ELSEVIER

Applied Mycology and Biotechnology
An International Series
Volume 6. Bioinformatics

A Survey of Computational Methods Used in Microarray Data Interpretation

Brian Tjaden[1] and Jacques Cohen[2]
[1]Computer Science Department, Wellesley College, Wellesley, MA 02481, USA (btjaden@wellesley.edu); [2]Volen Center for Complex Systems, Brandeis University, Waltham, MA 02454, USA (jc@cs.brandeis.edu).

In a companion chapter in this volume, Wilson et al. (this volume, chapter by Wilson et al.) provide a detailed account of the experimental design and statistical analysis of microarray data. Their chapter is of interest to researchers planning microarray experiments capable of yielding data that can be statistically analyzed to insure reliable levels of confidence. In contrast, the present chapter emphasizes what can be done with the gathered data so as to simplify the huge task of interpreting the expression levels of tens of thousands of genes. In the companion chapter the authors assume the availability of statistical programs that are often used in the design of experiments. In this chapter we explore in greater detail the algorithms that process the collected data to obtain further information about cell behavior. Many of the algorithms described here aim at grouping similar data. We also explore microarray usage that is not addressed in the companion chapter.

1. INTRODUCTION

Considering the storage capabilities of present-day computers, the size of the data generated by a *single* microarray experiment is relatively modest. An array with a capacity for measuring the transcription expression of 30,000 genes and involving two variants of cells identifiable by two colours can be internally represented in a computer using 60,000 storage units (possibly words). Half of them represent the relative degree of hybridization, the other half the percentage of each individual colour as measured for each array position. However, in a time-series microarray experiment, dozens if not

Corresponding author: Brian Tjaden

hundreds of single experiments may be required to estimate how gene product concentrations vary with time. In the pharmaceutical industry the effect of drugs is often estimated by microarray experiments. It is natural then to infer that, with the decreasing costs of microarrays and scanners, the available data will grow at a very fast rate, very likely exponentially as in the case of genomic data.

As explained in the accompanying chapter, the present quality of microarray data is relatively poor (noisy) compared to that of genomic sequences. In ranking order of accuracy of laboratory measurements, structural protein data (in the PDB) has the highest quality, followed by genomic data; unfortunately, microarray data takes a distant third position. Yet, storage capacity is not the most significant problem facing bioinformaticians dealing with microarrays. A major bottleneck is the processing time needed to properly *interpret* their data. By interpretation we mean obtaining quantitative information about a variety of biological facts that increase our knowledge about cell behavior. That interpretation has to take into account the noisy nature of microarray data and insure that the obtained results attain good levels of confidence. When dealing with tens of thousands of gene expression data it becomes necessary to group together the genes whose expression behavior is similar. The class of algorithms that perform this task is called *clustering*. From a computer science perspective the determination of the *best* among various possible clusters for a large set of data is a very time consuming task. By *best* it is meant the one that groups together objects that are most similar, using a mathematical measure of similarity. Thus, inevitably, we have to conciliate accuracy and efficiency by designing *approximate* microarray clustering algorithms that run in a reasonable time, say of the order of minutes, using presently available hardware.

The objective of this chapter is to present the algorithmic aspects of the interpretation of microarray data. The companion chapter by Wilson et al. contains valuable information about the biological facets of obtaining microarray data and using specific software to obtain results that are useful to biologists. In contrast, we are concerned about conveying to the reader the various algorithmic options that are available for interpreting microarray data, their advantages and drawbacks (Quackenbush 2001). In addition, we also describe types of microarray usage other than those for determining gene expression properties, for example, in finding single nucleotide polymorphisms (SNPs). In explaining the material in this chapter, we purposely avoid complex mathematical notation. It is assumed that the reader is savvy in elementary algebra, probability, and is familiar with some computer programming and web usage. This work is directed to bioinformaticians and biologists who wish to have a deeper understanding of the capabilities and drawbacks of the various types of microarray data analyses (Jones and Pevzner 2004). Section 2 covers the issues of adequately defining similarity in expression among several genes. Foremost in this definition is establishing how closely two or more gene expression data can be grouped into clusters having the same characteristics, for example being activated or repressed in tandem. Section 3 describes various clustering algorithms; these can be classified according to whether or not human intervention guides the grouping of similar gene data. In

supervised learning a user establishes an initial training set in which she defines subsets of objects as belonging to a desired grouping. When confronted with a new object the computer selects the most likely subset exhibiting similar characteristics; the new object is then added to that subset. These supervised methods are also known as *classification*.

In unsupervised learning no such training set is used. The unsupervised algorithms repeatedly group and regroup the various objects using a similarity measure; the regrouping is done using heuristics that, hopefully, converge to a near-optimal solution. These unsupervised methods are known as *clustering*. The term "clustering" is also used informally to indicate the ensemble of supervised and unsupervised methods. The reader already familiar with topics in bioinformatics will certainly notice that clustering algorithms are closely related to those that construct phylogenetic trees. The latter kind groups species according to genomic similarity, whereas the former groups data according to similarity in gene expression. In both cases the notion of *distance between two objects* guides the clustering, or the tree construction. Following the descriptions of the clustering algorithms, we present a summary of the ongoing work in attempting to deduce genetic networks from microarray time-series data. The three final sections of the paper deal respectively with: (1) open-source and commercial software available to perform clustering and how they relate to the two preceding sections, (2) the current status and applications of microarray usage, and (3) final general remarks concerning the material presented in this work and future directions in microarray analyses.

2. MICROARRAY FORMALISM

2.1 Microarray Data Notation and Formalism

From the conceptual and algorithmic point of view a microarray is a one-dimensional array M, indexed by an integer i identifying a known gene G_i. The content of M_i is initially a large set S_i of theoretically "identical" sequences s_i of nucleotides (A, C, G, U). Each sequence $s_i \in S_i$ represents the complement of a sequence s_i' that, when hybridized with the sequences s_i, can identify the expression of the given gene G_i. The relative number of hybridized sequences for each set S_i provides a relative measure of how much G_i is expressed for a given experiment. Let A_i be an array of real numbers expressing the relative amounts of hybridization for each G_i in a given experiment.

When multiple experiments are conducted, M can be viewed as a two-dimensional array, indexed by an integer i identifying a known gene G_i, and an integer j identifying a particular experiment trial E_j. Then A_{ij} is the relative amount of hybridization for each G_i in E_j. It is assumed throughout this chapter that the rows of the two-dimensional matrix A correspond to genes and the columns of A correspond to experiments. Thus, the i^{th} row of the matrix, denoted A_i, is a vector representing the expression pattern (or profile) of gene G_i across the set of experiments. In typical microarray applications, dozens of experiments are performed, often in replicate, for thousands of genes per array, so that the matrix M may contain thousands of rows and dozens of columns (Fig. 1). As described in the accompanying chapter, in the case of two-colour microarray experiments, each entry in the matrix reflects the log ratio of the relative

Deleted: e

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	⋱	Experiment *m*-1	Experiment *m*
G_1	0.6	4.4	1.3	1.0	…	3.1	2.2
G_2	1.5	2.6	5.2	0.8	…	2.8	2.9
G_3	0.7	3.7	2.4	1.9	…	1.5	1.6
G_4	0.3	0.7	0.2	1.3	…	4.9	3.0
G_5	3.1	3.0	2.1	1.4	…	4.2	0.9
…	…	…	…	…	…	…	…
G_{n-1}	1.8	2.5	1.8	0.7	…	2.7	3.1
G_n	0.5	3.4	3.0	0.5	…	1.8	2.5

Fig. 1: An example of the hybridization matrix *A* for *n* genes assayed by *m* microarray experiments. Each entry represents the relative amount of hybridization for each gene in each experiment.

amounts of hybridization in the two channels. The algorithms that we are interested in deal with extracting valuable biological information from *A*. While the previous chapter by Wilson et al. discusses experimental procedures for obtaining the matrix *A*, including microarray platforms, expression measurement protocols, image spot quantitation, and background correction, we use the expression matrix *A* as our starting point for downstream analyses of microarray data.

2.2 Data Transformation and Normalization

Often one of the first steps, once the matrix *A* is obtained, is *log* transforming the expression measurements of *A*. Taking the logarithm of the expression values provides several benefits. Measurement variation may be stabilized since variations of *log* expression values are less dependent on the absolute magnitude of the expression measurements. Logarithms can also reduce the skewness of highly variable gene expression distributions. Furthermore, when replicate expression measurements are performed for a single gene, the measurements follow a *log*normal distribution in many cases, justifying the *log* transformation. In addition to the *log* transformation, normalization of expression data is important to balance the expression measurement values so that meaningful biological comparisons can be made. The most common approach to normalizing expression levels is based on total hybridization normalization. It is assumed that for two or more microarray experiments, the RNA hybridized in each experiment is equivalent. With this assumption, the expression measurements of all genes in an experiment are normalized so that the mean or median expression values of all experiments are equal. In terms of the matrix *A*, this normalization corresponds to dividing each entry in a particular column of the matrix *A* by the mean value of the given column. In addition to the total hybridization normalization mentioned above, numerous advanced techniques exist such as locally weighted linear regression, rank invariant methods, and ratio statistics (Hogg and Craig 1994). Normalization helps address many of the biases inherent in microarray

experiments, including unequal quantities of starting RNA, variations in labeling or detection efficiencies, and differences in hybridization levels (Quackenbush 2002). The chapter by Wilson et al. in this volume provides an in depth description of the problems involved in normalization.

2.3 Distance Measures

When interpreting a matrix A of microarray data, one of the fundamental operations, which serves as a basis for sophisticated analyses such as clustering, is determining the similarity of two genes' expression patterns. For any two genes G_x and G_y, it is useful to determine the similarity (or dissimilarity) of the vectors A_x and A_y, representing the expression profiles for the two genes. Since similarity measures are inversely related to distance measures, one measure can readily be transformed to the other as appropriate to the application. Examples are provided below for both similarity and distance measures.

The most common distance measure for analyzing microarray data is the Euclidean distance, also called the L_2 norm. The Euclidean distance between two gene expression patterns, A_x and A_y, is given by:

$$d_{x,y} = \sqrt{\sum_{j=1}^{m} \left(A_{xj} - A_{yj} \right)^2},$$

where m is the number of experiments conducted, i.e., the length of the vectors A_x and A_y. This measure has the property of obeying the triangle inequality, namely that the sum of the distances $d_{x,y}$ and $d_{y,z}$ is at least $d_{x,z}$, for any x, y, and z. This property is useful in insuring the appropriateness of certain clustering methods. The Euclidean distance incorporates information on the magnitude of difference between expression patterns, which may not be the best measure of dissimilar expression for two genes. Figure 2 illustrates gene expression profiles for four genes, G_w, G_x, G_y, and G_z. While the expression profiles for genes G_w and G_y have the smallest Euclidean distance, these two genes do not appear co-regulated. Alternatively, the expression profiles for genes G_w and G_x have the same pattern of regulation even though their profiles have a larger distance. Rather than use the magnitude of difference between expression profiles, the shape of gene expression patterns is more commonly used to compare gene expression across various experimental conditions. To capture the shape of expression patterns, the rows of the matrix A can be normalized appropriately, such as subtracting from each entry in a row the mean of the row and dividing each entry by the standard deviation of the row. As an alternative to the Euclidean distance measure, correlation can be used as a measure of similarity for gene expression patterns.

The most common similarity measure for analyzing microarray data is the correlation coefficient, or Pearson correlation (Hogg and Craig 1994). The correlation coefficient for two gene expression patterns, A_x and A_y, is given by:

$$r_{x,y} = \frac{m\sum_{j=1}^{m}\left(A_{xj}A_{yj}\right) - \sum_{j=1}^{m}A_{xj}\sum_{j=1}^{m}A_{yj}}{\sqrt{\left[m\sum_{j=1}^{m}\left(A_{xj}\right)^2 - \left(\sum_{j=1}^{m}A_{xj}\right)^2\right]\left[m\sum_{j=1}^{m}\left(A_{yj}\right)^2 - \left(\sum_{j=1}^{m}A_{yj}\right)^2\right]}},$$

where m is the number of experiments conducted, i.e., the length of the vectors A_x and A_y. This measure assesses the correlation of two gene expression patterns. The value of the correlation coefficient ranges between 1.0 and -1.0, where 1.0 corresponds to perfectly correlated expression patterns, 0.0 corresponds to entirely uncorrelated expression patterns, and -1.0 corresponds to perfectly anti-correlated expression patterns. As an example, genes G_w and G_z in Figure 2 have perfectly anti-correlated expression patterns.

The correlation coefficient does not obey the triangle inequality. However, the correlation measure has the advantage of being invariant under linear transformations of gene expression patterns. In other words, if two expression patterns have the same relative shape but different magnitudes, then they will be perfectly correlated, such as the expression patterns for genes G_w and G_x in Figure 2. The correlation coefficient is based on the assumption that each of the gene's expression values follows a Gaussian

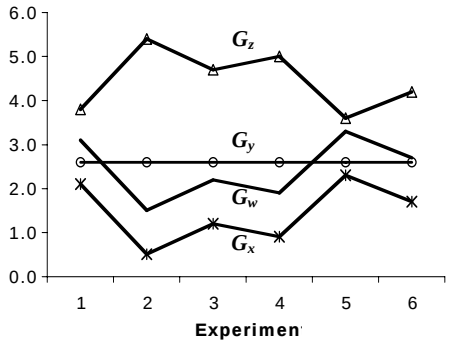

Fig. 2. Four gene expression patterns across 6 experiments are depicted. In terms of Euclidean distance, G_w and G_y have the closest expression patterns. However, since the shape of the expression patterns of G_w and G_x are identical, these two patterns are the most similar in terms of correlation. $r_{w,x} = 1.0$; $r_{w,y} = 0.0$; $r_{w,z} = -1.0$

distribution.

Two popular variations of the correlation coefficient are the Spearman rank correlation and Kendall's τ (Hogg and Craig 1994). In contrast to the Pearson correlation coefficient, these two similarity measures do not assume that gene expression values approximate a Gaussian distribution. Rather, they are non-parametric and they have the advantage of being robust with respect to outlier expression data values. The Spearman rank correlation represents the correlation between the rank of the magnitude of the expression values in the two gene expression patterns, and Kendall's τ represents correlation based on the relative ordering of the expression value ranks.

In the remainder of this chapter, the terms "distance" and "similarity," when relating to two genes' expression patterns, will refer to the abovementioned measures of Euclidean distance and correlation coefficient, respectively. Further, for a set of expression profiles, the term "mean" or "centroid" will be used to denote the average value in each coordinate (i.e., experiment) of all profiles in the set.

3. CLUSTERING

Cluster analysis is the art of finding groups in a given data set such that objects in the same group are as similar to each other as possible and as dissimilar to objects in other groups as possible (Jiang et al. 2004). Dozens of clustering algorithms have been proposed and applied to microarray data sets, each algorithm yielding different results. One of the challenges for researchers using exploratory techniques such as clustering is choosing among the various approaches. A few of the more common clustering methods are detailed in the following pages. In general, there is no one best clustering method, but rather each offers different advantages and disadvantages making it appropriate for particular sets of gene expression data.

One of the reasons for the variety of clustering methods is that there is rarely an unbiased criteria which can be used to evaluate whether one method better partitions data objects than another method. Unsupervised clustering is appropriate when there is no information about which data objects should be grouped together. In contrast, supervised clustering is applicable when cluster information is available for a subset of the data objects; this information is used in determining the needed criteria (training) to subsequently group data objects for which no cluster information is known. Both unsupervised and supervised clustering methods are described below.

Since most formulations of clustering approaches are computationally prohibitive for large data sets, such as those generated by microarray experiments, clustering methods tend to rely on heuristics and approximations. In addition, clustering data objects which are unrelated will still produce clusters, even though the clusters may not be meaningful. Thus, while clustering can be a useful approach for exploring large amounts of gene expression data, researchers must exercise care to ensure they are applying the clustering techniques appropriately.

Most commonly with microarray data (e.g., Figure 3), the objects to be clustered are genes, i.e., the rows of the expression matrix A. It is worth noting, however, that experiments can be clustered similarly, simply by employing the same clustering approaches on the transpose of the matrix A, thereby clustering the columns of the matrix, i.e., the experiments. Indeed, finding phenotypically similar groups can provide useful insights into the data; this analysis is known as *experiment-based* clustering as opposed to *gene-based* clustering.

Clustering gene expression data has numerous useful applications. Genes with related functions often have similar expression patterns, so identifying groups of genes with similar expression patterns may suggest possible roles for genes with unknown function based on the known functions of genes that are placed in the same cluster. For instance, clustering can be applied to identify groups of genes which are expressed at different phases of the cell cycle, such as sporulation. Clustering of genes can also be utilized as a preprocessing step in inferring regulatory networks. For example, sets of clustered genes can be used to reduce the size of regulatory networks to be inferred. Also, clustering can be employed, in connection with sequence data, to identify DNA sequence patterns specific to each expression cluster. For instance, given a set of co-expressed genes as determined from clustering analysis, regulatory motifs such as transcription factor binding sites can be identified by searching for patterns common to the upstream DNA sequences of the clustered genes. Each of the abovementioned applications has been employed successfully in the model organism *Saccharomyces cerevisiae* and has led to new genomic insights. In the following two sections, we review a few of the unsupervised (i.e., clustering) and supervised (i.e., classification) methods.

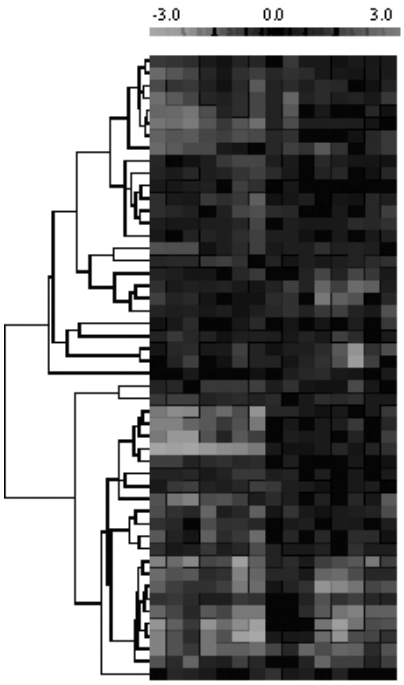

Fig. 3. An example of gene expression data from the organism *Saccharomyces cerevisiae* clustered using hierarchical clustering. The resulting hierarchy is depicted on the left-hand side of the figure. The stripe on the top indicates the spectrum of shades corresponding to the contents of the wells.

3.1 Unsupervised Methods

3.1.1 Hierarchical clustering

Many clustering methods are hierarchical in that they produce a set of nested clusters resembling a dendrogram or phylogenetic tree (Eisen et al. 1998). Each leaf of the hierarchy corresponds to a gene and levels of the hierarchy correspond to partitions of genes into different numbers of clusters (Fig. 4). Hierarchical clustering is a greedy clustering approach, meaning that the best choices are made at each step in the process without regard for future choices. This approach has the advantage of being simple and fast; it produces a final clustering result in the form of a hierarchical tree that is easy to visualize. As a result, hierarchical clustering is the most popular clustering technique. The hierarchical clustering approach proceeds as follows:

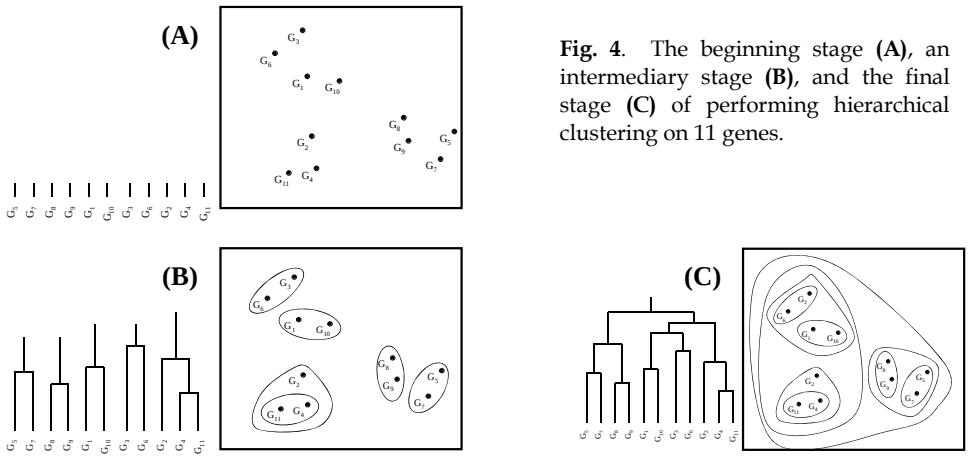

Fig. 4. The beginning stage **(A)**, an intermediary stage **(B)**, and the final stage **(C)** of performing hierarchical clustering on 11 genes.

- Let each gene expression pattern be a cluster containing one data point
- Repeat the following step until all clusters are merged into a single cluster (the root of the hierarchical tree): Find the two clusters with the smallest distance between them and merge them into a single cluster

The results of hierarchical clustering can vary depending on how the distance between two clusters of points is determined. In *single-link clustering*, the distance between two clusters is the shortest distance between any point in one cluster and any point in the other cluster. In *average-link clustering*, the distance between clusters is the distance between cluster centroids. In *complete-link clustering*, the distance between clusters is the farthest distance between any point in one cluster and any point in the other cluster. Different inter-cluster distance measures require different amounts of computation and affect the final topography of the clustering hierarchy.

3.1.2 K-Means

The *k*-means clustering method is a common and relatively simple heuristic method for partitioning data points into *k* clusters (Tavazoie et al. 1999). Unlike hierarchical clustering, no nested clusters are constructed. Instead, for a pre-determined number of clusters *k*, the *k*-means algorithm partitions genes into one of *k* groups such that the sum of distances from each gene expression profile to its cluster center is minimized. As mentioned in section 2, cluster centers (centroids) are generally calculated as the mean of all expression profiles in a given cluster (Figure 5). *k*-means is actually a special case of a maximum-likelihood algorithm applied to a mixture density in which the mixture components are Gaussian distributions with equal variance (Bishop 1995).

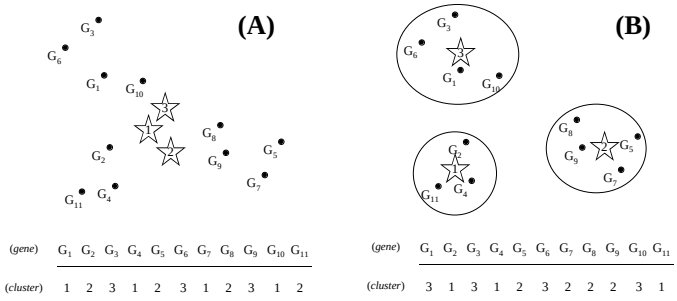

Fig. 5. The start **(A)** and end **(B)** stages of *k*-means clustering. Stars represent the mean of all points assigned to the same cluster. In this example, $k = 3$.

- Randomly assign each gene a cluster number between 1 and *k*
- Repeat until no gene is assigned to a different cluster from the one obtained in the previous iteration (i) For each of the *k* clusters, calculate the mean of all points which are assigned the same cluster number and (ii) For each gene, determine the distance from the gene's expression pattern to each of the *k* means, and assign the gene to the cluster number of the closest of these means

k-means is a fast method for clustering expression data, but has the limitation of requiring prior knowledge of the number of clusters, *k*. If *k* is unknown, one can try different values of *k* and choose the value which yields the most plausible clustering result.

3.1.3 Self-Organizing Maps

Like *k*-means clustering, self-organizing maps (SOMs) create a virtual expression profile for each cluster which serves as a surrogate for the genes belonging to the cluster (Tamayo et al. 1999). Virtual expression profiles do not correspond to actual gene expression profiles as determined from the microarray data, but rather are computer generated expression profiles that are meant to represent a group of actual expression profiles. In *k*-means clustering, for example, the mean of a group of profiles is a virtual profile. SOMs also assign each gene to the cluster whose virtual expression profile is most similar to the gene's expression profile, just as in the case of *k*-means. However, SOMs differ from *k*-means clustering in how they determine the virtual expression profiles for each cluster.

A meta-structure, such as a grid or a lattice, is imposed on the clusters, so that each cluster is connected to a set of neighboring clusters as defined by the meta-structure. The virtual expression profiles for each cluster are determined by sampling genes and determining the cluster which is most similar to the gene's expression profile. The virtual expression profile of the most similar cluster is then updated, along with the virtual expression profile of the neighboring clusters.

- Create random expression profiles for each cluster in the meta-structure
- Repeat until the virtual expression profiles stabilize

    - Sample genes at random and determine the virtual expression profile which is most similar to the gene's expression profile
    - Update the virtual expression profile, as well as that of its neighbors in the meta-structure, to reflect its similarity to the gene's expression profile
- Assign each gene to the cluster whose virtual expression profile is most similar to the gene's expression profile

SOMs are implemented as neural networks, where each neuron in the network corresponds to a cluster or a virtual expression pattern (Kohonen 1997). The neural network is utilized to adjust the meta-structure to better represent the clusters. In the above algorithm, neural networks perform the step of updating virtual profiles. SOMs have the advantage of constructing clusters that conform to a meta-structure, which is often a two-dimensional grid, and thus, is easily represented visually. However, one of the main drawbacks of utilizing SOMs for clustering is that they require the user to specify the meta-structure, including the number of clusters. Such structure in microarray data is rarely known prior to clustering.

3.1.4 Graph-theoretic approaches

Graph-theoretic clustering approaches typically model gene expression patterns as a graph with nodes and edges. Each node in the graph corresponds to a gene and each edge between two nodes in the graph is weighted based on the similarity of the two genes' expression profiles. CAST (Cluster Affinity Search Technique) is a classic heuristic algorithm for clustering which operates by searching for cliques (groups of closely connected nodes) in the graph (Hartuv et al. 1999). Like *k*-means, CAST has a user-defined parameter called a threshold which effectively dictates the number of clusters into which the algorithm will group points.

- Repeat until all points are clustered
    - Choose a nonclustered point and place it into its own cluster
    - Repeat until no points are added/removed from the cluster
        - If the average distance from any nonclustered point to the points within the cluster is less than some threshold, then add the point to the cluster
        - If the average distance from any point in the cluster to the other points in the cluster is greater than the threshold, then remove the point from the cluster
    - Mark all points in the cluster as clustered

3.1.5 Model based clustering

Model based clustering operates on the assumption that gene expression data originates from a finite mixture of underlying probability distributions (Ramoni et al. 2001). Each cluster corresponds to a different distribution, and generally, the distributions are assumed to be Gaussians. The parameters of each distribution (i.e., cluster) are estimated by maximizing the likelihood of the expression data (Hogg and

Craig 1994). The *k*-means approach is a special case of model based clustering, where *all* the distributions are assumed to be Gaussians with equal variance.

- Randomly generate parameters (in the case of Gaussians, the parameters would be the mean and standard deviation or covariance matrix) describing each probability distribution (i.e., cluster)
- Repeat until the parameters of each distribution converge
    - For each gene, estimate the probability that the gene's expression pattern was generated from each of the distributions
    - For each distribution, estimate the parameters of the distribution so as to maximize the likelihood of the expression data given the probability that each gene was generated from the distribution
- Assign each gene to the distribution which generates the gene's expression profile with maximum probability

Model based clustering has the advantage of providing the probability that each gene belongs in each cluster. However, model based clustering operates under the assumption that expression data comes from particular probability distributions, which may not be a reasonable assumption for many microarray data sets.

3.1.6 Principal component analysis

Principal component analysis (PCA) is a linear algebra technique that is akin to singular value decomposition (Raychaudhuri et al. 2000). Although PCA can be used to cluster expression data, PCA is more commonly used as a preprocessing step before applying other clustering algorithms. Its goal is to reduce the dimensionality of the expression data. Since some experiments may be more informative than others in a given set of microarray experiments, PCA offers a method for identifying a small set of virtual experiments which explains most of the variance in the data (Figure 6). The small set of virtual experiments is not necessarily a subset of the given microarray experiments, but rather each virtual experiment in the small set is a linear combination of the entire given set of microarray experiments.

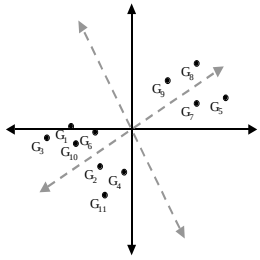

Fig. 6. The expression profiles of 11 genes are plotted along the standard *X* and *Y* coordinate axes (solid arrows). The principal components are shown (dotted arrows) and represent vectors which best distinguish the variance in the plotted data points.

- Calculate the covariance matrix of *A*, the matrix of gene expression data
- Compute the eigenvalues and eigenvectors of the covariance matrix (Hogg and Craig 1994)

- Choose the eigenvectors with the largest eigenvalues, and cluster the expression data (using any appropriate clustering method) in the reduced dimensional space defined by the chosen eigenvectors

Each eigenvector corresponds to a principal component. Intuitively, each component is a linear combination of microarray experiments. The eigenvectors with large eigenvalues are the ones that contain the most information. Those with small eigenvalues are assumed to capture only residual noise in the expression data. PCA has the advantage of reducing the dimensionality of the data by summarizing the microarray experiments. However, the summarization may come at the price of making the expression data less biologically interpretable by researchers.

3.2 Supervised Methods

In contrast to unsupervised clustering where gene clusters are *a priori* unknown, supervised classification is often used when clusters are known for a subset of genes. Those with known clusters, i.e., the training set, can then be used to guide the clustering of genes with unknown clusters. For instance, since genes with related functions often have similar expression patterns, supervised classification may be used to suggest possible roles for genes with unknown function based on the similarity of their expression patterns to those genes with known functions. In this case, the previously annotated genes correspond to genes with known clusters and hypothetical genes correspond to those with unknown clusters. In supervised methods it is very important to select the training set carefully. A good training set should exhibit all of the different patterns which we hope to discern in the data. One of the simplest supervised classification methods is called *k*-nearest neighbors. To classify a target gene, the *k* closest genes in the training set are found, i.e., the *k* genes whose expression profiles are most similar to that of the target gene. The target gene is then assigned to the cluster containing the highest number of the *k* nearest neighbors. The *k*-nearest neighbors' method is a straightforward classification approach that works well when the clusters are compact and the number of clusters is small. However, for genes on the border between clusters, which usually occurs as the number of clusters increases, *k*-nearest neighbors performs unreliably.

Support vector machines (SVMs) are a sophisticated supervised classification method (Brown et al. 2000). Assuming that genes from the training set fall into one of two classes or clusters, an SVM will first map gene expression profiles into a higher dimensional space and then attempt to find a hyperplane which effectively separates the expression profiles of genes from the two classes. Once a separating hyperplane has been established from the training data, genes with unknown classes can be classified based on where their expression profiles fall relative to the hyperplane. SVMs are generally employed as binary classifiers, e.g., when genes belong to one of two classes. While choosing appropriate functions and parameters for an SVM may be more of an art than a science, SVMs are often easier to implement and use than other machine learning techniques such as neural networks (Mitchell 1997).

4. BEYOND CLUSTERING

One of the most challenging applications of microarrays is to infer from their data the properties of the underlying genetic networks. In this section we first review the concept of genetic networks (GNs) in a context often used in bioinformatics; we then show how time-series microarray data can be employed to attempt to deduce GNs representing the dynamic behavior of a cell being studied.

A genetic network can be abstractly represented by a directed graph. Each of its nodes characterizes a gene, and the edges between two nodes *i* and *j* stand for the interaction between those genes. The edges are labeled by the signs + or – to denote activation or repression. For example, a directed edge between *i* and *j* labeled by a minus sign indicates that gene *i* represses the output of the product of gene *j*. A GN graph may also contain Boolean connectors (AND, OR) to indicate the conjunction or the disjunction of the effects of repressions and activations acting upon a given node. Note that the graph may contain loops, including self-loops (genes that affect themselves). The reader will notice that the clustering of microarray data indicating the behavior of thousands of genes allows researchers to replace many of the nodes of a GN graph by their centroid representatives; this in turn implies a significant decrease in processing times needed to analyze the reduced graphs.

It can be shown (Bower and Bolouri 2001) that for each graph representing a GN there is a corresponding set of non-linear differential equations. The variables in those equations reflect the concentrations of gene products at a given time. The non-linearity stems from the fact that gene product concentrations are not linear (they can be approximated by sigmoids).

Given initial conditions for the concentrations of gene products, a numerical solution to the system of differential equations yields a number of curves – each corresponding to a gene - showing how those concentrations vary with time. The curves may also indicate the convergence to stationary states in which all concentrations attain a constant level. The reader might already surmise that GNs are coarse representations of metabolic and signaling pathways. Nevertheless, by omitting details of individual reactions, GNs are convenient abstract tools for studying gene interactions. GNs are also used to develop simulation models of gene behavior (Tomita et al. 1999).

The present challenge of using microarray data to infer GNs amounts to solving a *reverse-engineering* problem: Given the set of curves representing how gene product concentrations vary with time, attempt to generate the corresponding GN. From a mathematical perspective the problem amounts to deriving a GN or a system of differential equations that best describes the set of curves obtained by microarray time series experiments.

In the previous section we have shown the role of clustering in determining the genes that exhibit analogous or opposite actions. Referring back to Figure 2, the correlation factors -1, 0, and +1 correspond to pairs of genes that suppress, remain unaltered, or activate each other. There is a relationship between these factors and the labeling of the edges of a GN. The actual direction of the arrows in a GN can be inferred, for example, by performing microarray experiments with cells having certain

genes knocked out. James Collins and his group have had significant success in reconstructing GNs from microarray data obtained by perturbations of individual genes (Gardner et al. 2003).

Other attempts to solve the reverse-engineering problem include using Bayesian approaches (Nachman et al. 2004), and information theory (Liang et al. 1998) . The on-line Proceedings of the Pacific Symposium in Biocomputing (http://psb.stanford.edu/psb-online) contain a wealth of information about the computational aspects of genetic networks.

5. TOOLS FOR MICROARRAY ANALYSIS

Numerous computer packages exist for microarray data analysis. Both commercial products as well as freely available packages can be found on the World Wide Web. Most products contain tools for data transformation, normalization, and clustering, as described in this chapter. A few of the more popular applications for microarray data analysis are listed below. The website for each application is listed along with the application's availability, either as freely available software or as a commercial product for purchase. The table below differs from the one in the companion chapter since it is focused on clustering packages.

Tool	Free	Website
ArrayVision		http://www1.amershambiosciences.com/
Bioconductor	*	http://www.bioconductor.org/
Cluster and TreeView	*	http://rana.lbl.gov/EisenSoftware.htm
ExpressionProfiler	*	http://www.ebi.ac.uk/expressionprofiler/
GeneCluster	*	http://www.broad.mit.edu/cancer/software/software.html
GeneDirector		http://www.biodiscovery.com/
Genes@Work	*	http://www.research.ibm.com/FunGen/FGDownloads.htm
GeneSifter		http://www.genesifter.net
GeneSpring		http://www.agilent.com/chem/genespring
J-Express Pro		http://www.molmine.com/
MAANOVA	*	http://www.jax.org/staff/churchill/labsite/software/anova/
MIDAS and MEV	*	http://www.tigr.org/software/
Resolver		http://www.rosettabio.com/products/resolver/default.htm
Spotfire		http://www.spotfire.com/

Most of these packages allow the user to select various clustering techniques such as those discussed throughout this chapter. By examining the results of several different clustering techniques, a user will be better prepared to assess the plausibility of the results. Gene expression data is available in a number of public repositories. The table below lists some of the publicly available microarray databases which contain mycology related expression information.

6. CURRENT STATUS AND FUTURE APPLICATIONS OF MICROARRAY USAGE

In addition to more traditional applications of microarray experiments, such as assaying gene expression levels, microarrays have been used for a variety of other

purposes. For example, one of the first steps for scientists after newly sequencing a genome is to annotate the genes by computational analysis. Gene prediction programs are fairly accurate for long, well-conserved genes. However, these programs are less reliable for (1) short genes (less than 100 nucleotides in length), (2) genes which do not code for proteins, and (3) regions of genes that are transcribed but not translated.

Database	Website
ArrayExpress	http://www.ebi.ac.uk/arrayexpress/
ChipDB	http://staffa.wi.mit.edu/chipdb/public/
CIBEX	http://cibex.nig.ac.jp/index.jsp
ExpressDB	http://salt2.med.harvard.edu/ExpressDB/
Gene Expression Omnibus	http://www.ncbi.nlm.nih.gov/geo/
MAD	http://mad.jax.org/
MUSC Database	http://proteogenomics.musc.edu/
PUMAdb	http://puma.princeton.edu/
Stanford Database	http://genome-www5.stanford.edu/
UNC Database	https://genome.unc.edu/
Yale Database	http://info.med.yale.edu/microarray/
yMGV	http://www.transcriptome.ens.fr/ymgv/

Microarrays can be used to assay transcript expression, not only of annotated coding regions of genes, but of the entire genome including intergenic regions (Kapranov et al. 2002). Transcripts which are detected in intergenic regions by microarray experiments may suggest previously undiscovered genes or untranslated regions (UTRs) of annotated genes. This is an example of the great enabling power of microarrays. A single microarray experiment can provide a snapshot of the entire transcript expression of an organism under given conditions. By assaying expression of known genes as well as providing predictive guides for identifying new genes, microarrays facilitate the analysis of rapidly growing genomic data thus increasing our understanding of the cellular machinery of microorganisms. Comparative genome hybridization and the study of genetic variability have become increasingly important in biology for understanding fungal diseases and the mechanisms by which fungi provide drugs such as antibiotics. Microarrays enable rapid comparative genome hybridizations, resequencing, and subsequent genotyping of any organism.

Resequencing applications have become increasingly important in ascertaining the genotypic differences between strains with different phenotypic characteristics. To study the variation of single nucleotide polymorphisms (SNPs), one may consider a small genomic sequence S, containing that nucleotide N. In designing a microarray to detect SNPs, one constructs 4 probes, each specifying variants of S containing a nucleotide that can replace N.

For applications that do not require a specific nucleotide sequence, a global view of genome content can be achieved through comparative genome hybridizations. Such comparisons can help us better understand the evolution of fungi strains, differentiate fungal pathogens from nonpathogens, and identify differences in gene content between fungi strains or their subtypes. This information, coupled with whole genome

expression analysis provides genotypic and phenotypic information that can be used to increase our knowledge of the underlying differences in pathogenesis between two related strains; this information also facilitates the characterization of genes and their functions. Combining *in silico* analyses with comparative genome hybridization assists in determining the genetic heterogeneity of an organism and provides a valuable tool for the mycology laboratory.

Another useful application of microarrays is genome-wide location analysis (Ren et al. 2000). It enables researchers to identify targets of DNA or RNA binding proteins throughout an entire genome via chromatin immunoprecipitation (ChIP). Genome-wide location analysis proceeds as follows. Cells are harvested and disrupted, and DNA fragments cross-linked to a protein of interest are enriched by immunoprecipitation with a specific antibody. Following reversal of the cross-links, enriched DNA is amplified, labeled, and hybridized to a microarray. Microarray probes which evince hybridization correspond to regions of the genome which interact with the protein of interest. Genome-wide location analysis has proven to be a useful technique for elucidating protein-DNA or protein-RNA interactions. For instance, using such an approach, the binding sites of a transcription factor can be identified throughout the genome.

Like DNA microarrays, protein arrays are emerging as a high-throughput platform for identifying protein-protein, protein-antibody, protein-ligand, or protein-drug interactions (Zhu and Snyder 2001). Traditional proteomics techniques, such as 2D gel electrophoresis or chromatography combined with mass spectrometry, are relatively expensive and they may miss proteins expressed at lower levels. Protein arrays are rapidly becoming a valuable tool both to detect protein expression and to investigate protein interactions and function. As with their DNA microarray cousins, protein arrays aim to provide high-throughput experiments, assaying the expression or interaction of many proteins in parallel (Emili and Cagney 2000).

The most common type of protein arrays contain a large number of probes consisting of either proteins or their ligands. The protein array platform is usually either a glass slide or a membrane. Analysis of protein arrays is similar to that of DNA microarrays; this nascent technology has already been used for diagnostics, prognostics, and drug discovery and development.

7. CONCLUSION

It is likely that in a decade or so we will enter the era of personalized medicine in which medication will be prescribed according to the genomic makeup of individual patients. To achieve this objective we will first have to increase the accuracy of microarray experiments and reduce their cost. Once this is done, we will be confronted with a most unusual situation. We will be generating an exponentially increasing number of microarray experiments each of which will have to be analyzed within reasonable times. As we mentioned in the introduction, the exponential nature of the exact clustering algorithms will force us to design increasingly efficient and effective approximate algorithms that, to be widely used, will have to be almost linear, pretty

much like BLAST performs in huge genomic data bases. That will be a great challenge for bioinformaticians in the years to come.

REFERENCES

Bishop CM (1995). Neural Networks for Pattern Recognition. Oxford University Press.

Bower JM and Bolouri H (2001). Computational Modeling of Genetic and Biochemical Networks. MIT Press, Cambridge, MA.

Brown MPS, Grundy WN, Lin D, Cristianini N, Sugnet C, Furey TS, Ares M and Haussler D (2000). Knowledge-based analysis of microarray gene expression data using support vector machines. Proceedings of the National Academy of Science USA 97: 262-267.

Eisen MB, Spellman PT, Brown PO and Botstein D (1998). Cluster analysis and display of genome-wide expression patterns. Proceedings of the National Academy of Sciences USA 95(25): 14863-14868.

Emili AQ and Cagney G (2000). Large-scale functional analysis using peptide or protein arrays. Nature Biotechnology 18: 393-397.

Gardner T, Bernardo D, Lorenz D and Collins J (2003). Inferring Genetic Networks and Identifying Compound Mode of Action Via Expression Profiling. Science 301: 102-105.

Hartuv E, Schmitt A, Lange J, Meirer-Ewert S, Lehrach H and Shamir R (1999). An algorithm for clustering cDNAs for gene expression analysis. Proceedings for the Third Annual International Conference on Research in Computational Molecular Biology: 188-197.

Hogg RV and Craig A (1994). Introduction to Mathematical Statistics. 5th edition. Prentice Hall.

Jiang D, Tang C and Zhang A (2004). Cluster analysis for gene expression data: A survey. IEEE Transactions on Knowledge and Data Engineering: 1370-1386.

Jones N and Pevzner P (2004). An Introduction to Bioinformatics Algorithms. MIT Press. http://www.bioalgorithms.info/

Kapranov P, Cawley SE, Drenkow J, Bekiranov S, Strausberg RL, Fodor SPA and Gingeras TR (2002). Large-scale transcriptional activity in chromosomes 21 and 22. Science 296(5569): 916-919.

Kohonen T (1997). Self-Organizing Maps. Springer-Verlag.

Liang S, Fuhrman S and Somogyi R (1998). REVEAL: A General Reverse Engineering Algorithm for Inference of Genetic Network Architectures. Pacific Symposium on Biocomputing 3:18-29.

Mitchell T (1997). Machine Learning. McGraw Hill.

Nachman I, Regev A and Friedman N (2004). Inferring Quantitative Models of Regulatory Networks from Expression Data. Bioinformatics 20 Suppl. 1:S248-S256.

Quackenbush J (2001). Computational Analysis of Microarray Data. Nature Review Genetics 2: 418-427.

Quackenbush J (2002). Microarray data normalization and transformation. Nature Review Genetics 32: 496-501.

Ramoni MF, Sebastiani P and Kohane IS (2001). Cluster analysis of gene expression dynamics. Proceedings of the National Academy of Sciences USA 99: 9121-9126.

Raychaudhuri S, Stuart JM and Altman RB (2000). Principal components analysis to summarize microarray experiments: application to sporulation time series. Pacific Symposium on Biocomputing: 455-466.

Ren B, Robert F, Wyrick JJ, Aparicio O, Jennings EG, Simon I, Zeitlinger J, Schreiber J, Hannett N, Kanin E, Volkert TL, Wilson CJ, Bell SP and Young RA (2000). Genome-wide location and function of DNA binding proteins. Science 290(5500): 2306-2309.

Tamayo P, Slonim D, Mesirov J, Zhu Q, Kitareewan S, Dmitrovsky E, Lander ES and Golub TR (1999). Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. Proceedings of the National Academy of Science USA 96(6): 2907-2912.

Tavazoie S, Hughes JD, Campbell MJ, Cho RJ and Church GM (1999). Systematic determination of genetic network architecture. Nature Genetics 22(3): 281-285.

Tomita M, Hashimoto K, Takahashi K, Shimizu T, Matsuzaki Y, Miyoshi F, Saito K, Tanida S, Yugi K, Venter JC and Hutchison C (1999). E-CELL: Software environment for whole cell simulation. Bioinformatics 15(1):72-84.

Zhu H and Snyder M (2001). Protein arrays and microarrays. Current Opinion in Chemical Biology 5: 40-45.