Spectral Methods for Automatic Multiscale Data Clustering

Arik Azran
Gatsby Computational Neuroscience Unit
University College London
London WC1N 3AR, UK
arik@gatsby.ucl.ac.uk

Zoubin Ghahramani*
Department of Engineering
University of Cambridge
Cambridge CB2 1PZ, UK
zoubin@eng.cam.ac.uk

Abstract

Spectral clustering is a simple yet powerful method for finding structure in data using spectral properties of an associated pairwise similarity matrix. This paper provides new insights into how the method works and uses these to derive new algorithms which given the data alone automatically learn different plausible data partitionings. The main theoretical contribution is a generalization of a key result in the field, the multicut lemma [7]. We use this generalization to derive two algorithms. The first uses the eigenvalues of a given affinity matrix to infer the number of clusters in data, and the second combines learning the affinity matrix with inferring the number of clusters. A hierarchical implementation of the algorithms is also derived. The algorithms are theoretically motivated and demonstrated on nontrivial data sets.

1. Introduction

Clustering is a fundamental unsupervised learning problem, where one needs to find a partitioning for a given set of items $S = \{s_n\}_{n=1}^N$ into K groups $\{S_k\}_{k=1}^K$ such that $\cup_{k=1}^K S_k = S$ and $S_k \cap S_l = \emptyset$ if $k \neq l$. Imagine you are given the task of designing an algorithm to cluster S, without any additional information such as the number of clusters K, bounds on the number of points in the k'th cluster $|S_k|$, the location of the clusters etc. Often there is more than one plausible way to partition the data. Thus we would like an algorithm which suggests different 'good' partitionings and associates each of them with a numerical measure that indicates how 'good' they are. This paper provides such an algorithm, using novel ideas in spectral clustering.

Spectral clustering is a technique for data partitioning based on similarities between pairs of data points. The spectrum of an affinity matrix, a matrix of pairwise similarities between points, is used to cluster the data points into groups. The ease of implementation, combined with the ability to cluster complex data sets, makes the method appealing to researchers in various fields, including bioinformatics [10], speech recognition [4] and software clustering [1]. In computer vision it is used to perform image segmentation [6, 8, 9]. As a nonparametric approach to unsupervised clustering, it often beats parametric models for clustering in machine learning (e.g. mixture models).

Many studies have been done on spectral clustering with relation to random walks [8], graph cuts and normalized cuts [6], and matrix perturbation theory [2], continuously improving our understanding of the technique. Alongside the development of algorithms [6, 8, 2, 5, 9] significant theoretical progress has also been achieved. In [7] it was shown that if some conditions apply then spectral clustering minimizes the multiway normalized cut, a generalization of the two way normalized cut criterion [6]. Since this result is the basis for some of the work presented here, it will be discussed more in the sequel.

Although much has been discovered various key issues remain open questions, e.g. (i) How can one choose a good function to measure the pairwise similarity when *all* that is available is S?, (ii) How can the number of clusters K be learned from S?, (iii) Why do spectral methods use the leading eigenvectors and (iv) When is spectral clustering expected to work?. These are the questions addressed in this paper, leading to the following contributions; (1) Analyzing the effect of taking multiple steps of the random walk, a direct generalization of the multicut lemma [7], which we briefly discuss in section 2, is derived in section 3. (2) A new algorithm that finds different plausible values for K is derived from theoretical considerations (section 4). (3) This algorithm is, in turn, generalized in section 5 to learn the parameter of the similarity function, and (4) a hierarchical implementation which is both efficient and intuitive is described in section 6. This is all done while keeping the additional complexity and computation burden minimal.

*Also at the Machine Learning Department, Carnegie Mellon University, Pittsburgh, USA

2. Overview of spectral clustering

This section gives a short review of spectral clustering and the main results from the literature used in our work. Consider the case where members of S are points in the data space $\mathbb{R}^t$. The indices of the points in the k'th group are denoted by $\mathcal{I}_k$ and the partition by $\mathcal{I} = \{\mathcal{I}_k\}_{k=1}^K$. Given a metric $d(x, y)$, defined over $\mathbb{R}^t$, the similarity between different points in S can be measured by any monotonically decreasing parameterized function $w_{mn}(\sigma) = w\,(d(s_m, s_n); \sigma)$, with σ being the length scale of w. The smaller it is, the smaller is the 'neighborhood' of a point. For multidimensional data, different length scales can be used along different dimensions. While the results in this paper can be applied to any metric space $(\mathcal{X}, d(x, y))$ with any function w, for brevity we only discuss $\left(\mathbb{R}^t, \|x - y\|^2\right)$ with the popular Gaussian function

$$w_{mn}(\sigma) = \exp\left(-\frac{\|s_m - s_n\|^2}{\sigma^2}\right) . \tag{1}$$

The pairwise similarities are conveniently summarized in the *Gram matrix*, also known as the *kernel matrix*.

DEFINITION 1 (Parameterized Gram Matrix) *Given a set $S = \{s_n\}_{n=1}^N$ and a parameterized function $w(\sigma) : \mathbb{R}^t \times \mathbb{R}^t \mapsto \mathbb{R}_+$, the parameterized $N \times N$ matrix $W(\sigma)$ with elements $[W(\sigma)]_{mn} = w_{mn}(\sigma)$ is referred to as the parameterized Gram matrix of $w(\sigma)$ w.r.t. S.*

For simplicity of notation we sometimes drop the explicit dependence on σ, but the reader should keep this dependence in mind.

Different algorithms use W differently to derive an affinity matrix P. In this paper we adopt the random walk view [8] for the definition of P (see [3] for relation to other definitions). Denote the volume of the n'th node $D_{nn} = \sum_{i=1}^N w_{ni}$ and the diagonal matrix $D = \mathrm{diag}(D_{11}, \ldots, D_{NN})$, then the affinity matrix is given by

$$P = D^{-1}W . \tag{2}$$

Notice that each row of P sums to 1, thus P_{mn} can be interpreted as the probability for a random walk that begins at s_m to end up at s_n after a single step. More formally, if we let x_j be the location of the walk at time j, then

$$P_{mn} = \mathbb{P}(x_{j+1} = s_n | x_j = s_m) . \tag{3}$$

2.1. The baseline method

Baseline algorithms assume the Gram matrix W and the number of clusters K are given with the data. First the affinity matrix P is computed, and then its *eigensystem* is used to cluster the data, as described in algorithm 1.

DEFINITION 2 (Eigensystem of a matrix) *Let λ_n and v_n be the n'th eigenvalue and eigenvector of a matrix P, i.e. $Pv_n = \lambda_n v_n$. Without loss of generality, let $\lambda_n \geq \lambda_{n+1}$ and $\|v_n\| = 1$. Then, $\{\lambda_n, v_n\}_{n=1}^N$ is the eigensystem of P.*

```
Input: Data set $S$, number of clusters $K$, Gram matrix $W$.
Output: A partitioning $\{S_k\}_{k=1}^K$.
Algorithm:
1.Compute $P$ according to (2).
2.Find the eigensystem of $P$ and define $V = (v_2, \ldots, v_K)$.
3.Consider the rows of $V$ to be points in $\mathbb{R}^{K-1}$ and cluster them using the $K$-means clustering algorithm.
4.Define $\mathcal{I}_k$ to be the index set of all the rows of $V$ belonging to the $k$'th cluster.
5.Define the partitioning $\{S_k\}_{k=1}^K$, where $S_k = \{x_n\}_{n \in I_k}$.
```

Algorithm 1: *Baseline spectral clustering algorithm.*

Some properties of P which are important for our discussion are summarized in the following Lemma.

LEMMA 1 *Assume W is full rank and P is given by (2). Then,*
1. P is full rank,
2. $\lambda_1 = 1$ and $v_1 = [1, 1, \ldots, 1]^\top / \sqrt{N}$,
3. λ_n is real and $|\lambda_n| \leq 1 \; \forall\, n = 2, 3, \ldots, N$.

The proof is trivial; D is diagonal thus P is defined by normalizing the rows of W and (1) follows, (2) can be verified by direct calculation, the first part of (3) is easily proved using the symmetry of W and the second part by using the equality $\lambda_n v_n(m) = \sum_i P_{mi} v_n(i) \; \forall n, m$, and for each n choosing $m = \mathrm{argmax}_j \, |v_n(j)|$. Property 2 explains why only eigenvectors 2 to K are used to define V; since the first eigenvector is all ones, it contains no grouping information.

2.2. Why should it work?

Spectral clustering was analyzed using tools from matrix perturbation theory in [2]. There, it was shown that if the affinity matrix is close to block diagonal with K blocks, then the K leading eigenvectors efficiently reflect this structure and the method is guaranteed to perform well. However, empirical studies show that the method is successful even in cases where the matrix is far from block diagonal. A different approach is pursued in [8, 7], where the notion of piecewise constant eigenvectors was introduced.

DEFINITION 3 (Piecewise Constant Eigenvectors (PCE)) *Let v be an eigenvector of P and $\mathcal{I} = \{\mathcal{I}_k\}_{k=1}^K$ be a partition of $1, 2, \ldots, N$ into K disjoint sets. Then, v is said to be a Piecewise Constant Eigenvectors of P with respect to $\mathcal{I}$ if $v(i) = v(j) \; \forall\, i, j \in \mathcal{I}_k$ and $k \in 1, 2, \ldots, K$.*

It was shown in [7] that if P's K leading eigenvectors are PCE with respect to $\mathcal{I}$,then spectral clustering minimizes the multiway normalized cut (MNCut)

$$\text{MNCut}\left(\mathcal{I}\right) = K - \sum_{k=1}^{K} \frac{\text{Cut}\left(\mathcal{I}_k, \mathcal{I}_k\right)}{\text{Cut}\left(\mathcal{I}_k, \mathcal{I}\right)} , \quad (4)$$

where $\text{Cut}\left(\mathcal{I}_k, \mathcal{I}_{k'}\right) = \sum_{m\in\mathcal{I}_k, n\in\mathcal{I}_{k'}} W_{mn}$. An intuitive understanding of the MNCut can be gained by expressing it [7] as $\text{MNCut}\left(\mathcal{I}\right) = \sum_{k=1}^{K}\left(1 - \mathbb{P}\left(\mathcal{I}_k \to \mathcal{I}_k | \mathcal{I}_k\right)\right)$, the sum of transition probabilities between different clusters in a *single step*. Thus, the MNCut is minimized by the partitioning for which the a random walk is most probable to stay in the cluster in which it is located.

LEMMA 2 (The Multicut Lemma [7]) *Assume $W_{N\times N}$ is symmetric with nonnegative elements, and define P according to (2). Assume further that P's K leading eigenvectors are PCE with respect to a partition $\mathcal{I}^*$ and their eigenvalues are not zero. Then, $\mathcal{I}^*$ minimizes the MNCut.*

Notice that for PCE, K-means clustering in the feature space (Alg1, step 3) is ideal since it is asked to find clusters whose members are all identical.

3. Structure exploration with random walks

We now examine what happens to the results of section 2 if we let the random walk take many steps instead of only one. The results turns out to be very informative, and they lead to the derivation of a new clustering algorithm that given S and W, automatically learns different plausible values for the number of clusters K, and scores the resulting partitionings.

Consider using the M'th order transition matrix P^M, whose elements are

$$P^M_{mn} = \mathbb{P}(x_M = s_n | x_0 = s_m) , \quad (5)$$

as the affinity matrix. P^M_{mn} gives the total probability that a random walk x_j, beginning at s_m, will end up in s_n after M steps, considering *all possible paths* between the nodes. P^M_{mn} is expected to be high if there is a good path between s_m, s_n and low otherwise, hopefully leading to a block diagonal matrix which is ideal for clustering data [2]. However, often in practice we observe a different behavior of P^M. If points i, j are in the same cluster, then often there are values of M for which P^M_i and P^M_j, the i'th and j'th rows of P^M, becomes very similar. The intuition here is that if points i, j are similar then after sufficient number of steps we can expect that a particle that begins a random walk in each of them will have the same distribution for its location after M steps. Another observation is that by varying the number of steps M we explicitly explore similarities at different scales in the data, and as M increases we expect to find coarser structure.

The following lemma shows how the results of spectral clustering with P are related to those with P^M as the affinity matrix.

LEMMA 3 *Assume W, K are given and let the partitioning $\mathcal{I}$ be the result of algorithm 1. Then, substituting P in step 1 of the algorithm with P^M, for any odd positive integer M, yields the same partitioning $\mathcal{I}$.*

It is well known, from the theory of markov chains, that P^M is given by multiplying P with itself M times, so that if $P = V\Lambda V^{-1}$ then $P^M = V\Lambda^M V^{-1}$, where V is the matrix whose n'th column is v_n. Thus, if $\{\lambda_n, v_n\}$ is the eigensystem of P, then $\{\lambda_n^M, v_n\}$ is the eigensystem of P^M. Next, if M is odd then the ordering of the eigenvalues is left unchanged and the same eigenvectors are picked to cluster the data, hence the lemma is proved. This equivalence between P and P^M reveals two important key ideas (1) spectral clustering implicitly searches for good paths between points, and (2) the eigenvalues can be used to indicate the scale and quality of partitioning, by separating between the eigenvalues that survive M steps and those that don't.

P^M can be analyzed into a sum of N matrices

$$P^M = \sum_{n=1}^{N} \lambda_n^M \frac{v_n v_n^\top D}{v_n^\top D v_n}, \quad (6)$$

each of which depends only on P's eigensystem. This is accomplished by exploiting the fact that $v_n^\top D v_m = \delta_{nm}$, which is due to P being defined by a normalized symmetric matrix (2). To appericiate the implications of this analysis, we first introduce the following two definitions.

DEFINITION 4 (Principal Matrix Component) *We refer to the matrix $T_n = \frac{v_n v_n^\top D}{v_n^\top D v_n}$ as the n'th principal matrix component of P^M, and λ_n^M as its weight.*

DEFINITION 5 (Idempotent-Orthogonal Basis of a Matrix) *Assume A is a square full rank matrix of size N, and let $\delta_{nm} = 1$ if $n = m$ and 0 otherwise. The set of square matrices of size N, $\{U_n\}_{n=1}^N$, is said to be an idempotent-orthogonal matrix basis of A if $U_n U_m = \delta_{nm} U_n$ and there is a set of N numbers $\{\mu_n\}_{n=1}^N$ such that $A = \sum_{n=1}^N \mu_n U_n$.*

An idempotent matrix satisfy $U_n U_n = U_n$, and the orthogonality is due to the condition $U_n U_m = \mathbf{0}$ for $n \neq m$. Although it seems redundant, it is worth mentioning here that the sum of all ranks of U_n with nonzero weights μ_n must equal the rank of A, and if all μ_n are nonzero, then the rank of any matrix U_n is one.

We can now give an intuitive interpretation for P's PMC and eigenvalues, analogous to PCA. In PCA the eigenvectors are pointing in an orthogonal set of directions with maximum variance, and the eigenvalues indicate the variance in these directions. Here, the PMCs are an analysis of P into an idempotent-orthogonal basis $\{T_n\}_{n=1}^N$, with weights λ_n^M. Notice that any T_n is independent of M, but the absolute value of its weight is monotonically decreasing with respect to M (recall that $|\lambda_n| \leq 1$). This motivates interpreting the eigenvalues as indicators to the structure in data revealed by T_n. If λ_n is very close to 1, such that λ_n^M is also close to 1, then T_n 'survives' the random walk and is related to stable groups in the data whereas the λ_n^M that tends towards zero are related to unstable groups. So, PMC can be interpreted as the projection of P (or P^M, since they have the same eigenvectors) into its principal matrix components that reveals structure in multi scales, with λ_n^M as an indication to how stable this component is with respect to M. Instead of variance, we talk here about robustness to the number of steps M in the random walk. The bigger $|\lambda_n^M|$ is, the more robust is T_n to a random walk of M steps, and the better is the structure revealed by it. Next, recall that $T_n = \frac{v_n v_n^\top D}{v_n^\top D v_n}$, so that given D, v_n is unique and there is a one to one correspondence between T_n and v_n. Thus, the higher is the value of λ_n^M, the larger is the scale of the structure revealed by v_n.

To help gain some more intuition, one can examine the case $M \to \infty$. Using lemma 1 we get $P^\infty = 1\frac{v_1 v_1^\top D}{v_1^\top D v_1} = \frac{1}{\sum_{n=1}^N D_{nn}}[\mathbf{1}D_{11}, \mathbf{1}D_{22}, \ldots, \mathbf{1}D_{NN}]$, which is the matrix whose rows are all equal to the stationary distribution of the markov chain with transition matrix P (first left eigenvector). This result recovers a well known property, that for infinite number of steps $\mathbb{P}(x_\infty = s_n | x_0 = s_m) = \frac{D_n}{\sum_{j=1}^N D_j}$, meaning that for infinite number of steps the random walk forgets where it began. If the data has well separated clusters, we can expect similar behavior for finite M too. In this case, a particle that begins its walk in one of the clusters is expected to stay there for a long time, until the distribution in the cluster will resemble the stationary distribution over it, as if other clusters did not exist. This can be formalized by the following Theorem.

THEOREM 1 *Let $\{S_k\}_{k=1}^K$ be a partition of S, and $s_m \in S_k$ for some $k = 1, 2, \ldots, K$. Then, if there is an odd number of steps M such that $\mathbb{P}(x_M \in S_l | x_0 = s_m)$ is equal for all $s_m \in S_k$ and $l \neq k$, then spectral clustering with P as the affinity matrix minimizes the MNCut.*

Using lemma 4 from [7], it can be shown that if the condition of Theorem 1 holds, then P^M has PCE. In this case, using the argument in the proof of lemma 3 shows that P also has PCE, and then lemma 2 completes the proof. This result gives an intuitive explanation to lemma 2 and is closely related to the idea of PCE. Instead of considering a random walk of a single step, it allows us to consider a random walk of any number of steps, while still maintaining the powerful

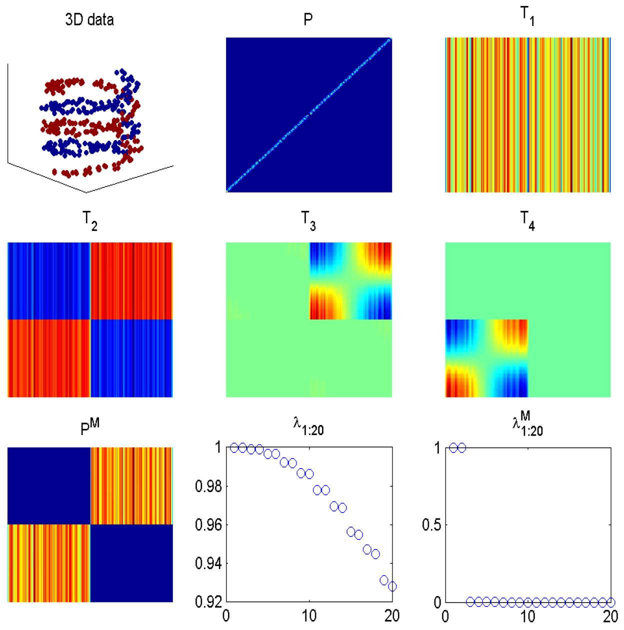

Figure 1. **Example of Principal Matrix Components**. The data set, its transition matrix P, and its leading PMC's (6) and eigenvalues are shown. Notice how T_1 is the matrix whose rows are all equal to the stationary distribution of P. The PMC's form an idempotent-orthogonal basis for P (definition 5). The leading eigenvalues of P (bottom row, middle) and their M'th power (bottom row, right) are shown (notice the different scale in the y-axis) for $M = 8076$, as automatically selected by algorithm 2. Notice how only two eigenvalues 'survive' the random walk such that $P^{8076} \simeq 1 \cdot T_1 + 1 \cdot T_2$ (bottom, left), and how the PMC's are related to hierarchical partitioning of the data.

guarantees of lemma 2. It also suggests a clear and intuitive explanation to why spectral clustering is so successful even if the data sets are extremely complex. Since we consider a random walk with any number of steps, we implicitly explore structure in data at multiple scales, and if there is a good path between points then spectral clustering will group them together.

To see the relation with PCE, assume that P has K PCE with respect to the partition and that there exist a value of M for which $\{\lambda_n^M \cong 1\}_{n=2}^K$ and $\{\lambda_n^M \cong 0\}_{n=K+1}^N$. Then, we can approximate P^M by its projection onto the K leading PMCs as $P^M \cong \sum_{n=1}^K \lambda_n^M T_n \cong \sum_{n=1}^K 1 \cdot \frac{v_n v_n^\top D}{v_n^\top D v_n}$. If v_n is PCE, it is easy to verify that the condition of theorem 1 is satisfied, and spectral clustering minimizes the MNCut.

4. Algorithm

We saw that by replacing λ with λ^M and interpreting it as an indication for the convergence of the random walk, the eigenvectors contain cluster information on multiple scales. So, given the transition matrix P and the number of clus-

```
**Input**: Data $S$, Gram matrix $W$
**Output**: $L$ partitionings of $S$ with associated stability and plausibility measures $\left\{\left\{S_k^l\right\}_{k=1}^{K_l}, \alpha_l, \beta_l\right\}_{l=1}^{L}$
**Algorithm**:
1. Compute, in the following order
    -$P$ according to (2)
    -$\{\lambda_n, v_n\}_{n=1}^N$, the eigen system of $P$
    -$\Delta(M)$ and $K(M)$ according to (7) for increasing values of $M$, beginning at $M_0 = 1$, until $K(M) = 1$ (set this number of steps to $M_{max}$)
2.Find all the local maxima of $\Delta(M)$, and order them $\{\Delta(M_l)\}_{l=1}^L$.
3.For every $l = 1, 2, \ldots, L$
    -Call Algorithm 1 with $K(M_l)$ as the number of clusters to obtain $\left\{S_k^l\right\}_{k=1}^{K_l}$.
    -Set $\alpha_l = \frac{M_l - M_{l-1}}{M_{max}}$, $\beta_l = \Delta(M_l)$.
```

Algorithm 2: An algorithm to automatically learn different partitionings of S.

ters K, one can measure the quality of the partitioning by searching for the number of steps M such that the eigengap $\lambda_K^M - \lambda_{K+1}^M$ is maximized. However, a closer look suggest a more powerful usage of this approach. Since different M are associated with different scales of data scattering, we can combine searching for good partitionings with inferring the number of clusters K - all from the eigenvalues of P. This is done by searching over M and looking for local maxima of the maximal eigengap. The number of clusters is inferred from the location of the maximal eigengap, and this maximal value can be used as a quality measure for the partitioning, as formally defined by

$$\begin{aligned} \Delta(M) &= \max_k \left(\lambda_k^M - \lambda_{k+1}^M\right) \ , \\ K(M) &= \operatorname*{argmax}_k \left(\lambda_k^M - \lambda_{k+1}^M\right) \ . \end{aligned} \tag{7}$$

So, for every M we have an estimate $K(M)$ for the number of clusters, and $\Delta(M)$ is interpreted as an indication for the plausibility of the partitioning. Algorithm 2 combines (7) with the following two key ideas. First, good partitionings should have a high value of $\Delta(M)$, ideally 1. Second, $\Delta(M)$ is not monotonic, and by increasing values of M, it is expected to have local maxima that are associated with partitionings in increasing scales.

The algorithm is demonstrated in Figure 2 with a data set consists of three interlocked rings, two of which have a bridge between them. By scanning over M we find that $\Delta(M)$ is locally maximized (bottom graph) at 3 points $M_{1,2,3} = 40, 312, 6309$ respectively. Each of these locations is associated with a different number of clusters $K(M_{1,2,3}) = 9, 3, 2$ respectively. Each of the resulting

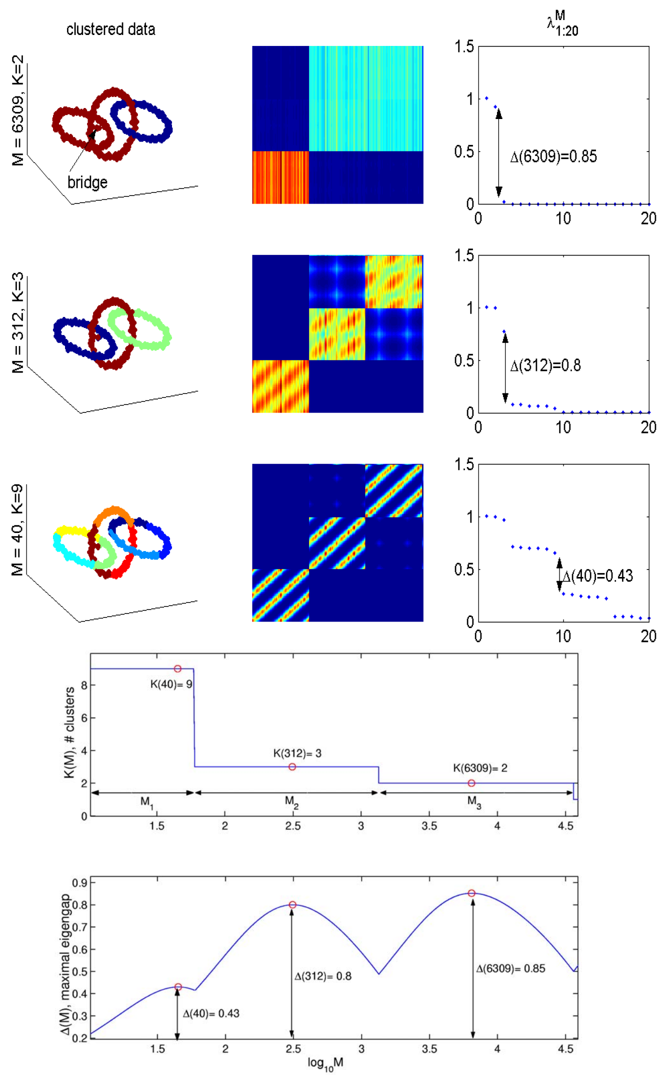

Figure 2. **A demonstration of Random walks.** S consists of three interlocked rings, two of which share a bridge. M is plotted on a base 10 logarithmic scale

partitionings is shown in the top plots. Notice how small values of M (few steps of the random walk) are associated with high number of clusters (each with a small number of points). We also define stability and plausibility measures (algorithm 2, step 3) which can be used to select between partitionings. The stability measure for the partitionings in Figure 2 is $\alpha_{1,2,3} \cong \frac{10^{1.7}}{10^{4.5}}, \frac{10^{3.2}}{10^{4.5}}, \frac{10^{4.3}}{10^{4.5}}$ respectively and the plausibility measure associated with each of them is given by $\beta_{1,2,3} = 0.43, 0.8, 0.85$. We deliberately chose an example which returns three different partitionings, to demonstrate how α and β can be used to help us decide between them.

Figure 3 shows the performance of the algorithm applied to a data set which consists of 300 noisy and randomly ro-

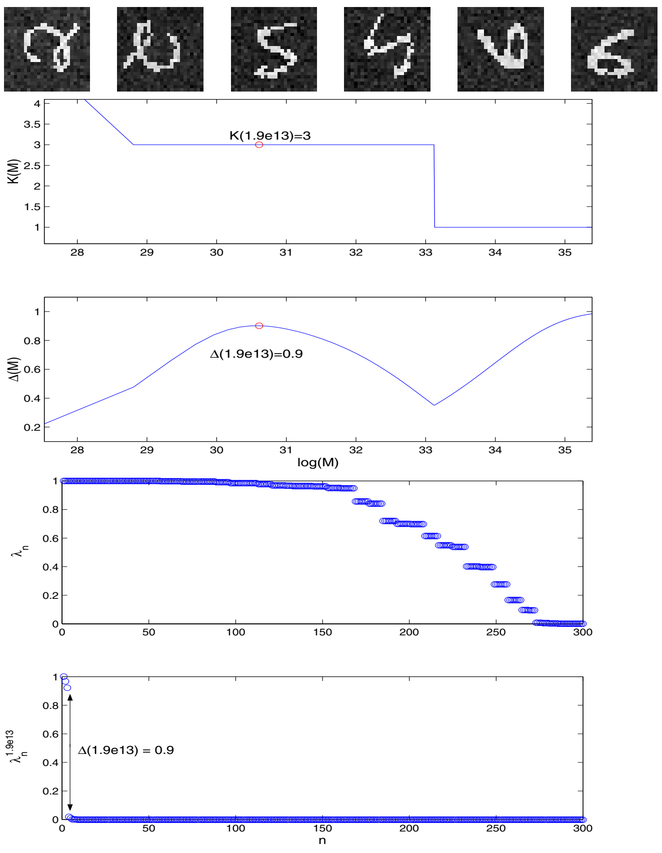

Figure 3. **Digits data set.** Algorithm 2 applied to a data set consists of 300 noisy and randomly rotated digits with 3 labels. This data set is similar in nature to the one in Figure 2 (see text for details). The algorithm automatically finds the number of clusters and a perfect partitioning of the data.

tated digits with 3 labels (100 images per label). Each image is given by a 26×26 matrix and is represented as a point in $\mathbb{R}^{676}$. Notice how P's spectrum is practically flat and noninformative, while its power series indicate that there are 3 eigenvalues that survive the random walk exploration. Due to the random rotation of the digits, where the rotation is uniformly distributed over $[0^{\text{o}}, 360^{\text{o}})$, this data set is formed of rings in $\mathbb{R}^{676}$. Applying K-means to this data set severely mixes the labels, indicating that the rings are interlocked. In addition, applying Algorithm 2 to this set results in a perfect separation of the digits. Thus, this data set is similar in nature to the one in Figure 2, and the fact that only one peak of $\Delta(M)$ emerges teaches us that there are no good bridges between clusters.

5. Learning the affinity matrix

Recall (definition 1) that the Gram matrix is parameterized by σ, the length scale of the affinity function. Choosing a good value for this parameter is crucial for the algorithm to be successful. How to choose the parameters of the affinity matrix is still an open question and an active research area, having a few possible solution strategies in the literature. For example, in [2] it was assumed that K is given and σ was chosen by minimizing the distortion of the K-means algorithm in the feature space (Algorithm 1, step 3). In [5] the affinity matrix was learned by forcing it to perform well on clustering a given labeled sequence. A training sequence is also used in [9], where the affinity matrix is learned by optimizing a regularized objective function.

A major drawback of the above methods is that they require K to be given with the data and they [5, 9] involve solving complex optimization problems to learn the affinity matrix. In section 4 we saw how, given W, different values for K can be easily inferred with almost no additional complexity, beside computing the powers of the leading eigenvalues of P for a range of possible values of M. However this procedure assumes the affinity matrix, parameterized by σ, is known. To learn this parameter, notice that its value eventually determines the transition probabilities between pairs of points, and by increasing it we implicitly increase the probability of a random walk to move between points that are further apart. It is worth stressing here the fundamental difference between scanning over the number of steps M and over the length scale parameter σ. Increasing the value of σ results in a transition matrix $P(\sigma)$ that spreads the probability of a single step to move between points that are further and further apart. By scanning over M, we simply discover plausible number of clusters in the data set, based on the single step transition probabilities given by $P(\sigma)$. It can be summarized as follows; *by setting the value of σ we explicitly define the structure in the data, and by scanning over M we reveal this structure.*

The interesting interval for σ roughly stretches from the smallest to the largest distance between pairs of data points. Σ is defined to satisfy this condition

$$\Sigma = \left[\sigma_{\min} : \frac{\sigma_{\max} - \sigma_{\min}}{|S| - 1} : \sigma_{\max}\right], \qquad (8)$$
$$\sigma_{\min} = \min_{m,n} \|x_m - x_n\|, \; \sigma_{\max} = \max_{m,n} \|x_m - x_n\| .$$

```
Input: Data set $S$
Output: $L$ partitionings of $S$ with associated stability and plausibility measures $\left\{\{S_k^l\}_{k=1}^{K_l}, \alpha_l, \beta_l\right\}_{l=1}^{L}$
Algorithm:
0. Initialize $\Sigma$ according to (8).
1. For $i = 1 : |S|$, set $\sigma_i = \Sigma(i)$ and compute $\Delta(\sigma_i, M)$ and $K(\sigma_i, M)$ according to (9)
2. Choose $\sigma^\star = \text{argmax}_{\sigma_i, M} \Delta(\sigma_i, M)$ to be the parameter that best reveals structure in $S$.
3. Call Algorithm 2 with $W(\sigma^\star)$ as the Gram matrix.
```

Algorithm 3: An algorithm to find different pairs of (σ, K).

where $[a:b:c]$ denotes the sequence $a, a+b, a+2b, \ldots, c$. The boundaries of Σ are defined such that for $\sigma_{\min}$ most of the mass of the transition probability, for each point in S, is distributed over the very nearest few data points, whereas for $\sigma_{\max}$ it is distributed more evenly over a large number of points. Using $|S|$ and not N simplifies the notation in section 6. Notice that there are now $|S|$ candidate transition matrices which can be used for the actual partitioning of the data. To take this into consideration we can generalize functions (7) to be a function of both M and σ

$$\begin{aligned}\Delta(M,\sigma) &= \max_k \left(\lambda_k^M(\sigma) - \lambda_{k+1}^M(\sigma)\right) \ , \\ K(M,\sigma) &= \operatorname*{argmax}_k \left(\lambda_k^M(\sigma) - \lambda_{k+1}^M(\sigma)\right) \ .\end{aligned} \tag{9}$$

The two dimensional functions (9) allow the derivation of a family of algorithms that learn σ and K simultaneously. The general idea is that different values of σ define relations between points in different scales and result in discovering different number of clusters in the data. The parameter σ defines the local neighborhood structure, while scanning over M reveals the global structure. We therefore optimize over both σ and M and look for the combination that yields the maximal value of $\Delta(M,\sigma)$. Algorithm 3 gives a formal description of how this can be implement.

6. Hierarchical clustering

Algorithm 3 is inherently inefficient, as it uses in step 1 the entire data set S for every length scale σ, completely ignoring the fact that different length scales result in different types of partitioning. In general, the larger σ is, the larger each cluster is expected to be. Algorithm 4 is essentially a hierarchical implementation of Algorithm 3, beginning by separation in a large scale, and then recursively partitioning each of the resulting clusters. Specifically, it first searches for a good initial partitioning of S, beginning with $\Sigma(|S|)$ and going backwards. Once a local maxima of $\Delta(\sigma, M)$ is found, S is partitioned and each cluster in the partitioning is then treated as a new data set and Algorithm 4 is applied to it recursively (step 4). The recursive nature of Algorithm 4 is implemented by growing a tree, adding more and more levels until the clusters reach a desired size or crossing a threshold in performance improvement. Notice that algorithm 4 is different from algorithms 2,3 by only producing one partitioning of the data. Nevertheless, the tree structure provides additional information about the scale of the relation between different clusters.

In addition to the many conceptual advantages of hierarchical data clustering such as simplicity of representation or the efficiency of the 'divide and conquer' principle, the computational burden is also eased significantly. Notice that in each recursion the data is clustered, so that the size of S that is used to call the algorithm recursively decreases exponentially. This is translated to easier computational tasks

```
Input: Data set $S$.
Output: A directed tree T, each leaf contains all the points in the associated cluster.
Algorithm:
0. initialize according to (8), and set T to be an empty tree.
1. starting with the largest value $\Sigma(|S|)$ and going backwards, compute $\Delta(\sigma_i, M)$ until a local maxima emerges at some pair $(\sigma^\star, M)$.
2. compute $K(\sigma^\star, M)$ according to (9)
3. call Algorithm 1 with $W = W(\sigma^\star)$ and $K = K(\sigma^\star)$ to obtain $\{S_k\}_{k=1}^K$.
4. for every $k = 1 : K$ call Algorithm 4 (recursive) with $S_k$ and define the output as the $k$'th child of T.
```

Algorithm 4: Hierarchical implementation of Algorithm 3.

by both using smaller matrices P and smaller number of elements in Σ. Another implementation issue is the spectral decomposition of P required for step 2. Since the maximal eigengap is all that we are looking for, it is only necessary to compute a small number of the largest eigenvalues, much smaller than N, the total number of eigenvalues. Recall that P is a usually sparse matrix, thus Lanczos algorithm can be used to achieve an efficient implementation and reduce execution time (for implementation in MATLAB® see also the command eigs).

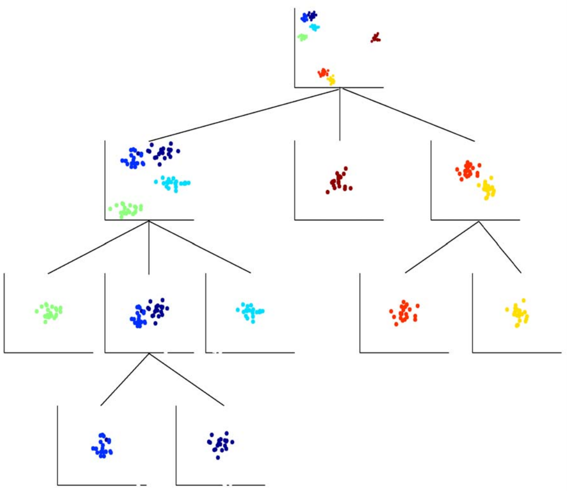

Figure 4. **Numerical demonstration of Algorithm 4**. The data S was generated from a mixture of 7 nonuniformly scattered Gaussians. Every new partitioning in the tree is achieved using the value of σ which maximizes $\Delta(\sigma, M)$ for the associated data set. Clustering in higher levels of the tree are obtained using larger values of σ and each split is associated with a different value of σ.

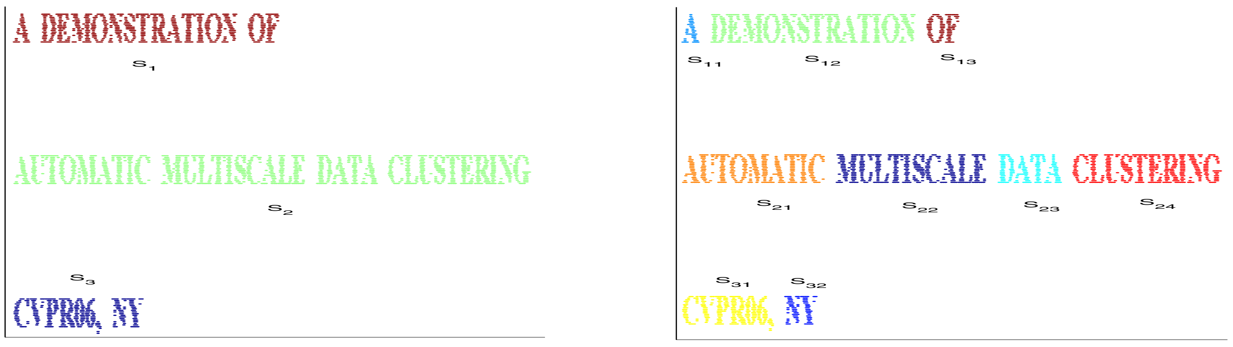

Figure 5. **Numerical demonstration of Algorithm 4**. Data S consists of 9 words arranged on 3 lines. $\{S_i\}_{i=1}^3$ are the initial clusters found by algorithm 4, using the top values of Σ. For each $i = 1, 2, 3$, algorithm 4 was called recursively with S_i, and the sets $S_{i,j}$ were clustered as the children of S_i. The input to the algorithm is the set of all points, and the output is a tree with 2 levels and 9 leafs.

7. Discussion

This paper sheds new light on spectral clustering. Specifically, lemma 3 and Theorem 1 generalize the results of [7] by considering P^M instead of P, leading to a new theoretical framework for understanding spectral clustering. It was shown how P's eigenvalues can be interpreted as an indication for the cluster structure of the data clarifying two aspects of spectral methods; first, it explains why we choose the eigenvectors associated with the largest eigenvalues and second, it gives a direct relation between the eigenvalues and the number of clusters in the data. These observations first motivated algorithm 2, where given the data set S and the Gram matrix W, several plausible values for the number of clusters K are inferred. In algorithm 3 only the data S is given, and the search for the affinity matrix is combined with the search for the number of clusters K. Finally, to have an efficient implementation of this algorithm, we defined a hierarchical algorithm that iteratively partitions the data so that the computational burden in each stage decreases exponentially. The algorithm for determining σ can be extended to finding any other single parameter of a parameterized Gram matrix.

Our results also give a general framework for matrix decomposition which may be useful for other data analysis and machine learning algorithms (e.g. PCA, markov chains). In relation to image segmentation algorithms, we already discussed how our results helps understanding when (there are no good bridges between segments) and why (the random walk doesn't easily move between segments) the method is expected to be successful. In addition, Theorem 1 can also be used to improve other existing algorithms for image segmentation. For example, Theorem 1 in [9] shows how the eigengap $\lambda_K - \lambda_{K+1}$ can be used to define a quality measure, but because this number often tends to be very small, using it directly leads to numerical instability of their algorithm and the authors say a less efficient objective function is optimized. Using our method, this eigengap can be easily replaced by $\lambda_K^M - \lambda_{K+1}^M$ with any M for which the numerical problems are minimized. On the other hand, and unlike [9], our method assigns all dimensions with the same length scale parameter. Thus, although it allows a very simple implementation, it has the weakness of not being able to perform feature selection. One of the directions for future work is extending our algorithms to allow using different σ along different dimensions.

References

[1] A. Shokoufandeh, and S. Mancoridis , and M. Maycock. Applying Spectral Methods to Software Clustering. *Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE02)*.

[2] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS),14*, 2001.

[3] D. Verma, and M. Meila. Comparison of spectral clustering methods. *UW CSE Technical report*, 2001.

[4] F.R. Bach, and M.I. Jordan. Blind one-microphone speech separation: A spectral learning approach. *Advances in Neural Information Processing Systems (NIPS),16*, 2004.

[5] F.R. Bach, and M.I. Jordan. Learning Spectral Clustering. *Advances in Neural Information Processing Systems (NIPS),16*, 2004.

[6] J. Shi, and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22.

[7] M. Meila. The multicut lemma. *UW Statistics Technical Report 417*, 2001.

[8] M. Meila, and J. Shi. A Random Walks View of Spectral Segmentation. *Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2001.

[9] M. Meila, and S. Shortreed, and L.Xu. Regularized spectral learning. *UW Statistics Technical Report 465*, 2005.

[10] W. Pentney, and M. Meila. Spectral Clustering of Biological Sequence Data. *To appear in AAAI 2005*.