


Figure 8: Four Gaussian clusters with different dispersion, centered at the vertices of a regular simplex

The above example was simple, in the sense that the intra-cluster dispersion was approximately the same for both clusters. The advantage of the asymmetric algorithm becomes apparent for the sets shown in Figures 5 (abbreviated in the tables as 2RG) and 9 (2S). Due to the different dispersions of clusters and, in the latter case, within clusters, no obvious σ can be chosen. For the latter data set, visualizations of the respective conductivity matrices based on the asymmetric and the symmetric method are also given in Figure 9. It can be seen that the asymmetric approach results in much clearer block structure than the symmetric one. The results also speak in favor of the asymmetric clustering. For the 2RG set, only the conductivity approach sometimes reaches a comparably good result with only two false assignments, but on average 16 points are falsely classified. For the 2S data set, Laplacian-based algorithms reach occasionally the performance of the asymmetric one, but on average perform notably worse.

Two more data sets on which the asymmetric algorithm performs best are the sets of

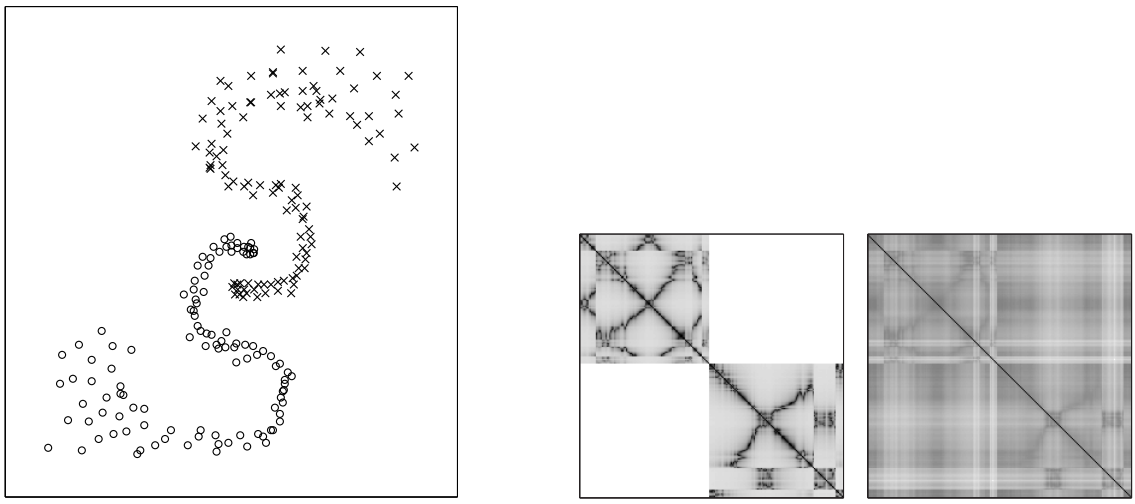


Figure 9: Two S-shaped clusters with varying dispersion. **Left:** The scatter plot of the data. **Middle:** The conductivity matrix based on the asymmetric approach has clear block structure. **Right:** With a symmetric approach, the block structure is less definite.

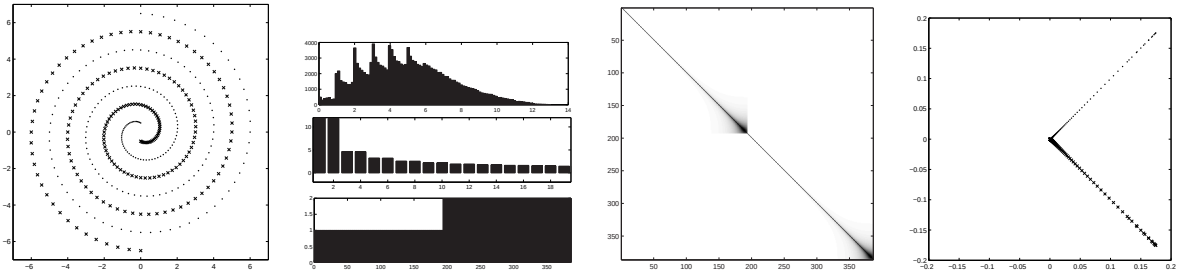


Figure 10: A successful clustering of two spirals. **Far left:** Scatter plot of the data. **Center left:** distance histogram (top); 20 largest eigenvalues (middle); cluster membership chart (bottom). **Center right:** Conductivity matrix $\boldsymbol{C}$. **Far right:** Spectral plot along the top two eigenvectors.

Gaussian clusters with different standard deviations. In the set denoted by 4G, the clusters lie in a 3D space, and the standard deviations within them range from 0.001 to 1 (see Figure 8). The 5G set is obtained by adding one dimension and one cluster of a high dispersion. Although the clusters are Gaussian, K-means performs weakly. Interestingly, the KEM algorithm can lead to reasonable results, at least for the 4G set. Symmetric spectral algorithms all have problems, with conductivity-based being somewhat better than the other two.

We have also tested the algorithm on a variant of Wieland's two spirals (Figure 10). This artificial data set is used as a standard benchmark in supervised learning (see 4). In Wieland's original set, each spiral consists of 97 points and coils three times around the origin. Near the center, points are relatively dense and at the outer ends of the spirals, points from the same spiral are further away than points from different ones — for clustering, an extremely unpleasant fact. We used spirals with double point density, resulting in 193 points per spiral. The set is still very inconvenient and even our conductivity matrix is far from being block-diagonal. Contrary to our expectations this is one of the examples where the asymmetric

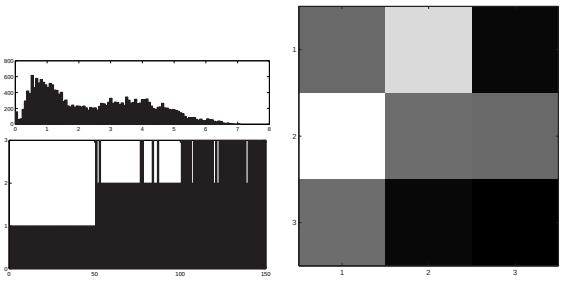


Figure 11: Clustering of the Iris data set. **Left column:** distance histogram (top); cluster membership chart (bottom). **Right column:** Matrix of prototypical vectors $\boldsymbol{M}$ (black $\equiv 1$, white $\equiv -1$).

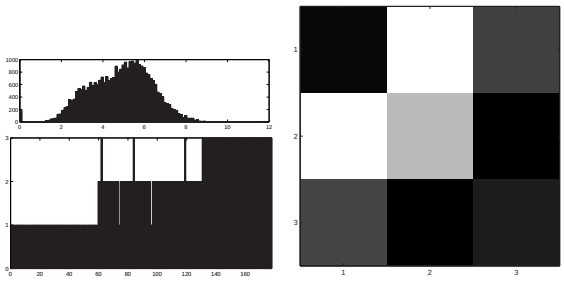


Figure 12: Clustering of the Wine data set. **Left column:** distance histogram (top); cluster membership chart (bottom). **Right column:** Matrix of prototypical vectors $\boldsymbol{M}$.

algorithm fails[1]. Nevertheless, with an appropriate σ, other spectral algorithms can achieve the correct clustering for all points, with the algorithm using Laplacian being also stable.

The last three data sets are common benchmarks with real-world data (see e.g. (9)): the Iris, the Wine and the Breast Cancer data set.

The Iris data set (5) contains 150 measurements, each of four physical dimensions for three sorts of iris flowers. Each of the sorts – setosa, versicolor and virginica – is represented with 50 measurements. The latter two classes are not linearly separable and overlap. The overlap is clearly visible in the affinity matrix, where the two last blocks form a larger block. The depicted affinity matrix was computed using Gaussian kernel with $\sigma = 0.75$, as the distance histogram suggests. This is a simple data set, and even the most basic algorithm, without applying the Laplacian or conductivity measure, reaches only 11 misclassifications (Figure 11). In the prototype vectors' matrix $\boldsymbol{M}$, graphically represented in Figure 11 right, we see that the second and the third entry in the second row are both large, suggesting that the last two clusters form a super-cluster. The conductivity matrix algorithm performs slightly better then the Laplacian-based ones, but the asymmetric outperforms them all, achieving only 7 misclassifications. The KEM algorithm, on average, does not perform much better than symmetric spectral ones, and not as well as the asymmetric one. It has been reported to perform much better, but we have not been able to reproduce that results.

The Wine data set contains 178 measurements of 13 different variables concerning wine grapes. This seems to be an easy set, where even K-means reaches quite acceptable accuracy. Curiously, our algorithms perform weaker: the conductivity-based one reaches not less than 12 misclassifications, compared to three, as obtained by KEM or Laplacian-based methods. The asymmetric algorithm performs even weaker, with 65 false assignments[2]. We attribute this anomaly to the curse of dimensionality: 13 dimensions for only 178 points.

On the Breast Cancer data set, all algorithms perform similarly, although the asymmetric still leads, with 20 misclassifications.

Table 4, comparing the running times of the algorithms, reveals further interesting details. Since the bulk of computation time of the first three algorithms is spent for computing the spectral decomposition, one could expect that the timings of these algorithms are almost

[1]The asymmetric algorithm performs perfectly on the fourfold-density problem, with 385 points per spiral.
[2]With conductivity switched off, the asymmetric approach leads to only four false classifications.

	K-means	KEM	Ng & al.	Conduct.	Laplace	Asymm.
1RG	0.04	1.54	2.24 + 2.43	2.10 + 0.32	2.24 + 3.22	2.14 + 0.54
2R3D.1	0.03	1.32	2.24 + 1.49	2.12 + 0.31	2.24 + 5.14	2.12 + 0.31
2R3D.2	0.04	1.02	2.23 + 6.18	2.11 + 0.54	2.24 + 6.39	2.09 + 0.54
2RG	0.02	0.26	0.27 + 1.35	0.34 + 0.18	0.27 + 0.63	0.34 + 0.18
2S	0.01	0.26	0.13 + 2.16	0.16 + 0.07	0.12 + 1.89	0.16 + 0.11
4G	0.02	0.12	0.09 + 0.29	0.12 + 0.14	0.09 + 0.13	0.12 + 0.09
5G	0.07	0.56	0.18 + 0.09	0.33 + 0.42	0.18 + 0.14	0.22 + 0.17
2 Spi	0.03	0.43	—	0.76 + 0.18	—	0.74 + 0.27
Iris	0.02	0.20	0.04 + 0.10	0.06 + 0.07	0.04 + 0.10	0.06 + 0.05
Wine	0.02	0.16	0.06 + 0.07	0.09 + 0.08	0.06 + 0.08	0.09 + 0.08
BC	0.03	1.66	3.34 + 0.42	3.38 + 0.42	3.35 + 0.42	3.39 + 0.42

Table 4: Running times in seconds of the algorithms on a 600 MHz Pentium III with 256 MB RAM. For the spectral algorithms, the time for block boosting (Laplacian or Conductivity) and the time for eigenvalue decomposition are given separately. On the 2 Spi data set, Lanczo's algorithm failed to converge for Laplacian-based methods, therefore no comparable timings are available.

identical. This is disproved by the observations. The reason is that the performance of Lanczo's method for calculating the principal eigenvectors highly depends on the structure of the matrix. It turns out that the matrix constructed by one of our methods is very favorable for the spectral decomposition, probably due to its relatively clear block structure. For Laplacian-based methods, the spectral decomposition sometimes even fails to be stable, if an iterative method (Lanczo's algorithm) is used. Actually the block-enhancing (conductivity and asymmetric) algorithms are almost always faster than the Laplacian-based variants. In some cases, they are even almost as fast as the KEM algorithm, which needs no eigenvector calculation at all. The pure K-means algorithm is, of course, faster.

Table 5 evaluates the quality of the choice of σ with the proposed histogram method. One can see that in most cases the σ chosen from the histogram is close to the best value, and the same is true even for the number of clustering errors. As already mentioned, there is a big deal of ambiguity in the histogram choice in many cases.

We shall finally discuss the question of how our algorithm scales up. Increasing the data dimensionality is not critical at all, as long as the pairwise distances are efficiently computed. After this initial step, the algorithm is completely independent of the dimensionality. Raising the number of data points N is more critical, since the algorithm relies on spectral decomposition and matrix inversion (for the conductivity matrix), resulting in a computational complexity of $O(N^3)$. Practically, the limiting factor is not the computational power but the memory available. The space complexity of the algorithm is $O(N^2)$. On our test system used for the above experiments (256 MB RAM), the algorithm works up to about 3000 points without notable slow-down due to low memory.

8 Conclusion

Three new methods for spectral clustering have been developed in this paper: The K-lines algorithm for performing the final clustering step, the conductivity algorithm for reinforcing

		KEM		Ng & al.		Conduct.		Laplace	
		σ	*#err*	σ	*#err*	σ	*#err*	σ	*#err*
1RG	best	2.29	0	1.79	0	1.00	2	1.44	1
	hist	2.00	1	1.50	1	2.00	2	1.50	1
2R3D.1	best	0.32	0	0.25	0	0.13	0	0.11	0
	hist	0.63	167	0.17	0	0.17	0	0.17	0
2R3D.2	best	0.33	68	0.12	4	0.14	6	0.12	4
	hist	0.33	125	0.17	15	0.17	11	0.17	14
2RG	best	0.32	66	0.55	101	1.27	2	0.62	101
	hist	0.25	94	0.55	101	1.10	2	0.55	101
2S	best	0.18	25	0.03	0	0.05	97	0.03	0
	hist	0.25	29	0.01	0	0.01	1	0.03	24
4G	best	0.29	2	0.59	18	0.30	2	0.79	19
	hist	0.20	2	0.60	18	0.20	4	0.90	19
5G	best	0.59	13	2.23	33	0.30	13	1.95	40
	hist	0.80	16	1.10	77	0.30	13	1.10	94
2 Spi	best	0.25	151	0.17	0	0.21	0	0.19	0
	hist	0.55	184	0.50	190	0.25	146	1.10	192
Iris	best	0.65	8	0.42	14	0.38	10	0.45	14
	hist	0.55	10	0.38	14	0.38	10	0.38	14
Wine	best	2.79	3	3.18	3	1.31	12	2.50	3
	hist	2.50	4	3.00	3	1.25	12	2.50	3
BC	best	6.17	20	8.73	22	8.09	21	11.91	22
	hist	4.00	31	4.00	24	4.00	36	3.20	23

Table 5: Comparison of kernel widths derived automatically and manually from the distances histogram. Even with noisy histograms, and varying data dispersion inside clusters, where several peaks can be observed, the histogram method leads to quite acceptable results. The kernel width σ and the number of misclassifications are in most cases similar to those obtained by the best σ found by checking of a large range of values and using the known class labels.

the block structure of the affinity matrix and the context-dependent similarity which allows a robust and automatic choice of σ even in the case of inhomogeneous clusters. Moreover, some analysis has been presented, in particular concerning the block structure of the affinity matrix, the behavior of the eigenvectors, and the clustering hierarchy.

The informal analysis as well as the simplicity and elegance of the methods proposed hopefully motivate their theoretical interest. However, a formal analysis remains for future research. Here, like for clustering in general, it is not even trivial to define a framework in which assertions can be proven, since the clustering task itself is not well-defined in general. The construction of an appropriate – probably flexible – framework could also contribute to the understanding of complex clustering methods, in particular spectral ones.

On the other hand, simulations show that our methods are very promising for practical application. Particularly when they are combined, they result in a very strong clustering algorithm with high robustness in many situations. We therefore hope that the methods will contribute to the usefulness and utilization of spectral clustering.

Acknowledgements

The authors would like to thank Yair Weiss and Andreas Zell for helpful suggestions and discussions.

References

[1] C. Alpert, A. Kahng, and S. Yao. Spectral partitioning: The more eigenvectors, the better. Technical report, October 1994. UCLA CS Dept. Technical Report #940036.

[2] A. Ben-Hur, H.T. Siegelmann D. Horn, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[3] P. S. Bradley and O. L. Mangasarian. k-plane clustering. *Journal of Global Optimization*, 16(2):23–32, 2000.

[4] S.E. Fahlman. Faster-learning variations on back-propagation: An empirical study. In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 38–51, San Mateo, CA, USA, 1988. Morgan Kaufmann.

[5] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7, Part II:179–188, 1936.

[6] M. Girolami. Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, May 2002.

[7] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. In *Proceedings of the 41st Symposium on Foundations of Computer Science*, 2000.

[8] M. Meilă and J. Shi. Learning segmentation by random walks. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 873–879. MIT Press, 2001.

[9] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases, 1994. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[10] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[11] P. Perona and W. Freeman. A factorization approach to grouping. *Lecture Notes in Computer Science*, 1406:655–670, 1998. URL `http://citeseer.nj.nec.com/-perona98factorization.html`.

[12] R. Rosales and B. Frey. Learning generative models of affinity matrices. In *19th Conference on Uncertainty in Artificial Intelligence (UAI)*, 8 2003.

[13] B. Schölkopf. *Support Vector Learning. PhD Thesis.* Oldenbourg Verlag, Munich, Germany, 1997.

[14] G.L. Scott and H.C. Longuet-Higgins. Feature grouping by relocalsation of eigenvectors of the proximity matrix. In *British Machine Vision Conference*, pages 103–108, 1990.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. URL `http://-citeseer.nj.nec.com/shi97normalized.html`.

[16] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82:93–133, 1989.

[17] D. A. Spielman and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *IEEE Symposium on Foundations of Computer Science*, pages 96–105, 1996.

[18] D. Verma and M. Meilă. A comparison of spectral clustering algorithms. Technical report, 2003. UW CSE Technical report 03-05-01.

[19] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *ICCV (2)*, pages 975–982, 1999. URL `http://citeseer.nj.nec.com/weiss99segmentation.html`.