Analysis of Gene Expression Data **Spring Semester, 2007**

Lecture 3: March 15, 2007

Lecturer: Ron Shamir *Scribe: Yifat Felder and Oved Ourfali*[1]

3.1 Introduction

3.1.1 Functional Genomics

Having reached the end of the Human Genome Project, the question that needs to be asked is: "What's next?". The complete sequencing of the Human Genome is an immense task, which is now nearing completion. While much work remains to be done even there, there are a number of areas this knowledge opens up to research, which have thus far been nearly impossible to pursue. Among those is "functional genomics" - the search for understanding the functionality of specific genes, their relations to diseases, their associated proteins and their participation in biological processes. Most of the knowledge gained so far in this area is the result of painstaking research of specific genes and proteins, based on complex biological experiments and homologies to known genes in other species. This "Reductionist" approach to functional genomics is hypothesis driven (i.e., it can be used to check an existing hypothesis, but not to suggest a new one). The advancements in both biological and computational techniques are now beginning to make possible a new approach: the "Holistic" research paradigm. This approach is based on high-throughput methods: global gene expression profiling ("transcriptome analysis") and wide-scale protein profiling ("proteome analysis"). In the holistic approach, a researcher simultaneously measures a very large number of gene expression levels throughout a biological process, thereby obtaining insight into the functions and correlations between genes on a global level. Unlike the reductionist approach, these methods can generate hypotheses.

3.1.2 Representation of gene expression data

Gene expression data can be represented as a real matrix R, called the *raw data matrix*. Each row in the matrix contains data regarding a specific gene, and each column represents a condition, or a tissue profile. Thus, R_{ij} is the expression level for gene i, at condition j. The expression data can contain ratios, absolute values, or distributions. The expression

[1]Based in part on a scribe by Seagull Chalamish and Itamar Elem, March 03, 2005;Michal Ozery-Flato and Israel Steinfeld, April 2004; Dror Fidler and Shahar Harrusi, April 2002; Giora Sternberg and Ron Gabo, May 2002.

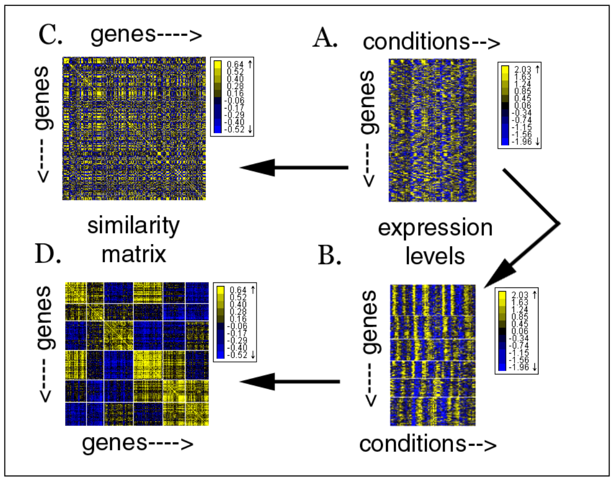

Figure 3.1: Data and similarity matrices [10]. The raw data matrix A and the clustered data matrix B map conditions with gene expression. The raw data *similarity matrix* C and the clustered data similarity matrix D are derived from the raw data matrix or the clustered data matrix, according to a similarity or distance function. In A and B, yellow color represents above-average expression level and blue represents below-average expression level. In C and D, yellow represents high similarity, and blue represents low similarity.

pattern (fingerprint vector) of a gene i is the i^{th} row of R. The condition profile (experiment profile) of a condition j is the j^{th} column of R. In some clustering algorithms the raw data matrix is preprocessed to compute a *similarity* matrix S, where S_{ij} reflects the similarity of the expression patterns of gene i and gene j. Note that the similarity matrix is larger than the data matrix since there are usually much more genes than conditions. Figure 3.1 shows the data and similarity matrices.

3.1.3 Clustering applications

Clustering genes or conditions is a basic tool for the analysis of expression profiles, and can be useful for many purposes, such as:

- Deducing functions of unknown genes from known genes with similar expression patterns (similar expression patterns are postulated to imply a similar function).
- Identifying disease profiles - tissues with similar pathology should yield similar expression profiles.
- Deciphering regulatory mechanisms - co-expression of genes may imply co-regulation.

- Classification of biological conditions.
- Genotyping.
- Drug development.
- And more ...

3.1.4 The clustering problem

Genes are said to be similar if their expression patterns correlate, and non-similar otherwise. The goal of gene clustering process is to partition the genes into distinct sets such that genes that are assigned to the same cluster should be similar, while genes assigned to different clusters should be non-similar. Usually there is no single solution that is the "true"/correct mathematical solution for this problem. A good clustering solution should have two merits:

1. High *homogeneity*: homogeneity measures the similarity between genes assigned to the same cluster.
2. High *separation*: separation measures the distantnce/dis-similarity between the clusters. Each cluster should represent a unique expression pattern. If two clusters have similar expression patterns, then probably they should be merged into one cluster.

Note that these two measures are in a way opposite - if you wish to increase the homogeneity of the clusters you would increase the number of clusters, but the price would be a reduction of the separation.
There are many formulations for the clustering problem, and most of them are NP-hard. For that reason, heuristics and approximations are used. Clustering methods have been used in a vast number of fields. We can distinguish between two types of clustering methods:

Agglomerative These methods build the clusters by looking at small groups of elements and performing calculations on them in order to construct larger groups. Hierarchal methods of this sort will be described in the next lecture.

Divisive A different approach which analyzes large groups of elements in order to divide the data into smaller groups and eventually reach the desired clusters. We shall see non-hierarchical techniques which use this approach.

There is another way to distinguish between clustering methods:

Hierarchical Here we construct a hierarchy or tree-like structure to see the relationship between entities. The following hierarchical algorithms will be presented in the next lecture: Neighbor Joining, Average Linkage and a general framework for hierarchical cluster merging algorithms.

Non-Hierarchical In non-hierarchical methods, the elements are partitioned into non-overlapping groups. The following non-hierarchical algorithms will be shown here: k-means, SOM, PCC and CAST. The CLICK algorithm will be presented in the next lecture.

3.2 k-means clustering

This method was introduced by MacQueen [7]. Given a set of n points $V = \{v_1, ..., v_n\}$, and an integer k, the goal is to find a k-partition of minimal cost for V. If P implies a partition of V into k subsets, $C_1 \dots C_k$, then a centroid or center of a cluster C_i is the center of gravity of its set of points. Let E^P be a function that measures the quality of the partition, the *solution cost*. In each iteration the algorithm moves one element between two clusters, in order to improve the clustering score. The two affected cluster centers are updated.

k-means clustering :

1. Initialize an arbitrary partition P into k clusters.
2. For cluster j, element $i \notin j$.
   $E^P(i,j) =$ Cost of the solution if i is moved to cluster j.
3. Pick $E^P(r,s)$ that is minimum.
4. Move element r to cluster s, if it improves E^P.
5. Repeat until no improvement is possible.

Note that this method requires knowledge of k, the number of clusters, in advance. Once k is fixed, the algorithm aims at optimizing homogeneity, but not separation, i.e., elements in different clusters can still remain similar.

The most common use of the k-means algorithm is based on the idea of moving elements between two clusters based on their distances to the centers of the different clusters. The solution cost function in that case is defined by:

$$E^P = \sum_{p \in P} \sum_{i \in p} D(v_i, c_p)$$

Where c_p is the center of cluster p and $D(v_i, c_p)$ is the distance of v_i from c_p.

This algorithm has few variations:

- k-means - in which $E^P = \sum_{p \in P} \sum_{i \in p} D(v_i, c_p)^2$. Hence, we punish samples that are far from the center.
- k-medians - in which $E^P = \sum_{p \in P} \sum_{i \in p} D(v_i, c_p)$.
- k-centers - in which $E^P = \max_{p \in P} \max_{i \in p} D(v_i, c_p)$.

An example for k-means application is the geometric k-clustering, in which the input V - a set of n points in R^n, and k - the number of clusters. In this application, E^P is the mean squared error of the distances between the samples and the centers, i.e. $E^P = \sum_{p \in P} \sum_{i \in p} \frac{D(v_i, c_p)^2}{n}$.

There are some variations of the algorithm involving changing of k. Also there are parallel versions in which we move each element to the cluster with the closest centroid simultaneously, but then convergence is not guaranteed. The k-means is a greedy algorithm in its nature and might get stuck at local minimum, but it is simple, easy for implementation and thus very popular.

3.3 Self organizing maps

Kohonen [6] introduced this method and Tamayo *et al.* [12], first applied it to gene expression data. Self organizing maps are constructed as follows. k is fixed and some topology on the centers is assumed. One chooses a grid, $k = l \times m$, of nodes, and a distance function between nodes, $D(N_1, N_2)$. Each of the grid nodes is mapped into a k-dimensional space, at random. The gene vectors are mapped into this space as well. As the algorithm proceeds, the grid nodes are iteratively adjusted (see Figure 3.2). Each iteration involves randomly selecting a data point P and moving the grid nodes in the direction of P. The closest node n_P is moved the most, whereas other nodes are moved by smaller amounts depending on their distance from n_P in the initial geometry of the grid. In this fashion, neighboring points in the initial geometry tend to be mapped to nearby points in the k-dimensional space. The process continues iteratively.

Self organizing maps :

1. Input: n-dimentional vector for each element (data point) p.
2. Start with a grid of $k = l \times m$ nodes, and a random n-dimentional associated vector $f_0(v)$ for each grid node v, representing the initial associated center.
3. Iteration i:

   Pick a data point p. Find a grid node n_p such that $f_i(n_p)$ is the closest to p.

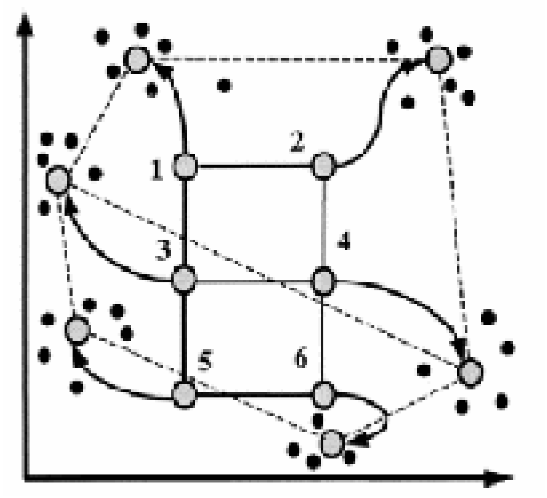

Figure 3.2: Self organizing maps : Initial geometry of nodes in a 3×2 rectangular grid is indicated by solid lines connecting the nodes. Hypothetical trajectories of nodes as they migrate to fit data during successive iterations of the self organizing maps algorithm are shown. Data points are represented by black dots, six nodes of the self organizing map by large circles, and trajectories by arrows.

Update all node vectors v as follows :

$$f_{i+1}(v) = f_i(v) + H(D(n_p, v), i)[p - f_i(v)]$$

Where H is a learning function which decreases with the number of iterations (i), as well as with $D(n_p, v)$. i.e. nodes that are farther from n_p are less affected.

4. Repeat until no improvement is possible.

The clusters are defined by the grid nodes. We assign each point (gene vector) to its nearest node n_p (cluster). The movement of a center is affected not only by the elements of its own cluster. Note that the number of clusters, k, is set a-priori in this method.

GENECLUSTER - an implementation of SOM

GENECLUSTER is a software that implements self organizing maps (SOM) clustering for gene expression analysis, developed by Tamayo *et al.* [12]. Some results can be seen in Figure 3.3. GENECLUSTER accepts an input file of expression levels from any gene-profiling method (e.g., oligonucleotide arrays or spotted cDNA arrays), together with a geometry for the nodes. The program begins with two preprocessing steps that greatly improve the ability

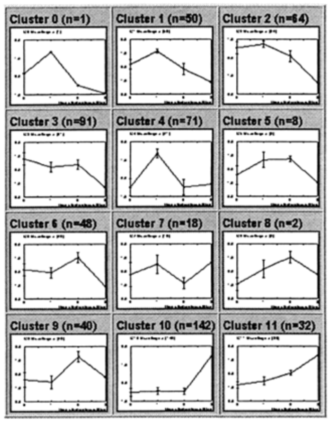

Figure 3.3: Macrophage differentiation in HL-60 cells. The self organizing map algorithm was applied to models of human macrophage differentiation. This process is largely controlled at the transcriptional level, and is related to the pathogenesis of leukemia. 567 genes were divided in to clusters using a 4x3 self organizing map. In each graph the normalized and averaged expression levels along with standard deviation values for each cluster are shown.

to detect meaningful patterns. First, the data is passed through a variation filter to eliminate those genes with no significant change across the samples. This prevents nodes from being attracted to large sets of invariant genes. Second, the expression level of each gene is normalized across experiments. This focuses attention on the "shape" of expression patterns rather than on absolute levels of expression. A SOM is then computed. Each cluster is represented by its average expression pattern along with the standard deviation values (see Figure 3.3), making it easy to discern similarities and differences between the patterns. The following learning function $H(n, r, i)$ is used, where n and r are nodes, and i stands for iteration:

$$H(n,r,i) = \begin{cases} \frac{0.02T}{T+100i} & \text{if } D(n,r) \leq \rho(i) \\ 0 & \text{otherwise} \end{cases}$$

Radius $\rho(i)$ decreases linearly with i ($\rho(0) = 3, \rho(T) = 0$), where T is the maximum number of iterations, and $D(n, r)$ denotes the distance within the grid.

3.4 Graph clustering approaches

The similarity matrix can be transformed into a *similarity graph*, G_θ, where the vertices are genes, and there is an edge between two vertices i and j if their similarity $S_{i,j}$ is above some threshold θ. More formally, for a pair of vertices i,j, $(i, j) \in E(G_\theta)$ iff $S_{i,j} > \theta$.

3.4.1 The corrupted clique graph model

The clustering problem can be modeled by a corrupted clique graph. A *clique graph* is a graph consisting of disjoint cliques. The true clustering is represented by a clique graph H (vertices are genes and cliques are clusters). Contamination errors introduced into gene expression data result in a *similarity graph* $C(H)$ which is not a clique graph. Under this model the problem of clustering is as follows: given $C(H)$, restore the original clique graph H and thus the true clustering.

Graph theoretic approach

A model for the clustering problem can be reduced to clique graph edge modification problems, stated as follows.

Problem 3.1 *Clique graph editing problem*
INPUT: $G(V, E)$.
OUTPUT: $Q(V, F)$ a clique graph which minimizes the size of the symmetrical difference between the two edge sets: $|E \Delta F| = (E \backslash F) \cup (F \backslash E)$.

Clique graph editing problem is NP-hard [11].

Problem 3.2 *Clique graph completion problem*
INPUT: $G(V, E)$.
OUTPUT: $Q(V, F)$ a clique graph with $E \subseteq F$ which minimizes $|F \setminus E|$.

The clique graph completion problem can be solved by finding all connected components of the input graph and adding all missing edges in each component. Thus the clique graph completion problem is polynomial.

Problem 3.3 *Clique graph deletion problem*
INPUT: $G(V, E)$.
OUTPUT: $Q(V, F)$ a clique graph with $F \subseteq E$ which minimizes $|E \setminus F|$.

The clique graph deletion problem is NP-hard [8]. Moreover, any constant factor approximation to the clique graph deletion problem is NP-hard as well [11].

Probabilistic approach

Another approach is to build a probabilistic model of contamination errors and try to devise an algorithm which, given $C(H)$, reconstructs the original clique graph H with high probability.

One of the simplest probabilistic models for contamination errors is a *random corrupted clique graph*. The contamination errors are represented by randomly removing each edge in

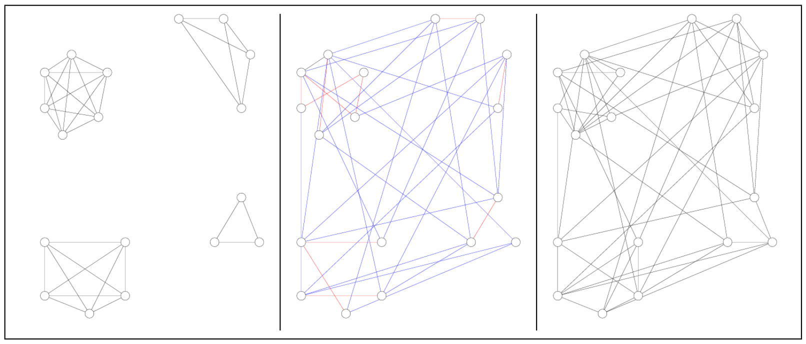
Figure 3.4: The randomly corrupted clique graph model. Left: the original clustering H of 4 clusters, 18 elements. Middle: random contamination (flip each edge with a probability $p < 0.5$), red edges denote edges that will be removed, blue edges denote added edges. Right: $G = C(H)$, the input (contaminated) graph.

the original clique graph H, with probability $p < 0.5$, and adding each edge not in H with the same probability, p (see Figure 3.4). We will denote by $\Omega(H, p)$ the set of all corrupted clique graphs derived from H with contamination error fraction p using this model.

3.4.2 The PCC algorithm

In this section we present a clustering algorithm of Ben-Dor *et al.* [2], called Parallel Classification with Cores (PCC). We begin with a few definitions.

Definition A *cluster structure* is a vector $(s_1, ..., s_d)$, where each $s_j > 0$ and $\sum s_j = 1$. Each s_i represent a fraction of the total number of genes that appear in cluster i. Thus, n-vertex clique graph has structure $(s_1, ..., s_d)$ if it consists of d disjoint cliques of sizes $ns_1, ..., ns_d$.

Definition A clique graph $H(V, E)$ is called γ-*clustering* (has a γ-cluster structure), if the size of each clique in H is at least $\gamma|V|$.

Algorithm idea

Assume that we already have a clustering of a subset U_1 of vertices. Let $\{W_1, ..., W_m\}$ denote this clustering. We will *extend* the clustering $\{W_1, ..., W_m\}$ to include the elements of another set U_2, by putting each vertex $v \in U_2$ into the cluster W_i, to which it has the highest *relative density* (affinity), that is, the highest ratio between the number of edges connecting v to vertices in W_i, and the size of W_i (see Figure 3.5). Formally put, we choose the cluster W_i which maximizes $\frac{|\{u|u \in W_i, (u,v) \in E\}|}{|W_i|}$.

After the extension, $\{W_1, ..., W_m\}$ is the clustering of $U_1 \cup U_2$. Note that during the extension procedure no new clusters are added, thus the number m of clusters is unchanged.

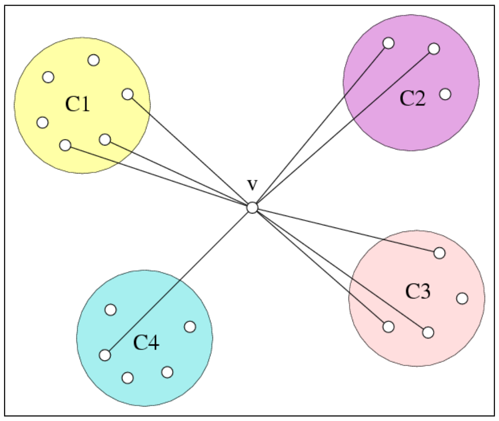

Figure 3.5: Relative Density - the highest relative density of v is with cluster C_3 (relative density of v with clusters C_1, C_2, C_3, C_4 is 1/2, 2/3, 3/4, 1/5, respectively).

Algorithm outline

Suppose we are given $G(V, E)$, a corrupted clique graph over n vertices, that is $G \in \Omega(H, p)$ for some clique graph H with γ-cluster structure. Because H has a γ-cluster structure, the maximum number of cliques in H is $m = \lceil 1/\gamma \rceil$.

The PCC algorithm will perform the following steps (see Figure 3.6):

1. Uniformly draw $U_1 \subset V$, such that $|U_1| = O(\log\log(n))$;
2. Uniformly draw $U_2 \subset V \backslash U_1$, such that $|U_2| = O(\log(n))$;
3. For each clustering of U_1 into at most m clusters $\{W_1, ..., W_l\}$, perform:
   (a) Extend the clustering $\{W_1, ..., W_l\}$ of U_1 into clustering X(W)=$\{X_1, ..., X_l\}$ of $U_1 \cup U_2$;
   (b) Extend the clustering $\{X_1, ..., X_l\}$ into a clustering Y(W)=$\{Y_1, ..., Y_l\}$ of V;
4. Each clustering $\{Y_1, ..., Y_l\}$ of V from the previous step determines a clique graph over V. Amongst all these clique graphs, choose the one which is closest (in the symmetric difference sense) to the input graph. Meaning, choose C = argmin $|E(G)\Delta E(C)|$.

Algorithm correctness and running time

Before presenting the proof of the algorithm, we introduce the following definitions:

Definition Given two probabilities, p and a, let $D(p\|a)$ denote the *relative entropy distance* from $(p, 1-p)$ to $(a, 1-a)$, that is, $D(p\|a) = p\log_2(p/a) + (1-p)\log_2((1-p)/(1-a))$. We use $k(\alpha)$ to denote $\lceil 2/D(1/2\|\alpha)\rceil$.

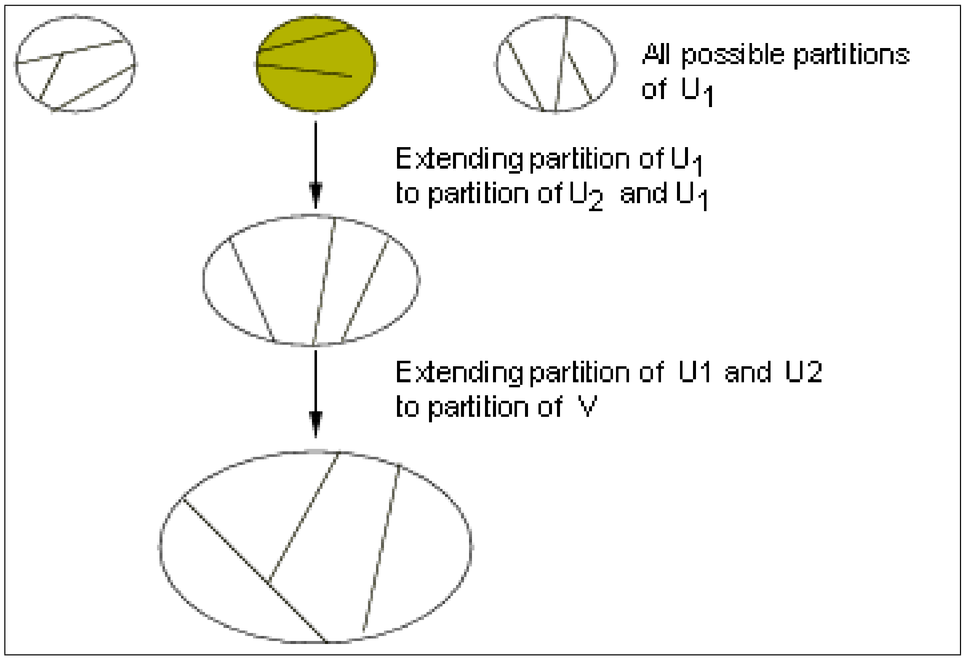
All possible partitions of U_1
Extending partition of U_1 to partition of U_2 and U_1
Extending partition of U1 and U2 to partition of V

Figure 3.6: PCC algorithm steps shown schematically.

Definition Given two graphs $G = (V, E)$ and $G' = (V', E')$, let $\Delta(G, G')$ denote the distance between both graphs, $\Delta(G, G') = |E \Delta E'|$.

Definition The random graph model $\mathcal{Q}(n, \alpha, S)$ (representing random corruption of clique graphs) is defined as follows: Given a clique graph H over n vertices with structure S, and a value $0 \leq \alpha < \frac{1}{2}$, the random graph $G_{H,\alpha}$ is obtained from H by randomly (1) removing each edge in H with independent probability α; (2) adding each edge not in H with independent probability α.

Definition Consider an algorithm A that takes an arbitrary graph G as input and returns a clique graph $A(G)$ on the same vertex set. Let $\delta > 0$. We say that A *clusters* $\mathcal{Q}(n, \alpha, S)$ *with probability* $1 - \delta$ if when applied to the random graph $G_{H,\alpha}$, the output graph is, asymptotically, as good a solution as the original clique graph with probability $1 - \delta$. More precisely, we require that for a large enough n, and for any clique graph H with structure S, we have

$$\mathbb{P}[\Delta(A(G_{H,\alpha}), G_{H,\alpha}) \leq \Delta(H, G_{H,\alpha})] > 1 - \delta.$$

Here and throughout this section $\mathbb{P}$ denotes the relevant probability measure, which is clear from the context.

To analyze the algorithm we need the following theorem and lemma:

Theorem 3.1 *Chernoff 1952 [3]*
Let $X \sim$ Binomial(n,p). Let $a < p < b$, then:

$$P(X \geq bn) < \exp(-nD(b\|p))$$
$$P(X \leq an) < \exp(-nD(a\|p))$$

Lemma 3.2 *Consider n objects of d different colors, where each color is represented by at least n/m objects. If s objects are sampled uniformly and independently without replacement, then*

$$\mathbb{P}\left(\begin{array}{c}\textit{The sample contains} \geq s/2m \\ \textit{representatives of each color}\end{array}\right) > 1-\delta,$$

provided that $16m^2\log(d/\delta) \leq s \leq \frac{n}{4m}$.

Proof:
Call a sample as above *bad* if it does not satisfy the condition for a fixed color A.

$$p = \mathbb{P}(\text{ bad sample }) \leq \mathbb{P}(X < s/2m)\,,$$

where $X \sim$ Binomial $\left(s, \frac{(n/m)-s}{n}\right)$. This is true since even with no replacement the proportion of A-colored elements left in the pile in each trial is more than $\frac{(n/m)-s}{n}$. Therefore, by the Chernoff bound above, and assuming $n > 4ms$,

$$\begin{aligned} p &< \exp\left(-s \cdot D(\frac{1}{2m}\|\frac{3}{4m})\right) \qquad (3.1) \\ &\leq \exp\left(-\frac{s}{16\log(2)m^2}\right). \end{aligned}$$

The inequality in (3.2) follows from the general inequality [4]: $D(p\|q) \geq (1/\ln(2)) \cdot (p-q)^2$. This last expression is less than δ/d by our assumption on the sample size s. A union over all colors yields the stated result.

■

Theorem 3.3 *Let S be a cluster structure and let $\alpha < 1/2$. For any fixed $\delta > 0$ the above algorithm clusters $\mathcal{Q}(n,\alpha,S)$ with probability $1-\delta$. The time complexity of the algorithm is $O\left(n^2 \cdot \log(n)^c\right)$, where c is a constant that depends on α and on $\gamma(S)$.*

Proof:
Since $m = \lceil\frac{1}{\gamma(S)}\rceil$, $d(S) \leq m$. m is considered a constant for our setup. Let $T = \langle T_1, \ldots T_m\rangle$ be the partition of V that represents the underlying clusters, where some clusters may be empty. For a vertex $v \in V$ let $i(T,v)$ be defined by $v \in T_{i(T,v)}$. Let $\eta > 0$ (η will be related to the tolerated failure probability, δ, at the end). Recall that $k(\alpha) = \lceil 2/D(1/2\|\alpha)\rceil$.

1. Uniformly draw a subset U_1 of vertices of size $2m \cdot k(\alpha)\log\log(n)$. If n is large enough, namely: $\log\log(n) > 8mk(\alpha)\log(1/\eta)$ and $n > 8m^2k(\alpha)\log\log(n)$ we know (by Lemma 3.2) that with probability $1-\eta$ each color has at least $k(\alpha)\log\log(n)$ representatives in this chosen subset.

2. Uniformly over the subsets of $V \setminus U_1$ draw a subset U_2 of vertices with $2m \cdot k(\alpha)\log(n)$ elements. Again, for n large enough, with probability $1-\eta$ each color has at least $k(\alpha)\log(n)$ representatives in this subset.

3. Consider all partitions of U_1 into m subsets (for n large enough there are less than $\log(n)^{2m\log(m)\cdot k(\alpha)}$ of them). Denote each such partition by $W = \langle W_1, \dots, W_m\rangle$ (some subsets may be empty). Run the following enumerated steps starting with all these partitions. For the analysis focus on a partition where each W_i is a subset of a distinct true cluster T_j.

   Such a partition is, indeed, considered, since we are considering all partitions. For this case we can further assume, without loss of generality, that for each i we have $W_i \subset T_i$.

   (a) Start with sets $X_i = W_i$. For all $u \in U_2$ let $i(X,u)$ be the index that attains the maximum ($1 \le i \le m$) of $\deg(u, W_i)/|W_i|$. Add u to that set. Let $W(u) = W_{i(T,u)}$. The collection of edges from u to $W(u)$ are independent Bernoulli($1-\alpha$) (the drawings of U_1 and U_2 were independent of everything else). Therefore $\deg(u, W(u)) \sim$ Binomial($|W(u)|, 1-\alpha$). Using the Chernoff bound stated above we therefore have

$$\begin{aligned} \mathbb{P}\left(\deg(u,W(u)) \le \frac{|W(u)|}{2}\right) &< \exp(-|W(u)|D(\frac{1}{2}\|\alpha)) \\ &< \log(n)^{-k(\alpha)D(\frac{1}{2}\|\alpha)} && (3.2) \\ &< \log(n)^{-2}, && (3.3) \end{aligned}$$

   where $|W(u)| \ge k(\alpha)\log\log(n)$ justifies (3.2). Similarly, for $i \ne i(T,u)$, we have

$$\deg(u, W_i) \sim \text{Binomial}(|W_i|, \alpha),$$

   and thus

$$\begin{aligned} \mathbb{P}(\deg(u,W_i) \ge |W_i|/2) &< \exp(-|W_i|D(\frac{1}{2}\|\alpha)) \\ &< \log(n)^{-2}, && (3.4) \end{aligned}$$

   whence $i(X,u) = i(T,u)$ with high probability: $\mathbb{P}(i(X,u) \ne i(T,u)) < m\log(n)^{-2}$. Finally, by a union bound

$$\mathbb{P}(i(X,u) \neq i(T,u) \text{ for some } u \in U_2\ ) < 2m^2 \cdot k(\alpha)\log(n)^{-1}. \tag{3.5}$$

(b) Focusing on the part of the measure space where no error was committed in the previous steps (in particular, all vertices were assigned to their original color), we now have m subsets of vertices $X_i \subset T_i$, $i = 1...m$, each of size at least $k(\alpha)\log(n)$, unless the corresponding T_i is empty. We take all other vertices and classify them using these subsets, as in the previous step. Let the resulting partition be $Y = \langle Y_1, \ldots, Y_m \rangle$ and for vertices $v \in V$ let $i(Y,v)$ be defined by $v \in Y_{i(Y,v)}$. Observe that all edges used in this classification are independent of the algebra generated by everything previously done. This is true since in the previous step only edges from U_2 to U_1 were considered, and these are of no interest here. Therefore, the equivalents of (3.3) and (3.4) hold, yielding

$$\mathbb{P}(i(Y,v) \neq i(T,v) \text{ for any } v \in V\ ) < 2m^2 \cdot k(\alpha)n^{-1}. \tag{3.6}$$

4. Amongst all outputs of the above, choose the partition which is closest (in the symmetric difference sense) to the input graph.

The total probability of failure in this process is estimated as follows

$$\mathbb{P}\left(\begin{array}{c} \text{The original partition } V = \bigcup_{i=1}^m T_i \\ \text{is not one of the outputs} \end{array} \right) \leq 2\eta + 2m^2 \cdot k(\alpha)\left(n^{-1} + \log(n)^{-1}\right), \tag{3.7}$$

which is arbitrarily small for large n and if η is chosen appropriately.

As noted above, we have less than $\log(n)^{2m\log(m)\cdot k(\alpha)}$ possible partitions of U_1. Each such partition leads to a clustering of all vertices in V, using the core clusters X_i $, i = 1...m$. For each partition $O(n\log(n))$ edges are considered in the classification step. Each edge is considered at most once, as sums of disjoint edge subsets are compared to a threshold. Computing the distance of each of the clique graphs produced to the input graph requires $O(n^2)$ operations. Thus the total time complexity of the algorithm is $O(n^2 \cdot \log(n)^{2m\log(m)\cdot k(\alpha)})$.

■

3.4.3 Practical heuristic - The CAST algorithm

Although the theoretical ideas presented in the previous section show asymptotic running time complexity of $O(n^2 \log^c n)$, their implementation is still impractical (the constants, for instance, are very large, as in the computation of all possible partitions of U_1 into at most m clusters in step 3). Therefore, based on ideas of the theoretical algorithm, CAST (Cluster

Affinity Search Technique), a simple and practical heuristic, was developed. All the tests described in the subsequent sections were performed using this practical implementation of the theoretical algorithm.
Suppose we are given $G(V, E)$, a corrupted clique graph over n vertices, that is $G \in \Omega(H, p)$ for some clique graph H. Let C be a cluster. Let $S_{i,j}$ be a similarity matrix and let $v \in V$ be a gene. We define the *affinity of v to cluster C* by $\frac{\sum_{u \in C} S_{u,v}}{|C|}$. Given an *affinity threshold* τ we will say that v is *a close gene to cluster C* if its affinity to C is above τ and we will say that *v is a weak gene in C* if its affinity to C is below τ. Following are the steps of the practical implementation. Repeat the following until all genes are clustered:

- Start a new cluster at a time by picking an unclustered gene, and denote it by CC. As long as changes occur, repeat the following steps:
  - Add a close gene to CC;
  - Remove a weak gene from CC;

  Close CC when no addition or removal is possible;

The main differences between the practical implementation and the theoretical algorithm are:

1. In the theoretical algorithm several partitions are formed and then the “best” partition is chosen. The clusters in a partition are extended by adding new elements to them. In the practical implementation one partition is formed by building one cluster at a time, and removal of weak elements from a cluster is allowed. This enables correction in case the seed of the formed cluster is wrong.
2. The theoretical algorithm considers the similarity graph, while the practical implementation processes the similarity matrix (the similarity value between any two genes can assume any real value).
3. In the theoretical algorithm addition is done independently, while the practical implementation adds genes incrementally.

Although little can be proved about the running time and performance of the practical implementation, the test results described in the next sections show that it performs remarkably well, both on simulated data and on real biological data.

BioClust

BioClust is an implementation package of the CAST heuristic. The following section presents results of applying BioClust on both synthetic data and real gene expression data.

Clustering quality assessment

There are several measures to asses the quality of a clustering C, some are calculated given the true clustering T, and some without assuming anything regarding the true clustering. Here, we only refer to the former. Given two elements, they are considered *mates* if they are a part of the same cluster, and *non-mates* otherwise. All the measures use the following notations:

- n_{11} - number of pairs of elements that are mates in both C and T.
- n_{10} - number of pairs of elements that are mates in C and non-mates in T.
- n_{01} - number of pairs of elements that are non-mates in C and mates in T.
- n_{00} - number of pairs of elements that are non-mates in both C and T.

The most common measures that asses the quality of a clustering are:

- *matching coefficient* - $\frac{n_{00}+n_{11}}{n_{00}+n_{01}+n_{10}+n_{11}}$, that is the total number of matching entries divided by the total number of entries.
- *Jaccard coefficient* - $\frac{n_{11}}{n_{01}+n_{10}+n_{11}}$, a score similar to the matching coefficient, only with n_{00}, the number of entries which are non-mates in both matrices, removed. In sparse graphs n_{00} will be a dominant factor, and thus the Jaccard coefficient is more sensitive when dealing with sparse graphs.

In all the measures above, the higher the value, the closer the result is to the real clustering. All measures have a maximum value of 1, which implies perfect clustering.

Clustering synthetic data

The simulation procedure is as follows (please refer to Figure 3.7A for visualization of the simulation procedure):

- Let H be the original clique graph.
- Generate G from H by independently removing each edge in H with probability p and adding each edge not in H with probability p.
- Randomly permute the order of vertices in G and run BioClust with affinity threshold $\tau = 0.5$.
- Compare BioClust's output to the original graph H.

cluster structure	n	p	matching coeff.	Jaccard coeff.
$\{0.4, 0.2, 0.1 \times 4\}$	500	0.2	1.0	1.0
$\{0.4, 0.2, 0.1 \times 4\}$	500	0.3	0.999	0.995
$\{0.4, 0.2, 0.1 \times 4\}$	500	0.4	0.939	0.775
$\{0.1 \times 10\}$	1000	0.3	1.0	1.0
$\{0.1 \times 10\}$	1000	0.35	0.994	0.943

Table 3.1: Performance of BioClust for different values of p and n. Mean values of matching coefficient and Jaccard coefficient are given.

Table 3.1 presents results of simulation for different values of contamination error p and/or number of cluster entities n. The values of the matching coefficient and the Jaccard coefficient are presented. It can be seen that the Jaccard coefficient is more sensitive. One can also observe the effect of p and n on the performance of the algorithm.

Figure 3.7 B presents results of simulations for different values of n and p. It can be seen that the properties of the theoretical algorithm are preserved in its practical implementation. We get better performance when the number of clustered entities (vertices in H) increases.

Clustering temporal gene expression data

The gene expression data used in this experiment is from [13]. In this paper the authors study the relationship among expression patterns of genes involved in the rat Central Nervous System (CNS).

Gene expression patterns were measured for 112 genes along 9 different development time points. The gene expression data for each gene was augmented with derivative values to enhance the similarity for closely parallel but offset expression patterns, resulting in a 112×17 expression matrix. The similarity matrix was obtained using Euclidean distance. The execution of BioClust resulted in eight clusters. Since partitioning to clusters is known from [13] this experiment was done mainly for validation of the algorithm.

Figure 3.7 C and D presents the clustering results. Note that all clusters, perhaps with the exception of cluster #1, manifest clear and distinct expression patterns. Moreover, the agreement with the prior biological classification is quite good.

Clustering *C. elegans* gene expression data

The gene expression data used in this analysis is from [5]. Kim *et al.* studied gene regulation mechanisms in the nematode *C. elegans*. Gene expression patterns were measured for 1246 genes in 146 experiments, resulting in a 1246×146 expression matrix. The similarity matrix was obtained using Pearson correlation.

The algorithm found 40 clusters. Only very few genes out of the 1246 were classified into families by prior biological studies. The algorithm clustered these families quite well into few homogeneous clusters (see Figure 3.8).

One example of the potential use of clustering for analyzing gene expression patterns is shown in Figure 3.8. A six-gene cluster (cluster #24) contained two growth-related genes and four anonymous genes. This suggests the possibility that the other four genes are also growth-related, paving the way for future biological research.

Tissue clustering

The gene expression data used in this experiment is from [1]. The authors describe an analysis of gene expression data obtained from 62 samples of colon tissue, 40 tumor and 22 normal tissues. Gene expression patterns were measured for 2000 genes in the 62 samples, using an Affymetrix chip. The similarity between each two samples was measured using Pearson correlation. Note that here, the similarity is measured between tissues, not genes.

BioClust formed 6 clusters of the data. Figure 3.9 shows the distribution of tumor and normal tissues in the six clusters produced.

The main goal of clustering here is to achieve a separation of tumor and normal tissues. This experiment demonstrates the usefulness of clustering techniques in learning more about the relationship of expression profiles to tissue types.

Improved theoretical results

Shamir & Tsur [9] have introduced a generalized random clique graph model with improved theoretical results, including reduction of the $\Omega(n)$ restriction on cluster sizes, and stronger results when cluster sizes are almost equal.

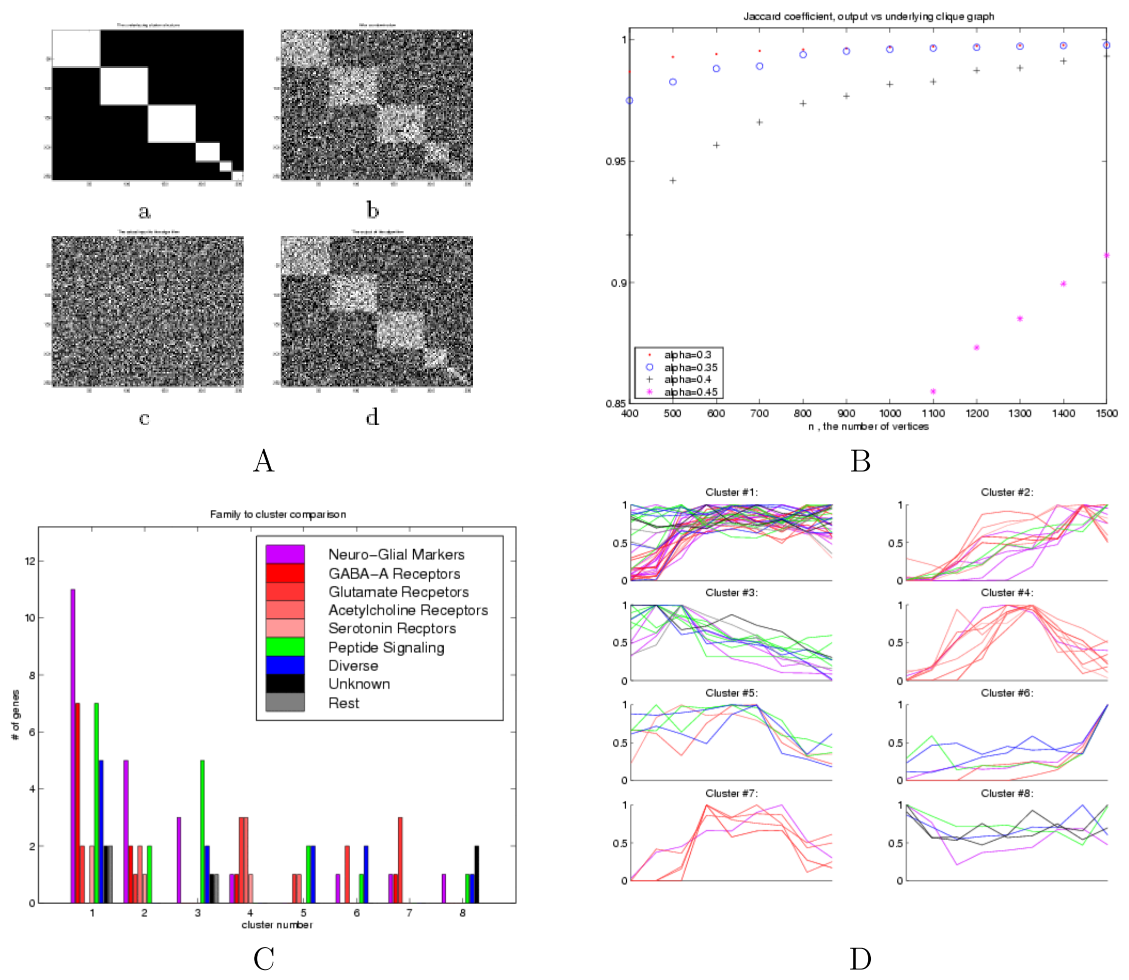

Figure 3.7: Source: [2]. A) A visualization of the simulation procedure. **a**: The adjacency matrix of the original clique graph H before introduction of errors. Position (i, j) is white if $(i, j) \in E(H)$, that is, if i and j belong to the same cluster. **b**: The same matrix after introduction of errors. Note that the cluster structure is still visible for all but the smallest clusters. **c**: The same as **b** but vertex order is randomly permuted. This is the actual input to the algorithm. **d**: Matrix **c** reordered according to solution produced by the algorithm. With the exception of perhaps the smallest clusters, the essential cluster structure is reconstructed. B) Simulation results for H with cluster structure of $\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}\}$. The x-axis is n, the number of vertices in H (clustered entities), and y-axis is the mean value of the Jaccard coefficient. Each curve corresponds to a specific probability $p = \alpha$ of contamination error. C) Applying the algorithm to temporal gene expression data [13]. The solution generated by the algorithm is compared to the prior classification. For each cluster (x-axis), bars composition in terms of biologically defined families. The height of each bar (y-axis) represents the number of genes of a specific cluster family. Most clusters contain predominantly genes from one or two families. D) Applying CAST to temporal gene expression data [13]. Each graph presents expression patterns of genes in a specific cluster. The x-axis represents time, while the y-axis represents normalized expression level.

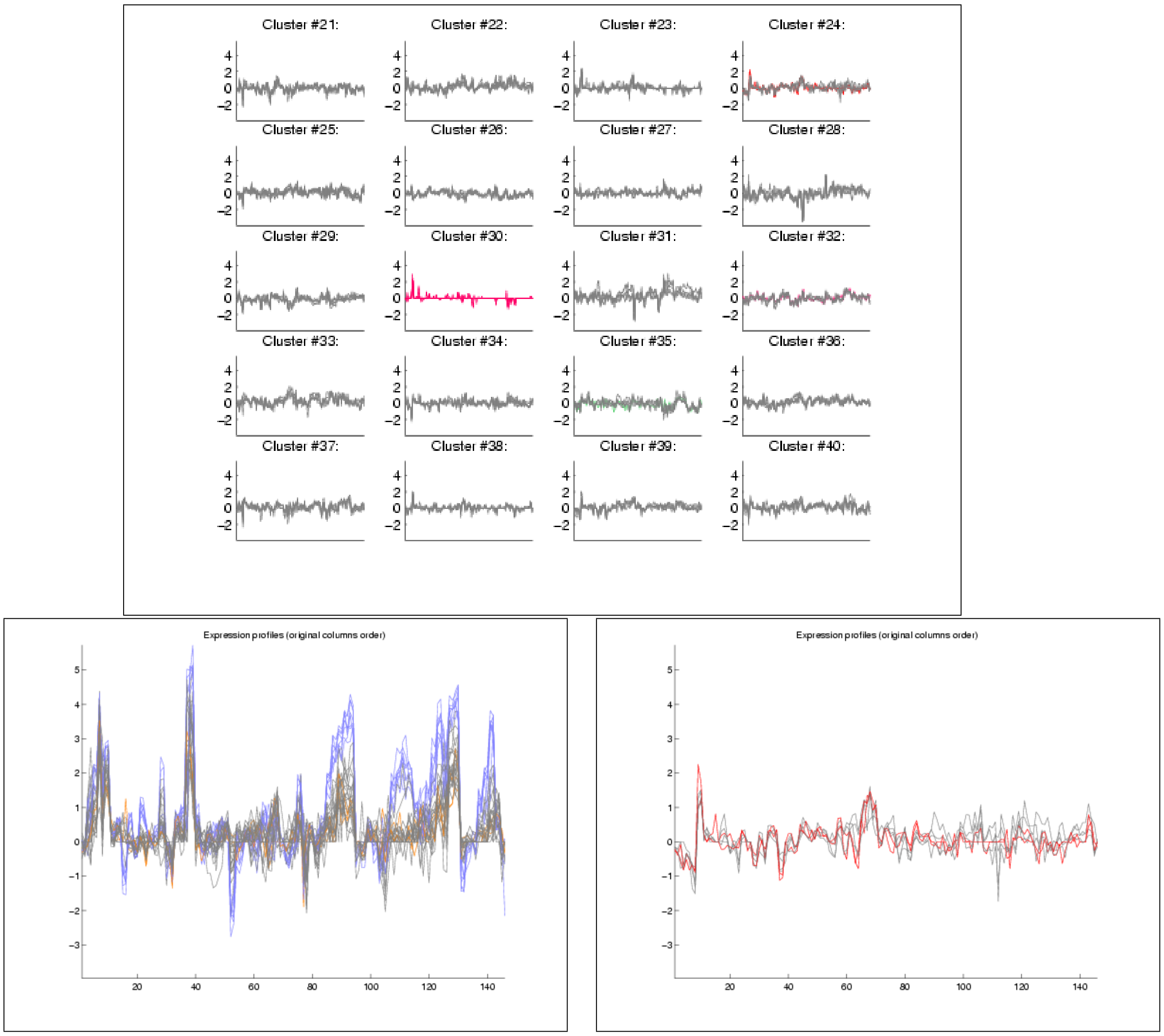

Figure 3.8: Source: [2]. Some results of the CAST algorithm applied to the nematode gene expression data of Kim *et al.* [5]. Top: expression patterns for clusters #21 to #40. x axis: conditions (matrix columns) in arbitrary order. y axis: intensity level. Most of the genes' functions are unknown, so only few genes are color coded. Blue: sperm genes; red: yeast genes (control) ; gray: unknown. Note the homogeneity of cluster #30. Bottom Left: expression patterns of the genes in cluster #1, consisting of 31 genes. Bottom Right: Expression patterns of the six genes in cluster #24. This cluster contains two growth related genes, lin15 and E2F. This suggests the hypothesis that the other four members of this cluster have related functions.

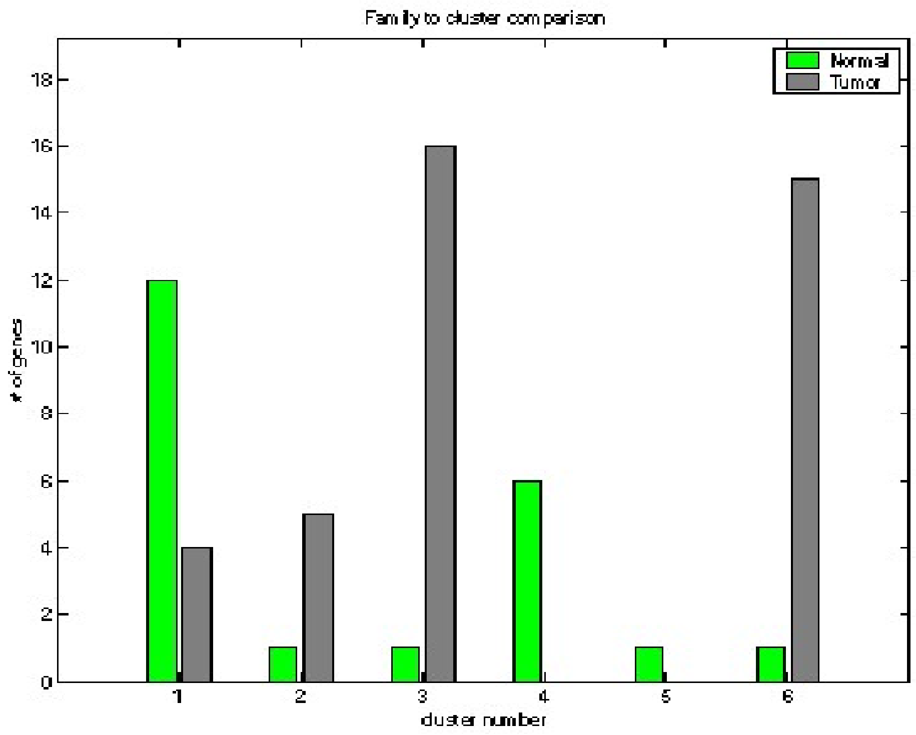

Figure 3.9: Source: [2]. Distribution of tumor and normal tissues in the six clusters produced by the CAST algorithm.

Bibliography

[1] 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.

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

[3] H. Chernoff. A measure of the asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[4] T. M. Cover and J. M. Thomas. *Elements of Information Theory.* John Wiley & Sons, London, 1991.

[5] S. Kim. Department of Developmental Biology, Stanform University, `http://cmgm.stanford.edu/∼kimlab/`.

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

[7] 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.

[8] A. Natanzon. Complexity and approximation of some graph modification problems. Master's thesis, Department of Computer Science, Tel Aviv University, 1999.

[9] R. Shamir and D. Tsur. Improved algorithms for the random cluster graph model. In *Proc. 8th Scandinavian Workshop on Algorithm Theory (SWAT '02)*, LNCS 2368, pages 230–239. Springer-Verlag, 2002.

[10] R. Sharan, A. Maron-Katz, N. Arbili, and R. Shamir. EXPANDER: EXPression ANalyzer and DisplayER, 2002. Software package, Tel-Aviv University, `http://www.cs.tau.ac.il/∼rshamir/expander/expander.html`.

[11] R. Sharan, R. Shamir, and D. Tsur. Cluster graph modification problems. *Discrete Appled Mathematics*, 144:173–182, 2004.

[12] 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.

[13] X. Wen, S. Fuhrman, G. S. Michaels, D. B. Carr, S. Smith, J. L. Barker, and R. Somogyi. Large-scale temporal gene expression mapping of central nervous system development. *PNAS*, 95(1):334–339, 1998.