ELSEVIER

Expert Systems with Applications 30 (2006) 42–49

Expert Systems with Applications

www.elsevier.com/locate/eswa

Model gene network by semi-fixed Bayesian network

Tie-Fei Liu [a], Wing-Kin Sung [a,*], Ankush Mittal [b]

[a] Department of Computer Science, National University of Singapore, Singapore
[b] Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee, India

Abstract

Gene networks describe functional pathways in a given cell or tissue, representing processes such as metabolism, gene expression regulation, and protein or RNA transport. Thus, learning gene network is a crucial problem in the post genome era. Most existing works learn gene networks by assuming one gene provokes the expression of another gene directly leading to an over-simplified model. In this paper, we show that the gene regulation is a complex problem with many hidden variables. We propose a semi-fixed model to represent the gene network as a Bayesian network with hidden variables. In addition, an effective algorithm based on semi-fixed structure learning is proposed to learn the model. Experimental results and comparison with the-state-of-the-art learning algorithms on artificial and real-life datasets confirm the effectiveness of our approach.

Keywords: Gene network; Bayesian networks; Hidden variable; Semi-fixed network; Semi-fixed structure EM learning algorithm

1. Introduction

The complexity of a living cell is achieved by the concerted activity of many genes and their products. Such activity is often coordinated by the organization of the genome into regulatory modules, or sets of co-regulated genes that share a common function (Segal et al., 2003). The global gene expression pattern is therefore the result of the collective behavior of individual regulatory pathways. The large number of regulation pathways comprise a complex network which is called a gene network (Kolpakov, Ananko, Kolesov, Kolchanov, & Genenet, 1998). Understanding the network as a whole is essential and learning the gene network is an important central theme in post genomic research (Cumiskey, Levine, & Armstrong, 2003; D'haeseleer, Liang, & Somogyi, 2000; Friedman, 2004; Hasty, McMillen, Isaacs, & Collins, 2001). The accurate reconstruction of gene network has many possible benefits: (1) gene network provides information to help the annotation of the genome (Brazhnik, Fuente, & Mendes, 2002); (2) it is an important step to uncover the biochemical network in a cell (Brazhnik et al., 2002); (3) it provides valuable clue and lead to new idea to treat some complex diseases (Friedman, 2004), etc. With the construction of the expert system of gene network, it is possible to predict the regulators of genes, the regulatory relationships among genes and the regulatory pathways, thus learn the regulatory patterns of the organisms. Learning gene network is also a necessary step to build an expert system to predict the outcome of the perturbation of the genome, such as diseases (Brazhnik et al., 2002). Yet, it is impractical to construct a detailed biochemical model of an organism containing hundreds or even tens of thousands of genes by analyzing each gene and determining all the binding and reaction constants one by one manually[1]. Therefore, the methods for automatically reconstructing gene network by computing are needed.

Gene network can be reconstructed by analyzing the gene expression data, which is extracted from the mi-croarray (Akutsu, Miyano, & Kuhara, 1999; Chen, He, & Church, 1999; Chen, Filkov, & Skiena, 1999; Cumiskey et al., 2003; D'Haeseleer, Wen, Fuhrman, & Somogyi, 1999; Friedman, Linial, Nachman, & Pe'er, 2000; Imoto, Goto, Miyano, 2002; Imoto et al., 2003; Murphy & Mian, 1999; Someren, Wessels, & Reinders, 2000; Tominaga, Okamoto, Watanabe, & Eguchi, 2001; Wagner, 2002; Watanabe, 1998; Wessels, Someren, & Reinders, 2001; Yaki & Friedman, 2004). Various methods are used for this purpose. The early computational approaches were based on learning the relationships among genes either by studying the mutual information or the correlation among their expression values. The representatives of such approaches are

* Corresponding author. Tel.: +65 68746772; fax: +65 67797465.
E-mail addresses: liutiefe@comp.nus.edu.sg (T.-F. Liu), ksung@comp.nus.edu.sg (W.-K. Sung), ankumfec@iitr.ernet.in (A. Mittal).

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.09.044

[1] http://www.cs.unm.edu/~patrik/networks/networks.html

Pair-wise interaction (Arkin, Shen, & Ross, 1997) and clustering (Wahde & Hertz, 2000), which directly find the correlations among genes. After that, Boolean networks (Akutsu et al., 1999; Liang, Fuhrman, & Somogyi, 1998) are used in several works, which assume the gene expression levels are represented by Boolean values and the gene regulatory relationship are represented by a set of Boolean functions. Linear (Someren et al., 2000) and non-linear (Watanabe, 1998) models are followed which assume the regulatory relationships are represented by linear functions and non-linear functions, respectively. In a famous paper by Friedman et al. (2000), an algorithm to learn gene network using Bayesian network is presented: Each gene is a vertex and each regulatory relationship is an edge in the Bayesian network. Bayesian network represents probabilistic multivariate statistical dependencies among variables in a compact and easy-to-decipher way (Barrientos & Vargas, 1998; Gustavo, Sucar, & Villavicencio, 1998; Kang & Gola, 1999; Oatley & Ewart, 2003; Sucar & Miriam, 1998; Viaene, Dedene, & Derrig, in press). As it can handle stochastic events, control noise and handle dataset with only a few replicates (Friedman et al., 2000; Imoto et al., 2002), it has been shown to have advantages over other methods in learning gene network (Friedman et al., 2000; Imoto et al., 2002; Murphy & Mian, 1999). Since then, many works based on Bayesian network frame work are proposed and more biological relevant results are obtained. Hartemink, Gifford, Jaakkola, & Young, (2001) extended Friedman's work by adding the annotations to the edges: '+', '−' or '±' which represent the positive, negative or unknown regulation. Imoto et al., extended Bayesian network by combining non-parametric regression (Imoto et al., 2002) to detect the nonlinear relationship among genes and by making use of relevant biological information (Imoto et al., 2003) to improve the learning performance. Murphy and Mian (Murphy & Mian, 1999) used the dynamic Bayesian network, which is an extension of Bayesian network, to model the time delay in the gene network. However, when referring to gene regulation, most works simply model that a gene is directly regulated by other genes. The regulations of genes in fact occur at various levels including transcription, translation, splicing, posttranslational protein degradation, and other processes (Kolpakov et al., 1998). To give a correct description of the gene network, we cannot just consider the gene expression level (that is, the level of mRNA transcription).

Based on the biological knowledge, it is known that protein plays a key role in gene network (Wyrick & Young, 2002). In fact, protein–DNA interaction and Protein-Protein interaction are the main activities in the regulatory system (Kolpakov et al., 1998). Therefore, when predicting microarray gene expression data, proteins should be included. Although their expression levels are still difficult to measure in the large scale, it is good if we can treat them as hidden variables. The following are the advantages of modeling proteins as hidden variables:

- The model will become more meaningful, more interpretable and closer to real-life system (Barbara, Jameson, & Witting, 1999; Elidan, Lotner, Friedman, & Koller, 2000; Friedman, 1997; Friedman, 1998).
- In the model, the proteins are decision-relevant. The network without considering hidden variable may omit some dependencies (Barbara et al., 1999; Elidan et al., 2000).

Introducing hidden variables introduces advantages as well as increases the complexity of the network learning. Moreover, microarray gene expression datasets often have missing values. Considering the above challenging issues, Bayesian network forms a natural choice with the advantage of supporting several principled methods for learning the causal relationships with incomplete data, both hidden variables and missing values (Friedman, 1998). Successful application of Bayesian network to learn structure with hidden variables can be found in for several applications (Barbara et al., 1999; Elidan et al., 2000; Friedman, 1997, 1998). The most widely used method for structure learning is EM algorithm (Friedman, 1998). In E step, the algorithm calculates the score of each possible structure using the structure and parameters learnt in M step. The procedure is repeated until the convergence criterion is met. In our problem, such kind of learning is difficult since the algorithm needs to learn the relationship among the hidden variables and the observed variables. In addition it is difficult to predetermine the correct number of hidden variables. To learn the optimum number of hidden variables is computationally complex. Besides, in the presence of hidden variables, the network is no longer decomposable and this makes the learning difficult (Friedman, 1998) (as described in Section 3).

We propose a system which models the gene network as a directed graph with hidden variables. In our model, the number of hidden variables is predefined using the biological knowledge and in addition, the relationships between hidden variables and observed variables are partially fixed. We propose a modified EM algorithm that takes advantages of the semi-fixed structure to decompose the network, and thus allow us to learn such network efficiently. Also, an approximation method to perform inference on the joint probability of two genes is presented in order to speed up the learning procedure.

2. Model gene network as a semi-fixed network with hidden variables

In the gene regulation system, the regulation process can be divided into two main steps. The first step is gene expression, which is represented by $g_i \rightarrow r_i \rightarrow p_i$ where g_i is a gene, r_i and p_i are the corresponding mRNA and Protein respectively. $g_i \rightarrow r_i$ is the transcription and $r_i \rightarrow p_i$ is the translation process. The transcription efficiency that is represented by mRNA level can be measured by microarray. The second step is the gene regulation, which is represented by $p_i \rightarrow g_j$. In this step, the generated protein p_i, possibly in collaboration with some other proteins, regulates the target gene g_i. These two steps are the most important steps in the gene regulatory system. All other steps such as protein degradation just adjust the expression and the regulation strengths. Let $Pa^j = \{g_1, \ldots, g_k\}$ denote a parent

set of gene g_j such that each $g_i \in Pa^j$ regulates gene g_j. We note that all proteins $\{p_1,\ldots, p_k\}$ in Pa^j act in a combinative manner to regulate g_j. In other words, the proteins may combine together to form a complex then bind to the binding site of the target gene and regulate it, or bind to their own binding sites then collaboratively regulate the target gene, or a mixture of the above two ways. An example network is shown in Fig. 1(a). Considering a single node representing the proteins' combined influence as the direct regulator to regulate the target gene, the model can be further simplified as shown in Fig. 1(b).

Based on the above discussion, for each gene g_j, we propose a hidden variable h_j, which represents the combination of a set of regulatory proteins expressed from Pa^j. Thereupon, Pa^j regulate h_j, then h_j regulates g_j. Based on the simplified regulation system, each gene is regulated by at most one hidden variable and the hidden variables are regulated by one or more than one genes. Such model means the interaction exists only between *Gene* $\leftarrow\rightarrow$ *Protein* or among proteins, and there is no edges from gene to gene or hidden variable to hidden variable. This model is termed as a semi-fixed network and is formally defined as follows:

- A semi-fixed network $N = < V, E >$ where vertices $V = O \cup H$, $O = \{g_1, \ldots, g_n\}$ is a set of observed variables representing the expression levels of genes, and $H = \{h_i, \ldots, h_n\}$ is a set of hidden variables representing the expression levels of combined proteins) and E is the edge set.
- For each $e_k \in E$, $e_k = (g_i, h_j)$ or (h_j, g_i). Thus, N is a bipartite graph on two partitions O and H.
- For each $g_i \in O$, there is exactly one incoming edge (h_i, g_i) while there can be many outgoing edges.
- For each $h_i \in H$, it has exactly one outgoing edge (h_i, g_i) while there can be many incoming edges $\{(g_{i1}, h_i), (g_{i2}, h_i), \ldots, (g_{ik}, h_i)\}$ where $Pa(h_i) = Pa_j = \{g_{i1}, g_{i2}, \ldots, g_{ik}\}$.

Compared to learning general network with several hidden variables, learning the semi-fixed network is easier since the number of hidden variables is fixed and the relationships between hidden variables and observed variables are partially known.

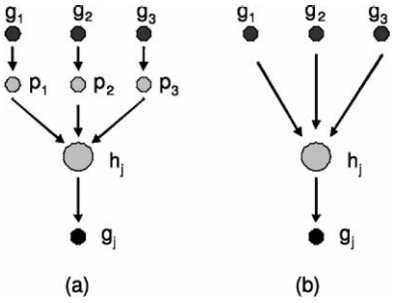

Fig. 1. Simplified gene regulation system. (a) Gene expression system can be simplified as the interaction of genes and proteins. (b) The system can be further simplified since the combined proteins are the direct regulator to the target genes.

3. Semi-fixed structure EM learning algorithm

When there is no hidden variable in the network and no missing values in the gene expression data, the probability of the variables given a network structure can be expressed as the production of the probabilities of independent sub-networks:

$$P(X_1 \ldots X_n : N_o, D) = \prod_i P(X_i | P_a(X_i)) \tag{1}$$

where X_1 to X_n are observed variables and N_o denotes the network without hidden variables. D is a complete dataset.

The decomposability property reduces the learning difficulty in the following manner. With score functions such as minimum description length (MDL) and Bayesian scoring metric, learning the structure of a network can be decomposed to learn each independent sub-network separately (Friedman, 1998; Friedman, Nachman, & Peer, 1999). Whereas, in the presence of incomplete data, the decomposability property is not valid and this makes the learning difficult (Friedman, 1998). However, if all parameters of all hidden variables are assigned, hidden variables are observed and thus in a sense, the incomplete dataset becomes 'complete'.

A useful property of semi-fixed network is that when all parameters of hidden variables have been assigned (as the number of hidden variables and partial relationships of hidden variables and observed variables are known ensuring that all hidden variables can be assigned parameters), the network can be decomposed into independent subnetworks. Each subnetwork comprises of a gene g_j, a hidden variable h_j and a parent set of h_j (denoted as $Pa(h_j)$) (*as* show in Fig. 1(b)). The probability is decomposed as:

$$P(g_1 \ldots g_n, h_{1\ldots hn} : N_{o,h}, D) = \prod_i P(h_i | Pa(h_i)) \prod_i P(g_i | h_i) \tag{2}$$

where $g_1, g_2, \ldots, g_n$ are observed variables (genes), $h_1, h_2, \ldots, h_n$ are hidden variables and $N_{o,h}$ denotes the network with hidden variables. D is an incomplete dataset.

Based on different decomposition methods, the decomposition of scores is also different. In our work, Bayesian score is used. For the detail of Bayesian score, please refer to (Friedman et al., 2000). When the network N_o has no hidden variable, the score is decomposed as follows:

$$\text{Score}(N_o) = \sum_i \text{Score}(g_i | Pa(g_i)) \tag{3}$$

For a semi-fixed network $N_{o,h}$, hidden variables are included. We can show that its Score can be decomposed because the hidden variables h_i's are attached to the corresponding genes, as follows:

$$\text{Score}(N_{o,h}) = \sum_i \text{Score}(h_i | Pa(h_i)) + \text{Score}(g_i | h_i) \tag{4}$$

Proof:. Given all hidden variables, $N_{o,h}$ is decomposable. By Formula 4, $\text{Score}(N_{o,h}) = \sum_i \text{Score}(v_i | Pa(v_i))$ where v_i is the variables, include both genes and proteins, in $N_{o,h}$: $\{v_1, v_2, \ldots,$

$v_{2n}\} = \{g_1, g_2, \ldots, g_n, h_1, h_2, \ldots, h_n\}$. Therefore, it is easy to see $\sum_i \text{Score}(v_i|Pa(v_i)) = \sum_i \text{Score}(h_i|Pa(h_i)) + \sum_i \text{Score}(g_i|Pa(g_i))$ where $P\ a(g_i) = h_i$ in $N_{o,h.}$ Thus, Formula (4) is proved.

With the aim of computing a network $N_{o,h}$ which maximizes $Score(N_{o,h})$ using the property in Eq. (4), we propose a EM algorithm known as Semi-fixed Structure EM (*SSEM*) algorithm to learn such network. The principle of the EM algorithm is as follows: In each iteration, given an initial network structure N_{i-1} and a parameter set θ_{i-1} (which includes both the parameters of observed variables θ_{i-1} and hidden variables θ^h_{i-1}), the step 1 is to calculate the missing values and hidden variables to build a complete dataset. The second step is to find a better structure N_i and a better parameter set θ_i based on the new complete dataset. A detailed description of the algorithm is shown at the end of this chapter.

For step 1, the missing values and hidden variables can be filled up as follows:

Given N and θ, for a missing value of a variable g_i, supposing $Pa(g_i)$ is the parent set of g_i in N and $D = \{d_1, \ldots, d_k\}$ is the discrete values of g_i, we fill up the missing value of g_i by a vector $S = (s_1, \ldots, s_k)$ where $s_j = P(g_i = d_j|Pa(g_i))$.

This is illustrated in the following example:

Example:. The variable g_1 has parents g_2 and g_3. As shown in Table 1(a), there is a missing value in an instance: $\{g_1, g_2, g_3\} = \{(), 1, 1\}$ where () indicate a missing value. Suppose $P(g_i = 0|g_2 = 1, g_3 = 1) = 0.4$ and $P(g_i = 1|g_2 = 1, g_3 = 1) = 0.6$, we fill up the instance by $\{g_1, g_2, g_3\} = \{(0.4, 0.6), 1, 1\}$. It means, the filled value adds 0.4 count to the case $g_1 = 0$ and 0.6 count to the case $g_1 = 1$. As shown in Table (b), there are 1.4 cases for which $g_1 = 0$ and 2.6 cases for which $g_1 = 1$. In other words, $P(g_1 = 0) = 1.4/4 = 0.35$ and $P(g_1 = 1) = 2.6/4 = 0.65$.

The values of hidden variables are treated as missing values too and are computed in a similar fashion. Given the filled dataset, N_{i-1} and θ_{i-1}, step 2 can learn N_i and θ_i using general structure learning method. The general method to learn the structure from a complete dataset is to decompose the network into independent subnetworks. Then, learn the structure of each subnetwork independently. Since we fix partial structure, we can decompose it as the independent subnetworks, each of which has a target gene, a hidden variable and a parent set of the hidden variable (similar to Fig. 1(b)). The main objective is to find the optimal parents for each hidden variable. Learning parents from a large number of candidates is difficult task. As gene network is a sparse network (Someren et al., 2000), a popular technique is to first measure the dependencies of candidates to target variable and to choose the best k genes as candidate parents. Then, the search for the parents from the candidate parent set can be performed. Friedman et al. (Friedman et al., 1999) proposed to calculate the dependency by KL-divergence (as Eq. (5)):

$$\text{MI}(X, Y|M) = \sum_{X,Y} P(X, Y)\log(P(X, Y)/P_M(X, Y)) \quad (5)$$

where $MI(X, Y|M)$ is the mutual information of variables X and Y with respect to the network M, $P(X, Y)$ is the observed joint probability of X and Y and $P_M(X, Y)$ is the estimated joint probability of X and Y given M.

Though we can compute $P_M(g_i, h_i)$ given the filled dataset, we cannot compute $P(g_j, h_i)$ to measure the dependency of g_j and h_i, as h_i is hidden. We propose an alternate way to measure the dependency: in each subnetwork, the target gene is the only descendant of the hidden variable. The probabilities of the hidden variable are passed to the target gene. Therefore, $MI(g_j, h_i\ |M)$ can be reflected by $MI(g_j, g_i|M)$. Thus, the MI can be calculated as following:

$$\text{MI}(g_j, g_i|M) = \sum_{gj,gi} P(g_j, g_i)\log(P(g_j, g_i)/P_M(g_j, g_i)) \quad (6)$$

Where $P(g_j, g_i)$ is the observed probability distribution of genes g_j and g_i while $P_M(g_j, g_i)$ is the estimated probability distribution of gene g_j and g_i in network M. Performing inference on joint probabilities in a big dataset is quite time intensive. Thus, we approximate the joint probability of g_i and g_j as follows:

- If $g_j \in \text{Pa}(g_i)$, then $g_j \to h_i \to g_i$. $\text{PM}(g_j, g_i) = \text{PM}(g_i|g_j)\text{PM}(g_j) \approx \text{PM}(g_i|h_i)\text{PM}(h_i|g_j)\text{PM}(g_j)$. Note that in this case, $\text{PM}(g_j, g_i) \neq \text{PM}(g_i, g_j)$.
- Otherwise, g_i and g_j are conditionally independent and $P_M(g_j, g_i)$ can be approximated as $P_M(g_j)P_M(g_i)$.

By the above approximation, only the joint probabilities of the gene pairs with the parent-child relationships need to be calculated.

The detail of the iterative algorithm is as follows:

- In iteration i, given N_{i-1}, θ_{i-1} and the original incomplete dataset D^0.

  In E step, the missing values and the values of hidden variables of D^0 are filled up based on the inference of N_{i-1} and $\theta_{i-1.}$ Then, we obtain a complete dataset $D_{i-1.}$ The structure N_i is learnt based on D_{i-1} by maximizing $Score(Ni: \theta_{i-1}, D_{i-1})$. Because of the decomposability, we learn the parents for each gene g_i independently:

Table 1
An example of filling missing values

g_1	g_2	g_3
(a)		
()	1	1
1	0	1
1	1	1
0	1	1
(b)		
(0.4, 0.6)	1	1
1	0	1
1	1	1
0	1	1

(1) Find k genes with best MI score with respect to g_i to build the candidate parent set *CPS*.
(2) Find parents $PS \in CPS$ which maximum $Score(SS_i: \theta_{i-1}, D_{i-1})$, where SS_i is a substructure $PS \rightarrow h_i \rightarrow g_i$.
– repeat the procedure until converging.

- In M step, θi is learnt based on Ni which maximum Score(N_i: θ_i, D_{i-1})
- Repeat the procedure until convergence or till the predefined number of iteration is reached.

In the above procedure, the network structure and parameters are refined iteratively. To get the initial network structure N_0 and parameters θ_0:

- learn the D^0 as a complete dataset. Pa^i for each g_i and the dependent probabilities are gotten.
- for each gene g_i, add a hidden variable h_i. Set the $P\ a(h_i) = Pa^i$ and $P(h_i|Pa(h_i)) = P(gi|Pa^i)$.
- A complete dataset D^0 by filling D^0 based on the network structure and dependent probabilities is obtained.
- N_0 and θ_0 can be learned from D^0.

4. Experimental results and comparison

The model and the algorithm are implemented in MATLAB and BNT[2]. Below, we present experimental results on both artificial and real life gene expression datasets.

4.1. Experiment on artificial datasets

An artificial dataset is generated by simulating the gene expression with hidden variables. The artificial network contains 5 genes and 4 edges (as shown in Fig. 2(a)). We built a network containing 5 genes, 5 proteins and 2 protein combinations whose topology is similar to Fig. 1(a) and assign parameters randomly. Then we generated the dataset with 150 replicates and delete the data of the proteins and protein combinations.

We compare the learning results with traditional learning algorithms K2 (a Bayesian network learning algorithm (Cooper & Herskovits, 1992)) and structural EM (SEM) (an algorithm in learning Bayesian network with hidden variables (Friedman, 1998)). K2 is a learning algorithm to learn the complete dataset. SEM allows hidden variables. However, it does not define either the number of hidden variables or any prior information of the relationships of hidden variables and observed variables. Our algorithm *SSEM* allows hidden variables. Moreover, the hidden variables are semi-fixed as stated in previous sections.

The comparison for artificial dataset is shown in Fig. 2. K2 and SEM recovered 0 and 1 correct edges, respectively, together with 2 false edges for each while SSEM recovered all edges together with 2 false positives. Hence, SSEM results in more correct edges as compared to K2 and SEM. Moreover,

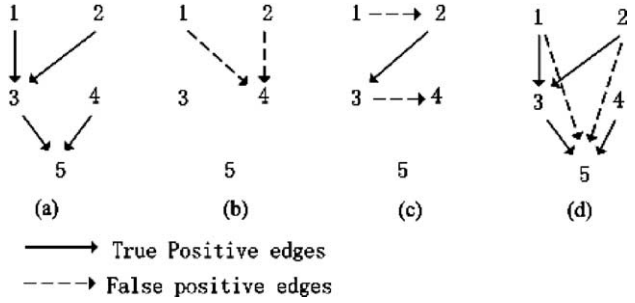

Fig. 2. Leaning performance comparison in the artificial dataset. (a) Structure of the artificial network, which contains 4 genes and 4 regulatory edges. (c) Structure learned by *K*2, there is no correct edge. (b) Structure learned by *SEM*. There is only 1 correct edge. (d) Structure learnt by *SSEM*. All edges are uncovered.

the two false positive parents of the variable 5 are the ancestors of the variable 5. Each of them is in a pathway with the variable 5. Not like the false edges in Fig. 2(b) and (c), they contain some regulatory information though they are false edges.

4.2. Experiments on real-life data

The microarray gene expression data for *S. Cerevisiae* contains 76 gene expression measurements (Spellman, Sherlock, & Futcher, 1998). To simplify the presentation and implementation, the expression levels were discretized to discrete values 0 and 1: a value is discretized to 0 if it is smaller than 0 and 1 otherwise. The theoretical support behind the binary discretization of gene expression level is that the genes are most of the time either maximally expressed or virtually not expressed (Louis & Becskei, 2002) ensuring that the binary gene regulatory network is biologically meaningful. The real-life gene network in our work is a Yeast transcriptional cell cycle subnetwork published in (Futcher, 2002), which includes 13 genes and 18 edges. It is a good example to demonstrate the learning power since the subnetwork is the backbone part of Yeast cell cycle gene regulatory network (Simon et al., 2001), which is robustly designed (Li, Long, Lu, Ouyang, & Tang, 2004), and the genes here are the most important transcription factors in Yeast cell cycle (Li et al., 2004; Simon et al., 2001). The directed pairwise regulatory relationships for the subnetwork are verified by YPD (Yeast Proteome Database)[3] (Hodges, McKee, Davis, Payne, & Garrels, 1999) and (Futcher, 2002), as shown in Fig. 3(a), and the cell cycle regulations are shown in Fig. 3(c). Since some time delays exist in the regulation system (Liu, Sung, & Mittal, 2004), a gene in time slice i may regulate another gene in time slice i to $i+k$ (k indicates the maximum time delay). Thus, we learn the edges in a k-window, i.e. the consecutive k time slices (Boyen, Friedman, & Koller, 1999). An approach based on k *-DBN* (Boyen et al., 1999) is implemented for the comparison purpose.

Besides the direct regulatory relationship between genes, we also employed a popular statistic feature, the Markov relation. A

[2] Bayes network toolbox, http://www.ai.mit.edu/~murphyk

[3] http://www.proteome.com/YPDhome.html

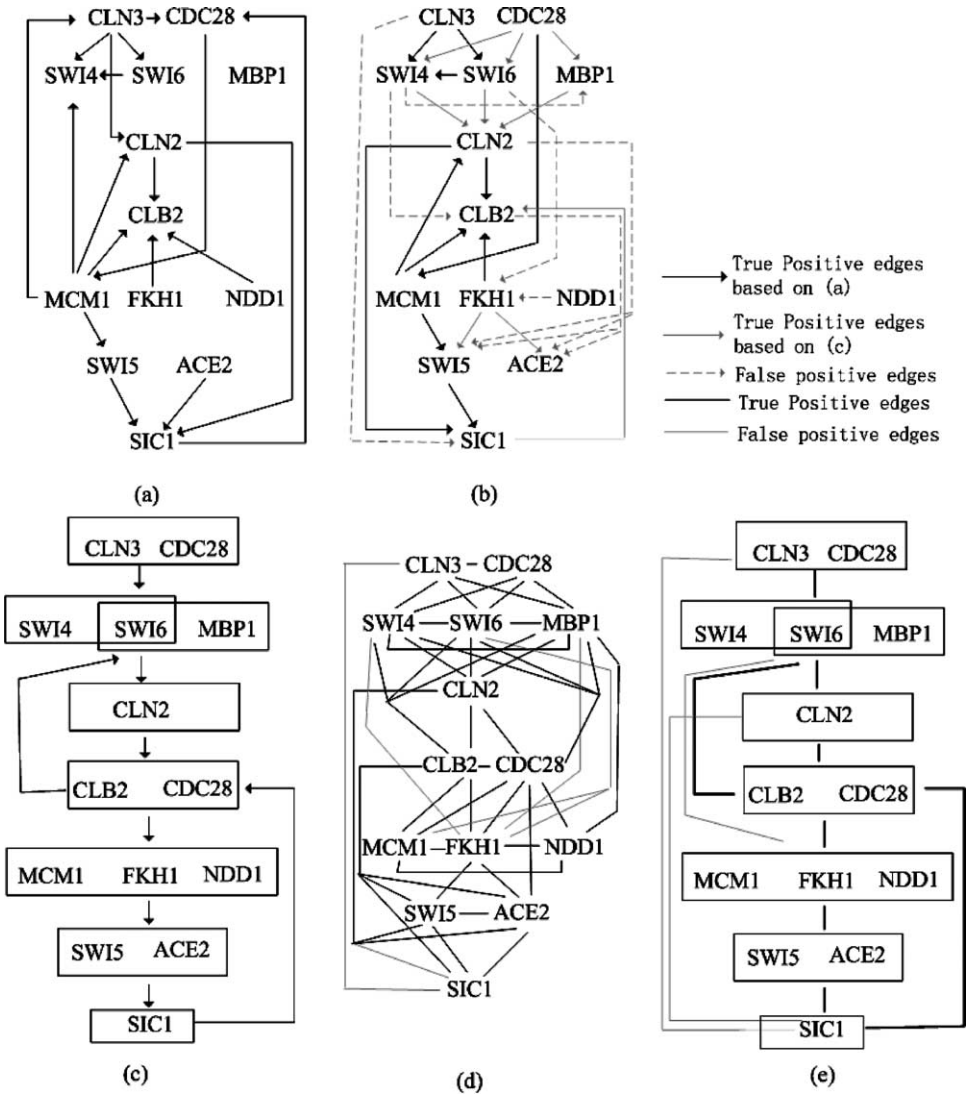

Fig. 3. Leaning performance of *SSEM* in real-life gene network. (a) The original subnetwork structure, which contain 13 genes and 18 edges. (b) The structure learnt by *SSEM*. There are sum 29 edges with confidence no smaller than 0.6. Among them, 11 edges are verified as true positives by (a). Considering (c), there are 9 more edges are regarded as true positives. (c) The Yeast cell cycle regulatory network. The genes in a box is a transcription factor or a complex. (d) Markov features with confidence no smaller than 0.6 learned by *SSEM*. There are 44 Markov features. Among them 38 can be verified by (a) and (c). (e) Learned cell cycle regulatory network which is simplified from (d).

Markov relation (Friedman, 2004; Friedman et al., 2000) describes whether a gene Y is in the Markov blanket of another gene X (denoted by $Y \in MB(X)$). $Y \in MB(X)$ if and only if there is either an edge between them or both are parents of another variable. It indicates whether two genes are involved in the same biological event and it has been regarded as a criterion by several works to evaluate the performance of gene network reconstruction (Friedman, 2004; Friedman et al., 2000). Bayesian network is a model of dependencies among random variables, rather than causality. $A \rightarrow B$ and $B \rightarrow A$ are the alternative ways of describing that A and B are not independent on each other. Markov relation is helpful in discovering the dependencies without caring about the directions of the edges. Moreover, in gene regulatory system, there are a lot of transcription factors and transcription complexes consisting of two or more proteins working collaboratively. Such collaborations cannot be uncovered by directed regulatory edges but can be discovered by the Markov feature.

Given the insufficient dataset, Bayesian network may give a set of models, which explain the data equally well (Pe'er, Regev, Elidan, & Friedman, 2001). In order to do further analysis, we used the statistical confidence to measure the likelihood of a learned edge or a statistic feature (Friedman, 2004; Friedman et al., 2000), which can be calculated by a bootstrap approach as below:

- For $i = 1 \ldots m$:
  - Generate a 'perturbed' version of the input transformed dataset by re-sampling q instances where q is smaller than the number of instances of the transformed dataset.
  - Apply the learning algorithm on the perturbed dataset and induce a network N_i.

- For each Markov relation f, the confidence is calculated as follows:

$$\mathrm{conf}(f) = 1/m \sum_i f(N_i) \quad (7)$$

where $f(N_i)$ is 1 if f is a feature in N_i and 0 otherwise.

We initialized q to be 35 and m to be 50 in our experiment. A high confidence value of a feature for two genes indicates that the learning algorithm can consistently recover the relationship between them.

The effectiveness of semi-fixed structure EM learning algorithm can be seen as compared to *k-DBN* algorithm as SSEM takes advantage of its semi-fixed structure and effective learning algorithm.

In the final results, we only adopt the edges whose confidence value is not less than 0.6. The comparison is shown in Table 2 and Fig. 3(b) where only the original structure and the structure learnt by SSEM are presented. K2, *SEM* and *K-DBN* gave 1, 4 and 5 edges, respectively, while SSEM gave 11 edges based on the knowledge of YPD (Fig. 3(a)). K2, *SEM*, *K-DBN* and *SSEM* gave 11, 12, 22 and 18 false positive edges, respectively. Fig. 3(b) shows the 29 edges learned by *SSEM* with confidence greater than 0.6. As shown in following discussion, some of the false positive edges of the 18 edges learnt by SSEM are meaningful in terms of the regulatory relationships between transcription factors or complexes instead of between genes.

As shown in Fig. 3(c), *CLN*3–*CDC*28 kinase activate transcription factors, *SBF* (comprised by *SWI*4 and *SWI*6 and *MBF* (comprised by *SWI*6) and *MBP* 1), to begin the cell cycle. *SBF* and *MBF* then activate about 200 genes in late G1 and S phase include *NDD*1 and *CLN*2. *CLN*2 is one of the important factors to activate *CLB*2–*CDC*28 kinase. The complex *CLB*2–*CDC*28 inactivates *SBF* and *MBP* to shut off G1/S events and the cell goes to G2 phase. *CLB*2–*CDC*28 continue to activate a transcription factor containing *MCM*1, *FKH*1 (or *FKH*2, in the experiment, *FKH*2 is not included as it has too many missing values in the dataset.) and *NDD*1. This factor activates a set of genes include *SWI*5 and *SCE*2 which activate *SIC*1. *SIC*1 and some other genes inhibit *CLB*2-*CDC*28. It is the yeast cell cycle transcriptional regulatory network, which is essential to yeast development and differentiation. For the detail of the genes and the cell cycle regulatory network, please refer to (Simon et al., 2001).

Table 2
Comparison of learning performances on real-life dataset

Algorithm	Yeast gene subnetwork		
	TE	TP1	TP2
K2	12	1	3
SEM	16	4	7
k-DBN	27	5	9
SSEM	29	11	20

TE (Total learnt edges) indicates the number of edges the algorithm learned from the dataset, TP1 indicates the number of true positives verified by Fig. 3(a) and TP2 indicates the number of true positives verified by Fig. 3(a) and (c).

When a gene (or a complex) activates or inhibits a transcription complex, it is possible that there is no directed edge from the regulator to any gene of the complex and vice versa. For example, for *CDC*28, it activates both *SWI*4-*SWI*6 and *SWI*6-*MBP* 1 when it forms complex with *CLN*3 while it inhibits them when it forms complex with *CLB*2. Whereas, there is no direct edge from *CDC*28 to *SWI*4, *SWI*6 and *MBP*1 in YPD. The use of the hidden variables in our model ensures that we can find such regulatory relationships. As shown in Fig. 3(b), with the verification by Fig. 3(c), 10 more edges are proved biological meaningful in Fig. 3(b), such as *CDC*28 → *SWI*4, *CDC*28 → *SWI*6, *CDC*28 → *MBP*1, *SWI*4 → *CLN*2, *SIC*3 → *CLB*2, etc. Using this criterion, *K*2, *SEM*, *k-DBN* and *SSEM* got 3, 7, 9 and 20 correct edges and 9, 9, 18 and 9 false positive edges, respectively.

We listed the Markov features with confidence greater than 0.6, as shown in Fig. 3(d). There are total 44 Markov features, in which 38 can be verified by Fig. 3(a) and Fig. 3(c). It is clear that there are strong inter-actions among the genes comprising a complex and between complexes which are connected in Fig. 3(c) . We then simplify the network by using the transcription factors or complex, which play roles as a unit in the regulatory system, as the node and the Markov relations between them as the undirected edges, as shown in Fig. 3(e). We completely discover the cell cycle regulatory network, together with 3 false edges between *CLN*3–*CDC*28 & *SIC*1 (which is weak since between them there are only one Markov feature between *CLN*3 and *SIC*1), *CLN*2 & *SIC*1 and *SBF* /*MBF* & *SWI*5-*ACE*2.

5. Conclusion

The semi-fixed hidden variable model introduces hidden variables to model the important components of gene network, i.e. regulatory proteins. The model makes the network decomposable and the parts of the gene network are fixed using biological knowledge. Compared to the current work on gene network, we integrate the biological knowledge to build the semi-fixed hidden variable model, which is effective and reflects the real life system. It increases the learning performance as well as finds several regulatory relationships, which are difficult to be discovered by employing traditional methods. As our model can model protein complex which plays key role in a lot of biological systems, we foresee our model can be applied to other applications in computational biology. In addition to the model, this paper presents an effective learning algorithm to learn our model. The main disadvantage of our model is that it needs more computing resource to model the hidden variables and it requires more iterations to converge. As the model is based on the Bayesian network framework which can be used to build an probabilistic expert system (Castillo, Gutierrez, & Hadi, 1997; EZ, Mira, Iturralde, & Diaval, 1997), it is easy to develop an probabilistic expert system by making use of the learned network structure and parameters to predict the outcome of the perturbation of the organism, such as diseases. This potential application makes the consequent works meaningful. In the future, we would like to further reduce the learning complexity by utilizing more

biological facts. Those biological facts include: (1) regulatory proteins are usually much less than the total proteins, and (2) it is common for several genes to share the same parent set.

References

Akutsu, T., Miyano, S., & Kuhara S. (1999). Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In *PSB* (pp. 17–28).

Arkin, A., Shen, P., & Ross, J. (1997). A test case of correlation metric construction of a reaction pathway from measurements. *Science*, *277*, 1275–1279.

Barbara, G., Jameson, A., & Witting, F. (1999). Learning Bayesian networks with hidden variables for user modeling. In *Proceedings of the IJCAI 99 workshop 'learning about users'* (pp. 29–34).

Barrientos, M. A., & Vargas, J. (1998). A framework for the analysis of dynamic processes based on Bayesian networks and case-based reasoning. *Expert Systems with Applications*, *15*, 287–294.

Boyen, X., Friedman, N., & Koller, D. (1999). Discovering the hidden structure of complex dynamic systems. In *UAI* (pp. 91–100).

Brazhnik, P., Fuente, A., & Mendes, P. (2002). Gene networks: How to put the function in genomics. *Trends in Biotechnology*, *1*(20), 467–472.

Castillo, E., Gutierrez, J., & Hadi, A. (1997). *Expert systems and probabilistic network models*. New York: Springer.

Chen, T., Filkov, V., & Skiena, S. (1999). Identifying gene regulatory networks from exprimental data. In *PSB* (pp. 94–103).

Chen, T., He, H., & Church, G. (1999). *Modeling gene expression with differential equations* (pp. 29–40).

Cooper, G., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*, 309–347.

Cumiskey, M., Levine, J., & Armstrong, D. (2003). *Gene network reconstruction using a distributed GA with a backprop local search* (pp.33–43).

D'haeseleer, P., Liang, S., & Somogyi, R. (2000). Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, *16*(2), 707–726.

D'Haeseleer, P., Wen, X., Fuhrman, S., & Somogyi, R. (1999). Linear modeling of mRNA expression levels during CNS development and injury. In *PSB* (pp. 41–52).

Elidan, G., Lotner, N., Friedman, N., & Koller, D. (2000). Discovering hidden variables: A structure-based approach. In *NIPS* (pp. 479–485).

EZ, F., Mira, J., Iturralde, E., & Diaval, Z. (1997). A Bayesian expert system for echocardiography. *Artificial Intelligence in Medicine*, 59–73.

Friedman, N. (1997). Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the 14th international conference on machine learning* (pp. 125–133).

Friedman, N. (1998). The Bayesian structure EM algorithm. In UAI (pp. 129–138)

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science*, *303*(6), 799–805.

Friedman, N., Linial, M., Nachman, I., & Pe'er, D. (2000). Using Bayesian networks to analyze expression data. In *RECOMB* (pp. 127–135).

Friedman, N., Nachman, I., & Peer, K. (1999). Learning Bayesian network structure from massive datasets: The 'sparse candidate' algorithm, In *UAI* (pp. 206–215).

Futcher, B. (2002). Transcriptional regulatory networks and yeast cell cycle. *Current Opinion in Cell Biology*, *14*, 676–683.

Gustavo, A., Sucar, L., & Villavicencio, A. (1998). Probabilistic temporal reasoning and its application to fossil power plant operation. *Expert Systems with Applications*, *15*, 317–324.

Hartemink, A., Gifford, D., Jaakkola, T., & Young, R. (2001). Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks, In *PSB* (pp. 437–449).

Hasty, J., McMillen, D., Isaacs, F., & Collins, J. (2001). Computational studies of gene regulatory networks: In numero molecular biology. *Nature Reviews Genetics*, *2*(4), 268–279.

Hodges, P., McKee, A., Davis, B., Payne, W., & Garrels, J. (1999). The yeast proteome database (YPD): A model for the organization and presentation of genome-wide functional data. *Nucleic Acids Research*, *27*(1), 69–73.

Imoto, S., Goto, T., & Miyano, S. (2002). Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression. In *PSB* (pp. 175–186).

Imoto, S., Higuchi, T., Goto, T., Tashiro, K., Kuhara, S., & Miyano, S. (2003). Combining microarrays and biological knowledge for estimating gene networks via Bayesian network. In *CSB* (pp. 104–113).

Kang, C. W., & Gola, M. W. (1999). A Bayesian belief network-based advisory system for operational availability focused diagnosis of complex nuclear power systems. *Expert Systems with Applications*, *17*(1), 21–32.

Kolpakov, F., Ananko, E., Kolesov, G., & Kolchanov, N. (1998). A gene network database and its automated visualization. *Bioinformatics*, *14*, 529–537.

Li, F., Long, T., Lu, Y., Ouyang, Q., & Tang, C. (2004). The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences USA*, *101*(14), 4781–4786.

Liang, S., Fuhrman, S., & Somogyi, R. (1998). Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *PSB* (pp. 18–20).

Liu, T. F., Sung, W. K., & Mittal, A. (2004). Learning multi-time delay gene network using Bayesian network framework. In *ICTAI* (pp. 640–645).

Louis, M., Louis, A., & Becskei (2002). Binary and graded responses in gene networks. *Science STKE*, *143*, pe33.

Murphy, K., & Mian, S. (1999). *Modelling gene expression data using dynamic Bayesian networks.* Technical report, University of California, Berkeley.

Oatley, G., & Ewart, B. (2003). Crimes analysis software: 'Pins in maps', clustering and bayes net prediction. *Expert Systems with Applications*, *25*, 569–588.

Pe'er, D., Regev, A., Elidan, G., & Friedman, N. (2001). Inferring subnetworks from perburbed expression profiles. *Bioinformatics*, *17*(Suppl. 1), S215–S224.

Segal, E., Shapira, M., Regev, A., Pe'er, D., Botstein, D., & Koller, D. (2003). Module networks: Identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics*, *34*(2), 166–176.

Simon, I., Barnett, J., Hannett, N., Harbison, C., Rinaldi, N., Volkert, T., et al. (2001). Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, *106*(6), 697–708.

Someren, E., Wessels, L., & Reinders, M. (2000). Linear modeling of genetic networks from experimental data. In *ISMB* (pp. 355–366).

Spellman, P., Sherlock, G., & Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, *9,*(3273), 3297.

Sucar, L. E., & Miriam, M. A. (1998). Interactive structural learning of Bayesian networks. *Expert Systems with Applications*, *15*(3–4), 325–332.

Viaene, S., Dedene G., & Derrig, R. A. (in press). Auto claim fraud detection using Bayesian learning neural networks. *Expert Systems with Applications*.

Wagner, A. (2002). Estimating coarse gene network structure from large-scale gene perturbation data. *Genome Research*, *12*(2), 309–315.

Wahde, M., & Hertz, J. (2000). Course-grained reverse engineering of genetic regulatory networks. *Biosystems*, *55*, 129–136.

Watanabe, S. (1998). Algorithms for inference of genetic networks AIGNET. In *International workshop on genome informatics* (pp. 274–275).

Wessels, L., Someren, E. V., & Reinders, M. (2001). A comparison of genetic network model. In *PSB, Vol. 6* (pp. 508–519).

Wyrick, J., & Young, R. (2002). Deciphering gene expression regulatory network. *Current Opinion in Genetics and Development*, *12*, 130–136.

Yaki, M., Tominaga, D., Okamoto, M., Watanabe, S., & Eguchi, Y. (2001). Development of a system for the inference of large scale genetic networks. In *PSB* (pp. 446–458).