Analysis of Gene Expression Data **Spring Semester, 2004**

Lecture 6: April 22, 2004

Lecturer: Ron Shamir *Scribe: Elena Zilberman and Igor Kotenkov*[1]

6.1 CLICK: Cluster Identification via Connectivity Kernels

6.1.1 Introduction

CLICK (CLuster Identification via Connectivity Kernels) is a new algorithm for clustering [13]. The input for CLICK is the gene expression matrix. Each row of this matrix is an "expression fingerprint" for a single gene. The columns are specific conditions under which gene expression is measured.

The CLICK algorithm attempts to find a partitioning of the set of elements into clusters, so that two criterias are satisfied: *Homogeneity* - fingerprints of elements from the same cluster, called *mates*, are highly similar to each other; and *Separation* - fingerprints of elements from different clusters, called *non-mates*, have low similarity to each other.

The goal is to identify highly homogeneous sets of elements - *connectivity kernels* (subsets of very similar elements) and add elements to kernels via similarity to average kernel fingerprints.

Uses tools from Graph Theory and probabilistic considerations.

6.1.2 Probabilistic Model

Mates - genes that belong to the same true cluster.

The CLICK algorithm makes the following assumptions:

1. Similarity values between mates are normally distributed with parameters μ_T, σ_T.
2. Similarity values between non-mates are normally distributed with parameters μ_F, σ_F.

[1]Based on scribes by Giora Sternberg, and Ron Gabor, May 2, 2002, and by Irit Gat and Amos Tanay, May 30, 2002

```
Basic-CLICK(G(V, E))
    if (V(G) = {v}) then
        move v to the singleton set R
    elseif (G is a kernel) then
        Output V(G)
    else
        (H, H̄, cut) ← MinWeightCut(G)
        Basic-CLICK(H)
        Basic-CLICK(H̄)
    end if
end
```

Figure 6.1: The Basic-CLICK algorithm

3. We expect that $\mu_T > \mu_F$, and σ_T, σ_F are small enough so we can separate the clusters.

These assumptions are justified empirically by simulations, and in some cases theoretically (by the Central Limit Theorem).

Parameters for the algorithm can be learned in two ways: from partially known solutions or estimated using EM algorithm.

6.1.3 The Basic CLICK Algorithm

The CLICK algorithm represents the input data as a weighted *similarity graph* $G = (V, E)$. In this graph vertices correspond to elements and edge weights are derived from the similarity values. The weight w_{ij} of an edge (i, j) reflects the probability that i and j are mates, and is set to be:

$$w_{ij} = \ln \frac{p\, f^M(S_{ij})}{(1-p)\, f^N(S_{ij})}$$

where $f^M(S_{ij})$ / $f^N(S_{ij})$ is the value of the probability density function for mates/non-mates at S_{ij}:

$$f^M(S_{ij}) = \frac{1}{\sqrt{2\pi}\sigma_T} e^{-\frac{(S_{ij}-\mu_T)^2}{2\sigma_T^2}}$$

and

$$f^N(S_{ij}) = \frac{1}{\sqrt{2\pi}\sigma_F} e^{-\frac{(S_{ij}-\mu_F)^2}{2\sigma_F^2}}$$

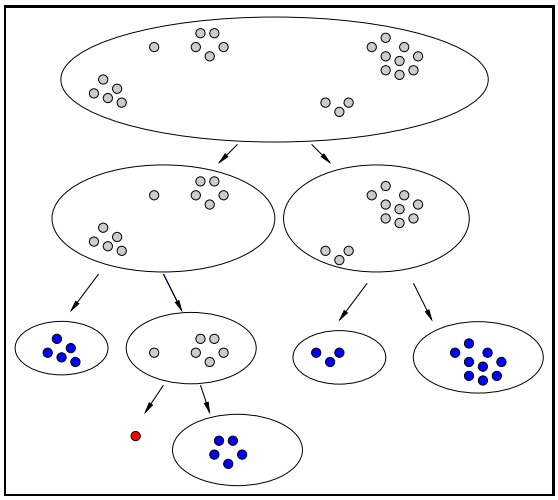

Figure 6.2: Basic scheme of the CLICK algorithm. Split subsets of G, that contain elements from two kernels.

The basic CLICK algorithm is described in Figure 6.1 and exemplified in Figure 6.2. The idea behind the algorithm is the following: given a connected graph G, we would like to decide whether $V(G)$ is a subset of some true cluster, or $V(G)$ contains elements from at least two true clusters. In the first case we say that G is *pure*. In order to make this decision, we test the following two hypotheses for each cut C in G:

- H_0^C: C contains only edges between non-mates.
- H_1^C: C contains only edges between mates.

G is declared a *kernel* if H_1 is more probable for all cuts. The decision of whether G is a kernel relies on the following theorem:

Theorem 6.1 *G is a kernel iff the weight of $MinCut(G) > 0$.*

Proof: At first, we will make two assumptions:

- $S_{i,j}$-s are independent
- mate relations are also independent

Then, using Bayes' Theorem, it can be shown that for any cut C in G

$$\begin{aligned}\ln\frac{Pr(H_1|C)}{Pr(H_0|C)} &= \ln\frac{Pr(H_1)f(C|H_1)}{Pr(H_0)f(C|H_0)} \\ &= \ln\frac{p^{|C|}\prod_{i,j\in C} f^M(S_{ij})}{(1-p)^{|C|}\prod_{i,j\in C} f^N(S_{i,j})} \\ &= \sum_{i,j\in C}\ln\frac{pf^M(S_{i,j})}{(1-p)f^N(S_{i,j})} \\ &= \sum_{i,j\in C} W_{ij} = W(C)\end{aligned}$$

Obviously, $W(C) > 0$ **iff** $Pr(H_1^C|C) > Pr(H_0^C|C)$. If the minimum cut is positive, then obviously so are all the cuts. Conversely, if the minimum cut is non-positive, then for that cut $Pr(H_1^C|C) \leq Pr(H_0^C|C)$, therefore G is not a kernel. ■

Removing Negative Weight Edges The MIN-CUT problem for a weighted graph with both positive and negative edges is NP-Complete[2]. In order to use the efficient MIN-CUT algorithms we must remove the negative edges. By modifying the algorithm slightly, we can still use the new graph to find kernels in the original graph.

Refinements

The Basic-CLICK algorithm divides the graph into kernels and singletons. These kernels are expanded to the full clustering, using several refinements:

Adoption Step In practice, "true" clusters are usually larger than just the kernel. To accommodate this, in the refined algorithm, kernels "adopt" singletons to create larger clusters. This is done by searching for a singleton v and a kernel K, whose pairwise fingerprint similarity is maximum among all pairs of singletons and kernels. The refined algorithm iteratively applies the adoption step and then the Basic-CLICK algorithm on the remaining singletons, stopping when there are no more changes.

Merge Step In this step we merge clusters whose fingerprints are similar (the justification for this is that, in practice, clusters can contain multiple kernels). The merging is done iteratively, each time merging two clusters whose fingerprint similarity is the highest (provided that the similarity exceeds a predefined threshold).

[2]MIN-CUT can be proved to be NP-Complete by reduction from MAX-CUT [5, page 210].

Quality Assessment

When the "correct" solution for the clustering problem is known, we can evaluate the algorithm's performance using comparison criteria as we have done with BioClust. The criteria used here are the Jaccard coefficient (as defined in the previous sections) and the Minkowski coefficient. The latter is defined by $\sqrt{\frac{n_{01}+n_{10}}{n_{11}+n_{10}}}$. Note that unlike the Jaccard and Matching coefficients, the Minkowski coefficient improves as it decreases, with optimal value at 0.

Unfortunately, in most cases the "correct" solution for the clustering problems is unknown. In these cases we evaluate the quality of the solution by computing two figures of merit to measure the *homogeneity* and *separation* of the produced clusters. For fingerprint data, homogeneity is evaluated by the average and minimum correlation coefficient between the fingerprint of an element and the fingerprint of its corresponding cluster. Separation is evaluated by the weighted average and the maximum correlation coefficient between cluster fingerprints. Formally:

Definition Homogeneity and Separation measures are defined as follows:
We define the fingerprint of a set of elements to be the mean vector of the fingerprints of the members of the set. Let $X_1, ..., X_t$ be clusters, $C(u)$ be the cluster of vertex u, $F(X)$ and $F(u)$ be the fingerprints of a cluster X and of element u respectively, and let $S(x, y)$ denote the similarity between fingerprints x and y, then:

Average Homogeneity

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

Minimum Homogeneity

$$H_{Min} = \min_{u \in N} S(F(u), F(C(u)))$$

Average Separation

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

Maximum Separation

$$S_{Max} = \max_{i \neq j} S(F(X_i), F(X_j))$$

Logically, a clustering improves when H_{Ave} and H_{Min} increase, and when S_{Ave} and S_{Max} decrease.

Another method of quality assessment is setting a certain similarity threshold and measuring the fraction of mates and non-mates above that threshold. Good clustering is expected to yield higher values of similarity between mates (indicating homogeneity) and lower values between non-mates (indicating separation).

Program (algorithm)	# Clusters	Homogeneity		Separation	
		H_{Ave}	H_{Min}	S_{Ave}	S_{Max}
CLICK	30	0.8	-0.19	-0.07	0.65
GENECLUSTER	30	0.74	-0.88	-0.02	0.97

Table 6.1: A comparison between CLICK and GENECLUSTER [15] on the yeast cell-cycle dataset [3]. Expression levels of 6,218 S. cerevisiae genes, measured at 17 time points over two cell cycles.

6.1.4 Algorithm Performance Comparisons

This section contains examples of comparisons between CLICK and other clustering algorithms, in various problems, including expression data, oligo-fingerprinting data and protein similarity data (Tables 6.1, 6.2, 6.3, 6.4, 6.5 and Figures 6.3, 6.4, 6.5). Analysis of the results (see Table 6.6) shows that CLICK outperforms all the compared algorithms in terms of quality. In addition, CLICK is very fast, allowing clustering of thousands of elements in minutes, and over 100,000 elements in a couple of hours on a regular workstation. Figure 6.6 shows the result of a comparison in which the authors of each clustering algorithm were allowed to run the test on their own. The graph shows a tradeoff between the homogeneity and separation scores; The further the algorithm is from the origin the "better" its overall performance.

In addition, CLICK was tested in simulations which included varying cluster structures and different distribution parameters. Similarity values for mates and non-mates were distributed normally: for each cluster structure, standard deviation σ was set at 5 for both mates and non-mates, while the difference between the means of mates μ_T and non-mates μ_F was set at $t \times \sigma$ for $t = 2, 1, 0.8, 0.6$. Results are shown in Table 6.7, evaluated using the Jaccard coefficient. As expected, the larger the distance between the means of mates and non-mates, the better the performance of the algorithm. It also seems that better results are obtained when cluster sizes are larger.

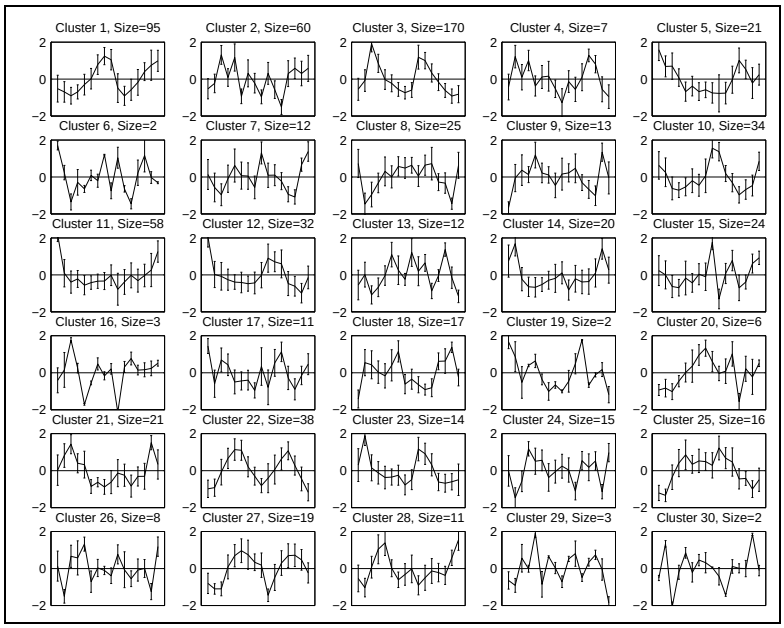

Figure 6.3: Source: [13]. CLICK's clustering of the yeast cell-cycle data [3]. x-axis: time points 0-80, 100-160 at 10-minute intervals. y-axis: normalized expression levels. The solid line in each sub-figure plots the average pattern for that cluster. Error bars display the measured standard deviation. The cluster size is printed above each plot.

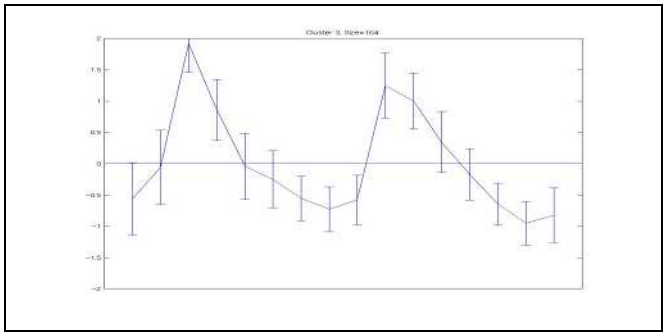

Figure 6.4: Yeast Cell Cycle: late G1 Cluster (cluster 3 from Figure 6.3). The cluster found by CLICK contains 91% of the late G1-peaking genes. In contrast, in GeneCluster 87% are contained in 3 clusters.

Program	#Clusters	#Singletons	Minkowski	Jaccard	Time(min)
CLICK	31	46	0.57	0.7	0.8
HCS	16	206	0.71	0.55	43

Table 6.2: A comparison between CLICK and HCS on the blood monocytes cDNA dataset [9]. 2,329 cDNAs purified from peripheral blood monocytes, fingerprinted with 139 oligos. Correct clustering is known from back hybridization with long oligos.

Program	#Clusters	#Singletons	Minkowski	Jaccard	Time(min)
CLICK	2,952	1,295	0.59	0.69	32.5
K-Means	3,486	2,473	0.79	0.4	–

Table 6.3: A comparison between CLICK and K-Means [10] on the sea urchin cDNA dataset. $20,275$ cDNAs purified from sea urchin eggs, and fingerprinted with 217 oligos. Correct clustering of 1,811 cDNAs is known from back hybridizations.

Program	#Clusters	Homogeneity		Separation	
		H_{Ave}	H_{Min}	S_{Ave}	S_{Max}
CLICK	10	0.88	0.13	-0.34	0.65
Hierarchical	10	0.87	-0.75	-0.13	0.9

Table 6.4: A comparison between CLICK and Hierarchical [4] clustering on the dataset of response of human fibroblasts to serum [11]. Human fibroblast cells starved for 48 hours, then stimulated by serum. Expression levels of 8,613 genes measured at 13 time points.

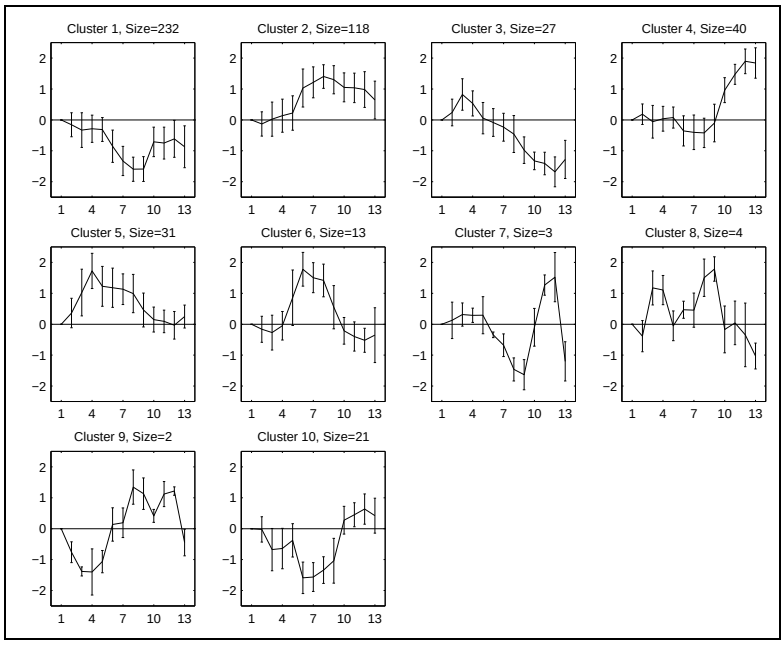

Figure 6.5: Source: [13]. CLICK's clustering of the fibroblasts serum response data [11]. x-axis: 1-12: synchronized time-points. 13: unsynchronized point. y-axis: normalized expression levels. The solid line in each sub-figure plots the average pattern for that cluster. Error bars display the measured standard deviation. The cluster size is printed above each plot.

Program	#Clusters	#Singletons	Homogeneity	Separation	Time(min)
CLICK	9,429	17,119	0.24	0.03	126.3
SYSTERS	10,891	28,300	0.14	0.03	–

Table 6.5: A comparison between CLICK and SYSTERS on a dataset of 117,835 proteins [12]. Measures based on similarity when no correct solution is known: For a fixed threshold t, homogeneity is the fraction of mates with similarity above t, and separation is the fraction of non-mates with similarity above t.

Elements	Problem	Compared to	Improvement	Time(min)
517	Gene Expression Fibroblasts	Cluster [4]	Yes	0.5
826	Gene Expression Yeast cell cycle	GeneCluster [15]	Yes	0.2
2,329	cDNA OFP Blood Monocytes	HCS [9]	Yes	0.8
20,275	cDNA OFP Sea urchin eggs	K-Means [10]	Yes	32.5
72,623	Protein similarity	ProtoMap [17]	Minor	53
117,835	Protein similarity	SYSTERS [12]	Yes	126.3

Table 6.6: A Summary of the time performance of CLICK on the above mentioned datasets. CLICK was executed on an SGI ORIGIN200 machine utilizing one IP27 processor. The time does not include preprocessing time. The "Improvement" column describes whether the solution of the CLICK algorithm was better than the compared algorithm.

Ataxia Telangiectasia

The following experiment shows how clustering methods can aid our understanding of biological processes. Its aim was to study the expression patterns of a genetic disease, Ataxia Telangiectasia (A-T).

A-T is a rare autosomal recessive disorder, characterized by cerebellar and thymic degeneration and predisposition to cancer. The gene found to be responsible for A-T is ATM - an important mediator of cell responses to DNA damage, in particular those that control

Structure	2	1	0.8	0.6
6×60	1	1	0.98	0.85
10×30	1	0.96	0.71	0.1
$10, ..., 80$	1	1	0.97	0.83

Table 6.7: CLICK simulation results (mean Jaccard score over 20 runs). The test included various cluster structures (rows) and distances between μ_T and μ_F (in each column, the distance appearing in the title was used as a factor of the standard deviation σ. The first column denotes a distance of 2σ, etc.).

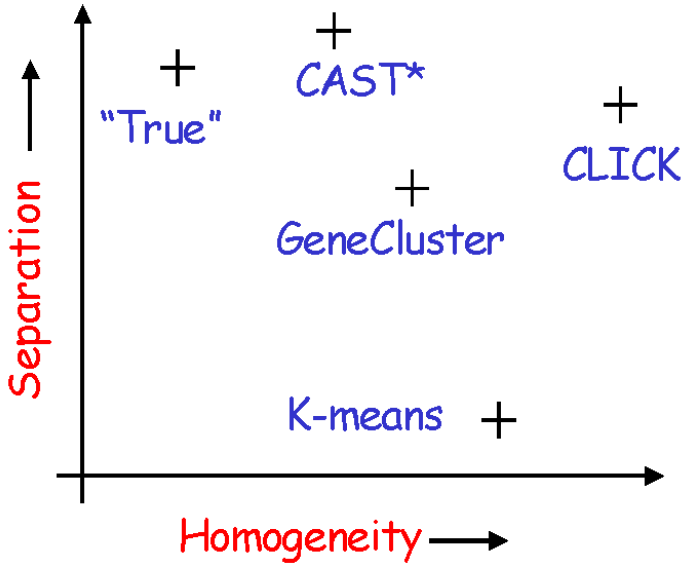

Figure 6.6: Comparison of clustering algorithms using homogeneity and separation criteria. The data consisted of 698 genes, 72 conditions [14]. Each algorithm was run by its authors in a "blind" test.

progression through the cell cycle. The study of A-T is facilitated by the existence of a mouse homologue of ATM.

In this experiment, DNA damage was induced both in normal mice and in ATM-deficient mice via irradiation. Tissues were then obtained in 3 time-points - 0, 30 and 120 minutes after irradiation - in order to check for differences in the cellular response. cDNA's from thymus, cerebellum and brain were hybridized to microarrays representing $8,000$ transcripts. Each gene was sampled at 18 combinations of tissue, genotype and time-point. The resulting data was then processed by the CLICK algorithm.

The results are shown in Figure 6.7. Interestingly, some of the clusters showed differences between the normal and ATM-deficient samples. These differences included constitutive expression of genes in ATM-deficient mice: whereas in normal mice a rise in the expression level of these genes was noted in the time-points after 0, in ATM-deficient mice the level was high already at 0 time, and did not change significantly afterwards. These results may suggest that these genes are involved in the response to the induced DNA damage, which is hampered in ATM-deficient mice.

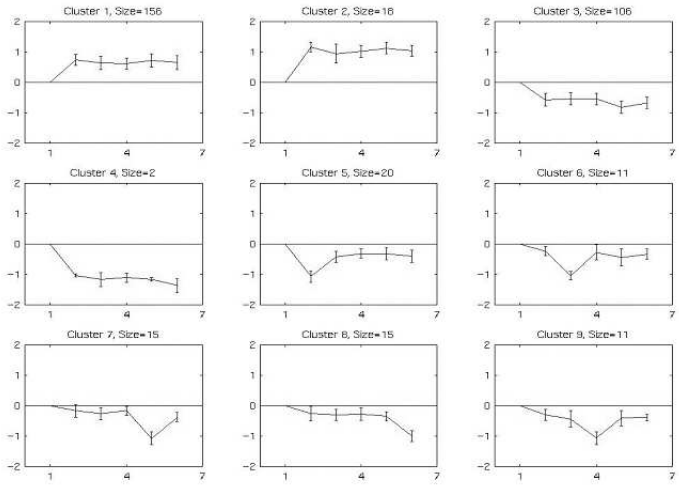

Figure 6.7: Clusters generated by the CLICK algorithm for the A-T experiment. y-axis: normalized expression levels. x-axis: 1-3: normal mice at times 0, 30, 120 (minutes); 4-6: ATM-deficient mice at times 0, 30, 120. Note that in cluster #1 we see constitutive expression of the genes in ATM-deficient mice at all three time-points, whereas in normal mice expression rises to about that level only after irradiation.

6.1.5 Clustering - Future Research

Clustering pops up in many questions and problems in various fields of interest, not only gene clustering. In many practical situations, the complexity of the clustering is the dominant part of the complexity of the whole solution. In many cases, though, clustering heuristics may not perform well and produce poor clustering results.

Therefore there is still room for improvement, both in theoretical clustering algorithms (and proofs), and in practical algorithms and heuristics. Some of the possible avenues for progress are indicated below.

Theoretical Clustering

- Reducing the restrictions on cluster sizes.
- Novel approaches and better algorithms, improving on both quality of results and time-complexity.

Practical Clustering

- Determining which algorithms and heuristics are best suited for particular clustering jobs.
- Improving on the optimization criteria.
- Better control over the tradeoff of accuracy versus speed, for instance using plausible assumptions about the clustered entities in each specific implementation.
- Devising new clustering/viewing tools, which would allow interactive setting of parameters and modifying algorithm behavior, while viewing the results.

6.2 Introduction to Biclustering

As we have seen in previous lectures, a very central method in the analysis of gene expression data is clustering. Clustering algorithms are used to transform a very large matrix of expression values to a more informative collection of gene or condition sets. The members of each cluster are assumed to share function or form some biological module. Clustering is a *global* technique and as such has several limitations. First, clustering partitions the elements into groups, so each element may appear in at most one group. Second, when clustering gene expression data, we group the genes according to their behavior over *all* experiments (similarly for conditions). This may be problematic when working with large databases that may include many different conditions, only few of which trigger some common gene behavior.

Suppose we are studying yeast cell biology. We may try to use gene expression data in order to identify functional modules, i.e., large sets of genes sharing some important cellular function or process. Clustering the genes may give good results as long as we are targeting a very concrete subsystem. Indeed several works ([14, 6]) used clustering to vastly expand our knowledge on cell cycle or stress response. However, many genes, even in our particular example, are important to both stress response and cell cycle as the two systems are intimately related, so clustering the joint data set would have to create some arbitrary partition of the two systems, loosing information on their common parts.

A second example for the limitations of clustering comes from clinical studies of cancer. We may cluster tissues of many patients suffering from several type of cancer in order to try and identify clinically important subclasses. We may also try to compare our classes with some additional information on the patients (age, sex, type of cancer, smoking years, prognosis). When using global clustering of the tissues we can only find one (hopefully the most significant) signal in the data, for example we may separate the different cancer types. Other signals, which may be important, will be missed since we are associating tissues with a single effect.

To try and address these shortcomings, the concept of **biclustering** was introduced to gene expression analysis. Biclustering was first defined in the seventies [8] and was applied to several domains before Cheng and Church [2] coined its usage in computational biology. Given a gene expression matrix, we search for submatrices that are tightly co-regulated according to some scoring criterion. We do not require the identified submatrices to be disjoint or to cover the entire matrix, instead we wish to build a diverse collection of submatrices that will capture all the significant signals in our data.

Before going into details, we state again the basic reasoning of using biclustring and the key differences between biclustering and standard clustering. Biclustering is a **local** technique by nature, i.e., we try to find local, significant signals in the data. Clustering, on the other hand, tries to model the whole dataset by reducing it to a collection of subsets. A successful collection of biclusters will provide a more detailed model of the data and can uncover more biological implications of it. However, biclustering results will be harder to interpret.

6.3 Cheng and Church's Algorithm

The first application of biclustering to gene expression data was the work of Cheng and Church (2000). Stating its goal as the ability to find signals more delicate than clusters, the methodology is based on a simple uniformity goal (the Mean Residue Score, defined below) and uses a greedy algorithm to find one bicluster, combined iteratively to produce a collection of biclusters.

6.3.1 The Algorithm

We denote the input matrix of expression data as $A = (a_{ij})$ and the rows (columns) set by R (C). A submatrix is denoted by $A_{IJ}(I \subseteq R, J \subseteq C)$ and we use the auxiliary notation $a_{Ij} = \frac{\sum_{i \in I} a_{ij}}{|I|}$ (sub column average) $a_{iJ} = \frac{\sum_{j \in J} a_{ij}}{|J|}$ (sub row average) and $a_{IJ} = \frac{\sum_{i \in I, j \in J} a_{ij}}{|I||J|}$ (submatrix average).

We define the *Residue Score* of an element a_{ij} in a submatrix A_{IJ} as $RS_{IJ}(i,j) = a_{ij} - a_{Ij} - a_{iJ} + a_{IJ}$ and the *Mean Residue Score* of the submatrix as $H(I,J) = \sum_{i \in I, j \in J} \frac{RS_{ij}^2}{|I||J|}$. A completely uniform matrix will score zero. A submatrix in which all entries are the sum of a column parameter and a row parameter ($a_{ij} = b_i + c_j$) would also score zero. On the other hand a random submatrix (normally distributed with any parameter) would have the variance of the distribution as its expected score. We define a δ bicluster to be a submatrix (I,J) for which $H(I,J) \leq \delta$. The biclustering algorithm will search for a δ-bicluster assuming that the parameter δ was chosen appropriately to avoid random signal identification. For example, we may choose δ as the minimal (i.e. best) score of the output of a clustering algorithm.

The optimization problem of identifying the largest δ-bicluster (the one for which $|I| = |J|$ is the largest) is NP hard as can be seen by a simple reduction from BALANCED COMPLETE BIPARTITE SUBGRAPH. We are thus interested in heuristics for finding a large δ bicluster in reasonable time. We next present such heuristic which is a greedy algorithm in essence, show how to speed it up and use it as a subroutine for finding many biclusters.

A naive greedy algorithm for finding a δ-bicluster may start with the entire matrix and at each step try all single rows/columns addition/deletion, applying the best operation if it improves the score and terminating when no such operation exists or when the bicluster score is below δ. However, simply recalculating all averages and mean residues for each operation may be too expensive for large matrices. Cheng and Church's algorithm uses the structure of the mean residue score to enable faster greedy steps. The idea is based on the following lemma:

Lemma 6.2 *The set of rows that can be completely or partially removed with the net effect of decreasing the mean residue score of a bicluster A_{IJ} is :*

$$R = \{i \in I; \frac{1}{|J|} \sum_{j \in J} RS_{I,J}(i,j) > H(I,J)\}$$

In words, it is safe to remove any row for which the average contribution to the total score is greater then its relative share. The same argument is correct for columns and gives rise to the following greedy algorithm that iteratively remove rows/columns with the maximal average residue score (Figure 6.8).

Input: Expression matrix A on genes S, conditions C and a parameter δ.
Output: A_{IJ} a δ-bicluster.
Init: $I = S$,$J = C$.
Iteration:
Calculate a_{Ij}, a_{iJ} and $H(I,J)$. If $H(I,J) \leq \delta$ output I, J.
For each row calculate $d(i) = \frac{1}{|J|}\sum_{j \in J} RS_{I,J}(i,j)$.
For each column calculate $e(j) = \frac{1}{|I|}\sum_{i \in I} RS_{I,J}(i,j)$.
Take the best row or column and remove it from I or J.

Figure 6.8: Single node deletion algorithm.

Note that since a 1 by 1 submatrix is always a δ-bicluster we should hope that the deletion algorithm will terminate with a large bicluster. It is natural to try and add rows/columns in an analogous way, using the equivalent lemma and algorithm (Figure 6.9):

Lemma 6.3 *The set of rows (columns) that can be completely or partially added with the net effect of decreasing the score of a bicluster A_{IJ} is :*

$$R = \{i \notin I; \frac{1}{|J|}\sum_{j \in J} RS_{I,J}(i,j) \leq H(I,J)\}$$

Input : Expression matrix A, the parameter δ and I, J specifying a δ bicluster.
Output : AI', J' - a δ-bicluster with $I \subseteq I', J \subseteq J'$.
Iteration
Calculate a_{Ij}, a_{iJ} and $H(I,J)$.
Add the columns with $\frac{1}{|I|}\sum_{i \in I} RS_{I,J}(i,j) \leq H(I,J)$.
Calculate a_{Ij}, a_{iJ} and $H(I,J)$.
Add the rows with $\frac{1}{|J|}\sum_{j \in J} RS_{I,J}(i,j) \leq H(I,J)$.
If nothing was added, halt.

Figure 6.9: Node addition algorithm.

The exact details of the heuristic are not necessarily optimal for any situation. For example, the algorithm presented here is tailored for cases where there are much more rows than columns.

The Cheng-Church algorithm suggests two additional improvements to the basic deletion-addition algorithm. The first improvement suggests a **multiple node deletion** in cases where the data set is large. This is done by removing at each deletion iteration all rows/columns for which $d(i) > \alpha H(I, J)$ for some choice of α. The idea is to perform large steps until the submatrix is relatively small and indeed it is shown that such steps can be done safely (without increasing the score).

The second algorithmic improvement involves the addition of **inverse** rows to the matrix, allowing the identification of biclusters which contains co-regulation and inverse co-regulation (i.e., cases where two genes always change in opposite directions).

As mentioned in the introduction, the goal of a biclustering algorithm is to identify all (or many of) the signals in the data set, so clearly, finding one bicluster is not enough. The Cheng-Church solution to this requirement uses the δ-bicluster algorithm as a subroutine and repeatedly applies it to the matrix. In order to avoid finding the same bicluster over and over again, the discovered bicluster is masked away from the data, by replacing the values of its submatrix by random values. The general biclustering scheme is outlined in Figure 6.10.

Input : Expression matrix A, the parameter δ, the number of biclusters to report n
Output : n δ-biclusters in A.
Iteration
Apply multiple node deletion on A giving I', J'.
Apply node addition on I', J' giving I'', J''.
Store I'', J'' and replace $A_{I'',J''}$ values by random numbers.

Figure 6.10: Cheng-Church biclustering algorithm.

6.3.2 Experiments

We next describe some of the experiments done by Cheng and Church to validate their approach. Experiments were done using two datasets, one of human lymphoma ([1]) and the other of yeast data ([16]). Working with the yeast data, the parameter δ was chosen to be a bit more then the minimal score of the reported clusters. A large set of random submatrices of varying sizes was then scored and compared to the selected threshold. The simulation showed that small δ-biclusters have a considerable chance of being random (15% for 3 by 6 matrices, 0.06% for 10 by 6 matrices), but larger $\delta - biclusters$ may be far from random (although random here referees to random submatrices rather than random expression data).

Application of the algorithm to both datasets produced 100 biclusters on each, and some effort was made to test their relation with global clustering and biological information.

For example, the hierarchical clustering of the lymphoma tissues was compared with the biclusters by testing the condition sets of each bicluster and its partition among the two main branches of the hierarchy.

6.4 Coupled two-way clustering

6.4.1 Motivation.

The results of every gene microarray experiment are organized in an *expression level matrix* $\mathscr{A}$. A row of this matrix corresponds to a single gene, while each column represents a particular sample. In a typical experiment simultaneous expression levels of thousands of genes are measured.

Gene expression is influenced by the cell type, cell phase, external signals, and more. The expression level matrix is therefore the result of all these processes mixed together. Our goal is to separate and identify these processes and to extract as much information as possible about them. The main difficulty is that each biological process on which we wish to focus may involve a relatively small subset of the genes; the large majority of those present on the microarray constitute a noisy background that may mask the effect of the small subset. The same may happen with respect to samples. A straightforward approach to finding *pairs* of subsets, $(\mathscr{O}_j, \mathscr{F}_i)$ of gens and samples that lead to "meaningful" clusters, could be to take all possible submatrices of the original data and apply the standard (uncoupled) two-way clustering procedure to every one of them. By keeping track of all stable clusters that are formed in this process, and storing the identity of both genes and samples that define the particular submatrix, one is guaranteed to find every possible stable partition in the data. This approach is, of course, impossible to implement, because the number of such submatrices grows exponentially with the size of the problem. CTWC provides an efficient heuristic to generate such pairs of object and feature subsets by an iterative process that severely restricts the possible candidates for such subsets; we consider and test only those submatrices whose rows (columns) belong to genes (samples) that were identified (in a previous iteration!) as a stable cluster. [7]

6.4.2 The algorithm.

Coupled two-way clustering defines a generic transformation from a one-dimensional clustering algorithm into a biclustering algorithm. The algorithm relies on having a one-dimensional (standard) clustering algorithm that can discover significant (*stable*) clusters. Given such an algorithm, the coupled two-way clustering procedure will recursively apply the one-dimensional algorithm to submatrices, aiming to find subsets of genes giving rise to significant clusters of conditions, or subsets of conditions giving rise to significant gene clusters.

The submatrices defined by such pairing are called *stable submatrices* and correspond to biclasters. The algorithm operates on a set of gene subsets $\mathcal{V}$ and a set of condition subsets $\mathcal{U}$. Initially $\mathcal{V} = \{V\}$ and $\mathcal{U} = \{U\}$.

The algorithm then iteratively selects a gene subset $V' \in \mathcal{V}$ and a condition subset $U' \in \mathcal{U}$ and applies the one dimensional clustering algorithm twice, to cluster V' and U' on the submatrix $U' \times V'$. If stable clusters are detected, their genes/conditions subsets are added to the respective sets $\mathcal{V}$, $\mathcal{U}$. The process is repeated until no new stable clusters can be found. The implementation makes sure that each pair of subsets is not encountered more than once.

Note that the procedure avoids the consideration of all rows and columns subsets, by starting from an established row subset when forming subclusters of established column subsets, and vica versa. The success of the coupled two-way clustering strategy depends on the performance of the given one-dimensional clustering algorithm. We note that many popular clustering algorithms (e.g. K-means, Hierarchical, SOM) cannot be plugged into the coupled two-way machinery, as they do not readily distinguish significant clusters from non-significant clusters or make a-priori assumption on the number of clusters. It has been reported by Getz et al. using SPC clustering algorithm is having good results. The results of the algorithm can be viewed in a hierarchical form: each stable gene (condition) cluster is generated given a condition (resp. gene) subset. This hierarchical relation is important when trying to understand the context of joint genes or conditions behavior. For example, when analyzing clinical data, Getz et al. have focused on gene subsets giving rise to stable tissue clusters that are correlative to known clinical attributes. Such gene sets may have an important biological role in the disease under study.

6.4.3 Choosing one-dimentional algorithm for CTWC

- Any reasonable clustering method can be used within the framework of CTWC, but the optimal algorithm should have the following properties: the number of clusters should be determined by the algorithm itself and not externally prescribed [as is done when using self-organizing maps (SOMs) and K-means] ; stability against noise; generating a hierarchy (dendrogram) and providing a mechanism to identify in it robust stable clusters; and ability to identify a dense set of points, which form a cloud of an irregular nonspherical shape, as a cluster., also it must be limited to nested row & column subsets, but not limited to nested submatrices

- Cluster stability is measured wrt parent bicluster

- Two biclusters are either disjoint or one is contained in the other.

- No. of (bi)clusters not prespecified

"Super Paramagnetic Clustering" (SPC)

SPC, a hierarchical clustering method recently introduced by Blatt, is the algorithm that best fits these requirements. The intuition that led to it is based on an analogy to the physics of inhomogeneous ferromagnets.

The input for SPC is a distance or similarity matrix d_{ij} between the objects $\mathscr{O}$, calculated according to the feature set $\mathscr{F}$. A tunable parameter T ("temperature") controls the resolution of the performed clustering. One starts at $T = 0$, with a single cluster that contains all the objects. As T increases, phase transitions take place, and this cluster breaks into several subclusters that reflect the structure of the data.

At each T, probability of two objects having the same label is measured and if it is high enough, they are given the same label.

Clusters keep breaking up as T is further increased, until at high enough values of T each object forms its own cluster. As opposed to most agglomerative algorithms, SPC has a natural measure for the relative stability of any particular cluster: the range of temperatures, ΔT, over which the cluster remains unchanged. The more stable a cluster is, the larger the range ΔT through which it is expected to "survive". For a stable cluster s, the corresponding ΔT_s constitutes a significant fraction of T_{max}, the temperature at which the data break into single-point clusters. Inspection of the gene dendrograms of Fig. 6.12 reveals stable clusters and stable branches. The choice of the value ΔT_c, above which a cluster is considered as stable, in the following way. We permuted at random elements of the expression matrix under investigation, and applied SPC to the randomized matrix. ΔT_c was selected so that for 500 different random permutations no clusters that survived for $\Delta T > \Delta T_c$ were found. This gives a bound on the probability that clusters that we labeled as stable were in fact an artifact of noisy data.

TWOWAY(U, V, E, ALG):
U : conditions. V : genes.
E : Gene expression matrix.
ALG : one-dimensional clustering algorithm. Inputs a matrix and outputs significant (stable) clusters of columns or rows
Initialize a hash table *weight*
Initialize $\mathcal{U}_1 = \{U\}$, $\mathcal{V}_1 = \{V\}$
Initialize $\mathcal{U} = \emptyset$, $\mathcal{V} = \emptyset$
Initialize the sets hierarchy table H_V storing for gene clusters the condition subsets used to generate them.
Initialize the sets hierarchy table H_U storing for condition clusters the gene subsets used to generate them.
While ($\mathcal{U}_1 \neq \emptyset$ or $\mathcal{V}_1 \neq \emptyset$) do
    Initialize empty sets $\mathcal{U}_2, \mathcal{V}_2$.
    For all $(U', V') \in (\mathcal{U}_1 \times \mathcal{V}_1) \cup (\mathcal{U}_1 \times \mathcal{V}) \cup (\mathcal{U} \times \mathcal{V}_1)$ do
        Run $ALG(E_{U'V'})$ to cluster the genes in V':
            Add the stable gene sets to $\mathcal{V}_2$
            Set $H_V[V''] = U'$ for all new clusters V''.
        Run $ALG(E_{U'V'})$ to cluster the conditions in U':
            Add the stable condition sets to $\mathcal{U}_2$
            Set $H_U[U''] = V'$ for all new clusters U''.
    Assign $\mathcal{U} = \mathcal{U} \cup \mathcal{U}_1$, $\mathcal{V} = \mathcal{V} \cup \mathcal{V}_1$
    Assign $\mathcal{U}_1 = \mathcal{U}_2$, $\mathcal{V}_1 = \mathcal{V}_2$
Report $\mathcal{U}, \mathcal{V}$ and their hierarchies H_U, H_V.

Figure 6.11: Coupled two-way clustering.

Bibliography

[1] A.A. Alizadeh et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[2] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proc. ISMB'00*, pages 93–103. AAAI Press, 2000.

[3] RJ. Cho et al. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell*, 2:65–73, 1998.

[4] 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.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco, 1979.

[6] A. P. Gasch et al. Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell*, 11:4241–57, 2000.

[7] Gad Getz, Erel Levine, and Eytan Domany. Coupled two-way clustering analysis of gene microarray data *PNAS*, 97:12079–12084, 2000.

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

[9] 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.

[10] R. Herwig, A.J. Poustka, C. Muller, C. Bull, H. Lehrach, and J. O'Brien. Large-scale clustering of cDNA-fingerprinting data. *Genome Research*, 9:1093–1105, 1999.

[11] V.R. Iyer, M.B. Eisen, D.T. Ross, G. Schuler, T. Moore, J.C.F. Lee, J.M. Trent, L.M. Staudt, J. Hudson Jr., M.S. Boguski, D. Lashkari, D. Shalon, D. Botstein, and P.O. Brown. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283 (1), 1999.

[12] A. Krause, J. Stoye, and M. Vingron. The systers protein sequence cluster set. *Nucleic Acids Research*, 28(1):270–272, 2000.

[13] R. Sharan and R. Shamir. Click: A clustering algorithm with applications to gene expression analysis. In *Proceedings of the 8th Annual International Conference on Intelligent Systems for Molecular Biology, (ISMB '00)*, pages 307–316. AAAI Press, 2000.

[14] 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.

[15] 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.

[16] S. Tavazoie, JD. Hughes, MJ. Campbell, RJ. Cho, and GM. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22:281–285, 1999.

[17] G. Yona, N. Linial, and M. Linial. Protomap: Automatic classification of protein sequences, a hierarchy of protein families, and local maps of the protein space. *Proteins: Structure, Function, and Genetics*, 37:360–378, 1999.