Summer 8-2009

Inferring Gene Regulatory Networks from Time Series Microarray Data

Peng Li
University of Southern Mississippi

The University of Southern Mississippi

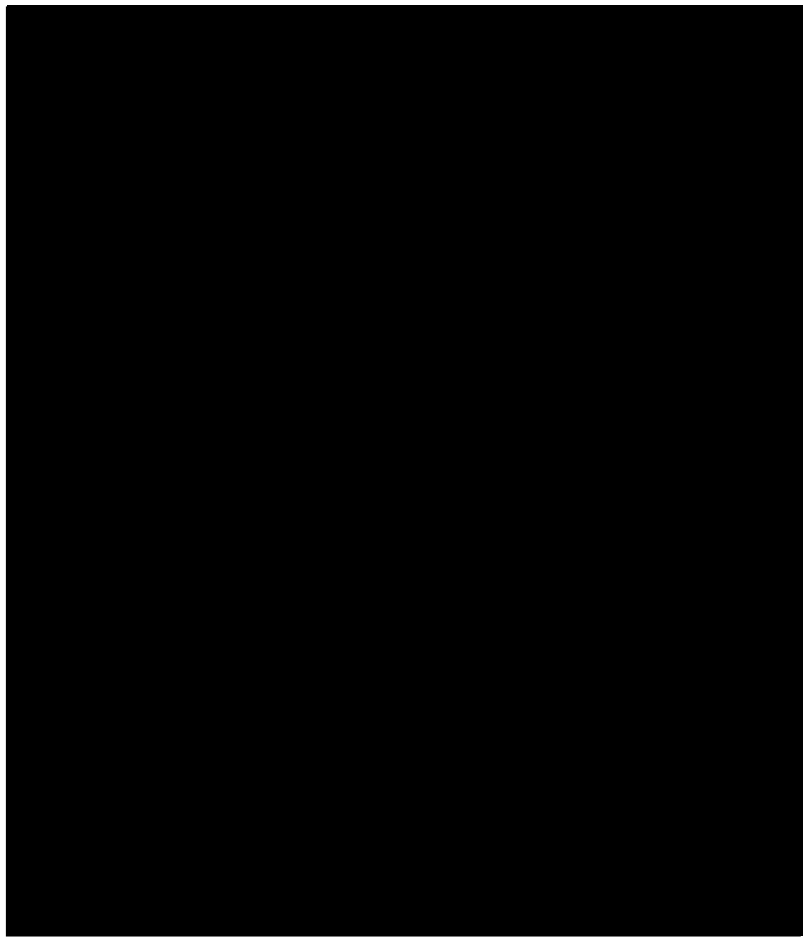
INFERRING GENE REGULATORY NETWORKS FROM TIME SERIES MICROARRAY DATA

by

Peng Li

A Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved:



August 2009

The University of Southern Mississippi

INFERRING GENE REGULATORY NETWORKS FROM TIME SERIES MICROARRAY DATA

by

Peng Li

Abstract of a Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

August 2009

ABSTRACT

INFERRING GENE REGULATORY NETWORKS FROM TIME SERIES MICROARRAY DATA

by Peng Li

August 2009

The innovations and improvements in high-throughput genomic technologies, such as DNA microarray, make it possible for biologists to simultaneously measure dependencies and regulations among genes on a genome-wide scale and provide us genetic information. An important objective of the functional genomics is to understand the controlling mechanism of the expression of these genes and encode the knowledge into gene regulatory network (GRN). To achieve this, computational and statistical algorithms are especially needed.

Inference of GRN is a very challenging task for computational biologists because the degree of freedom of the parameters is redundant. Various computational approaches have been proposed for modeling gene regulatory networks, such as Boolean network, differential equations and Bayesian network. There is no so called "golden method" which can generally give us the best performance for any data set. The research goal is to improve inference accuracy and reduce computational complexity.

One of the problems in reconstructing GRN is how to deal with the high dimensionality and short time course gene expression data. In this work, some existing inference algorithms are compared and the limitations lie in that they either suffer from low inference accuracy or computational complexity. To overcome such difficulties, a new approach based on state space model and Expectation-Maximization (EM)

algorithms is proposed to model the dynamic system of gene regulation and infer gene regulatory networks. In our model, GRN is represented by a state space model that incorporates noises and has the ability to capture more various biological aspects, such as hidden or missing variables. An EM algorithm is used to estimate the parameters based on the given state space functions and the gene interaction matrix is derived by decomposing the observation matrix using singular value decomposition, and then it is used to infer GRN. The new model is validated using synthetic data sets before applying it to real biological data sets. The results reveal that the developed model can infer the gene regulatory networks from large scale gene expression data and significantly reduce the computational time complexity without losing much inference accuracy compared to dynamic Bayesian network.

TO MY PARENTS

FOR THEIR LOVE AND SUPPORT

ACKNOWLEDGEMENTS

I would like to thank Dr. Chaoyang Zhang and all the other committee members, Dr. Ray Seyfarth, Dr. Jonathan Sun, Dr. Ping Gong, Dr. Andrew Strelzoff and Dr. Youping Deng for their suggestions and advices to improve my work during the whole process. I gratefully acknowledge my advisor, Dr. Zhang, for his generous helps in the past four years. Along the path of research that has led to this dissertation, he has constantly been there to provide me guidance, support and suggestions. I also would like to thank Dr. Ping Gong, Dr. Youping Deng and Dr. Edward Perkins for their comments and providing high quality data during my doctoral research.

I am forever indebted to my parents, who have been extremely supportive and helpful despite living thousands miles away. My father always encourages me to do what I want to do, and my mother believes I am the best in the world, which gave me a lot of confidence. I am especially grateful to Yaling, for her endless love, support and encouragements.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

CHAPTER I

INTRODUCTION

1.1 Biological Background

1.1.1 Central Dogma of Molecular Biology

The Central Dogma of genetics is: DNA is transcribed to RNA which is translated to protein. Protein is never back-translated to RNA or DNA, and DNA is never directly translated to protein. RNA is a temporary intermediary between DNA and the protein making factories, the ribosomes. RNA could be compared to information stored in a cache in that the lifetime of RNA is much shorter than that of either DNA or the average protein, and also RNA serves to carry information from the genome, located in the nucleus of the cell, to the ribosomes, which are located outside of the nucleus either in the cytosol or on the endoplasmic reticulum (which is a large set of folded membranes proximal to the nucleus that help manufacture proteins for extra-cellular export). Proteins are the physical representation of the abstract information contained within the genome. Proteins vary greatly in their activity and half-life. Trying to classify proteins is like classifying programs; they come in all shapes and sizes. Figure 1.1 shows the central dogma of molecular biology.

The relationships of DNA, RNA, and Proteins are as following: DNA is long-term storage and it is stable, packaged, and inert; RNA is short-term storage. It is unstable and lacks secondary structure. Some RNA has enzymatic activity; Proteins are the programs of the cells. They are the physical manifestations of the abstract information recorded in the genome.

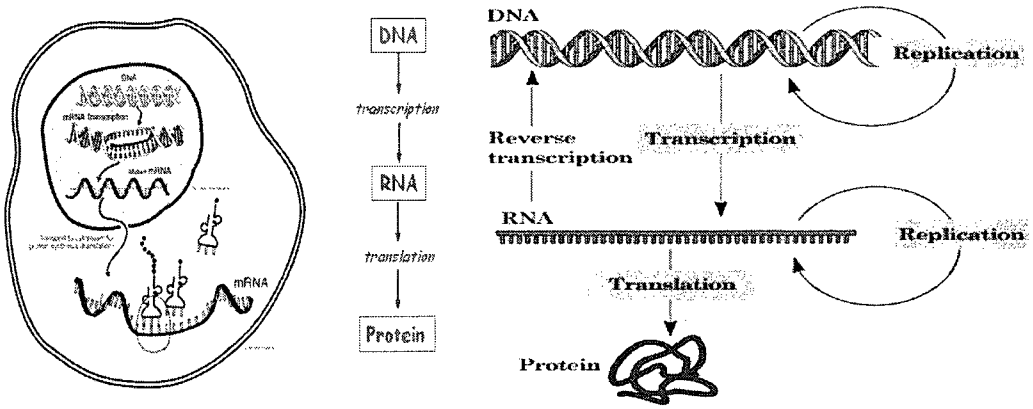


Figure 1.1 Central dogma of molecular biology

1.1.2 Genes

The genetic information carried by an organism is primarily inscribed in deoxyribonucleic acid (DNA). DNA is helix-shaped molecule whose constituents are two parallel strands of nucleotides. There are four types of nucleotides in DNA denoted by letters A (for adenine), T (thymine), C (cytosine) and G (guanine). The two strands of DNA are reversing complementary, which means that the second strand is always derivable from the first by pairing As with Ts and Cs with Gs and vice versa. Some contiguous pieces of DNA strand have been associated with certain functions in the living organism; we name them "genes". Genes are basic unit of heredity in a living organism and working subunits of DNA [1]. In cells, a gene is a portion of DNA that contains both "coding" sequences which determine what the gene does, and "non-coding" sequences that determine when the gene is active (expressed). Figure 1.2 shows the coding region in a segment of eukaryotic. When a gene is active, the coding and non-coding sequences are copied in a process called transcription, producing an RNA copy of the gene's information. This piece of RNA can then direct the synthesis of proteins via

the genetic code. In other cases, the RNA is used directly, for example as part of the ribosome. The molecules resulting from gene expression, whether RNA or protein, are known as gene products, and are responsible for the development and functioning of all living things.

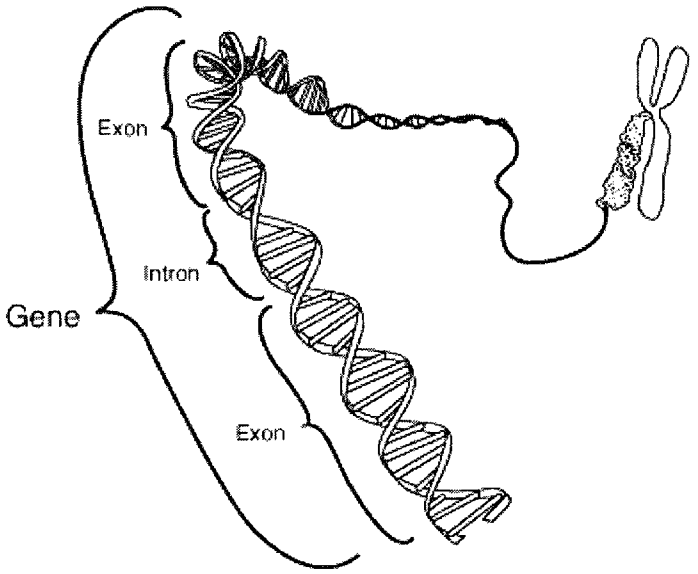


Figure 1.2 The coding region in a segment of eukaryotic [2]

DNA is a vast chemical information database that carries the complete set of instructions for making all the proteins a cell will ever need. Each gene contains a particular set of instructions, usually coding for a particular protein. DNA exists as two long, paired strands spiraled into the famous double helix. Each strand is made up of millions of chemical building blocks called bases. While there are only four different chemical bases in DNA (adenine, thymine, cytosine, and guanine), the order in which the bases occur determines the information available, much as specific letters of the alphabet combine to form words and sentences.

1.1.3 Gene Expression

The process by which information from a gene's DNA sequence is used in the synthesis of functional gene products is usually called gene expression. Here, the products are often proteins or functional RNA. Protein consists of a linear sequence of amino acids, and the type of each amino acid is defined by three consecutive bases on DNA which is known as codon. For example, three bases "AUG" comprise a codon which would code for the amino acid methionine. Theoretically, there are 64 types of amino acid, while in reality some combinations of three bases point to the same amino acid, leaving the number of amino acid types to be 20. Protein is considered the most basic building block of life. Its roles include constituting cell structures, regulating cellular processes, catalyzing biochemical reactions in metabolic pathways, and many other functions. A protein's functions are determined by its particular physical structure and chemical properties. These properties in turn are determined by the particular sequence of 20 possible biologically-active amino acids as well as the exact manner the amino acid chain is folded into a three-dimensional structure. The existence of life is made possible by thousands of different proteins acting at the right times and right places in a cell.

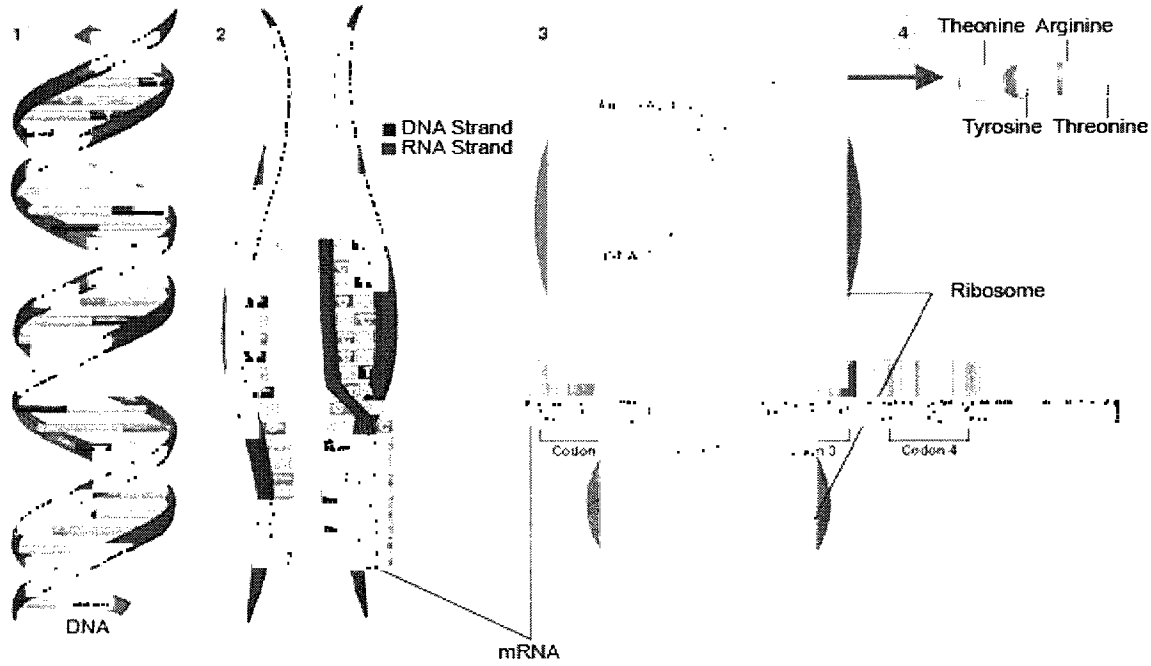


Figure 1.3 Transcription and translation process (source from http://publications.nigms.nih.gov/thenewgenetics/chapter1.html)

In Figure 1.3, it shows that there are two major steps in gene expression. The first step is transcription of DNA, where mRNAs (messenger RNA) are made based on the information of the gene sequence and the mRNA sequence (dark red strand) is complementary to the DNA sequence (blue strand). The RNA nucleotides are complementary to those on the DNA: a C on the RNA strand matches a G on the DNA strand. The second step is translation, which occurs after the transcription of DNA to mRNA. The translation of mRNA into protein depends on adaptor molecules that recognize both an amino acid and a triplet of nucleotides, where transfer RNA (tRNA) helps convert mRNA into protein and Amino acids link up to make a protein. These adaptors consist of a set of small RNA molecules known as tRNA, each about 80 nucleotides in length. The ribosome is a complex of more than 50 different proteins associated with several structural rRNA molecules. rRNA is a machinery for synthesizing

proteins by translating mRNA. Each ribosome is a large protein synthesizing machine, on which tRNA molecules position themselves for reading the genetic message encoded in an mRNA molecule.

The gene expression processes also depend on other factors, which include chromosomal activation or deactivation, control of transcription initiation, processing of RNA, RNA transport, mRNA degradation, initiation of translation and post-translational modifications.

1.1.4 Regulation of Gene Expression

Of the 35,000 genes in the human genome, only a fraction is expressed in a cell at any given time. Some gene products are present in very large amounts. Other gene products occur in much smaller amounts; for instance, a cell may contain only a few molecules of the enzymes that repair rare DNA lesions. Requirements for some gene products change over time. The need for enzymes in certain metabolic pathways may wax and wane as food sources change or are depleted. Given the high cost of protein synthesis, regulation of gene expression is essential to making optimal use of available energy. The cellular concentration of a protein is determined by a delicate balance of at least seven processes, each having several potential points of regulation.

A gene regulation system consists of genes, cis-elements, and regulators [3]. Figure 1.4 illustrates the regulatory process of genes. In most cases the regulators are proteins, but sometimes they also can be small molecules, such as RNAs and metabolites. The proteins that participate in regulatory system are usually called transcription factors (TFs), and sometimes they are also referred to as trans-regulatory elements. Cis-regulatory elements (or cis-elments in simple form), the complementary to trans-regulatory elements, are the

DNA segments in the same strand of genes that control the expression of correspondent genes. The regulatory mechanism involves the binding of certain TFs to cis-elements in the cis-region of genes, and consequently controls the level of target gene's expression during transcription. A gene might be two-faced: its expression is regulated by some regulators, while its own expressed products can be regulators for other genes. The complex regulatory connections, together with an interpretation scheme form gene regulatory network (GRN).

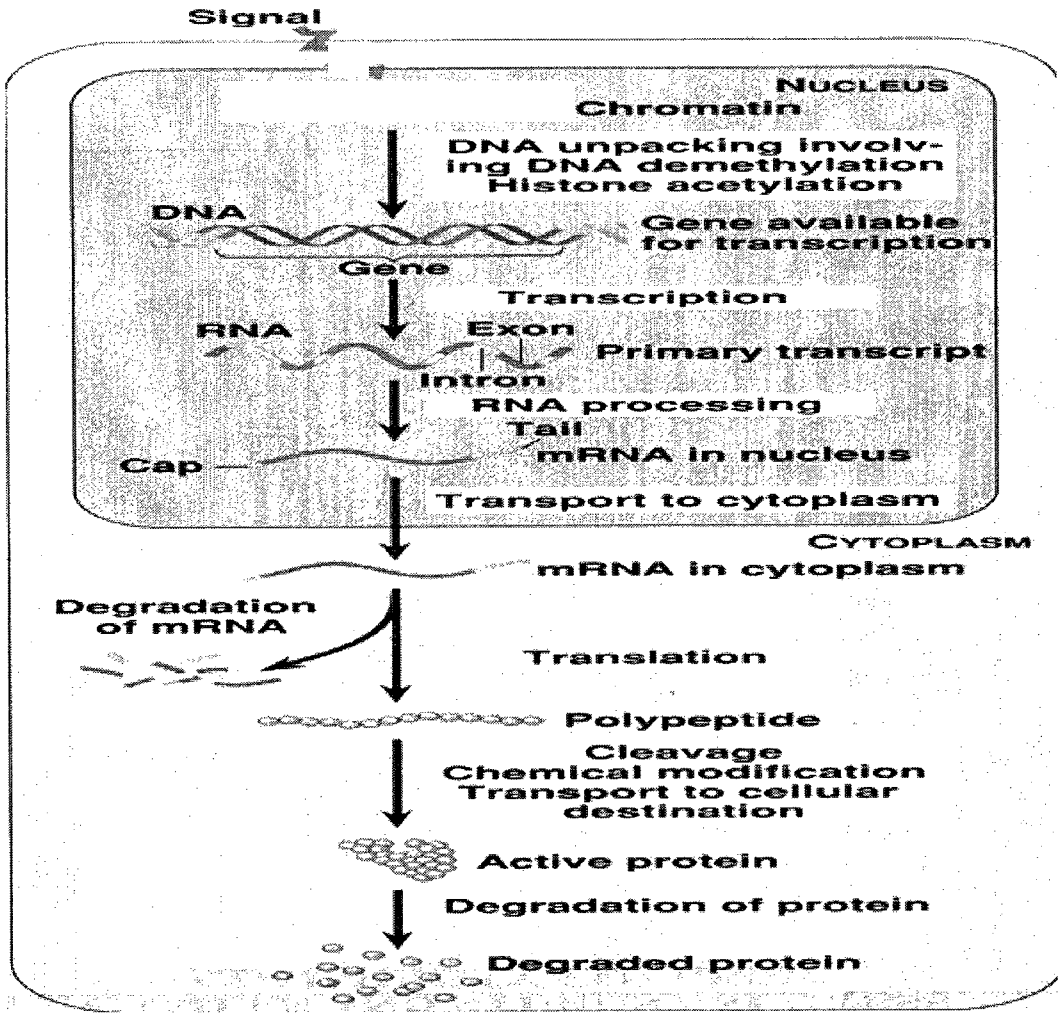


Figure 1.4 Regulations of genes

1.2 Computational Analysis of Biological Networks

Over the past decade, research in molecular and cell biology has increasingly looked beyond reductionism and towards and integrated understanding of molecular and cellular systems. This is in part due to the virtuous cycle in which new technologies enable faster, cheaper, higher-resolution, and more comprehensive measurements. In the last 20 years technologies improvement in genomics, transcriptomics and proteomics have generated a huge amount data, enabling the possibility of a genome-level analysis. These advances are bringing systems biology to the main stream of biological sciences in this century. Since the gathered data sets are often too large to be analyzed manually, computational methods are highly needed first to preprocess the raw data, and then to extract meaning and insight from the data. The tasks required for the understanding of biological networks, as summarized as following:

- Data analysis (Preprocessing raw data)
- System structure identification (Analysis of network topology),
- System behavior analysis (Analysis of network dynamics),

Mathematical modeling and approaches are natural tools for the performance of these key tasks. One of the values of mathematical models is that their formal study allows us to investigate generic properties and test hypotheses while the modeling process itself gives insight about the functional activity of the system and the dynamic interactions of its components.

1.2.1 Reverse Engineering Problem

Reverse engineering is the process of elucidating the structure of a system by reasoning backwards from observations of its behavior. In reverse engineering biological

networks, one of the first problems to overcome is semantic. The term "network" has come to mean different things throughout biology, and the semantic overload is magnified when computational and statistical interpretations are added. Even in networks whose nodes are the same objects (for example, genes and their protein products), the network edges can mean vastly different things and should be interpreted with care.

In the context of molecular biology, the reverse-engineering of biochemical networks from experimental data has become a central focus in systems biology. Two different types of analysis are of interest when developing a reverse engineering method as a modeling framework:

- Analysis of network topology. In this type of analysis, a network of molecular interactions is viewed as a directed graph: a pair (V,E) where V is a set of vertices (or nodes) and E a set of directed edges, i.e. pairs (i, j) of nodes, where i is the source node and j is the target node. In some instances, undirected graphs are used instead, for example, when only describing existence of a correlation between two nodes rather than a causal direction. This directed graph is also known as "wiring diagram". Some of the top-down methods are statistical in nature, allowing identification of the network's structure or wiring diagram. Depending on the method used, the edges represent either a statistical correlation of two variables
- Analyses of network dynamics. The phase space of the network is analyzed for a description of the dynamic rules that describe how the system evolves in time or changing conditions. Dynamical properties of interest include the identification of steady states or limit cycles, identification of multi-stable (e.g. switch-like) behavior, environmental changes, genetic perturbation, etc. Top-down modeling

methods to discover a network's behaviors commonly use dynamical systems modeling frameworks.

1.2.2 Gene Regulatory Network (GRN)

Gene regulatory networks are pathways of genes whose induced proteins regulate the expression of other genes and their products. They orchestrate biochemical processes that specify spatial and temporal patterns or govern the formation of tissues and organs. GRN reveals the causality of these processes through activation or repression of targets by regulatory proteins. With current technologies, activity levels of each biomolecule in a network can be measured directly, while causal linkages remain unobservable. A challenge for molecular biologists is to identify the causal links that constitute the pathways in a GRN. Identification and ultimately control of these signaling pathways are important first steps in repairing developmental defects, including those that cause tissue-specific cancers.

Mathematical and statistical tools have been employed to *reverse engineer*, or reconstruct, the pathways in a GRN from microarray data. The data, which record expression levels of genes, are typically limited to tens of measurements, while the number n of biomolecules in a GRN is in the hundreds to thousands. A standard reverse engineering approach can be described as follows:

- Choose a modeling class, such as Boolean networks or linear differential equations.
- Use biological properties to constrain the class, for example, by limiting the number of links per biomolecule.

- Construct a model defined by a set of n functions that fit the data. The pathway structure can then extracted from the model.

In Chapter 2, we will discuss more details about how the gene regulatory network works and how to infer from given biological data, such as microarray gene expression data.

1.3 Contributions

In this dissertation, we have made a number of contributions in comparing different computational models and modeling gene regulatory networks, which we summarize below.

1.3.1 Comparison of Inference Models in GRN

Previously, lots of computational approaches have been proposed to model gene regulatory networks. Among them probabilistic Boolean network (PBN) [63, 64, 66, 67, 68] and dynamic Bayesian network (DBN) [34, 37, 95, 96, 97] are two very popular and powerful methods to model gene regulatory networks. In our work, probabilistic Boolean network and dynamic Bayesian network were compared using a biological time series dataset from Drosophila Interaction Database to construct a Drosophila gene network. We used a subset of time points and gene samples from the whole dataset to evaluate the performance of these two approaches. We also compared the performance of dynamic Bayesian network and Bayesian network (R package) using the yeast cell cycle data sets. More details will be discussed in Chapter 3.

1.3.2 Improvement of Dynamic Bayesian Network (DBN)

Due to the limitations of real biological data, the simulated data from *in silico* gene networks provides a feasible means to systematically evaluate the performances of different genetic networks inferring algorithms. In this work, relative change ratios and

dynamic Bayesian network are combined together to infer gene regulatory networks from synthetic data sets, which are provided by the Laboratory of Intelligent Systems of the Swiss Federal Institute of Technology in Lausanne and used by DREAM [123] challenge. The data we used in this work consists of different size of datasets that were produced from *in silico* networks, including E.Coli and Yeast. The given data sets are composed of three different data sets: Gene knock-out data, gene knock-down data and time series trajectories data. First we use relative change ratios to analyze the gene knock-out and knock-down data and select potential regulators regarding to target genes, and then dynamic Bayesian network is used to infer gene regulatory network from time series gene expression data. Then we combine the two results together to get the final gene regulatory networks.

1.3.3 Proposed a New Model to Infer GRN

One major problem of existing computational algorithms is the fact that the number of time points is much smaller than that of genes. The traditional time series analysis such as the autoregressive model will fail due to the over-learning problem, because the degree of freedom of the parameters is redundant. If we want to reconstruct a large gene regulatory network which may include more than 1000 genes, it will be very difficult with due to its computational time complexity. To overcome such difficulty, a model based on state space model is proposed to establish gene regulatory networks. In this dissertation, we derive gene interaction matrix based on state space model and EM algorithms for inferring gene regulatory networks and implement these algorithms in MATLAB. Then two synthetic data sets are used to test our new model before applying to real biological data sets.

1.4 Dissertation Organization

This dissertation is organized as follows: In Chapter 2, we introduce some basic concepts and backgrounds of microarray experiments and gene regulatory networks. Then some data preprocessing analysis methods based on microarray data are introduced, such as image processing analysis, normalization, etc. We also discuss some hierarchical and non-hierarchical clustering methods based on microarray data.

In Chapter 3, we discuss computational approaches and algorithms to infer gene regulatory networks. We present a review of existing inferring algorithms such as Boolean networks, Bayesian networks, and further discuss dynamic Bayesian network. We also address their shortcomings and possible improvements

In Chapter 4, a new model is proposed based on the state space model to represent gene regulatory networks, and then expectation-maximization (EM) algorithms are used to estimate the parameters in given observation and measurement functions. In the process of learning parameters Kaman filter and Kalman smoother are needed to calculate the conventional Kalman smoothing estimators. And then, gene interaction matrix is derived from the learned parameters by EM.

In Chapter 5, some results from our previous work, which are based on probabilistic Boolean networks and dynamic Bayesian network, are given and discussed. We use four different gene expression data sets, in which one is synthetic data set and the other three are real biological data sets. In Chapter 6, we present the applications of our proposed new model, and then show how to validate our new model using two synthetic data sets before applying to real biological data sets. Some results based on model validation are also discussed in this chapter.

We complete the dissertation by summarizing and concluding our work, and providing a set of issues appropriate for future work in Chapter 7.

CHAPTER II

GENE REGULATORY NETWORKS

The recent development of high-throughput genomic technologies, such as protein array, Chip-Chip, DNA microarray and Chip-Seq, makes it possible for us to measure dependencies and regulation among genes on a genome-wide scale simultaneously and provides us a huge amount of data and genetic information. Thus focus of biological scientists shifted from obtaining gene sequence data to identifying gene functions and extracting useful information. The traditional methodology which separately studies basic units of information (or genes) is not sufficient since it is widely believed that biological systems are complex, where thousands of genes and their products interact in concert to enable life. Contrary to the traditional approaches, recently emerged functional genomics develops the genome-wide or system-wide experiments in hope of obtaining a global view of such a complex biological system. An important objective of the functional genomics is to understand the controlling mechanism of the expression of these genes as well as the consequent synthesis of proteins, and ultimately encode the knowledge learned from the experiments into the format of a graph, namely gene regulatory network (GRN). To achieve this, computational and statistical tools are especially needed.

2.1 Microarray Experiments and Microarray Data Analysis

Microarray technology evolved from Southern blotting, where fragmented DNA is attached to a substrate and then probed with a known gene or fragment. The use of a collection of distinct DNAs in arrays for expression profiling was first described in 1987, and the arrayed DNAs were used to identify genes whose expression is modulated by interferon [4]. These early gene arrays were made by spotting cDNAs onto filter paper

with a pin-spotting device. The use of miniaturized microarrays for gene expression profiling was first reported in 1995 [5], and a complete eukaryotic genome on a microarray was published in 1997 [6]. The high-throughput technologies, such as DNA microarray, are highly needed for measuring the mRNA gene expression values simultaneously and these generated gene expression data are very crucial to classify the diseases and discover the gene expression patterns [7-9].

2.1.1 Microarray Experiment

A microarray experiment requires a large array of cDNA or oligonucleotide DNA sequences (probes) that are fixed on a glass, nylon, or quartz wafer (adopted from the semiconductor industry and used by Affymetrix, Inc. [10]). This array is then mixed with material containing RNA which is obtained from the biological samples to be studied, for example, the mixture of normal tissues and cancer tissues. There are a lot of types of microarray technologies and the DNA microarray is the most popular one so far, so we choose DNA microarray to describe the microarray experiments design. Figure 2.1 shows the basic process of cDNA microarray experiments. For cDNA microarrays, two samples are needed, one for real test, another one is called reference sample (normal) which is used to be compared with the test sample. These two samples are labeled with different fluorochromes [11] for the purpose of obtaining gene expression level, then the test sample is labeled with the flurochrome Cy3, and the control sample is labeled with a different flurochrome Cy5. Because of the sequence similarity and complementariness the probes are likely to hybridize with their correspondent target but not the other RNA molecules. The amount of the hybridization products obtained is an indicator of how much RNA is being expressed by each one of the being studied. After the hybridization

of the probes, the scanner has a laser (or lasers) producing light with a wavelength appropriate for the excitation spectra of the dyes being used (for Cy3 this is green light around 540 nm; for Cy5 red light around 650 nm). The light passes through a standard microscope objective and illuminates a single point on the slide. The emitted light gathered by the objective passes through a series of filters (to remove the excitation beam), a collimating lens, and a pinhole (to minimize noise; this makes it a confocal device) and is quantified in a photomultiplier tube. The slide is rapidly scanned over the laser beam and a raster image of the array is acquired. The intensity of the fluorescent signal at each spot is taken as a measure of the levels of the mRNA associated with the specific sequence at that spot. Figure 2.1 shows the basic process of microarray experiment.

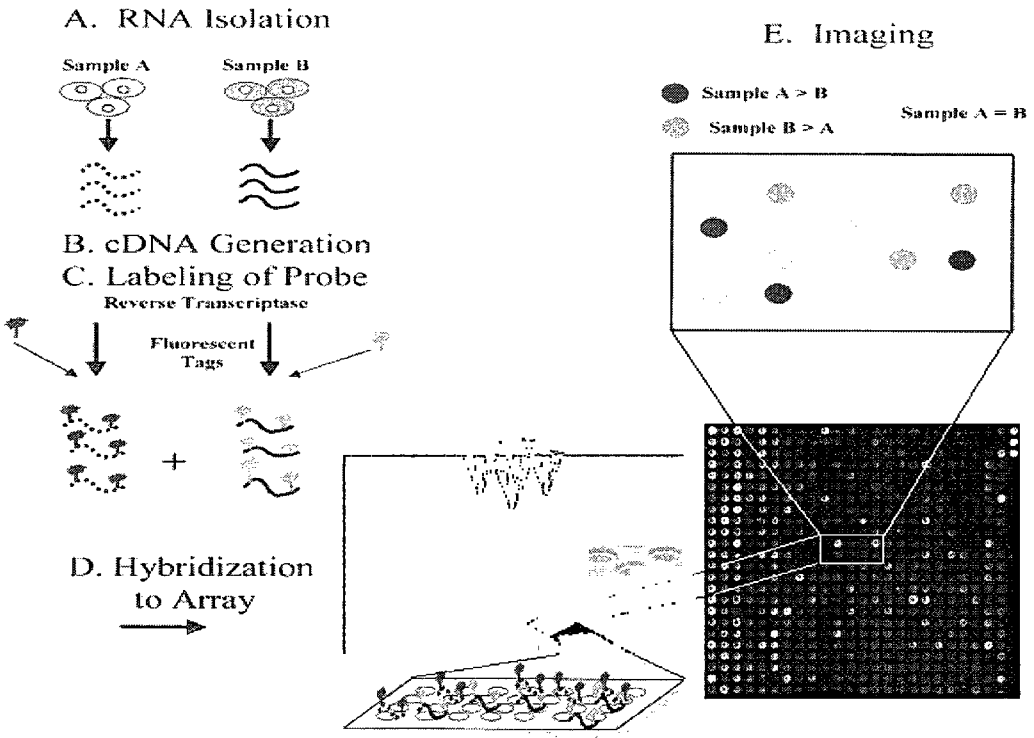


Figure 2.1 Process of cDNA microarray experiment design (source from http://www.fao.org/docrep/003/x6884e/x6884e00.jpg)

In cDNA microarrays, two independent images are generated for both samples (one only detects Cy3 and one only detects Cy5), the ratios of the measurement of the two samples is used in the subsequent data analysis [12], for example, to identify differentially expressed genes. The image of all the spots is analyzed using sophisticated software linked with information about the sequence of the DNA at each spot. This generates a general profile of gene expression level for the selected experimental and control conditions.

2.1.2 Image Processing Analysis

The relative expression level for each gene (population of RNA in the two samples) can be stored as an image. The first step in the analysis of microarray data is to process this image. Most manufacturers of microarray scanners provide their own software; however, it is important to understand how data is actually being extracted from images, as this represents the primary data collection step and forms the basis of any further analysis.

Image processing involves the following steps: 1. Identification of the spots and distinguishing them from spurious signals. The microarray is scanned following hybridization and a TIFF image file is normally generated. Once image generation is completed, the image is analyzed to identify spots. This information is then used to identify regions that correspond to spots; 2. Determination of the spot area will be surveyed and determination of the local region is used to estimate background hybridization. After identifying regions that correspond to sub-arrays, an area within the sub-array must be selected to get a measure of the spot signal and an estimate for background intensity; 3. Reporting summary statistics and assigning spot intensity after

subtracting for background intensity. In this step, once the spot and background areas have been defined, a variety of summary statistics for each spot in each channel (red and green channels) are reported.

Another consideration in image processing is the number of pixels to be included for measurement in the spot image [13]. For many scanners, the default pixel size is 10μm. This means that an average spot of diameter of 200μm will have ~314 pixels. However, for a smaller spot diameter, it is better to use a smaller pixel size to ensure enough pixels are sampled. Most scanners now allow the pixel size of 5μm. Even though using a smaller pixel size increases our confidence in the measurement, the only disadvantage is that the image file size tends to be much bigger when compared with image file sizes created using larger pixel sizes.

2.1.3 Data Normalization

Normalization is a term that is used to describe the process of eliminating such variations to allow appropriate comparison of data obtained from the two samples. The first step in a normalization procedure is to choose a gene-set (which consists of genes for which expression levels should not change under the conditions studied, that is the expression ratio for all genes in the gene-set is expected to be 1. From that set, a normalization factor, which is a number that accounts for the variability seen in the gene set, is calculated. It is then applied to the other genes in the microarray experiment. One should note that the normalization procedure changes the data, and is carried out only on the background corrected values for each spot.

There are many approaches to normalizing expression levels. Some, such as total intensity normalization, are based on simple assumptions. Given that there are millions of individual RNA molecules in each sample, we will assume that the average mass of each

molecule is approximately the same, and that, consequently, the number of molecules in each sample is also the same. If the arrayed genes are selected to represent only those we know will change, then we will likely over- or under-sample the genes in one of the biological samples being compared. If the array contains a large enough assortment of random genes, we do not expect to see such bias. This is because for a finite RNA sample, when representation of one RNA species increases, representation of other species must decrease. Consequently, approximately the same number of labeled molecules from each sample should hybridize to the arrays and, therefore, the total hybridization intensities summed over all elements in the arrays should be the same for each sample. Using this approach, a normalization factor is calculated by summing the measured intensities in both channels

$$N_{total} = \frac{\sum_{i=1}^{Narray} R_i}{\sum_{i=1}^{Narray} G_i},$$

where G_i and R_i are the measured intensities for the ith array element (for example, the green and red intensities in a two-color microarray assay) and *Narray* is the total number of elements represented in the microarray. Figure 2.2 shows an R-I plot displays the log2 (*Ri/Gi*) ratio for each element on the array as a function of the log10 (*Ri*Gi*) product intensities.

There are a number of alternative approaches to normalizing expression ratios, including linear regression analysis [14], log centering, rank invariant methods [15] and Chen's ratio statistics [16], among others. However, none of these approaches takes into account systematic biases that may appear in the data. Several reports have indicated that

the log2 (ratio) values can have a systematic dependence on intensity [17, 18], which most commonly appears as a deviation from zero for low-intensity spots. Locally weighted linear regression (lowess) [19] analysis has been proposed [17, 18] as a normalization method that can remove such intensity-dependent effects in the log2 (ratio) values.

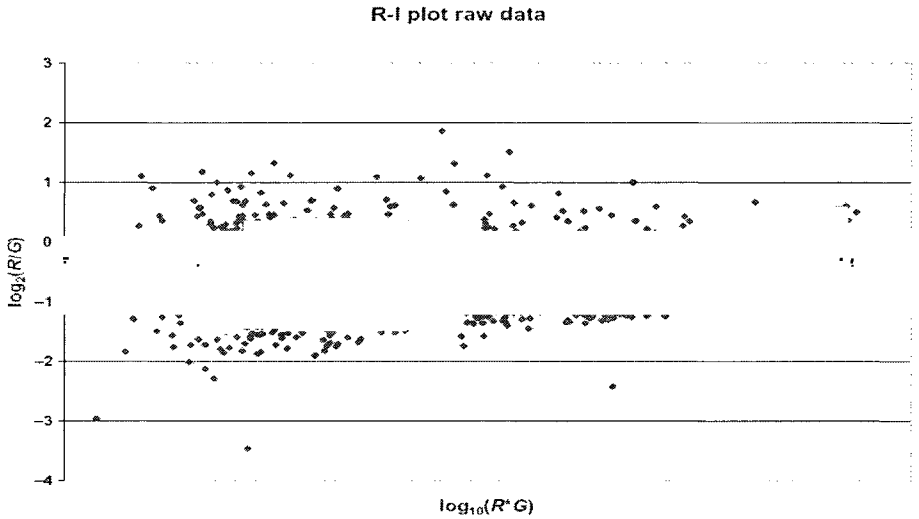


Figure 2.2 R-I plot of log2 ratio as a function of the log10 product intensities

2.2 Identification of Differentially Expressed Genes

After the experiment performed, our interest is the identification of genes that are differentially expressed between one or more pairs of samples in the data set. After normalizing, filtering and averaging the data, the genes with expression ratio which significantly different from 1, will be identified as important genes which we might be interested in later cluster analysis or gene regulatory network reconstruction.

Even if data mining analysis is done using some clustering methods [20-22], it is still extremely useful to reduce the data set to those genes that are most variable between samples. There are many methods to identify the genes exhibiting the most significant

variation, such as a fixed fold-change cut-off (usually two folds) method or a more complicated approach which use a so-called "*Z*-score" to calculate the mean and standard deviation of the distribution of log2(ratio) values and defining a global fold-change difference and confidence. By this way, differentially expressed genes at the 95% confidence level will have a value of $Z > 1.96$ [23]. Figure 2.3 shows an example of a *Z*-score selection application.

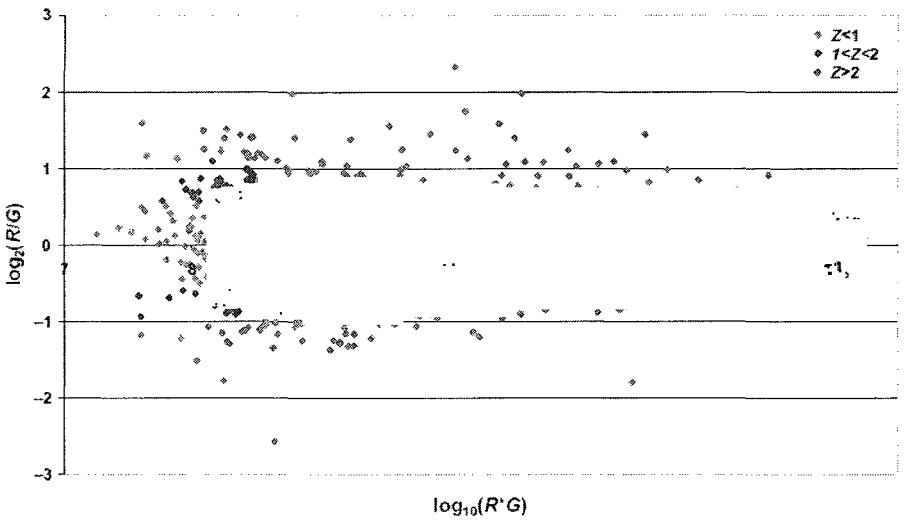


Figure 2.3 Intensity-dependent *Z* -scores for identifying differential expression

2.3 Clustering Methods

One of the goals of microarray data analysis is to cluster genes or samples with similar expression profiles together, to make meaningful biological inference about the set of genes or samples. Clustering is one of the unsupervised approaches to classify data into groups of genes or samples with similar patterns that are characteristic to the group. Clustering methods can be hierarchical (grouping objects into clusters and specifying relationships among objects in a cluster, resembling a phylogenetic tree) or non-hierarchical (grouping into clusters without specifying relationships between objects in a

cluster). An object may refer to a gene or a sample, and a cluster refers to a set of objects that behave in a similar manner.

2.3.1 Hierarchical Clustering

Hierarchical clustering may be agglomerative (starting with the assumption that each object is a cluster and grouping similar objects into bigger clusters) or divisive (starting from grouping all objects into one cluster and subsequently breaking the big cluster into smaller clusters with similar properties). The basic idea behind agglomerative and divisive hierarchical clustering is shown in Figure 2.4. There are many different types of clustering methods and a few commonly used ones are described below [24].

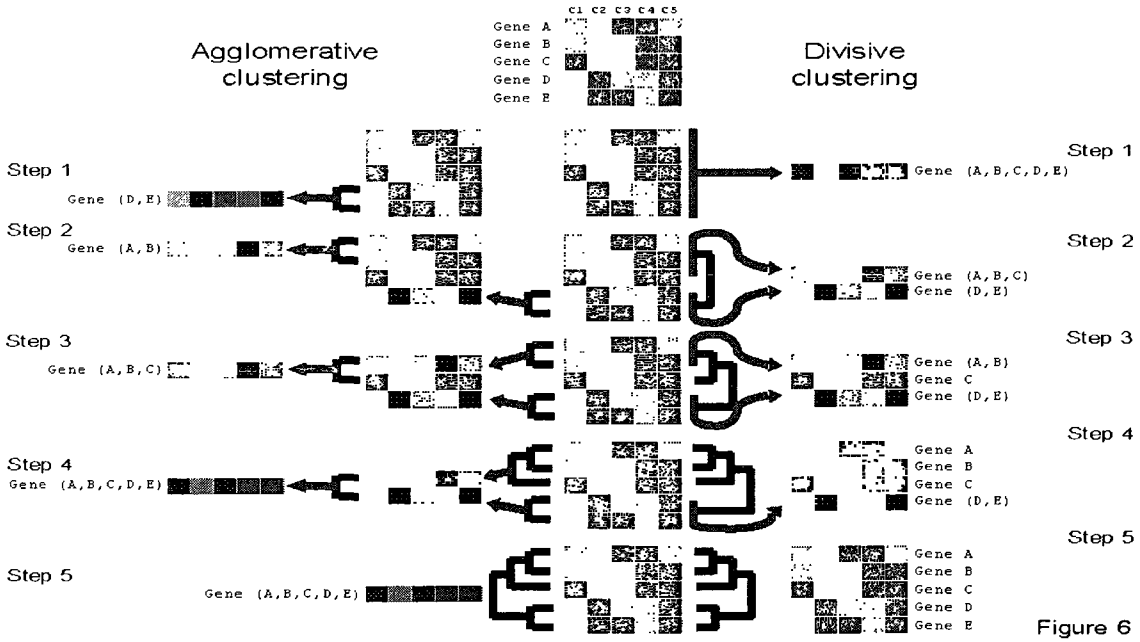


Figure 2.4 Agglomerative clustering and divisive clustering (source from http://www.mrc-lmb.cam.ac.uk/genomes/madanm/microarray)

For a hierarchical agglomerative clustering procedure, each object is considered as a cluster. The first step is the calculation of pairwise distance measures for the objects to be clustered. Based on the pairwise distances between them, objects that are similar to each

other are grouped into clusters. After this is done, pairwise distances between the clusters are re-calculated, and clusters that are similar are grouped together in an iterative manner until all the objects are included into a single cluster. Comparison of clusters with another cluster or an object can be carried out using four approaches: Single linkage clustering (Minimum distance), complete linkage clustering (Maximum distance), average linkage clustering and Centroid linkage clustering.

For a hierarchical divisive clustering procedure, it is the opposite of the agglomerative method, where the entire set of objects is considered as a single cluster and is broken down into two or more clusters that have similar expression profiles. After this is done, each cluster is considered separately and the divisive process is repeated iteratively until all objects have been separated into single objects. The division of objects into clusters on each iterative step may be decided upon by principal component analysis which determines a vector that separates given objects. This method is less popular than agglomerative clustering, but has successfully been used in the analysis of gene expression data by [25].

2.3.2 Non-hierarchical Clustering

One of the advantages of hierarchical clustering is that there is no compelling evidence that a hierarchical structure best suits grouping of the expression profiles. An alternative to this method is a non-hierarchical clustering, which requires predetermination of the number of clusters. Non-hierarchical clustering then groups existing objects into these predefined clusters rather than organizing them into a hierarchical structure.

K-means is a popular non-hierarchical clustering method (Figure 2.4A). In K-means clustering, the first step is to arbitrarily group objects into a predetermined number of clusters. The number of clusters can be chosen randomly or estimated by first performing a hierarchical clustering of the data. Following this step, an average expression profile (centroid) is calculated for each cluster, this is called initialization. Next, individual objects are reattributed from one cluster to the other depending on which centroid is closer to the gene (or sample). This procedure of calculating the centroid for each cluster and re-grouping objects closer to available centroids is performed in an iterative manner for a fixed number of times, or until convergence (state when composition of clusters remains unaltered by further iterations). Typically, the number of iterations required to obtain stable clusters ranges from 20,000 to 100,000. However, there is no guarantee that the clusters will converge. This method has an advantage that it is scalable for large datasets [24].

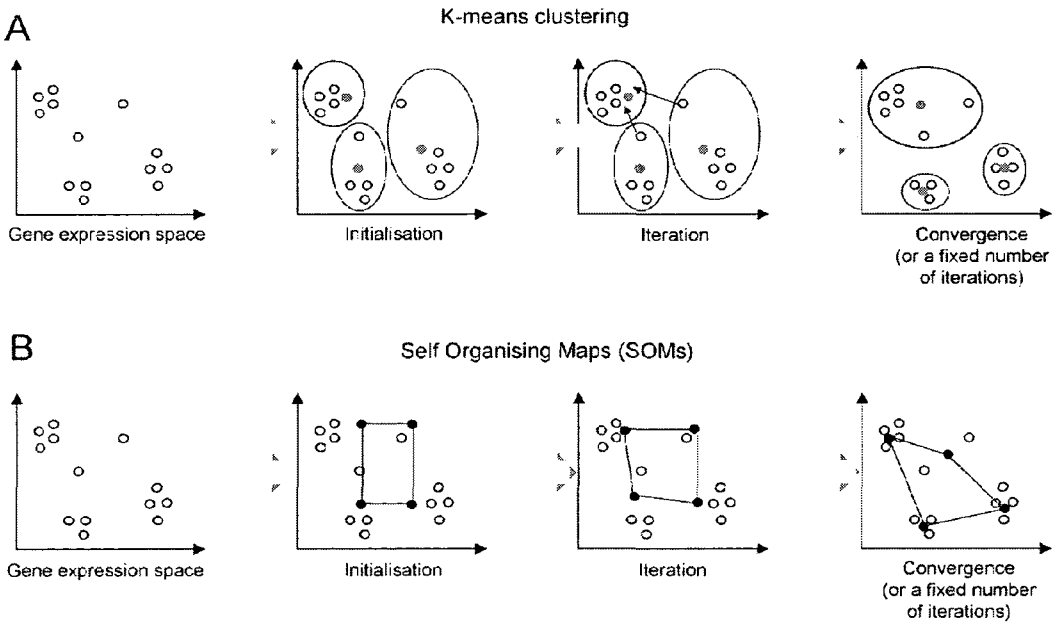


Figure 2.5 The principle behind K-means and self organizing maps (SOMs) [24]

Self organizing maps (SOMs) [26, 27] work in a manner similar to K-means clustering (Figure 2.5). In K-means clustering, one chooses the number of clusters to fit the data, whereas with SOM the first step is to choose the number and orientation of the clusters with respect to each other. For example, a two-dimensional grid of nodes (which may end up being clusters) could be the starting point. The grid is projected onto the expression space, and each object is assigned a node that is nearest to it – this is called initialization. In the next step, a random object is chosen and the node (called a reference vector) which is in the neighborhood of the object is moved closer to it. The other nodes are moved to a small extent depending on how close they are to the object chosen. In successive iterations, with randomly chosen objects, the positions of the nodes are refined and the radius of neighborhood becomes confined. In this way, the grid of nodes (initially a two-dimensional grid) is deformed to fit the data. The advantage of this method, unlike K-means, is that SOM does not force the number of clusters to be equal to the number of starting nodes in the chosen grid. This is because some nodes may have no objects associated with them when the map is complete.

Other advantages of SOM include providing information on the similarity between the nodes, and the ability of SOM to produce reliable results even with noisy data.

2.4 Inference of Gene Regulatory Network (GRN)

Besides clustering, gene expression data can also be used to infer regulatory relationships, which is also known as problem of reverse engineering. This approach is known as reverse engineering of regulatory networks. Research by [28] and [29] clearly states that we are able to use expression data to make predictions about the transcriptional regulators for a given gene or sets of genes.

A gene regulatory network (also called a *GRN* or genetic regulatory network) is a collection of DNA segments in a cell which interact with each other (indirectly through their RNA and protein expression products) and with other substances in the cell, thereby governing the rates at which genes in the network are transcribed into mRNA. In general, each mRNA molecule will be translated to a specific protein or a set of proteins. In some cases this protein will be structural, and will accumulate at the cell-wall or within the cell to give it particular structural properties. In other cases the protein will be an enzyme; a micro-machine that catalyses a certain reaction, such as the breakdown of a food source or toxin. Some proteins only activate other genes, and these are the transcription factors which play very important roles in regulatory networks. By binding to the promoter region at the start of other genes they turn them on, initiating the production of another protein, and so on. Some transcription factors are inhibitory.

In a gene regulatory network, the nodes of this network are proteins, their corresponding mRNAs, and protein/protein complexes. The edges between nodes are individual molecular reactions, the protein/protein and protein/mRNA interactions, through which the products of one gene affect those of another. A series of edges indicates a chain of such dependences, with cycles corresponding to feedback loops. The structure of gene regulatory network is an abstraction of the system's chemical dynamics, describing the ways in which one substance affects all the others to which it is connected. In practice, such gene regulatory networks are inferred from the biological literature on a given system and represent a distillation of the collective knowledge about a set of related biochemical reactions.

GRN can be viewed as a cellular input-output device. A simple GRN would consist of one or more input signaling pathways, regulatory proteins that integrate the input signals, several target genes, and the RNA and proteins produced from those target genes. In addition, such networks often include dynamic feedback loops that provide for further regulation of network architecture and output. Figure 2.6 shows a typical gene regulatory network.

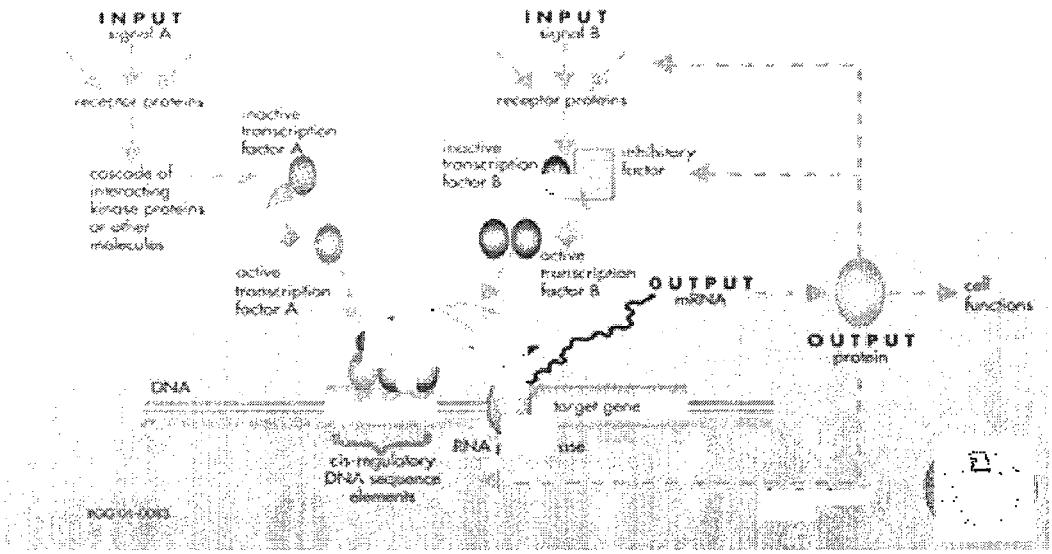


Figure 2.6 A typical gene regulatory network (source from http://biophysics.asu.edu/workshop/report/html/bio_mat_files/image063.jpg)

The gene regulatory networks are only beginning to be understood, and the next step for biologists is to attempt to deduce the functions for each gene, which will help us to understand the behavior of the system in increasing levels of complexity, from gene to signaling pathway, cell or tissue level.

In general, inference of gene regulatory networks is the process of analyzing a subject system to identify the components of the system and their relationships, as well as to create an abstract representation of the system that facilitates its study. Engineers and

scientists have previously developed techniques or computational algorithms in the fields of computer science, engineering, and statistics, which are respectively called machine learning, system identification, and statistical learning.

The inference or reconstruction of gene regulatory networks from experimental data (i.e., microarray data) has become a critical problem in systems biology or computational biology. Two different types of analysis are of interest when developing an inference method as a modeling such gene regulatory networks.

First one is analysis of network topology. In network topology analysis, a network of molecular interactions is considered as a directed graph: a pair (V, E) where V is a set of nodes (or genes) and E a set of directed or undirected edges (interactions), i.e. pairs (i, j) of nodes, where i is the source node and j is the target node. In some instances, undirected graphs are used instead of directed graphs, for example, when only describing existence of a correlation between two nodes rather than a causal direction. This directed graph is also known as "wiring diagram". Some of the top-down methods are statistical in nature, allowing identification of the network's structure or wiring diagram. Depending on the method used, the edges represent either a statistical correlation of two variables [30-31] or a causal relationship [32]. The most commonly-used modeling framework is that of dynamic Bayesian networks [33-38].

Second one is called analyses of network dynamics. The phase space of the network is analyzed for a description of the dynamic rules that describe how the system evolves in time or changing conditions. Dynamical properties of interest include the identification of steady states or limit cycles, identification of multi-stable behavior, environmental changes, genetic perturbation, etc. Top-down modeling methods to discover a network's

behavior commonly use a dynamical systems modeling framework, such as Boolean networks [39-42] systems of differential equations [29, 43, 44].

CHAPTER III

COMPUTATIONAL APPROACHES TO INFER GRN

Mathematical models of GRN have been developed to capture the behavior of the system being modeled, and in some cases generate predictions corresponding with experimental observations. In some other cases, models have proven to make accurate novel predictions, which can be tested experimentally, thus suggesting new approaches to explore in an experiment that sometimes wouldn't be considered in the design of the protocol of an experimental laboratory. Several promising modeling techniques have been used, including Boolean networks, Petri nets, Bayesian networks, graphical Gaussian models, Stochastic, and differential equations. Conversely, techniques have been proposed for generating models of GRN that best explain a set of time series observations.

Currently, clustering, classification and visualization methods are used for reconstruction or inference of gene regulatory networks from gene expression data sets. These methods generally group genes based on the similarity of expression patterns. Based on large-scale microarray data retrieved from biological experiments, many computational approaches have been proposed to reconstruct gene regulatory networks, such as information theory [46, 47, 49, 52, 55], Boolean networks [59, 60, 62, 64, 65, 68], differential equations [72, 73, 84, 85, 88], Bayesian networks [32, 34, 37, 38, 69, 95, 96] and neural networks [7]. Many computational methods have been proposed for modeling or simulating gene regulatory network.

3.1 Information Theory

Information theoretic approaches are increasingly being used for reconstructing regulatory networks from gene expression microarray data. It is one of the simplest

network architectures [45], which can be represented by an undirected graph with edges that are weighted by correlation coefficients. Thereby, two genes are predicted to interact if the correlation coefficient of their expression levels is above some set threshold. The higher the threshold is set, the sparser is the inferred gene regulatory networks.

Besides correlation coefficients, also Euclidean distances and information theoretic scores, such as the mutual information, were applied to detect gene regulatory dependencies [46]. The network inference algorithms RELNET (RELevance NETworks, [47]), ARACNE (Algorithm for the Reverse engineering of Accurate Cellular Networks, [48, 49]) and CLR (Context Likelihood of Relatedness, [50]) apply network schemes in which edges are weighted by statistic scores derived from the mutual information. In [51], proposed an asymmetric version of the mutual information measure to obtain directed networks. Likewise, graphical Gaussian models (GGMs) using partial correlations to detect conditionally dependent genes also allow us to distinguish direct from indirect associations [52].

Simplicity and low computational costs are the major advantages of information theory models. Because of their low data requirements, they are suitable to infer even large-scale networks. Thus, they can be used to study global properties of large-scale regulatory systems. In comparison to other formalisms, a drawback of such models is that they do not take into account that multiple genes can participate in the regulation. A further disadvantage is that they are static.

3.1.1 Information Theoretic Entropy

Entropy is a measure of uncertainty of a random variable. Information theory provides us with a quantitative information measure H, which is called the Shannon entropy. For

discrete random variables X, which could be either a vector or a scalar, the Shannon entropy $H(X)$ is defined in terms of the probability of observing a particular symbol or event within a given sequence (Shannon & Weaver [53]) as following:

$$H(X) = -\sum_{x \in \Omega_x} p(x) \cdot \log p(x) \qquad (3.1)$$

where $p(x)$ is the probability mass function, and Ω_x stands for the alphabet of X. The entropy of a discrete variable is always nonnegative. For a continuous-valued random variable X, the differential entropy $h(X)$ is defined as

$$h(X) = \int_{x \in Sx} f(x) \cdot \log f(x) dx \qquad (3.2)$$

where $f(x)$ denotes the probability density function, and Sx represents the support of X. The differential entropy is also denoted as $h(f)$ and can take negative values. Therefore, some discrete network inference algorithms, for example, REVEAL [39], cannot be deployed for continuous-valued gene expression data, unless the data are quantized, and the associated information loss is tolerated.

Just as with probabilities, we can compute joint and conditional entropies. Joint entropy is the randomness contained in two variables, while conditional entropy is a measure of the randomness of one variable given knowledge of another. Joint entropy is defined as:

$$H(X,Y) = -\sum_{x \in \Omega_x, y \in \Omega_y} p(x,y) \cdot \log p(x,y) \qquad (3.3)$$

while the conditional entropy is defined as:

$$H(Y \mid X) = -\sum_{x \in \Omega_x, y \in \Omega_y} p(x,y) \cdot \log p(y \mid x) \qquad (3.4)$$

There are several facts that follow from these definitions. For example, two random variables, X and Y, are considered independent if and only if $H(Y \mid X) = H(Y)$ or $H(X,Y) = H(X) + H(Y)$. The other fact is, the discrete entropy H, do not hold for continuous or differential entropy h.The most important is that while $H(X) \geq 0$, h can actually be negative. Luckily, for us, even though differential entropy cannot provide us with an absolute measure of randomness, it is still that case that if $h(X) \geq h(Y)$ then X has more randomness than Y.

3.1.2 Mutual Information

Although conditional entropy can tell us when two variables are completely independent, it is not an adequate measure of dependence [54]. A small value for $H(Y \mid X)$ may imply that X tells us a great deal about Y or that $H(Y)$ is small to begin with. Thus, we measure dependence using *mutual information*:

$I(X;Y) = H(Y) - H(Y \mid X)$

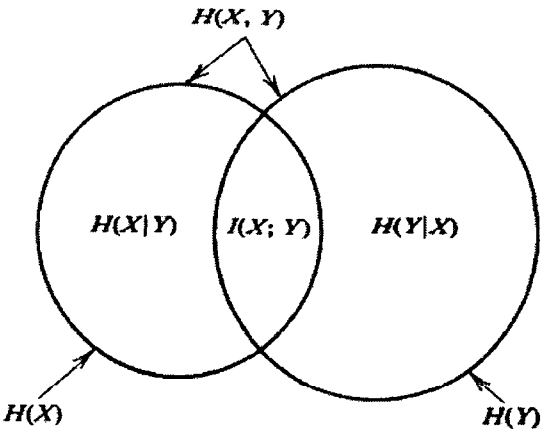


Figure 3.1 Relationship between entropy and mutual information [54]

The relationship between entropy and mutual information is described in Figure 3.1. Both discrete and continuous versions of $I(X;Y)$ are nonnegative and assume the value zero if and only if X and Y are independent. Continuous-valued random variables should be employed to describe the original DNA microarray data, whereas discrete-valued random variables are used to model quantized gene expression data. If gene X interacts with gene Y, in the steady state, it is hypothesized that the expression values of X and Y show a strong dependence. This is partially evidenced by the study of chemical kinetics. When the chemical reaction achieves the equilibrium, the concentrations of all participating complexes can be modeled by an equation, and they depend on each other. Therefore, if $I(X;Y)$ assumes a very small value, it can be reasonably inferred that X and Y are disconnected in the genetic regulatory network. However, the opposite statement does not hold. Given a large $I(X;Y)$, X and Y can be either directly connected or connected through an intermediate gene.

3.1.3 Inference of Information Theory Model

The inference algorithm of information theory model is proposed by [55]. In the first step, the continuous-valued expressions of each gene x are rank transformed. For instance, let $X_1, X_2, \cdots, X_m$ stand for m gene expression observations of X. If $X_i (i \in [1, m])$ is the k th smallest from the m values, then X_i is assigned the value k/m. Only the ranks of data are preserved; therefore, outliers with incredible great values are removed, and the negative preprocessing effects are reduced. Then, all pairwise mutual information terms $I(x_i; x_j)$ are calculated and stored into the mutual information matrix M. Let $M_{i,j}$ stand for the entry of matrix M. If $M_{i,j}$ is less than a threshold T_M, x_i is

assumed disconnected from x_j. Otherwise, we have to proceed to evaluate all the conditional mutual information terms, given any other gene x_k. If x_k is a gene belonging to a totally different biological process, the conditional mutual information $I(x_i; x_j \mid x_k)$ approximates the mutual information $I(x_i; x_j)$, and both assume large values. If x_k is a hub gene between x_i and x_j, $I(x_i; x_j \mid x_k)$ assumes a small value. Hence, given any other gene, if the least conditional mutual information is greater than a threshold T_S, it can be inferred that x_i connects x_j. The inference algorithm is formulated as following by W. Zhao [55], and it returns the connectivity matrix C, in which a null entry means disconnection.

```
1:  Input gene expression data set;
2:  Initialize n, M ∈ ℜ^{n×n}, L ∈ ℜ^{1×n}, C ∈ {0, 1}^{n×n}, t_M, t_S;
3:  Preprocess the input data set, perform rank
    transformation
4:  for i = 1 to n − 1 do
5:     for j = i + 1 to n do
6:        M_{i,j} ⇐ I(x_i; x_j);
7:        if M_{i,j} < t_M then
8:           C_{i,j} = 0, C_{j,i} = 0;
9:        else
10:          C_{i,j} = 1, C_{j,i} = 1;
11:          for k = 1 to n and k ≠ i, j do
12:             L_k ⇐ I(x_i; x_j|x_k);
13:             if L_k < t_S then
14:                C_{i,j} = 0, C_{j,i} = 0;
15:                Break;
16:             end if
17:          end for
18:       end if
19:    end for
20: end for
21: return C.
```

Figure 3.2 Connectivity inference algorithms [55]

3.2 Boolean Network and Probabilistic Boolean Network (PBN)

The Boolean Network model, originally introduced by Kauffman [56-58], is also very useful to infer gene regulatory networks because it can monitor the dynamic behavior in complicated systems based on large amounts of gene expression data [59-61]. One of the main objectives of Boolean network is to study the logical interactions of genes without knowing the specific details [61, 62]. In a Boolean network (BN), the target gene is predicted by other genes through a Boolean function. A probabilistic Boolean network (PBN), first introduced by Shmulevich et al. in [63], is the stochastic extension of Boolean network. It consists of a family of Boolean networks, each of which corresponds to a contextual condition determined by variables outside the model. As models of genetic regulatory networks, the PBN method has been further developed in many papers. In [64], a model for random gene perturbations was developed to derive an explicit formula for the transition probabilities in the new PBN model. In [65], intervention is treated via external control variables in context-sensitive PBN by extending the results for instantaneously random PBN in several directions. Some learning approaches for PBN have also been explored by [66, 67, and 68]. Considering the same joint probability distribution over common variables, several fundamental relationships of two model classes, PBN and dynamic Bayesian network (DBN), have been discussed in [69].

In a Boolean network, the expression level of a target gene is functionally related to the expression states of other genes using logical rules, and the target gene is updated by other genes through a Boolean function. There are only two gene expression levels (states) in a Boolean network (BN): on and off, which are represented as "activated" and "inhibited". A probabilistic Boolean network consists of a family of Boolean networks

and incorporates rule-based dependencies between variables. In a PBN model, BNs are allowed to switch from one to another with certain probabilities during state transitions. Since Boolean network is just a special case of probabilistic Boolean networks, PBN is usually used by computational biologist to reconstruct gene regulatory networks.

3.2.1 Boolean Network

We use the same definition as in [62, 70] for Boolean network. A Boolean network $G(V,F)$ is defined by a set of nodes (variables) representing genes $V=\{x_1,x_2,...,x_n\}$ (where $x_i \in \{0,1\}$ is a binary variable) and a set of Boolean functions $F=\{f_1,f_2,...,f_n\}$, which represents the transitional relationships between different time points. A Boolean function $f(x_{j_1(i)},x_{j_2(i)},...,x_{j_{k(i)}(i)})$ with $k(i)$ specified input nodes is assigned to node x_i. The gene status (state) at time point $t+1$ is determined by the values of some other genes at previous time point t using one Boolean function f_i taken from a set of Boolean functions F. So we can define the transitions as

$$x_i(t+1)=f(x_{j_1(i)}(t),x_{j_2(i)}(t),...,x_{j_{k(i)}(i)}(t)) \qquad (3.5)$$

where each x_i represents the expression value of gene i, if $x_i=0$, gene i is inhibited; if $x_i=1$, it is activated. The variable $j_{k(i)}$ represents the mapping between gene networks at different time points. Boolean function F represents the rules of regulatory interactions between genes. In Figure 3.3, an example of a Boolean network is shown. The connected graph is represented by (a), and the transition function is defined by (b).

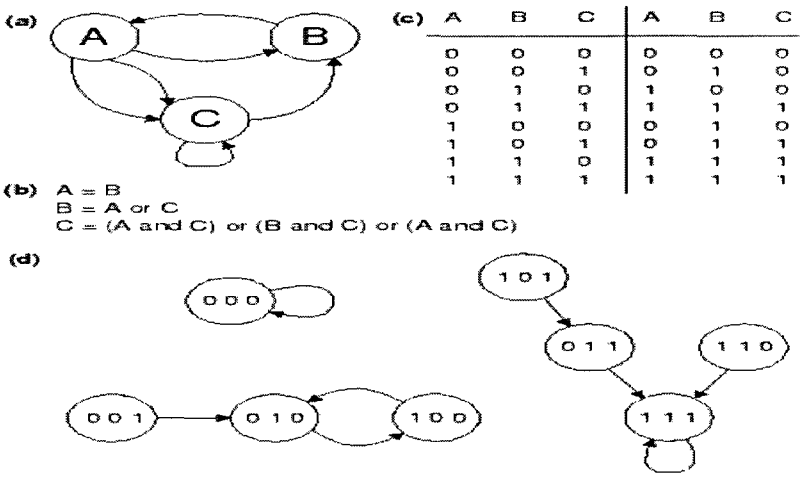


Figure 3.3 An example of a Boolean network: (a) the wiring diagram; (b) the updating rules; (c) a state transition table; (d) the state space of the network

3.2.2 Probabilistic Boolean Network

Probabilistic Boolean network is the extension of Boolean network to combine more than one possible transition Boolean functions, so that each one can be randomly selected to update the target gene based on the selection probability, which is proportional to the coefficient of determination (COD) of each Boolean function. Here we briefly give the same notation of PBN as in [63]. The same set of nodes $V = \{x_1, x_2, ..., x_n\}$ as in Boolean network is used in a PBN $G(V, F)$, but the list of function sets $F = \{f_1, f_2, ..., f_n\}$ is replaced by $F = \{F_1, F_2, ...F_n\}$, where each function set $F_i = \{f_j^{(i)}\}_{j=1,2,...l(i)}$ composed of $l(i)$ possible Boolean functions corresponds to each node x_i. A realization of the PBN at a given time point is determined by a vector of Boolean functions. Each realization of the PBN maps one of the vector functions $f_k = (f_{k(1)}^{(1)}, f_{k(2)}^{(2)}, ...f_{k(n)}^{(n)})$, $1 \le k \le N$, $1 \le k(i) \le l(i)$, where $f_{k(i)}^{(i)} \in F_i$ and N is the number of possible realizations. Given the values of all genes in network at time point t and a realization f_k, the state of the genes after one updating

step is expressed as

$$(x_1(t+1), x_2(t+1), \ldots x_n(t+1)) = f_k(x_1(t), x_2(t), \ldots x_n(t)) \qquad (3.6)$$

Let $\mathrm{f} = (f^{(1)}, f^{(2)}, \ldots f^{(n)})$ denote a random vector taking values in $F_1 \times F_2 \cdots \times F_n$. The probability that a specific transition function $f_j^{(i)}, (1 \le j \le l(i))$ is used to update gene i is equal to

$$c_j^{(i)} = \Pr\{f^{(i)} = f_j^{(i)}\} = \sum_{k: f_{k(i)}^{(i)} = f_j^{(i)}} \Pr\{\mathrm{f} = f_k\} \qquad (3.7)$$

Given genes $V = \{x_1, x_2, \ldots, x_n\}$, each x_i is assigned to a set of Boolean functions $F_i = \{f_j^{(i)}\}_{j=1,2,\ldots l(i)}$ to update target gene. The PBN will reduce to a standard Boolean network if $l(i) = 1$ for all genes. A basic building block of a PBN describing the updating mechanism is shown in Figure 3.4.

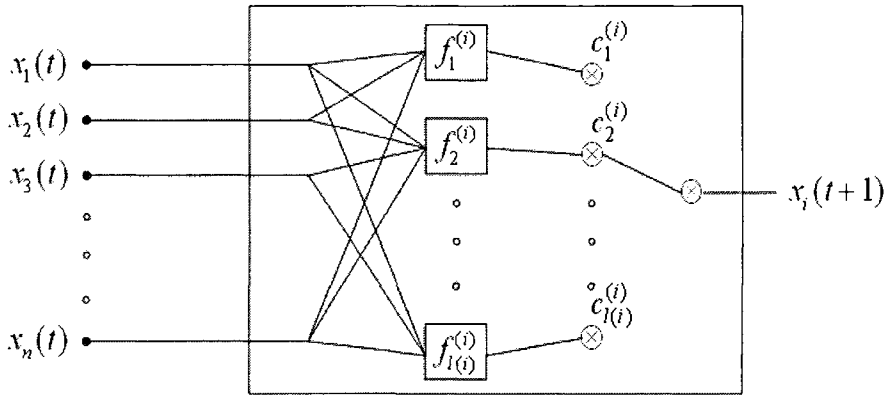


Figure 3.4 A basic building block of a PBN

3.2.3 Inference of Probabilistic Boolean Network

Coefficient of Determination (COD) is used to select a list of predictors for a given gene [63, 67]. So far, most learning methods for reconstructing gene regulatory network

use Coefficient of Determination (COD) to select predictors for each target gene at any time point t. COD is used previously for the steady state data sets. Monte Carlo approaches can be used with probabilistic Boolean networks to approximate dynamics [60] and some theoretical results are given in [71]. Here we use upper case letters to represent random variables: Let X_i be the target gene, $X_1^{(i)}, X_2^{(i)}, \ldots, X_{l(i)}^{(i)}$ be sets of genes and $f_1^{(i)}, f_2^{(i)}, \ldots, f_{l(i)}^{(i)}$ be available Boolean functions. Thus, the optimal predictors of X_i can be defined by $f_1^{(i)}(X_1^{(i)}), f_2^{(i)}(X_2^{(i)}), \ldots, f_{l(i)}^{(i)}(X_{l(i)}^{(i)})$ and the probabilistic error measure can be represented as $\varepsilon(X_i, f_k^{(i)}(X_k^{(i)}))$. For each k, the COD for X_i relative to the conditioning set $X_k^{(i)}$ is defined by

$$\omega_k^i = \frac{\varepsilon_i - \varepsilon(X_i, f_k^{(i)}(X_k^{(i)}))}{\varepsilon_i} \qquad (3.8)$$

where ε_i is the error of the best estimate of X_i [67].

Now, if a class of gene sets $X_1^{(i)}, X_2^{(i)}, \ldots, X_{l(i)}^{(i)}$ which have high CODs has been selected, we can use the optimal Boolean functions $f_1^{(i)}, f_2^{(i)}, \ldots, f_{l(i)}^{(i)}$ as the rule set for gene X_i, with the probability of $f_j^{(i)}$ being chosen. Then the approximations are given by

$$c_k^{(i)} = \frac{\omega_k^i}{\sum_{j=1}^{l(i)} \omega_j^i} \qquad (3.9)$$

According to the above expressions [63, 67], those Boolean functions corresponding to the highest CODs will be selected in the probabilistic network. The selected Boolean functions are used to predict the gene expression status at the next time point, and they also will be used to reconstruct gene regulatory networks.

3.3 Differential and Difference Equations

Differential equations are one of the most important mathematical models in computational biology. Because they can model complex dynamic behavior like oscillations, cyclical patterns, multi-stationarity and switch-like behavior which can be, for example, detected in bacteria infected with lambda phage, it was easy to extend them to be used for modeling gene regulatory networks. There are a lot of profound theories to study systems of differential equations, such that the analysis of these systems is manifold. For computational biologists, the first step is to find differential equations which can represent the system under study precisely. It is essential to know about the processes in the system and to have large amounts of data available to infer the unknown parameters in gene regulatory networks. Necessary foundations for a good description with differential equations would be the knowledge about which gene regulates another and in which way, as well as knowledge about the degradation and maximal production rates of the associated proteins. As in gene regulatory networks most of this information is often missing, there need to be models which can capture the network behaviors and robust methods to estimate the parameters from existing data. Differential equations describe gene expression changes as a function of the expression of other genes and environmental factors. Thus, they are adequate to model the dynamic behaviors of gene regulatory networks in a more quantitative manner. Their flexibility allows us to represent more complex relations among components. A modeling of the gene expression dynamics may apply ordinary differential equations (ODEs): $dx/dt = f(x, p, s, t)$, where $x(t) = \{x_1(t), x_2(t), \ldots, x_n(t)\}$ is the gene expression vector of the genes 1, 2, . . . , n at time

t, f is the function that describes the rate of change of the state variables x_i in dependence on the model parameter set p, and the externally given perturbation signals s. Here, network inference means the identification of function f and parameters p from measured signals x, s and t.

In general, if the gene regulatory networks are without constraints, there are multiple solutions, i.e. the ODE system is not uniquely identifiable (It is also called the problem of identifiablity) from gene expression data achieved from experiments. Thus, the identification of model structure and model parameters requires specifications of the function f and constraints representing prior knowledge, simplifications or approximations. For instance, the function f can be linear or non-linear. Evidently, regulatory processes are characterized by complex non-linear dynamics. However, many gene regulatory network inference approaches based on differential equations only consider linear models or are limited to very specific types of non-linear functions [72, 73].

In recent years, some more complex variants of differential equation models have been proposed to represent complicated gene regulatory networks, such as stochastic differential equations that are incorporated with the stochasticity of gene expression, which may be very useful especially in transcriptional regulatory networks [74, 75].

3.3.1 Linear Differential Equations

Modeling biological data with linear differential equations was first considered theoretically by Chen [76]. In this model, both the mRNA and the protein concentrations were described by a system of linear differential equations. Such a system can be described as following:

$$\frac{dx_i}{dt} = \sum_{j=1}^{n} w_{i,j} \cdot x_j + b_i \cdot s \text{, where } i = 1,2,\ldots,n \qquad (3.10)$$

In this linear differential equation model, it can represent the gene expression dynamics $x_i(t)$ of n genes by $n \times (n+1)$ parameters, which include the $n \times n$ components $w_{i,j}$ of the interaction matrix W and n parameters b_i quantifying, for example, the impact of the perturbation s on gene expression. In general, the simplification obtained by linearization is still not enough to identify large-scale GRN from gene expression data unequivocally.

Several algorithms have been proposed to deal with this problem, such as methods for inferring sparse interaction matrices by reducing the number of non-zero weights $w_{i,j}$.

In order to solve linear differential equations by well-established methods of linear algebra, differential equations can be approximated by difference equations (discrete-time models). After approximation, linear differential Eq. (3.10) becomes the linear difference Eq. (3.11):

$$\frac{x_i[t+\Delta t] - x_i[t]}{\Delta t} = \sum_{j=1}^{n} w_{i,j} \cdot x_j[t] + b_i \cdot s \text{, where } i = 1,2,\ldots,n \qquad (3.11)$$

By this way, we can obtain a linear algebraic equation system that can be solved by methods of linear algebra, such as singular value decomposition (SVD) [77, 78] and regularized least squares regression are the most prominent ones that solve the linear equation system with the constraint of sparseness of the interaction matrix. For instance, the LASSO (Least Absolute Shrinkage and Selection Operator) provides a robust estimation of a network with limited connectivity and low model prediction error [79]. Further inference algorithms based on linear difference equation models are NIR

(Network Identification by multiple Regressions [80]), MNI (Microarray Network Identification [81]) and TSNI (Time-Series Network Identification [82]). Under the steady-state assumption, NIR and MNI use series of steady-state RNA expression measurements, whereas TSNI uses time-series measurements to identify gene regulatory interactions [83].

3.3.2 Non-linear Ordinary Differential Equations

Ordinary differential equations (ODEs) have been widely used to analyze gene regulatory networks, and it is probably the most popular formalism to model dynamical systems in computational biology. The ODE formalism models the concentrations of RNAs, proteins, and other elements of the system by time-dependent variables with values contained in the set of non-negative real numbers. Regulatory interactions take the form of functional and differential relations between the concentration variables.

More specifically, gene regulation is modeled by reaction-rate equations expressing the rate of production of a gene product (a protein or an mRNA) as a function of the concentrations of other components of the system. Reaction-rate equations have the mathematical form:

$$\frac{dx_i}{dt} = f_i(x), \;\; x_i \geq 0, \;\; 1 \leq i \leq n \qquad (3.12)$$

where x is the vector of concentrations of proteins, mRNAs, or small metabolites, and f_i usually is a nonlinear function. The rate of synthesis of i is consider to be dependent upon the concentrations x, possibly including x_i. The equations can be extended to take into account concentrations of $s \geq 0$ input elements, e.g. externally-supplied nutrients:

$$\frac{dx_i}{dt} = f_i(x,s), \;\; x_i \geq 0, \;\; 1 \leq i \leq n \qquad (3.13)$$

They may also take into account discrete time delays arising from the time required to complete transcription, translation, and diffusion to the place of action of a protein:

$$\frac{dx_i}{dt} = f_i(x_1(t-d_{i1}),\ldots,x_n(t-d_{in})),\ \ x_i \geq 0,\ 1 \leq i \leq n \qquad (3.14)$$

where $d_{i1},\ldots,d_{in} > 0$ represent time delays [73, 84, 85, 86] for other ways to represent time delays in gene regulatory networks.

The identification of non-linear models is not only limited by mathematical difficulties and computational efforts for numerical ODE solution and parameter identification, but also mainly by the fact that the sample size M is usually too small for the reliable identification of non-linear interactions. Thus, the search space for non-linear model structure identification has to be stringently restricted. For that reason, inference of non-linear systems employ predefined functions that reflect available knowledge. In [87], Sakamoto and Iba used genetic programming to identify small-scale networks (up to three genes) by fitting polynomial functions f of ordinary differential equations. In [88], Spieth applied different search strategies, such as evolutionary algorithms, for the inference of small-size networks (less than 10 genes). They studied different types of non-linear models: generalized linear network models [89], S-systems [90] and models composed of a linear interaction matrix and an additional non-linear term (called 'H-systems'). Non-linear models such as S-systems consist of many parameters demanding a large number of experiments to fit them to the data [91]. Therefore, the problem of data insufficiency still limits the practical relevance of non-linear models.

3.3.3 Partial Differential Equations

The gene regulatory networks are implicitly assumed to be spatially homogeneous. There are some cases in which these assumptions are not correct. It might be necessary,

for instance, to distinguish between different compartments of a cell, say the nucleus and the cytoplasm, and to consider diffusion of regulatory proteins or metabolites from one compartment to another. Moreover, gradients of protein concentrations across cell tissues are a critical feature in embryonal development. The introduction of time delays for diffusion effects allows some aspects of spatial inhomogeneities to be dealt with, while preserving the basic form of the reaction-rate equations [73]. However, in the case that multiple compartments of a cell, or multiple cells, need to be explicitly modeled, a more drastic extension of Eq. (3.12) becomes necessary.

Suppose that a multi-cellular regulatory system is considered, where the p cells are arranged in a row. A new vector $x^{(l)}(t)$ is introduced, in which the time-varying concentration of gene products is denoted in cell l, l is a discrete variable ranging from 1 to p. Within each cell, regulation of gene expression occurs in the manner described by equation Eq. (3.12). Between pairs of adjacent cells l and $l+1, 1 \le l \le p-1$, diffusion of gene products is assumed to take place proportional to the concentration differences $x_i^{(l+1)} - x_i^{l}$, $x_i^{l} - x_i^{(l-1)}$ and a diffusion constant δ_i. Taken together, this leads to a system of coupled ODEs, so-called reaction-diffusion equations:

$$\frac{dx_i^{(l)}}{dt} = f_i(x^{(l)}) + \delta_i\left(x_i^{(l+1)} - 2x_i^{(l)} + x_i^{(l-1)}\right),\ x_i^{(l)} \ge 0,\ 1 \le i \le n, 1 < l < p \qquad (3.15)$$

If the number of cells is large enough, the discrete variable l in Eq. (3.15) can be replaced by a continuous variable ranging from 0 to λ, where λ represents the size of the regulatory system. The concentration variables x are now defined as functions of both l and t, and the reaction-diffusion equations Eq. (3.15) become partial differential equations (PDEs):

$$\frac{\partial x_i}{\partial t} = f_i(x) + \delta_i \frac{\partial^2 x_i}{\partial l^2}, \quad x_i^{(l)} \geq 0, \; 0 \leq l \leq \lambda, 1 \leq i \leq n \tag{3.16}$$

Reaction-diffusion equations and partial differential equations have been used in computational biology to study pattern formation in development, such as [92, 93]. The induction of models from measurements of x at a sequence of time-points is made attractive by the growing availability of gene expression data. However, precise measurements of absolute expression levels are currently difficult to achieve. In addition, as a consequence of the dimensionality problem, the models need to be simple and are usually strong abstractions of biological processes [94]. For larger and more complicated models, the computational costs of finding an optimal match between the parameter values and the data may be extremely high.

3.4 Bayesian Network and Dynamic Bayesian Network (DBN)

Among the many computational approaches that infer gene regulatory networks from time series data, Bayesian network draws significant attention because of its probabilistic nature. Dynamic Bayesian network is the temporal extension of Bayesian network. It is a general model class that is capable of representing complex temporal stochastic processes. It captures several other often used modeling frameworks as its special cases, such as hidden Markov models (and its variants) and Kalman filter models.

Bayesian network approach has been used in modeling genetic regulatory networks because of its probabilistic nature. However, it has drawbacks such as failing to capture temporal information and modeling cyclic networks. Dynamic Bayesian network approaches are more popular than static Bayesian network because it is easy to interpret and learn. DBN is better suited for characterizing time series gene expression data than its static version. Much recent work has been done to reconstruct gene regulatory

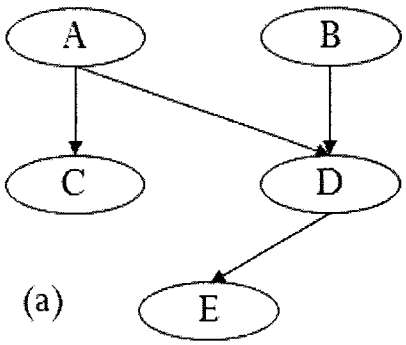
networks from expression data using Bayesian networks and DBN. Perrin BE et al. [95] showed stochastic machine learning algorithm to model gene interactions capable of handling missing variables. Min Zou et al [37] presented a DBN-based approach, in which the number of potential regulators is limited to reduce search space. Yu, J. et al. [34] developed a simulation approach to make advance in dynamic Bayesian network inference algorithm, especially in the context of limited quantities of biological data. In [96], Z.Z Xing and Dan Wu proposed a higher order Markov dynamic Bayesian network (DBN) to model multiple time units in a delayed gene regulatory network. Recently, likelihood maximization algorithms such as the Expectation-Maximization (EM) algorithm have been used to infer hidden parameters and deal with missing data [97].

3.4.1 Bayesian Network

Given a set of variables $U = \{x_1, x_2, ...x_n\}$ in gene network, a Bayesian network, for U is a pair $B = (G, \Theta)$ which encodes a joint probability distribution over all states of U. It is composed of a directed acyclic graph (DAG) G whose nodes correspond to the variables in U and Θ which defines a set of local conditional probability distributions (CPD) to qualify the network. Let $\mathrm{Pa}(x_i)$ denote the parents of the variables x_i in the acyclic graph G and $\mathrm{pa}(x_i)$ denote the values of the corresponding variables. Given G and Θ, a Bayesian network defines a unique joint probability distribution over U given by

$$\Pr\{x_1, x_2, ...x_n\} = \prod_{i=1}^{n} \Pr\{x_i \mid \mathrm{pa}(x_i)\} \qquad (3.17)$$

For more details of Bayesian network, see in [32, 68]. Figure 3.5 shows a simple example of Bayesian network.



$$P(A,B,C,D,E) = P(A)*P(B)*P(C|A) *P(D|A,B)*P(E|D)$$

(b)

Figure 3.5 A simple example of Bayesian network. (a) Graph representation of a Bayesian network. (b) Probability representation corresponding to network in (a).

To describe the joint distribution over n variables (genes), we need to store the probability of every possible event as defined by the values of all the variables. There are exponentially many such events, therefore the space complexity is $O(2^n)$. In Bayesian network representation, if the maximum number of parents is denoted as p, it is easy to see that the space complexity of Bayesian network is $\Theta(2^p \cdot n)$. Since p is usually a much smaller than n, the space requirement of Bayesian networks is much lower than the method which exhaustively enumerates all the possible events.

Bayesian networks reflect the stochastic nature of gene regulation and make use of the Bayes' rule. Here, the assumption is that gene expression values can be described by random variables, which follow probability distributions. As they represent regulatory relations by probability, BNs are thought to model randomness and noise as inherent features of gene regulatory processes [32]. Most importantly, BNs provide a very flexible framework for combining different types of data and prior knowledge in the process of gene regulatory network inference to derive a suitable network structure [98]. Besides, BNs have a number of features that make them attractive candidates for modeling gene

regulatory networks, such as their ability to avoid over-fitting a model to training data and to handle incomplete noisy data as well as hidden variables (e.g. Transcription Factor activities).

Methods for learning BNs are covered in detail in [99]. In short, there are three essential parts for learning a BN: Model selection, parameter learning and model scoring. In model selection, usually heuristics are used to efficiently learn a BN instead of using brute-force search, which will grow exponentially as the number of genes increase in directed acyclic graph. In parameter learning step, given a graph and experimental data, our goal is to find the best conditional probabilities (CP) for each node. In model scoring step, we need to score each candidate model. The higher the score, the better the network model (the DAG and the learned CP distribution) fits to the data. The model with the highest score will represent the GRN inference result.

3.4.2 Dynamic Bayesian Network

Even though Bayesian networks have been widely used in modeling genetic regulatory networks, it has drawbacks such as failing to capture time series information and modeling cyclic networks. Dynamic Bayesian network (DBN) [100] is considered as the temporal extension of Bayesian network. It can represent complex temporal and cyclic relations among genes by incorporating time course (or time slice) information. Here the term "dynamic" means we are modeling a dynamic system, not that the gene regulatory network changes over time. Figure 3.6 shows the DBN can represent cyclic relations among genes by incorporating time series information.

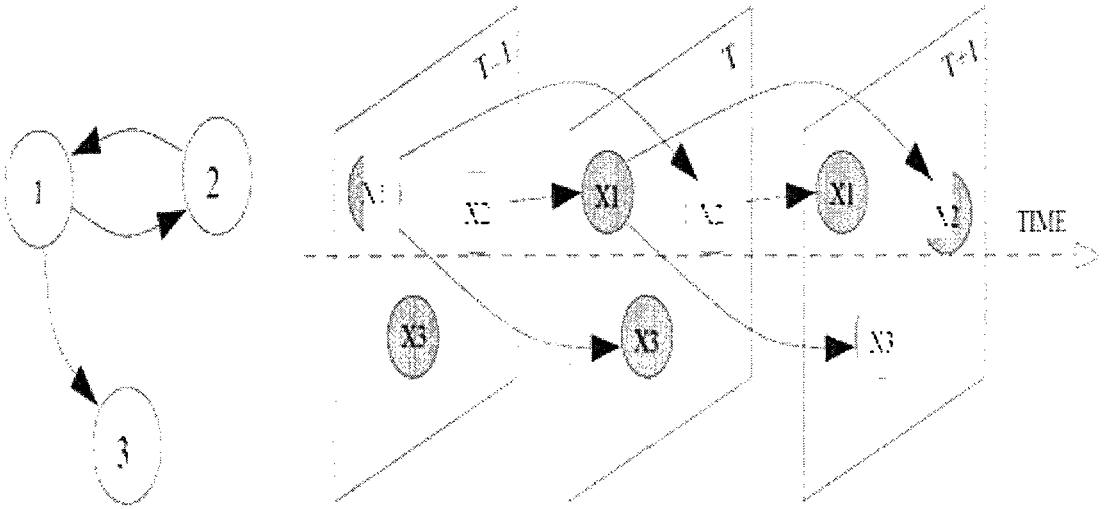


Figure 3.6 Static Bayesian network (left) and DBN (right)

A DBN is defined by a pair (B_0, B_1) represents the joint probability distribution over all possible time series of variables $\mathrm{X} = \{X_1, X_2, ...X_n\}$, where $X_i (1 \le i \le n)$ represents the binary-valued random variables in the network, besides, we use lower case x_i $(1 \le i \le n)$ denotes the values of variable X_i. It is composed of an initial state of Bayesian network $B_0 = (G_0, \Theta_0)$ and a transition two-slice temporal Bayesian network (2TBN) $B_1 = (G_1, \Theta_1)$, where B_0 specifies the joint distribution of the variables in $X(0)$ and B_1 represents the transition probabilities $\Pr\{\mathrm{X}(t+1) \mid \mathrm{X}(t)\}$ for all time slices t. In time slice 0, the parents of $X_i(0)$ are assumed to be those specified in the prior network B_0, which means $\mathrm{Pa}(X_i(0)) \subseteq \mathrm{X}(0)$ for all $1 \le i \le n$; in slice $t+1$, the parents of $X_i(t+1)$ are nodes in slices t, $\mathrm{Pa}(X_i(t+1)) \subseteq \mathrm{X}(t)$ for all $1 \le i \le n$ and $t \ge 0$, as stated in [69], the connections only exist between consecutive slices. Figure 3.7 shows an example of a DBN. The joint distribution over a finite list of random variables $\mathrm{X}(0) \cup \mathrm{X}(1) \cup \cdots \cup \mathrm{X}(\mathrm{T})$ can be expressed as [68, 69]

$$\Pr\{x(0),x(1),\ldots,x(T)\}$$
$$= \Pr\{x(0)\}\prod_{t=0}^{T-1}\Pr\{x(t+1)\,|\,x(t)\} \tag{3.18}$$
$$= \prod_{i=1}^{n}\Pr\{x_i(0)\,|\,\mathrm{pa}(X_i(0))\}\times\prod_{t=0}^{T-1}\prod_{j=1}^{n}\Pr\{x_j(t+1)\,|\,\mathrm{pa}(X_j(t+1))\}$$

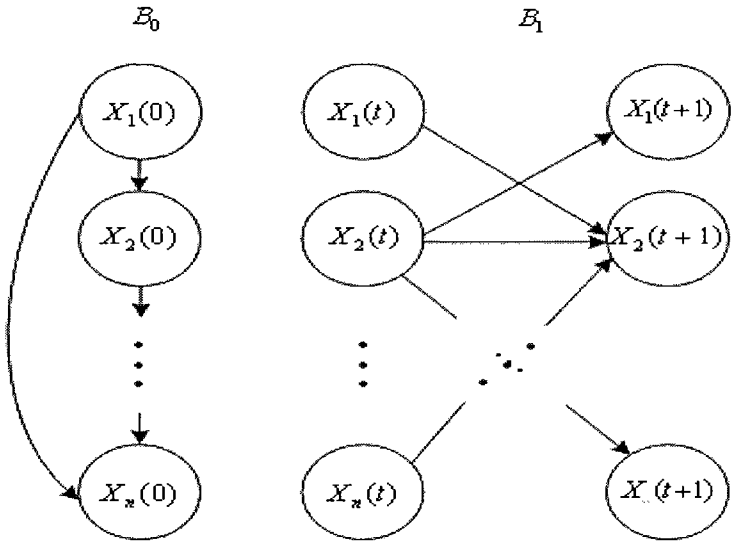


Figure 3.7 A basic building block of DBN

Dynamic Bayesian network allows us to monitor and update the system as time goes by, and even predict further behaviors of the system. In such models word dynamic is connected with a motive force. Changing the nature of the static Bayesian network can be thought of as adapting it to dynamic model. Considering time representation, temporal approaches could be classified into two categories, which represent time as points or as time intervals. Usually time intervals can be thought of as a set of consecutive time points. Therefore, time-points representation seems to be more appropriate and more expressive.

3.5 Learning Bayesian Network

Since it is often either difficult or impossible in practice to manually specify the complete structure and parameters for Bayesian network, learning Bayesian networks

from gene expression data becomes and very important problem. There are two key issues to learn a Bayesian network from given gene expression data: estimating the parameters of the model, and inferring the structure of the gene regulatory network.

3.5.1 Parameter Learning

For a given structure G, traditional parameter learning methods estimate the parameters by maximizing the joint likelihood of gene expression data, which is known as generative parameter learning [101]. Recently, with Bayesian networks becoming widely used as classifiers, some algorithms have been proposed to learn the parameters by maximizing conditional likelihood of the class variable given the observed evidence variables. These methods are known as discriminative parameter learning [102, 103, and 104].

3.5.1.1 Generative Parameter Learning

Given a set of training data D, which is composed of a set of independent and identically distributed instances $[x^1,\ldots,x^N]$, where all components are observed, generative parameter learning methods estimate the parameters of a Bayesian network either by directly maximizing the joint likelihood of training data, or by computing the posterior over parameters θ given a prior distribution $P(\theta)$.

The first method is called maximum likelihood (ML) estimation. Using the conditional independence assumptions encoded in the structure, the joint log likelihood of training data can be factored in the following way

$$\log P(D \mid \theta) = \sum_{i=1}^{N} \sum_{j=1}^{n} \log P(x_j^i \mid x_{\pi(j)}^i, \theta) \qquad (3.19)$$

The second generative method is called maximum a posteriori estimation (MAP). One of the benefits of Bayesian networks is the convenience of incorporating prior domain knowledge and prior knowledge can often be used to effectively avoid over fitting problem, especially when we have limited training data sets. Given a prior density $P(\theta)$, we can learn the parameters to maximize the posterior as following

$$\log P(\theta \mid D) = \log \frac{P(\theta)P(D \mid \theta)}{P(D)} \qquad (3.20)$$

This is equivalent to maximizing $P(\theta)P(D \mid \theta)$ since $P(D)$ is invariant with respect to θ. The Expectation Maximization (EM) algorithm is a classical algorithm which is very popular to learn Bayesian network from incomplete data sets. More details of EM algorithm will be covered later in Chapter 4.

3.5.1.2 Discriminative Parameter Learning

The standard generative parameter learning methods is not the best way to train models for classification. The disadvantage of generative learning is that it optimizes a criterion, such as maximum likelihood or MAP, which is not consistent with performance. In recent years, researchers proposed a new discriminative approach for supervised parameter learning, which takes conditional likelihood as the optimization criterion. For more details of discriminative parameter learning, see in [102 - 106].

Although generative parameter learning given complete data is quite easy, it is very difficult to find the global maximum when using the discriminative conditional likelihood criterion for general Bayesian networks. It has been proved that it is NP-hard problem to find the parameters for a fixed Bayesian network structure that maximize the conditional likelihood of a given sample of incomplete data [102]. Whether this remains true for complete data is an open problem.

3.5.2 Structural Learning

If the topology of the target Bayesian network is fixed, the task is to estimate the CPTs or CPDs for every node in the network. On the other hand, if the topology is unknown, structure learning is required to learn the graph topology of the target BN before the parameters could be determined. Also the observation data used for BN learning may be either complete or incomplete, based on these varieties [107] categorized four cases of learning structure of BNs. Table 3.1 shows the four cases for learning Bayesian network structure.

Table 3.1 Methods for learning Bayesian network structure and parameter determination

Structure / Observability	**Method**
Known, full	Sample statistics
Known, partial	EM or gradient ascent
Unknown, full	Search through model space
Unknown, partial	Structural EM

Full observability means that the values of all variables are known; partial observability means that we do not know the values of some of the variables. This might be because they cannot be measured (in which case they are usually called hidden variables), or because they just happen to be unmeasured in the training data (in which case they are called missing variables). Note that a variable can be observed intermittently.

Unknown structure means we do not know the complete topology of the gene regulatory network. Typically we will know some parts of it, or at least know some

properties the graph is likely to have, e.g., the maximum number of parents (fan-in) that a node can take on, and we may know that nodes of a certain "type" only connect to other nodes of the same type. These constraints or information are called prior knowledge. When the structure of the model is known, the learning task becomes one of parameter estimation. Here, we only cover the aspects that are most relevant to learning genetic networks from time series data. For further details, see the review papers [108-110].

3.5.2.1 Known Structure, Full Observability

In this case, the goal of learning is to find the values of the parameters of each CPD which maximizes the likelihood of the training data, which contains S independent sequences, each of which has the observed values of all n nodes (genes) per time slice for each of T slices. Now, we assume the parameter values for all nodes are constant across time, so that for a time series of length T, we get one data point for each CPD in the initial slice, and $T-1$ data points for each of the other CPDs. If there is only one sequence, we cannot reliably estimate the parameters of the nodes in the first slice, so we usually assume these are fixed apriori. Then we have $N = S(T-1)$ samples for each of the remaining CPDs. In cases where N is small compared to the number of parameters that require fitting, we can use a numerical prior to regularize the problem. In this case, we call the estimates Maximum a posterori (MAP) estimates, as opposed to Maximum Likelihood (ML) estimates.

By the chain rule of probability, we find that the joint probability of all the nodes in the graph is as following:

$$\begin{aligned} P(X_1,\ldots,X_m) &= \prod_i P(X_i \mid X_1,\ldots,X_{i-1}) \\ &= \prod_i P(X_i \mid Pa(X_i)) \end{aligned} \tag{3.21}$$

where $m = n(T-1)$ is the number of nodes in the network (excluding the first slice), $Pa(X_i)$ are the parents of node I, and nodes are numbered in topological order. The normalized log likelihood of the training data set $L = \frac{1}{N}\log \Pr(D \mid G)$, where $D = \{D_1, \ldots, D_S\}$ a sum of terms [100], one for each node is:

$$L = \frac{1}{N}\sum_{i=1}^{m}\sum_{l=1}^{S}\log P(X_i \mid Pa(X_i), D_l) \qquad (3.22)$$

The next step is to determine how to estimate the parameters of each type of CPD given its local data $D_l(X_i \mid Pa(X_i))$ by several different supervised learning methods. For more discussions, see in [100, 107].

3.5.2.2 Known Structure, Partial Observability

When the variables are partially observed, the likelihood is

$$L = \sum_l \log P(D_l) = \sum_l \log \sum_h P(H = h, V = D_l) \qquad (3.23)$$

where the innermost sum is over all the assignments to the hidden nodes H , and $V = D_l$ means the visible nodes take on the values specified by case D_l . Unlike the fully observed case, the log-likelihood in this case cannot be decomposed into a sum of local terms. The output of parameter estimation would be a distribution over the parameters.

Since the likelihood surface becomes multimodal in this case, and we must use iterative methods, such as Expectation Maximization [111] algorithm or gradient ascent [112], to find a local maximum of the ML/MAP function. These algorithms need to use an inference algorithm to compute the expected sufficient statistics (or related quantity) for each node.

3.5.2.3 Unknown Structure, Full Observability

If the knowledge of a number and type of some states in the network is known, but the knowledge of their relations and mutual independences are kept unknown, our goal is to find the best way to learn the structure of DBN from observable data and expert knowledge. In this part, we first introduce the scoring function which we used to select models, and then we discuss algorithms which attempt to optimize this scoring function over the space of models.

First we introduce the objective function which is used for model selection. Note that our goal is to find the model with maximum likelihood, this is stated as finding the model in which the sum of the mutual information (MI) [113] between each node and its parents maximal. But the problem is, the maximum likelihood model will be a complete graph, since this has the largest number of parameters and hence can fit the given gene expression data the best. A well-principled way to avoid this kind of over-fitting is to put a prior on models, specifying that we prefer sparse models. Then, by Bayes' rule, we gain the MAP model that maximizes

$$\Pr(G \mid D) = \frac{\Pr(D \mid G)\Pr(G)}{\Pr(D)} \qquad (3.24)$$

If we take the logarithms of Eq. (3.24), we will simplify the problem of minimizing

$$\log \Pr(G \mid D) = \log \Pr(D \mid G)\Pr(G) + \Pr(G) + c \qquad (3.25)$$

where $c = \Pr(D)$ is a constant independent of G. This approach is known as Minimum Description Length (MDL) approach. An exact Bayesian approach to model selection is usually unfeasible, since the marginal likelihood $\Pr(D) = \sum_{G} \Pr(D, G)$ needs to be computed, which is a sum over an exponential number of models.

However, we can use asymptotic approximation to the posterior, and there are two popular Bayesian scoring metrics called BIC (Bayeisan Information Criterion) score [114] and BDe (Bayesian Dirichlet equivalence) score [115]. Both of these scoring metrics combine the likelihood of the data according to the network with some penalty relating to the complexity of the network. When learning the structure of BNs, this complexity penalty is very important for modeling the DBN, since the maximum likelihood network is usually completely connected network.

Here we take Bayesian Information Criterion as example, it is define as following

$$\log \Pr(G \mid D) \approx \log \Pr(D \mid G, \hat{\Theta}_G) - \frac{\log N}{2} \#G \qquad (3.26)$$

where N is the number of samples, $\#G$ is the dimension of the model, and $\hat{\Theta}_G$ is the maximum likelihood estimate of the parameters. Since the number of parameters in the model is the sum of parameters in each node, BIC score decomposes just like the log likelihood in Eq. (3.23).

Given that the score is decomposable, we can learn the parent set for each node independently. Obviously there are total $\sum_{k=0}^{n} \binom{n}{k} = 2^n$ such parents sets. The problem is to find the highest scoring point in all of the sets. For example, the approach taken by REVEAL [39] is to start from one parent sets and evaluate the score at all points in each successive level until a point is found with a score of 1.0. The scoring function they used in [39] is $I(X, Pa(X)) / \max\{H(X), H(Pa(X))\}$, where $I(X,Y)$ is the mutual information between X and Y , and $H(X)$ is the entropy of X (see definitions in Chapter

3.1), 1.0 is the highest possible score and corresponds to $Pa(X)$ being a perfect predictor of X, for instance, $H(X \mid Pa(X)) = 0$.

If we do not know if we could achieve the maximum possible score of model, we do not know when to stop searching and hence we must evaluate all points in all the possible subsets. If the number of n is large, it is usually computationally infeasible, so a common approach is to only search up until level K (a bound on the maximum number of parents of each node), which takes $O(n^K)$ time. Unfortunately, in real gene regulatory networks, it is known that some genes can have very high fan-in or fan-out (i.e., the hub genes in genetic networks), so restricting the bound to a fixed smaller value (for example, we can set $K = 3$) would make it impossible to discover all of the important "hub" genes.

The obvious way to avoid the exponential cost (and the need for a bound, K) is to use heuristics to avoid examining all possible subsets. Since the problem of learning optimal structure is NP-hard [116], we must use some sort of heuristics. One of the approaches called Extended Dependency Analysis (EDA) [118], which starts by evaluating all subsets of size up to two, keep all the ones with significant mutual information with the target node, and take the union of the resulting set as the set of parents. For more discussions, see in [100, 107, and 117].

3.5.2.4 Unknown Structure, Partial Observability

When the structure of model is unknown and there are hidden variables or missing data, the difficulty is that the score does not decompose. In this case, we need to sum out all the latent variables Z as well as integrate out all the parameters θ:

$$P(X \mid G) = \sum_{Z} \int_{\theta} P(X, Z \mid G, \theta) P(\theta \mid G) \qquad (3.27)$$

This score does not decompose into a product of local terms. But, as observed in [68, 119, 120], the *expected* score does decompose, so we can use an iterative method, which alternated between evaluating the expected score of a model, and then change the model structure, until a local maximum is reached. This is called the Structural Expectation Maximization (SEM) algorithm. SEM was successfully used to learn the structure of discrete DBN with missing data in [68].

SEM algorithm has the same E-step as EM, completing the data by computing expected counts based on the current structure and parameters. The M-step has two parts. First it is used in the same way as described above for recalculating maximum likelihood. In the second part, the M-step of SEM can use the expected counts according to the current structure to evaluate any other candidate structure. This is usually done by performing a complete search over all possible structures similar to one we have. The key step in SEM algorithm is to use and existing DBN to complete the data. All possible trajectories that are consistent with the partial information we have are considered. Then we average the counts in each one of these trajectories based on the probability that our current model assigns to that trajectory.

3.5.3 DBN Implementation

Kevin Murphy [100, 107] implemented a dynamic Bayesian network toolbox using the mathematical programming language (MATLAB). The toolbox is called the Bayes Net Toolbox (BNT), which is an open-source package for directed graphical models. It can support many types of nodes (probability distributions), exact and approximate inference, parameter and structure learning, and static as well as dynamic models. BNT can be freely downloaded from [121].

In order to analyze gene expression data, it was first transformed from continuous to discrete form. The number of discrete steps to be used is arbitrary, but for computational reasons it should be kept as low as possible. Using three steps, one for unchanged, one for up-regulated, and one for down-regulated expression levels, is the natural choice.

The actual structure learning was performed by calling the one of BNT functions learn_struct_dbn_reveal, which uses the previously discussed REVEAL algorithm [39]. The resulting inter-slice adjacency matrix, which defines the transition network, was then visualized using built-in BNT function called draw_layout. Usually the number of nodes is restricted to less than 30, and more nodes will be too much time consuming according to tests. There are many other DBN implementations based one time series gene expression data, such as [34, 37, 95, and 96].

3.6 Time-delayed Dynamic Bayesian Network

There are two major problems with current DBN methods that greatly reduce their effectiveness. The first problem is the lack of a systematic way to determine a biologically relevant transcriptional time lag, which results in relatively low accuracy of predicting gene regulatory networks. The second problem is the excessive computational cost of these analyses, which limits the applicability of current DBN analyses to a large-scale microarray data. Therefore, Min Zou [37] introduces a DBN-based analysis that can predict gene regulatory networks from time course expression data with significantly increased accuracy and reduced computational time. In our DBN analysis, we also use the implementation based on Min Zou. Figure 3.8 shows the process of approach in [37].

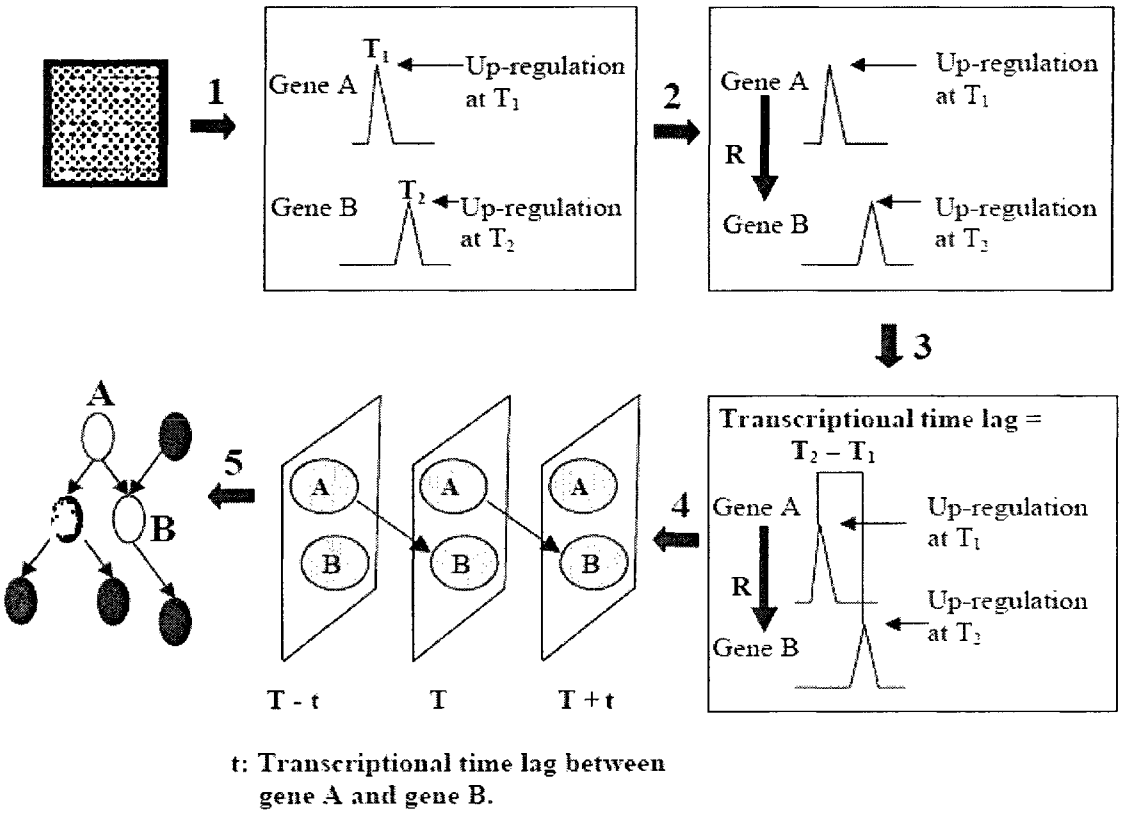


Figure 3.8 Process of time lag DBN. (1) Identification of the initial expression changes. (2) Potential regulators. (3) Estimation of the transcriptional time lag. (4) DBN: statistical analysis of the expression relationship between the potential regulator and its target gene in time slices. (5) Predicted gene regulatory network.

In Murphy's BNT, all the genes in the dataset are considered as potential regulators of a given target gene, which makes it impossible to model large scale gene network because of exponentially increasing computational time. According to [122], most transcriptional regulators exhibit either an earlier or simultaneous change in the expression level when compared to their targets. This is able to limit the potential regulators of each target gene and thus significantly reduce the computational time. The other improvement by Zou is to perform an estimation of the transcriptional time lag between potential regulators and their target genes. The time difference between the initial expression change of a potential regulator and its target gene represents a

biologically relevant time period. Figure 3.9 shows the initial expression change of a potential regulator. This is expected to allow a more accurate estimation of the transcriptional time lag between potential regulators and their targets, because it takes into account variable expression relationships of different regulator–target pairs. These improvements are related to transcriptional time-delayed lags between regulators and target genes, so it can be considered as a time-delayed DBN.

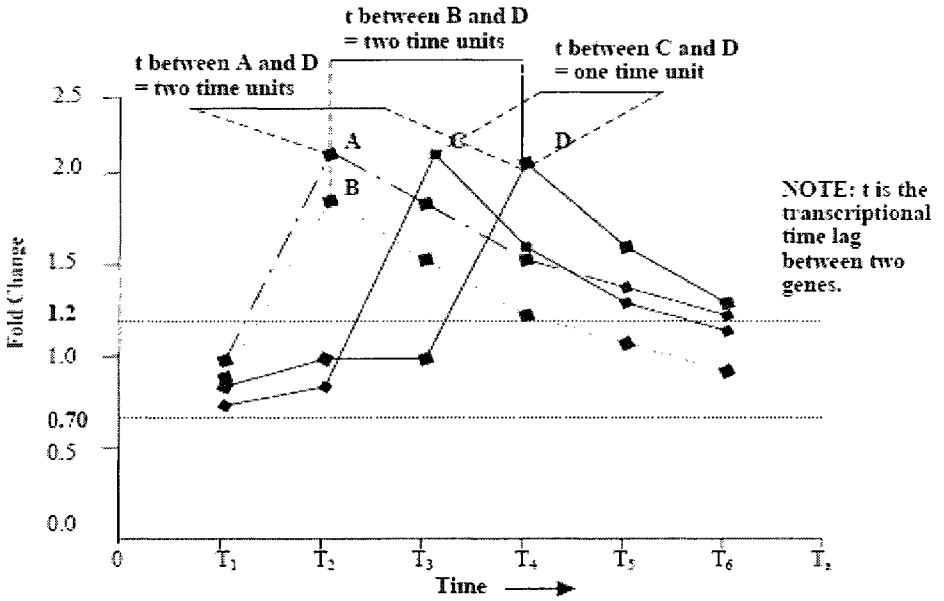


Figure 3.9 The transcriptional time lag between the potential regulator and target gene

CHAPTER IV

A NEW MODEL BASED ON STATE SPACE MODEL AND EM ALGORITHMS

It is a very challenging task to infer a network which has high dimensionality and short time-course gene expression data. The so-called "golden method" does not exist. All the existing computational methods have their advantages and limitations. Boolean network and probabilistic Boolean network are simple dynamic models, and they are also very easy to implement, but they can only deal with discrete data. It will lose lots of useful information when discretizing the continuous data. For dynamic Bayesian network, it can handle both discrete and continuous data, but the limitation of DBN is time complexity. DBN is a NP-hard method, so it is almost impossible for DBN to infer a network with thousands of genes. Some approaches based on DBN have been proposed to limit the potential regulators (parents) of genes, such as [37], so that they can run more genes than original DBN approach (Kevin Murphy's Bayes Net Toolbox for MATLAB [68, 100, and 107]). However, the maximum number of genes DBN can handle on a typical desktop PC is still less than 400, which is not sufficient for real biological experimental data. Ordinary differential equation and partial differential equation are very precise way to model gene regulatory networks, but differential models depend on numerical parameters and need large amount of data, which are very often difficult to get from experiments, and its time complexity is very high.

One of the challenges in GRN inference is that the number of time points is much smaller than that of genes. The traditional time series analysis such as the autoregressive model will fail due to the over-learning problem, because the degree of freedom of the parameters is redundant. To overcome such difficulty, a state space model is proposed to

describe the dynamic system of gene regulation and then the EM algorithms and Kalman filter are used for inferring gene regulatory networks.

4.1 State Space Model

We use state space model to represent the gene regulatory network.

$$x_t = Fx_{t-1} + w_t \tag{4.1}$$

where x_t is a k (number of hidden variables) dimensional vector, called state vector. F is a $k \times k$ matrix called system matrix. A k dimensional vector w_t is a noise term, called system noise. Assuming w_t is distributed according to the Gaussian distribution $w_t \sim N_k(0, Q)$. Given a state vector, observation data are generated by the observation model,

$$y_t = Hx_t + v_t \tag{4.2}$$

Here, $y_t = (y_{1n},..., y_{ln})' \in R^l$ is the observation vector where y_{ln} is the expression value of *i*-th gene measured by the *n*-th time point microarray. The H is a $l \times k$ matrix (l means the number of genes), often called measurement or observation matrix, which describes the relationship between data and state vectors. Here $v_t \in R^l$ is an observation noise, and is distributed as $v_t \sim N_l(0, R)$.

The initial state vector x_0 is assumed to be a Gaussian random vector with mean vector μ_0 and covariance matrix Σ_0, which is $x_0 \sim N_k(\mu_0, \Sigma_0)$. In this problem, we need to estimate unknown parameters and state vector in the model. The dimension of state vector k is also unknown and thus needs to be determined for the optimal one. Figure 4.1 shows the representation of a state space model.

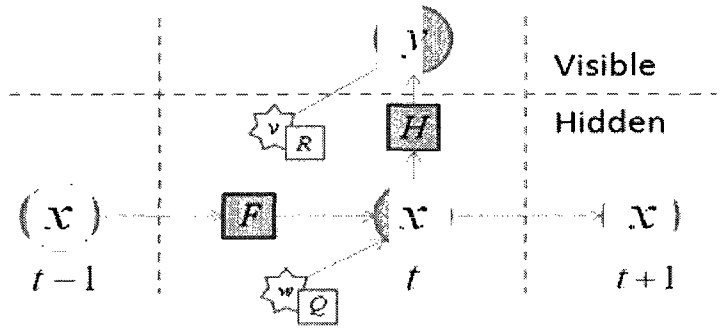


Figure 4.1 Representation of state space model

By estimating the parameter vector $\theta = (H, F, R, Q, \mu_0)$ and the state vector x_t, we can obtain some useful information of the biological system, such as the gene regulatory networks, which gives us the insight of biological networks.

4.2 Maximum Likelihood Estimation with EM Algorithms

Here we derive an EM algorithm [123, 124, and 125] to estimate the parameters of linear dynamic system. There is another previous work which uses Variational Algorithms for Approximate Bayesian Inference [126].

Let $\{Y_T, X_T\}$ be the complete data, where $Y_T = \{y_1, ..., y_T\}$ is the observation data and $X_T = \{x_1, ..., x_T\}$ is the set of state variables (unobserved data). Then the joint likelihood for the complete data is given by

$$P(Y_T, X_T; \theta) = P(x_0) \prod_{t=1}^{T} P(x_t \mid x_{t-1}) P(y_t \mid x_t) \qquad (4.3)$$

The probability densities $P(x_0)$, $P(x_t \mid x_{t-1})$ and $P(y_t \mid x_t)$ are given by the Gaussian distributions $N_k(\mu_0, \Sigma_0)$, $N_k(Fx_{t-1}, Q)$ and $N_l(Hx_t, R)$, respectively.

Now, say Y_t has a multivariate normal distribution, which means $Y_t \sim N(\mu_t, \Sigma_t)$, and the dimension of Y_t is n, then, the density function of a multivariate normal distribution is

$$\phi(Y_t) = \frac{1}{(2\pi)^{n/2} |\Sigma_t|^{1/2}} \exp\left(-\frac{1}{2}(Y_t - \mu_t)' \Sigma_t^{-1} (Y_t - \mu_t) \right) \tag{4.4}$$

It is very hard to maximize the likelihood of a multivariate normal distribution and the linear transformations of multivariate normal random variables are also normally distributed, so we are going to deal with the log of likelihood.

$$P(Y_1, ..., Y_T) = \prod_{t=1}^{T} \phi(Y_t) \tag{4.5}$$

$$\begin{aligned} \log P(Y_1, ..., Y_T) &= \log\left(\prod_{t=1}^{T} \phi(Y_t) \right) \\ &= \sum_{t=1}^{T} \left[-\frac{1}{2} n \log 2\pi - \frac{1}{2} \log |\Sigma_t| - \frac{1}{2}(Y_t - \mu_t)' \Sigma_t^{-1} (Y_t - \mu_t) \right] \\ &= -\frac{1}{2} nT \log 2\pi - \frac{1}{2} \sum_{t=1}^{T} \left[\log |\Sigma_t| - (Y_t - \mu_t)' \Sigma_t^{-1} (Y_t - \mu_t) \right] \end{aligned} \tag{4.6}$$

Back to our case, the log-likelihood of the $P(x_0)$, $P(x_t \mid x_{t-1})$ and $P(y_t \mid x_t)$ is given by following equations, separately [127, 128]:

For $P(x_0)$, since $x_0 \sim N_k(\mu_0, \Sigma_0)$ then

$$\begin{aligned} \log P(x_0) = &-\frac{1}{2} k \log 2\pi - \frac{1}{2} \log |\Sigma_0| \\ &-\frac{1}{2}(x_0 - \mu_0)' \Sigma_0^{-1} (x_0 - \mu_0) \end{aligned} \tag{4.7}$$

For $P(x_t \mid x_{t-1})$, we have $w_t = x_t - Fx_{t-1}$ and $w_t \sim N_k(0, Q)$,

$$\begin{aligned} \log\left(\prod_{t=1}^{T} P(x_t \mid x_{t-1}) \right) = &-\frac{1}{2} kT \log 2\pi - \frac{T}{2} \log |Q| \\ &-\frac{1}{2} \sum_{t=1}^{T} (x_t - Fx_{t-1})' Q^{-1} (x_t - Fx_{t-1}) \end{aligned} \tag{4.8}$$

Similarly, for $P(y_t \mid x_t)$ we have $v_t = y_t - Hx_t$ and $v_t \sim N_l(0,\ R)$,

$$\log\left(\prod_{t=1}^{T} P(y_t \mid x_t)\right) = -\frac{1}{2} lT \log 2\pi - \frac{T}{2}\log|R| - \frac{1}{2}\sum_{t=1}^{T}(y_t - Hx_t)' R^{-1}(y_t - Hx_t) \tag{4.9}$$

After we put them together, the joint log-likelihood [128] of the complete data becomes

$$\begin{aligned}
\log P(Y_T, X_T;\theta) &= \log\left(P(x_0)\prod_{t=1}^{T} P(x_t \mid x_{t-1})P(y_t \mid x_t)\right) \\
&= \log P(x_0) + \sum_{t=1}^{T} \log P(x_t \mid x_{t-1}) + \sum_{t=1}^{T} \log P(y_t \mid x_t) \\
&= -\frac{1}{2}\log|\Sigma_0| - \frac{1}{2}(x_0 - \mu_0)'\Sigma_0^{-1}(x_0 - \mu_0) \\
&\quad -\frac{T}{2}\log|Q| - \frac{1}{2}\sum_{t=1}^{T}(x_t - Fx_{t-1})' Q^{-1}(x_t - Fx_{t-1}) \\
&\quad -\frac{T}{2}\log|R| - \frac{1}{2}\sum_{t=1}^{T}(y_t - Hx_t)' R^{-1}(y_t - Hx_t) \\
&\quad -\frac{k + T(k+l)}{2}\log 2\pi
\end{aligned} \tag{4.10}$$

4.3 Expectation and Maximization Step

An iteration of EM algorithm is composed of two steps, one is expectation step (E-step) and the other one is maximization step (M-step).

4.3.1 E-step

In $\log P(Y_T, X_T;\theta)$, we use following identities for substitution:

$$x_t - Fx_{t-1} = x_{t|T} - Fx_{t-1|T} - (x_{t|T} - x_t) + F(x_{t-1|T} - x_{t-1}) \tag{4.11}$$

$$y_t - Hx_t = y_t - Hx_{t|T} + H(x_{t|T} - x_t) \tag{4.12}$$

The following operations of trace of matrix are needed [127]:

$$
\begin{aligned}
&Trace(A+B) = Trace(B+A) \\
&Trace(AB) = Trace(BA) \ \text{if both } A \text{ and } B \text{ are square} \\
&Trace(d^T Bd) = d^T Bd \quad \text{if d is vector}
\end{aligned}
\tag{4.13}
$$

Substitute Eq. (4.11), (4.12) into Eq. (4.10) and apply above rules, we can rewrite

$$
\begin{aligned}
&\log P(Y_T, X_T; \theta) = \\
&-\frac{1}{2}\log|\Sigma_0| - \frac{1}{2}Tr\left\{\Sigma_0^{-1}\left((x_0 - x_{0|T}) + (x_{0|T} - \mu_0)\right)\left((x_0 - x_{0|T}) + (x_{0|T} - \mu_0)\right)^T\right\} \\
&-\frac{T}{2}\log|Q| - \frac{1}{2}Tr\left\{Q^{-1}\sum_{t=1}^{T}\left\{\begin{matrix}\left(x_{t|T} - Fx_{t-1|T} - (x_{t|T} - x_t) + F(x_{t-1|T} - x_{t-1})\right) \\ \times\left(x_{t|T} - Fx_{t-1|T} - (x_{t|T} - x_t) + F(x_{t-1|T} - x_{t-1})\right)^T\end{matrix}\right\}\right\} \\
&-\frac{T}{2}\log|R| - \frac{1}{2}Tr\left\{R^{-1}\sum_{t=1}^{T}\left\{\begin{matrix}\left(y_t - Hx_{t|T} + H(x_{t|T} - x_t)\right) \\ \times\left(y_t - Hx_{t|T} + H(x_{t|T} - x_t)\right)^T\end{matrix}\right\}\right\} \\
&-\frac{k + T(k+l)}{2}\log 2\pi
\end{aligned}
\tag{4.14}
$$

To estimate the maximum likelihood of parameter θ, the conditional expectation of the joint log-likelihood of the complete data $q(\theta \mid \theta_i)$ can be calculated by

$$
\begin{aligned}
q(\theta \mid \theta_i) &= E[\log P(Y_T, X_T \mid \theta) \mid Y_T, \theta_i] \\
&= -\frac{1}{2}\log|\Sigma_0| - \frac{1}{2}Trace\left\{\Sigma_0^{-1}(P_{0|T} + (x_{0|T} - \mu_0)(x_{0|T} - \mu_0)')\right\} \\
&\quad -\frac{T}{2}\log|Q| - \frac{1}{2}Trace\left\{Q^{-1}(A - BF' - FB' + FCF')\right\} \\
&\quad -\frac{T}{2}\log|R| - \frac{1}{2}Trace\left\{R^{-1}(D+E)\right\} - \frac{k + T(k+l)}{2}\log 2\pi
\end{aligned}
\tag{4.15}
$$

where θ_i is the parameter we estimated in i-th iteration, and, for simplicity,

$$
A = \sum_{t=1}^{T}(P_{t|T} + x_{t|T}x'_{t|T}) \tag{4.16}
$$

$$
B = \sum_{t=1}^{T}(P_{t,t-1|T} + x_{t|T}x'_{t-1|T}) \tag{4.17}
$$

$$
C = \sum_{t=1}^{T}(P_{t-1|T} + x_{t-1|T}x'_{t-1|T}) \tag{4.18}
$$

$$D = \sum_{t=1}^{T} \{(y_t - Hx_{t|T})(y_t - Hx_{t|T})'\} \tag{4.19}$$

$$E = \sum_{t=1}^{T} (HP_{t|T}H') \tag{4.20}$$

The conventional Kalman smoothing estimators $x_{t|T}$, $P_{t|T}$ and $P_{t,t-1|T}$ can be calculated by using Kalman Filter and Kalman smoother algorithms [126, 128].

$$x_{t|T} = E\{x_t \mid Y_T\} \tag{4.21}$$

$$P_{t|T} = E\{(x_t - x_{t|T})(x_t - x_{t|T})' \mid Y_T\} \tag{4.22}$$

$$P_{t,t-1|T} = E\{(x_t - x_{t|T})(x_{t-1} - x_{t-1|T})' \mid Y_T\} \tag{4.23}$$

4.3.2 M-step

In this step, θ_i will be updated to θ_{i+1} by taking the partial derivatives of $\theta_{i+1} = \arg\max q(\theta \mid \theta_i)$.

Estimate F: $$\frac{\partial q(\theta \mid \theta_i)}{\partial F} = 0 \tag{4.24}$$

Estimate H: $$\frac{\partial q(\theta \mid \theta_i)}{\partial H} = 0 \tag{4.25}$$

Estimate R: $$\frac{\partial q(\theta \mid \theta_i)}{\partial R} = 0 \tag{4.26}$$

Estimate Q: $$\frac{\partial q(\theta \mid \theta_i)}{\partial Q} = 0 \tag{4.27}$$

Thus $\theta_{i+1} = \{H(i+1), F(i+1), R(i+1), Q(i+1), \mu_0(i+1)\}$ will be obtained by

$$H(i+1) = \left(\sum_{t=1}^{T} E\{y_t x_t' \mid Y_T\} \right) A^{-1} \tag{4.28}$$

$$F(i+1) = BC^{-1} \tag{4.29}$$

$$R(i+1) = T^{-1}(D+E) \tag{4.30}$$

$$Q(i+1) = T^{-1}(A - BC^{-1}B') \tag{4.31}$$

4.4 Kalman Filter and Kalman Smoother

The conventional Kalman smoothing estimators $x_{t|T}$, $P_{t|T}$ and $P_{t,t-1|T}$ can be calculated by using Kalman Filter [129, 130] and Kalman smoother algorithms [131].

$$x_{t|T} = E\{x_t \mid Y_T\} \tag{4.32}$$

$$P_{t|T} = E\{(x_t - x_{t|T})(x_t - x_{t|T})' \mid Y_T\} \tag{4.33}$$

$$P_{t,t-1|T} = E\{(x_t - x_{t|T})(x_{t-1} - x_{t-1|T})' \mid Y_T\} \tag{4.34}$$

4.4.1 The Filter

The process of Kalman filter can be summarized as following steps:

1. Estimate $\hat{x}_{0|0}$
2. Solve for $\hat{x}_{t|t-1}$ and $P_{t|t-1}$
3. Estimate the best forecast of $\hat{Y}_{t|t-1}$
4. Use $\hat{x}_{t|t-1}$, $P_{t|t-1}$ and the information from the forecast error from step (3) to get $\hat{x}_{t|t}$ and $P_{t|t}$

4.4.1.1 Initialization

Assume that $x_t \sim N(\mu, \Sigma)$, we have $\hat{x}_{0|0} = \mu$ and $P_{0|0} = \Sigma$.

4.4.1.2 Best guess (prediction) of $\hat{x}_{t|t-1}$ and $P_{t|t-1}$

Let Φ_t represents previously known information, then,

$$\begin{aligned} \hat{x}_{t|t-1} &= E[x_t \mid \Phi_{t-1}] = E[Fx_{t-1} \mid \Phi_{t-1}] + E[w_t \mid \Phi_{t-1}] \\ &= F\hat{x}_{t-1|t-1} + \vec{0} = F\hat{x}_{t-1|t-1} \end{aligned} \tag{4.35}$$

$$\begin{aligned} P_{t|t-1} &= E\left[(\hat{x}_{t|t-1} - x_t)(\hat{x}_{t|t-1} - x_t)'\right] \\ &= E\left[\left(F(\hat{x}_{t-1|t-1} - x_{t-1}) - w_t\right)\left(F(\hat{x}_{t-1|t-1} - x_{t-1}) - w_t\right)'\right] \\ &= E\left[F(\hat{x}_{t-1|t-1} - x_{t-1})(\hat{x}_{t-1|t-1} - x_{t-1})'F'\right] - E\left[F(\hat{x}_{t-1|t-1} - x_{t-1})w_t'\right] \\ &\quad -E\left[F(\hat{x}_{t-1|t-1} - x_{t-1})'w_t\right] + E\left[w_t w_t'\right] \end{aligned} \tag{4.36}$$

Because x_t and w_t are independent, so

$$0 = E\left[F(\hat{x}_{t-1|t-1} - x_{t-1})w_t'\right] = E\left[F(\hat{x}_{t-1|t-1} - x_{t-1})'w_t\right] \tag{4.37}$$

Then we have

$$\begin{aligned} P_{t|t-1} &= FE\left[(\hat{x}_{t-1|t-1} - x_{t-1})(\hat{x}_{t-1|t-1} - x_{t-1})'\right]F' + E\left[w_t w_t'\right] \\ &= FP_{t-1|t-1}F' + Q \end{aligned} \tag{4.38}$$

4.4.1.3 Updating from $\hat{x}_{t|t-1}$, $P_{t|t-1}$ to $\hat{x}_{t|t}$, $P_{t|t}$

With the assumption of a prior estimate $\hat{x}_{t|t-1}$, we now seek to use the measurement Y_t to improve the prior estimate. We choose a linear blending of the noisy measurement and the prior estimate in accordance with the equation

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(Y_t - H\hat{x}_{t|t-1}) \tag{4.39}$$

where $\hat{x}_{t|t}$ is updated estimate and K_t is the blending factor (or Kalman gain) to be determined. The problem now is to find the particular blending factor K_t which yields an updated estimate that is optimal. The expression for the error covariance matrix associated with the updated estimate is

$$P_{t|t} = E\left[(x_t - \hat{x}_{t|t})(x_t - \hat{x}_{t|t})'\right] \tag{4.40}$$

Next, we substitute Eq. (4.2) into Eq. (4.39) and then substitute the resulting expression for $\hat{x}_{t|t}$ into Eq. (4.40). The result is

$$
\begin{aligned}
P_{t|t} &= E\left[(x_t - \hat{x}_{t|t})(x_t - \hat{x}_{t|t})^T\right] \\
&= E\left[(x_t - \hat{x}_{t|t-1} - K_t(Hx_t + v_t - H\hat{x}_{t|t-1}))(x_t - \hat{x}_{t|t-1} - K_t(Hx_t + v_t - H\hat{x}_{t|t-1}))^T\right] \\
&= E\left[\left((x_t - \hat{x}_{t|t-1}) - K_t(Hx_t + v_t - H\hat{x}_{t|t-1})\right)\left((x_t - \hat{x}_{t|t-1}) - K_t(Hx_t + v_t - H\hat{x}_{t|t-1})\right)^T\right] \\
&= E\left[\left((x_t - \hat{x}_{t|t-1}) - K_tH(x_t - \hat{x}_{t|t-1}) - K_tv_t\right)\left((x_t - \hat{x}_{t|t-1}) - K_tH(x_t - \hat{x}_{t|t-1}) - K_tv_t\right)^T\right] \\
&= E\left[(x_t - \hat{x}_{t|t-1})(x_t - \hat{x}_{t|t-1})^T\right] - E\left[(x_t - \hat{x}_{t|t-1})(x_t - \hat{x}_{t|t-1})^T H^T K_t^T\right] \\
&\quad - E\left[(x_t - \hat{x}_{t|t-1})v_t^T K_t^T\right] - E\left[K_tH(x_t - \hat{x}_{t|t-1})(x_t - \hat{x}_{t|t-1})^T\right] \\
&\quad + E\left[K_tH(x_t - \hat{x}_{t|t-1})(x_t - \hat{x}_{t|t-1})^T H^T K_t^T\right] + E\left[K_tH(x_t - \hat{x}_{t|t-1})v_t^T K_t^T\right] \\
&\quad - E\left[K_tv_t(x_t - \hat{x}_{t|t-1})^T\right] + E\left[K_tv_t(x_t - \hat{x}_{t|t-1})^T H^T K_t^T\right] + E\left[K_tv_tv_t^T K_t^T\right]
\end{aligned}
\tag{4.41}
$$

After combination, the result will be

$$
\begin{aligned}
P_{t|t} &= P_{t|t-1} - P_{t|t-1}H^T K_t^T - K_tHP_{t|t-1} + K_tHP_{t|t-1}H^T K_t^T + K_tRK_t^T \\
&= P_{t|t-1}(I - H^T K_t^T) - K_tHP_{t|t-1}(I - H^T K_t^T) + K_tRK_t^T \\
&= (I - K_tH)P_{t|t-1}(I - H^T K_t^T) + K_tRK_t^T \\
&= (I - K_tH)P_{t|t-1}(I - K_tH)^T + K_tRK_t^T
\end{aligned}
\tag{4.42}
$$

Returning to the optimization problem, we hope to find the particular blending factor K_t that minimizes the individual terms along the major diagonal of $P_{t|t}$, because these terms represent the estimation error variances for the elements of the state vector being estimated. There are a lot of ways to optimize it. We will use a straightforward differential calculus approach to do so. Here, two matrix differential formulas are needed [128]:

$$
\frac{d\left[Trace(AB)\right]}{dA} = B^T \quad \text{(AB must be square)} \tag{4.43}
$$

$$
\frac{d\left[Trace(ACA^T)\right]}{dA} = 2AC \quad \text{(C must be symmetric)} \tag{4.44}
$$

Now rewrite Eq. (4.42) in the following form,

$$P_{t|t} = P_{t|t-1} - P_{t|t-1}H^T K_t^T - K_t H P_{t|t-1} + K_t(HP_{t|t-1}H^T + R)K_t^T \qquad (4.45)$$

We wish to minimize the trace of $P_{t|t}$ because it is the sum of the mean-square errors in the estimates of all the elements of the state vector. Now we differentiate the trace of $P_{t|t}$ with respect to K_t, note that the trace of $P_{t|t-1}H^T K_t^T$ is equal to the trace of its transpose $K_t H P_{t|t-1}$. The result is

$$\frac{d(Trace\ P_{t|t})}{dK_t} = -2(HP_{t|t-1})^T + 2K_t(HP_{t|t-1}H^T + R) \qquad (4.46)$$

Now we solve for the optimal gain by setting the derivative equal to zero and the gain is

$$K_t = P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1} \qquad (4.47)$$

This particular K_t which minimizes the mean-square estimation error, is named the *Kalman gain.* Once we know the Kalman gain, the covariance matrix with the optimal estimate now can be computed. Substituting Eq. (4.47) into Eq. (4.45), we have

$$P_{t|t} = P_{t|t-1} - P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1}HP_{t|t-1} \qquad (4.48)$$

or

$$P_{t|t} = P_{t|t-1}(I - K_t H) \qquad (4.49)$$

This concludes the filtering process. Now we continue to derive the Kalman smoother.

4.4.2 The Smoother

The smoothing problem is where we hope to form the optimal estimate at some point back in the past, relative to the current measurement. We smoothed by forming an inference about the value of x_t based on the full set of data, denoted by

$$x_{t|t-1} = E[x_t \mid \Phi_{t-1}] \qquad (4.50)$$

$$P_{t|T} = E\left[(x_t - x_{t|T})(x_t - x_{t|T})'\right] \tag{4.51}$$

Recall the equations Eq. (4.35), Eq. (4.38), Eq. (4.39), Eq. (4.40) and Eq. (4.47) and simplify them before we continue:

$$\hat{x}_{t|t-1} = F\hat{x}_{t-1|t-1} \tag{4.35}$$

$$P_{t|t-1} = FP_{t-1|t-1}F' + Q \tag{4.38}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(Y_t - H\hat{x}_{t|t-1}) \tag{4.39}$$

$$P_{t|t} = E\left[(x_t - \hat{x}_{t|t})(x_t - \hat{x}_{t|t})'\right] \tag{4.40}$$

$$K_t = P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1} \tag{4.47}$$

4.4.2.1 Updating from $x_{t|t}$ to $x_{t|T}$

Using the linear projection techniques, we can get a new estimate of x_t

$$E[x_t \mid x_{t+1}] = x_{t|t} + (x_{t+1} - x_{t+1|t}) \times \frac{E\left[(x_t - x_{t|t})(x_{t+1} - x_{t+1|t})'\right]}{E\left[(x_{t+1} - x_{t+1|t})(x_{t+1} - x_{t+1|t})'\right]} \tag{4.48}$$

where

$$\begin{aligned}
&E\left[(x_t - x_{t|t})(x_{t+1} - x_{t+1|t})'\right] \\
&= E\left[(x_t - x_{t|t})(Fx_t + w_{t+1} - Fx_{t|t})'\right] \\
&= E\left[(x_t - x_{t|t})(x_t'F' + w_{t+1}' - x_{t|t}'F')\right] \\
&= E\left[(x_t - x_{t|t})(x_t'F' - x_{t|t}'F') + (x_t - x_{t|t})w_{t+1}'\right] \\
&= E\left[(x_t - x_{t|t})(x_t' - x_{t|t}')F'\right] \\
&= E\left[(x_t - x_{t|t})(x_t - x_{t|t})'\right]F' \\
&= P_{t|t}F'
\end{aligned} \tag{4.49}$$

$$E[x_t \mid x_{t+1}] = x_{t|t} + (x_{t+1} - x_{t+1|t})P_{t|t}F'P_{t+1|t}^{-1} \tag{4.50}$$

Let $J_t = P_{t|t}F'P_{t+1|t}^{-1}$, then we can write

$$E[x_t \mid x_{t+1}] = x_{t|t} + J_t(x_{t+1} - x_{t+1|t}) \tag{4.51}$$

Thus,

$$\begin{aligned} E[x_t \mid \Phi_T] &= E\left[x_{t|t} + J_t(x_{t+1} - x_{t+1|t}) \mid \Phi_T \right] \\ &= x_{t|t} + J_t\left(E[x_{t+1} \mid \Phi_T] - x_{t+1|t} \right) \end{aligned} \tag{4.52}$$

Now we have

$$x_{t|T} = x_{t|t} + J_t(x_{t+1|T} - x_{t+1|t}) \tag{4.53}$$

4.4.2.2 Updating from $P_{t|t}$ to $P_{t|T}$

We can rewrite Eq. (4.53) as following

$$x_t - x_{t|T} = x_t - x_{t|t} - J_t x_{t+1|T} + J_t x_{t+1|t} \tag{4.54}$$

$$x_t - x_{t|T} + J_t x_{t+1|T} = x_t - x_{t|t} + J_t x_{t+1|t} \tag{4.55}$$

Now multiply both sides by their respective transposes, foil, and take expectations.

$$LHS = (x_t - x_{t|T} + J_t x_{t+1|T})(x_t - x_{t|T} + J_t x_{t+1|T})' \tag{4.56}$$

$$RHS = (x_t - x_{t|t} + J_t x_{t+1|t})(x_t - x_{t|t} + J_t x_{t+1|t})' \tag{4.57}$$

The expectation of left side of Eq. (4.55) is

$$\begin{aligned} E[LHS] &= E\left[(x_t - x_{t|T} + J_t x_{t+1|T})(x_t - x_{t|T} + J_t x_{t+1|T})' \right] \\ &= E\left[(x_t - x_{t|T})(x_t - x_{t|T})' \right] + E\left[(x_t - x_{t|T}) x'_{t+1|T} J'_t \right] \\ &\quad + E\left[J_t x_{t+1|T} (x_t - x_{t|T})' \right] + E\left[J_t x_{t+1|T} x'_{t+1|T} J'_t \right] \\ &= P_{t|T} + J_t E\left[x_{t+1|T} x'_{t+1|T} \right] J'_t \end{aligned} \tag{4.58}$$

$$\begin{aligned} E[RHS] &= E\left[(x_t - x_{t|t} + J_t x_{t+1|t})(x_t - x_{t|t} + J_t x_{t+1|t})' \right] \\ &= E\left[(x_t - x_{t|t})(x_t - x_{t|t})' \right] + E\left[(x_t - x_{t|t}) x'_{t+1|t} J'_t \right] \\ &\quad + E\left[J_t x_{t+1|t} (x_t - x_{t|t})' \right] + E\left[J_t x_{t+1|t} x'_{t+1|t} J'_t \right] \\ &= P_{t|t} + J_t E\left[x_{t+1|t} x'_{t+1|t} \right] J'_t \end{aligned} \tag{4.59}$$

Because $E[LHS] = E[RHS]$, so we have

$$P_{t|T} = P_{t|t} + J_t \left\{ E\left[x_{t+1|t} x'_{t+1|t} \right] - E\left[x_{t+1|T} x'_{t+1|T} \right] \right\} J'_t \tag{4.60}$$

In order to simplify the Eq. (4.60), we rewrite above bracketed term as

$$\begin{aligned} & E\left[x_{t+1|t} x'_{t+1|t} \right] - E\left[x_{t+1|T} x'_{t+1|T} \right] \\ & = \left(E\left[x_{t+1} x'_{t+1} \right] - E\left[x_{t+1|T} x'_{t+1|T} \right] \right) - \left(E\left[x_{t+1} x'_{t+1} \right] - E\left[x_{t+1|t} x'_{t+1|t} \right] \right) \end{aligned} \tag{4.61}$$

Notice that

$$E\left[x_{t+1|T} x'_{t+1|T} \right] = E\left[x_{t+1|T} x'_{t+1} \right] = E\left[x_{t+1} x'_{t+1|T} \right] \tag{4.62}$$

Now, substituting Eq. (4.62) into Eq. (4.61) and combine them,

$$\begin{aligned} & E\left[x_{t+1|t} x'_{t+1|t} \right] - E\left[x_{t+1|T} x'_{t+1|T} \right] \\ & = E\left[(x_{t+1} - x_{t+1|T})(x_{t+1} - x_{t+1|T})' \right] - E\left[(x_{t+1} - x_{t+1|t})(x_{t+1} - x_{t+1|t})' \right] \\ & = P_{t+1|T} - P_{t+1|t} \end{aligned} \tag{4.63}$$

Substituting Eq. (4.63) into Eq. (4.60), we have

$$P_{t|T} = P_{t|t} + J_t (P_{t+1|T} - P_{t+1|t}) J'_t \tag{4.64}$$

At the end of filtering process, we have calculated and stored $x_{t|t}$, $x_{t+1|t}$, $P_{t|t}$ and $P_{t+1|t}$. We use $J_t = P_{t|t} F' P_{t+1|t}^{-1}$ to generate a sequence $\{J_t\}_{t=1}^{T-1}$. Then, we use

$$x_{t|T} = x_{t|t} + J_t (x_{t+1|T} - x_{t+1|t}) \quad \text{for } t = T-1 \tag{4.65}$$

to calculate

$$x_{T-1|T} = x_{T-1|T-1} + J_t (x_{T|T} - x_{T|T-1}) \tag{4.66}$$

Iteratively using this equation, we can continue on and generate sequence $\{x_{t|T}\}_{t=1}^{T-1}$.

4.4.3 Identification of Dimension of State Variable

In order to use EM algorithms to estimate the parameters, we need to know the dimension of state variables x. BIC (Bayesian Information Criterion) [114] is introduced to determine an optimal dimension of the state vector k

$$BIC(k) = \beta \log T - 2 \log L(Y_T \mid \theta_k) \tag{4.67}$$

where $\log L(Y_T \mid \theta_k)$ is the maximum log-likelihood with the parameter vector θ_k which is estimated in EM algorithm. The number of parameters to be estimated is denoted by β and T is the number of time points in our gene dataset. The dimension of the state vectors which has the minimum BIC is determined as the optimal one. In our work, we mainly focused on inferring the gene regulatory networks, which means that we only care about relationships between observation variables. So, for simplicity, we set the dimensionality of state variable as a fixed value (4 or 5).

4.4.4 Problem of Identifiablity

There is a substantial problem for system identification by using linear dynamic system (state space model). It will lack the identifiablity if there is not any constraint for the parameter space [125]. Lacking the identifiablity means that there exists infinite number of parameters yielding the same likelihood.

Let Φ be an arbitrary non-singular $(k \times k)$ matrix. The state function Eq. (4.1) and observation function Eq. (4.2) can be replaced by following equations

$$\Phi x_t = \Phi F \Phi^{-1} \Phi x_{t-1} + \Phi w_t \tag{4.68}$$

$$y_t = H \Phi^{-1} \Phi x_t + v_t \tag{4.69}$$

If we let $x_t' = \Phi x_t$, $F' = \Phi F \Phi^{-1}$, $H' = H\Phi^{-1}$ and $w_t' = \Phi w_t$ (or we can write as $Q = \Phi Q \Phi'$), the above equations are equivalent under arbitrary transformations, that is,

$$x_t' = F' x_{t-1}' + w_t' \tag{4.70}$$

$$y_t = H' x_t' + v_t \tag{4.71}$$

To avoid the problem of identifiablity, we need add some constraints to parameter space of our model, Q need to be an identity matrix $Q = I_k$ and $H'R^{-1}H = \Lambda$ need to be a diagonal matrix.

4.5 Derivation of Gene Interaction Matrix

In order to get the connectivity matrix of genes based on gene expression data, we need to get the relationship between y_t and y_{t-1}, recall Eq. (4.1) and (4.2),

$$x_t = F x_{t-1} + w_t \tag{4.1}$$

$$y_t = H x_t + v_t \tag{4.2}$$

According to singular value decomposition, we can decompose the observation matrix H into products of matrices $H = U\Sigma V^T$, where U and V are orthogonal, and $\Sigma = diag(\delta_1, ..., \delta_r)$, $r = \min(l, k)$, with $\delta_1 \geq \delta_2 \geq ... \geq \delta_r \geq 0$. From this decomposition, we have

$$x_t = H^+ y_t - H^+ v_t \tag{4.72}$$

where $H^+ = VL^+U^T$. The $H^+ y_t$ is a mapping between R^l and R^k. Thus, we have

$$y_t = HFH^+ y_{t-1} - HFH^+ v_{t-1} + Hw_t + v_t \tag{4.73}$$

Let $K = HFH^+$ and $\xi_t = Hw_t + v_t - HFH^+ v_{t-1}$, then we have

$$y_t = Ky_{t-1} + \xi_t \tag{4.74}$$

If the condition number is small, we can multiply transpose matrix of H to both sides of Eq. (4.2), which will become

$$H^T y_t = H^T H x_t + H^T v_t \qquad (4.75)$$

Now the $H^T H$ is a $l \times l$ matrix and it has normal inverse matrix, then we can rewrite Eq. (4.75) as following

$$x_t = (H^T H)^{-1} H^T y_t - (H^T H)^{-1} H^T v_t \qquad (4.76)$$

Substitute Eq. (4.76) into Eq. (4.1), we have

$$y_t = HF(H^T H)^{-1} H^T y_{t-1} - H(H^T H)^{-1} H^T v_{t-1} + Hw_t + v_t \qquad (4.77)$$

Let $K = HF(H^T H)^{-1} H^T$ and $\xi_t = Hw_t + v_t - H(H^T H)^{-1} H^T v_{t-1}$, then we also have Eq. (4.74) $y_t = Ky_{t-1} + \xi_t$

If the condition number is too large, we need to decompose H using SVD (singular value decomposition) or other methods to solve the pseudo-inverse matrix H. The matrix K represents the gene-gene interactions. We can reconstruct gene networks by estimating the matrix K. The parameters of state space model (linear dynamical system) can be estimated by the Kalman filter and the EM algorithms (For details, please check Chapter 4.2 and Chapter 4.3).

CHAPTER V

GRN INFERENCE FROM PBN AND DBN

Both synthetic and real biological data sets are used to infer gene regulatory networks from time series gene expression data based on DBN approach. For synthetic data, we use the DREAM project [132] *in silico* data, which is provided by the Laboratory of Intelligent Systems of the Swiss Federal Institute of Technology in Lausanne. For real biological data, there are three different data sets. First one is Drosophila muscle development gene regulatory networks from Drosophila Interaction Database [133]. Second one is Yeast *Saccharomyces cerevisiae* from Spellman [134]. The last one is fish ovary data, which is provided by the Environmental Laboratory of U.S. Army Engineer Research and Development Center.

5.1 DREAM Synthetic Data

Due to the limitations of real biological data, the simulated data from *in silico* gene networks is becoming the only possibility to systematically evaluate the performances from different genetic networks inferring algorithms. Here, *in silico* means "computer generated". In simulated data, all aspects of the networks are under full control and different types of data and levels of noise are allowed in such *in silico* networks, which are composed of a network topology that determines the structure and a model for each of the interactions among the genes.

DREAM is a Dialogue for Reverse Engineering Assessments and Methods. The main objective is to catalyze the interaction between experiment and theory in the area of cellular network inference. In the given synthetic datasets, there are three *in silico* data sets corresponding to gene networks with 10, 50, and 100 genes. Each data set consists of

five gold standard networks, which are sub-networks with a topology of connections subtracted from the E.Coli and Yeast gene regulatory networks. The rationale is that in this way it will be possible to assess how consistently a method predicts the topology in five independent networks of the same type and size.

strain	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
wt	0.529871	0.517426	0.818152	0.077024	0.514933	0.84938	0.446296	0.466263	0.862243	0.892563
G1(+/-)	0.269535	0.462516	0.87452	0	0.538218	0.919624	0.494488	0.467534	0.917328	0.872384
G2(+/-)	0.48037	0.197521	0.881663	0	0.421724	0.864374	0.436379	0.478189	0.882061	0.883246
G3(+/-)	0.558407	0.453816	0.418655	0.029003	0.592439	0.920981	0.485018	0.433777	0.928191	0.87003
G4(+/-)	0.460412	0.469809	0.788792	0	0.539214	0.792342	0.405181	0.449303	0.917888	0.893721
G5(+/-)	0.542123	0.433699	0.8523	0.037269	0.313838	0.854951	0.428169	0.459481	0.892322	0.854349
G6(+/-)	0.503332	0.45974	0.925796	0.107999	0.472891	0.399046	0.46591	0.481628	0.757479	0.854172
G7(+/-)	0.489092	0.498641	0.901486	0	0.568886	0.869988	0.320065	0.463536	0.90247	0.821258
G8(+/-)	0.474718	0.491761	0.904017	0.08554	0.482761	0.860365	0.490878	0.20447	0.908446	0.858928
G9(+/-)	0.510933	0.525838	0.886594	0.015735	0.544708	0.915439	0.45121	0.442537	0.4477	0.902918
G10(+/-)	0.520382	0.405724	0.867028	0.027686	0.576888	0.941134	0.44844	0.475925	0.914623	0.427202

Figure 5.1 Heterozygous knock-down data from yeast network

For every network, three experiments are generated from both E.Coli and Yeast gene regulatory networks: Heterozygous knock-down data, Null-mutants data and time series trajectories data. Heterozygous knock-down data contains the steady state levels for the wild-type and the heterozygous knock-down (a gene is reduced by half) strains for each gene; Null-mutants contains the steady state levels for the wild-type and the null-mutant (a gene is set to zero) strains; Time series trajectories data contains time courses of the network recovering from several external perturbations. The three kinds of data from one of the Yeast sub-networks (10 genes) are shown in Figures 5.1, 5.2 and 5.3.

strain	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
wt	0.529871	0.517426	0.818152	0.077024	0.514933	0.84938	0.446296	0.466263	0.862243	0.892563
G1(-/-)	0.010053	0.815176	0.845015	0.376999	0.248059	0.859406	0.474825	0.495988	0.641061	0.875929
G2(-/-)	0.510487	0.082249	0.942792	0.092006	0.501758	0.897611	0.426991	0.461354	0.915775	0.878408
G3(-/-)	0.794319	0.533865	0	0	0.714275	0.919026	0.462578	0.507089	0.907252	0.807122
G4(-/-)	0.533501	0.476569	0.85278	0	0.577055	0.900193	0.509145	0.441179	0.86297	0.901971
G5(-/-)	0.54996	0.487726	0.945599	0	0	0.858854	0.849638	0.458988	0.906015	0.9125
G6(-/-)	0.460369	0.501849	0.917398	0.239301	0.522597	0	0.545508	0.434083	0.751722	0.946483
G7(-/-)	0.427145	0.492922	0.884602	0.002402	0.520915	0.876362	0	0.850223	0.832538	0.917185
G8(-/-)	0.513985	0.54441	0.859676	0	0.554409	0.879938	0.443015	0.031132	0.903685	0.888021
G9(-/-)	0.533942	0.537358	0.896238	0.049496	0.579922	0.945879	0.417966	0.460969	0	0.919536
G10(-/-)	0.522307	0.487185	0.974815	0.013396	0.526814	0.868488	0.81827	0.420615	0.851067	0.04081

Figure 5.2 Null-mutant knock-out data from yeast network

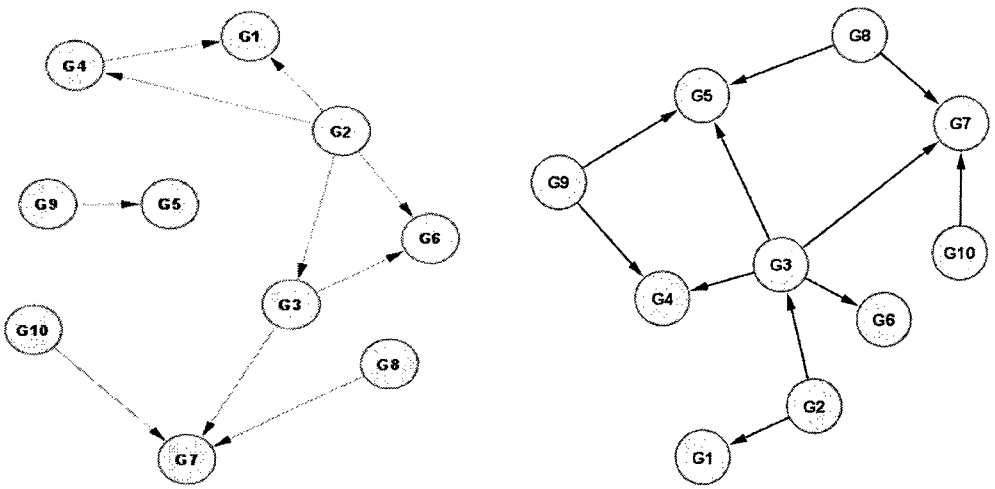
Time	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
0	0.827248	0.725743	0.389413	0.006674	0.156499	0.235751	0.673909	0.330327	0.740032	0.703382
10	0.654252	0.647066	0.50889	0.019373	0.408929	0.462383	0.58482	0.361024	0.796993	0.71666
20	0.544903	0.599011	0.555778	0.062856	0.322246	0.576507	0.566352	0.379408	0.688547	0.769349
30	0.587448	0.580601	0.556083	0.003298	0.31155	0.708266	0.595944	0.387101	0.77206	0.8245
40	0.578262	0.544896	0.687272	0.009736	0.469409	0.724303	0.572994	0.387153	0.856956	0.823019
50	0.538142	0.550524	0.611639	0.015817	0.492165	0.851393	0.580864	0.300948	0.783365	0.803734
60	0.566914	0.530258	0.72554	0.087896	0.492601	0.881654	0.513531	0.433371	0.846601	0.811533
70	0.591807	0.541304	0.843607	0.008827	0.497733	0.88429	0.581799	0.471869	0.833021	0.773539
80	0.497869	0.520274	0.861386	0	0.541297	0.869937	0.50844	0.469636	0.899533	0.881152
90	0.490107	0.455365	0.812616	0	0.612829	0.851405	0.480543	0.449487	0.868628	0.797148
100	0.501994	0.567246	0.795296	0	0.583852	0.865882	0.530948	0.397863	0.85477	0.817933
110	0.509833	0.522452	0.796389	0	0.477467	0.880751	0.514849	0.358707	0.912258	0.851607
120	0.583105	0.404564	0.825172	0.037173	0.501665	0.855132	0.511895	0.492986	0.898491	0.882451
130	0.412013	0.434665	0.808568	0	0.433559	0.961338	0.494141	0.408143	0.863247	0.851573
140	0.534837	0.518527	0.860758	0.041141	0.569422	0.85369	0.456817	0.414821	0.762971	0.81529
150	0.503571	0.452819	0.901765	0.082637	0.512369	0.829728	0.562962	0.35326	0.872778	0.900087
160	0.482243	0.49144	0.960037	0.024032	0.503758	0.98344	0.453367	0.475796	0.941885	0.868355
170	0.472727	0.486258	0.888965	0	0.491788	0.839797	0.43665	0.394543	0.832124	0.930392
180	0.467733	0.514798	0.841251	0.081116	0.479153	0.917861	0.445116	0.528315	0.817893	0.883831
190	0.453636	0.501861	0.773469	0.019636	0.476226	0.865831	0.552368	0.470282	0.78797	0.893402
200	0.567812	0.505348	0.882146	0	0.504835	0.903033	0.488191	0.444352	1	0.894393

Figure 5.3 Time series trajectories data from one of the perturbations in yeast network with 10 genes and 21 time points

The approaches given in Chapter IV are used to infer GRNs from the synthetic data sets provided by Marbach, D [135]. For each gene from the given dataset, we take the gene expression value of wild-type as reference and calculate the relative change ratios of

expression values compared to reference. If the change ratio is more than 30%, we select this gene as potential key gene and assume it wills play critical roles in our network. If the change ratio is less than 5%, it will be considered as noise and ignored from our potential regulatory genes list. Then DBN is used to infer GRN based on above information.

In Figure 5.4, the inferred gene regulatory network is shown in size of 10 (E.Coli). We can clearly see that there are 7 correct matching edges represented in green lines, compared to true network. The predicted gene regulatory network in size of 50 (Yeast) is shown in Figure 5.5 and the matching network is shown in Figure 5.6. There are 52 correct edges inferred out of total 77 edges in true network.



(a) (b)

Figure 5.4 Gene regulatory networks (Size 10) from E.Coli.
(a) Predicted network (b) True network

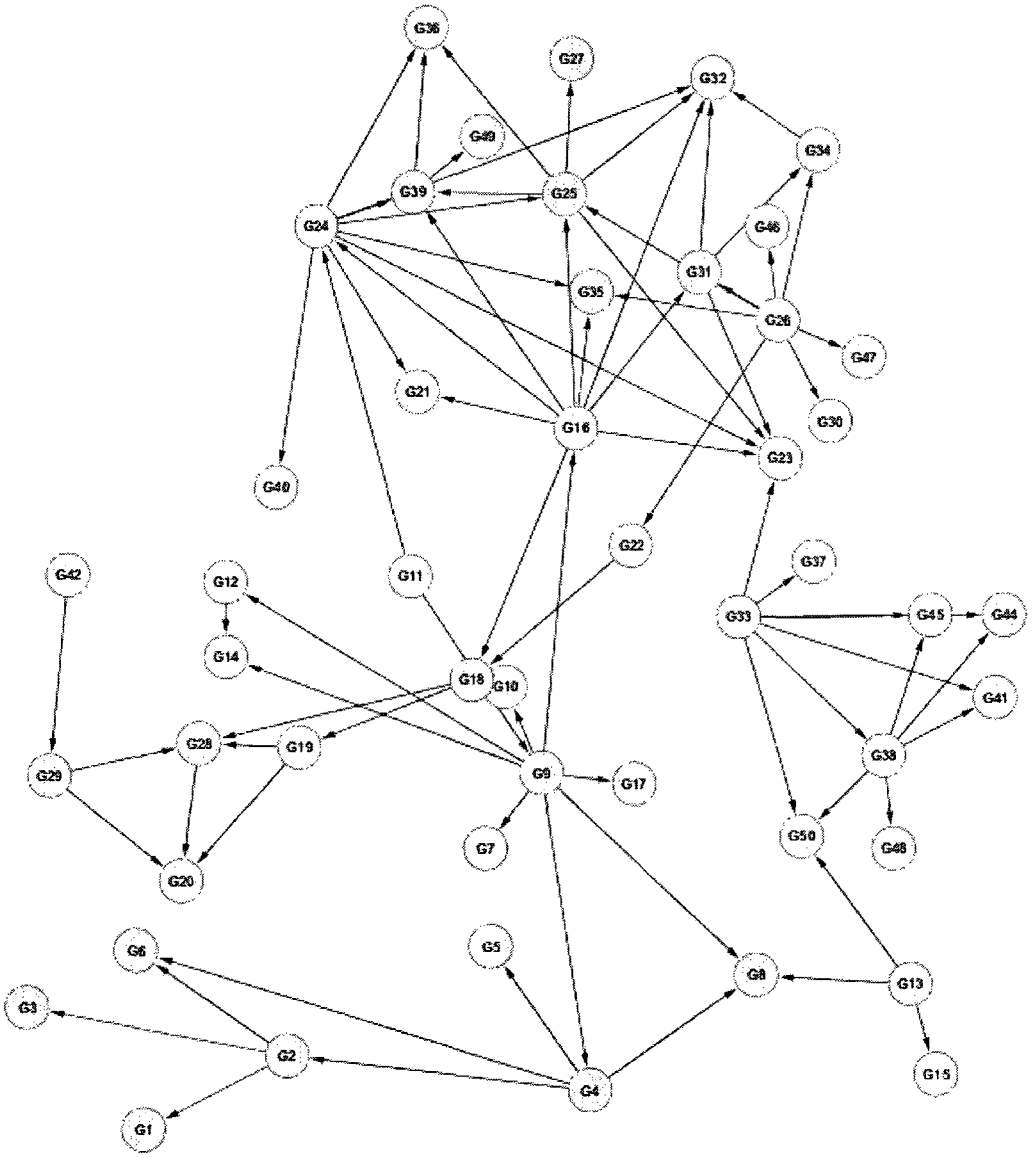


Figure 5.5 Inferred gene regulatory networks from Yeast (size 50)

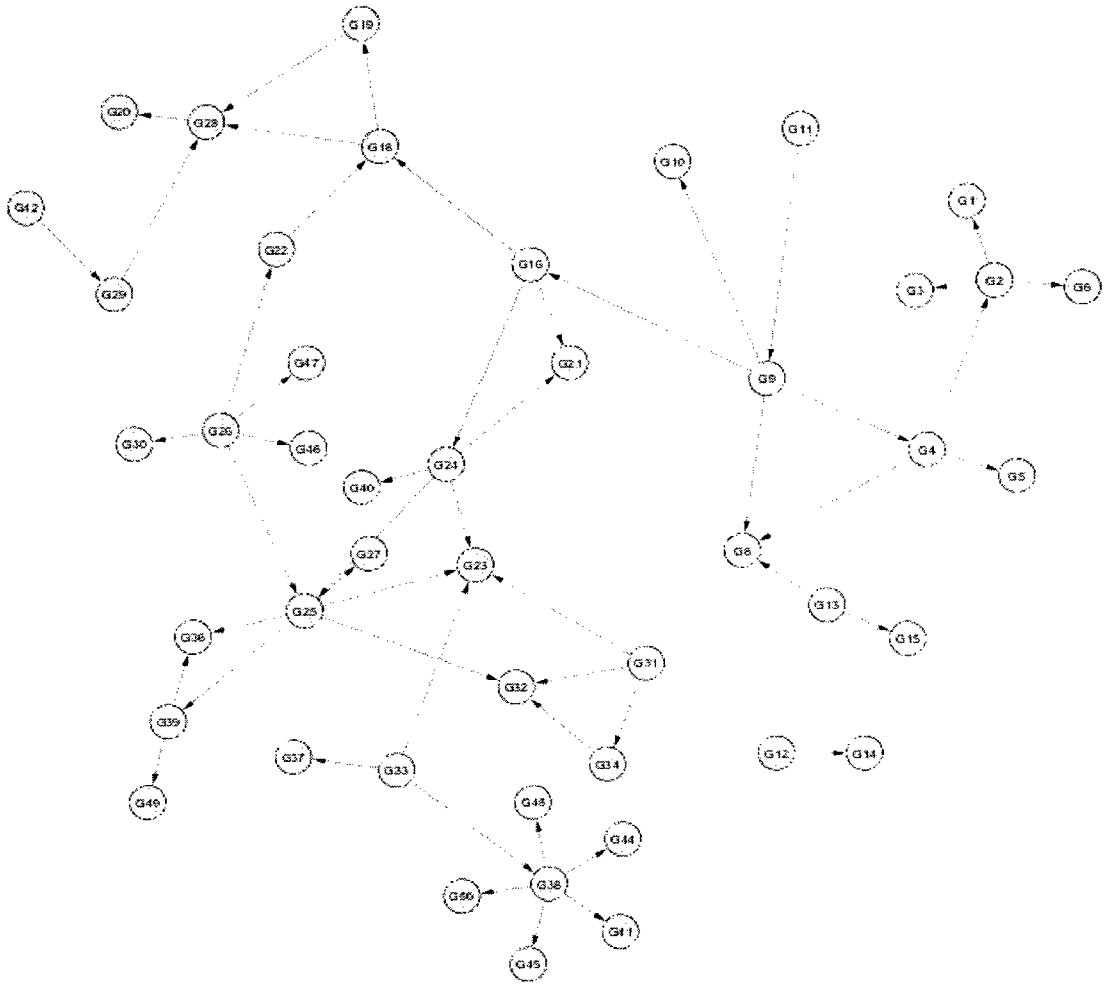


Figure 5.6 Matching network of prediction with true network (size 50)

Here, we use the following criterions provided by DREAM [132] to evaluate our results. **AUROC:** Area under the receiver operating characteristic (ROC) curve. **AUPR:** Area under the precision-recall curve. Precision is a measure of fidelity whereas recall is a measure of completeness. **Overall P-value:** The geometric mean of the ***n*** individual p-values, computed as $(p1 * p2 * \ldots pn) \wedge (1/n)$. **Overall AUROC P-value:** The geometric mean of the five AUROC p-values (Ecoli1, Ecoli2, Yeast1, Yeast2, Yeast3). **Overall**

AUPR P-value: The geometric mean of the five AUPR p-values (Ecoli1, Ecoli2, Yeast1, Yeast2, Yeast3). Larger scores indicate greater statistical significance of the prediction.

Table 5.1 AUC, AUROC, P_AUC and P_AUOC values for E.Coli1 and Yeast1

Size	E.Coli 10	Yeast 10	E.Coli 50	Yeast 50	E.Coli 100	Yeast 100
AUPR	5.43e-001	7.71e-001	6.71e-001	4.86e-001	1.45e-002	1.55e-002
AUROC	7.94e-001	9.44e-001	8.62e-001	8.35e-001	5.21e-001	4.61e-001
P_AUPR	1.34e-004	2.09e-006	8.57e-055	3.91e-039	2.27e-001	8.91e-001
P_AUROC	5.47e-004	1.29e-006	3.19e-020	4.64e-018	2.02e-001	9.60e-001
Overall AUPR	1.085e-04		2.539e-46		4.833e-03	
Overall AUROC	2.103e-04		8.192e-18		2.128e-02	

In Table 5.1, our results from different size of E.Coli and Yeast data sets are evaluated by AUPR, AUPR p-values, AUROC and AUROC P-value (P_AURP).

5.2 Drosophila Muscle Development Network Data

In this part, we use a real biological time series data set (Drosophila genes network from Drosophila Interaction Database [133]) to compare the PBN and DBN for modeling gene regulatory networks [136, 137]. The raw data is preprocessed in the same way as given in [138]. There are 4028 gene samples with 74 time points available in *Drosophila melanogaster* genes network through the four stages of the life cycle: embryonic, larval, pupal and adulthood [136]. An example network of drosophila muscle development is given in [138], in which muscle-specific protein 300 (Msp-300) is treated as hub gene in their inferred network. We use a different subset of the genes which participate in the development of muscle [139]. Particularly, Mlp84B and other genes which contribute to larval somatic muscle development are used to infer gene regulatory networks.

The *D. melanogaster* gene Muscle LIM protein at 84B (abbreviated as Mlp84B) has

also been known in FlyBase as Lim3. It encodes a product with putative protein binding involved in myogenesis which is a component of the cytoplasm. It is expressed in the embryo (larval somatic muscle, larval visceral muscle, muscle attachment site, pharyngeal muscle and two other listed tissues). Table 5.2 shows the scores of Mlp84B interacting with other related genes [133].

Table 5.2 The interactions and scores of Mlp84B with other genes

High Confidence	Scores	Other interactions	Scores
CG10722	0.5642	Cdk7	0.3569
CG13501	0.9005	Impe1	0.1108
CG17440	0.5811	Pfk	0.3155
CG7046	0.6626	TfIIB	0.2436
CG7447	0.5411	Stck	0.2523
CG11115 (Ssl1)	0.7917	tup	0.1094

Here, we first select 12 genes to infer gene regulatory networks using PBN and DBN. The reconstructed networks are shown in Figure 5.7(a) and Figure 5.7(b), respectively. There exist 18 interactions totally among this small larval somatic muscle network [133]. 10 and 12 interactions in the networks have been successful identified. Most interactions between Mlp84B and genes with high confidence have been referred.

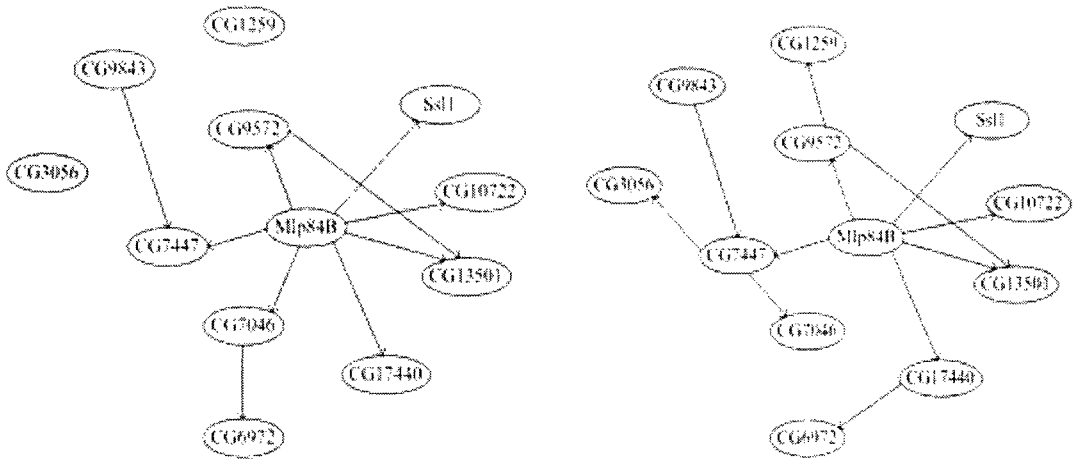


Figure 5.7 Drosophila larval somatic muscle development network. (a) The genetic network inferred by PBN. (b) The genetic network inferred by DBN

More comparison results of PBN(n, e) and DBN(n, e) are given in Table 3.4, where n is the number of nodes (genes) in network and e the number of edges (interactions) among the nodes. PBN(30,60) means that there are 30 nodes and 60 edges in that PBN simulation. We use the benchmark measures *recall* and *precision* to evaluate the performances of inference algorithms for PBN and DBN. Here, *recall* is defined as $Ce/(Ce+Me)$ and *precision* as $Ce/(Ce+Fe)$, where Ce is the number of correct edges, Me is the total number of missed edges (miss errors), and Fe is the number of false alarm errors. Miss error is defined as the connection between genes that exists in real networks, but the inference algorithms miss or make wrong orientations. False alarm error is the connection that the inference algorithms create but does not exist in real networks. The calculation of *recall*, *precision* and selection of subset genes in network are based on the existing gene interactions and network diagram in Drosophila genes network [133].

If more genes are selected for inferring GRN, the network contains more edges and it is challenging to identify in the limited number of interactions among genes. Thus, both miss errors and false alarm errors may increase, which results in lower inferring accuracy, if a larger subset of genes is selected for constructing GRN. For each combination of genes/edges in genes network, we run the same case for five times and get the average values of correct edges, miss errors and false alarm errors, respectively. The accuracy of *recall* and *precision* are also given. The results are shown in Table 5.3.

Table 5.3 Comparison of PBN and DBN methods using different sample networks

	Miss errors *Me*			False alarm errors *Fe*			Correct edges *Ce*			Accuracy (%) (*R*, *P*)		Time(s) *T*
	min	max	avg	min	max	avg	min	max	avg	recall	precision	avg
PBN(12,18)	2	9	6.4	0	4	2.4	6	9	7.8	54.9	76.5	13.2
PBN(20,35)	12	22	16.8	3	6	4.8	11	15	13.6	44.7	73.9	19.7
PBN(30,60)	33	41	36.0	7	10	8.0	17	20	18.4	33.8	69.6	27.9
PBN(40,80)	48	63	55.4	4	6	5.6	18	22	19.6	26.1	77.8	39.2
DBN(12,18)	3	8	5.8	1	3	2.2	9	11	10.4	64.2	82.5	20.1
DBN(20,35)	13	17	15.2	4	7	5.4	14	18	16.8	52.5	75.7	36.0
DBN(30,60)	30	39	33.6	11	15	12.6	24	30	20.2	37.5	61.6	50.6
DBN(40,80)	46	57	51.2	5	9	7.4	28	34	22.8	30.8	75.5	87.6

The results show that PBN method can reduce the computational complexity, false alarm errors significantly, while DBN method can give better accuracy of deriving genetic network interactions, but DBN is more time-consuming than PBN.

5.3 Yeast Cell Cycle Data

The gene microarray data we used is from Spellman *et al.* [134]. The Spellman experiment was chosen because it provides a comprehensive series of gene expression datasets for yeast cell cycle. Four time series expression datasets were generated using four different cell synchronization methods: Cdc15, Cdc28, alpha-factor and elutriation

with 24, 17, 18 and 14 time points respectively (Table 5.4). The alpha-factor dataset contained more time points than Cdc28 and Elutriation datasets with fewer missing values than Cdc15. Therefore, we choose to use time series expression data from alpha-factor method to infer the yeast cell cycle gene regulatory network. In our previous work [139], probabilistic Boolean network and dynamic Bayesian network have been compared using the ***Drosophila melanogaster*** finding dynamic Bayesian network analysis outperformed probabilistic Boolean network analysis. Here, we focused on Bayesian network and dynamic Bayesian network analysis. Both Zou's time-lags DBN implementation [37] and Chen's Bayesian network R package [140] were carried out to infer gene regulatory networks, and then the results inferred by the two different methods were compared.

Table 5.4 Gene expression data from four methods in yeast cell cycle

Method	Sample Frequency	Cell Cycle length	Time points	Start	End
Cdc15	Every 20 min for 1 hr, every 10 min for 3 hr, every 20 min for the final hr	112m	24	10m	290m
Cdc28	Every 10 min	85m	17	0m	160m
Alpha	Every 7 min	64m	18	0m	119m
Elutriation	Every 30 min	-	14	0m	390m

From previously published work, some transcription factors have been identified which play very important roles in regulating a small set of yeast genes with cell-cycle dependent expression. These genes include *Mbp1*, *Swi4*, *Swi6*, *Mcm1*, *Fkh1*, *Fkh2*, *Ndd1*, etc [141]. Among them, *Ndd1* and *Mcm1* are essential for yeast cell survival. There were a total of 2467 genes in the alpha-factor experiment. For simplicity, we choose 36 genes related to these transcriptional factors in inferring the gene regulatory network, a few of

which are described in Table 5.5.

Table 5.5 Descriptions of transcription factors and genes related to the yeast cell cycle process

ORF	Gene Name	Description
YIL131C	*FKH1*	negatively regulates transcriptional elongation
YNL068C	*FKH2*	substrate of the Cdc28p/Clb5p kinase
YAL040C	*CLN3*	Cyclin,G1/S-specific
YGR108W	*CLB1*	Cyclin,G2/M-specific
YGR109C	*CLB6*	Cyclin, B-type
YMR043W	*MCM1*	Transcription factor of the MADS box family
YLR182W	*SWI6*	Transcription factor, subunit of SBF and MBF factors
YBR160W	*CDC28*	Cyclin-dependent protein kinase
YOR372C	*Ndd1*	activates the expression of a set of late-S-phase-specific genes
YDL056W	*MBP1*	Transcription factor, subunit of the MBF factor
YDR054C	*CDC34*	E2 ubiquitin-Conjugating enzyme
YDR146C	*SWI5*	Transcription factor
YER111C	*SWI4*	Transcription factor, subunit of SBF factor

Using the DBN method, *SWI4* is obviously a hub gene during the process of yeast cell cycle consistent with known interactions. For example, *SWI4* has been shown to regulate *MBP1*, and *NDD1* is believed to regulate *SWI4* according to [141, 142]. Some interactions found in the inferred network have not been previously described in the literature or existing pathway databases. For example, the potential pairing of *MCM1* with *MFA1* has not been shown before this work. We compared both inferring methods with known gene relationships found in BioGrid interaction database [143], KEGG database [144] and other literature [141, 142]. The DBN and BN inferred gene regulatory networks are shown in Figures 5.8 and Figure 5.9.

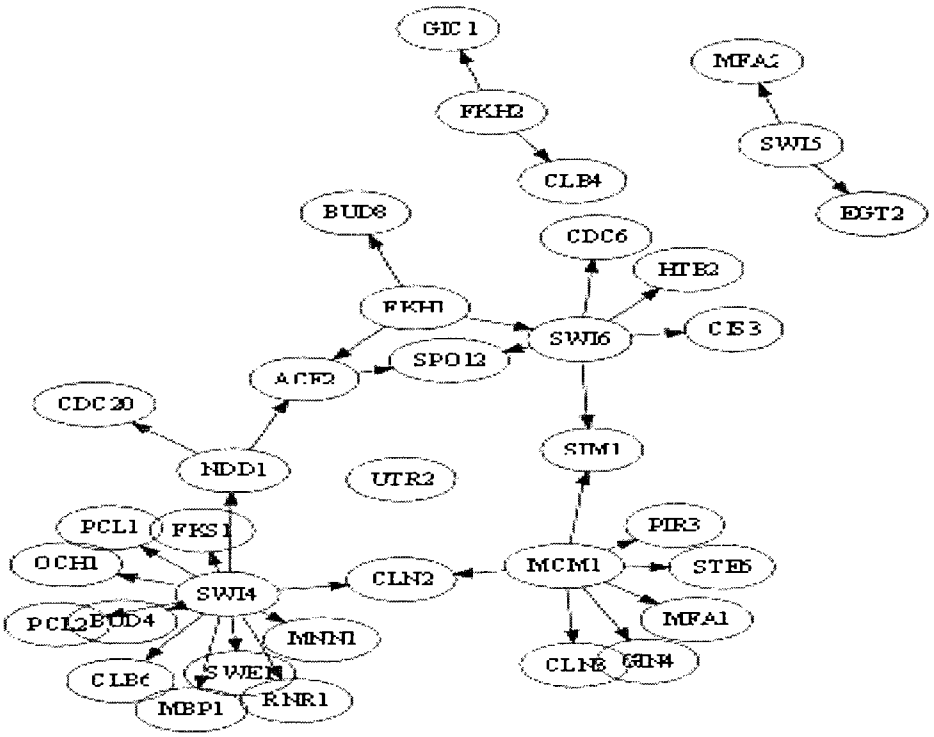


Figure 5.8 Yeast cell cycle gene regulatory network inferred by DBN

The specificity and sensitivity of each analysis are shown in Table 5.6. It can be seen from the table that the specificity of both methods increases when known transcriptional factors are incorporated, as does the sensitivity. The time delayed dynamic Bayesian network method has higher specificity and sensitivity compared to Bayesian network method.

Table 5.6 Comparison of specificity and sensitivity of two inferring methods

	time delayed DBN		Bayesian R package	
	Gene expression data only	Gene expression data with TF binding site	Gene expression data only	Gene expression data with TF binding site
Correct edges	6	14	4	9
Inferred edges	29	33	24	31
Specificity	21%	42%	17%	29%
Sensitivity	16%	27%	10%	22%

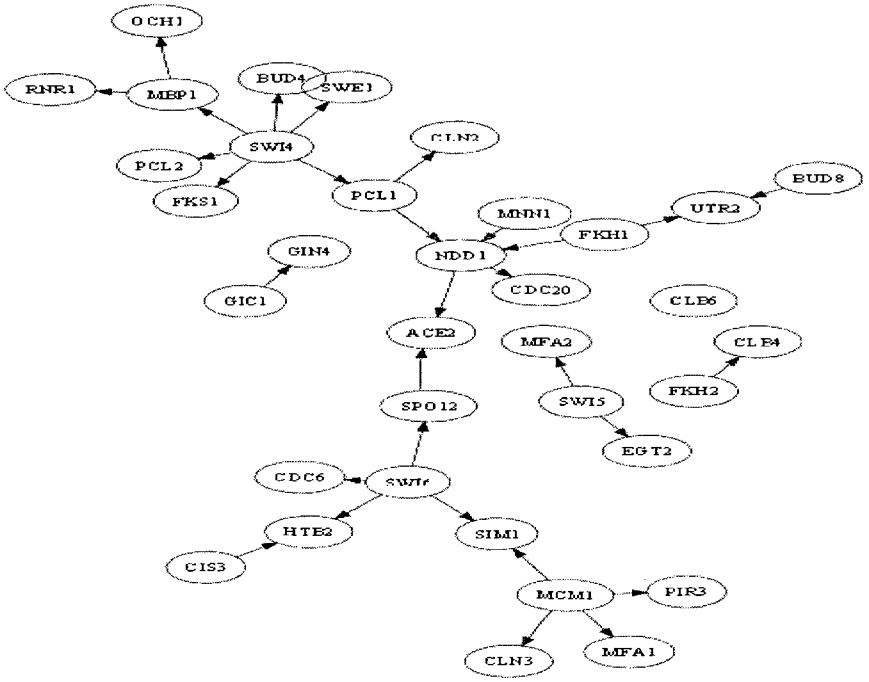


Figure 5.9 Yeast cell cycle gene regulatory network inferred by PBN

5.4 Fish Ovary Data

Sex hormone metabolism in fish ovaries provides a highly complex system within which fundamental properties of control and organization of networks can be investigated. The hypothalamus-pituitary-gonad axis in which sex hormone metabolism occurs is highly conserved from humans to fish. The fungicide ketoconazole (KTC) inhibits metabolism converting cholesterol into sex hormones in isolated ovary tissues but does not affect hormone levels in fish where the hypothalamus and pituitary exert global control over local ovary metabolic inhibition.

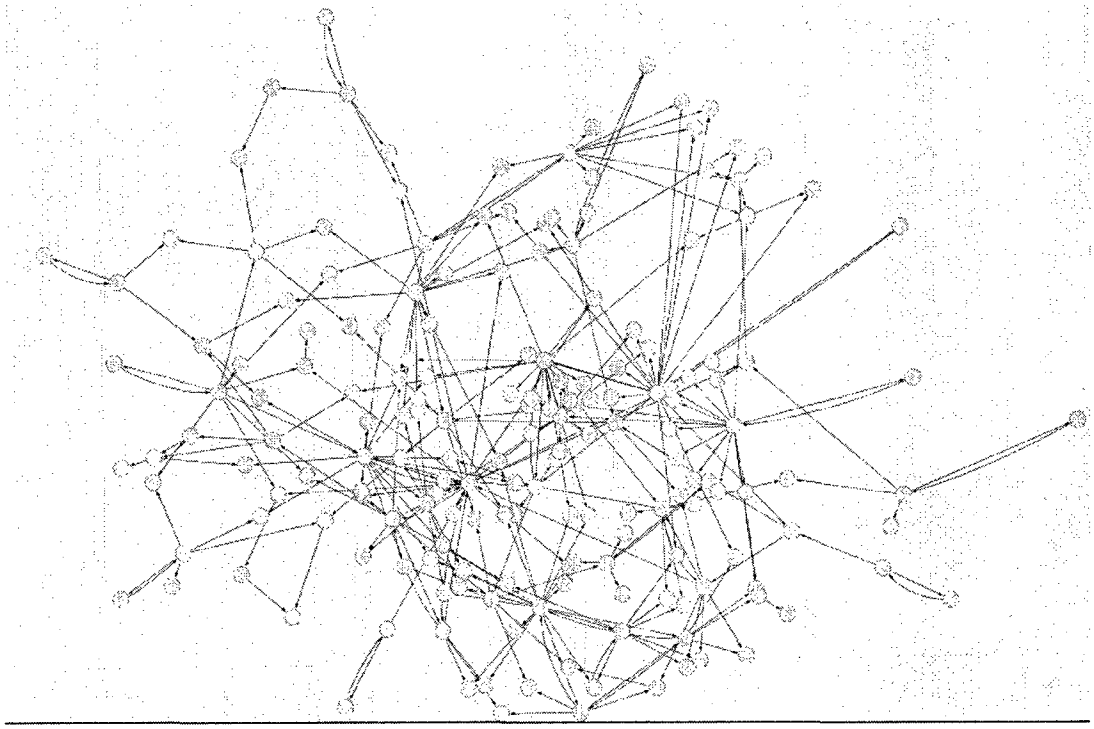


Figure 5.10 Gene regulatory network inferred from fish ovary data

A fish ovary gene regulatory network has been inferred from genes which are affected over time by ketoconazole in isolated ovary slices using a Dynamic Bayesian Network algorithm. Here, the ketoconazole data that we used is in KTC_TNT_0-150m condition, which includes 319 genes and 11 time points (first time point is control). The inferred GRN includes 329 edges and the running time is 84 hours and 41 minutes (DELL Precision workstation T3400 with 4 CPUs and 4GB memory). Figure 5.10 shows the reconstructed fish ovary GRN, which is visualized by Cytoscape.

The network is scale free with genes for transcription factors, small molecule metabolism, ion channels and receptors as hubs. The recovered GRN suggests potential feedback loops and transcription factor interactions with the critical genes StAR (transports cholesterol into mitochondria), P450scc (1rst metabolic step), and cyp19a

(final conversion to estradiol) which may play roles in dynamic regulation of steroid metabolism. One important hub in the GRN is Low Density Lipoprotein Receptor which transports cholesterol/proteins into cells (Figure 5.11). Several modules or subcircuits in other gene regulatory networks have been identified that encode specific logic commands controlling gene expression, cell behavior and development suggesting similar motifs.

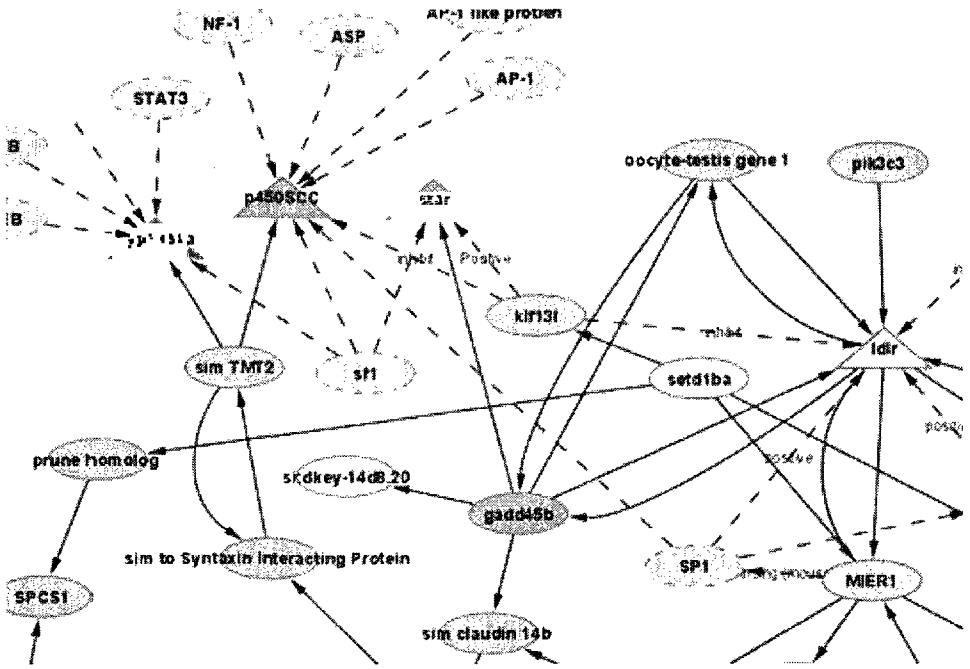


Figure 5.11 Subnetwork involving several key steroidogenesis genes.

In Figure 5.11, the color in gold represents transcription factor, yellow means receptor or channel, triangle means steroidogenesis gene and grey is gadd45b. Solid lines were inferred from expression data. Dashed lines are known interactions added post hoc. The direction of arrows indicates direction of influence. The above information is provided by Environmental Laboratory at the US Army Engineer Research and Development Center (ERDC).

CHAPTER VI

GRN INFERENCE FROM OUR NEW MODEL

Now we have already built the model, before applying any new model to inferring gene regulatory networks from real biological microarray data, we need to evaluate the performance of the model using synthetic data, or so called "*in silico*" data. Two types of synthetic data are used to test our model: one is generated by GeneSim, the other one is simulated by GeneNetWeaver. We will discuss them respectively.

6.1 Model Validation

Advances in high-throughput technologies, such as DNA microarray, have spurred the development of a lot of computational methods for modeling gene regulatory networks. However, the strengths, weaknesses and relative performance of different methods remain poorly understood [145]. Since most of gene regulatory networks are kept unknown or incomplete, it causes an inherent difficulty in evaluating the performance of gene network inference methods, because it is nearly impossible to systematically validate the predictions of unknown interactions in vivo, Figure 6.1A shows the model validation procedure for real biological data sets. Consequently, synthetic data from *in silico* (computer generated) gene networks often becomes the only possibility for systematic performance assessment. Figure 6.1B shows that performance evaluations become possible for *in silico* benchmark, because in simulation the structures of gene regulatory networks are known and fully controlled. This allows characterization of GRN inference methods for different types of data and levels of noise. In addition to performance assessment, *in silico* studies are of great relevance for optimal experimental design for subsequent real biological applications [146].

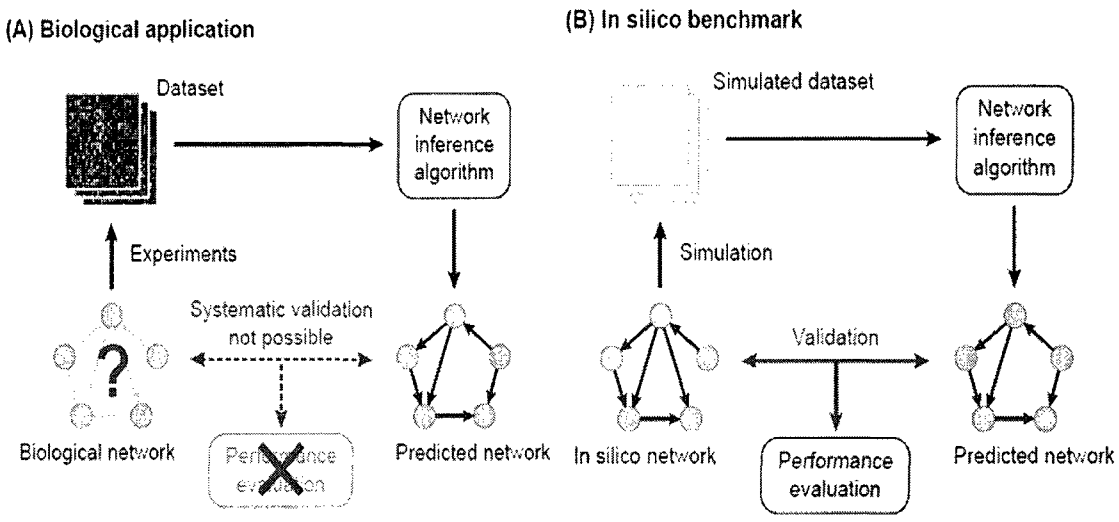


Figure 6.1 Biological application and in silico benchmark

The generated synthetic data needs to have biological meaningful. In recent years, many approaches are proposed for generating such *in silico* gene network structures, such as [31, 135, and 147].

6.2 Synthetic Data and Results

6.2.1 GeneSim

GeneSim was developed at Duke University as part of a project which tried to understand songbird singing behaviors. It is used in most of the GRN research in computational biology community. GeneSim models genetic regulatory pathways of arbitrary network structure and produces values for gene expression levels at discrete time-steps. Values are produced by a combination of two processes. First, values at each time step are updated by a simple stochastic process:

$$Y_{t+1} - Y_t = f(Y_t) = A(Y_t - T) + \varepsilon \qquad (6.1)$$

where Y_t is a vector representing the expression levels of all genes at time t . Second, expression levels are restricted by a floor and ceiling function to range from 0 to 100 (arbitrary units). Expression levels are initialized to random values uniformly sampled from this range. The matrix A represents the regulatory interactions in the simulated networks. More details about GeneSim are described in [34, 148]

We use GeneSim to simulate 10 different randomly generated genetic regulatory networks. Each of the networks has 20 genes; 8–12 of these genes have regulatory interactions with at least one other gene. A total of 100 interactions are present across all 10 networks, 60 of which are one-parent links, 34 are two-parent links, and six are three-parent links.

Then we compare the performance of our new model (based on state space model and Expectation-Maximization algorithms) with dynamic Bayesian network, the details are shown in Table 6.1. From the comparing results, we can see that both recall and precision values of our model and DBN increase as number of time points increases. The recall values of both models are very similar but precision values of our model are little lower than DBN, which means more correct edges are inferred as well as more wrong edges by our new model. The computational time of our new model is much better than DBN. The computational time of our model is around 19 times faster than DBN for the case of 25 time points, and the speedups are 16 for 50 time points and 15 for 100 time points, respectively. Figure 6.2 shows one example of inferred gene regulatory network, which has 20 genes and 19 interactions.

Table 6.1 Comparison of DBN and our model using GeneSim synthetic data

Number of time points	Dynamic Bayesian Network (DBN)			EM and Kalman Filter (EMKF)			Speedup
	Recall	Precision	Time (seconds)	Recall	Precision	Time (seconds)	
25	14% (14/100)	11% (14/127)	748.2	15% (15/100)	9% (15/166)	39.7	18.85
50	29% (29/100)	23% (29/126)	862.9	26% (26/100)	16% (26/162)	53.0	16.28
100	47% (47/100)	52% (47/90)	1391.0	45% (45/100)	27% (45/167)	90.6	15.35

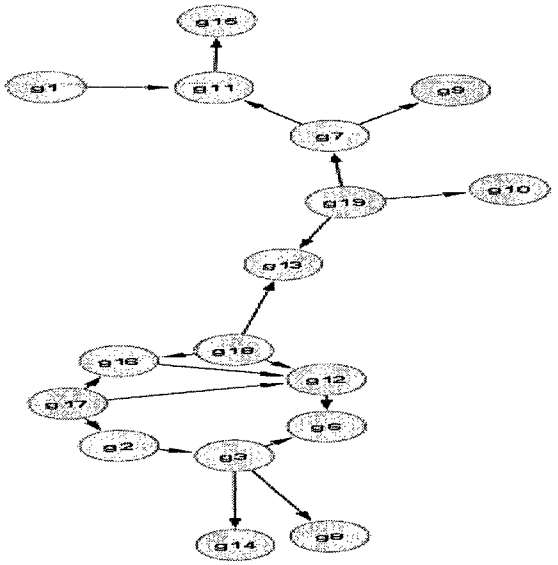


Figure 6.2 Inferred gene network (20 genes, 19 interactions)

After we get the connectivity matrices from our model, selection of cut-off will affect the recall and precision for the inferred gene regulatory networks. If we select higher value of cut-off, there will be less inferred edges, and vice versa. In the above results, we set the cut-off as a "fixed value", which is 0.65. We want to see how the performance of our model is different when different cut-off values are selected. Here we use different

cut-off values (range from 0.5 to 0.85) to compare the inferring accuracy and select the best performance based on each dataset (for time points 25, 50 and 100).

Table 6.2 Performance comparison for different cut-off values

	Cut-off (R means Recall, P means Precision)							
Time points	0.5		0.55		0.6		0.65	
	R	P	R	P	R	P	R	P
25	19% (19/100)	8.60% (19/221)	17% (17/100)	8.90% (17/192)	17% (17/100)	9.70% (17/175)	15% (15/100)	9.00% (15/166)
50	31% (31/100)	15.70% (31/197)	29% (29/100)	16.70% (29/174)	26% (26/100)	15.40% (26/169)	26% (26/100)	16.50% (26/162)
100	51% (51/100)	25.00% (51/204)	46% (46/100)	25.60% (46/180)	45% (45/100)	26.20% (45/172)	45% (45/100)	26.90% (45/167)
	Cut-off (R means Recall, P means Precision)							
Time points	0.7		0.75		0.8		0.85	
	R	P	R	P	R	P	R	P
25	11% (11/100)	7.90% (11/140)	10% (10/100)	7.80% (10/128)	6% (6/100)	6.10% (6/99)	4% (4/100)	6.60% (4/61)
50	23% (23/100)	15.00% (23/153)	18% (18/100)	12.90% (18/139)	17% (17/100)	15.90% (17/107)	11% (11/100)	16.10% (11/68)
100	43% (43/100)	32.60% (43/132)	41% (41/100)	36.70% (41/112)	29% (29/100)	31.20% (29/93)	45% (15/100)	28.80% (15/52)

From the results in Table 6.2, we can clearly see that: 1) for data with 25 time points, the cut-off value 0.6 gave us the best inference accuracy with respect to both recall and precision; 2) for data with 50 time points, 0.55 is the best choice; 3) for 100 time points, the recall with cut-off value=0.65 has higher value and 0.75 has better precision values. It actually cannot tell which cut-off is better for the third case. Because the recall will always decrease as the cut-off values increase, we only compare the precision values when cut-off values increase. The plots of precision and cut-off selection are shown in Figure 6.3, Figure 6.4 and Figure 6.5.

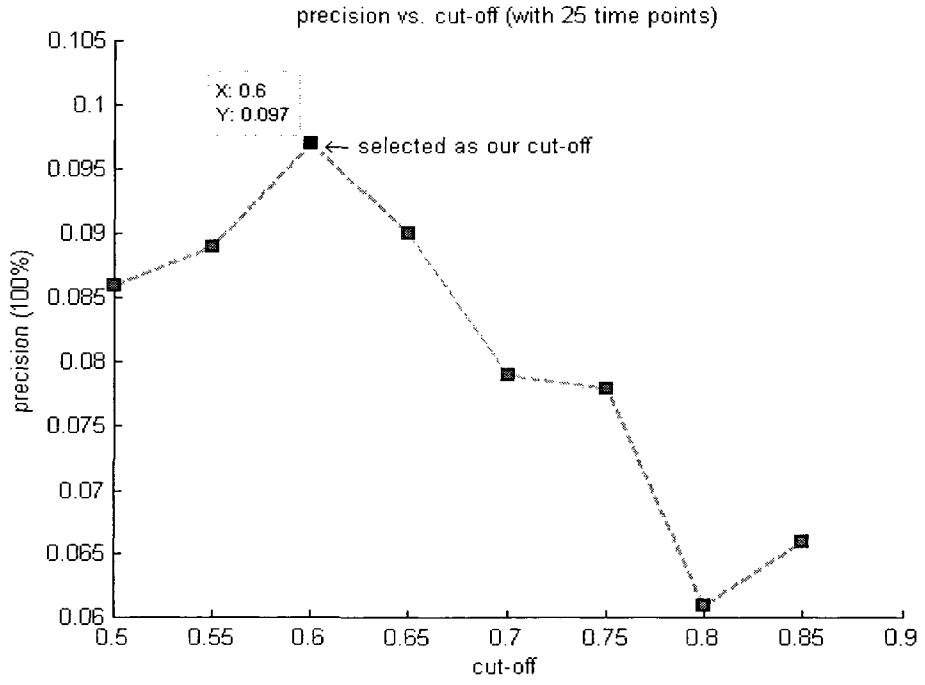


Figure 6.3 Precision vs. cut-off (25 time points)

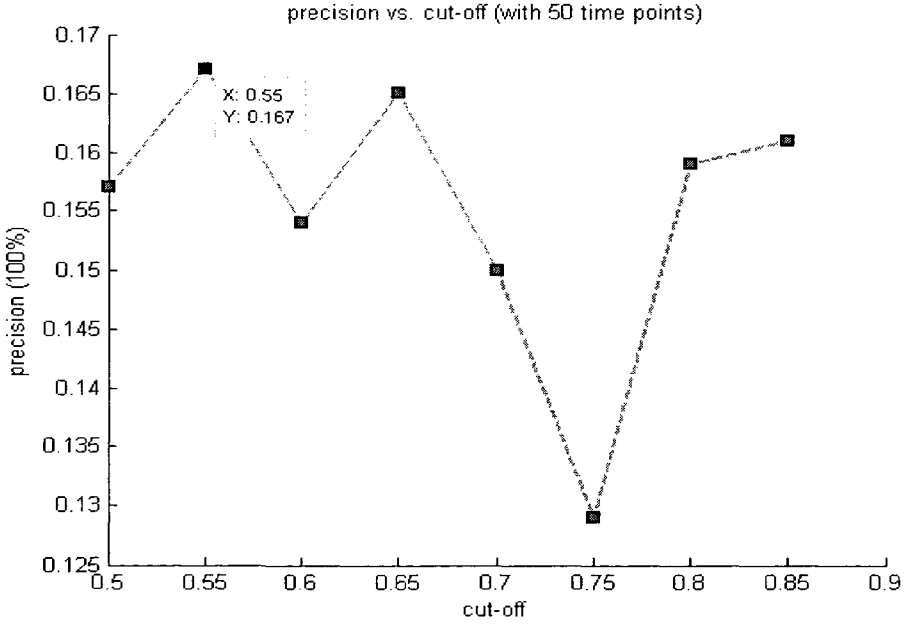


Figure 6.4 Precision vs. cut-off (50 time points)

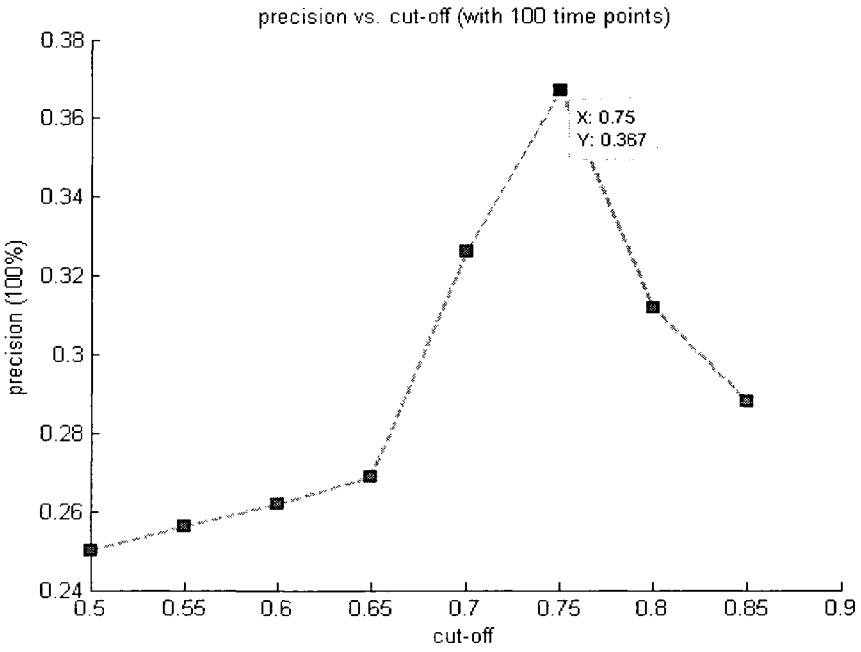


Figure 6.5 Precision vs. cut-off (100 time points)

6.2.2 GeneNetWeaver

GeneNetWeaver (GNW) is a tool for the automatic generation of *in silico* gene networks and reverse engineering benchmarks. GNW was used to generate the DREAM [132] *in silico* challenges, which are currently the most widely used gene network reverse engineering benchmark in the community. There are two different types of data supported by GNW: E.coli transcriptional regulatory network consists of 1502 nodes and 3587 edges, corresponding to the TF-gene interactions of RegulonDB. Yeast transcriptional regulatory network includes 4441 nodes and12873 edges. Note that this is a signed network and dynamical models will be initialized accordingly.

We choose different sizes of gene networks to test the scalability of our model. Networks with 100, 385, and 906 genes are selected from E.coli transcriptional regulatory network and inferred by our model, respectively. For size of 100 genes, there

are 179 inferred interactions, and the running time is 102 seconds; for 385 genes, 448 interactions are reconstructed and running time is 579 seconds; for 906 genes, the inferred network includes 1493 interactions and it took 2193 seconds to get the results. Figure 6.6, Figure 6.7 and Figure 6.8 shows the three inferred gene regulatory networks, respectively.

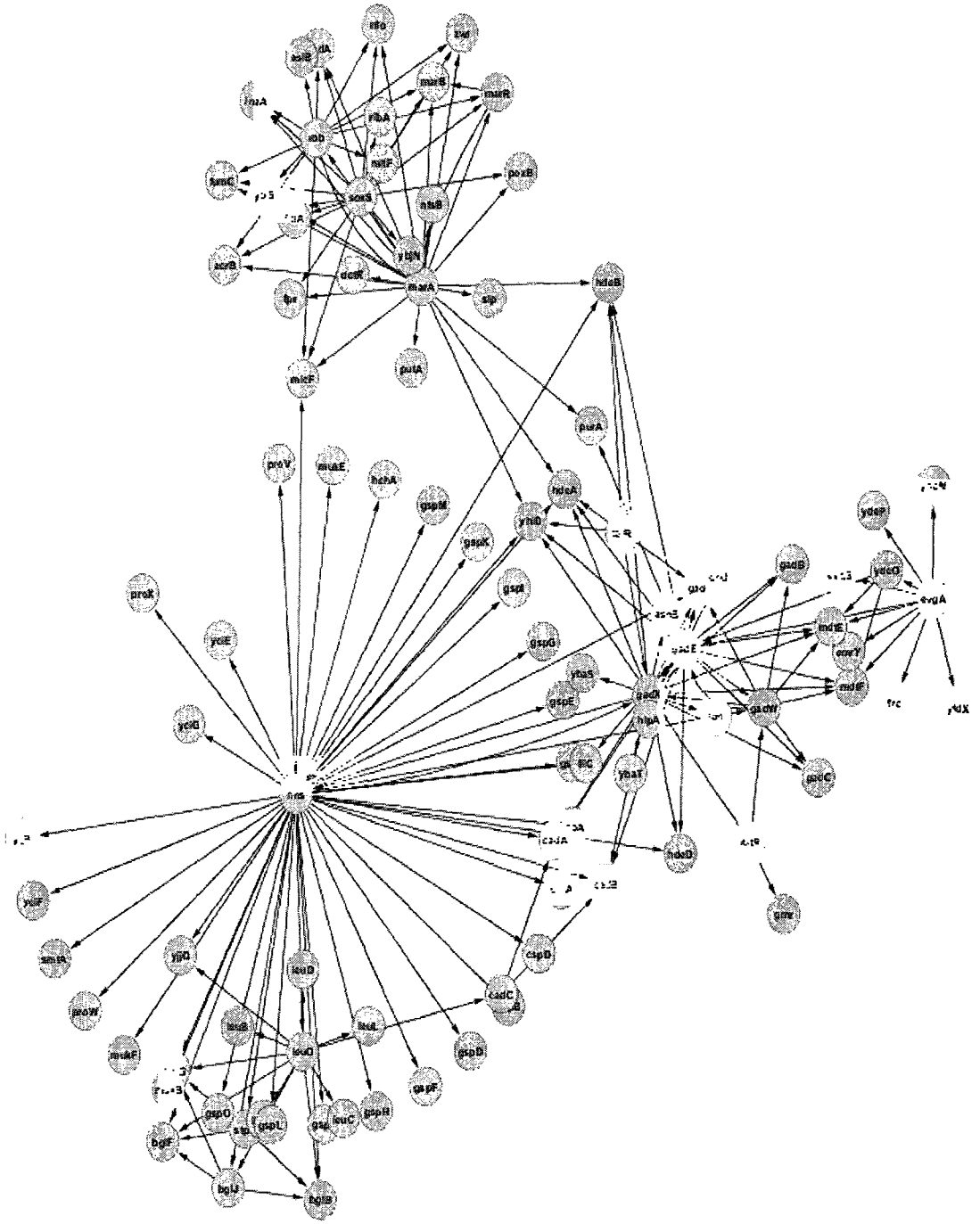


Figure 6.6 Inferred E.coli gene regulatory network (100 genes and 179 interactions)

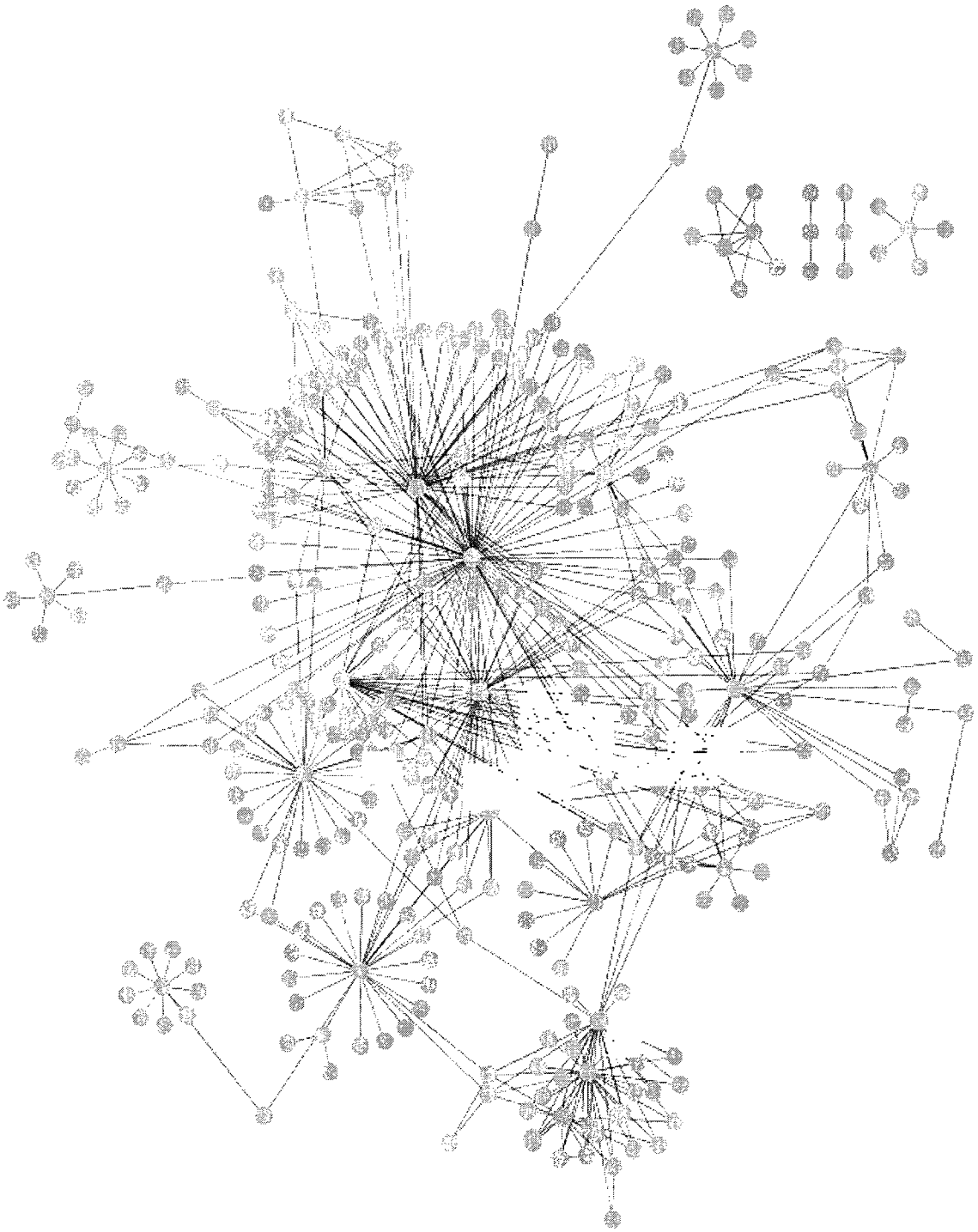


Figure 6.7 Inferred E.coli gene regulatory network (385 genes and 448 interactions)

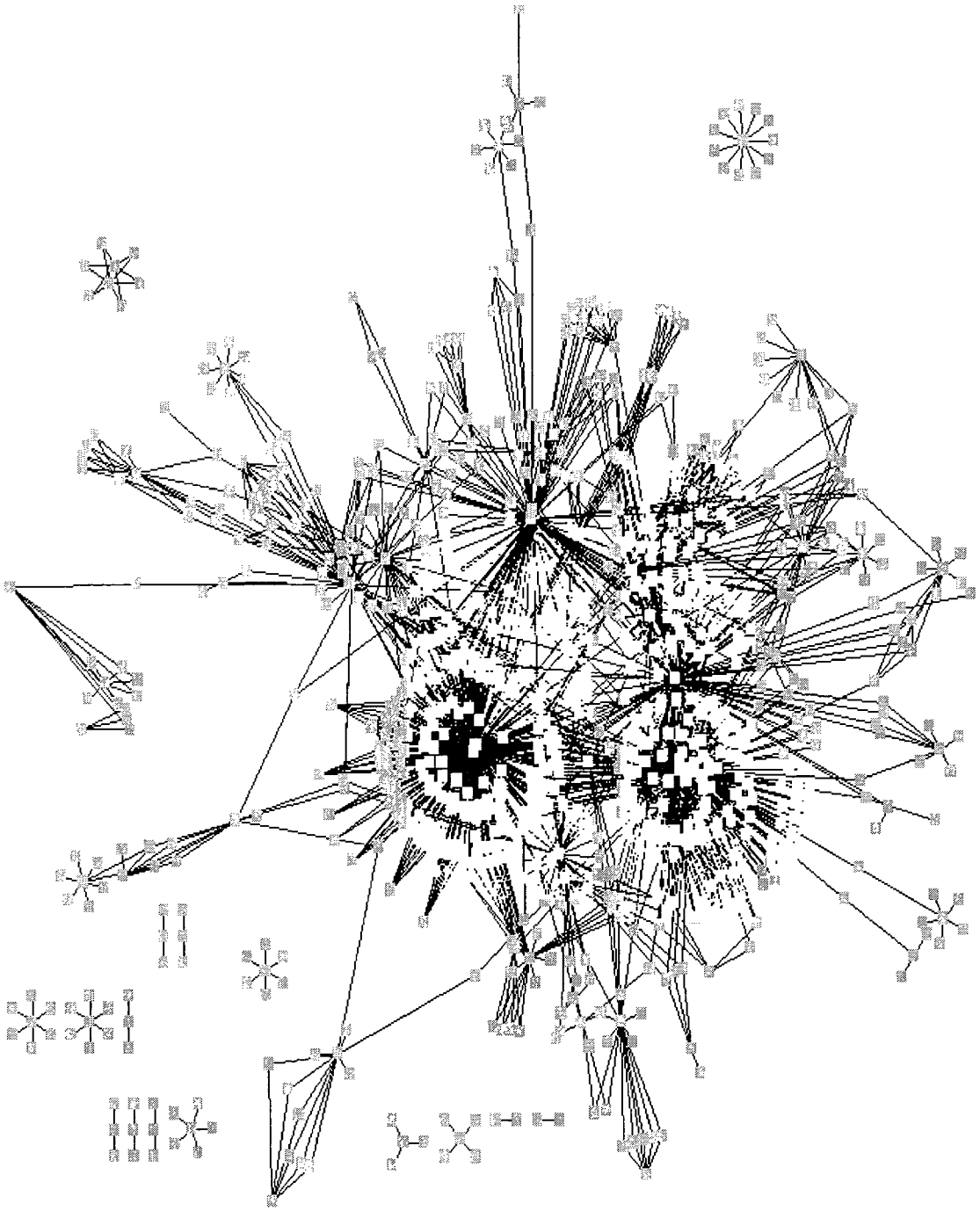


Figure 6.8 Inferred E.coli gene regulatory network (906 genes and 1493 interations)

6.3 Biological Data and Results

After model validation from synthetic data, we need to use real biological data to test our model. Yeast cell cycle data sets are widely used in computational biological community. It is well constructed in pathway databases, such as BioGrid [143] and KEGG [144] and many other publications. In this section, we reconstruct a gene regulatory network from yeast cell cycle pathway using the model that we proposed in Chapter 4.

The yeast cell cycle is an ordered series of events leading to replicating of cells. In eukaryotic cells, the cell cycle consists of two basic processes: DNA synthesis (S phase) and mitosis phase (M phase). During S phase double stranded DNA molecules are replicated to produce pairs of "sister chromatids", held together by proteins called cohesions. M phase consists of four sub-phases. Prophase is the first phase when chromosomes condense into compact structures. Metaphase is the next phase when chromosomes are aligned on the mid plane of the mitotic spindle. In anaphase, cohesions are degraded and finally in telophase, daughter nuclei forms and the cell begin to divide. S and M phases are separated in time by two gap phases (G1 and G2), constituting the generic cell cycle: G1-S-G2-M. Figure 6.9 shows the four phases of cell cycle in yeast (*Saccharomyces cerevisiae*).

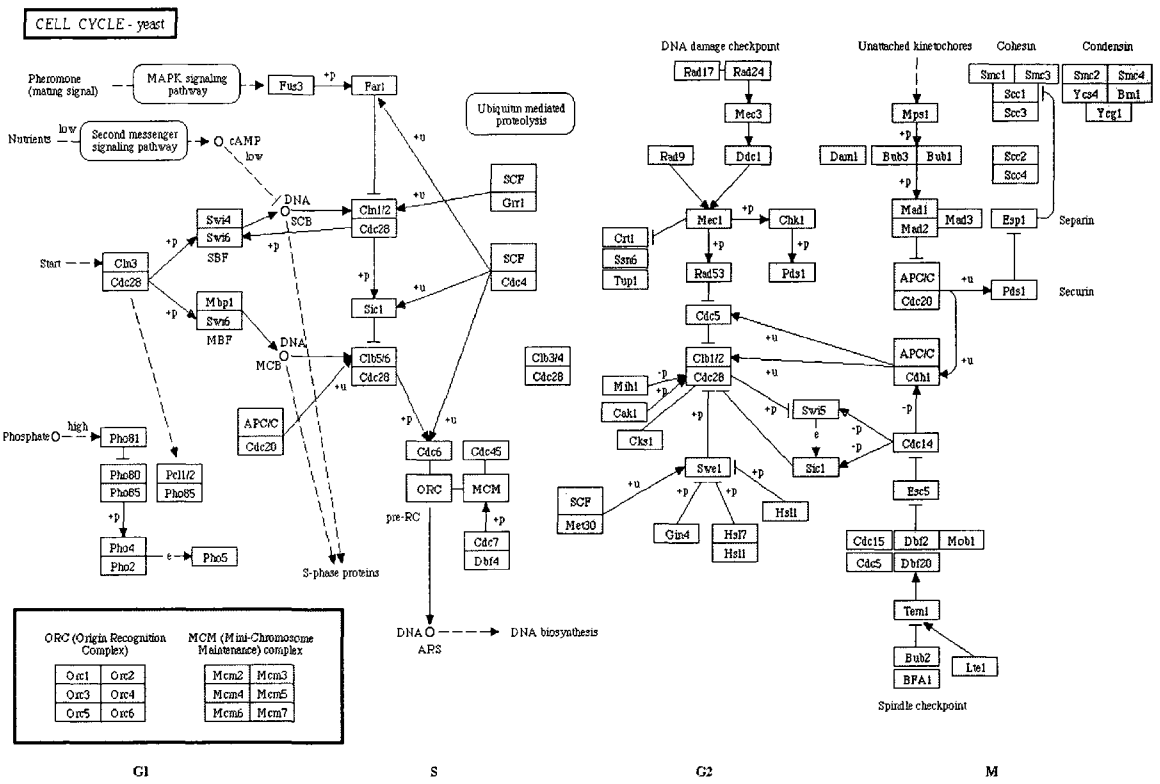
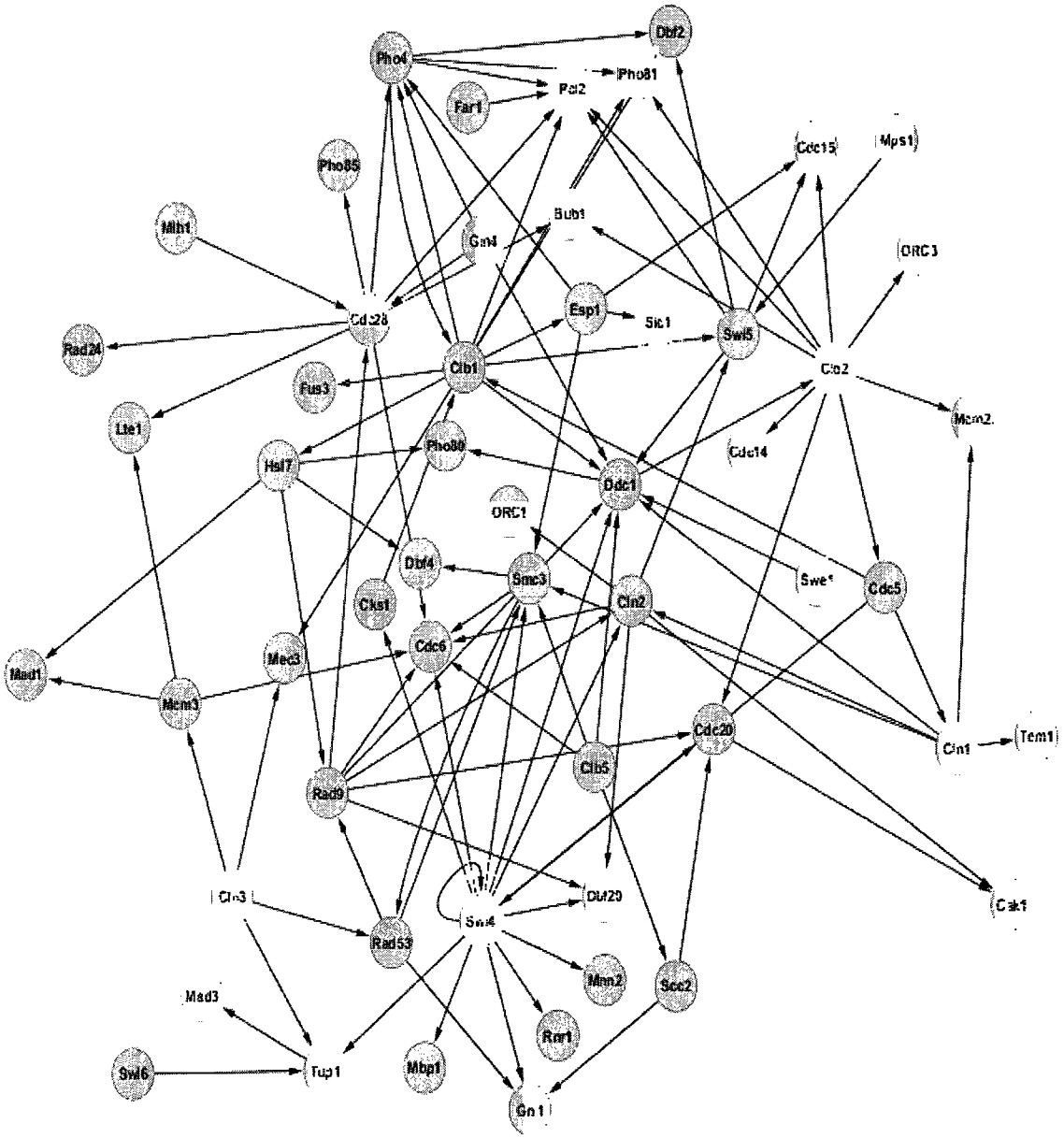


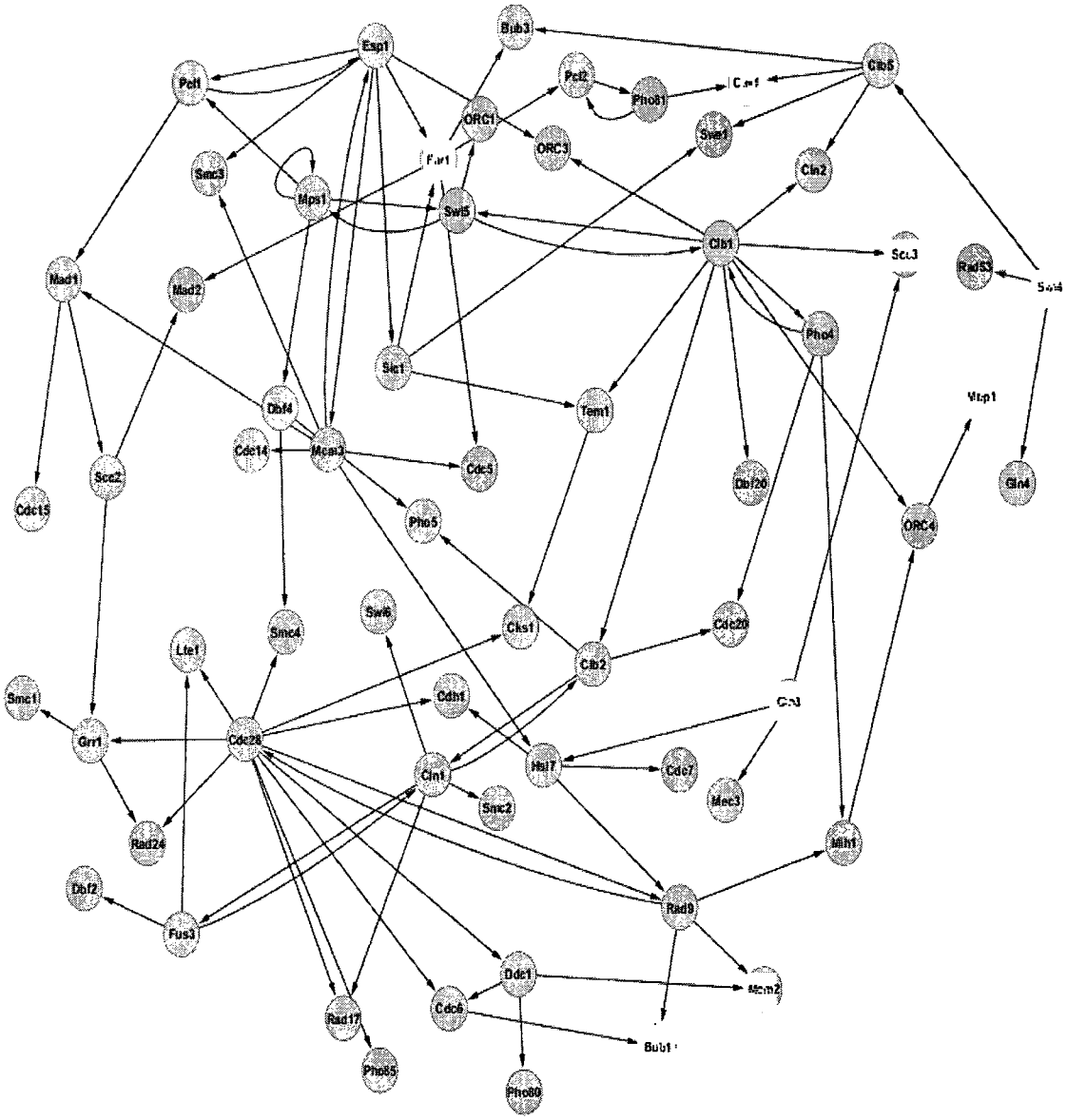
Figure 6.9 Cell cycle regulations in *Saccharomyces cerevisiae* (image source from http://www.genome.jp/kegg/pathway/sce/sce04111.html)

Here, 68 genes related to yeast cell cycle pathway are selected from [144], then gene expression data sets corresponding to these genes are extracted from Spellman's data sets [134]. The gene regulatory networks are inferred from both BLOM and DBN, which are shown in Figure 6.10 (a) and Figure 6.10 (b), respectively. There are 97 interactions inferred by DBN and 26 of them are correctly identified in Saccharomyces Genome Database (SGD) [149]; For BLOM, 24 interactions are identified out of 108 total inferred interactions. 9 interactions are identified by both DBN and BLOM. Interestingly, our model correctly identified *SWI4* and *Cln1/2* as the hub gene, which are missed by DBN but very critical in the process of yeast cell cycle according to [141]. The other hub gene *Cdc28* is successfully identified by DBN but missed by our model.



(a)

Figure 6.10 Inference of yeast GRN by BLOM and DBN. (a) BLOM, (b) DBN.


Bub3
Esp1
Clb5
Pcl1
Pcl2
Pho81
ORC1
ORC3
Swi4
Cln2
Mps1
Swi5
Clb1
Mad1
Mad2
Rad53
Pho4
Sic1
Tem1
Vtc1
Dbf4
Cdc14
Mcm3
Cdc5
Dbf20
Gln4
Cdc15
Scc2
Pho5
ORC4
Swi6
Cks1
Cdc20
Smc4
Lte1
Clb2
Smc1
Cdh1
Grr1
Cdc28
Cln1
Hsl7
Cdc7
Mec3
Smc2
Rad24
Mih1
Dbf2
Rad9
Fus3
Ddc1
Mcm2
Cdc6
Rad17
Bub1
Pho85
Pho80

(b)

CHAPTER VII

CONCLUSIONS

7.1 Summary and Conclusions

Inference of gene regulatory network from time series gene expression data is a very challenging task for computational biologists. Lots of mathematical algorithms and computational approaches have been proposed for modeling gene regulatory networks, such as Boolean network, differential equations and Bayesian network. There is no so called "golden method" which can generally give us the best performances for any kind of data sets. Some models and approaches can better describe the biological networks such as partial differential equation, but the computational time is not acceptable. While other approaches can infer gene regulatory networks from a large scale data set, i.e., information theory model, but it only can infer undirected networks and inference accuracy is very low. In the field of gene regulatory networks, the research goal is to improve the inference accuracy and reduce computational overhead.

In our work, probabilistic Boolean network (PBN) and dynamic Bayesian network (DBN) were compared using a biological time series dataset from Drosophila Interaction Database to construct a Drosophila gene network. A subset of time points and gene samples from the whole dataset is used to evaluate the performance of these two approaches. The performance of DBN and Bayesian network (R package) is also compared using the yeast cell cycle data sets. Our work shows that in most comparison cases, DBN outperforms PBN and Bayesian networks in term of accuracy.

We also improved inference accuracy of dynamic Bayesian network by preprocessing data using an innovative method called relative change ratios. In this work, relative change ratios and dynamic Bayesian network are combined together to infer gene

regulatory networks from synthetic data sets, which are used by DREAM challenge and widely popular in computational community. Relative change ratios method is used to preprocess the gene knock-out and knock-down data and potential regulators are selected regarding to target genes. Based on these preprocessed prior knowledge (potential gene regulators), dynamic Bayesian network is used to infer gene regulatory networks from time series gene expression data. In DREAM project, the gene regulatory networks inferred by our combining methods are very impressive and improve inference accuracy in a significant manner.

The traditional autoregressive methods will fail to infer gene regulatory networks which have high dimensionality and short time course gene expression data because the degree of freedom of the parameters is redundant. To overcome such difficulties, in this work, a new approach based on state space model and expectation-maximization algorithms is proposed to reconstruct gene regulatory networks. In our model, gene regulatory networks are represented by a state space model, which incorporates noises into observation and system functions and has the ability to capture more various biological aspects, such as hidden or missing variables. An EM algorithm is used to estimate the parameters based on the given state space functions, and then the conventional Kalman smoothing estimators are calculated by Kalman filter and Kalman smoother models. We derive the gene interaction matrix by decomposing the observation matrix using singular value decomposition and then use it to reconstruct GRNs.

We implement the above algorithms in MATLAB, and then two synthetic data sets are used to validate our new model before applying to real biological data sets. The results show that our model has the ability to infer the gene regulatory networks from large scale

gene expression data sets and can significantly reduce the computational time complexity without losing much inference accuracy compared to dynamic Bayesian network.

7.2 Future Directions

There are three possible areas that we can work on in the future with respect to this topic.

- Extending state space model by integrating multi-order time information into system functions and observation functions. In current model, the variables are related by first-order Markov process, which is not sufficient to provide more information for inferring GRN because it only captures the transition relationships between the adjacent two time points. By integrating multi-order Markov process into state space model, we can get to know how the multiple previous statuses affect the current variables and by this way we can more accurately infer the gene interactions.
- Integrating biological replicates or missing variables as a fact of noise into state space model. The biological and technical replicates in experiments design are considered as perturbations in the field of computational biological modeling. We can also take into account missing variables as perturbations in gene regulatory networks. These perturbations can be considered as observation noises and naturally integrated into current model, which can help us to capture more biological aspects.
- Modeling hidden variables in state space model and finding out the effects that have not been included in experiments. For example, we can learn how the genes interact with each other as groups and how specific genes bind together to regulate other genes. Hidden variables could also model levels of regulatory

proteins as well as possible effects of mRNA and protein degradations. In current work, we are mainly focused on predicting the gene interactions based on given gene expression data, however, learning hidden variables can be a strong plus to our inferred networks which provides us more biological insights.

REFERENCES

[1] Bodenreider, O. (2004). The unified medical language system (umls): Integrating biomedical terminology. *Nucleic Acids Res*, **32** (Database issue):D267–D270.

[2] National Human Genome Research Institute. http://www.genome.gov/Images/header/header_left.jpg

[3] David P. Clark (2005). Molecular Biology (Understanding the Genetic Revolution), Chapter 6. Elsevier academic press.

[4] Kulesh DA, Clive DR, Zarlenga DS, Greene JJ (1987). Identification of interferon-modulated proliferation-related cDNA sequences. *Proc Natl Acad Sci USA* **84**: *8453–8457. doi:*10.1073/pnas.84.23.8453.

[5] Schena M, Shalon D, Davis RW, Brown PO (1995). Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science* **270**: 467–470. *doi:*10.1126/science.270.5235.467.

[6] Lashkari DA, DeRisi JL, McCusker JH, Namath AF, Gentile C (1997). Yeast microarrays for genome wide parallel genetic and gene expression analysis. *Proc Natl Acad Sci USA* **94**: 13057–13062. *doi:*10.1073/pnas.94.24.13057.

[7] Lockhart, D. J. and Winzeler, E. A. (2000). Genomics, gene expression and DNA arrays. *Nature*, **405**(6788):827–836.

[8] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular Classification of Cancer Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, **286**(5439):531–537.

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

[10] Affymetrix: http://www.affymetrix.com/index.affx

[11] Shalon D, Smith SJ, Brown PO (1996). A DNA microarray system for analyzing complex DNA samples using two-color fluorescent probe hybridization. *Genome Res* **6**: 639–645.

[12] Tang T, François N, Glatigny A, Agier N, Mucchielli MH, Aggerbeck L, Delacroix H (2007). Expression ratio evaluation in two-colour microarray experiments is significantly improved by correcting image misalignment. *Bioinformatics* **23**: 2686–2691.

[13] D. J. Duggan, M. Bittner, Y. Chen, P. Meltzer and J. M. Trent (1999). Expression profiling using cDNA microarrays. *Nature Genetics*, **21**(1 Suppl.):10–14.

[14] Chatterjee, S. & Price, B (1991). Regression Analysis by Example. (John Wiley & Sons, New York).

[15] Tseng, G.C., Oh, M.K., Rohlin, L., Liao, J.C. & Wong, W.H (2001). Issues in cDNA microarray analysis, quality filtering, channel normalization, models of variations and assessment of gene effects. *Nucleic Acids Res.* **29**, 2549–2557.

[16] Chen, Y., Dougherty, E.R. & Bittner, M.L (1997). Ratio-based decisions and the quantitative analysis of cDNA microarray images. *J. Biomed. Optics* **2**, 364–374.

[17] Yang, Y.H. et al (2002). Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res.* **30**, e15.

[18] Yang, I.V. et al (2002). Within the fold: assessing differential expression measures and reproducibility in microarray assays. *Genome Biol* **3**, research0062.1–0062.12.

[19] Cleveland, W.S (1979). Robust locally weighted regression and smoothing scatterplots. *J. Amer. Stat. Assoc.* **74**, 829–836.

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

[21] Wen, X., Fuhrman, S., Michaels, G.S., Carr, D.B., Smith, S., Barker, J.L. & Somogy, R (1998). Large-scale temporal gene expression mapping of central nervous system development. *Proc. Natl Acad. Sci. USA* **95**, 334–339.

[22] Tamayo, P. *et al* (1999). Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Natl Acad. Sci. USA* **96**, 2907–2912.

[23] Quackenbush, J (2002). Microarray data normalization and transformation. *Nature Genetics* **32**, 496-501.

[24] Grant, R. P. (Ed.) September 2004. *Computational Genomics: Theory and Application.* Chapter 11. Horizon Scientific Press, UK. ISBN: 1904933017.

[25] Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., and Levine, A.J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA* **96**, 6745-6750.

[26] Ikemura, T (2002). Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions, *Proc. 9th ACM-SIAM Symp.* Discrete Algorithms, 695-702.

[27] Kanaya, S., Kinouchi, M., Abe, T., Kudo, Y., Yamada, Y., Nishi, T., Mori, H., and Ikemura, T (2001). Analysis of codon usage diversity of bacterial genes with a self-organizing map(SOM): char-acterization of horizontally transferred genes with emphasis on the E. coli O157 genome, *Gene*, **276**:89-99.

[28] Segal, E., Shapira, M., Regev, A., Pe'er, D., Botstein, D., Koller, D., and Friedman, N (2003). Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat. Genet.* **34**, 166-176.

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

[30] A. de la Fuente *et al* (2004). Hoeschele, and P. Mendes, Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics* **20**, 3565-74.

[31] J.J. Rice, Y. Tu, et al (2005). Reconstructing biological networks using conditional correlation analysis. *Bioinformatics* **21**(6): 765-773.

[32] N. Friedman, et al (2000) Using Bayesian Networks to Analyze Expression Data. *Journal of Computational Biology* 7(3-4): 601-620.

[33] I. Pournara, L. Wernisch (2004). Reconstruction of gene networks using Bayesian learning and manipulation experiments. *Bioinformatics* **20**(17): 2934-2942.

[34] J. Yu, V. Smith, P. Wang, A. Hartemink and E Jarvis (2004). Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics* **20**: 3594-603.

[35] M.J. Beal, F. Falciani (2005). A Bayesian approach to reconstructing genetic regulatory networks with hidden factors. *Bioinformatics* **21**(3): 349-356.

[36] N. Nariai, et al. (2005). Estimating gene regulatory networks and protein-protein interactions of Saccharomyces cerevisiae from multiple genome-wide data. *Bioinformatics* **21**(suppl 2): ii206-212.

[37] M. Zou and S. D. Conzen (2005). A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics* **21**(1): 71-79.

[38] N. Dojer, et al (2006). Applying dynamic Bayesian networks to perturbed gene expression data. BMC *Bioinformatics* **7**(1): 249.

[39] S. Liang, S. Fuhrman, et al (1998). Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Pac Symp Biocomput*: 18-29.

[40] T. Akutsu, S. Miyano, and S Kuhara (2000). Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics* **16**(8): 727-34.

[41] S. Mehra, W.S. Hu, et al (2004). A Boolean algorithm for reconstructing the structure of regulatory networks. *Metabolic Engineering* **6**(4): 326.

[42] S. Martin, Z. Zhang, et al (2007). Boolean Dynamics of Genetic Regulatory Networks Inferred from Microarray Time Series Data. *Bioinformatics*: btm021.

[43] M. Andrec, B. N. Kholodenko (2005). Inference of signaling and gene regulatory networks by steady-state perturbation experiments: structure and accuracy. *Journal of Theoretical Biology* **232**(3): 427.

[44] J. Kim, D. Bates, et al (2007). Least-squares methods for identifying biochemical regulatory networks from noisy measurements. *BMC Bioinformatics* **8**(1): 8.

[45] Stuart, J.M., Segal, E., Koller, D., Kim, S.K (2003). A gene-coexpression network for global discovery of conserved genetic modules. *Science* **302** (5643), 249–255.

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

[47] Butte, A., Kohane, I (2000). Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements. *Proceeding of the Pacific Symposium on Biocomputing*, pp. 418–429.

[48] Margolin, A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Favera, R., Califano, A (2006). ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*. **7** (Suppl. 1), S7.

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

[50] Faith, J.J., Hayete, B., Thaden, J.T., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J.J., Gardner, T.S (2007). Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol.* **5**(1), e8.

[51] Rao, A., Hero III, A.O., States, D.J, Engel, J.D (2007). Using directed information to build biologically relevant influence networks. *Comput. Syst. Bioinformatics Conf.* **6**, 145–156.

[52] Opgen-Rhein, R., Strimmer, K (2007). From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Syst. Biol.* **1**, 37.

[53] Shannon, C.E. and Weaver, W (1963). The Mathematical Theory of Communication. University of Illinois Press.

[54] Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. Chapter 2. John Wiley & Sons, New York, NY.

[55] W. Zhao, E. Serpedin, and E. R. Dougherty (2008). Inferring connectivity of genetic regulatory networks using information-theoretic criteria," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. **5**, no. 2, pp.262-274.

[56] Kauffman SA (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, **9**:3273-3297.

[57] K. Glass and S. A. Kauffman (1973). The logical analysis of continuous, nonlinear biochemical control networks. *J. Theoret. Biol.*, vol. **39**, pp. 103–129.

[58] S. A. Kauffman (1974): The large scale structure and dynamics of genetic control circuits: an ensemble approach. *J. Theoret. Biol.*, vol. **44**, pp. 167–190.

[59] Huang S (1999). Gene expression profiling, genetic networks and cellular states: An integrating concept for tumorigenesis and drug discovery. *Journal of Molecular Medicine*, **77**:469-480.

[60] Shmulevich I, Gluhovsky I, Hashimoto RF, Dougherty ER, Zhang W (2003). Steady-state analysis of genetic regulatory networks modeled by probabilistic Boolean networks. *Comparative and Functional Genomics*, **4**:601-608.

[61] Kim H, Lee JK, Park T (2007). Boolean networks using the chi-square test for inferring large-scale gene regulatory networks. *BMC Bioinformatics*, **8**:37.

[62] Shmulevich I., Dougherty R., Zhang W (2002). From Boolean to Probabilistic Boolean Networks as Models of Genetic Regulatory Networks. *Proceeding of the*

IEEE, **90**(11), 1778-1792.

[63] Shmulevich I., E.R. Dougherty, K. Seungchan, W. Zhang (2002). Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, **18**(2): 261–274.

[64] Shmulevich I., Dougherty E. R. and Zhang W (2002). Gene perturbation and intervention in probabilistic Boolean networks. *Bioinformatics*, Vol. **18** no. 10:1319-1331.

[65] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja (2003). On Learning Gene **52**, pp. 147 167.

[66] X. Zhou, X. Wang, and E. R. Dougherty (2003): Construction of genomic networks using mutual information clustering and reversible-jump Markov-Chain-Monte-Carlo predictor design. *Signal Processing*, vol. **83**, no. 4, pp. 745--761.

[67] Dougherty, E. R., Kim, S. and Chen, Y (2000). Coefficient of determination in nonlinear signal processing. *Signal Processing*, **80**:10, pp. 2219-2235. 16.

[68] N. Friedman, K. Murphy, S (1998). Russell: Learning the structure of dynamic probabilistic networks. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 139–147.

[69] H. Lahdesmki, S. Hautaniemi, I. Shmulevich, O.Yli-Hrja (2006): Relationships between probabilistic Boolean networks and dynamic Bayesian networks as models of gene regulatory networks. *Signal Processing,* volume **86**, issue 4, 814 - 834.

[70] Akutsu T, Miyano S, Kuhara S (1999). Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. *Pacific Symposium on Biocomputing*, **4**:17-28.

[71] M. Brun, E. R. Dougherty, I. Shmulevich (2005). Steady-State Probabilities for Attractors in Probabilistic Boolean Networks. *Signal Processing*, **85**, 1993–2013.

[72] Voit, E.O (2000). Computational analysis of biochemical systems: a practical guide for biochemists and molecular biologists. Cambridge University Press, Cambridge, New York.

[73] De Jong, H (2002). Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.* **9**, 67–103.

[74] Climescu-Haulica, A., Quirk, M.D (2007). A stochastic differential equation model for transcriptional regulatory networks. *BMC Bioinformatics*, **8** (Suppl. 5), S4.

[75] Kaern, M., Elston, T.C., Blake,W.J., Collins, J.J (2005). Stochasticity in gene expression: from theories to phenotypes. *Nat. Rev. Genet.* **6** (6), 451–464.

[76] Chen T, He HL, Church GM (1999): Modeling gene expression with differential equations. *Pacific Symposium Biocomputing*, **4**:29-40.

[77] Holter, N.S., Maritan, A., Cieplak, M., Fedoroff, N.V., Banavar, J.R (2001). Dynamic modeling of gene expression data. *Proc. Natl. Acad. Sci. U.S.A.* **98** (4), 1693–1698.

[78] Yeung, M.K.S., Tegner, J., Collins, J.J (2002). Reverse engineering gene networks using singular value decomposition and robust regression. *Proc. Natl. Acad. Sci. U.S.A.* **99** (9), 6163–6168.

[79] van Someren, E.P., Wessels, L.F., Backer, E., Reinders, M.J (2002). Genetic network modeling. *Pharmacogenomics* **3**, 507–525.

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

[81] Di Bernardo, D., Thompson, M., Gardner, T., Chobot, S., Eastwood, E., Wojtovich, A., Elliott, S., Schaus, S., Collins, J (2005). Chemogenomic profiling on a genome-Wide scale using reverse-engineered gene networks. *Nat. Biotechnol.* **23** (3), 377–383.

[82] Bansal, M., Gatta, G.D., di Bernardo, D (2006). Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics* **22** (7), 815–822.

[83] Hartemink, A.J., et al. (2002) Combining location and expression data for principled discovery of genetic regulatory network. ***Pacific Symp. Biocomput***, . **7**, 437–449.

[84] P. Smolen, D.A. Baxter, and J.H. Byrne (2000). Modeling transcriptional control in gene networks: Methods, recent results, and future directions. *Bulletin of Mathematical Biology*, **62**:247_292.

[85] H.D. Landahl (1969). Some conditions for sustained oscillations in biochemical chaines. *Bulletin of Mathematical Biophysics*, **31**:775_787.

[86] J. Richelle (1979). Comparative analysis of negative loops by continuous, boolean and stochastic approaches. In R. Thomas, editor, Kinetic Logic: A Boolean Approach to the Analysis of Complex Regulatory Systems, volume **29** of Lecture Notes in *Biomathematics*, pages 281_325. Springer-Verlag, Berlin.

[87] Sakamoto, E., Iba, H (2001). Inferring a system of differential equations for a gene regulatory network by using genetic programming. In: *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press, pp. 720–726.

[88] Spieth, C., Hassis, N., Streichert, F (2006). Comparing mathematical models on the problem of network inference. In: *Proceeding of the 8th Annual Conference on Genetic and evolutionary computation (GECCO 2006)*,Washington, USA, pp. 279–285.

[89] Weaver, D., Workman, C., Stormo, G (1999). Modeling regulatory networks with weight matrices. *Proceeding of the Pacific Symposium on Biocomputing*, pp.112–123.

[90] Kimura, S., Ide, K., Kashihara, A., Kano, M., Hatakeyama, M., Masui, R., Nakagawa, N., Yokoyama, S., Kuramitsu, S., Konagaya, A (2005). Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics* **21** (7), 1154–1163.

[91] Vilela, M., Chou, I.C., Vinga, S., Vasconcelos, A.T., Voit, E.O., Almeida, J.S (2008). Parameter optimization in S-system models. *BMC Syst. Biol.* **16** (2), 35.

[92] P.K. Maini, K.J. Painter, and H. Nguyen Phong Chau (1997). Spatial pattern formation in chemical and biological systems. *Journal of the Chemical Society, Faraday Transactions*, **93**(20):3601_3610.

[93] A. Gierer and H. Meinhardt (1972). A theory of biological pattern formation. *Kybernetik*, **12**:30_39.

[94] S.A. Kauffman (1974). The large-scale structure and dynamics of gene control circuits: An ensemble approach. *Journal of Theoretical Biology*, **44**:167_190.

[95] Perrin BE, Ralaivola L et al (2003). Gene networks inference using dynamic Bayesian networks. *Bioinformatics,* **19** Suppl 2:II138-II148.

[96] Zhengzheng Xing, Dan Wu (2006): Modeling Multiple Time Units Delayed Gene Regulatory Network Using Dynamic Bayesian Network. *ICDM Workshops,* 190-195.

[97] Zhang, L., Samaras, D., Alia-Klein, N., Volkow, N., and Goldstein, R (2006). Modeling neuronal interactivity using dynamic Bayesian networks. *Advances in Neural Information Processing Systems* ***18***. Eds. Y. Weiss and B. Scholkopf, and J. Platt. Cambridge, MA: MIT Press.

[98] Werhli, A.V., Husmeier, D (2007). Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Stat. Appl. Genet. Mol. Biol.*, **6**:Article 15.

[99] Needham, C.J., Bradford, J.R., Bulpitt, A.J.,Westhead, D.R (2007). A primer on learning in Bayesian networks for computational biology. *PLoS Comput. Biol.* **3** (8), e129.

[100] Murphy, K. and Mian, S (1999). Modelling gene expression data using dynamic Bayesian networks. Technical report, Computer Science Division, University of California, Berkeley, CA.

[101] Z. Ghahramani (2002). Graphical Models: Parameter Learning. The handbook of Brain Theory and Neural Networks (2^{nd} edition).

[102] R. Greiner and W. Zhou (2002). Structural extension to logistic regression: Discriminant parameter learning of belief net classifiers. In *AAAI.*

[103] H. Wettig, P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri (2003). When discriminative learning of Bayesian network parameters is easy. In *IJCAI2003*, pages 491-496.

[104] P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri (1999). On supervised selection of Bayesian networks. In *UAI99*, pages 334-342.

[105] A. Ng and M. Jordan (2001). On discriminative versus generative classifiers: A comparison of logistic regression and naive Bayes. In *NIPS*.

[106] B. Shen, X. Su, R. Greiner, P. Musilek, and C. Cheng (2003). Discriminative parameter learning of general Bayesian network classifiers. In *ICTAI-03*.

[107] Murphy, K (2002). Dynamic Bayesian Networks: Representation, Inference and Learning. PhD dissertation, University of California, Berkeley.

[108] W. L. Buntine (1994). Operations for learning with graphical models. *J. of AI Research*, pages 159–225.

[109] W. Buntine (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Trans. On Knowledge and Data Engineering*, **8**(2).

[110] D. Heckerman (1996). A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research.

[111] S. L. Lauritzen (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, **19**:191–201.

[112] J. Binder, D. Koller, S. J. Russell, and K. Kanazawa (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, **29**:213–244.

[113] T. M. Cover and J. A. Thomas (1991). *Elements of Information Theory*. JohnWiley.

[114] G. Schwarz (1978). Estimating the dimension of a model. *Ann. Stat.*, **6**:461–464.

[115] D. Heckerman, D. Geiger, and D. M. Chickering (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learning*, **20**:197–243.

[116] D. Chickering, D. Heckerman, and D. Geiger (1995). Learning Bayesian networks: search methods and experimental results. In *AI + Stats*.

[117] V. Mihajlovic and M. Petkovic (2001). Dynamic Bayesian networks: A state of the art. *CTIT technical reports series*, vol. TR-CTIT-34.

[118] R. C. Conant (1988). Extendeddependencyanalysis of large systems. *Intl. J. General Systems*, **14**:97–141.

[119] N. Friedman (1997). Learning Bayesian networks in the presence of missing values and hidden variables. In *Proc. of the Conf. on Uncertainty in AI.*

[120] N. Friedman (1998). The Bayesian structural EM algorithm. In *Proc. of the Conf. on Uncertainty in AI.*

[121] Bayes Net Toolbox. http://www.cs.berkeley.edu/~murphyk/Bayes/bnt.html.

[122] Yu,H., Luscombe,N.M., Qian,J. and Gerstein,M (2003). Genomic analysis of gene expression relationships in transcriptional regulatory networks. *Trends Genet,* **19**, 422–427.

[123] Hartley, H (1958): Maximum likelihood estimation from incomplete data. *Biometrics*, **14**:174–194.

[124] Minka, T: Expectation-Maximization as lower bound maximization. Tutorial published on the web at http://www-white.media.mit.edu/ tpminka/papers/em.html.

[125] Beal, M.J., Ghahramani, Z (2003). The Variational Bayesian EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures. *Bayesian Statistics* 7, Oxford University Press.

[126] O. Hirose, R. Yoshida, S. Imoto, R. Yamaguchi, T. Higuchi, D. S. Charnock Jones, C. Print, and S. Miyano (2008). Statistical inference of transcriptional module-based gene networks from time course gene expression profiles by using state space models. *Bioinformatics*, **24**(7): 932 - 942.

[127] Wu,F., Zhang,W. and Kusalik,A (2004). Modeling gene expression from microarray expression data with state-space equations. *Pac. Symp. Biocomput.* **9**, 581-592.

[128] Ted Rosenbaum & Ariel Zetlin-Jones (2006). The Kalman Filter and the EM Algorithm. December 17.

[129] Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* **82** (1): 35–45

[130] Kalman, R.E.; Bucy, R.S. (1961). New Results in Linear Filtering and Prediction Theory. *Trans. ASME, Series D, Journal of Basic Engineering,* **83**:95—108

[131] Julier, S.J.; Uhlmann, J.K. (1997). A new extension of the Kalman filter to nonlinear systems. *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls* **3**

[132] The DREAM project. http://wiki.c2b2.columbia.edu/dream/index.php/

[133] Drosophila Interaction Database. http://portal.curagen.com/cgibin/interaction/flyHome.pl

[134] Spellman PT, et al (1998). Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization. *Molecular Biology of the Cell*, **9**: 3273-3297.

[135] Marbach, D., Schaffter, T., Mattiussi, C. and Floreano, D (2009). Generating Realistic *in silico* Gene Networks for Performance Assessment of Reverse Engineering Methods. Journal of Computational Biology, **16**(2), *in press*.

[136] Arbeitman,M.N. et al (2002). Gene expression during the life cycle of Drosophila melanogaster. *Science*, **297**, 2270–2275.

[137] Giot,L. et al (2003). A protein interaction map of Drosophila melanogaster.

Science, **302**, 1727–1736.

[138] W. Zhao, E. Serpedin, E. Dougherty (2006). Inferring Gene Regulatory Networks from Time Series Data Using the Minimum Description Length. *Bioinformatics*, Vol **22**, No. 17, p2129-2135.

[139] P. Li, C. Zhang, E. J. Perkins, P. Gong, Y. Deng (2007). Comparison of probabilistic Boolean network and dynamic Bayesian network approaches for inferring gene regulatory networks. *BMC Bioinformatics*, Vol. **8**(Suppl 7):S13.

[140] Chen X, Chen M, Ning K (2006). BNArray: An R package for constructing gene regulatory networks from microarray data by using Bayesian network. *Bioinformatics*, **22**(23): 2952-2954.

[141] Simon, I et al (2001). Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, **106**:697-708.

[142] Lee T, et al (2002). Transcriptional Regulatory Networks in Saccharomyce Cerevisiae. *Science*, **298**, pp. 799-804.

[143] BioGRID Interaction Database. http://www.thebiogrid.org

[144] Kyoto Encyclopedia of Genes and Genomes Pathway Database (KEGG) http://www.genome.ad.jp/kegg/pathway.html

[145] Stolovitzky, G., Monroe, D., and Califano, A (2007). Dialogue on reverse-engineering assessment and methods: The dream of highthroughput pathway inference. *Ann. N.Y. Acad. Sci.* **1115**, 1–22.

[146] Tegner, J., Yeung, M.K.S., Hasty, J., and Collins, J.J (2003). Reverse engineering gene networks: Integrating genetic perturbations with dynamical modeling. *Proc. Natl. Acad. Sci. U.S.A.* **100**, 5944–5949.

[147] Van den Bulcke, T., van Leemput, K., Naudts, B., van Remortel, P., Ma, H., Verschoren, A., de Moor, B., and Marchal, K (2006). SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*. **7**, 43.

[148] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis (2002). Using Bayesian network inference algorithms to recover molecular genetic regulatory networks. In *International Conference on Systems Biology* (ICSB02).

[149] Saccharomyces Genome Database (SGD). http://www.yeastgenome.org/