Spectral Clustering of Biological Sequence Data *

William Pentney
Department of Computer Science and Engineering
University of Washington
bill@cs.washington.edu

Marina Meila
Department of Statistics
University of Washington
mmp@stat.washington.edu

Abstract

In this paper, we apply spectral techniques to clustering biological sequence data that has proved more difficult to cluster effectively. For this purpose, we have to (1) extend spectral clustering algorithms to deal with asymmetric affinities, like the alignment scores used in the comparison of biological sequences, and (2) devise a hierarchical algorithm that can handle many clusters with imbalanced sizes robustly. We present an algorithm for clustering asymmetric affinity data, and demonstrate the performance of this algorithm at recovering the higher levels of the Structural Classification of Proteins (SCOP) on a data base of highly conserved subsequences.

Introduction

The problem of clustering data items into related groups based on similarity is an extremely common problem arising in a variety of disciplines and applications, and clustering algorithms for various applications have been studied for decades. Such algorithms depend upon the knowledge or acquisition of similarity information to relate data items to each other, e.g. the affinity between points in Euclidean space. Spectral techniques, which make use of information obtained from the eigenvectors and eigenvalues of a matrix, have attracted increasing research attention with respect to clustering in recent times.

Here we present a method to apply spectral techniques to the clustering of biological sequence data. The points in the data used represent proteins, considered as amino acid sequences, and the affinities between sequences are evaluated using an algorithm for sequence comparison, like BLAST (Altschul *et al.* 1997) or the Smith-Waterman algorithm (Smith & Waterman 1981). The pairwise *alignment scores* for biological sequences differ from the the similarities usually considered by spectral clustering algorithms in that they are not necessarily symmetric: the affinity between sequences a and b may not be the same as the similarity between b and a for several reasons, including the different lenghts of the sequences as well as the fact that the (amino acid) substitution matrices on which the scores are based can be themselves asymmetric.

The breakdown of this work is as follows: First, we will introduce preliminary concepts and notation, a discussion of the techniques used to partition this data, and a brief overview of the SCOP hierarchy. We then present the algorithm used to cluster this protein data by class and fold, and then discuss some related work. Next, we will present the results of experiments on this data against other common techniques. Finally, we present conclusions and suggest future directions of study on the subject.

Preliminaries

Notation and Spectral Clustering

We are concerned with a set of data items which will be viewed as vertices in a weighted directed graph $G = (V, E)$. Say $|V| = n$. The weight on an edge $(i, j) \in E$ represents the affinity from i to j according to some measure. (Since G is directed, note that (i, j) may not be equal to (j, i).) Let $S \in \mathbb{R}^{n \times n}$ be the matrix of affinities between items, i.e. S_{ij} is the affinity of i for j. A *clustering* of G will be defined as a partition $\Delta = (C_1, C_2, ...C_K)$ of the vertices V in G, where each $C_i \subset V$. We can also define a clustering in terms of a set of indicator vectors $x_1, x_2, ...x_K$ where $x_k(i) = 1$ if point i is in cluster k and 0 otherwise.

We will define $d_i^{(o)} = \sum_j S_{ij}$ to be the *outdegree* of i and $d_i^{(i)} = \sum_j S_{ji}$ to be the *indegree* of i. Let $D^{(o)}$ and $D^{(i)}$ be the diagonal matrices whose ith diagonal entries are $d_i^{(o)}$ and $d_i^{(i)}$, respectively.

Consider the matrix $P = {D^{(o)}}^{-1} S$. Let $u^1, u^2, ...u^n$ be the eigenvectors and $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ the corresponding eigenvalues of P. Note that $\lambda_1 = 1$. When S is symmetric the components of each u^i are all real (Meila & Shi 2001); if P is asymmetric, however, these values may be complex.

The SCOP Data Set

The SCOP database is a collection of protein sequences used for evolutionary classification; a detailed description of its features can be found in (Andreeva *et al.* 2004). The proteins have been classified in terms of similarity in conserved structural features, and divided into the following groups,

*The authors acknowledge Bill Noble for useful discussions and partial support from NSF grant IIS-0313339.

Class	No. of samples	No. of folds
1	804	128
2	950	87
3	694	93
4	737	166
5	54	25
6	121	10
7	992	52

Figure 1: The number of proteins and folds in each class of the selected SCOP data.

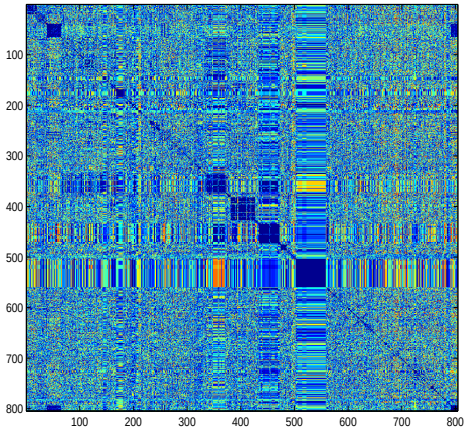

Figure 2: Image of the distance matrix for class 1 of the SCOP data.

from most general to most specific: class, fold, superfamily, family, and domain.

For our experiments, we used data collected from version 1.53 of the SCOP database and compiled for the experiments in (Liao & Noble 2002), which contained 4,352 proteins, seven classes, 564 folds, and 859 superfamilies. Similarities between proteins were calculated using the Smith-Waterman algorithm. The number of proteins in the subdivisions along each level of the hierarchy can vary wildly, which can present another issue for clustering algorithms; some contain very few items indeed, and thus are more difficult to cluster. Information on the number of proteins and folds in each of the seven classes is contained in Figure 1. Figure 2 contains an image of the distance matrix for the first class in SCOP data; note that the clusters vary in size and that the matrix is somewhat asymmetric.

Spectral mapping for asymmetric data

P is a stochastic matrix which can be considered to represent the transition matrix for a random walk along G; from vertex i, the probability of moving to vertex j under such a process will be $\frac{S_{ij}}{d_i^{(o)}}$. This random walks model will be the basis for our spectral clustering. This random walks model of spectral clustering problems is introduced and further detailed in (Meila & Shi 2001). It should be noted that the model we are presenting here allows for the use of an asymmetric affinity function, like that computed by BLAST or the Smith-Waterman algorithm.

Spectral algorithms for clustering data with symmetric affinities have been detailed in many other sources, e.g. (Meila & Shi 2001), (Shi & Malik 2000), and (Ng, Jordan, & Weiss 2002). In (Meila & Xu 2003) it is shown that several spectral clustering algorithms minimize the *multiway normalized cut*, or $MNCut$, induced by a clustering Δ on G, measured as

$$MNCut(\Delta, G) = \sum_{k=1}^{K} \frac{\sum_{i,j \in E} x_i x_j S_{ij}}{\sum_{i \in V} d_i^{(o)}}$$

These guarantees, however, have not been shown to apply to asymmetric data. Therefore we will assume here that the data we are dealing with is sufficiently concentrated in the diagonal blocks representing the true clusters to allow one to use the first K eigenvectors of P to recover the clustering. This hypothesis is plausible for biological sequence data, because we expect that two sequences that are highly similar to a third will also be sufficiently similar to each other, but we will further verify it in our experiments.

Now we prove a result that allows us to proceed with spectral clustering of asymmetric data and make sense of the results. The following proposition is an extension of a result proved in (Meila & Shi 2001) for symmetric data.

We define a vector v to be *piecewise constant (PC)* relative to a partition $\Delta = (C_1, C_2, \ldots C_k)$ of V iff $v_i = v_j$ for i, j in the same set C_s, $s = 1, \ldots k$. Note that the first eigenvector of P, being $\mathbf{1}$, is always piecewise constant.

Proposition 1 *Let P be a matrix with rows and columns indexed by V that has independent eigenvectors. Let $\Delta = (C_1, C_2, \ldots C_K)$ be a partition of V. Then, P has K eigenvectors that are piecewise constant w.r.t. Δ if and only if the sums $P_{is} = \sum_{j \in C_s} P_{ij}$ are constant for all $i \in C_{s'}$ and all $s, s' = 1, \ldots K$.*

The proof is given in the extended version of the paper. Intuitively, Proposition 1 says that a stochastic matrix has piecewise constant eigenvectors if the Markov chain represented by P can be aggregated into a Markov chain with state space $\Delta = \{C_1, \ldots C_K\}$ and transition probability matrix $\hat{P}$. If we look at the transition probability from a point $i \in C_s$ to any of the clusters C_k denoted P_{iC_k}, this probability is equal to $\hat{P}_{C_s C_k}$ and it is the same no matter which $i \in C_s$ is chosen. Hence, when the eigenvectors of P are PC, even though complex, we can say that the data points are being clustered based on how similarly they transition to each of the clusters.

Practically, we will not use complex eigenvectors in our experiments, but vectors of the form $\tilde{u} = \mathrm{Re}(u) + \mathrm{Im}(u)$. Recalling that for every complex eigenvector u of a real matrix P the complex conjugate $\bar{u}$ is also an eigenvector of P, and that $\tilde{\bar{u}} = \mathrm{Re}(u) - \mathrm{Im}(u)$, we see that the preceed-

ing operation preserves all the original information about the eigenvectors.

In the next section we discuss how we use the eigenvectors for clustering. In the remainder of this section we offer an example contrasting our choice of computing eigenvectors with the popular approach of transforming S into a symmetric matrix, then using standard spectral clustering on the new S.

Hierarchical Clustering with the Eigenvectors

We adopt a hierarchical strategy, in which will we begin with one large cluster containing all data points and repeatedly partition clusters into smaller clusters. We will assume that the "real" clusters in the data are represented by nearly piecewise constant segments of the eigenvectors of P.

Because the data itself has a hierarchical classifi cation we prefer to use a recursive partitioning method similar to (Shi & Malik 2000; Vempala, Kannan, & Vetta 2000) and to use one eigenvector (the second largest) at every step. This method is also convenient as more robust to noise: When the number of clusters K is relatively large, the more eigenvectors one considers for clustering data the noisier the $\mathbb{R}^K$ clustering problem becomes. In (Meila & Verma) the advantages of recursive partitioning for noisy affi nities are demonstrated experimentally.

To extract the PC parts of an eigenvector u, we smooth it with a Gaussian kernel obtaining the (continuous) function

$$\Phi_u(x) = \frac{1}{nh} \sum_{j=1}^{n} \exp\left[-\frac{1}{2}(\frac{(x-u(j))^2}{h^2}\right]$$

The parameter h represents the *kernel width*; a proper choice of kernel width will vary depending on the range of the eigenvectors' components and the closeness of the clusters. A choice of h too high will not produce the correct peaks in the data, while a value of h too low will decrease resistance to noise. We selected a value of approximately $\frac{1}{2^6}(\max_j u_j - \min_j u_j)$ for our experiments. We then select a threshold T, and form a cluster from the points covered by each of the peaks found in $\Phi_u(x)$. This is demonstrated in Figure 4. The points j for which $\Phi(u_j) < T$ are considered *outliers* and not assigned to any cluster. In our experiments we have chosen $T = 2/(2\pi nh)$ which ensures that no cluster with less than 3 points is formed, but preserves larger, more spread clusters well; this value may be altered as appropriate to a particular data set. Note that making T dependent on h effectively reduces our parameter choices to one: that of the width h.

While this process does extract clusters that are nearly piecewise constant in this eigenvector, it is not foolproof; peaks may emerge in this function from multiple real clusters sharing very similar values in this particular eigenvector, and thus possibly appearing to represent a single cluster. Therefore we must check if the obtained clusters cannot be recursively split. If a cluster produces no further split, we consider that cluster to be "terminal". We found this to be a generally effective partitioning strategy in practice.

To partition a cluster into smaller clusters, we compute this KDE function Φ with respect to the components of the

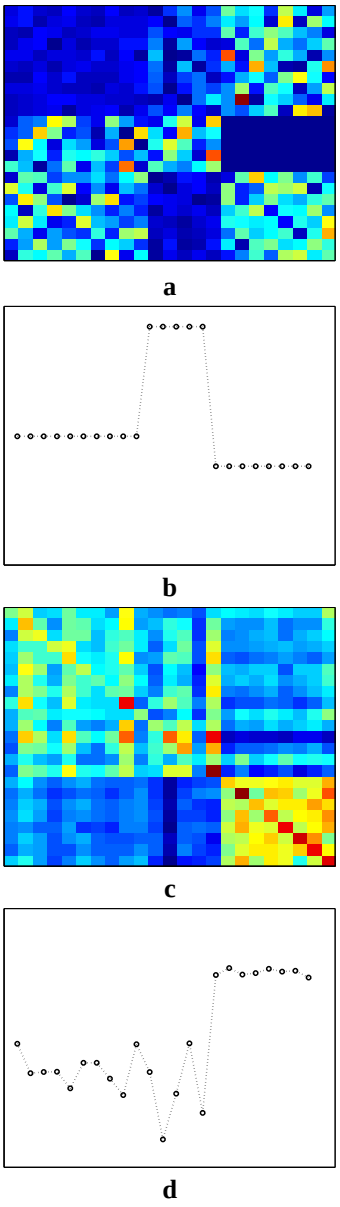

Figure 3: Structure loss by symmetrization: A transition matrix P obtained from an asymmetric S **(a)**. The first eigenvector of P is constant since P is stochastic. The second and third eigenvectors are piecewise constant and complex conjugate. The real part of one of them is plotted in **(b)** vs the data points, demonstrating that there are 3 clusters in the data. The transition matrix $\tilde{P}$ obtained from the symmetrized affinity $\tilde{S} = S^T S$ **(c)**. The matrix $\tilde{P}$, representing a reversible Markov chain, has real eigenvectors. The clustering in P is partly effaced in $\tilde{P}$. The eigenvectors of $\tilde{P}$ are not piecewise constant **(d)**. Therefore, spectral clustering algorithms working on $\tilde{P}$ are unlikely to discover this clustering. (Here only the second eigenvector is plotted, but all of them have been examined).

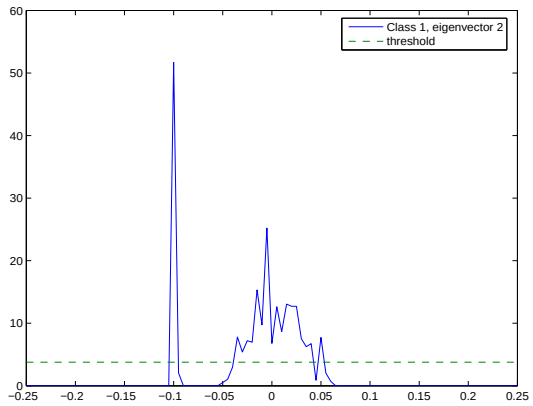

Figure 4: The function produced for an eigenvector for the P matrix of one class of protein data using a Gaussian kernel. Points whose corresponding components in the eigenvector occur in regions above the threshold, denoted by the dotted line, are assigned to a new cluster, while those below the threshold are considered outliers.

eigenvector in that cluster. Note that at each step, we do not consider the points outside of the cluster being split when we compute $\Phi(x)$. We thus produce a new set of clusters $c_1, c_2, ...c_r$, where each c_i is the set of points within a distinct peak (region exceeding the threshold) of the kernel function. The points in the set $c_{(u)} = V - \bigcup_{i=1..r} c_i$ will still be unclustered.

At each step of the algorithm, we consider all the clusters currently discovered, beginning with one cluster for the entire data set, and extract the c_i sets above. We then cluster the points in c_i by creating a new random walk matrix P_{c_i}, consisting of the normalized distances between all points in c_i, and recursively clustering them. We then cluster all points in c_u recursively as well, combining the clusterings produced by the recursive calls. If no sets c_i exist, or we have no unclustered points and all points are above the threshold, we simply return a single cluster; this is a terminal condition of the algorithm.

Upon completion of this process, there are often many points that have not been placed in a cluster. We complete the algorithm by then placing each of these points into a cluster containing only itself, and then performing iterations of the single linkage algorithm on the resulting clustering until the desired number of clusters is achieved. Note that this process may merge some of the clusters produced by the recursive divisions.

Clustering Algorithm

The pseudocode for the recursive portion of our hierarchical clustering algorithm to cluster G with random walk matrix P is as follows. We assume an appropriate choice of kernel width h here, as described.

RHC(P**)**

- Find the second eigenvector u^2 of P.
- Calculate KDE function $\Phi_{u^2}(x)$.
- Find all regions of maximal size (a_i, b_i) of $\Phi_{u^2}(x)$ such that $\Phi_{u^2}(x) \geq T$ for all $a_i \leq x \leq b_i$.
- Place all $u^2(x)$ such that $a_i \leq u^2(x) \leq b_i$ into sets c_i, and all other points into the set $c_{(u)}$. Return a single cluster if no c_i exist, or only one c_i exists and $c_{(u)}$ is empty.
- Calculate the matrix P_{c_i} for each c_i and run **RHC(**P_{c_i}**)** to cluster the points in c_i.
- Run **RHC(**P_{c_u}**)** to cluster $c_{(u)}$.
- Combine the clusterings of all c_i and $c_{(u)}$ to produce the clustering Δ.

After this process, we then place each point never assigned to a c_i into a new cluster containing only itself, and then run single linkage beginning from this clustering of the data, as described above.

Related Work

An early example of a spectral clustering algorithm is the Shi/Malik algorithm for spectral segmentation (Shi & Malik 2000), which uses a matrix of affi nities between regions and partitions the regions based on the components of the second smallest eigenvector of L. Our algorithm is similar to the Shi/Malik algorithm in its use of the second eigenvector of the matrix P; in the symmetric case, the eigenvectors of P and L can be easily shown to be related. Our algorithm differs in its method of selecting a partition of a cluster and in its use of a kernel function rather than simply sorting the components of the eigenvector.

In (Meila & Shi 2001), a random walks view of spectral segmentation was proposed which presented a k-clustering algorithm based on the use of the eigenvectors corresponding to the k largest eigenvalues of a normalized affi nity matrix; this model is used in the algorithm described here as well. (Ng, Jordan, & Weiss 2002) presents a similar algorithm using the eigenvectors of the Laplacian. (Vempala, Kannan, & Vetta 2000) contains a theoretical analysis of spectral clustering algorithms and provides worst-case guarantees on performance.

Finally, the issue of clustering biological sequence data has attracted intense interest in recent years - examples are (Yeung *et al.* 2001) and (Yeung & Ruzzo 2001), among many others. As of yet, however, little research has been performed on applying spectral algorithms to this problem.

Experiments

We compared the performance of our algorithm on the SCOP data set against four other techniques: a single-linkage (nearest neighbor) hierarchical algorithm, MDS with single-linkage to cluster the points in the lower dimension, SVD with single-linkage to cluster singular vectors, and the Meila/Shi algorithm of (Meila & Shi 2001). Experiments were also performed on the last three methods using k-means instead of single linkage, but results were comparable or worse. For MDS and Meila/Shi, which require a symmetric affi nity function in their traditional form, we fi rst

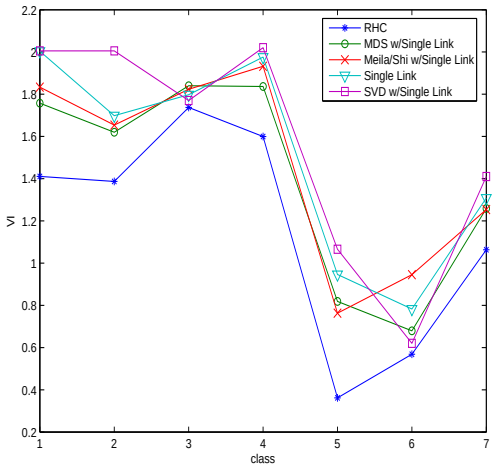

Figure 5: Plot of performance of each algorithm on clustering the folds of one class of the SCOP protein data.

symmetrized the SCOP affi nity matrix by creating a matrix S', where $S'_{ij} = \frac{S_{ij}+S_{ji}}{2}$, then computing the equivalent stochastic matrix P' and using this affi nity matrix for our calculations. We tested these algorithms by clustering the folds within each of the seven classes of the SCOP data. The single linkage clustering used in each technique was terminated when the number of clusters in the original data was obtained.

To measure the difference between clustering, we use the variation in information (VI) criterion to compare each clustering to the true clustering of the data by folds; a full description of this criterion may be found in (Meila 2003). Here, when compared against the true clustering of the protein data, a higher VI value indicates a clustering less similar to the true clustering, and thus a clustering of lower quality. A VI of 0 indicates a perfect reproduction of the true clustering.

Results

A plot of the performance of each of the algorithms on each of the seven classes can be seen in Figure 5. We refer to the algorithm described above as RHC. We see that RHC outperforms the other methods of clustering the data, providing the closest clustering to the original.

One likely explanation for this is that RHC is particularly effective at fi nding disproportionately large clusters. The largest clusters in each class are usually extracted from the data within the fi rst few iterations of the algorithm, while other algorithms were more likely to break these clusters up inappropriately.

The recusrive procedure used by the algorithm is less effective at fi nding smaller, more isolated clusters, many of which would often be clustered together regardless of distance; these clusters were presumably not large enough to produce a peak in Φ_{u^2}. However, the single linkage iterations used in the fi nal step of the algorithm can be effective at fi nding these clusters.

The algorithm also has some problems with handling data with clusters of considerably different "tightness". Some clusters in the data consist of very dense subclusters with some distance between them, while other clusters may have greater distance between their points; currently the algorithm is apt to further divide clusters with dense components into smaller clusters, at the expense of fi nding sparser clusters. Addressing this problem can be diffi cult, but could perhaps be addressed with an appropriate scheme to alter kernel width and threshold in recursive runs of the algorithm.

Conclusions and Future Work

The algorithm we have presented offers competitive performance on the clustering of biological sequence data. It also introduces to technical innovations of general applicability: First, a extension of spectral methods to the clustering of asymmetric affi nity data; as this paper has been shown, symmetrizing the data may loose information relevant to clustering. Second, a new hierarchical method of partitioning an eigenvector which, using kernel smoothing, is more robust to perturbations than the classical methods based on sorting the eigenvector values.

Currently, the choice of kernel height and threshold parameters has been done manually; it would be preferable were there an unassisted or more sophisticated means of deciding upon these parameters. Other means of partitioning or adaptively selecting a kernel could be explored.

Finally, the study of applying spectral techniques, or indeed any particular techniques, to clustering asymmetric pairwise data has still not been well explored ((Ozawa 1983) and (Wang & Kitsuregawa 2001) are exceptions). Development of spectral clustering techniques for such data is more challenging, as less is known about the behavior of eigenvectors of the distance matrix in such matrices; they can be unpredictable, and the eigenvalues are susceptible to significant change with minor perturbation of the matrix as well. The question of how to handle the clustering of such data, which can emerge in other domains such as social networks as well, is an interesting direction for both theoretical and experimental research.

References

Altschul, S.; Madden, T.; Schaffer, A.; Zhang, J.; Anang, Z.; Miller, W.; and Lipman, D. 1997. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucl. Acids Res.* 25:3389–3402.

Andreeva, A.; Howorth, D.; Brenner, S.; Hubbard, T.; Chothia, C.; and Murzin, A. 2004. Scop database in 2004: refi nements integrate structure and sequence family data. *Nucleic Acids Research* 32:226–229.

Liao, L., and Noble, W. 2002. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Proceedings of the Sixth Internationl Conference on Computational Molecular Biology* 225–232.

Meila, M., and Shi, J. 2001. A random walks view of spectral segmentation. *AI and Statistics (AISTATS)*.

Meila, M., and Verma, D. A comparison of spectral clustering algorithms. *University of Washington Department of Computer Science Tecnical Report 03-05-01*.

Meila, M., and Xu, L. 2003. Multiway cuts and spectral clustering. *UW Dept. of Statistics Technical Report*.

Meila, M. 2003. Comparing clusterings. *16th Annual Conference on Learning Theory*.

Ng, A.; Jordan, M.; and Weiss, Y. 2002. On spectral clustering: Analysis and an algorithm. *NIPS*.

Ozawa, K. 1983. Classic: A hierarchical clustering algorithm based on asymmetric similarities. *Pattern Recognition* 16(2):202–211.

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

Smith, T., and Waterman, M. 1981. Identifi cation of common molecular subsequences. *Journal of Molecular Biology* 147:195–197.

Vempala, S.; Kannan, R.; and Vetta, A. 2000. On clusterings - good, bad and spectral. *Proc. 41st Symposium on the Foundation of Computer Science*.

Wang, Y., and Kitsuregawa, M. 2001. Link based clustering of Web search results. *Lecture Notes in Computer Science* 2118:225–235.

Yeung, K., and Ruzzo, W. 2001. An empirical study of principal component analysis for clustering gene expression data. *Bioinformatics*.

Yeung, K.; Fraley, C.; Murua, A.; Raftery, A.; and Ruzzo, W. 2001. Model-based clustering and data transformations for gene expression data. *Bioinformatics* 17:10:987–997.