Inference methods for gene regulatory networks

By

Nair Asha

Thesis submitted in accordance with requirement for the degree of

MASTER OF SCIENCE

Supervisor:
Dr. Claudio Altafini
SISSA-ISAS International School for Advanced Studies, Trieste

International School for Advanced Studies - SISSA
Trieste
August 31st, 2006.

Acknowledgements

I want to sincerely express my profound gratitude to my supervisor *Dr. Claudio Altafini.* I owe a great deal to him for his valuable assistance, guidance and constant encouragement throughout the course of work.

I wish to offer words of appreciation to Prof. Paolo Carloni and other faculty members in this program.

I wish to extend my appreciation to my colleagues and friends for their many suggestions and support.

Most of all, I express my gratitude to my beloved parents and Anil for giving me moral support throughout my stay in Trieste.

Abstract

Advances in molecular biological and computational technologies are enabling us to systematically investigate the complex molecular processes underlying biological systems. In particular, using high throughput gene expression analysis, we are able to measure the output of the gene regulatory network of a cell. Here, we aim to review some datamining and modeling approaches for conceptualizing and unraveling the functional relationships implicit in these datasets. We discuss some aspects of clustering, ranging from distance measures to clustering algorithms. More advanced analysis aims to infer causal connections between genes directly. We discuss some approaches of reverse engineering of genetic networks and continuous linear model. We conclude that the combination of predictive modeling with systematic experimental verification will be required to gain a deeper insight into living organisms and therapeutic targeting.

Contents

Chapter 1

Introduction

The *genome* i.e the genetic information of an individual is encoded in double stranded deoxyribonucleic acid (**DNA**) molecules. These DNA molecules are arranged into chromosomes in the cell. A ”gene” is written using four-letter alphabet A, C, G and T which are abbreviations for the chemicals adenine, cytosine, guanine and thymine respectively. These chemicals are ’bases’ and together they make up DNA. A gene is ”read” by the cell and directs the cell to synthesize a specific protein. It is the proteins that control all the activities of a cell. Proteins give the cell its shape and function, provide the means to send message within the cells and communicate with other cells, fight invaders.

The *central dogma of biology* (Crick 1958) describes how genes are first transcribed to messenger RNA (mRNA), and then the mRNA is translated into a corresponding protein sequence. Proteins can then be post-translationally modified, localized to certain sites within the cells, and ultimately degraded. (See Fig.(1.1)). The degradation of proteins and intermediate RNA products can also be regulated in the cell. The proteins fulfilling the above regulatory functions are produced by other genes. This gives rise to *genetic regulatory systems* structured by networks of *regulatory interactions* between DNA, RNA, proteins and small molecules. The concerted efforts of genetics, molecular biology, biochemistry and physiology have led to the accumulation of enormous amounts of data on the molecular components of genetic regulatory networks and their interactions. Even though there is an advancement in the mapping of the network structure, surprisingly little is understood about the dynamic behavior of the system that emerges from the interactions between the network components. This has inspired an increasingly large group of researchers to understand the complex patterns of behavior from the interactions between genes in a regulatory network.

The study of genetic regulatory systems has received a major impetus from the recent development of experimental techniques like cDNA microarrays and oligonucleotide chips,

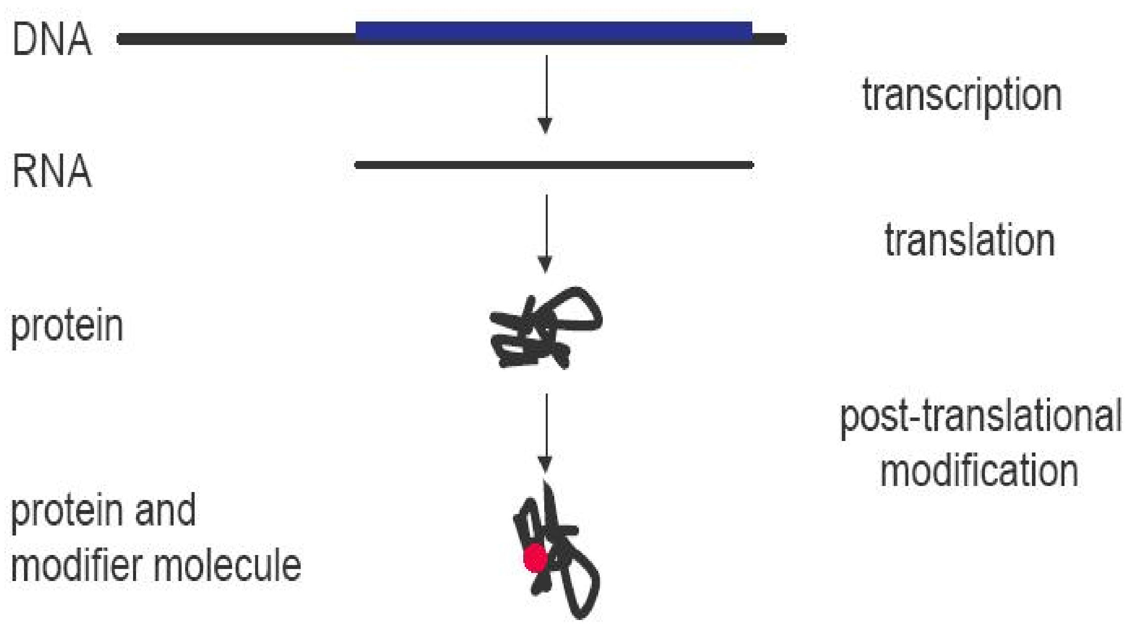

DNA
transcription
RNA
translation
protein
post-translational modification
protein and modifier molecule

Figure 1.1: Genes code for proteins essential for the development and functioning of an organism. Schematic diagram showing gene expression in eukaryotes.

which permit the spatiotemporal expression levels of genes to be rapidly measured in a massively parallel way (Brown and Botstein, 1999; Lipshutz et al., 1999; Lockhart and Winzeler, 2000). Other techniques, such as mass spectrometric identification of gel-separated proteins, allow the state of the cell to be characterized on the proteomic level as well (Kahn, 1995; Mann, 1999; Pandey and Mann, 2000; Zhu and Sydner, 2001). These techniques have become prominent experimental tools for the understanding of the dynamics of gene expression.

In addition to powerful experimental tools, the study of the dynamic behavior of genetic regulatory networks also requires the support of mathematical and computational tools. As most genetic regulatory systems of interest involve many genes connected through interlocking positive and negative feedback loops, their dynamics is hard to understand. The aim is to describe the structure of regulatory systems and to predict the behavior in a systematic way. Modeling and simulation methods and various computer tools permit large and complex genetic regulatory systems to be analyzed.

Figure (1.2) shows the combined application of experimental and computational tools. Starting from an initial model, suggested by the knowledge of regulatory mechanisms and available expression data, the behavior of the system can be simulated for a variety of experimental conditions. Comparing the predictions with the observed gene expression profiles gives an indication of the adequacy of the model. If the predicted and observed behavior do not match, and the experimental data is considered reliable, the model must be revised. The activities of constructing and revising the models of the regulatory network, simulating the behavior of the system, and testing the resulting predictions are repeated until an adequate model is obtained.

The formal basis for computer tools supporting the modeling and simulation tasks in Figure (1.2) lies in methods developed in mathematical biology and bioinformatics. Since the 1960s with some notable precursors in the two preceding decades, a variety of mathematical formulations for describing regulatory networks have been proposed (de Jong. H et. al 2002). These formalisms are complemented by simulation techniques to make behavioral predictions from a model of the system, as well as modeling techniques to construct the model from experimental data and knowledge on regulatory mechanisms. Traditionally, the emphasis has been on simulation techniques, where the models are assumed to have been obtained from the experimental literature. With more experimental data becoming available and easily accessible through databases, modeling techniques are currently gaining popularity.

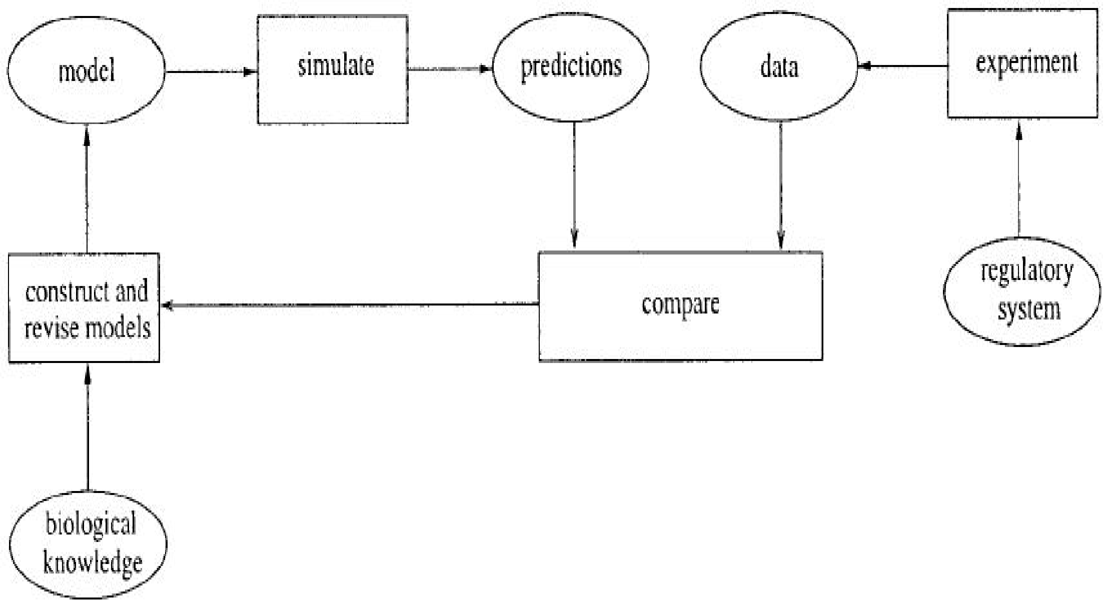


Figure 1.2: Analysis of genetic regulatory systems. Boxes represent activities, ovals information sources and the arrows represent information flows.

1.1 A simple model of gene regulation

The most familiar representation to molecular and cell biologists is a directed graph, with the nodes representing the key elements - often genes, proteins or metabolites - being modelled, and the arcs representing how these influence the production or destruction of others. To formalize this sort of description, one would add weights -positive or negative- to these arcs, and define how the inputs to a node interact. Figure (1.3) illustrates how a simple network model might be represented. Even though it consists of only six nodes, the dynamical behavior of the network is far from obvious. Nevertheless, the network representation provides a clear and concise summary of the regulatory interactions, and higher level structures (such as the two pathways from a to e) can be easily extracted.

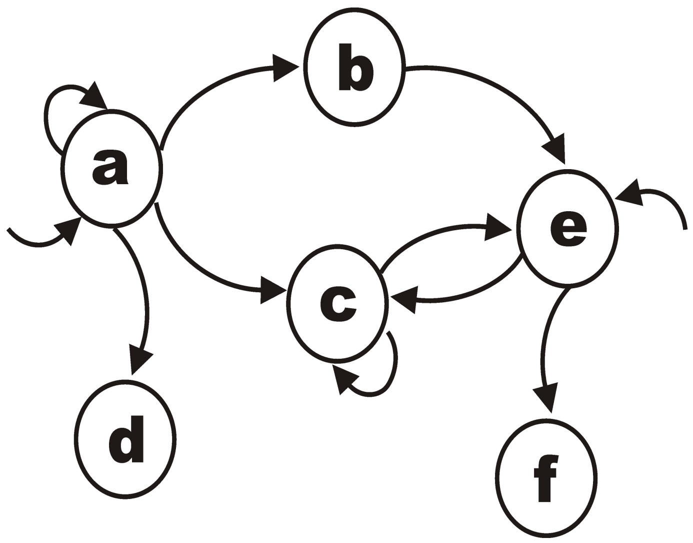


Figure 1.3: Graphical representation of a simple 6-node regulatory network model.

1.1.1 Additive regulation model

One of the simplest ways to model a system of interacting variables is to assume that the change in each variable over time is given by a weighted sum of all other variables.

$$\Delta y_i = \Sigma_j w_{ji} y_j + b_i \tag{1.1}$$

where y_i is the level of the i^{th} variable, b_i is a term indicating whether i is expressed or not in the absence of regulatory inputs, and the weight w_{ji} represents the influence of j on the regulation of i. A is a regulator of B if the network model predicts a causal relationship between the level of A and the change in level of B (i.e., an arrow in the network), regardless of the underlying mechanism of this regulation. This is a more general interpretation of the terms *regulator* and *regulate* than is normally used in biology.

For a continuous time system we get the corresponding differential equation:

$$\frac{dy_i}{dt} = \Sigma_j w_{ji} y_j + b_i \tag{1.2}$$

Because of the nature of interactions between regulatory factors, gene regulation is often context sensitive, e.g. A upregulates C, but only if B is present as well. The model presented here cannot implement such a nonlinear interaction between A and B in the regulation of C. However, the model should be able to extract the linear component of this regulation, i.e both A and B upregulate C, even if the regulation is not independent.

The model like this will be a gross simplification for almost any natural system, but modeling a gene network with such a minimal model might allow us to extract at least the most significant information we are looking for: Which genes regulate which other genes (i.e, which interaction factors w_{ji} are nonzero)?. If gene j regulates gene i, is j an inducer or repressor of i (i.e is w_{ji} positive or negative)?

Chapter 2

Unsupervised methods

Clustering can be considered the most important *unsupervised learning problem* and, as every other problem of this kind, it deals with finding a structure in a collection of unlabelled data. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way". A *cluster* is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.

In order to identify genes of interest from the large amount of data, we need software tools capable of selecting and screening candidate genes for further investigation. (Somogyi, 1999). Normally there is a higher number of genes than experimental points. Hence all classification and reference problems are under-determined (i.e too many degrees of freedom), and the results always depend on the method chosen. This is the main problem one has to deal with, in gene expression analysis. The straightforward method is to classify gene expression patterns to explore shared functions and regulations. This simple approach to clustering consists in selecting a gene and determining its nearest neighbors in expression space within a certain defined distance cut-off. Genes sharing the same expression pattern are likely to be involved in the same regulatory process. Clustering allows us to extract groups of genes that are tightly co-expressed over a range of different experiments.

2.1 Distance measures and preprocessing of data

Most clustering algorithms take a matrix of pairwise distances between genes as input. The choice of distance measure - used to quantify the difference in expression profiles between two genes - may be as important as the choice of clustering algorithm.

2.1.1 Linear metrics

Clustering studies in the gene expression literature use for example Euclidean distance or Pearson correlation between expression profiles as a distance measure. Both these measures are the easiest and most commonly used.
Euclidean distance is given by

$$d_{fg} = \sqrt{\sum_i (e_{fi} - e_{gi})^2} \tag{2.1}$$

The distance induced by Pearson correlation is

$$d_{fg} = 1 - r_{fg}, \quad with \quad r_{fg} = \frac{\sum_i (e_{fi} - \bar{e_f})(e_{gi} - \bar{e_g})}{\sqrt{\sum_i (e_{fi} - \bar{e_f})^2 \sum_i (e_{gi} - \bar{e_g})^2}} \tag{2.2}$$

where d_{fg} is the distance between expression for genes f and g, r_{fg} is the Pearson correlation, e_{gi} is the expression level of gene g under condition i. $\overline{e_f}$ and $\overline{e_g}$ are the average expression level of gene f and gene g respectively.

Different clustering methods can have very different results and at this point it is not yet clear which clustering methods are most useful for gene expression analysis. Each combination of distance measure and clustering algorithm will emphasize different types of regularities in the data. Some may be useless for what we want to do. Others may give us complementary pieces of information.

A related issue is normalization and other preprocessing of the data. Distance measures that are sensitive to scaling and/or offsets (such as Euclidean distance) may require normalization of the data. Normalization can be done with respect to the maximum expression level of each gene, with respect to both minimum and maximum expression levels or with respect to the mean and standard deviation of each expression profile.

2.1.2 Nonlinear metric: Mutual information

With the size of available datasets steadily increasing, it has become feasible to consider other, more general, definitions as well, apart from distance measure. One alternative, based on information theory, is the mutual information, providing a general measure of dependencies between variables. Variables which are not statistically independent suggest the existence of some functional relation between them. While there are several approaches to quantify the linear dependence between variables, the framework of information theory (Shannon, 1948) provides a general measure of dependencies between variables. In particular, Pearson correlation provides the linear relationship between the variables, while the mutual information provides a better and more general criterion to investigate non-linear relationships between variables.

The concept was initially developed for discrete data. Consider a system A, with a finite set of M possible states $a_1, a_2, \cdots, a_M$, the Shannon entropy $H(A)$ is defined as

$$H(A) = -\sum_{i=1}^{M_A} p(a_i) log p(a_i) \tag{2.3}$$

where $p(a_i)$ denotes the probability of the state a_i. The Shannon entropy is a measure for how evenly the states A are distributed. If the state A is completely determined to be a_i, thus if $p(a_i) = 1$ and $p(a_j) = 0$ for all $i \neq j$, one has $H(A) = 0$, whereas the entropy becomes maximal if all probabilities are equal. The joint entropy $H(A, B)$ of two systems A and B is defined as

$$H(A, B) = -\sum_{i=1,j=1}^{M_A, M_B} p(a_i, b_j) log p(a_i, b_j) \tag{2.4}$$

This leads to the relation

$$H(A, B) \leq H(A) + H(B) \tag{2.5}$$

which fulfils equality only in the case of statistical independence of A and B. Mutual information $MI(A, B)$ can be defined as

$$MI(A, B) = H(A) + H(B) - H(A, B) \geq 0 \tag{2.6}$$

It is zero if A and B are statistically independent and increases the less statistically independent A and B are.

Mutual information is defined both for continuous and discrete distributions, but the discrete form is much easier to use. To apply this technique we need to first discretize the gene expression data by partitioning the expression levels into bins. Some regulatory genes exhibit a close approximation to on/off behavior, with several orders of magnitude of difference between basal and induced expression levels. In such a case, the gene expression levels can be discretized without loss of information. However, if part of the regulatory activity of the gene depends on small fluctuations superimposed on the on/off behavior, then this will not be captured by a discretized model. Similarly, the expression levels of some genes mirror continuously varying environmental parameters and have a regulatory effect over their entire range. Discretization of expression levels of such genes will lead to a loss of information. The fewer bins we use to discretize the data, the more information about the original data we ignore. On the other hand, too fine a binning will leave us with too few points per bin to get a reasonable estimate of the frequency of each bin, especially when calculating the joint entropy.

2.2 Clustering algorithms

The art in applying a clustering algorithm in a particular application depends greatly on the particular features of items to be clustered, and on the number of partitions within which these items are to be clustered (Spellman et al 1998, Chen et al 2002).

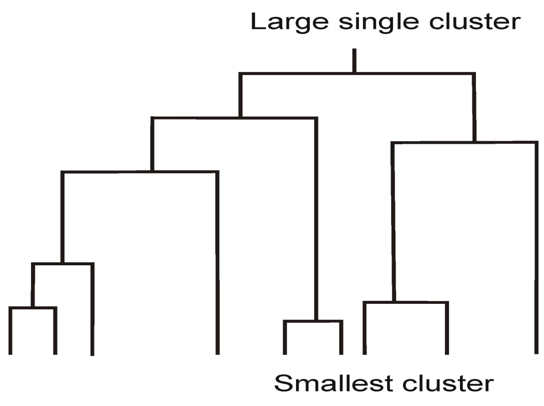
Clustering algorithms can be divided into hierarchical and non-hierarchical methods. Hierarchical clustering represents the relationships between genes as a phylogenetic-type tree structure called a dendrogram, whereas non-hierarchical clustering groups genes into a predefined number of clusters. Non-hierarchical methods typically cluster N objects into K groups in an iterative process until certain goodness criteria are optimized. Examples of non-hierarchical methods include K-means, EM (Expectation - maximization). Hierarchical methods return a hierarchy of nested clusters where each cluster typically consists of the union of two or more small clusters.

2.2.1 Hierarchical clustering

In hierarchical clustering the data are not partitioned into a particular cluster in a single step. Instead, a series of partitions take place, which may run from a single cluster containing all objects to K clusters each containing a single object. Hierarchical clustering is subdivided into *agglomerative* methods, which proceed by series of fusions of N objects into groups, and *divisive* methods, which separate N objects successively into finer groupings. Agglomerative techniques are more commonly used.

Inputs	Data points $x_1, x_2, \cdots, x_N$
Output	Clustering tree

The data points are the leaves and the branching points indicate similarity between the sub-trees. The horizontal cut in the tree produces the data clusters.



General algorithm:

1. Place each element x_i, in its own cluster $C_i = \{x_i\}$.
2. Compute the merging cost between each pair of clusters.
3. Merge the pair of clusters C_i, C_j with cheapest merging cost.
4. Repeat until only one cluster is remaining.

There are differences in the algorithms, because of the different ways of defining *distance* between clusters (i.e the merging cost).

Single Linkage : $min_{x \in C_i, y \in C_j} \quad d(x, y)$

Average Linkage : $\frac{1}{|C_i||C_j|} \sum_{x \in C_i}, \sum_{y \in C_j} \quad d(x, y)$

Complete Linkage : $max_{x \in C_i, y \in C_j} \quad d(x, y)$
where $d(x, y)$ is the distance measure between the elements x and y.

Characteristics of Hierarchical clustering:

- Greedy algorithm - suffers from local optima and builds few big clusters.
- There is a need to choose a threshold on the number of clusters.

2.2.2 K-means clustering

The K-mean algorithm is a popular clustering method, that subdivides the genes into a predetermined number K of clusters. The algorithm is initialized with K randomly chosen cluster centroids (i.e the mean point of each cluster). Each gene is assigned to the cluster with the closest centroid. Elements in the clusters and cluster centroid are updated iteratively until no more genes change the cluster.

Inputs	Data points $x_1, x_2, \cdots, x_N$ and K number of clusters.
Output	K clusters

General algorithm:

1. Select K initial cluster centroids $C_1, C_2, \cdots, C_K$.
2. Assign each element x_i to the cluster with the nearest centroid.
3. For each cluster, recompute its centroid by averaging the data points in it.
4. Repeat until it converges.

Characteristics of K-means clustering:

- Number of clusters K should be chosen in advance.
- It is sensitive to perturbations.
- Results depend on initial choice for centers.

2.2.3 Example

We use the data obtained from the Microarray Hybridization technique for cell cycle regulated genes of yeast Saccharomyces Cerevisiae. (Spellman et. al 1998).
This data was obtained from a publicly accessible website:
http://genome-www.stanford.edu/cellcycle
We applied to it the hierarchical and K-means clustering algorithm. The data consists of 6178 genes $\times$ 82 experiments, in which different external agents are used to synchronize the cell cycle in the entire population, so as to have a measurable pattern. Only the "alpha" synchronization part is chosen for the clustering (i.e 6178 genes $\times$ 18 experiments). See (Spellman et al 1998) for the details. MATLAB is used to do the cluster analysis. The main task here is to hightlight periodicities in the data, which can be used to identify genes involved in the cell cycle of the yeast. At first the data is filtered by eliminating genes with lower expression values, small variances, lower entropies and by taking the log_2 of the values. Now, we have a managable list of gene expression profiles (i.e 3434 genes $\times$ 18 experiments) (shown in Fig (2.1)). This is the data that we use to look for relationships between the profiles using different clustering techniques from the Statistical Toolbox of MATLAB.

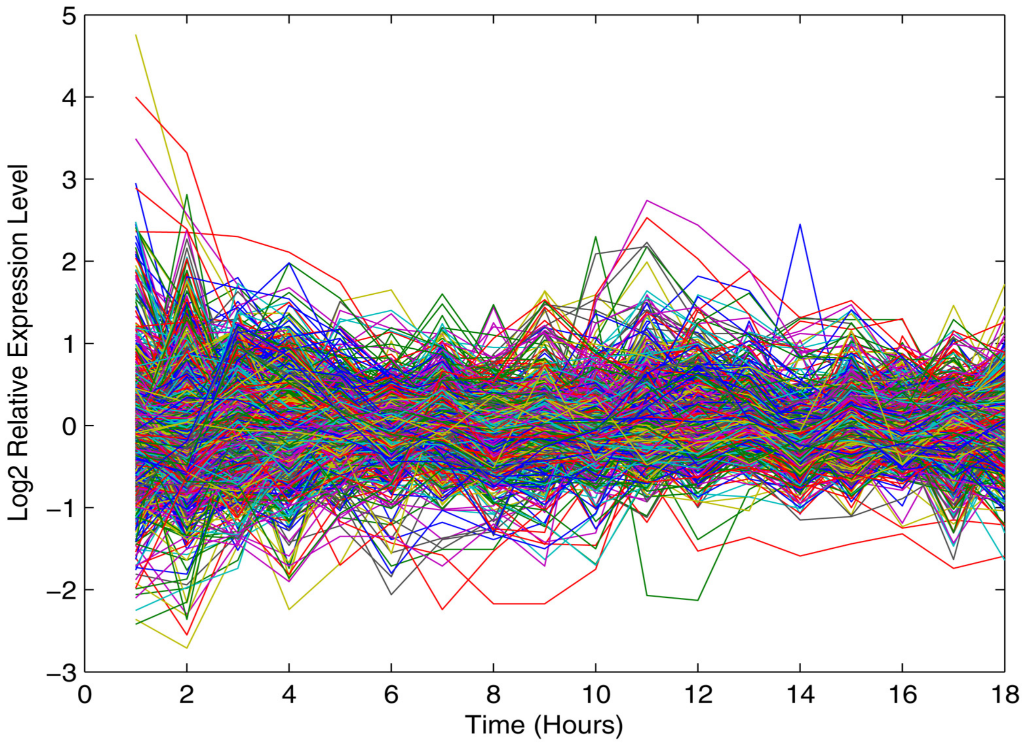


Figure 2.1: Time series of the log of expression profiles of genes.

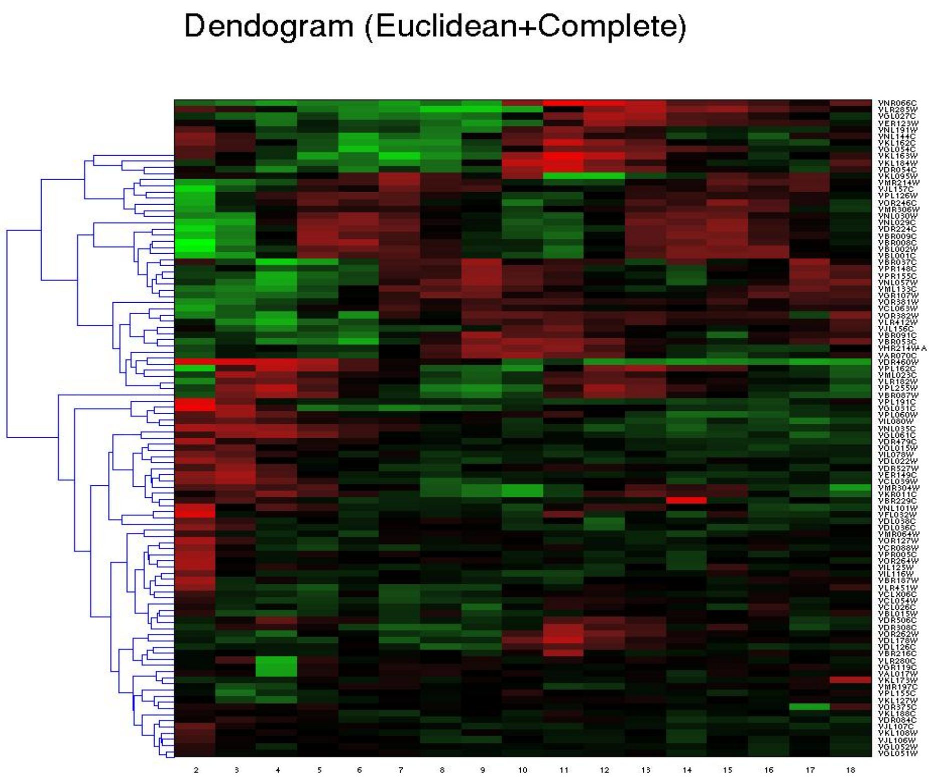


Figure 2.2: A heatmap of hierarchically clustered genes with Euclidean distance measure and complete linkage as cost of aggregation. The rows are the genes and the columns experiments. The dendrogram on the left side defines how the genes are clustered. The intersection of gene and experiment is colored according to the expression value - red indicates high expression, green indicates low expression and the intensity of the color indicates how high or low it is.

For hierarchical clustering, the Matlab function *pdist* calculates the pairwise distance between profiles and *linkage* creates the hierarchical cluster tree. The *cluster* function is used to calculate the clusters based on either a cuttoff distance or a maximum number of clusters. In our case the ”maxclust” option is used to identify 16 distinct clusters.
The result of the hierarchical clustering on the time series is shown in Fig (2.3) and (2.4) for two different choices in *pdist* and *linkage*.

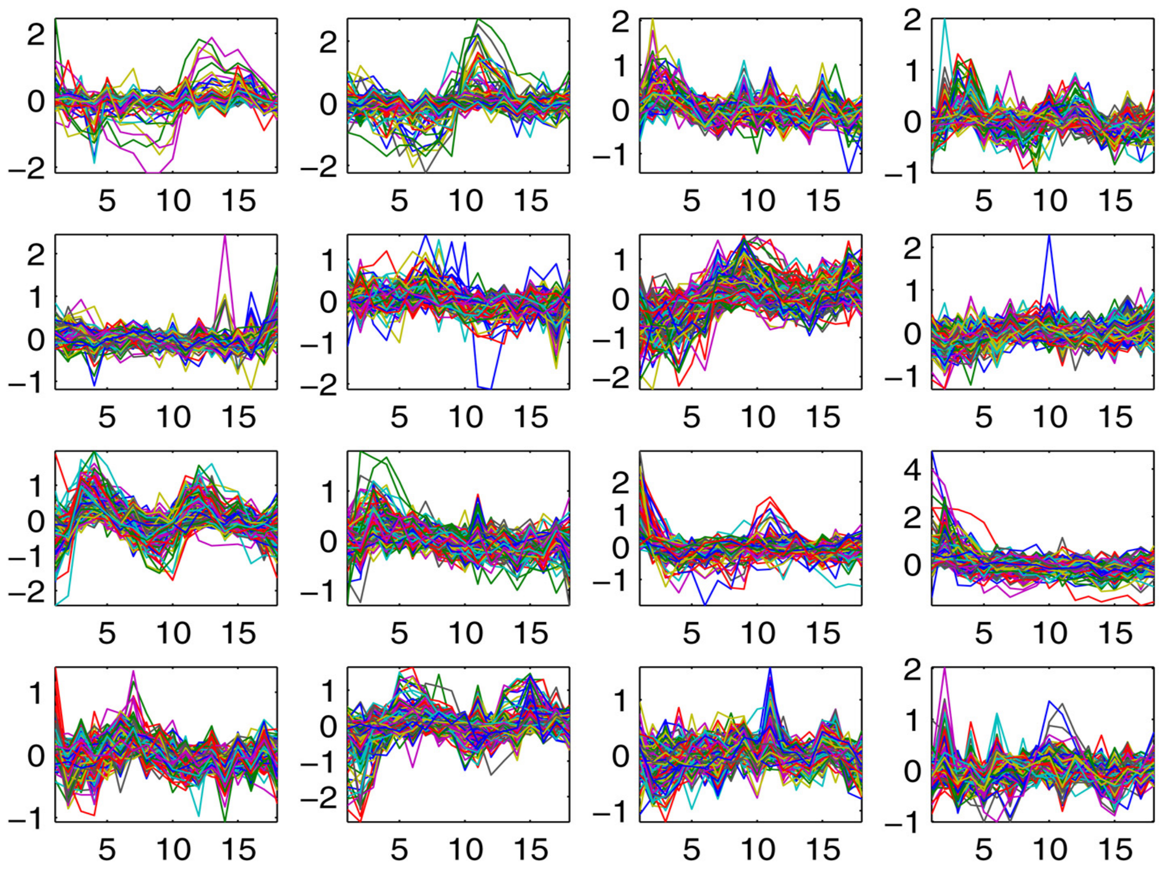


Figure 2.3: Hierarchical clustering of the log of the expression profiles with Euclidean distance measure and complete linkage. 16 clusters, each shown in a square box. Each plot shows the time series of the expression profiles of the genes in each cluster.

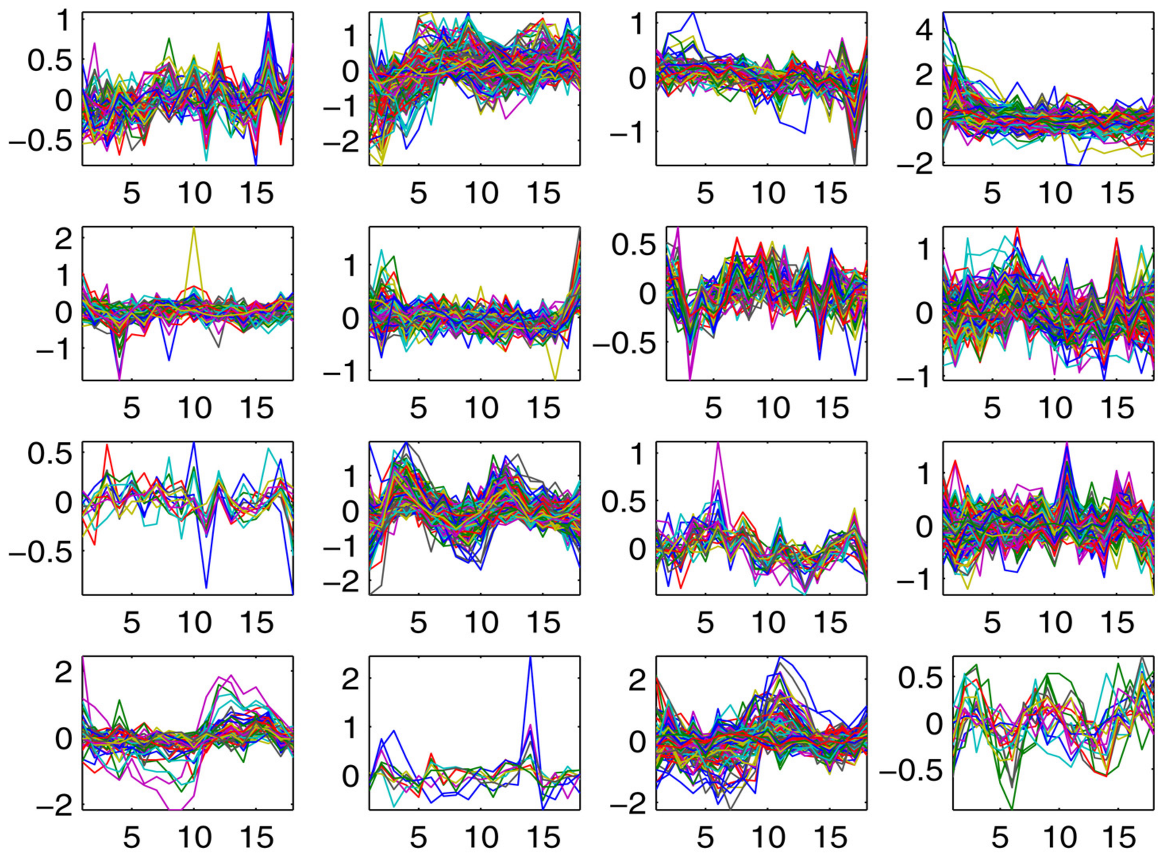


Figure 2.4: Hierarchical clustering of the log of the expression profiles with Correlation distance measure and average linkage. 16 clusters, each shown in a square box. Each plot shows the time series of the expression profiles of the genes in each cluster.

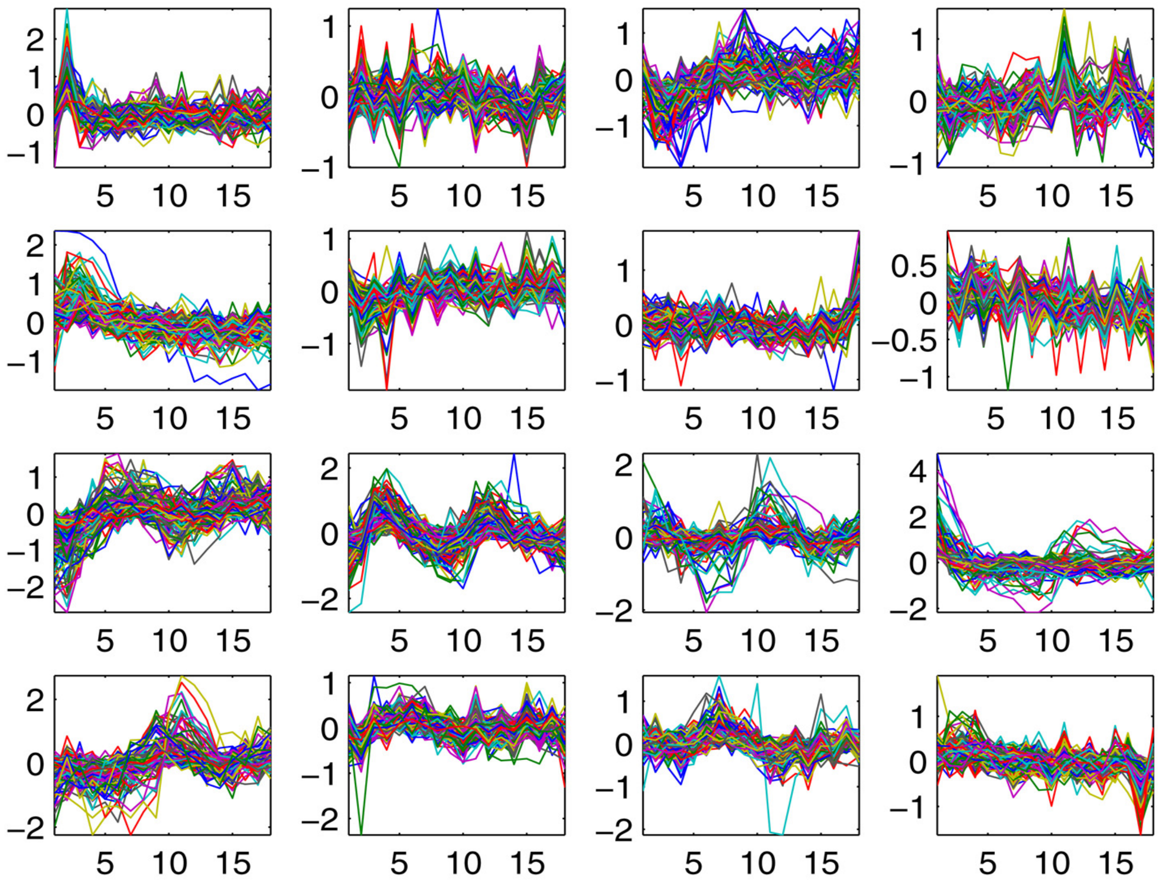


Figure 2.5: K-means clustering expression profiles with correlation distance measure. 16 clusters, each shown in a square box. Each plot shows the time series of the expression profiles of the genes in each cluster.

To see how clusters change with the distance measure chosen we have compared the 16 clusters of Fig (2.5) obtained by a K-means algorithm with correlation distance measure with the 16 clusters of K-means based on Euclidean distance measure Fig (2.6), initialized from the same random seed.
The heat map of Fig. (2.7) shows that only a percentage of the clusters overlap (identical result should correspond to diagonal red squares and blue off diagonal).
Looking to the time profile, however, we see that the ”shared clusters” correspond to some of the most significant patterns (e.g cluster no.(12) from Fig (2.5) and cluster no. (1) from Fig (2.6) contains the most down regulated, non periodic set of genes for the two algorithms).

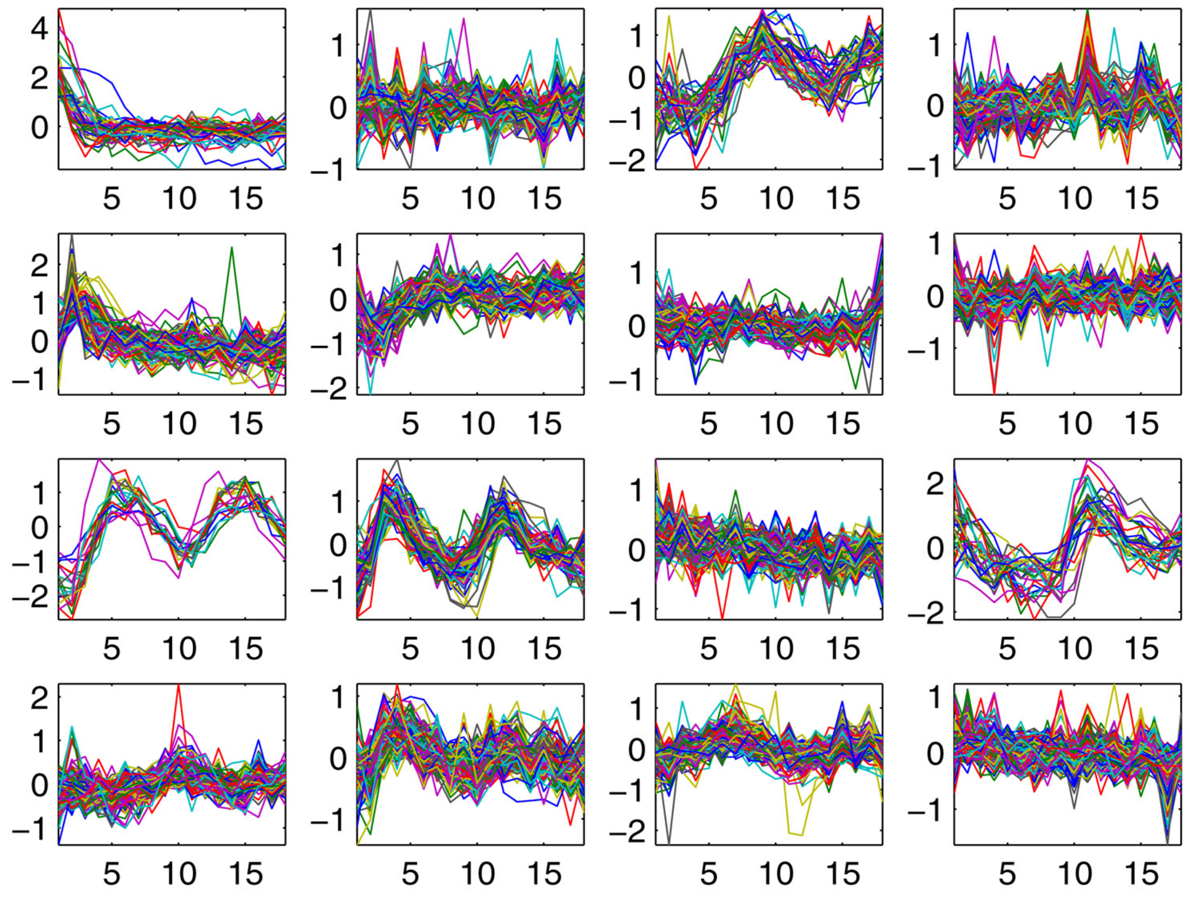


Figure 2.6: K-means clustering expression profiles with Euclidean distance measure. 16 clusters, each shown in a square box. Each plot shows the time series of the expression profiles of the genes in each cluster.

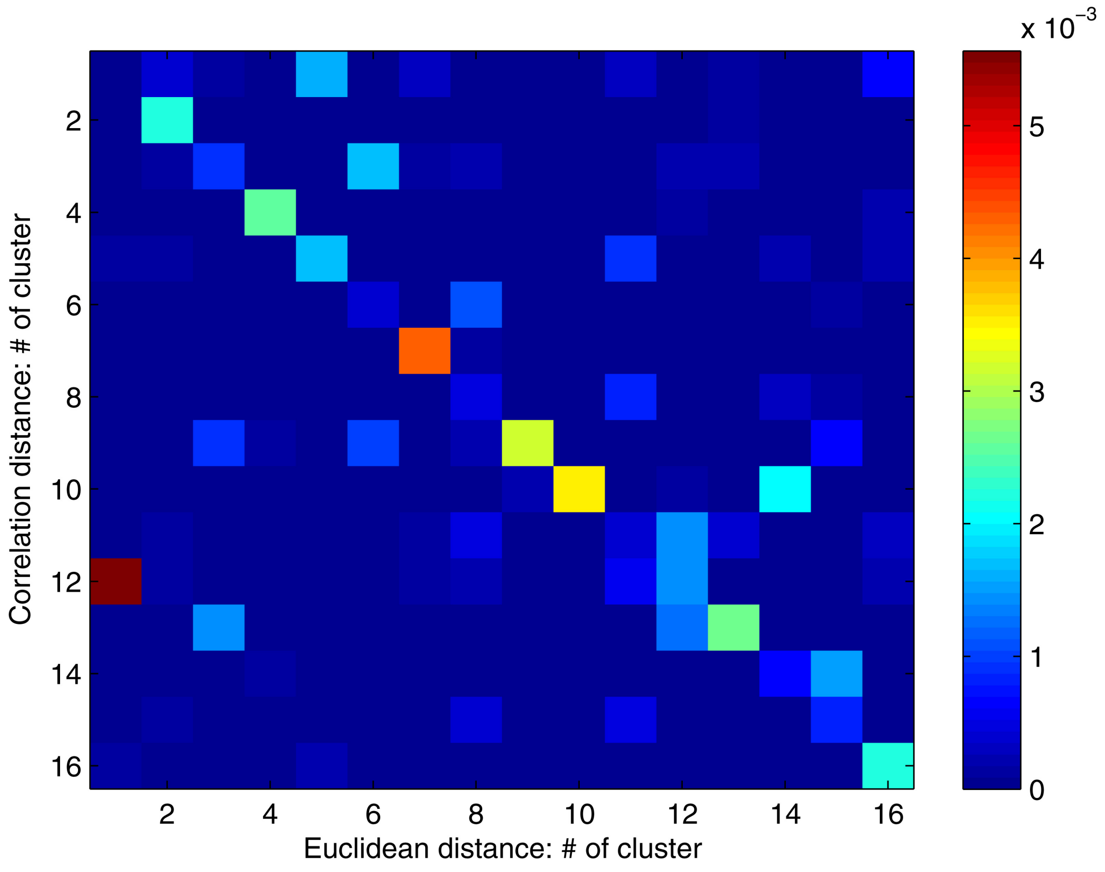


Figure 2.7: A heatmap for the comparison of K-means clustered genes with Euclidean and Correlation distance measure. The two clustering are initialized by the same seed. The perfect matching would be the "red squares" showing 100% overlap of the genes in the same cluster. "Blue squares" indicates 0% overlap. The intensity of the colors in the square indicate the percent overlap of the genes in the same cluster for both algorithms.

2.3 Related methods

The clustering methods are inherently *local* methods, depending on local clustering of genes in high dimensional space determined by the experiments. An alternative approach is to examine some of the more *global* properties of the data, using principal component analysis (PCA) or the related singular value decomposition (SVD). The assumption is that the principal components of gene expression may represent independent regulatory processes. This assumes that the expression level of each gene can be decomposed into a linear superposition of regulatory processes. We proceed with the summary in order to suggest the directions for SVD analysis of gene expression data. (E.Wall et al 2003).

2.3.1 Mathematical definition of SVD

Let X denote an $m \times n$ matrix of real valued data and *rank* r, where without loss of generality $m \geq n$, and therefore $r \leq n$. In the case of microarray data, x_{ij} is the expression level of the i^{th} gene in the j^{th} assay. The elements of the i^{th} row of X form the n-dimensional vector g_i, which we refer to as the transcriptional response of the i^{th} gene. Alternatively, the elements of the j^{th} column of X form the m-dimensional vector a_j, which refer to as the expression profile of the j^{th} assay.

The equation of the singular value decomposition of X is the following:

$$X = USV^{\tau} \tag{2.7}$$

where U is an $m \times n$ matrix, S is an $n \times n$ matrix. The columns of U are called the left singular vectors, (u_k), and form an orthonormal basis for the assay expression profiles, so that $u_i \cdot u_j = 1$ for $i = j$, and $u_i \cdot u_j = 0$ otherwise. The rows of V^{τ} contain the elements of the right singular vectors, (v_k), and form an orthonormal basis for the gene transcriptional responses. The elements of S are only nonzero on the diagonal, and are called the singular values. The $S = diag(s_1, \cdots, s_n)$. Furthermore, $s_k > 0$ for $1 \leq k \leq r$, and $s_i = 0$ for $(r+1) \leq k \leq n$. By convention, the ordering of the singular vectors is determined by high-to-low sorting of singular values, with the highest singular value in the upper left index of the S matrix. It should be noted that for a square, symmetric matrix X, singular value decomposition is equivalent to diagonalization, or solution of the eigenvalue problem. One important result of SVD of X is that

$$X^l = \Sigma_{k=1}^{l} u_k s_k v_k^{\tau} \tag{2.8}$$

is the closest rank-l matrix to X. The term "closest" means that $X^{(l)}$ minimizes the sum of the squares of the difference of the elements of X and X^l, $\Sigma_{ij} \mid x_{ij} - x_{ij}^l \mid^2$.

One way to calculate the SVD is to first calculate V^{τ} and S by diagonalizing $X^{\tau}X$.

$$X^{\tau}X = VS^2V^{\tau} \tag{2.9}$$

and then to calculate U as follows:

$$U = XVS^{-1} \tag{2.10}$$

where the $(r+1), \cdots, n$ columns of V for which $s_k = 0$ are ignored in the matrix multiplication of equation (2.10). Choices for the remaining $n - r$ singular vectors in V or U

may be calculated using some other extension method.

Relation to principal component analysis: There is a direct relation between PCA and SVD in the case where principal components are calculated from the *covariance matrix*. If one conditions data matrix X by centering each column, then $X^\tau X = \Sigma_i g_i g_i^\tau$ is proportional to the covariance matrix of the variables of g_i (i.e the covariance matrix of the assays). By equation (2.9), diagonalization of $X^\tau X$ is done by the matrix V^τ, and also yields the principal components of (g_i). So, the right singular vectors (v_k) are the same as the principal components of (g_i). The eigenvalues of $X^\tau X$ are equivalent to s_k^2, which are proportional to the variance of the principal components. The matrix US then contains the principal component scores, which are the coordinated of the genes in the space of principal components.

If instead each row of X is centered, $X^\tau X = \Sigma_j g_j g_j^\tau$ is proportional to the covariance matrix of the variables if a_j (i.e. the covariance matrix of the genes). In this case the left singular vectors (u_k) are the same as the principal components of a_j. The s_k^2 are again proportional to the variances of the principal components. The matrix SV^τ again contains the principal component scores, which are the coordinates of the assays in the space of principal components.

2.3.2 SVD analysis of gene expression data

In this section, we discuss ways of interpreting the SVD in the context of gene expression analysis.

The biological significance of SVD depends on the specific application. We can, however, consider classes of experiments and provide them as a guide for individual cases. For this purpose, we define two broad classes of applications under which most studies will fall: systems biology applications, and diagnostic applications. In both case, the n columns of the gene expression data matrix X corresponds to assays, and the m rows correspond to the genes. The SVD of X produces two orthonormal bases, one defined by right singular vectors and the other by left singular vectors. Referring to the definitions in previous section, the right singular vectors can span the space of the gene transcriptional responses (g_i) and the left singular vectors span the space of the assay expression profiles (a_j). We refer the left singular vectors (u_k) as *eigenassays* and to the right singular vectors (v_k) as *eigengenes*. By analogy with PCA, it can be referred as a component. Eigengenes, eigenassays and other definitions are shown in Fig.(2.8).

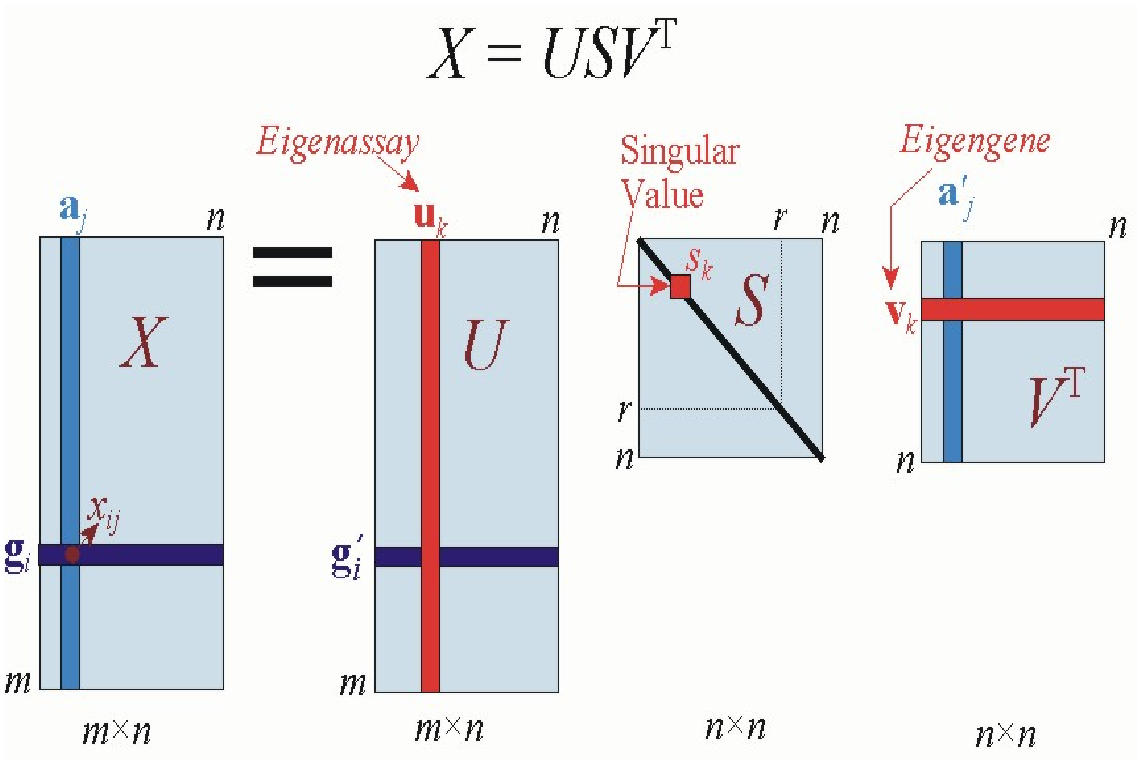


Figure 2.8: Graphical representation of SVD of a matrix X.

In systems biology applications, we generally wish to understand relations among genes. The signal of interest in this case is the gene transcriptional response g_i. From equation (2.7), the SVD equation for g_i is

$$g_i = \Sigma_{k=1}^{r} u_{ik} s_k v_k, i = 1, \cdots, m \tag{2.11}$$

which is the linear combination of eigengenes (v_k). The i^{th} row of U, g'_i (See Fig (2.8)), contains the coordinates of the i^{th} gene in the coordinate system (basis) of the scaled eigengenes, $s_k v_k$. If $r < n$, the transcriptional responses of the genes may be captured with fewer variables using g'_i rather than g_i. This property of the SVD is sometimes referred to as *dimensionality reduction*. In order to reconstruct the original data, however, we still need to access to the eigengenes, which are n-dimensional vectors. Due to the presence of noise in the measurements, $r = n$ in any real gene expression analysis application, though the last singular values in S may be very close to zero and thus irrelevant.

In diagnostic application, we may wish to classify tissue samples from individuals with and without a disease. Referring to the definitions in previous section the signal of interest in this case is the assay expression profile a_j. By equation (2.7) the SVD equation for a_j

is

$$a_j = \Sigma_{k=1}^{r} v_{jk} s_k u_k, j = 1, \cdots, n \tag{2.12}$$

which is a linear combination of the eigenassays (u_k). The j^{th} column of V^{τ}, a'_j (See Fig.(2.8)), contains the coordinates of the j^{th} assay in the coordinate system of the scaled eigenassays, $s_k u_k$. By using the vector a'_j, the expression profiles of the assays may be captured by $r \leq n$ variables, which is always fewer than the m variables in the vector a_j. So, in contrast to gene transcriptional responses, SVD can generally reduce the number of variables used to represent the assay expression profiles. Similar to the case for genes, however, in order to reconstruct the original data, we need access to the eigenassay, which are m-dimensional vectors.

In the literature the number of components that results from such analysis is sometimes associated with the number of underlying biological processes that give rise to the patterns in the data. It is then of interest to ascribe biological meaning to the significant eigenassays (in the case of diagnostic applications), or eigengenes (in the case of systems biology applications). Even though each component on its own may not necessarily be biologically meaningful, SVD can aid in the search for biologically meaning signals. When the data are noisy, it may not be possible to resolve gene groups, but it still may be of interest to detect underlying gene expression patterns, this is the case where the utility of the SVD distinguishes itself with respect to other gene expression analysis methods.

Raychaudhari *et. al.*'s study of yeast sporulation data (Raychaudhari *et al.*2000) is an early application of PCA to microarray analysis. In this study 90% of the variation in the data was explained by the first two components of the PCA. It suggests that much of the observed variability in the experiment can be summarized in just 2-components, i.e two variables capture most of the information. Subsequent reports supported these results (Alter *et al.*, 2000; Holter *et al.*, 2000). Alter *et al.* 2000 analyzed yeast cell-cycle expression data (Spellman *et al.*, 1998), identified sinusoidal modes in the SVD which correspond to cell-cycle modes, and found that 641 out of 784 previously identified cell-cycle genes had at least 25% of their normalized expression signal due to cell-cycle modes. In similar work, Holter *et al.* 2000 analyzed cell-cycle data (Spellman *et al.*, 1998), sporulation data (Chu *et al.*, 1998), demonstrating that groups obtained by cluster analysis tend to cluster in the space of appropriately chosen SVD matrix elements.

We use MATLAB for the PCA analysis of the same filtered data set used in section (2.2.3) consisting of 3434 genes $\times$ 18 experiments. The MATLAB command used to get the principal components is [*pc*, *zscores*, *pcvars*] = *princomp(yeastvalues)*, where *yeastval-*

ues is the manageable list of genes filtered from the original dataset.
The first output *pc*, is a matrix of the principal components of the *yeastvalues* data. The first column of the matrix is the first component, the second column is the second principal component and so on. The second output *zscores*, are the principal component scores. That is, the representation of *yeastvalues* in the principal component space. The third output *pcvars* contains the principal component variances. The values of *pcvars* give a measure of how much of the variance of the data is accounted for by each of the principal components. The function *clusterdata* is used to group the points in the dataset according to the desired distance measure. The function *gscatter* creates a grouped scatter plot where points from each group have a different color or marker (as shown in Fig (2.9) and (2.10)).

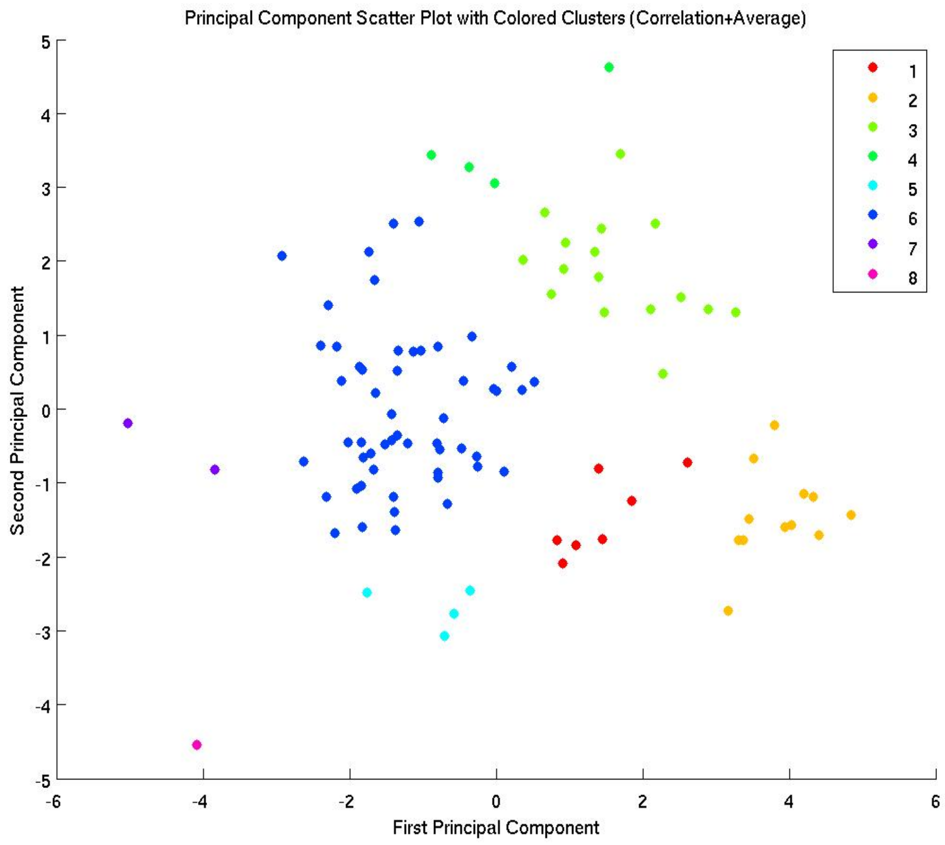


Figure 2.9: The scatter plot of the first and second principal component, with Correlation distance measure and average linkage. 8 clusters, where points from each cluster have a different color.

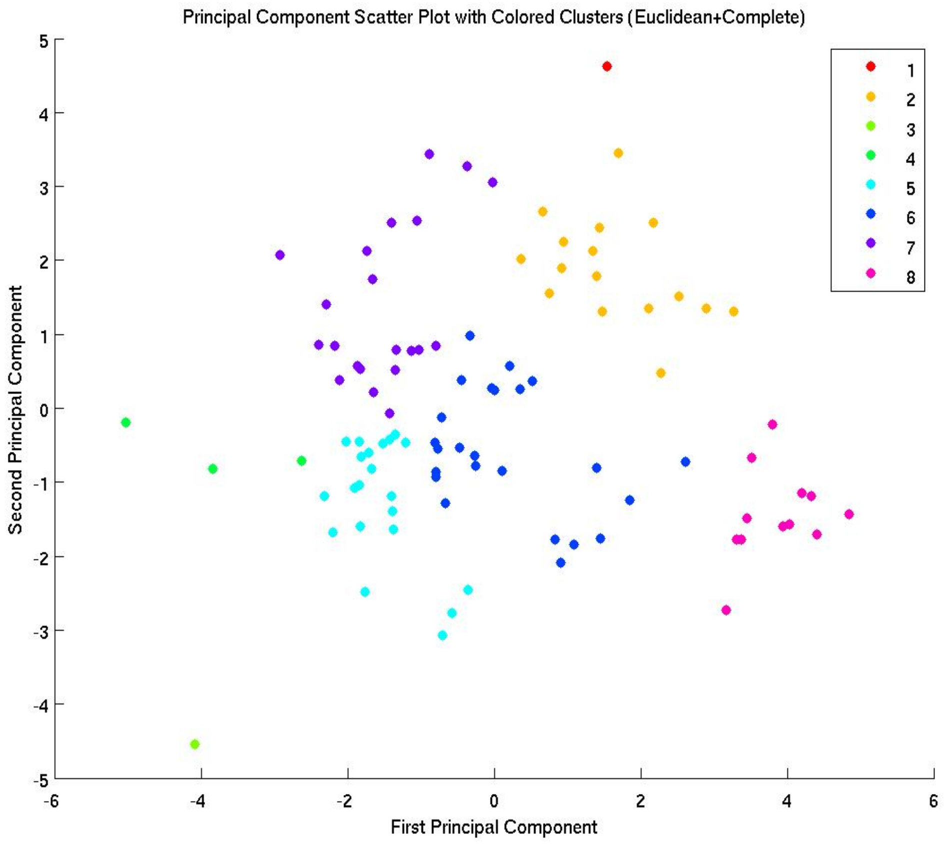


Figure 2.10: The scatter plot of the first and second principal component, with Euclidean distance measure and complete linkage. 8 clusters, where points from each cluster have a different color.

Chapter 3

Gene network inference: reverse engineering

Clustering is relatively easy way to extract useful information out of large scale gene expression data sets. It typically (only) tells us which genes are co-regulated, not what is regulating what. However, the organization of gene expression profile data into functionally meaningful genetic information has proven difficult and so far has fallen short of revealing the intricate structure of cellular interactions (Basso et al 2005). This challenge - called network *reverse engineering* or *deconvolution* - has led to an entirely new class of methods aimed at producing high-fidelity representations of cellular networks as graphs, where nodes represent genes and edges between them represent interactions, either between the encoded proteins or between the encoded proteins and the genes (as discussed in the first chapter).

Reverse engineering is the process of elucidating the structure of a system by reasoning backwards from observations of its behaviors.

The available methods fall into four broad categories:

1. Optimization methods - which maximize a scoring function over alternative network models.
2. Regression techniques - which fit the data to a priori models.
3. Integrative bio-informatics approaches - which combine data from a number of independent experimental clues.
4. Statistical methods - which rely on a variety of measure of pairwise gene expression correlation.

All available approaches suffer to various degrees from problems such as over-fitting, high computational complexity, reliance or non-realistic network models, or a critical dependency on supplementary data that is only available for simple organisms. These limitations have relegated the successful large scale application of most methods to relatively simple organisms, such as the *Saccharomyces cerevisiae* or have produced networks with only a handful of interactions. No method is currently available for the genome wide reverse engineering of mammalian cellular networks.

3.1 Partial correlation

Inferring the topology of gene networks rests mainly on the ability to distinguish direct from indirect relations. Many studies focused on statistical correlation (Pearson or Spearman correlation) between gene expression levels, mostly for dimension reduction techniques and also for network inference. Correlation graphs are formed by connecting any pair of gene nodes by undirected edges whenever the correlation between them is statistically significant. However, it is known that such correlation graphs do not correspond to the actual underlying causal graph of the regulatory system, not only because correlations are undirected, but also because many correlations will be induced indirectly. Higher order correlation could assist in determining which of the correlations in the correlation graph are due to direct effects and which of those are indirectly caused. Thus, although correlation network is still informative, the most important concept in the study of genomic datasets is the partial correlation coefficient (de la Fuente et al 2004). A partial correlation coefficient quantifies the correlation between two variables (e.g gene activities) when there is conditioning on one or several other variables. For example, the correlation $r_{xy.z}$ between variables x and y conditioning on z, is the correlation between the parts of x and y that are uncorrelated with z. The order of the partial correlation coefficient is determined by the number of variables it is conditioned on. $r_{xy.z}$ is a first order partial correlation coefficient, because it is conditioned only one one variable (z).

$$\text{zeroth-order correlation: } r_{xy} = \frac{cov(xy)}{\sqrt{var(x)var(y)}} \tag{3.1}$$

$$\text{first-order correlation: } r_{xy.z} = \frac{r_{xy} - r_{xz}r_{yz}}{\sqrt{(1-r_{xz}^2)(1-r_{yz}^2)}} \tag{3.2}$$

$$\text{second-order correlation: } r_{xy.zq} = \frac{r_{xy.z} - r_{xq.z}r_{yq.z}}{\sqrt{(1-r_{xq.z}^2)(1-r_{yq.z}^2)}} \tag{3.3}$$

Thus, partial correlation coefficients can be used to distinguish between the correlations between two variables due to direct casual relationships from the correlations between the same two variables that originate via intermediate variables or directly due to other variables. In other words, it can be used to eliminate the indirect interaction between the genes.

3.2 ARACNE algorithm

ARACNE (Algorithm for the Reconstruction of Accurate Cellular Networks), is a novel information theoretic algorithm for the reverse engineering of transcriptional networks from microarray data that overcomes some of the limitations of most reverse engineering algorithms, like computational complexity (Margolin et al 2006). Under certain biologically realistic assumptions about the network topology, the ARACNE algorithm provides a framework to reconstruct undirected interaction networks reliably from a finite number of samples in a computationally feasible time.

ARACNE defines an edge as an *irreducible statistical dependency* between gene expression that cannot be explained as an artifact of other statistical dependencies in the network. The presence of such irreducible statistical dependencies is likely to identify direct regulatory interactions mediated by a transcription factor binding to a target gene's promoter region, although other types of interactions may also be identified.

Theoretical Background:
We start by noting that with little temporal gene expression data available for higher eukaryotes, one is forced to study state inter-gene statistical dependencies. Briefly by analogy with statistical physics, we write the joint probability distribution (JPD) of the stationary expressions of all genes, $P(\{g_i\})$, $i = 1, \cdots, N$ as:

$$P(\{g_i\}) = \frac{1}{Z} exp \left[-\sum_i \phi_i(g_i) - \sum_{i,j} \phi_{ij}(g_i, g_j) - \sum_{i,j,k} \phi_{ijk}(g_i, g_j, g_k) - \cdots \right] \equiv exp[-H(\{g_i\})] \tag{3.4}$$

where N is the number of genes, Z is the partition function, $\phi_i(g_i)$ are potentials, and $H(\{g_i\})$ is the Hamiltonian that defines the system dynamics. Then a set of variables is called interacting if and only if the single potential that depends exclusively on these variables is nonzero.
Since the number of realistically obtainable expression profile samples, M is rather small, it is infeasible to infer the exponential number of potential n-way interactions suggested by the expansion in equation (3.4). Rather, a set of simplifying assumptions must be made about the dependency structure. Equation (3.4) provides a principled and controlled way to introduce such approximations. The simplest model is one where genes

are assumed independent, i.e., $H(\{g_i\}) = \sum \phi_i(\{g_i\})$, such that the first order potentials can be evaluated from the marginal probabilities., $P(g_i)$, which are in turn estimated from samples. As more data become available, we should be to reliably estimate higher order marginals and incorporate the corresponding potentials progressively, such that for $M \to \infty$ the complete form of the JPD is restored. In fact, $M > 100$ is generally sufficient to estimate 2-way marginals in genomics problems, while $P(g_i, g_j, g_k)$ requires about an order of magnitude more samples. Thus we truncate equation (1) at pairwise interactions only, $H(\{g_i\}) = \sum_i \phi_i(g_i) + \sum_{ij} \phi_{ij}(g_{ij})$. Within this approximation, two genes are declared non-interacting if they are statistically independent (i.e $P(g_i, g_j) \approx P(g_i)P(g_j)$), and more complex interactions are not investigated.

Algorithm:
ARACNE relies on a two step process. First, candidate interactions are identified by estimating pairwise gene-gene mutual information, $MI(g_i, g_j) = MI_{ij}$, $i, j = 1 \cdots N$ (obtained from equation (2.6)) (See section 2.1.2), and by filtering them using an appropriate threshold, MI_0, computed for a specific p-value, p_0, in the null hypothesis of two independent genes. This is done for different sample sizes M and for 10^5 gene pairs so that reliable estimates of $MI_0(p)$ are produced up to $p = 10^{-4}$.

In the second step, ARACNE removes the vast majority of indirect interactions using a well-known information theoretic property, the data processing inequality (DPI).

Data Processing Inequality: The DPI states that if genes g_1 and g_3 interact only through a third gene, g_2, (i.e if the interaction network is $g_1 \longleftrightarrow g_2 \longleftrightarrow g_3$ and no alternative path exists between g_1 and g_3), then

$$MI(g_1, g_3) \leq min[MI(g_1, g_2); MI(g_2, g_3)] \tag{3.5}$$

Thus the least of the three MI's can come from indirect interactions only, and checking against the DPI may identify those gene pairs for which $MI_{ij} = 0$ even though $P(g_i, g_j) \neq P(g_i)P(g_j)$. Correspondingly, ARACNE starts with a network graph where each $MI_{ij} > MI_0$ is represented by an edge (ij). The algorithm then examines each gene triplet for which all three MI's are greater than MI_0 and removes the edge with the smallest value. Each triplet is analyzed irrespectively of whether its edges have been marked for removal by prior DPI applications to different triplets. Thus the network reconstructed by the algorithm is independent of the order in which the triplets are examined. Since this approach focuses only on the reconstruction of pairwise interaction networks, a pair of mutually independent genes, $MI_{ij} < MI_0$, will never be connected by an edge.

Algorithmic complexity:
Because of a network of N genes there are at most N^3 gene triplets, ARACNE's complexity is $O(N^3 + N^2M^2)$, where M is the number of samples and N is the number of genes. The first term relates to the DPI analysis and the second to the mutual information estimation. In practice, the DPI is applied to a small subset of triplets for which all three edges survive the mutual information thresholding. Therefore, for large M, the computationally intensive part is generally associated with the second term (i.e computing mutual information), which scales as $O(N^2M^2)$. As a result, ARACNE can efficiently analyze networks with tens of thousands of genes.

Chapter 4

Gene network inference: differential equations

Ordinary differential equations offers a description of the network as a continuous time dynamical system that can be used to infer the genes with the major regulatory functions in the network. In addition, it can be applied to the RNA expression measurements obtained from pharmacological perturbations to identify the genes that directly mediate a compound's bio-activity in the cell. In a recent study (Gardener et al.(2003)), an algorithm was developed to identify a genetic network of nine genes, as a set of linear differential equations, starting from measurements of gene expression at steady state following transcriptional perturbations.

4.1 Network model description

A network can be described by a set if ordinary differential equations describing the time evolution of the mRNA concentration of the genes in the network.

$$\dot{x} = f(x, u) \tag{4.1}$$

where x represents the mRNA concentrations of the genes in the network, and u is a set of transcriptional perturbations. Assuming that the cell under investigation is at equilibrium near a stable steady state point, a small perturbation can be applied to each of its genes. A perturbation is small if it does not drive the network out of the basin of attraction of its stable steady state point and if the stable manifold in the neighborhood of the steady state point is approximately linear. With these assumptions, the set of nonlinear rate equations can be linearized near its stable equillibrium point. Thus, for each gene, i, in a network of N genes we can write the following equation:

$$\dot{x_{il}} = \sum_{j=1}^{N} a_{ij}x_{jl} + u_{il} = a_i^{\mathsf{T}} \cdot x_l + u_{il}, \quad i = 1 \cdots N, l = 1 \cdots M, \tag{4.2}$$

where x_{il} is the mRNA concentration of gene i following the perturbation in experiment l; a_{ij} represents the influence of gene j on gene i; u_{il} is an external perturbation to the expression of gene i in experiment l. For all N genes, equation(4.2) can be rewritten in more compact form using matrix notation:

$$\dot{x_l} = \mathbf{A} \cdot x_l + u_l, \quad l = 1 \cdots M, \tag{4.3}$$

where x_l is an $N \times 1$ vector of mRNA concentrations of the N genes in experiment l, $\mathbf{A}$ is an $N \times N$ connectivity matrix, composed of elements a_{ij}, and u_l is an $N \times 1$ vector of the perturbations applied to each of the N genes in experiment l.

To identify the network, using the model described above, means to retrieve matrix $\mathbf{A}$. This is possible if we measure mRNA concentration of all the N genes at steady state (i.e, $\dot{x_l} = 0$) in M experiments and then solve the system of equations:

$$\mathbf{A} \cdot \mathbf{X} = -\mathbf{U} \tag{4.4}$$

where $\mathbf{X}$ is an $N \times M$ matrix composed of columns x_l; $\mathbf{U}$ is an $N \times M$ with each column u_l. Equation (4.4) can be solved only if $M \geq N$. However, the recovered weights $\mathbf{A}$, will be extremely sensitive to noise both in the data $\mathbf{X}$ and in the perturbations, $\mathbf{U}$, and thus unreliable unless we over-determine the system of equations (4.4). This can be accomplished either by increasing the number of experiments ($M > N$), or by assuming the maximum number of regulators acting on each gene, k is less than M (i.e, the network is not fully connected) thus reducing the number of weights a_{ij} to be recovered.

4.2 Algorithm

A genetic network can be described by the system of linear differential equations (4.2). For each gene i at steady state ($\dot{x_{il}} = 0$) in experiment l, we can therefore write:

$$-u_{il} = a_i^{\mathsf{T}} \cdot x_l \tag{4.5}$$

where u_{il} is the transcriptional perturbation applied to gene i in experiment l, a_i^{T} is a row of $\mathbf{A}$, and x_l ($N \times 1$) are the mRNA concentrations at the steady state following the perturbation in experiment l. The algorithm assumes that only k out of N weights in a_i for gene i are different from zero. For each possible combination of k out of N weights, the algorithm computes the solution to the following linear regression model:

$$y_{il} = b_i^{\tau} \cdot z_l + \epsilon_{il} \tag{4.6}$$

where $y_{il} = -u_{il}$ is the perturbation applied to gene i in experiment l; b_i is a $k \times 1$ vector representing one of $\frac{N!}{k!(N-k)!}$ possible combinations of weights for gene i; z_l is a $k \times 1$ vector of mRNA concentrations following the perturbation in experiment l (i.e a sub-vector of x_l), with added noise ϵ_{il}. Equation (4.6) represents a multiple linear regression model.

If we collect data in M different experiments, then we can write equation (4.6) for each experiment and obtain the system of equations:

$$y_i^{\tau} = b_i^{\tau} \cdot \mathbf{Z} + \epsilon_i^{\tau} \tag{4.7}$$

where y_i is an $M \times 1$ vector of measurements of the perturbation y_{il} to gene i in the M experiments; $\mathbf{Z}$ is a $K \times M$ matrix, where each column is the vector z_l for one of the M experiments; ϵ_i is an $M \times 1$ vector of noise in M experiments. From equations (4.8), it follows that a predictor for y_i given the data matrix $\mathbf{Z}$ is:

$$\widehat{y_i}^{\tau} = b_i^{\tau} \cdot \mathbf{Z} \tag{4.8}$$

The following cost function is minimized to find the k weights b_i, for gene i:

$$C_i^k = \sum_{l=1}^{M} (y_{il} - \widehat{y_{il}})^2 = \sum_{l=1}^{M} (y_{il} - b_i^{\tau} \cdot z_l)^2 \tag{4.9}$$

The solution can be obtained by computing the pseudo inverse of $\mathbf{Z}$:

$$\widetilde{b_i} = (\mathbf{Z} \cdot \mathbf{Z}^{\tau})^{-1} \cdot \mathbf{Z} \cdot y_i \tag{4.10}$$

The solution $\widetilde{b_i}$, is not the maximum likelihood estimate for the parameters b_i, when the regressors $\mathbf{Z}$ are stochastic variables, but it is nevertheless a good estimate for the unknown $\mathbf{A}$. The best approximation of the weights in equations (4.2) for gene i, is selected the one with the smallest least-square error, C_i^k, among the (N choose k) possible solutions $\widetilde{b_i}$.

It is possible to use the recovered network $\widetilde{\mathbf{A}}$ to deconvolve the results of an experiment, i.e., to recover the unknown perturbations u_0 in an experiment, given the measurements of the response to that perturbation, x_0. The predicted perturbations $\widetilde{u_0}$ can be computed from:

$$\widetilde{u_0} = -\widetilde{\mathbf{A}} \cdot x_0 \tag{4.11}$$

The recovered network can be used for target prediction, which can be very useful for drug discovery. Using measurements of mRNA concentration changes at steady state following the application of a compound to a cell population, the direct targets of that drug in the large gene network can be predicted, by means of the recovered network model.

Chapter 5

Conclusion

We are witnessing the transition of biology from a mainly qualitative and descriptive science into quantitative science. This transition, can be meaningful only if the focus is on generating models that allows to derive systematic predictions about important biological processes. This will find important applications in pharmaceutical development and bioengineering. We have reviewed conceptual foundations for understanding complex biological networks, but there are still major challenges ahead. Some of them are quoted below:

Computational data analysis must identify the most essential molecular parameters to guide the experimental measurements, and critically evaluate measurement precision and reproducibility with appropriate statistical measures.

Methods for clustering according to co-expression profiles should select the appropriate experimental sets for analysis, and provide flexible solutions that more accurately reflect the biological reality. Well designed cluster analysis can be helpful in identifying new pathway relationships and gene functions that may be critical to cellular control in health and disease.

Top priority should be given to develop reverse engineering methods that provide significant predictions. Alternative computational approaches should be applied to given data sets, and their predictions tested in the experiments to identify the most reliable methods.

With the current focus on the analysis of large scale gene expression data, there are other established sources of information ranging from sequence homology to disease association and a wide variety of functional knowledge from targeted experiments. The gene regulatory system investigated here is just a "layer" of the "fundamental dogma". Proteomic and metabolic levels of investigation should be added likewise, in order to make the reverse engineering methods more realistic. Ideally, all these categories of information should be included in the model building.

Bibliography

[1] Alter, O., Brown, P., Botstein, D., Singular value decomposition for genome wide expression data processing and modeling. (2000) *Proceedings of the Nat. Acad. of Sci.* **97**, 10101-10106.

[2] Basso, K., Margolin, A., Stolovitzky, G., Klein, U., Dalla-Favera, R. and Califano, A. (2005) Reverse engineering of regulatory networks in human B cells. *Nat Genet*, **37**, 382-390.

[3] Butte, A and Kohane, I.S. (2000). Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements. *Pac. Symp. Biocomput.*, **5**, 415-426.

[4] Chen, G., Jaradat, S.A., Benerjee, N., (2002) Evaluation and comparison of clustering algorithms in analysis of ES cell gene expression data. *Statistica Sinica*, **12**, 241-262.

[5] Cover, T.M. and Thomas, J.A. (1991). *Elements of Information Theory.* Wiley, New York.

[6] Chu, T., Glymour, C., Scheines, R., and Spirtes, P., (2003) A statistical problem for inference to regulatory structure from associations of gene expression measurements with microarrays. *Bioinformatics*, **19**, 1147-1152.

[7] D'haeseleer, P., Liang, S. and Somogyi, R. (2000). Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, **16**, 707-726.

[8] de Jong, H. (2002) Modeling and simulation of genetic regulatory systems: A literature review. *J. comp. Biol.*, **9**, 67-103.

[9] de la Fuente, Nan, B., Ina, H., Mendes, P., Discovery of meaningful assoiciations in genomic data using partial correlation coefficients. *Bioinformatics*, (2004) **20**, 3565-3574.

[10] di Bernardo, D., Gardener, T.S and Collins, J., (2004) Robust identification of large genetic networks. *Proc. Pac. Symp. Biocomp.,* **9**, 486-497.

[11] di Bernardo, D., Thompson, M., Gardner, T., Chobot, S., Eastwood, E., Wojtovich, A., Elliott, S., Schaus, S., and Collins, J. (2005) Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nature Biotechnology*, **23**, 377-383.

[12] Gardner, T., di Bernardo, D., Lorenz, D., and Collins, J., (2003) Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, **301**, 102-105.

[13] Eisen, M.B., Spellman, P.T., Brown, P.O, and Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl Acad. Sci. USA*, **95**, 14863-14868.

[14] Holter, S., Mitra, M., Maritan, A., Cieplak, M., Banavar, J.R., Fedoroff, N.V., Fundamental patterns underlying gene expression profiles: Simplicity from complexity. *Proceedings of Nat. Acad. of Sci* (2000). **97**, 8409-8414.

[15] Hughes, T.R., Marton, M.J., Jones, A.R., Roberts, C.J., Stoughton R, Armour, C.D., Bennett, H.A., Coffey, H.A., Dai H, He, Y.D., Kidd, M.J., King, A.M., Meyer, M.R., Slade D, Lum, P,Y., Stepaniants, S.B., Shoemaker, D,D., Gachotte D, Chakraburtty K, Simon J, Bard M, Friend S,H: Functional discovery via a compendium of expression profiles. *Cell*, 2000. **102**, 109-126.

[16] Lee, T.I., Rinaldi, N.J., Robert, F., Odom, D.T., Bar-Joseph, Z., Gerber, G.K., Hannett, N.M., Harbison, C.T., Thompson, C.M., Simon, I., Zeitlinger, J., Jennings, E.G., Murray, H.L., Gordon, D.B., Ren, B., Wyrick, J.J., Tagne, J.B., Volkert, T.L., Fraenkel, E., Gifford, D.K., and Young, R.A. (2002) Transcriptional regulatory networks in Saccharomyces cerevisiae. *Science*, **298**, 799-804.

[17] Margolin, A.A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Dalla Favera, R., Califano, A. ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context. (2006), *BMC Bioinformatics*, 7 Suppl 1:S7.

[18] Mendes, P., Sha, W. and Ye, K., (2003) Artificial gene networks for objective comparison of analysis algorithm. *Bioinformatics.* **19**, Suppl 2: II122-II129.

[19] Mukesh, B., Guisy, D.G., di Bernardo, D., Inference of gene regulatory networks and compund mode of action from time course gene expression profiles. *Bioinformatics*, (2006). **22**, 815-822.

[20] S. Raychaudhari, Staurt, J.M., Altman, R. B., (2000) Principal component analysis to summarize microarray experiments: application to sporulation time. *Pac Symp Biocomp* (2000), 455-466.

[21] Steuer, R., Kurths, J., Daub, C.O., Weise, J. The mutual information: Detecting and evaluating dependencies between variables. *Bioinformatics.* (18), Suppl 2: S231-S240 (2002).

[22] Tegner, J., Yeung, M.K., Hasty, J., and J, C.J. (2003) Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling. *Proc. Natl Acad. Sci. USA*, **100**, 5944-5949.

[23] Wall, Michael E., Rechtsteiner, A., Rocha, L.M., (2003) Singular value decomposition and principal component analysis. *A Practical Approach to Microarray Data Analysis*, 91-109.

[24] Yeung, M.K.S., Tegner, J., and Collins, J.J. (2002)Reverse engineering gene networks using singular value decomposition and robust regression. *Proc. Natl. Acad. Sci., USA*. **99**, 6163-6168.