Algorithmic Approaches to Clustering Gene Expression Data

Ron Shamir

School of Computer Science

Tel-Aviv University

Tel-Aviv 69978, Israel

rshamir@tau.ac.il

Roded Sharan

School of Computer Science

Tel-Aviv University

Tel-Aviv 69978, Israel

roded@tau.ac.il

11.1 Introduction

Technologies for generating high-density arrays of cDNAs and oligonucleotides are developing rapidly, and changing the landscape of biological and biomedical research. They enable, for the first time, a global, simultaneous view on the transcription levels of many thousands of genes, when the cell undergoes specific conditions or processes. For several organisms that had their genomes completely sequenced, the full set of genes can already be monitored this way today. The potential of such technologies is tremendous: The information obtained by monitoring gene expression levels in different developmental stages, tissue types, clinical conditions and different organisms can help understanding gene function and gene networks, and assist in the diagnostic of disease conditions and of effects of medical treatments. Undoubtedly, other applications will emerge in coming years.

A key step in the analysis of gene expression data is the identification of groups of genes that manifest similar expression patterns. This translates to the algorithmic problem of clustering gene expression data. A clustering problem consists of elements and a characteristic vector for each element. A measure of similarity is defined between pairs of such vectors. (In gene expression, elements are usually genes, the vector of each gene contains its expression levels under each of the monitored conditions, and similarity can be measured, for example, by the correlation coefficient between vectors.) The goal is to partition the elements into subsets, which are called *clusters*, so that two criteria are satisfied: *Homogeneity* - elements in the same cluster are highly similar to each other; and *separation* - elements from different clusters have low similarity to each other.

In this chapter we describe some of the main algorithmic approaches to clustering gene expression data. Clustering is a fundamental problem which has numerous other applications in biology as well as in many other disciplines. It also has a very rich literature, going back at least a century, and according to some authors, all the way to Aristo. Any such review is thus necessarily incomplete, and reflects the background, taste and biases of the authors.

11.2 Biological Background

In this section we outline three technologies that generate large scale gene expression data. All three are based on performing of a large number of hybridization experiments in parallel on high density arrays (a.k.a. "DNA chips"), between probes and targets. They differ in the nature of the probes and the targets and in other technological aspects, which raise different computational issues in analyzing the data. For more on the technologies and their applications see, e.g., [Marshall and Hodgson, 1998, Ramsay, 1998, Eisen and Brown, 1999, Chi, , Lockhart and Winzeler, 2000]

11.2.1 cDNA Microarrays

cDNA microarrays [Schena *et al.*, 1996, Schena, 1996, Marshall and Hodgson, 1998, Ramsay, 1998] are microscopic arrays which contain large sets of cDNA sequences immobilized on a solid substrate. In an array experiment, many gene-specific cDNAs are spotted on a single matrix. The matrix is then simultaneously probed with fluorescently tagged cDNA representations of total RNA pools from test and reference cells, allowing one to determine the relative amount of transcript present in the pool by the type of fluorescent signal generated. Current technology can generate arrays with over 10,000 cDNAs per square centimeter.

cDNA microarrays are produced by spotting PCR products of approximately 0.6-2.4 KB representing specific genes onto a matrix. The spotted cDNAs are usually chosen from appropriate databases, e.g., GenBank [Benson *et al.*, 1999] and UniGene [Schuler, 1997]. Additionally, cDNAs from any library of interest (whose sequences may be known or unknown) can be used. Each array element is generated by the deposition of a few nanoliters of purified PCR product. Printing is carried out by a robot that spots a sample of each gene product onto a number of matrices in a serial operation.

To maximize the reliability and precision with which quantitative differences in the abundance of each RNA species are detected, one directly compares two samples (test and reference) by labeling them with spectrally distinct fluorescent dyes and mixing the two probes for simultaneous hybridization to one array. The relative representation of a gene in the two samples is assayed by measuring the ratio of the (normalized) fluorescent intensities of the two dyes at the target element. Cy3-dUTP and Cy5-dUTP are frequently used as the fluorescent labels. For the comparison of multiple samples, e.g., in time-course experiments, one often uses the same reference sample with each of the test samples.

11.2.2 Oligonucleotide Microarrays

In oligonucleotide microarrays [Fodor *et al.*, 1993, Lipshutz *et al.*, 2000, Harrington *et al.*, 2000], each spot on the array contains a short synthetic oligonucleotide (oligo), typically 20-30 bases long. The oligos are designed based on the knowledge of the DNA (or EST) target sequences, to ensure high-affinity and specificity of each oligo to a particular target gene. Moreover, they should not be near-complementary to other RNAs that may be highly abundant in the sample (e.g., rRNAs, tRNAs, alu-like sequences etc.).

One of the leading approaches to construction of high-density DNA probe arrays employs photolithography and solid-phase DNA synthesis. First synthetic linkers, modified with a photochemically removable protecting groups, are attached to a glass substrate. At each phase, light is directed through a photolithographic mask to specific areas on the surface to produce localized deprotection.

Specific Hydroxyl-protected deoxynucleosides are incubated with the surface, and chemical coupling occurs at those sites that have been illuminated. Current technology allows for approximately 300,000 oligos to be synthesized on a 1.28×1.28 cm array. Key to this approach is the use of multiple distinct oligonucleotides designed to hybridize to different regions of the same RNA. This use of multiple detectors greatly improves signal-to-noise ratio and accuracy of RNA quantitation, and reduces the rate of false-positives and miscalls.

An additional level of redundancy comes from the use of mismatch control probes that are identical to their perfect match partners except for a single base difference in a central position. These probes act as specificity controls: They allow the direct subtraction of both background and cross-hybridization signals, and allow discrimination between 'real' signals and those due to non-specific or semi-specific hybridizations.

11.2.3 Oligonucleotide Fingerprinting

Historically, the Oligonucleotide Fingerprinting (ONF) method preceded the other two [Lennon and Lehrach, Drmanac *et al.*, 1991, Vicentic and Gemmell, 1992, Drmanac and Drmanac, 1994, Drmanac *et al.*, 1996, Meier-Ewert *et al.*, 1995, Milosavljevic *et al.*, 1995]. It was initially proposed in the context of Sequencing By Hybridization, as an alternative to DNA sequencing. While that approach to sequencing is currently not competitive, ONF has found other good applications, including gene expression. It can be used to extract gene expression information about a cDNA library from a specific tissue under analysis, without prior knowledge on the genes involved. Conceptually, it takes the "reverse" approach to that of the oligo microarrays: The target is on the array, and the oligos are "in the air".

The ONF method is based on spotting the cDNAs on high density nylon membranes (about 31,000 different cDNA can be spotted currently in duplicates on one filter [Drmanac *et al.*, 1996]). A large quantity of a short synthetic oligo, typically 7-12 bases long, radioactively labeled, is put in touch with the membrane in proper conditions, and the oligos hybridize to those cDNAs that contain a DNA sequence complementary to that of the oligo. By inspecting the filter one can detect to which of the cDNAs the oligo hybridized. Hence, ideally, the result of such an experiment is one 1/0 bit for each of the cDNAs.

The experiment is repeated with p different oligos, giving rise to a p-long vector for each cDNA spot, indicating which of the (complements of) oligo sequences are contained in each cDNA. This *fingerprint* vector, similar to a bar-code, identifies the cDNA. Thus, distinct spots of cDNAs originating from the same gene should have similar fingerprints. By clustering these fingerprints, one can identify cDNAs originating from the same gene, and the larger that number - the higher the expression level of the corresponding gene. Gene identification can subsequently be obtained by sample sequencing, or by comparison of average cluster fingerprints to a sequence database [Poustka *et al.*, 1999].

Because of the short oligos used, the hybridization information is rather noisy, but this can be compensated by using a longer fingerprint. The method is probably less efficient than the other two methods, which measure abundance directly in a single spot. However, it has the advantage of applicability to species with unknown genomes, which oligo microarrays cannot handle, and it requires relatively lower mRNA quantities than cDNA microarrays.

11.3 Mathematical Formulations and Background

Let $N = \{e_1, \ldots, e_n\}$ be a set of n elements, and let $\mathcal{C} = (C_1, \ldots, C_l)$ be a *partition* of N into subsets. That is, the subsets are disjoint and their union is N. Each subset is called a *cluster*, and $\mathcal{C}$ is called a *clustering solution*, or simply a *clustering*. Two elements e_i and e_j are called *mates with respect to* $\mathcal{C}$ if they are members of the same cluster in $\mathcal{C}$. In the gene expression context, the elements are the genes and we often assume that there exists some correct partition of the genes into "true" clusters. When $\mathcal{C}$ is the true clustering of N, elements that belong to the same true cluster are simply called mates.

The input data for a clustering problem is typically given in one of two forms: (1) *Fingerprint data* - each element is associated with a real-valued vector, called its *fingerprint*, or *pattern*, which contains p measurements on the element, e.g., expression levels of an mRNA at different conditions (cf. [Eisen and Brown, 1999]), or hybridization intensities of a cDNA with different oligos (cf. [Lennon and Lehrach, 1991]). (2) *Similarity data* - pairwise similarity values between elements. These values can be computed from fingerprint data, e.g. by correlation between vectors. Alternatively, the data can represent pairwise dissimilarity, e.g. by computing distances. Fingerprints contain more information than similarity, but the latter is completely generic and can be used to represent the input to clustering in any application. (Note that there is also a practical consideration regarding the presentation: The fingerprint matrix is of order $n \times p$ while the similarity matrix is of order $n \times n$, and in gene expression applications often $n >> p$.)

The goal in a clustering problem is to partition the set of elements N into homogeneous and well-separated clusters. That is, we require that elements from the same cluster will be highly similar to each other, while elements from different clusters will have low similarity to each other. Note that this formulation does not define a single optimization problem: Homogeneity and separation can be defined in various ways, leading to a variety of optimization problems. Note also that even when the homogeneity and separation are precisely defined, those are two objectives which are typically conflicting: The higher the homogeneity - the lower the separation, and vice versa. The lack of a single objective agreed upon by the community is inherent in the clustering problem, and we will

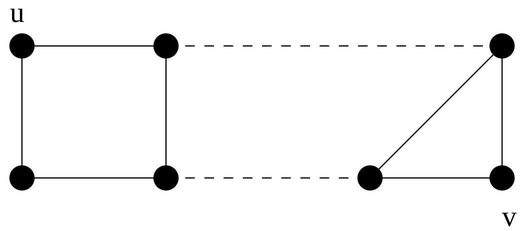

Figure 11.1: A graph and a corresponding minimum weight cut, assuming that all edge weights are 1. Minimum cut edges are denoted by broken lines. The length of the shortest path between u and v is 3, and this is also the diameter of the graph.

return to this point in the sequel.

Clustering problems and algorithms are often represented in graph-theoretic terms. We therefore include some basic definitions on graphs. We refer the readers to [Golumbic, 1980, Even, 1979] for more background and terminology on graphs.

Let $G = (V, E)$ be a weighted graph. We denote the vertex set of G also by $V(G)$. For a subset $R \subseteq V$, *the subgraph induced by* R, denoted G_R, is obtained from G by deleting all vertices not in R and the edges incident on them. That is, $G_R = (R, E_R)$ where $E_R = \{(i,j) \in E | i,j \in R\}$. For a vertex $v \in V$, define the *weight* of v to be the sum of weights of the edges incident on v. A *cut* C in G is a subset of its edges, whose removal disconnects G. The *weight* of C is the sum of the weights of its edges. A *minimum weight cut* is a cut in G with minimum weight. In case of non-negative edge weights, a minimum weight cut C partitions the vertices of G into two disjoint non-empty subsets $A, B \subset V$, $A \cup B = V$, such that $E \cap \{(u,v) : u \in A, v \in B\} = C$. For a pair of vertices $u, v \in V$, the *distance* between u and v is the length of the shortest path which connects them. The *diameter* of G is the maximum distance between a pair of vertices in G. For an example of these definitions see Figure 11.1.

For a set of elements $K \subseteq N$, we define the *fingerprint* or *centroid* of K to be the mean vector of the fingerprints of the members of K. For two fingerprints x and y we denote their similarity by $S(x, y)$ and their dissimilarity by $d(x, y)$. A *similarity graph* is a weighted graph in which vertices correspond to elements and edge weights are derived from the similarity values between the corresponding elements. Hence, the similarity graph is just another representation of the similarity matrix.

An alternative formulation of the clustering problem is hierarchical: Rather than asking for a single partition of the elements, one seeks an iterated partition: A *dendogram* is a rooted weighted tree, with leaves corresponding to elements. Each edge defines the cluster of elements contained in the subtree below that edge. The edge's weight (or length) reflects the dissimilarity between that cluster and the remaining elements. In this formulation the clustering solution is the dendogram, and each non-singleton cluster, corresponding to a rooted subtree, is split into subclusters. The determination of disjoint clusters is left to the judgment of the user. Typically, one tends to consider as genuine clusters elements of a subtree just below a connecting edge of high weight.

Irrespective of the representation of the clustering problem input, judicious preprocessing of the raw data is key to meaningful clustering. This preprocessing is application dependent and must be chosen in view of the expression technology used and the biological questions asked. The goal of the preprocessing is to normalize the data and calculate the pairwise element (dis)similarity, if applicable. Common procedures for normalizing fingerprint data include transforming each fingerprint to have mean zero and variance one, a fixed norm or a fixed maximum entry. Statistically based methods for data normalization have also been developed recently (cf. [Kerr *et al.*, 2000]).

11.4 Algorithms

Several algorithmic techniques were previously used in clustering gene expression data, including hierarchical clustering [Eisen *et al.*, 1998], self organizing maps [Tamayo *et al.*, 1999], and graph theoretic approaches [Hartuv *et al.*, 2000, Ben-Dor *et al.*, 1999, Sharan and Shamir, 2000b]. We describe these approaches in the sequel. For other approaches to clustering expression patterns, see [Milosavljevic *et al.*, 1995, Alon *et al.*, 1999, Getz *et al.*, 2000b, Heyer *et al.*, 1999]. Much more information and background on clustering is available, cf. [Hartigan, 1975, Everitt, 1993, Mirkin, 1996, Hansen and Jaumard, 1997].

Several algorithms for clustering were developed by first designing a "clean" algorithm that has proven properties, either in terms of time complexity, or in terms of (deterministic or probabilistic) solution quality. Then a more efficient yet heuristic algorithm is developed based on the same idea. We shall describe here the heuristics used in practice, but refer also briefly to the properties of the theoretical algorithm that motivated them.

11.4.1 Hierarchical Clustering

Hierarchical clustering solutions are typically represented by a dendogram. Algorithms for generating such solutions often work either in a top-down manner, by repeatedly partitioning the set of elements, or in a bottom-up fashion. We shall describe here the latter. Such *agglomerative* hierarchical clustering algorithms are among the oldest and most popular clustering methods [Cormack, 1971]. They proceed from an initial partition into singleton clusters by successive merging of clusters until all elements belong to the same cluster. Each merging step corresponds to joining two clusters. The general scheme due to Lance and Williams [Lance and Williams, 1967] is presented in Figure 11.2. It is assumed that $D = (d_{ij})$ is the input dissimilarity matrix.

1. Find a minimal entry $d_{i^*j^*}$ in D, and merge clusters i^* and j^*.

2. Modify D by deleting rows and columns i, j and adding a new row and column $i^* \cup j^*$, with their dissimilarities defined by:

$$d_{k,i^*\cup j^*} = d_{i^*\cup j^*,k} = \alpha_{i^*} d_{ki^*} + \alpha_{j^*} d_{kj^*} + \gamma |d_{ki^*} - d_{kj^*}|$$

3. If there is more than one cluster, then go to Step 1.

Figure 11.2: The agglomerative hierarchical clustering scheme.

Common variants of this scheme are the following:

- *Single-linkage*: $d_{k,i^*\cup j^*} = \min\{d_{ki^*}, d_{kj^*}\}$. Here $\alpha_{i^*} = \alpha_{j^*} = 1/2$ and $\gamma = -1/2$.

- *Complete-linkage*: $d_{k,i^*\cup j^*} = \max\{d_{ki^*}, d_{kj^*}\}$. Here $\alpha_{i^*} = \alpha_{j^*} = 1/2$ and $\gamma = 1/2$.

- *Average-linkage*: $d_{k,i^*\cup j^*} = n_{i^*} d_{ki^*}/(n_{i^*} + n_{j^*}) + n_{j^*} d_{kj^*}/(n_{i^*} + n_{j^*})$, where n_i denotes the number of elements in cluster i. Here $\alpha_{i^*} = \frac{n_{i^*}}{n_{i^*}+n_{j^*}}$, $\alpha_{j^*} = \frac{n_{j^*}}{n_{i^*}+n_{j^*}}$, and $\gamma = 0$.

Eisen et al. [Eisen *et al.*, 1998] developed a clustering software package based on the average-linkage hierarchical clustering algorithm. The software package is called Cluster, and the accompanying visualization program is called TreeView. Both programs are available at `http://rana.Stanford.EDU/software/`. The gene similarity metric used is a form of correlation coefficient. The algorithm iteratively merges elements whose similarity value is the highest, as explained above. The output of the algorithm is a dendogram and an ordered fingerprint matrix. The rows in the matrix are permuted based on the dendogram, so that groups of genes with similar expression patterns are adjacent. The ordered matrix is represented graphically by coloring each cell according to its content. Cells with log ratios of 0 are colored black, increasingly positive log ratios with reds of increasing intensity, and increasingly negative log ratios with greens of increasing intensity.

11.4.2 K-Means

K-means [MacQueen, 1965, Ball and Hall, 1967] is another classical clustering algorithm. It assumes that the number of clusters k is known, and aims to minimize the distances between elements and the centroids of their assigned clusters. Let M be the $n \times m$ fingerprint matrix. For a partition P of the elements in $\{1, \ldots, n\}$ denote by $P(i)$ the cluster assigned to i, and by $c(j)$ the centroid of cluster j. Let $d(v_1, v_2)$ denote the Euclidean distance between the fingerprint vectors v_1 and v_2. K-means tries to find a partition P for which the error-function $E_P = \sum_{i=1}^{n} d(i, c(P(i)))$ is minimum.

Each iteration of K-means modifies the current partition by checking all possible modifications of the solution in which one element is moved to another cluster, and making a switch that reduces the error function. Figure 11.3 describes the most basic scheme. A more efficient variant moves in one iteration all elements which would benefit from a move: For each i simultaneously, if $min_j E_P^{ij} < E_P$, move i to the cluster j minimizing E_P^{ij}. This algorithm is very easy to implement and is used in many applications.

1. Start with an arbitrary partition P of N into k clusters.

2. For each element i and cluster $j \neq P(i)$ let E_P^{ij} be the cost of a solution in which i is moved to cluster j. **If** $E_P^{i^*j^*} = min_{ij} E_P^{ij} < E_P$ **then** move i^* to cluster j^* and repeat Step 2. Otherwise **halt**.

Figure 11.3: The K-means algorithm.

Another heuristic inspired by K-means was developed by Herwig et al. [Herwig *et al.*, 1999] to cluster cDNA oligo-fingerprints. Unlike the regular K-means algorithm, this algorithm does not require a prespecified number of clusters. Instead, it uses two parameters: γ is the maximal admissible similarity of two distinct clusters, and ρ is the maximal admissible similarity between an element and a cluster different from its own cluster. (Similarity to a cluster is similarity to its centroid.) Elements are handled one at a time, added to sufficiently close clusters, or otherwise, forming a new cluster. Whenever centroids become too close, their clusters are merged. Unlike the K-means algorithm, an element may be tentatively assigned to more than one cluster, and thus influence the location of several centroids to which it is sufficiently close. The algorithm is shown in figure 11.4. Here $S(i, C)$ is the similarity between element i and cluster C.

1. Start with a set of sufficiently different elements as clusters.
2. **For** each remaining element i **do:**
   - For each cluster C s.t. $S(i, C) \geq \rho$ **do:**
     - add i to C.
     - **While** there exists a cluster C' s.t. $S(C, C') > \gamma$, merge C' into C.
   - **If** i was not added to any cluster **then** form a new cluster $\{i\}$.
3. Assign each element to the cluster to which it is most similar.

Figure 11.4: The algorithm of Herwig et al.

11.4.3 HCS and CLICK

The HCS [Hartuv *et al.*, 2000, Hartuv and Shamir, 2000] and CLICK [Sharan and Shamir, 2000a, Sharan anc algorithms use a similar graph theoretic approach to clustering: The input data is represented as a similarity graph. The algorithm recursively partitions the current set of elements into two subsets.

Before a partition, the algorithm considers the subgraph induced by the current subset of elements. If the subgraph satisfies a stopping criterion then it is declared a *kernel*. Otherwise, a minimum weight cut is computed in that subgraph, and the set is split into the two subsets separated by that cut. The output is a list of kernels which serve as a basis for the eventual clusters. This scheme is detailed in Figure 11.5.

Form-Kernels(G):

If $V(G) = \{v\}$ **then** move v to the singleton set.

Else if G is a kernel **then** output $V(G)$.

Else

  $(H, \bar{H}) \leftarrow$MinWeightCut(G).

  Form-Kernels(H).

  Form-Kernels$(\bar{H})$.

Figure 11.5: The basic scheme of HCS and CLICK. Procedure MinWeightCut(G) computes a minimum weight cut of G and returns a partition of G into two subgraphs H and $\bar{H}$ according to this cut.

HCS and CLICK differ in the similarity graph they construct, their stopping criteria and the postprocessing of the kernels. We describe each of the algorithms below.

HCS

The HCS algorithm [Hartuv *et al.*, 2000, Hartuv and Shamir, 1999] builds from the input data an *unweighted* similarity graph G (each edge has weight 1 and each non-edge has weight 0) in which there is an edge between two vertices if and only if the similarity between their corresponding elements exceeds a predefined threshold.

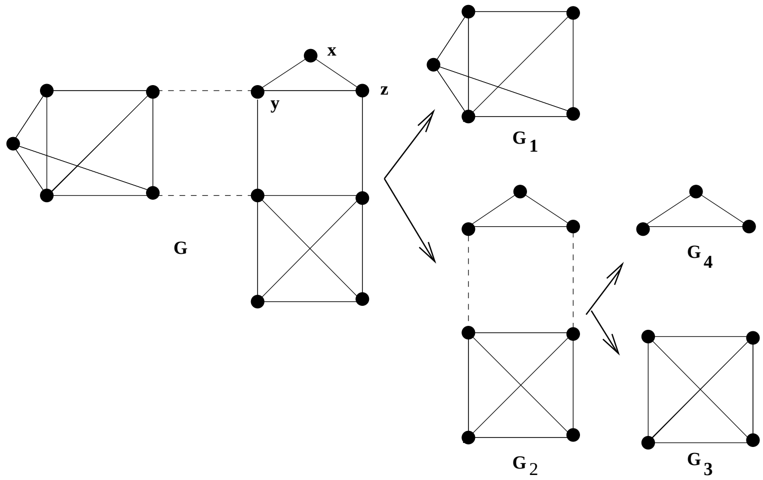

Figure 11.6: An example of applying the HCS algorithm to a graph. Minimum cut edges are denoted by broken lines.

The following notion is key to the algorithm: A *highly connected subgraph* (HCS) is an induced subgraph H of G, whose minimum cut value exceeds $|V(H)|/2$. That is, H remains connected if any $\lfloor |V(H)|/2 \rfloor$ of its edges are removed. The algorithm identifies highly connected subgraphs as kernels. Figure 11.6 demonstrates an application of the algorithm.

The HCS algorithm possesses several good properties for clustering [Hartuv and Shamir, 1999]: The diameter of each cluster it produces is at most two, and each cluster is at least half as dense as a clique. Both properties indicate strong cluster homogeneity. Inter-cluster separation is harder to prove, but it is argued that if errors are random, any non-trivial set split by the algorithm is unlikely to have diameter two unless the involved sets are small.

To improve separation in practice, several heuristics are used to expand the kernels and speed up the algorithm:

Iterated-HCS: When the minimum cut value is obtained by several distinct cuts, the HCS algorithm chooses one arbitrarily. This process may break small clusters into singletons. (For example, a different choice of minimum cuts by the algorithm for the graph in Figure 11.6 may split x from G_2 and eventually find the clusters G_1 and G_3, leaving x, y, z as singletons.) To overcome this, several (1-5) HCS iterations are carried out until no new cluster is found.

Singletons Adoption: Singletons can be "adopted" by clusters: For each singleton element x we compute the number of neighbors it has in each cluster and in the singletons set $\mathcal{S}$. If the maximum number of neighbors is sufficiently large, and is obtained by one of the clusters (rather than by $\mathcal{S}$), then x is added to that cluster. The process is repeated several times.

Removing Low Degree Vertices: When the similarity graph contains vertices with low degrees, one iteration of the minimum cut algorithm may simply separate a low degree vertex from the rest of the graph. This is computationally very expensive, not informative in terms of the clustering, and may happen many times if the graph is large. Removing low degree vertices from G eliminates such iterations, and significantly reduces the running time. The process is repeated with several thresholds on the degree. This simple procedure is very powerful for large problems.

CLICK

The CLICK algorithm (CLuster Identification via Connectivity Kernels) [Sharan and Shamir, 2000b], available at `http://www.math.tau.ac.il/~rshamir/click/click.html`, builds on a statistical model. The model gives probabilistic meaning to edge weights in the similarity graph and to the stopping criterion. The key probabilistic assumption of CLICK is that pairwise similarity values between elements are normally distributed: Similarity values between mates are normally distributed with mean μ_T and variance σ_T^2, and similarity values between non-mates are normally distributed with mean μ_F and variance σ_F^2, where $\mu_T > \mu_F$. This situation often holds on real data, and can be asymptotically justified [Sharan and Shamir, 2000b].

The algorithm uses the values of μ_T, μ_F, σ_T and σ_F, as well as the probability p_{mates} that two randomly chosen elements are mates. These parameters can be computed directly from a known solution on a subset of the elements (which is often available in ONF experiments [Poustka *et al.*, 1999]), or estimated using the EM algorithm, assuming the above probabilistic model for similarity values (see, e.g., [Mirkin, 1996, Sec. 3.2.7]).

Let $S = (S_{ij})$ be the input similarity matrix. Form a weighted similarity graph $G = (V, E)$, in which the weight w_{ij} of the edge (i, j) reflects the probability that i and j are mates, and is derived from the normal density function $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ and Bayes Theorem:

$$w_{ij} = \ln \frac{Prob(i, j \text{ are mates } |S_{ij})}{Prob(i, j \text{ are non-mates } |S_{ij})} = \ln \frac{p_{mates}\sigma_F}{(1 - p_{mates})\sigma_T} + \frac{(S_{ij} - \mu_F)^2}{2\sigma_F^2} - \frac{(S_{ij} - \mu_T)^2}{2\sigma_T^2}$$

CLICK uses the same basic scheme as HCS (see Figure 11.5) to form kernels. The current subgraph is determined to be a kernel if the value of a minimum cut in it is positive. This is the case if and only if for every cut C in the current subgraph, the probability that it contains only edges between mates exceeds the probability that C contains only edges between non-mates.

The actual implementation omits from the graph all edges with values below some predefined non-negative threshold, computes the minimum cut in that simplified graph, and corrects the solution value for the missing edges.

CLICK first produces kernels which form the basis of the eventual clusters. Subsequent processing includes singleton adoption, recursive clustering process on the set of remaining singletons, and an iterative *merging step*. The singletons adoption step is based on computing similarities between singletons' and clusters' fingerprints. The merging step iteratively merges two kernels whose fingerprint similarity is the highest, provided that this similarity exceeds a predefined threshold. The use of the fingerprints (rather than average similarity values) here is very powerful. Similar ideas were employed in [Milosavljevic *et al.*, 1995, Hartuv *et al.*, 2000]. Finally, a last singleton adoption step is performed. The full algorithm is detailed in Figure 11.7.

$R \leftarrow N$.

While some change occurs **do**:

    Form-Kernels(G_R).

    Adoption($\mathcal{L}, R$).

Merge($\mathcal{L}$).

Adoption($\mathcal{L}, R$).

Figure 11.7: The CLICK algorithm. N is the complete set of elements (all the vertices in the similarity graph). Throughout the algorithm, $\mathcal{L}$ is the current list of kernels and R is the set of singletons. G_R is the subgraph of G induced by the vertex set R. Adoption($\mathcal{L}, R$) performs the iterative singletons adoption procedure. Merge($\mathcal{L}$) is the iterative merging procedure.

In order to reduce the running time of CLICK on very big instances a screening heuristic is applied, which is similar to the low-degree heuristic of the HCS algorithm. Low weight vertices are screened from large components in the following manner: First, the average vertex weight W of the component is computed, and is multiplied by a factor which is proportional to the logarithm of the size of the component. Denote the resulting threshold by W^*. Then vertices whose weight is below W^* are removed repeatedly, each time updating the weight of the remaining vertices, until the updated weight of every (remaining) vertex is greater than W^*. The removed vertices are marked as singletons and handled at a later stage.

11.4.4 CAST

Ben-Dor et al. [Ben-Dor *et al.*, 1999] developed a polynomial algorithm for finding the true clustering with high probability, under the following stochastic model of the data: The underlying correct cluster structure is represented by a graph that is a disjoint union of cliques, and errors are subsequently introduced in the graph by independently removing and adding edges between pairs of vertices with probability α. If all clusters are of size at least cn, for some constant $c > 0$, the algorithm solves the problem to a desired accuracy with high probability.

CAST uses as input the similarity matrix S. The *affinity* of an element v to a putative cluster C is $a(v) = \sum_{i \in C} S(i, v)$. The polynomial algorithm motivated the use of affinity to develop a faster heuristic called CAST (Clustering Affinity Search Technique) [Ben-Dor *et al.*, 1999], which is implemented in the package BIOCLUST. The algorithm uses a single parameter t. Clusters are generated one by one. The next cluster is started with a single element, and elements are added or removed from the cluster if their relative affinity is larger or lower than t, respectively, until the process stabilizes. The algorithm is shown in Figure 11.8.

While there are unclustered elements **do**:

> Pick an unclustered element to start a new cluster C.
>
> Repeat ADD and REMOVE until no changes occur:
>
> > ADD: add an unclustered element v with maximum affinity to C if $a(v) > t|C|$.
> >
> > REMOVE: remove an element u from C with minimum affinity if $a(u) \leq t|C|$.
>
> Add C to the list of final clusters.

Figure 11.8: The CAST algorithm.

An additional heuristic is employed at the end of the algorithm: A series of moving steps aims at a clustering in which the affinity of every element is higher to its assigned cluster than to any other cluster.

11.4.5 Self Organizing Maps

The self organizing maps were developed by Kohonen [Kohonen, 1997] as a method for fitting a number of ordered discrete reference vectors to the distribution of vectorial input samples. A self organizing map (SOM) assumes that the number of clusters is known. Those clusters are organized as a set of nodes in a hypothetical "elastic network", with a simple neighborhood structure on the nodes, e.g., a two-dimensional $k \times l$ grid. Each of these nodes is associated with a reference vector in $\mathcal{R}^n$. In the process of running the algorithm, the input vectors direct the movement of the reference vectors, so that an organization of the input vectors over the network emerges. In the following we describe the SOM algorithm in the Euclidean space, and use $d(x, y)$ to denote the distance between points x and y.

The SOM process is iterative. Denote by $f_i(n)$ the position of node n at the i-th iteration. The initial positioning f_1 is random. The algorithm iteratively selects a random data point p, identifies the nearest reference node n_p, and updates the reference nodes according to a learning function $\tau(\cdot)$, where nodes closer to n_p are updated more. The updates also decrease with the iteration number. The algorithm is described in Figure 11.9. The function $\tau(\cdot)$ represents the "stiffness" of the network. The intuition for this learning process is that the nodes that are close enough to p will "activate" each other to learn something from p.

Two popular choices for the learning function are:

Arbitrarily set the reference vectors $f_1(v) \in R^n$ for each node v.

For $i = 1$ until no node location is changed by more than ϵ **do**:

Randomly pick a data point p.

Compute the node n_p with reference vector $f(n_p)$ closest to p.

Update all reference vectors: $f_{i+1}(n) = f_i(n) + \tau(n, n_p, i)[p - f_i(n)]$.

Assign each data point to the cluster with the closest reference vector.

Figure 11.9: The Self Organizing Map algorithm. The learning function $\tau(\cdot)$ monotonically decreases with $d(n, n_p)$ and with the iteration number i.

- Neighborhood function: For each node n we denote by $N_i(n)$ the set of nodes within some distance from n. (These distances are with respect to the neighborhood structure in the network.) We then define $\tau(n, n_p, i) = 0$ if $n \notin N(n_p)$ and $\tau(n, n_p, i) = \alpha(i)$ otherwise. $\alpha(i)$ is called the learning-rate and decreases with i.

- Gaussian function: $\tau(n, n_p, i) = \alpha(i) \cdot exp(-\frac{d(n,n_p)^2}{2\sigma^2(i)})$, where $\alpha(i)$ and $\sigma(i)$ decrease with i.

For much more on self organizing maps the reader is referred to [Kohonen, 1997].

Tamayo et al. [Tamayo *et al.*, 1999] devised a gene expression clustering software, GeneCluster, which uses the SOM algorithm. The software is available at `http://waldo.wi.mit.edu/MPR/`. In their implementation they incorporated a neighborhood learning function, for which $\alpha(i) = 0.02T/(T+100i)$, where T is the maximum number of iterations; and $N_i(n_p)$ contains all nodes whose distance to n_p is at most $\rho(i)$, where $\rho(i)$ decreases linearly with i, $\rho(0) = 3$.

GeneCluster accepts an input file of expression levels together with a two dimensional grid geometry for the nodes. The number of grid points is the prescribed number of clusters. The resulting clusters are visualized by presenting for each cluster its average expression pattern with error-bars. Clusters are presented in their grid order, as clusters of close nodes tend to be similar.

Another implementation of SOM for clustering gene expression profiles was developed by [Toronen *et al.*, 1

11.5 Assessment of solutions

A key question in the design and analysis of clustering techniques is how to evaluate solutions. We present in this section figures of merit for measuring the quality of a clustering solution. Different measures are applicable in different situations, depending on whether a partial true solution is known or not, and whether the input is fingerprint or similarity data. We describe below some of the applicable measures in each case. For other possible figures of merit we refer the reader to [Everitt, 1993, Hansen and Jaumard, 1997, Yeung *et al.*, 2000].

11.5.1 Assessment given the true solution

Suppose at first that the true solution is known, and we wish to compare it to a suggested solution. Any clustering solution can be represented by a binary $n \times n$ matrix C, in which $C_{ij} = 1$ if and only if i and j belong to the same cluster in that solution. Let T and C be the matrices for the true solution and the suggested solution, respectively. Let n_{kl}, $k, l = 0, 1$, denote the number of pairs (i, j) $(i \neq j)$ for which $T_{ij} = k$ and $C_{ij} = l$. Thus, n_{11} is the number of true mates which are also mates in the suggested solution, n_{00} is the number of non-mates correctly identified as such, while n_{01} and n_{10} count the disagreements between the true solution and the suggested one.

The *Minkowski measure* (see, e.g., [Sokal, 1977]) is defined as

$$\sqrt{\frac{n_{01} + n_{10}}{n_{11} + n_{10}}}$$

Hence, it measures the proportion of disagreements to the total number of mates in the true solution. A perfect solution has score zero, and the lower the score - the better the solution. The *Jaccard coefficient* (see, e.g., [Everitt, 1993]) is the ratio

$$\frac{n_{11}}{n_{11} + n_{10} + n_{01}}$$

It is the proportion of correctly identified mates to the sum of the correctly identified mates plus the total number of disagreements. Hence, a perfect solution has score one, and the higher the score - the better the solution. This measure is a lower bound for both sensitivity ($\frac{n_{11}}{n_{11}+n_{10}}$) and specificity ($\frac{n_{11}}{n_{11}+n_{01}}$) of the solution.

Note, that both measures do not (directly) involve the term n_{00}, since solution matrices tend to be sparse and this term would dominate the other three in good and bad solutions alike. When the true solution is known only for a subset $N^* \subset N$, the Minkowski and Jaccard measures can be computed on the sub-matrices corresponding to N^*. In some cases, e.g., for cDNA oligo-fingerprint data, we have the additional information that no element of N^* has a mate in $N \setminus N^*$. In such a case, the Minkowski and Jaccard measures are evaluated using all the pairs $\{(i,j) : i \in N^*, j \in N \cup N^*, i \neq j\}$.

11.5.2 Assessment when the true solution is unknown

When the true solution is not known, we evaluate the quality of a suggested solution by computing two figures of merit that measure its homogeneity and separation. For fingerprint data, homogeneity is evaluated by the average similarity between the fingerprint of an element and that of its cluster. Precisely, if $cl(u)$ is the cluster of u, $F(X)$ and $F(u)$ are the fingerprints of a cluster X and an element u, respectively, then

$$H_{Ave} = \frac{1}{|N|} \sum_{u \in N} S(F(u), F(cl(u)))$$

Separation is evaluated by the weighted average similarity between cluster fingerprints. That is, if the clusters are $X_1, \ldots X_t$, then

$$S_{Ave} = \frac{1}{\sum_{i\neq j}|X_i||X_j|}\sum_{i\neq j}|X_i||X_j|S(F(X_i), F(X_j))$$

Related measures that take a worst case instead of average case approach are minimum homogeneity: $H_{Min} = \min_{u\in N} S(F(u), F(cl(u)))$, and minimum separation: $S_{Max} = \max_{i\neq j} S(F(X_i), F(X_j))$. Hence, a solution improves if H_{Ave} or H_{Min} increase, and if S_{Ave} or S_{Max} decrease. In computing all the above measures singletons are considered as additional one-member clusters.

11.6 A Case Study

In order to highlight the characteristics of each of the methods described above, we applied them to a yeast cell-cycle dataset containing the gene expression levels of yeast ORFs over 79 conditions. This dataset is available at `http://cellcycle-www.stanford.edu`.

The original dataset [Spellman *et al.*, 1998] contains samples from yeast cultures synchronized by four independent methods: α factor arrest (samples taken every 7 minutes for 119 minutes), arrest of a cdc15 temperature sensitive mutant (samples taken every 10 minutes for 290 minutes), arrest of a cdc28 temperature sensitive mutant (this part of the data is from [Cho *et al.*, 1998]; samples taken every 10 minutes for 160 minutes), and elutriation (samples taken every 30 minutes for 6.5 hours). It also contains separate experiments in which G1 cyclin Cln3p or B-type cyclin Clb2p were induced.

Spellman et al. identified in this data 800 genes that are cell-cycle regulated [Spellman *et al.*, 1998]. The dataset that we used contains the expression levels of 698 out of those 800 genes, which have no missing entries, over the 72 conditions. that cover the α factor, cdc28, cdc15, and elutriation experiments. (As in [Tamayo *et al.*, 1999], the 90 minutes datapoint was omitted from the cdc15 experiment.) Each row of the 698×72 matrix was normalized to have mean 0 and variance 1. (Note that by normalizing the variance different gene amplitudes are deemphasized and periodicity is more prominent.)

Based on the analysis conducted by Spellman et al., we expect to find in the data five main clusters: G1-peaking genes, S-peaking genes, G2-peaking genes, M-peaking genes, and M/G1-peaking genes. Each of these was shown to contain biologically meaningful sub-clusters.

The 698×72 dataset was clustered using five of the methods described above: K-means, SOM, CAST, hierarchical, and CLICK. The similarity measure used was Pearson correlation coefficient. The authors of each of the programs were given the dataset and asked to provide a clustering solution. The identity of the dataset was not described and genes were permuted in an attempt to perform a "blind" test. (Yet, anyone familiar with the gene expression literature could have identified the nature of the data.) The authors were told that the average homogeneity and average separation would be used to evaluate the quality of the solutions.

We present below the results for each of the methods. To allow the reader an impression of the results, we added for each of the clusterings (except the hierarchical one which does not produce a hard partition of the elements), a reference figure prepared using MATLAB. This figure depicts the average pattern of the clusters along with error-bars for the first 18 datapoints, which correspond to the α factor experiment. We have chosen not to show full expression patterns over all the 72 conditions, as these are much harder to interpret visually. Over the first 18 datapoints one expects to view periodic behavior, with a distinct, typical pattern in each cluster. We also omitted from these figures small clusters with less than 4 members. As most programs output a variation of this figure, we have chosen to include the characteristic graphical output only for the programs Cluster and CAST.

The following table summarizes the solutions produced by each program (except for Cluster), and

their homogeneity and separation parameters. The so-called 'True' clustering, reported in [Spellman *et al.*, 199
is that obtained manually by inspecting the expression patterns and comparing to the literature. The solution produced by CLICK contains 67 unclustered singletons.

Program	#Clusters	Homogeneity		Separation	
		H_{Ave}	H_{Min}	S_{Ave}	S_{Max}
K-Means	49	0.629	-0.339	0.086	0.911
CAST	5	0.6	0.037	-0.146	0.322
GeneCluster	6	0.617	0.067	-0.073	0.584
CLICK	6	0.656	0.097	-0.098	0.546
'True'	5	0.572	-0.322	-0.133	0.73

Table 11.1: A summary of the clustering solutions and their figures of merit.

The reference figures for each of the solutions are given in Figures 11.10 to 11.14. The output of CAST is shown in Figure 11.15. It depicts the similarity matrix before and after ordering its rows and columns based on the clustering. The output of Cluster is shown in Figure 11.16. It includes a dendogram and a graphical representation of the ordered fingerprint matrix. (Experiments are also clustered and the solution is represented as a second dendogram on the same figure.) Figure 11.17 depicts the values of each solution on a plot of the homogeneity vs. separation.

11.7 Concluding Remarks

Clustering remains, to certain extent, an art. There are no universal, agreed-upon criteria for evaluating solutions, and there is no ultimate algorithm: The clustering problem is so general, covering diverse

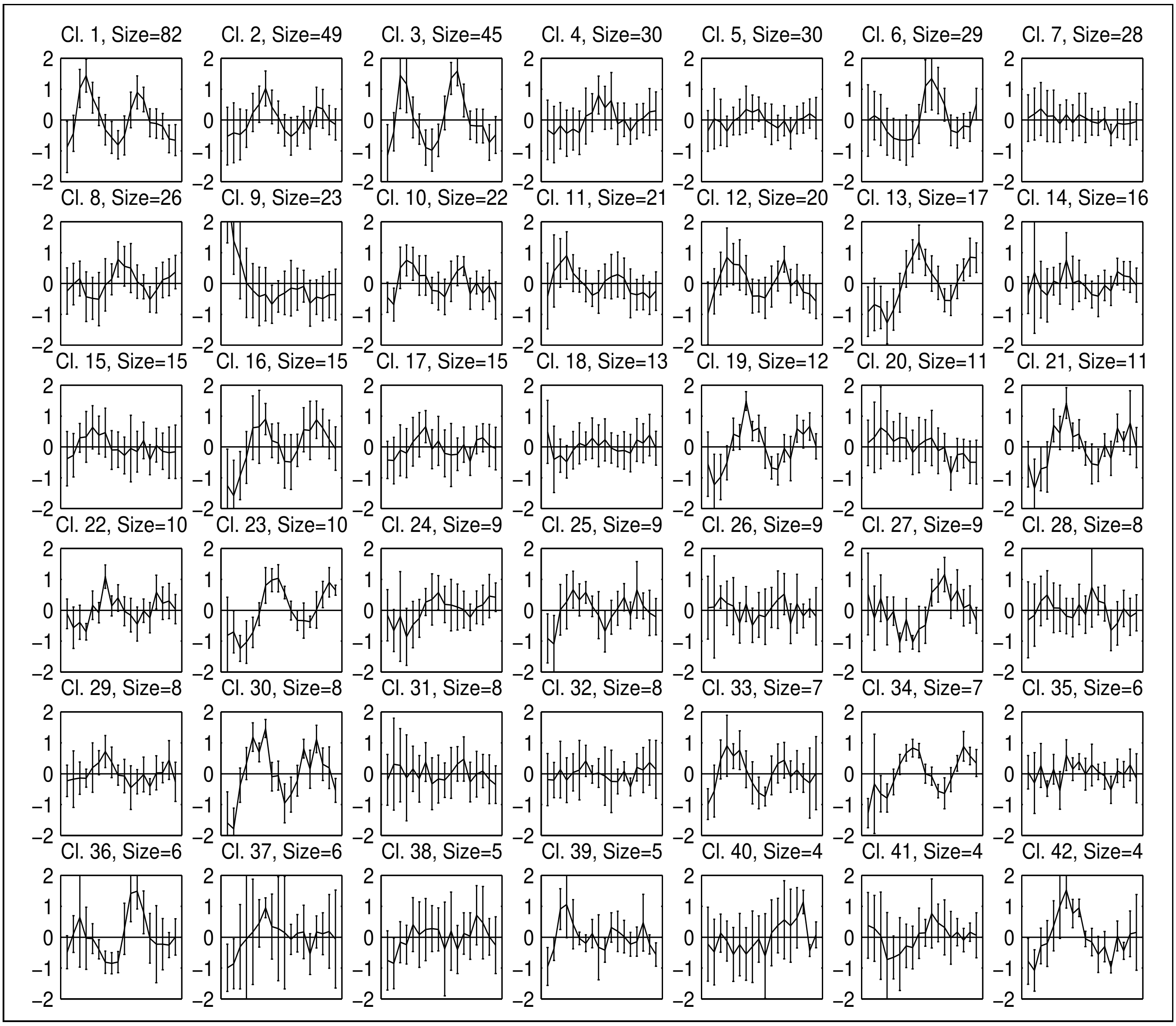

Figure 11.10: The clustering produced by the K-means algorithm of Herwig et al. x axis: time points 1-18 for the α factor experiment. y axis: normalized expression levels. The solid line in each sub-figure plots the average pattern for that cluster. Error bars display the measured standard deviation. The cluster size is printed above each plot.

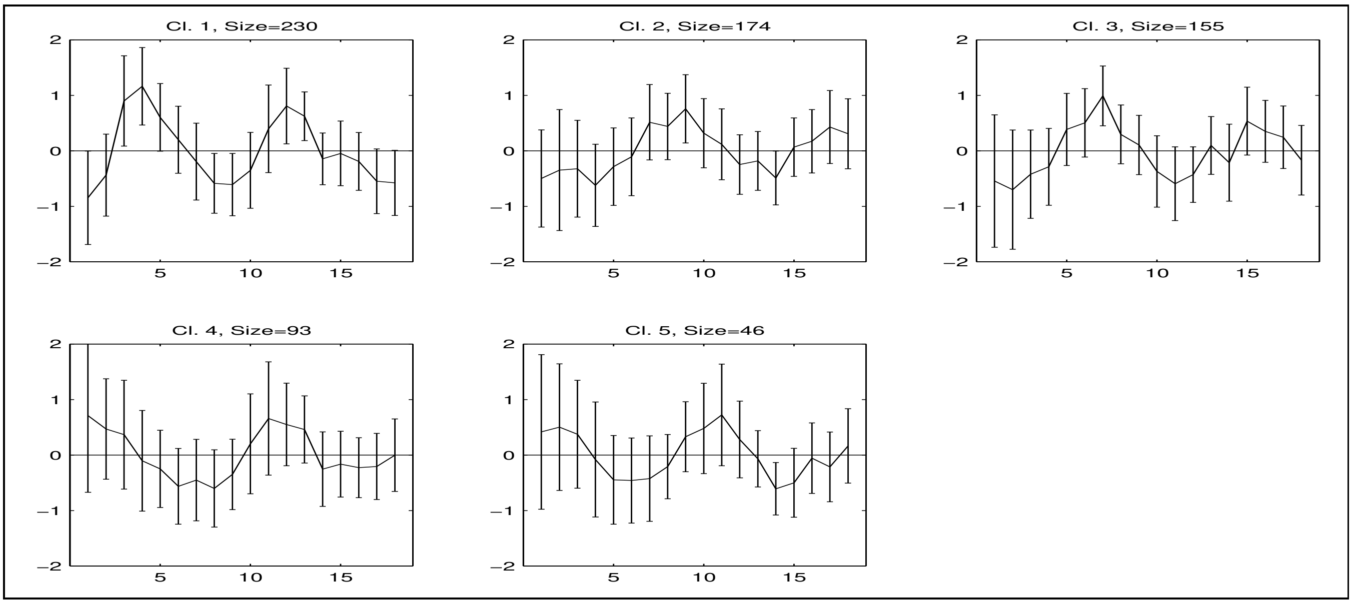

Figure 11.11: The clustering produced by the CAST algorithm of Ben-Dor et al.

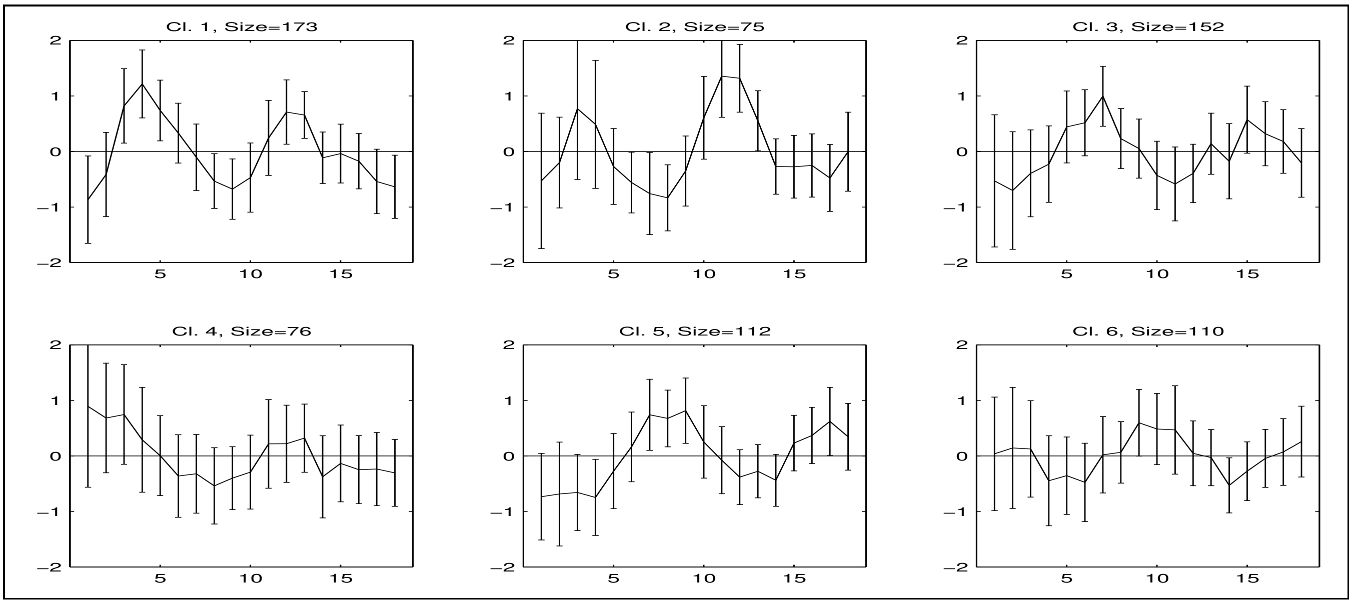

Figure 11.12: The clustering produced by the GeneCluster algorithm of Tamayo et al.

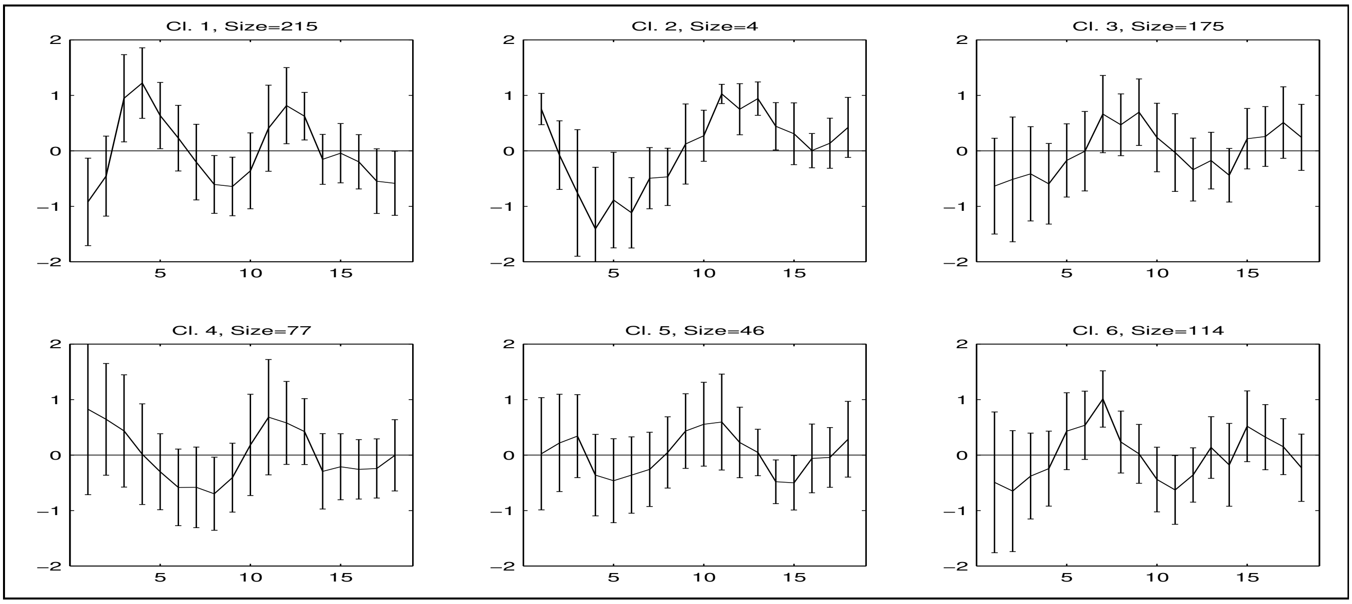

Figure 11.13: The clustering produced by the CLICK algorithm.

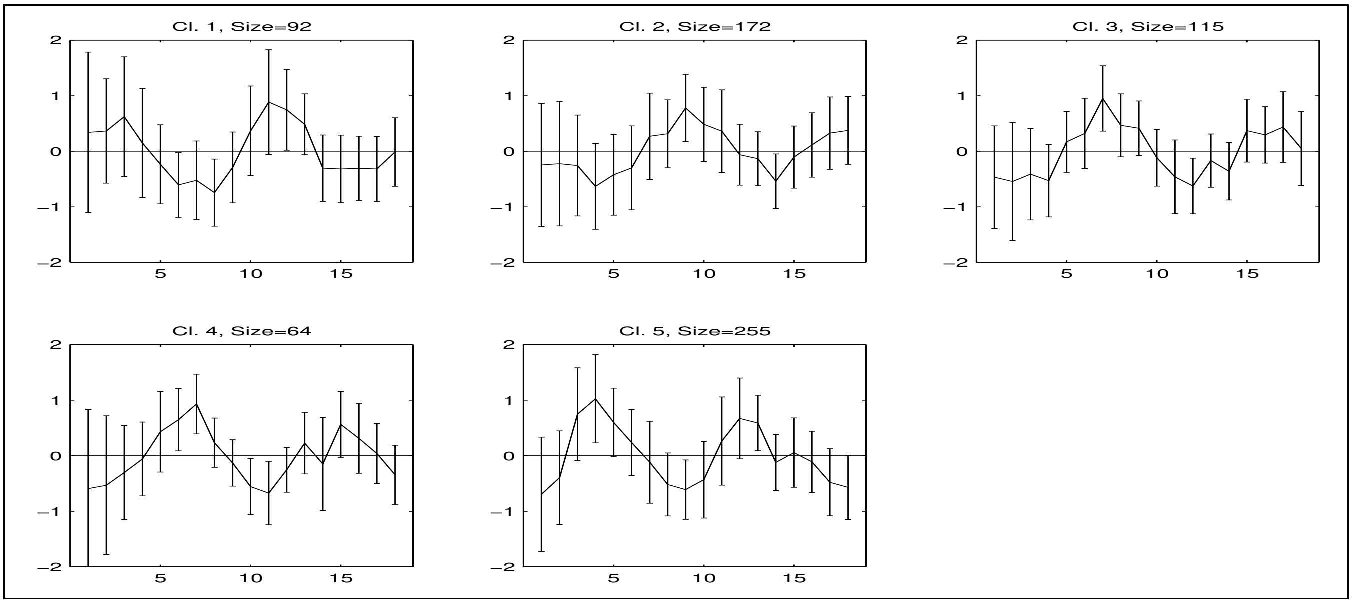

Figure 11.14: The 'True' clustering of Spellman et al.

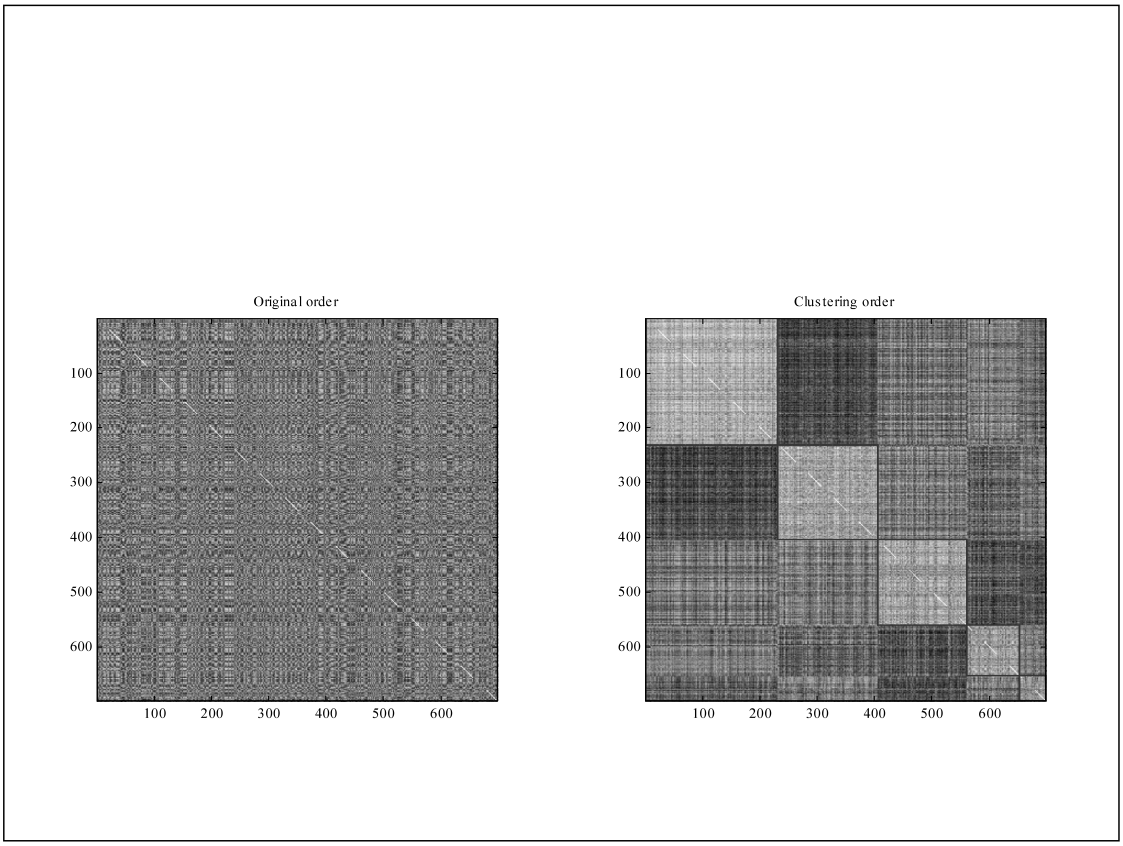

Figure 11.15: Representation of the solution produced by CAST. Left: the original similarity matrix. Right: the same matrix reordered according to the clustering. Gray level is inversely proportional to similarity. Genes belonging to the same cluster appear contiguously.

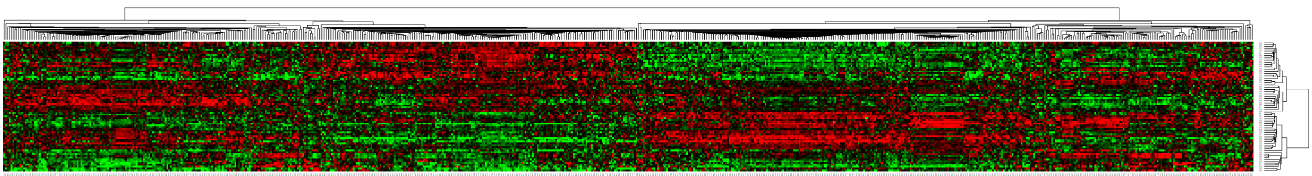

Figure 11.16: The output of Cluster. Actual output is in color, where red denotes increase vs. the reference level, and green denotes decrease. The gene clustering dendogram is on the top, and the experiment clustering dendogram is on the right.

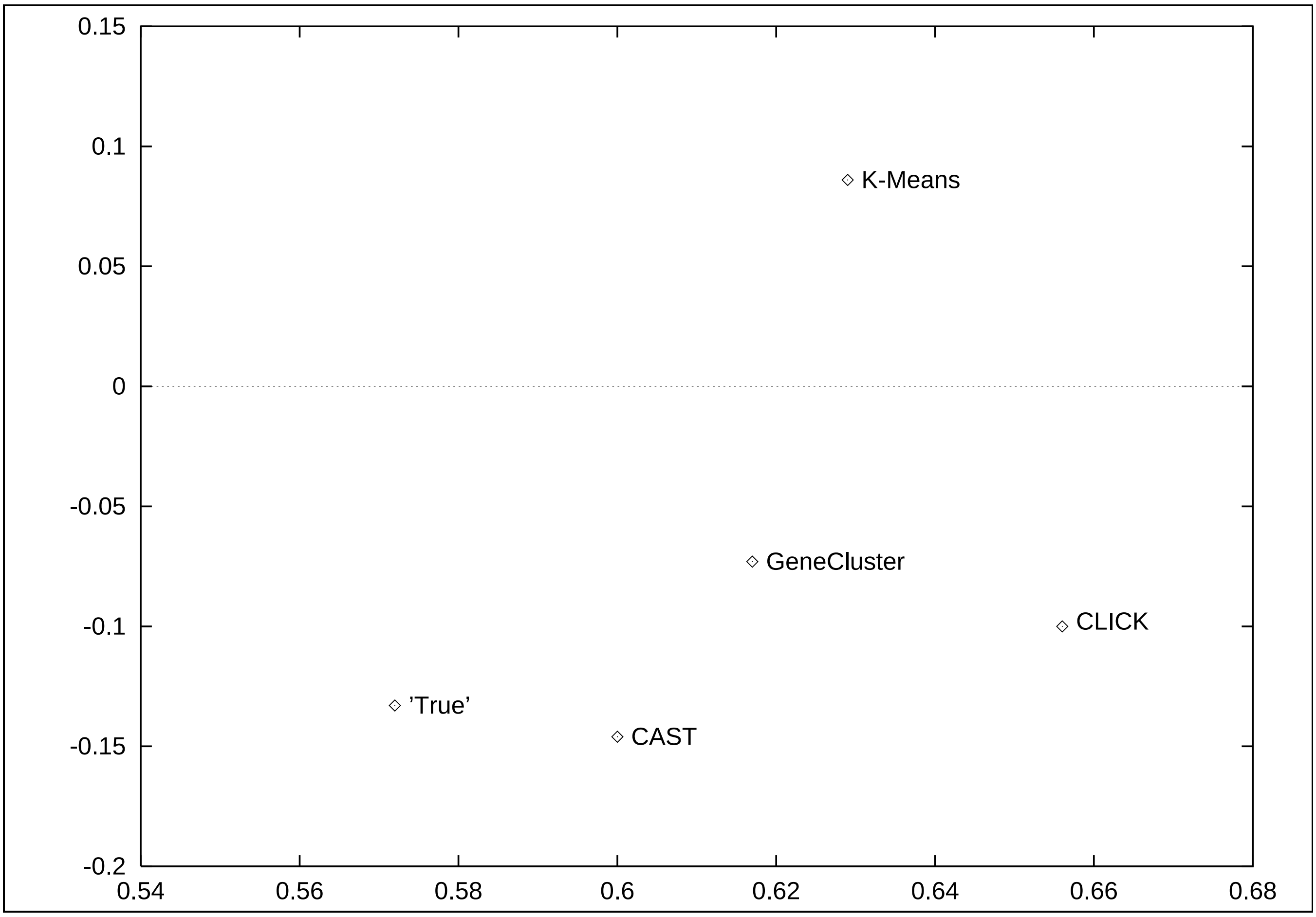

Figure 11.17: A comparison of homogeneity (x-axis) and separation (y-axis) values for all solutions. Recall that a solution improves if homogeneity increases or separation decreases.

disciplines and applications, that it is impossible to choose a single, “best” algorithm for solving the problem. This holds true even for the specific application of gene expression that we have addressed here. The eventual decision on what solution and what algorithm works best depends on the user, and on the specific questions the clustering process is supposed to answer. Each of the algorithms that we have described has its strong points and its disadvantages. We shall address briefly below several key issues.

- **Choosing the clustering approach:** The hierarchical method is exceptional in our review, as it gives an overall view of the structure without an attempt to force a hard clustering. This can be viewed as an advantage or a disadvantage, depending on the experimental goals. The other methods aim to split the universe of elements into clusters, either by geometric approaches that move cluster centers (SOM, K-Means) or by using a graph theoretic approach. The latter may take a global view (CLICK) or single out one affinity-stable cluster at a time (CAST). As noted above, many other approaches were developed in other applications.

- **How should we evaluate solution quality?** We have described above several measures that evaluate solutions, in the presence of a "correct" solution, and in its absence. The obvious advantage of having an objective function is the ability to compare solutions and measure progress in algorithm development. The caveat is that the measures may not reflect exactly the intuition that the biologist may have.

  Even if one accepts the need of a numerical measure, the clustering literature is not in agreement on which measure to use, and we have presented two measures instead: An intra-cluster measure (homogeneity) and an inter-cluster measure (separation). The two are inherently conflicting, as an improvement in one will correspond to worsening of the other. One idea of overcoming this is by presenting a curve of homogeneity vs. separation (A. Ben-Dor, private communication). Such a curve can naturally be obtained in CAST (by varying the single threshold parameter used) and can also be obtained by multiple runs of other algorithms. This curve can tell that one algorithm dominates another if it provides better homogeneity for all separation values, but typically each algorithm will dominate in a particular range. For another approach for comparing solutions across a range of parameters, see [Yeung *et al.*, 2000].

One way of getting around the "two objectives" problem is to fix the number of clusters. This is done by SOM and by the classical K-means. When the number of clusters is known this is of course the way to go. When it is not known, what users often do is run such algorithms several times with several numbers of clusters (or grid topologies, in the case of SOM). However, this brings back the problem of evaluating and comparing solutions, so algorithms that seek a globally optimal solution seem preferable. Alternative methods of determining the number of clusters are given, e.g., in [Hartigan, 1975, Tibshirani *et al.*, 2000].

- **Should we cluster all elements?** The SOM, K-means, and hierarchical algorithms require that the solution will constitute a partition of all the elements. Other algorithms, e.g. CLICK, allow some singletons to be left unclustered. By allowing singletons to be discarded, intra-cluster deviations can be reduced, perhaps at the expense of weaker separation. (Obviously, the number of discarded singletons must be kept to a small fraction of all elements, or else the solution would be meaningless.) In gene expression applications, one often does not seek an identification of *all* the genes involved, particularly since many genes have already been discarded in pre-processing steps, because of insignificant fingerprint variations. It is thus desirable to allow some room for discarding elements from a solution. It is not hard to add such flexibility into virtually all clustering algorithms that we have discussed.

- **Fingerprints vs. similarity:** Some algorithms use only similarity values between elements, while others use the fingerprints themselves. Obviously, one loses some information by using the fingerprints to compute pairwise similarities only. One of the advantages of CLICK over HCS, for example, is by explicit use of the fingerprints for merging and adoption. Geometric algorithms like K-means and SOM use only fingerprints. Other algorithms like CAST may benefit from using such information more.

- **Visualization is crucial:** As the datasets and the solutions are very large, it is imperative to have tools to visualize summaries of the data and its solution from various viewpoints. The average patterns figures are useful to show trends, and SOM goes a step further by putting similar patterns in neighboring cells in a grid, generating a convenient "executive summary". The dendograms of Eisen et al viewed together with the color-coded expression patterns of the genes are also very useful. Yet, devising additional novel, sophisticated (and ideally interactive) visualizations is an important challenge.

- **We need more testing data:** In order to improve the algorithms we need more data. The best kind is actual gene expression data, along with a known clustering solution, so that it can be compared to the algorithmic solution. This is quite hard to obtain (except perhaps for oligofingerprint data) in the current status of biological knowledge. A second best is generating synthetic (simulated) datasets with known solutions, in which one can directly control individual parameters (cluster structure, errors, etc.). Some initial work has been done in this direction [Ben-Dor *et al.*, 1999, Hartuv *et al.*, 2000], but more work is needed in order to understand how to make the simulations realistic. Generating a publicly accessible benchmark of datasets - both synthetic and real - with known solutions, would be of great benefit to developing better algorithms. In the absence of such resource, the available real data can be combined with evaluation methods as demonstrated here.

- **Clustering is only the first step:** In analyzing gene expression data, clustering is an essential initial step, but there is a lot more that can be done with the data. For example, one can use supervised learning techniques to cluster or classify the conditions. Such methods were recently shown to yield very good results in determining cancer types, with important potential applications to diagnostics [Golub *et al.*, 1999, Alizadeh *et al.*, 2000, Ben-Dor *et al.*, 2000, Brown *et al.*, 2000, Califano *et al.*, 2000]. Another useful idea is to cluster both the genes and the conditions, and to pinpoint subsets of the genes and the conditions ("biclustering") [Getz *et al.*, 2000a, Cheng and Church, 2000]. Given the clusters, a variety of biological inference steps are possible. For example, identification of common control regions of upstream regions of genes from the same cluster (see Chapter 10 of this book).

Acknowledgments

We thank Rani Elkon and Erez Hartuv for valuable help in preparing the manuscript. We also thank Ralf Herwig, Amir Ben-Dor, Michael Eisen, and Pablo Tamayo, for providing us with the results of their algorithms. R. Shamir was supported in part by a grant from the Ministry of Science, Israel. R. Sharan was supported by an Eshkol fellowship from the Ministry of Science, Israel.

Bibliography

[Alizadeh *et al.*, 2000] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson, L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D. Weisenburger, J. O. Armitage, R. Warnke, and L. M. Staudt. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[Alon *et al.*, 1999] U. Alon, N. Barkai, D. A. Notterman, G. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS*, 96:6745–6750, June 1999.

[Ball and Hall, 1967] G. Ball and D. Hall. A clustering technique for summarizing multivariate data. *Behaviorial Sciences*, 12(2):153–155, 1967.

[Ben-Dor *et al.*, 1999] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[Ben-Dor *et al.*, 2000] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini. Tissue classification with gene expression profiles. In *Proceedings of the Fourth Annual International Conference on Computational Moleculaer Biology (RECOMB 2000)*, 2000.

[Benson *et al.*, 1999] D. A. Benson, M. S. Boguski, D. J. Lipman, J. Ostell, B. F. Ouellette, B. A. Rapp, and D. L. Wheeler. Genbank. *Nucleic. Acids. Res.*, 27(1):12–17, 1999.

[Brown *et al.*, 2000] M. P. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. U. S. A.*, 97(1):262–267, 2000.

[Califano *et al.*, 2000] A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarrays for phenotype classification. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 75–85, 2000.

[Cheng and Church, 2000] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 93–103, 2000.

[Chi] The chipping forecast. Special supplement to Nature Genetics Vol 21, 1999.

[Cho *et al.*, 1998] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, TG. Wolfsberg, A. E. Gabrielian, D. Landsman, D .J. Lockhart, and R. W. Davis. A genome-wide transcirptional analysis of the mitotic cell cycle. *Mol Cell*, 2:65–73, 1998.

[Cormack, 1971] R. M. Cormack. A review of classification (with discussion). *J. Royal Statustical Society, Series A*, 134:321–367, 1971.

[Drmanac and Drmanac, 1994] S. Drmanac and R. Drmanac. Processing of cDNA and genomic kilobase-size clones for massive screening mapping and sequencing by hybridization. *Biotechniques*, 17:328–336, 1994.

[Drmanac *et al.*, 1991] R. Drmanac, G. Lennon, S. Drmanac, I. Labat, R. Crkvenjakov, and H. Lehrach. Partial sequencing by oligohybridization: Concept and applications in genome analysis. In *Proceedings of the first international conference on electrophoresis supercomputing and the human genome Edited by C. Cantor and H. Lim*, pages 60–75, Singapore, 1991. World Scientific.

[Drmanac *et al.*, 1996] S. Drmanac, N. A. Stavropoulos, I. Labat, J. Vonau, B. Hauser, M. B. Soares, and R. Drmanac. Gene-representing cDNA clusters defined by hybridization of 57419 clones from infant brain libraries with short oligonucleotide probes. *Genomics*, 37:29–40, 1996.

[Eisen and Brown, 1999] M. B. Eisen and P. O. Brown. DNA arrays for analysis of gene expression. In *Methods in Enzymology, Vol. 303*, pages 179–205. 1999.

[Eisen *et al.*, 1998] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

[Even, 1979] S. Even. *Graph Algorithms.* Computer Science Press, Rockville, Maryland, 1979.

[Everitt, 1993] B. Everitt. *Cluster analysis.* Edward Arnold, London, third edition, 1993.

[Fodor *et al.*, 1993] S. P. Fodor, R. P. Rava, X. C. Huang, A. C. Pease, C. P. Holmes, and C. L. Adams. Multiplexed biochemical assays with biological chips. *Nature*, pages 555–556, 1993.

[Getz *et al.*, 2000a] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci. USA*, 97(22):12079–12084, 2000.

[Getz *et al.*, 2000b] G. Getz, E. Levine, E. Domany, and M.Q. Zhang. Super-paramagnetic clustering of yeast gene expression profiles. *Physica*, A279:457, 2000.

[Golub *et al.*, 1999] T. R. Golub, D. K. Slonim, et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, October 1999.

[Golumbic, 1980] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.

[Hansen and Jaumard, 1997] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79:191–215, 1997.

[Harrington *et al.*, 2000] C. A. Harrington, C. Rosenow, and J. Retief. Monitoring gene expression using DNA microarrays. *Curr. Opin. Microbiol.*, 3(3):285–291, 2000.

[Hartigan, 1975] J.A. Hartigan. *Clustering Algorithms.* John Wiley and Sons, 1975.

[Hartuv and Shamir, 1999] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. Technical report, Tel Aviv University, Dept. of Computer Science, March 1999.

[Hartuv and Shamir, 2000] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(200):175–181, 2000.

[Hartuv *et al.*, 2000] E. Hartuv, A. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, and R. Shamir. An algorithm for clustering cDNA fingerprints. *Genomics*, 66(3):249–256, 2000. A preliminary version appeared in Proc. RECOMB '99, pp. 188–197.

[Herwig *et al.*, 1999] R. Herwig, A. J. Poustka, C. Meuller, H. Lehrach, and J. O'Brien. Large-scale clustering of cDNA-fingerprinting data. *Genome Research*, 9(11):1093–1105, November 1999.

[Heyer *et al.*, 1999] L.J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome Research*, 9(11):1106–1115, November 1999.

[Kerr *et al.*, 2000] Kerr, Martin, and Churchill. Analysis of variance for gene expression microarray data. Technical report, The Jackson Laboratory, 2000.

[Kohonen, 1997] T. Kohonen. *Self-Organizing Maps.* Springer, Berlin, 1997.

[Lance and Williams, 1967] G.N. Lance and W.T. Williams. A general theory of classification sorting strategies. 1. hierarchical systems. *The computer journal*, 9:373–380, 1967.

[Lennon and Lehrach, 1991] G.S. Lennon and H. Lehrach. Hybridization analysis of arrayed cDNA libraries. *Trends Genet*, 7:60–75, 1991.

[Lipshutz *et al.*, 2000] R. J. Lipshutz, S. P. A. Fodor, T. R. Gingeras, and D. J. Lockhart. High density synthetic oligonucleotide arrays. *Nature Genetics Supplement*, 21:20–24, 2000.

[Lockhart and Winzeler, 2000] D. J. Lockhart and E. A. Winzeler. Genomics, gene expression and DNA arrays. *Nature*, 405(6788):827–836, 2000.

[MacQueen, 1965] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1965.

[Marshall and Hodgson, 1998] A. Marshall and J. Hodgson. DNA chips: an array of possibilities. *Nat Biotechnol*, 16:27–31, 1998.

[Meier-Ewert *et al.*, 1995] S. Meier-Ewert, R. Mott, and H. Lehrach. Gene identification by oligonucleotide fingerprinting – a pilot study. Technical report, MPI, 1995.

[Milosavljevic *et al.*, 1995] A. Milosavljevic, Z. Strezoska, M. Zeremski, D. Grujic, T. Paunesku, and R. Crkvenjakov. Clone clustering by hybridization. *Genomics*, 27:83–89, 1995.

[Mirkin, 1996] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer, 1996.

[Poustka *et al.*, 1999] A. J. Poustka, R. Herwig, A. Krause, S. Hennig, S. Meier-Ewert, and H. Lehrach. Toward the gene catalogue of sea urchin development: the construction and analysis of an unfertilized egg cDNA library highly normalized by oligonucleotide fingerprinting. *Genomics*, 59:122–133, 1999.

[Ramsay, 1998] G. Ramsay. DNA chips: State-of-the art. *Nat Biotechnol*, 16:40–44, 1998.

[Schena *et al.*, 1996] M. Schena, D. Shalon, R. Heller, A. Chai, P. O. Brown, and R. W. Davis. Parallel human genome analysis: microarray-based expression monitoring of 1000 genes. *Proc. Natl. Acad. Sci. U.S.A.*, 93:10614–9, 1996.

[Schena, 1996] M. Schena. Genome analysis with gene expression microarrays. *Bioessays*, 18:427–431, 1996.

[Schuler, 1997] G. D. Schuler. Pieces of the puzzle: expressed sequence tags and the catalog of human genes. *J. Mol. Med.*, 75(10):694–698, 1997.

[Sharan and Shamir, 2000a] R. Sharan and R. Shamir. CLICK: A clustering algorithm for gene expression analysis. In S. Miyano, R. Shamir, and T. Takagi, editors, *Currents in Compuational Molecular Biology*, pages 6–7. Universal Academy Press, 2000.

[Sharan and Shamir, 2000b] R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 307–316, 2000.

[Sokal, 1977] R. R. Sokal. Clustering and classification: Background and current directions. In J. Van Ryzin, editor, *Classification and Clustering*, pages 1–15. Academic Press, 1977.

[Spellman *et al.*, 1998] P. T. Spellman, G. Sherlock, et al. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9:3273–3297, 1998.

[Tamayo *et al.*, 1999] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T.R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *PNAS*, 96:2907–2912, 1999.

[Tibshirani *et al.*, 2000] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistics. Technical report, Stanford University, 2000.

[Toronen *et al.*, 1999] P. Toronen, M. Kolehmainen, G. Wong, and E. Castren. Analysis of gene expression data using self-organizing maps. *FEBS Letters*, 451:142–146, 1999.

[Vicentic and Gemmell, 1992] R. Drmanac S. Drmanac I. Labat R. Crkvenjakov A. Vicentic and A. Gemmell. Sequencing by hybridization: towards an automated sequencing of one milion M13 clones arrayed on membranes. *Electrophoresis*, 13:566–573, 1992.

[Yeung *et al.*, 2000] K.Y. Yeung, D.R. Haynor, and W.L. Ruzzo. Validating clustering for gene expression data. Technical Report UW-CSE-00-01-01, University of Washington, 2000.