Profile matrix analyzer : clustering microarray data

by Julie Lynn Taubman

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University

Abstract:
Biologists seeking to elucidate genetic regulatory mechanisms are using microarray experiments to observe gene expression levels for many genes simultaneously. By repeating these experiments over time scientists can trace gene expression profiles for thousands of genes. Clustering the profiles is a method that has been used to identify co-regulated genes and gain insight into the complex biological mechanisms described within this data.

As a further step in this investigative process, we translate the expression profiles of cluster centers to binary values so that a center is labeled either on or off at a given time. These binary profiles can then be input to a program, the Partitioned Markov Hypercube (PMH), to find probabilistic genetic regulatory networks.

In order to automate this process and compare different clustering techniques we developed a tool that allows a user to perform these steps and view the results through a graphical interface. The tool, Profile Matrix Analyzer (PMA), offers users' a choice of preprocessing and clustering methods, and then converts cluster centers to binary in order to run the data through the PMH program to find probabilistic regulatory networks. In addition, PMA allows the user to view expression profiles and their subsequent binary conversion. Finally, PMA is able to compare the different clusterings and evaluate them statistically.

PROFILE MATRIX ANALYZER: CLUSTERING MICROARRAY DATA

by

Julie Lynn Taubman

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 2001

APPROVAL

of a thesis submitted by

Julie Lynn Taubman

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Brendan Mumey Brendan Mumey 11/19/2001
(Signature) Date

Approved for the Department of Computer Science

Denbigh Starkey J Denbigh Starkey 11/19/2001
(Signature) Date

Approved for the College of Graduate Studies

Bruce McLeod Bruce R. McLeod 11-26-01
(Signature) Date

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U. S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature Julie Laubman

Date 11/11/01

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Biologists seeking to elucidate genetic regulatory mechanisms are using microarray experiments to observe gene expression levels for many genes simultaneously. By repeating these experiments over time scientists can trace gene expression profiles for thousands of genes. Clustering the profiles is a method that has been used to identify co-regulated genes and gain insight into the complex biological mechanisms described within this data.

As a further step in this investigative process, we translate the expression profiles of cluster centers to binary values so that a center is labeled either on or off at a given time. These binary profiles can then be input to a program, the Partitioned Markov Hypercube (PMH), to find probabilistic genetic regulatory networks.

In order to automate this process and compare different clustering techniques we developed a tool that allows a user to perform these steps and view the results through a graphical interface. The tool, Profile Matrix Analyzer (PMA), offers users a choice of preprocessing and clustering methods, and then converts cluster centers to binary in order to run the data through the PMH program to find probabilistic regulatory networks. In addition, PMA allows the user to view expression profiles and their subsequent binary conversion. Finally, PMA is able to compare the different clusterings and evaluate them statistically.

CHAPTER 1

INTRODUCTION

Biological processes in living cells are orchestrated by the organism's genome. Subsets of genes are expressed and translated into gene products at different times. Some genes are responsible for the activation of other genes, thus constituting a part of the control mechanism for gene expression.

Microarray technology is a recent advance in biotechnology that allows researchers to measure the expression levels of thousands of genes simultaneously. The technology hinges upon a central characteristic of DNA, that complementary single stranded sequences will anneal or *hybridize* to each other. A typical microarray experiment begins by extracting sample mRNA from a cell. The mRNA is then copied into cDNA and a fluorescent tag is attached. The sample is washed over a microarray chip which contains a matrix of immobilized single stranded DNA or *oligonucleotides*. A fluorescent scan reveals where the sample DNA hybridized. Studies have shown that the fluorescence value is directly proportional to the abundance of the nucleic acid to within a factor of 2.5 [26]. In one variation of the microarray experiment, a control sample is fluorescently labeled with another dye and the ratio of the sample to control fluorescence is obtained. (Figure 1) Therefore, by repeating assays over time and/or

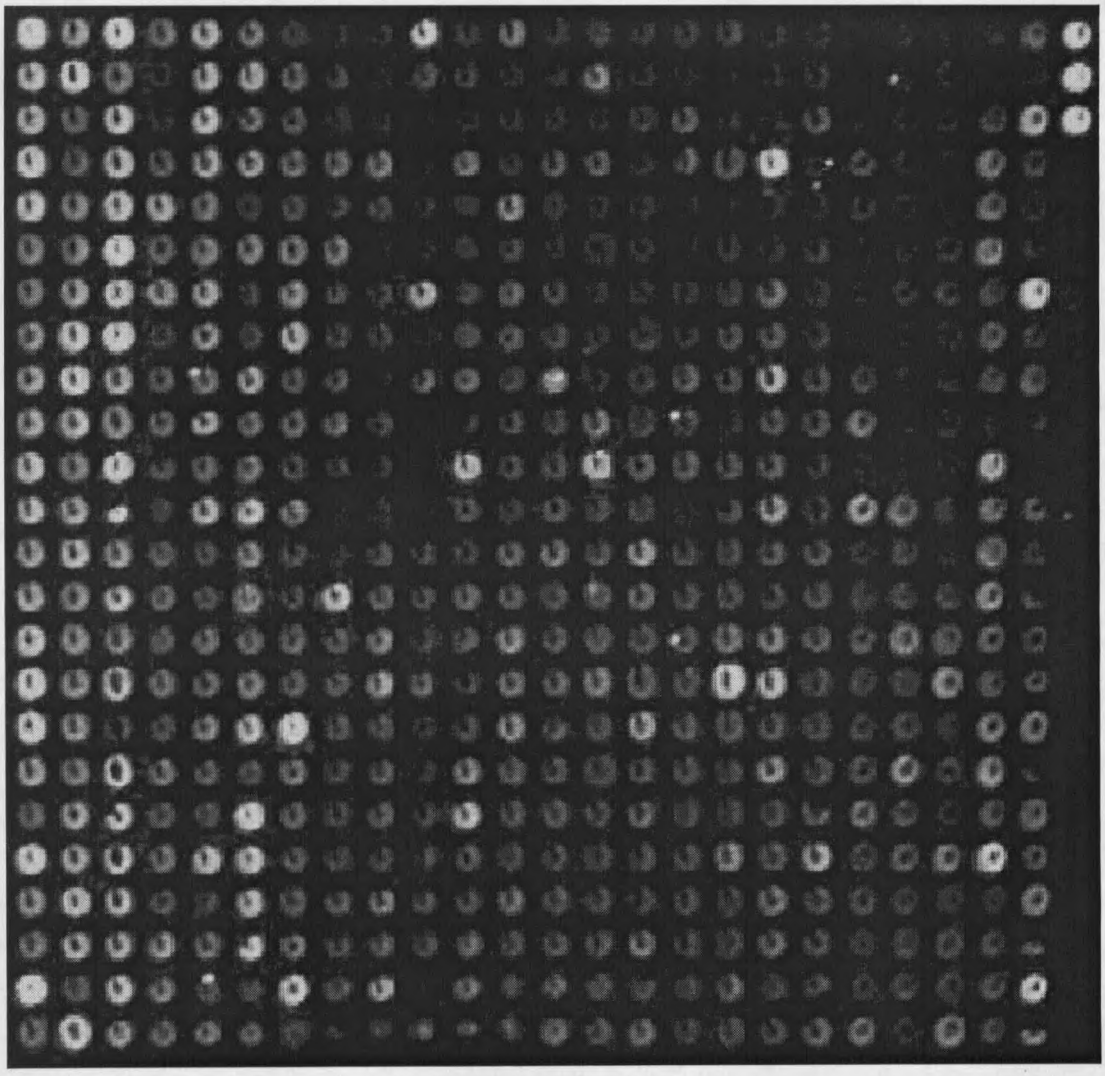

Figure 1. Microarray Image [1] .

varying conditions, expression profiles for thousands of genes are observed.

Clustering is the problem of assigning n elements, each represented by a vector, to K clusters. For gene expression data, the elements to cluster are the gene expression profiles. Clustering gene expression profiles can shed light on underlying complex biological processes by characterizing gene function and identifying co-regulated genes [9, 2]. In the common yeast, *Saccharomyces cerevisiae*, average linkage clustering grouped genes with known similar functions [9]. On another data set, average linkage clustering identified genes with similar 5'[1] regions, thus providing evidence that the genes within a cluster shared common promoter elements [21].

Our work focused on the yeast *Saccharomyces cerevisiae* and three data sets. These data sets were time series experiments where a typical matrix of data included several experiments. Each experiment involved exposing a cell to various conditions, such as temperature changes, and recording mRNA expression levels for usually less than 20 time points. We developed a program, the Profile Matrix Analyzer (PMA), in order to analyze the results of different clustering techniques applied to this data. We used the k-means, average linkage, taxmap and self-organizing maps clustering methods and used two different similarity metrics, Euclidean distance and a form of the correlation coefficient, and then compared the results of these different methods. The PMA program also has the ability to convert cluster expression profiles into

[1] the two ends of genes are called 5' and 3', the 5' region has promoter elements which help control transcription

binary values by classifying an expression level as either "on" or "off". These binary profiles can then be input to a program that finds probabilistic regulatory elements using a novel approach called the Partitioned Markov Hypercube (PMH). In the current work we compare the results of different clustering methods in order to obtain the best clustering results to be input to PMH.

CHAPTER 2

CLUSTERING METHODS

Clustering is the problem of partitioning n patterns into k clusters so that the members of a cluster are similar in some way. There are several ways to formally define the clustering problem. One way is to view it as an optimization problem. The input to the problem are the elements to be clustered $S_1, \ldots, S_n$ drawn from a d-dimensional metric space, an integer k defining the number of clusters, and a cost function, c, that associates a cost with each cluster, C_i. The goal of the clustering algorithm is to minimize $\sum_{i=1}^{k} c(C_k)$. A common cost function is the *sum-of-squares* criteria. This definition of the problem is known to be NP-hard [11].

The input to the clustering problem is either *fingerprint* data or *similarity* data. Fingerprint data is a vector describing a pattern that is associated with each element and contains a number of measurements for the element. For expression data, this fingerprint vector is usually expression levels of mRNA at different conditions. Similarity data is a matrix of pairwise similarity values between elements.

Clustering is an unsupervised method of classification, and differs from discriminant analysis because there are no cluster center, or centroid, patterns known *a priori*. Most clustering algorithms will find clusters in data whether they truly exist or not.

Therefore, clustering solutions must be evaluated to determine if clusters have similar properties and exhibit internal cohesion and external isolation.

Clustering methods applied to gene expression data have successfully partitioned an input set of expression profiles into sets of co-regulated genes [9]. These methods offer a promising approach to discovering valuable information about this data. We have developed a program, Profile Matrix Analyzer (PMA), and implemented the k-means, average linkage, taxmap, and self-organizing maps clustering algorithms using either Euclidean distance or a type of correlation coefficient as similarity metrics. In this chapter we will review common similarity metrics, clustering methods and algorithms and their application to microarray data, as well as some cluster evaluation techniques.

Similarity Metrics

All clustering algorithms require a means of determining the similarity or distance between two elements. The only input to some algorithms is this proximity matrix describing the pairwise similarity or dissimilarity between elements. Typically, similarity or distance is a symmetric relationship.

The two most common metrics used in clustering gene expression data are a form of the correlation coefficient, called the *gene similarity metric* [9] and Euclidean distance.

For any two genes X and Y observed over a series of N conditions, the gene similarity metric is computed by

$$S(X,Y) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{X_i - X_{offset}}{\Phi_X}\right)\left(\frac{Y_i - Y_{offset}}{\Phi_Y}\right)$$

where

$$\Phi_G = \sqrt{\sum_{i=1}^{N}\frac{(G_i - G_{offset})^2}{N}}$$

G_{offset} can be set to the mean of observations of G, so that Φ_G becomes the standard deviation of G. G_{offset} can be set to other values to represent a reference state against which changes are analyzed. In order to fix the control sample as the reference state G_{offset} is set to 0 to correspond to a fluorescence ratio of 1.0, where the control and test samples are exhibiting the same level of expression.

A third metric, information entropy, has been used with the FITCH software [12]. To compute the information entropy for each expression profile the expression values are first discretized into a number of equidistant bins. For genes I and J, the information entropy, $H(I)$ and $H(J)$, and mutual information, $M(I,J)$ are computed from the probabilities, $P(i)$, of the occurrence of a member of one of the bins:

$$H(I) = -\sum[P(i)\cdot\log P(i)]$$

$$H(I,J) = P(i,j)\cdot\log\frac{P(i,j)}{P(i)\cdot P(j)}$$

$$M(I,J) = H(I) + H(J) - H(I,J)$$

The normalized mutual information, $M_{norm}(I, J) = \frac{M(I,J)}{max H(I), H(J)}$ is a measure of pairwise similarity between two gene expression series.

One study [27] developed a metric for evaluating clusters called the *Figure of Merit* (FOM). They used the FOM to compare three similarity metrics, the correlation coefficient, Euclidean distance, and information entropy with two algorithms, CAST [2] and an iterative algorithm on microarray data sets. They found similar FOM's for both algorithms using either the correlation coefficient or Euclidean distance, with slightly worse performance when information entropy was used. The study also compared the effect of discretizing expression values for the mutual information metric into ten bins versus three and found no significant difference on the FOM.

A fourth similarity measure is the dot-product. As a result of some normalization schemes each vector will have the same average. If these p dimensional profiles are viewed as points in the Euclidean space, then the points lie on a p-dimensional sphere and the dot-product is equal to the product of each vector's magnitude and the cosine of the angle between them. If the vectors have a mean of zero and a variance of 1, the dot-product of two vectors equals their correlation coefficient [24].

Hierarchical Methods

Hierarchical clustering produces a *dendrogram* on the input set. Nodes on the tree are subsets of the input set and the union of nodes at a given level is the input

set. The levels in the tree are produced by a series of iterative partitions. At each iteration, clusters are merged in the agglomerative bottom-up method or divided in the divisive top-down method. Under the more common agglomerative method, all elements begin in their own cluster and the two most similar clusters are merged at each step. To produce a clustering on the data, a distance or similarity threshold is required to stop the mergers/divisions. Naturally, the method used to calculate the similarity between clusters affects the clusters that are produced. There are three common methods for calculating the similarity/distance between two clusters, the single linkage, complete linkage, and average linkage clustering methods.

The *single linkage* method is also called the nearest neighbor technique as the similarity between two clusters is defined as the similarity between the two closest members of the two clusters. This method is efficient on large data sets and closely related to clustering using minimum spannning trees. The *complete linkage*, or furthest neighbor, defines similarity between two clusters as similarity the most dissimilar elements in the two clusters. In *average linkage* clustering, a popular method for gene expression data [9], the similarity between two clusters is the similarity between their center or average profiles. A disadvantage of the average linkage method is that the profiles of small clusters will be subsumed when merged with large clusters. One possible solution assumes the clusters are of equal size. This is called *median clustering* [10].

Both median clustering and single linkage clustering have difficulty with data where there are well separated clusters with intermediate noise points since these methods chain together individuals linked by intermediaries. Average and complete linkage tend to impose a spherical solution on the data. Complete linkage can be too restrictive in creating a cluster if there are a lot of measurement errors, while average linkage can ignore some errors and outliers in the data. The hierarchical methods were developed to construct taxonomies in biology. It has been questioned on how well they fit gene expression data since expression patterns are not thought to be related by hierarchical descent. However, the hierarchical methods have proven useful because they produce a dendrogram that shows organized relationships in the data [9]. There are measurements such as the *cophenetic correlation coefficient* [10] to assess the match between the dendrogram and the proximity matrix and determine if the data exhibits a hierarchical structure.

Partitional Methods

Traditionally, non-hierarchical clustering methods are referred to as partitional methods since they generate a single partition in the data [17]. Under partitional methods the problem is viewed as partitioning the n patterns in d dimensional space into K clusters so that the patterns in a cluster optimize a cost function such as the *sum-of-squares*.

The most common relocation method is k-means clustering which uses a random hill climbing strategy. The user selects the number of clusters k, and the algorithm tries to minimize the sum of the distance from each data point to its centroid or maximize the sum of the similarity between each data point and its centroid. The algorithm begins by randomly assigning k genes to the center profiles and then iterates between assigning genes to the closest center and recomputing the center as the average of the cluster's members. The iteration stops when no genes are moved between clusters. The algorithm returns a local minimum or maximum. The algorithm is fast, $\Theta(kn)$ where k is the number of clusters and n is the number of elements to be clustered.

In one comprehensive clustering comparison study on ten sets of artificial data, nine of which had errors, k-means clustering gave better results than various hierarchical methods when the starting partition was close to the final solution [19]. Complaints about k-means clustering center upon the fact that it is an unstructured approach that produces an unorganized collection of clusters that is difficult to interpret [25].

A popular partitional algorithm that finds organized clusters that are easier to interpret is self-organizing maps. It has been used to find clusters on gene expression data [25]. The user selects a geometry of "nodes" (usually a two dimensional grid). The nodes are mapped at random into the data space and then iteratively adjusted.

The rows of data are permuted randomly and at each iteration i a data point P is selected and the node N_p that is closest to P is identified. The location of each node N is then adjusted according to the formula:

$$f_{i+1}(N) = f_i(N) + \tau(d(N, N_p), i)(P - f_i(N))$$

where the learning rate τ is defined as $\tau(x, i) = \frac{0.02T}{(T+100i)}$ for $x \leq \rho(i)$ and $\tau(x, i) = 0$ otherwise. The radius $\rho(i)$ is initially set to 3 and decreases linearly with i and eventually becomes zero and T is the maximum number of iterations, which is set to roughly 50,000. The algorithm is fast and on the $\Theta(kn)$ where k is the number of clusters and n is the number of elements to be clustered.

The problems with these methods are that they favor spherical clusters and do not deal with "noise" or outliers well.

Density Based Models

An alternative method that addresses the problems of cluster shape and "noise" is density based models [11]. These models approach clustering by viewing the data as points in space and searching for regions that are densely populated and surrounded by relatively empty regions.

The taxmap method [4] attempts to compare relative distances between points and to search for continuous populated regions surrounded by some empty space. The algorithm takes a similarity matrix and two thresholds as parameters. The first is used

as a threshold to initiate a cluster. Clusters are initially formed by the single linkage approach, finding the two closest genes. The second threshold is used in considering additions to the cluster. Prospective members are added if they do not provoke a drop in similarity above the threshold. This drop is calculated by considering the decrease in the average similarity in the cluster on addition of the prospective member and then subtracting this decrease from the new average similarity. According to the authors, this measure has been found to decrease smoothly up to a discontinuity, whereas the drop in the average similarity itself varies widely. We report results of this algorithm in the Results chapter.

Other density based approaches use the statistical framework established in mixture models which are discussed below.

Graphical Models

The graphical approach to clustering represents the data points as nodes in a graph, with an edge between two nodes if the two nodes exhibit a similarity based on some definition. Ideally, a cluster structure in the graph is exhibited by vertex-disjoint cliques.

An older approach creates a fully connected graph with weights on the edges reflecting the similarities between the nodes. A minimum spanning tree (MST) is found on the graph. The edges on the MST are then cut in the order of weight. Each

cut will define a new clustering that is equivalent to the clustering produced by single linkage clustering [10].

Another approach developed [28], involves identifying inconsistent edges in the MST and removing them to form connected components which are identified as clusters. Criteria for identifying inconsistent edges, which represent cluster separation, can be local or global.

A graph theoretic approach developed specifically for gene expression data is CLICK, Cluster Identification via Connectivity Kernels [24]. This algorithm assumes that the similarity values between profiles are normally distributed with genes within the same cluster, or mates, having a mean μ_T and a variance σ_T^2 and genes from different clusters, non-mates, having a mean μ_F and variance σ_F^2. The authors note that these normal distributions were observed on real data, but that if the similarity values are not normally distributed than their distributions can be approximated. The first step in the algorithm is to estimate the distribution parameters, μ_T, σ_T^2, μ_F, σ_F^2 and the probability, p_{mates}, that two elements are mates. These parameters can be estimated either from a subset of genes whose clusters are identified by prior knowledge, or clusters identified experimentally [16]. Based on a subset of the data, the sample mean and variance for similarity values between mates and non-mates can be used as maximum likelihood estimates for the distribution parameters. Then a weighted similarity graph $G = (V, E)$ is derived from the similarity matrix so that

the weight, $w_{i,j}$ of an edge (i, j) corresponds to the probability that i and j are mates and is set to be:

$$w_{i,j} = \log \frac{p_{mates} f(S_{i,j} \mid i, j \, are \, mates)}{(1 - p_{mates}) f(S_{i,j} \mid i, j \, are \, non - mates)}$$

The value of $f(S_{i,j} \mid i, j \, are \, mates)$ is the value of the mates probability density function at $S_{i,j}$. The algorithm recursively makes cuts in the graph to find connected components, called kernels, that are considered "true" clusters. A cut in a graph is a subset of the graph's edges that disconnect the graph. A minimum weight cut is a cut in G with minimum weight. The algorithm evaluates all cuts in the graph and tests whether each cut contains edges between non-mates or only edges between mates. If all the cuts contain only edges between mates, then that connected component is deemed a kernel or pure cluster. Otherwise the cut that gives the weakest bipartition of G is made and the algorithm operates on the two connected components created. Rough psuedocode of the algorithm is as follows:

```
Basic-CLICK(G)
        If V(G) = {v} then move v to the singleton set R.
        Else if G is a kernel then
                Output V(G).
        Else
                (H1, H2) = MinWeightCut(G).
```

```
Basic-CLICK(H1).

Basic-CLICK(H2).
```

The basic algorithm is followed by an adoption step where the similarity between singletons and kernels is evaluated and if the similarity exceeds a threshold the singleton and kernel are merged. Then, in a merging step, kernels are merged if their similarity exceeds a threshold. Again, the adoption of singletons step is run. The CLICK algorithm is fast and compares favorably against self-organizing maps [25], and hierarchical methods [9] on two criteria, cluster homogeneity and cluster separation, that are described at the end of the chapter.

Another graph based clustering algorithm developed for microarray data is CAST, Cluster Affinity Search Technique [2]. The authors approach the clustering problem as the problem of recovering corrupted cliques. A clique graph is a disjoint union of complete graphs. The input to the algorithm is a pair $< S, t >$ where S is a $n \times n$ similarity matrix and an affinity threshold t. The clusters are constructed individually and the current cluster under construction is denoted C_{open}. The affinity of a gene to C_{open} is defined by

$$a(x) = \sum_{y \epsilon C_{open}} S(x, y)$$

and an element has high affinity if

$$a(x) \geq t \mid C_{open} \mid$$

The algorithm alternates between adding high affinity elements to C_{open} and removing low affinity elements. This last step is especially important because of the random seeding of the clusters. In the final step the algorithm evaluates all cluster memberships to determine if members are in their closest cluster and any necessary moves are made. The authors report good results as some of their clusters were verified with biological knowledge and on simulated data the program recovered cluster structure well. The algorithm is a heuristic one, and they are unable to prove bounds on running times, but state that it is fast enough to allow user interaction.

Mixture Models

In the previous clustering methods discussed, the data points are not considered to be samples from a population, a violation of an important tenet of statistics. An approach that attempts to remedy this oversight are mixture models. These models assume that the data are samples from G clusters and that within each cluster the variables have a *multivariate normal density* with a particular mean and covariance matrix. The probability density function is defined as:

$$f(x) = \sum_{i=1}^{G} p_i \alpha(x, \mu_i, \sigma_i)$$

where the $\alpha(x, \mu_i, \sigma_i)$ is a Gaussian distribution and the probability sums to unity. Clustering decisions are based on the maximum values of the estimated posterior

probabilities

$$\hat{p}(s \mid x) = \frac{\hat{p}_s \alpha(x, \hat{\mu}_s, \hat{\sigma}_s)}{\sum_{i=1}^{g} \hat{p}_i \alpha(x, \hat{\mu}_i, \hat{\sigma}_i)}$$

where $\hat{p}(s \mid x)$ is the estimated probability that an individual with vector of observations, x, belongs to group s [10]. The initial values of the parameters, μ_k and σ_k, can be estimated from a previous classification. Then maximum-likelihood parameter estimates can be found through the expectation maximization (EM) algorithm [10]. The limitations of the EM algorithm are that the rate of convergence can be slow if the mixtures are not well separated, the algorithm may not be practical for models with large numbers of clusters, and if the number of components is larger than the true number of groups then the algorithm may fail [10, 13]. An advantage of the mixture-model approach to clustering is that Bayes factors can be used to compare clusters [13]. If EM is used to estimate the maximum mixture likelihood, an approximation to twice the log Bayes factor called the *BIC* [23] is applicable

$$2 \log p(x \mid M) + constant \approx 2 l_M(x, \hat{\theta}) - m_M \log(n) \equiv BIC$$

where $p(x \mid M)$ is the likelihood of the data for the model M, $l_M(x, \hat{\theta})$ is the maximized log-likelihood for the model with θ being the parameters μ and σ, and m_M is a penalty for the number of independent parameters to be estimated in the model. The larger the BIC store, the stronger the evidence for the model. The BIC can be used to compare different models and to find the optimal number of clusters. Fraley and

Raferty give a mixture model based strategy for clustering that uses the BIC score [13]. First determine a maximum number of clusters, M, and a set of parameterizations of the Gaussian model to consider. Then do a hierarchical clustering and obtain the corresponding classifications for up to M groups. Next, perform the EM algorithm for each parameterization and each number of clusters 2,..., M, using the hierarchical classification. Then, compute the BIC for each combination of parameterization and number of clusters 2,....,M. By plotting the BIC values for each model and identifying the first decisive local maxima over all parameterizations, the best model is identified.

This approach works well on intersecting and non-spherical clusters, a problem with most other clustering methods. However, this method falters when attempting to identify clusters that are smaller than the number of conditions. This situation causes the covariance matrix to become ill-conditioned with values near zero. The mixture model approach is of questionable use to the problem of clustering microarray data. This approach assumes that the clusters are concentrated locally around linear subspaces. Gene expression data may not be normally distributed and can have a large number of clusters in the data with the corresponding need to estimate many parameters. However, this last point could be worked around by using a smaller sample of the expression matrix and assigning the remaining data to the prescribed clusters. This algorithm, called *mclust*, is available as a component of the freeware statistical package *R* [13].

Fuzzy Clusterings

Fuzzy clustering can assign a data point, or gene expression profile, to more than one cluster or perhaps to none at all. One such method, called the *Plaid model*, was developed with the goal of finding interpretable biological structure in gene expression microarray data [18]. This model allows a cluster to be based on a subset of the samples of its members. The central goal of the Plaid model is to form a color image of the data on a $n \times p$ grid where each cell is colored according to the value of $Y_{i,j}$. Then this image's rows and columns can be re-ordered to group together rows and columns of similar color. The ideal re-ordering produces an image with a number of K rectangular blocks of nearly uniform color on the diagonal of the matrix. In this manner every gene in gene-block K is expressed only within those samples in that block. The algebraic representation is

$$Y_{i,j} = \mu_0 + \sum_{k=1}^{K} \mu_k \rho_{ik} \kappa_{jk}$$

where μ_0 is the background color, μ_k is the color in block k, ρ_{ik} is 1 or 0 describing if gene i is in the k'th gene-block and κ_{jk} is 1 or 0 describing if sample j is in the sample block k. The constraints that every sample and every gene are in one cluster are $\sum_k \rho_{ik} = 1$ for all i and $\sum_k \kappa_{jk} = 1$ for all j.

However, this ideal rarely exists on real data [18]. Usually some genes and conditions will fit into more than one cluster and there may be some that do not fit well

into any cluster. If we remove the constraints the model becomes a representation of the data as a sum of possibly overlapping layers that do not have to cover the whole array.

In order to identify sets of genes that have an identical response to a set of samples or sets of samples with a common expression patterns for a set of genes, the following models can be used:

$$Y_{ij} = \mu_0 + \sum_{k=1}^{K} (\mu_k + \alpha_{ik}) \rho_{ik} \kappa_{jk}$$

$$Y_{ij} = \mu_0 + \sum_{k=1}^{K} (\mu_k + \beta_{jk}) \rho_{ik} \kappa_{jk}$$

$$Y_{ij} = \mu_0 + \sum_{k=1}^{K} (\mu_k + \alpha_{ik} + \beta_{jk}) \rho_{ik} \kappa_{jk}$$

Each $\rho_{ik} \in$ 0,1, and $\kappa_{jk} \in$ 0,1 and with the models including α or β the constraints that $\sum_i \rho_{ik} \alpha_{ik} = 0$ and $\sum_j \kappa_{jk} \beta_{jk} = 0$ to avoid overparameterization. The notation θ_{ijk} is introduced to represent either μ_k, or $\mu_k + \alpha_{ik}$, or $\mu_k + \beta_{jk}$, or $\mu_k + \alpha_{ik} + \beta_{jk}$. Layer types, represented by α_{ik} or β_{jk} can be mixed so they might appear in some but not all of the θ_{ijk}. θ_{ij0} represents the background layer. The model is written as a sum of layers,

$$Y_{ij} = \sum_{k=0}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk}$$

where each layer may identify a particular set of biological processes or conditions. The values of α_{ik} and β_{jk} give information on the effects of a layer k on the genes and samples. Genes with a larger values of $\mid \mu_k + \alpha_{ik} \mid$ are more greatly affected under the

conditions of layer k. If $\mu_k + \alpha_{ik}$ is positive it means that this gene is upregulated under the conditions of layer k, if it is negative the gene is downregulated. If layers overlap, these interpretations must be made after the other layers have been subtracted.

In order to find the model, the authors adopt an iterative approach updating the θ, ρ, and κ values in turn, where the ρ and κ values are continuous until the final iterations. The starting values are computed by setting $\theta_{ijk} \doteq 1$ for i and j and then doing iterations updating ρ and κ values only. The algorithm adds one a layer at a time, and the stopping criteria is calculated by determining if the importance of layer k is greater than the most important layer found in randomly permuted data.

This algorithm was tested on the data from [9] comprising 10 experiments for a total of 79 conditions on 2467 genes, and found 34 layers. Almost a third of the genes and conditions were explained by background. There was little overlap in the layers found and the model largely placed conditions from the same experimental series in the same layer. Biological knowledge confirmed the unifying function behind several layers. This approach is an interesting one as it can determine the number of layers or clusters independently, and its ability to cluster a gene in more than one layer can identify genes that may perform different functions under different conditions.

Cluster Evaluation

Since most clustering algorithms will return a partition on the data regardless of whether the data exhibits a clustering tendency, evaluating the clustering is a vital step. If the true classification is known, this is easy. The *Jaccard coefficient*, defined below, can be used to compare two $n \times n$ binary matrices where a matrix entry is a 1 if the two genes were clustered together and a 0 if they were not. If T is the matrix for the true solution and C is the matrix for the suggested solution, then a 2 x 2 matrix, N, can record the number of agreements and disagreements on clustering. N_{11} stores the number of agreements on classification. N_{00} stores the number of agreements on not classifying, and the number of disagreements are stored in N_{10} and N_{01}. The *Jaccard coefficient* [10] is the ratio

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

It is unlikely, however, that the true classification is known and so we must look to other measures for evaluating a clustering. Two important properties of clustering are internal homogeneity and external separation. Based on these properties two metrics have been developed [24]. For fingerprint data homogeneity can be determined by the average and minimum correlation coefficient (or any other similarity measure) between cluster members and their cluster profiles. If $cl(u)$ is the cluster of u, and

$S(x, y)$ is the similarity between elements x and y, the average homogeneity is

$$H_{Ave} = \frac{1}{\mid n \mid} \sum_{u \epsilon n} S(u, cl(u))$$

and the minimum homogeneity is

$$H_{Min} = min_{u \epsilon n} S(u, cl(u))$$

Separation can be defined by the weighted average and the maximum similarity between cluster vectors, $X_1, \ldots X_K$

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

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

If the correlation coefficient is used, a solution improves if H_{ave} and H_{min} increase and S_{ave} and S_{max} decrease. We implemented these metrics of cluster evaluation in our tool, Profile Matrix Analyzer (PMA).

Another metric used to evaluate clusterings was detailed in [27]. They applied a *jackknife* approach and clustered data with the omission of one condition. The clustering solution can be assessed by finding the mean error of the expression level of gene x in the omitted condition e and the average expression level in condition e of the genes clustered together. Three figures of merit (FOM)'s were used in total, one calculating the mean error squared, one the Manhattan distance, and one computing the range from maximum to minimum values of condition e for each cluster. By

omitting each condition in turn from the data set and clustering, an overall picture of the predictive power of the clustering method emerges from the aggregate FOM's. Graphing the different FOM's over an increasing number of clusters for different clustering methods reveals which methods predict best. The minimum achievable range FOM, which can be calculated in $\Theta(n \log n)$ time, can be used to determine how well a clustering method matches the "perfect" solution at a given number of clusters. The other FOM's can be used to compare clusterings of different methods and similarity metrics. A slight drawback to this approach is that in order to find the aggregate FOM's, the clustering method must be run on the data the number of conditions - 1 times for each number of clusters to be assessed. Otherwise this is an interesting approach that has yielded some valuable results that are mentioned above. Specifically the conclusions reached by initial testing with this approach are that there are a set of clustering methods and similarity metrics that yield approximately the same FOM's. This indicates that there may not be a "perfect" method or metric for the clustering problem on gene expression data.

CHAPTER 3

GENETIC REGULATORY NETWORKS

All the cells in an organism carry the same genomic data while the protein composition in the cells differs dramatically. While this differing protein composition is due to many factors, mRNA transcription, loosely called gene expression in this paper, is a central control mechanism. With our new ability to observe the expression profiles of thousands of genes over varying conditions through microarray experiments, a goal of the data analysis is to uncover genetic regulatory networks. A genetic regulatory network discovered by observing mRNA transcription levels could reveal sub-networks of genes regulating each other through their protein products.

Clustering, a frequent first step in analyzing gene expression data, has successfully identified groups of co-regulated genes. By using the clusters to determine probabilistic genetic regulatory networks, we seek to discern structural relationships between clusters of co-regulated genes. In this chapter we will provide a brief introduction to a common approach to finding these networks, Bayesian Networks, and detail our own approach, the Partitioned Markov Hypercube.

Bayesian Networks

Bayesian networks have become a popular method for extracting and encoding knowledge from data. Within the gene expression domain, Bayesian networks exhibit favorable features that allow the modeling of genetic regulatory networks. First, the networks can handle incomplete data. Second, the networks yield information about causal relationships. Third, Bayesian networks associate probabilities with causal relationships. This is an important feature for genetic regulatory networks as elements of the network most likely behave in a probabilistic way. Bayesian networks are suited for learning in sparse domains. It is assumed that genetic regulatory networks are sparse since it is expected that less than a dozen genes affect the transcription of one. Finally, these networks avoid overfitting the data.

A Bayesian network is a graph-based model for joint multi-variable probability distributions that captures properties of conditional independence between variables [15]. Bayesian networks have been applied to microarray data [15]. Bayesian networks consists of two components, G a directed acyclic graph whose vertices correspond to the random variables $X_i, \ldots X_n$ and Θ, a conditional distribution for each variable given its parents in G. For modeling genetic networks, the expression level of each gene is a random variable, and the conditions tested in the experiment, or any other

attributes affecting the system can be modeled as random variables. The Markov assumption holds for the graph G so each variable is independent of its non-descendants given its parents in G. Conditional independent assumptions are also encoded in the graph. For these two reasons, the joint distribution can be calculated by

$$P(X_1, \ldots X_n) = \prod_{i=1}^{n} P(X_i | Pa(X_i))$$

where $Pa(X_i)$ is X_i's parents in G.

The problem of learning a Bayesian network is stated as follows: Given a training set $D = x^1, \ldots, x^N$ of independent instances of X, find a network $B = \langle G, \Theta \rangle$ that best matches D [15]. Prospective networks are scored according to the *Bayesian Score* and *BDe priors* which have two important features: graphs of equivalent structures will have the same score and the score is decomposable which means it can be written as the sum of terms

$$S_{BDe}(G \mid D) = \sum_i ScoreContribution_{BDe}(X_i, Pa(X_i) \mid D)$$

With the decomposable scoring function, a local search changing one edge at a time can be pursued. In order to focus the search space, the sparse candidate algorithm can be used to define a set of candidate parents for each gene based on local statistics, such as correlation or mutual information. At each iteration n, for each variable X_i, the algorithm finds the set $C_i^n = Y_i, \ldots, Y_k$ of promising candidate parents for X_i.

Then the algorithm searches for the optimal network where $Pa^{G_n}(X_i) \subseteq C_i^n$. The score is insured to increase between iterations by requiring $Pa^{G_{n-1}}(X_i) \subseteq C_i^n$. The algorithm stops when the candidate sets stabilize.

In [15], they discretize the gene expression data to -1, 0, or 1 depending on whether the expression is lower than, similar to, or greater than the respective control. The Bayesian network model approach was tested on the data set reported in [21], where 800 genes were identified as cell cycle regulated. The network found on those 800 genes revealed the existence of a small set of *dominant genes* that appeared as parents for many genes. Biological knowledge verified the importance of some of these dominant genes.

Dynamic Bayesian Networks (DBN) differ from Bayesian Networks (BN) in that they attempt to model a stochastic system over time. A DBN represents a distribution of trajectories of the system. Two assumptions are made that reduce the representation problem [3]. First, the Markov assumption is made meaning that there is condition independence $I(X^{t+1}; X^0, \ldots, X^{t-1} \mid X^t)$. The second assumption is that the process under study is *stationary* so that $P(X^{t+1} \mid X^t)$ is the same for all t. With these assumptions, the network can be specified with a prior network B_0 which establishes the distribution over initial states $X^{(0)}$, and a transition network $B_{\rightarrow}$ which represents the transition probability from states X^t to states X^{t+1}. The

transition network is a BN fragment over the nodes $X_1, \ldots, X_n, X'_1, \ldots, X'_n$. A node X_i represents X_i^t and X'_i represents X_i^{t+1}.

In order to learn the DBN from data containing a trajectory $d^{(0)}, \ldots, d^{(T)}$ through the system, a scoring function such as the BIC score is used to score candidate networks. The BIC score is the log-likelihood function, $\ell(B : D) = \log P(D \mid B)$ with a penalty for network complexity. Since finding the highest scoring network structure is NP-hard, Boyen *et al.* resort to greedy local search procedures applying local structural changes [3].

This algorithm has not been applied to gene expression data. The data sets examined with this approach had more conditions that variables and thus had a very different shape than the typical gene expression data set. It is a promising approach because it identifies hidden variables by looking for violations of the Markov property. However, problems with this approach are that it does not allow cycles in the data and it does not model mutually exclusive events.

Binary Profiles

The first step in our approach to find probabilistic genetic regulatory networks is translating cluster expression profiles to binary values. By viewing gene expression profiles as either "on", or "off" we are losing information and simplifying the data considerably. Arguments in support of this translation begin with the nature of the

data itself. Unless a gene is activated, where a signal is bound to the activation site of a gene, only low levels of expression will occur. Correspondingly, if a gene is inhibited no expression of that gene will occur [5]. Therefore it appears that in nature expression of a gene largely means that it is activated or "on" or that it is "off". Microarrays, as our source of observation of mRNA transcription levels, are known to be noisy [14]. The observation of levels of mRNA with their different breakdown rates in the cell also does not give a direct observation of the rate of transcription. With these thoughts, and the added benefit that discretizing the data enables us to construct a more powerful algorithm to find regulatory networks, we have translated our cluster profiles to binary values where 1 signifies the gene is on at that time and 0 signifies the gene is off.

We have developed two ways of discretizing the data in this manner. They are both simple approaches. In the first, we convert expression profiles to binary by supplying two thresholds, one marking where the gene is turned on and the other where it is turned off. These thresholds can be the same, or different, to allow an intermediate stage where a switch has not occurred and the previous state, on or off, continues. This intermediate stage can be considered a cushion for noise. If the experiment being studied gives expression values in the form of a ratio where a value of 1 means that the test sample has the same level of expression as the control sample and the thresholds supplied center around 1, then the binary translation divides the

profile into two categories, expression over and under the control. This can be useful in some cases. A difficulty with this approach is choosing the thresholds.

Our second approach to translating the expression profiles to binary is based on their rate of change. A positive slope between two points (ignoring any experimental error) indicates an increase in expression and a negative slope indicates a decrease in expression. This approach is easy as there are no parameters to choose, but without smoothing the profiles a lot of noise could be translated as well.

Partitioned Markov Hypercube

The Partitioned Markov Hypercube finds probabilistic regulatory elements by analyzing clusters' expression over time [20]. In the (PMH) approach, the elements of the network are binary random variables $x_1, \ldots, x_n$. These variables are assumed to constitute a Markov process so that the probability that x_i changes at time $t+1$ depends only on the values of $x_1, \ldots x_n$ at time t. The dynamical state of the network is specified as $< X_i(t) >$. The set of variables are partitioned into mutually exclusive sets of colors so that if one variable of color C_i changes then the other variables in C_i do not change at that time. This feature models the fact that many processes have mutual exclusion constraints. The dynamics of the model are viewed as occurring on a hypercube whose dimensions represent the variables in the model. Each dimension receives the same color as its associated variable. The current state of the model is

a vertex on the hypercube, called the trackpoint. If the trackpoint is projected into the dimensions of one color, then it traverses at most one edge per time step.

The directed edges between each pair of neighboring vertices, separated by a Hamming distance of 1, have an associated transition probability. The constraints on the edge probabilities is that the sum of outgoing edges of the same color at each vertex sum to ≤ 1. In order to represent these edge probabilities efficiently, we represent a set of equivalent edges using the notation

$$x_i \rightarrow 1(x_j = 0, x_k = 1), p = .25$$

for what is called an x_i crossing edge that goes to 1 with a probability of .25 when its split variables $x_j = 0$ and $x_k = 1$. The model is initialized so that each variable has one positive and negative crossing edge that is not dependent on other variables. The model is scored by

$$P(data \mid model) = \prod_{c \epsilon Color} \prod_{t \epsilon time} \sum_{s=t \rightarrow t+1} P(s)$$

where the last sum is over all the shortest paths $P(s)$ in color c between observed time points t and $t+1$. The shortest path $P(s)$ is a product of the probabilities of the edges of the path. If there is no change between time points, then $P(s)$ is 1 minus the sum of the outgoing edges of the trackpoint at time t. By summing over all the shortest paths, we are assuming that the observation rate is fast enough that the system under study only had time to take the shortest path between observations.

By taking the negative log, we define a cost function that we want to minimize:

$$cost(model) = -\sum_{c \epsilon Color} \sum_{t \epsilon time} \log \sum_{s=t \rightarrow t+1} P(s)$$

The cost function is non-linear so we use a local search technique to find a minimum. The unity constraint on the probabilities defines a convex polytope that represents our solution space.

The search for the best model iterates between a search for good candidate split variables and the search for assigning edge probabilities. The search stops when there is no significant model improvement. The PMH model is similar to the Dynamic Bayesian Network model discussed above, but allows the modeling of mutually exclusive events. An advantage of the PMH model is its ability to investigate dynamical features such as feedback loops.

CHAPTER 4

DATA SETS ANALYZED

We clustered three data sets studying the common yeast, *Saccharomyces cerevisiae*. Two were created using DNA microarrays with a protocol similar to the one established by [22], while the third, what we have called the Cho *et al.* Data, used the common commercial oligonucleotide arrays produced by Affymetrix. The data sets overlap to a degree. The Eisen *et al.* Data has some experiments from the Spellman Data. Common preprocessing techniques are discussed at the end of the chapter.

Cho *et al.* Data

This data set was created by using four Affymetrix oligonucleotide arrays covering the entire yeast genome, more than 260,000 oligonucleotides complementary to 6,218 yeast genes [6]. *cdc28-13* yeast cells were synchronized by arresting them in late G1 at START by raising the temperature, and then the cell cycle was reinitiated by lowering the temperature. Cells were collected 17 times at 10 minute intervals over approximately two cell cycles.

In all of the time points, RNA was isolated from each sample, converted to cDNA, fluorescently labeled, and hybridized to the Affymetrix yeast whole genome oligonucleotide array. Bacterial probes were placed on the arrays as a control. A linear range of detection was determined by plotting the fluorescence value for the bacterial probes on the array against their expected fluorescence. Then a time point, t, containing the median overall fluorescence was found. The fluorescence for the set of genes at the time point t that fell within the linear range of detection was identified and summed. The fluorescence of this set of genes at each time point was summed. The ratio of these fluorescence values was used as the factor for normalizing each time point. This experiment created a complete 6601×17 matrix.

Spellman *et al.* Data

This data set is associated with the Yeast Cell Cycle Analysis Project. Their goal is to identify all genes whose mRNA levels are regulated by the cell cycle. In their experiments they identified roughly 800 genes that are cell cycle regulated by using DNA microarrays to analyze mRNA levels in cell cultures that had been synchronized by three independent methods [21]. The data set is comprised of four experiments. The first contains experiments with overactivated Cln3 and Clb2, two cyclins thought to control approximately one half of the genes that are cell cycle regulated. This experiment yielded 5 data points that are not in a time series [21]. The second

experiment contains the profiles of a yeast culture synchronized by α factor (also called the Pheromone experiment) sampled at 7 minute intervals for 119 minutes to yield a series of 18 time points [21]. The third experiment involved placing the culture in cdc15 arrest and then releasing it and observing it for 25 time points [21]. In the final experiment, yeast cells were synchronized by elutriation and observed over one cell cycle in a series of 14 time points [21].

In all of these experiments, RNA was extracted from the samples, and a control sample, asynchronous cultures growing at the same temperature in the same medium. cDNA was synthesized from the samples and control sample and fluorescently labeled using Cy3, green, for the controls and Cy5, red, for all experimental samples. Mixtures of labeled control and experimental cDNA were competitively hybridized to individual microarrays containing essentially all yeast genes [8]. The ratio of sample, red, to control, green, were output by scanning laser microscopy. This yielded a data matrix of 6,177 genes under 76 conditions. The data is 87.6% complete.

Eisen *et al.* Data

This data set is comprised of ten experiments on the budding yeast *Saccharomyces cerevisiae* [9]. Gene expression for every open reading frame (ORF) from this fully sequenced organism was collected by spotted DNA microarrays [22]. Gene expression was studied during the diauxic shift in a series of 7 time points [8], the mitotic cell

division cycle in three series of 18, 14, and 15 time points [21], sporulation in three series of 6, 3 and 2 time points [7], and temperature and reducing shocks in three series of 6, 4, and 4 time points which had been previously unpublished.

In all of these experiments, RNA was extracted at selected times during the experiment and was labeled during reverse transcription with the red-fluorescent dye Cy5 and was mixed with a reference sample labeled with the green-fluorescent dye Cy3. The reference sample was taken from time point 0 for all experiments except the cell division cycle experiment, where asynchronous cells were used. The intensity values are ratios of the fluorescence of the Cy5 and Cy3 dyes and are log transformed (base 2). Some observations were rejected by the image analysis software and the data set contains 2467 ORF's with 79 data points. The data is 98.07% complete.

Common Preprocessing Methods

There are a variety of preprocessing steps that can be applied to microarray data. We will discuss the common approaches, some of which have been included as user options in our program, PMA. Unfortunately, there is no straightforward answer to what are the best preprocessing methods to apply and different methods typically change the outcome of further analysis.

Taking the Log

One of the most common prefiltering measures is taking the log of the data. This is done to make the distribution of the data more appropriate for further analysis. In particular, it makes the variation of intensities and ratios of intensities more independent of absolute magnitude. It evens out highly skewed distributions so that the standard deviations are correctly defined on the data. If the data set contains negative values, it is necessary to scale the data in order to make all values positive before applying the log.

Normalizing

Normalizing is typically distinguished from the preprocessing step. Normalization techniques are used to make transformations that compensate for systematic variations in data sets.

One method of normalizing the data is to make each column have a mean of 0 and a variance of 1. Our program instituted this type of normalization as it is beneficial when using Euclidean distance as a distance metric since all features get equal weight. When using the correlation coefficient as a similarity metric, there is a mean-centering calculating inherent in the metric so that some types of normalization are less important. When the data set is comprised of multiple cell cycles, each cell cycle can be normalized to have a mean of 0 and a variance of 1 [25].

Global normalization is used to correct for differences in hybridization between arrays. In one approach each array's intensities are manipulated so that the average intensities on each array is the same. However, if one array actually has higher expression for many genes, then global normalization can flatten and obscure these results.

Normalization can also have undesirable effects. Interpoint distances are changed and this can reduce or eliminate the natural separation between clusters [17].

Variation Filter

It is important to filter the data to remove genes with low levels of expression and profiles that are relatively unchanging, as they may represent noise. Typically, a variation filter requiring a relative change of expression and an absolute level of expression is used [25]. In PMA, we give the user three options of control in this area. The user can specify a threshold for change requiring that

$$\frac{(MAX - AVG)}{AVG} > threshold$$

We give the user another option for selecting genes without adequate change by allowing the user to set a threshold for a change in expression that must be met in at least two consecutive time points. In order to filter genes with low level expression, the user can set a threshold for absolute level of expression.

Incomplete Data

Frequently, microarray data sets are incomplete due to poor intensity values. Our PMA program makes the data complete by:

$$Incomplete_{ij} = \mu_i + \mu_j - averageIntensity$$

The user can also require a "complete" percentage that a gene's profile must meet or be excluded.

Singular Value Decomposition

Singular Value Decomposition normalizes the data by filtering out the "eigengenes" and "eigenarrays" that are assumed to represent noise or experimental artifacts [1]. This normalization step allows comparison across different experiments and arrays. SVD is known as principal-component analysis in statistics.

SVD calculates a linear transformation of the expression data from the N-genes $\times$ M-arrays matrix $\hat{e}$ to the reduced L- "eigenarrays" $\times$ L- "eigenegenes" space, with L defined as the minimum of M and N. The linear transformation is calculated by

$$\hat{e} = \hat{u}\hat{\varepsilon}\hat{v}^T$$

where $\hat{u}$ is the genes x eigenarray matrix, $\hat{\varepsilon}$ is the eigenarray x eigengenes matrix and $\hat{v}^T$ is the eigengenes by arrays matrix. In $\hat{v}$ an eigengene, l, is only expressed in the lth eigenarray. Therefore, the expression of each eigengene, $\hat{\varepsilon}_L$ is decoupled from the

expression of all other eigengenes. In addition the expression of each eigengene is also decorrelated so that some of the eigengenes possibly represent independent processes. For details on calculating these matrices, see [1]. The fraction of eigenexpression indicates the relative significance of the lth eigengene and eigenarray. The Shannon entropy of the data set can be calculated to determine the complexity of the data from the distribution of eigenexpression.

Using the eigen matrices, the data can be normalized by filtering the eigengenes and eigenarrays assumed to represent noise. After this normalization step, the data is sorted by similarity to a subset of the eigengenes' expression. On the elutriation data set published in [21], the first and most significant eigengene was determined to capture more than 90% of the overall relative expression, so that the entropy of the data set was quite low. Other eigengenes were assumed to represent experimental artifacts, and weak perturbations on the steady state of the system. Using this technique, underlying processes can be identified in the data, and genes ordered to show their contribution to these processes. However, if the data was originally time series data, the eigenarrays no longer represent consecutive time points.

CHAPTER 5

RESULTS

By attempting to find clusters of co-regulated genes from microarray data, the biologist or computer scientist is confronted with difficult questions. Which clustering algorithm is best to use? Which similarity metric is best to use? What pre-processing should be done?

Most likely the answers to these questions will differ for different data sets. In our attempt to contribute to these answers, we have created a program, Profile Matrix Analyzer (PMA), that allows a user to choose from a number of preprocessing options and clustering algorithms. We used the PMA and experimented with different clustering algorithms and preprocessing methods on three data sets which observed gene expression profiles over time for the common yeast, *Saccharomyces cerevisiae*. We review some of the features of the PMA, and report our findings.

Profile Matrix Analyzer Features

PMA allows the user to create clustering experiments. In a single experiment, the data is preprocessed in some manner and any number of clustering methods are run on the data. Currently, the clustering algorithms available are k-means using