Available online at www.sciencedirect.com

ScienceDirect

NEUROCOMPUTING

Neurocomputing 71 (2008) 600–610

www.elsevier.com/locate/neucom

A clustering-based approach for inferring recurrent neural networks as gene regulatory networks

Wei-Po Lee[a,*], Kung-Cheng Yang[b]

[a]Department of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan
[b]Department of Management Information Systems, National Pingtung University of Science and Technology, Pingtung, Taiwan

Available online 1 October 2007

Abstract

Constructing genetic regulatory networks is one of the most important issues in system biology research. Yet, building regulatory models manually is a tedious task, especially when the number of genes involved increases with the complexity of regulation. To automate the procedure of network construction, in this work we establish a clustering-based approach to infer recurrent neural networks as regulatory systems. Our approach also deals with the scalability problem by developing a clustering method with several data analysis techniques. To verify the presented approach, experiments have been conducted and the results show that it can be used to infer gene regulatory networks successfully.

Keywords: Gene regulatory network; Recurrent neural network; Reverse engineering; System modeling

1. Introduction

With the newly invented experimental tools and techniques, it is now possible to simultaneously measure and record information of multiple genetic reactors and their interactions. By analyzing and studying the interactions between genes in a regulatory network, we can now uncover some complex behavior patterns [19]. Gene network modeling provides a methodology to manipulate time series data for building a model that can describe the observed phenotypic behavior of a system to be studied. But it should be noted that gene regulation networks (GRNs) are complex biological systems consisting of many interacting components. In many cases, the detailed molecular mechanism that governs the interactions among system components is not yet fully understood. Generally, it is difficult to model these complex processes mathematically, because the networks involve many genes connected through interlock feedback loops and their description requires a representation general enough to capture the characteristics of experimentally observed responses. Also the model must be robust against noises and can be enforced by some constraints on the connectivity, stability, and redundancy.

Many gene regulation models have been proposed [7,8]. The models can range from very abstract models (involving Boolean values only) to very concrete ones (including fully biochemical interactions with stochastic kinetics). The former approach is mathematically tractable that provides the possibility of examining large systems; but it cannot infer networks with feedback loops. The latter approach is more suitable in simulating the biochemical reality and is more realistic to the experimental biologists. However, due to its computational complexity, this approach can only be applied to very small systems. To reconstruct a gene regulatory network, one can start from an initial model, simulate the system behaviors for a variety of experimental and environmental conditions, and then compare the predictions with the observed gene expression data to give an indication of the adequacy of the model. If the experimental data is considered reliable, but the observed and predicted system behavior does not match the data, the model must be revised. The activities of constructing and revising models of the regulatory network, simulating the behavior of the system, and testing

*Corresponding author.
E-mail address: wplee@mail.nsysu.edu.tw (W.-P. Lee).

0925-2312/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.07.023

the resulting predictions are repeated until an adequate model is obtained.

As the above procedure of constructing a network from data takes a considerable large amount of time, an automated procedure is thus advocated. Reverse engineering is a paradigm with great promise for analyzing and constructing GRNs [5,13]. It is an effective way to utilize either literature or laboratory experimental data to determine the underlying network structure of a given model. To reconstruct a network, experimental iterations and priori knowledge are required until sufficient data is available to perform computational inference of the network structure. In the case of GRNs, the experiment involves altering the gene network in some way, observing the outcome, and using mathematics and logic to infer the underlying principles of the network. It can thus be concluded that the key issues of this process lie in selecting network model and fitting network parameters and structures into the available data.

Neural networks have been used in recognizing patterns in a variety of fields including genetics studies [4,18]. They are biological plausible and noise resistance. A special kind of neural networks, namely recurrent neural networks (RNNs), considers the feedback loops and takes internal states into account. It is able to show the system dynamics of the network over time. With such unique characteristics, this kind of network thus offers an ideal model to work as GRNs. The learning algorithms associated with RNNs can also be used to solve the abovementioned data-fitting problem. Therefore, in this work we establish an approach that takes RNNs to represent GRNs and also exploits the network learning algorithms to reconstruct regulatory systems from collected expression data. In order to deal with the scalability problem, a clustering method with some data analysis techniques for feature extraction is applied to develop GRNs hierarchically. To verify the presented approach, three series of experiments have been conducted to demonstrate how it works.

2. Modeling gene regulatory networks

Models for gene regulatory networks can mainly be categorized into two types that use discrete and continuous variables, respectively [7,8]. The first type of GRN models assumes that genes only exist in discrete states. This approximation is usually implemented by Boolean variables in which the gene is in either on or off state. Boolean networks are easy to simulate in a cheaper computational cost, but it has been proven that Boolean networks are not able to capture some system behaviors that can only be observed on continuous models [7]. Also the dynamics of a Boolean network are deterministic and they depend on the initialization of the node states [15]. These unexpected features make the Boolean network model not realistic. Another popular discrete variable model is Bayesian network that explicitly establishes probabilistic relationships between nodes [9,11,16]. Bayesian models have rich statistics and probability semantics, but learning network structure for such models is computationally expensive. In addition, Bayesian models are inherently static. As the directed network graphs are acyclic by definition, there can be no auto-regulation and no time-course regulation. Some methods have been proposed to solve this problem but the computational complexity will be increased significantly.

Unlike the above discrete variables models, the other type of models uses continuous variables. One of the popular continuous variables models is the one based on differential equations that can describe more accurately the system dynamics of a GRN. Many models based on differential equations have been proposed, including the traditional linear ordinary differential equations and the non-linear power law ones [7]. The well-researched model S-system is a kind of power law model. It consists of a particular set of ordinary differential equations in which the component processes are characterized by power law functions. Compared to the discrete variables models, the differential equations models can more accurately represent the underlying physical phenomena due to its continuous variables. In addition, there are many theories of system analysis and of control design on dynamical systems to support this type of models. Especially, with the non-linear feature of ordinary differential equations (such as S-system), steady-state evaluation, control analysis, and sensitivity analysis of a given system can be established mathematically [23]. But it should be noted that the non-linear ordinary differential equations are hard to solve. It is too difficult for the traditional optimization methods to estimate all the large number of parameters involved in a GRN.

The other commonly used continuous variables model is the neural network-based model, among which the RNNs are the most successful ones [4,22]. This model is biologically plausible and noise resistance. It is continuous in time, and uses a transfer function to transfer the inputs into a shape close to the one observed in natural processes. Also its non-linear characteristics provide information about the principles of control and natural interactions of elements of the modeled system. Furthermore, this model can record and consider its internal states in its operating process, and present the dynamical system behavior over time. It is thus a suitable candidate for modeling time series data.

As is analyzed above, different models have been proposed to simulate GRNs, and computational methods have also been developed to reconstruct networks from the expression data correspondingly. More details can be found in the review work (for example, Refs. [7,8]). From these literatures, it can be seen that work in modeling GRNs shared similar ideas in principle. They improved the performance of network model over iterations and have shown the preliminary results to support the idea of reverse engineering. However, depending on the research motivations behind the work, different researchers explored the same topic from different points of view; thus the implementation details of individual work are different.

Instead of subjectively arguing which approach is better to offer for network reconstruction, our work here focuses on investigating whether the presented approach, in practice, can be used to model GRNs and on how to develop complex networks. In this work, we adopt the RNN model. To simulate the GRNs and make it to learn networks from expression data, we develop a software system that includes an enhanced version of RNN learning algorithm, coupled with some heuristic techniques of data processing to improve the learning performance. The details are presented in the following sections.

3. Developing RNNs as gene regulatory networks

The behavior expressions of a gene regulatory network are in fact, coordinated patterns of activity in time and space. Therefore, GRNs can be regarded as dynamical systems that are perturbed by their interaction with the environment. To have such characteristic, it is important that the chosen network model for modeling gene expression data must be able to produce abundant intrinsic dynamical behavior. RNNs are appropriate choices working as GRNs, because they have been shown to operate as dynamical systems that do not explicitly perform input–output mappings as other computational mechanisms. RNNs have recurrent connections that provide the possibility of generating oscillatory and periodic activities. The complex activities can be coordinated in the time domain, and the network behaviors can be governed by a set of differential equations.

One class of RNNs, namely dynamical neural networks, has the advantage of being able to generate temporally continuous responses [2,17]. In principle, the states of such mechanisms are the same as that of finite-state machines. Because dynamical neural networks maintain states, their responses to identical environmental stimuli can differ at different time points. Internal state is the key point that allows this kind of networks to oscillate when no external feedback appears, and it does not correspond to any particular element in addition to the concentrations of reactants. Compared to discrete-time neural networks that switch between arbitrary points in their state spaces from one discrete-time step to the next step, continuous-time networks can furthermore change their states through time smoothly. As the current microarray method normally collects experimental data in discrete-time points, we thus adopt discrete-time neural networks as genetic regulatory networks. They can be extended to continuous-time dynamical neural networks without introducing significant problems, though the latter are more general and powerful class of asynchronous networks and are not easy to implement.

3.1. Network model

There are several RNN architectures, ranging from restricted classes of feedback to full interconnection between nodes. Vohradsky and colleagues have proposed the use of fully RNN architecture in studying regulatory genetic networks such as those involved in the transcriptional and translational control of gene expression [4,24]. In this work, we also take the same architecture to model GRN, but unlike their work that mainly simulates regulatory effects, our goal is to reconstruct regulatory networks from expression data measured.

In a fully recurrent net, each node has a link to any node of the net, including itself. Using such a model to represent a GRN is based on the assumption that the regulatory effect on the expression of a particular gene can be expressed as a neural network in which each node represents a particular gene, and the wiring between the nodes define regulatory interactions. When activity flows through the network in response to an input, each node influences the states of all nodes in the same net. If the connection weight is positive, it means that the sending node tries to put the receiving node into the same state of activity as itself. On the contrary, if the connection weight is negative, it tries to put the receiving node into the opposite state. Since all activity changes are determined by these influences, each input can be seen as setting constraints on the final state that the system can settle into. When the system operates, the activities of individual nodes will change in order to increase the number of constraints satisfied. The ideal final state would be a set of activities for the individual nodes where all the constraints are satisfied. The network would then be stable because no node would be trying to change the state of any of the nodes to which it is connected. Similarly, in a GRN, the level of expression of genes at time t can be measured from a gene node, and the output of a node at time $t+\Delta t$ can be derived from the expression levels and connection weights of all genes connected to the given gene at the time t. That is, the regulatory effect to a certain gene can be regarded as a weighted sum of all other genes that regulate this gene. Then the regulatory effect is transformed by a sigmoidal transfer function into a value between 0 and 1 for normalization.

The same set of the above transformation rules is applied to the system output in a cyclic fashion until the input does not change any further. As in Ref. [22], here we use the basic ingredient to increase the power of empirical correlations in signaling constitutive regulatory circuits. It is to generate a network with nodes and edges corresponding to the level of gene expression measured in microarray experiments, and to derive correlation coefficients between genes. To calculate the expression rate of a gene, the following transformation rules are used:

$$\frac{\mathrm{d}y_i}{\mathrm{d}t} = k_{1,i}G_i - k_{2,i}y_i$$

and

$$G_i = \left\{ 1 + \mathrm{e}^{-\left(\sum_j w_{i,j}y_j + b_i\right)} \right\}^{-1}$$

where y_i is the actual concentration of the ith gene product; $k_{1,i}$ and $k_{2,i}$ are the accumulation and degradation rate constants of gene product, respectively; G_i is the regulatory effect on each gene that is defined by a set of weights (i.e., $w_{i,j}$) estimating the regulatory influence of gene j on gene i, and an external input b_i representing the reaction delay parameter.

3.2. Learning algorithm

After the network model is decided, the next phase is to find settings of the thresholds and time constants for each neuron, and the weights of the connections between the neurons so that the network can produce the most approximate system behavior (i.e., the measured expression data). By introducing a scoring function for network performance evaluation, the above task can be regarded as a parameter estimation problem with the goal of maximizing the network performance (or minimizing an equivalent error measure). To achieve this goal, here we use the backpropagation through time (BPTT, [26]) learning algorithm to update the relevant parameters of recurrent networks in discrete-time steps.

Instead of mapping a static input to a static output as in a feedforward network, BPTT maps a series of inputs to a series of outputs. The central idea is the "unfolding" of the discrete-time recurrent neural network (DTRNN) into a multi-layer feedforward neural network when a sequence is processed. It is to use time units to index layers in a feedforward network, and approximate the gradients of the weights in the recurrent network by using a feedforward network with a fixed number of layers. Each layer t contains all activations of the recurrent network at time step t. In this way, to calculate an exact gradient for an input pattern sequence of length l, the feedforward network needs $(l+1)$ layers if an output pattern is desired after the last pattern of the input sequence is shown to the network. Fig. 1 shows a typical example of unfolding a RNN.

Once a DTRNN has been transformed into an equivalent feedforward network, the resulting feedforward network can then be trained using the standard backpropagation algorithm. But it should be noted that because layers have been obtained by replicating the DTRNN, the weights between successive layers have to be identical. To satisfy this restriction, BPTT updates all equivalent weights using the sum of the gradients obtained for weights in those replicated layers, which is equal to the gradient of the error function for the DTRNN. More details on BPTT are referred to in Refs. [10,26].

The goal of BPTT is to compute the gradient over the trajectory and update network weights accordingly. As mentioned above, the gradient decomposes over time. It can be obtained by calculating the instantaneous gradients and accumulating the effect over time. In BPTT, weights can only be updated after a complete forward step during which the activation is sent through the network and each processing element stores its activation locally for the entire length of the trajectory. That is, as in the traditional backpropagation algorithm, the weights are updated in the backward step according to the following rules [26]:

$$\Delta w_{i,j} = -\eta \frac{\partial E(t_0, t_1)}{\partial w_{i,j}} = \eta \sum_{\tau=t_0}^{t_1} \delta_j(\tau) x_i(\tau - 1)$$

and

$$\delta_j(\tau) = \begin{cases} f'(\mathrm{net}_j(\tau)) e_j(\tau) & \text{if } \tau = t_1 \\ f'(\mathrm{net}_j(\tau)) \left[e_j(\tau) + \sum\limits_{\ell \in U} \delta_\ell(\tau + 1) w_{i,\ell} \right] & \text{if } t_0 \leqslant \tau < t_1 \end{cases}$$

where δ_j is the derivative of the error surface, η is the learning rate, e_j means the corresponding error, and U is the set of all output units.

To avoid deep networks for long sequences, it is possible to use only a fixed number of layers to store the activations back in time. The number of additional copies of network activations is controlled by a specific parameter. As the expression data set obtained from the microarray experiments contains typically only a few time points, it is not necessary to restrict the number of layers in particular. Therefore, in our modeling experiments presented in Section 4, a RNN is unfolded as a multi-layer feedforward network according to the actual time points available.

In the above learning procedure, learning rate is an important parameter, but it is difficult to choose an appropriate value to achieve an efficient training because the cost surface for multi-layer networks can be complicated

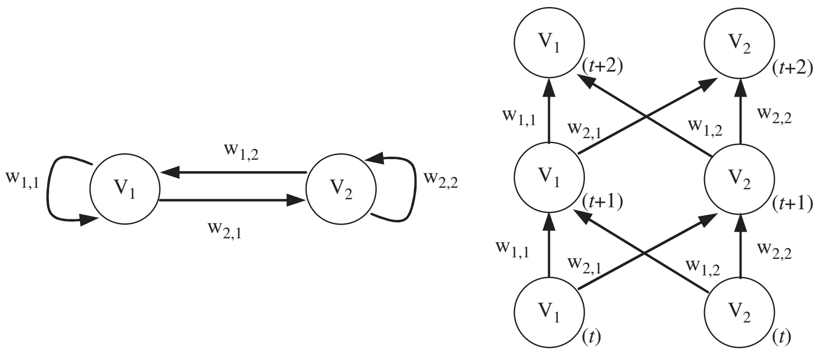


Fig. 1. The unfolding of a two nodes recurrent neural network.

and what works in one location of the cost surface may not work well in another location. Delta-bar-delta is a heuristic algorithm for modifying the learning rate in the training procedure [12]. It is inspired by the observation that the error surface may have a different gradient along each weight direction so that each weight should have its own learning rate. In our modeling work, to save the effort in choosing appropriate learning rate, this algorithm is implemented for automatic parameter adjustment.

3.3. Network robustness

In addition to the above two issues of model selection and parameter fitting, network robustness is a critical factor to be taken into account in GRN modeling. Robustness means the ability of a system to maintain function in the changing environment (e.g., the presence of invalid or conflicting inputs). It is difficult to measure, but nevertheless an important concept to be considered. In the process of inferring GRNs from the measured set of gene expression, the target networks may be perfectly constructed (or learnt). But it must be noted that the data samples collected at different time points from the microarray experiments are very limited, and which often causes the over-fitting problem. That is, the models perfectly fitting the training data generally give poor performance in prediction. Also as is known, gene expression measurements are unreliable; they always contain a certain amount of noises. Under such circumstances, the obtained networks are not robust as expected.

As previously mentioned, a recurrent network has internal states and it works as a dynamical system. The set of possible states into which the network can settle are the attractors. A sequence of time series data with T time points can thus be regarded as consecutive T-1 state transition in the state space. In a RNN, the time the network takes to settle into one of the states depends on the shape of the basin of attraction and the position within the basin where the input landed. As the activity level of genes in a GRN changes smoothly over time, the corresponding state transition in the RNN state space can be expected to be stable. Hence, RNNs are relatively robust to noisy input. Provided an input falls within a basin, the exact position does not matter because the response will eventually be the same (i.e., the state corresponding to the attractor).

Apart from using the characteristics of the chosen model, to infer a robust gene network (i.e., insensitive to noises), the criterion of adding noises to training data is proposed [21]. In fact, it has been shown that adding noises (with a small amplitude) to the training data set is equivalent to a Tikhonov regularization [3]. Also it is known that the Tikhonov regularizers are especially suitable to modeling work with insufficient sampling data. Therefore, when a regularization term is hard to find in the modeling process, adding noises is a simple alternative to regularization. In this way, the prediction capability of the inferred network on the training data set is not perfect any more, whereas the robustness will be optimal and the solution will be stable [21]. In order to make the simulation environment of gene regulation more realistic and to enhance the robustness of the inferred network, in the experimental Section 4.2, a certain amount of noises will be injected to the original data to train a robust network.

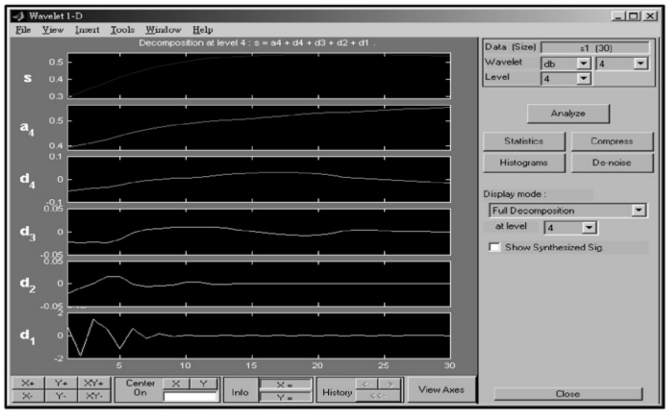
3.4. Gene clustering and network decomposition

As can be expected, when the number of genes in a regulatory network and the interactions between the genes increase in respect to the increasing functional complexity the network has to deal with, the direct modeling for a network becomes more and more difficult to achieve. To scale up our approach to solve more complicated reconstruction task, we take an engineering point of view to tackle the problem in a "divide and conquer" way. A clustering technique is firstly employed to group the genes into some small-scale networks, based on the analysis of their corresponding expression data, and then the small networks are reconstructed from the expression data by the RNN-based approach described in the above sections. Once all the small networks have been obtained, each of them is regarded as a self-contained system component of the original network, and the learning algorithm indicated in Section 3.2 are used to determine the interactions between different system components. The small networks can be decomposed again in the similar way until the resulting networks can be directly modeled.

In our current work, the self-organization feature map (SOM) method is adopted for gene clustering. Before a clustering method is applied to the expression data, some features on the data set have to be decided so that the clustering method can find the relationships between the data accordingly. Here we use the wavelet transform (WT) technique to extract data features from the waveforms derived from the gene expression data of different time points.

The WT theory has been widely used in many signal-processing applications [14]. WT decomposes a signal into a set of basis functions called wavelets. It involves representing a time function in terms of simple and fixed building blocks, termed wavelets. These building blocks are actually a family of functions derived from a single generating function (i.e., the mother wavelet) by translation and dilation operations. It is known that the WT is more suitable in analyzing non-stationary signals, since it is well localized in time and frequency [6]. With its important ability on data manipulation, WT can compress an original signal that consists of many data points, into a few parameters called wavelet coefficients that characterize the behavior of the signal. The wavelet coefficients can be computed by using the discrete WT. The computed wavelet coefficients provide a compact representation that shows the energy distribution of the signal in time and frequency. Therefore, the wavelet coefficients derived from the time-varying gene regulatory signals can be used as features of the signals for gene clustering.

Fig. 2 is the typical result of WT for a certain gene (produced by MATLAB Wavelet toolbox), in which S is the original gene expression data, a_4 is the wavelet approximation (taken from the Daubechies function with wavelets of order 4) by the relevant subsequences, and d_1 to d_4 are the wavelet detailed subsequences (with four levels multi-resolution analysis). The coefficients of the high frequent wavelet subsequences are then used as data features for SOM clustering.


Decomposition at level 4 : s = a4 + d4 + d3 + d2 + d1 .

Data (Size) s1 (30)
Wavelet db 4
Level 4
Display mode : Full Decomposition
at level 4
Show Synthesized Sig

Fig. 2. The wavelet transform for the expression data of a gene.

4. Experiments and results

Following our RNN model for GRN and the corresponding learning mechanism for network reconstruction, we conduct three series of experiments to evaluate our methodology. The first series is to examine whether the presented model can be learnt from a given set of time series data. In this phase, three different data sets for small-scale networks are used. The second series is to investigate whether our approach can infer robust networks from noisy and unreliable data. Finally, to deal with the scalability problem, in the third series experiments our approach is coupled with a clustering technique to reconstruct complicated networks with more gene nodes.

4.1. Modeling small networks

To evaluate our approach, we first used the well-known GRN simulation software Genexp (reported in Ref. [22]) to produce expression data. A four genes network was defined in which the accumulation and degradation rate constants of gene product k_1 and k_2 for all genes were all set to 0.3 (chosen from preliminary test). This simulation was run for 30 time steps for data collection. After the data set was obtained, our approach was used to learn a model reversely from data, and to achieve the desired network behavior. After learning, a small error accumulated from the 30 time steps for the four genes was obtained (for example, in one typical run the error is

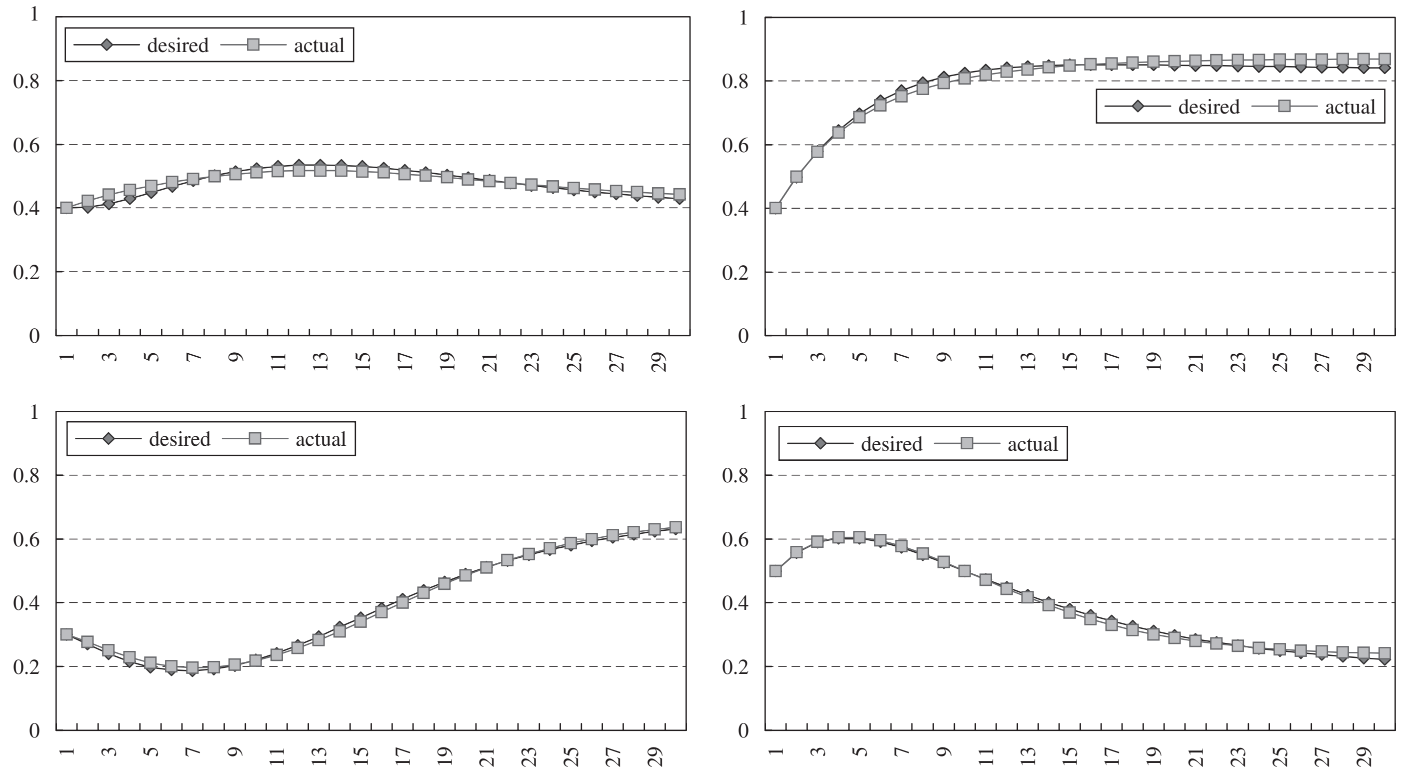


Fig. 3. Behaviors of the target and synthesized systems for the first data set.

0.004316). Fig. 3 compares the system behaviors of the original and reconstructed networks, in which the x-axis represents time step (for collecting data) and y-axis, the concentrations of different gene components. As can be seen, after training the RNN can successfully learn the system behavior of the original four nodes regulatory network.

The second data set used to test our RNN model is the one reported in Ref. [1], which is the expression data of a metabolic network consisting of three substances (X_1, X_2, and X_3 in the equations below). As described in Ref. [1], the target network is a part of the biological phospholipids pathway, and their experimental data was derived from the E-cell simulation environment (i.e., a software package for cellular and biochemical modeling and simulation, see Ref. [20]). This network can be described approximately as:

$$\dot{X}_1 = -10.3176X_1X_2$$
$$\dot{X}_2 = 9.7149X_1X_3 - 17.5084X_2$$
$$\dot{X}_3 = -9.7018X_1X_3 + 17.4766X_2$$

The algorithm presented in Section 3.2 was used to learn a network model from the expression data. In the experiment, the constants k_1 and k_2 for the above three genes X_1, X_2, and X_3 were set to (0.2, 0.2), (0.4, 0.4), and (0.2, 0.2), respectively (chosen from preliminary test). Fig. 4 shows the results. It indicates that after training, the desired system behavior can be learnt successfully.

The last experiment for evaluating our approach toward the reconstruction of small-scale GRNs is to model a S-system that has a power law form and has been proposed as a precise model for regulatory networks. Here, we chose the same regulatory network reported in Ref. [1] as the target network. It consists of five nodes and their relationship can be described as the following:

$$\dot{X}_1 = 15.0X_3X_5^{-0.1} - 10.0X_1^{2.0}$$
$$\dot{X}_2 = 10.0X_1^{2.0} - 10.0X_2^{2.0}$$
$$\dot{X}_3 = 10.0X_2^{-0.1} - 10.0X_2^{-0.1}X_3^{2.0}$$
$$\dot{X}_4 = 8.0X_1^{2.0}X_5^{-0.1} - 10.0X_4^{2.0}$$
$$\dot{X}_5 = 10.0X_4^{2.0} - 10.0X_5^{2.0}$$

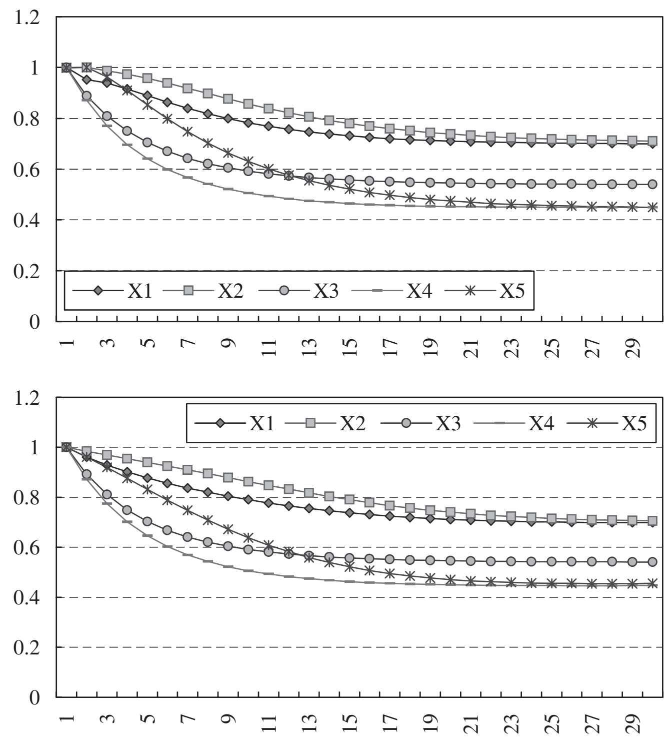


Fig. 5. Behaviors of the target (up) and inferred (down) systems for the third data set.

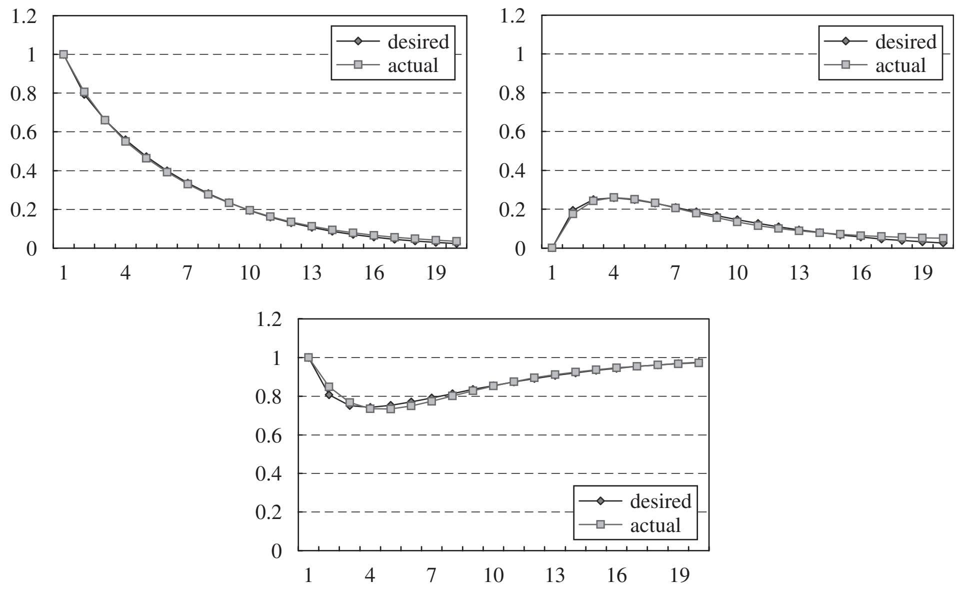


Fig. 4. Behaviors of the target and synthesized systems for the second data set.

Again, the proposed approach was employed to infer the above network. The upper part of Fig. 5 shows the original (desired) time series data for the five nodes, and the lower, the expressions of the synthesized dynamic neural network system. As can be observed, the behaviors of the two systems are nearly identical and the accumulated error for the five nodes is very small. It shows that a S-system can be modeled by our RNN-based network, and the network can be reconstructed from the expression data by the learning mechanism presented.

4.2. Robustness

After showing that the proposed RNN model can be used to efficiently construct GRNs from expression data, this section investigates how a robust RNN model can be inferred. In the experiments, the first data set described in Section 4.1 was regarded as the desired expression data produced by the original GRN. As indicated in Section 3.3, on one hand, the collected data could be disturbed in some unexpected experimental situations; on the other hand, the perfect model may cause the over-fitting problem. Therefore, adding certain level of noises to the training data has been advocated to increase the robustness of the inferred networks. Yet, instead of expanding the original measured data set with a set of noisy duplicates (to overcome the problem of insufficient training samples, as reported in Ref. [21]), here we introduced some random noises to the desired data to simulate the effect of data disturbance. In this work, the data obtained at each time point were randomly added/deleted a small amount of noises up to 5% of the original data (i.e., +5% to −5% uniformly). The left part of Fig. 6 is the expression data of the data set with random noises.

The right part of Fig. 6 shows the behavior of the RNN learnt from the disturbed data. Due to inherited fault-tolerant characteristic of the neural network, the noise effect on the data can be overcome by this model. Therefore, the behavior generated here is almost the same as the one described in Section 4.1.

4.3. Modeling large systems

To model large systems with more genes, the data available are usually not sufficient to determine accurately the interactions between all genes in a given data set. It is thus important to be able to construct a coarse-grained description of the system at first. This section demonstrates how the clustering method can help inferring networks from data. The first data set was an artificial data set obtained from the software Genexp. To collect data, initial parameters were specified for a 10 genes network and then the expression data were recorded for 30 consecutive time points. The upper part of Fig. 7 shows the data. To reconstruct the original network from these time series data, the gene clustering method described in Section 3.4, including procedures of WT and SOM, was applied to group genes. Two clusters were observed, one for genes 1, 2, and 3, the other for genes 5, 6, 7, 8 and 9. Genes 4 and 10 did not obviously belong to any of the above two clusters. After furthermore measuring the gene distance and calculating the Pearson's correlation coefficients between the genes, we decided to organize the genes into four parts as shown in Fig. 8, in which genes 4 and 10 were considered as outliers. In addition, based on the above measurement and calculation, some relationships between the four grouped elements were eliminated and the initial settings (with all weights — 1) were specified.

After the genes were grouped, the clusters $C1$ and $C2$ were both represented as fully RNNs and trained by our approach. With the trained results, the overall network with schematic shown in Fig. 8 was trained again at a higher level. Fig. 7 presents the system behaviors of the original and reconstructed networks over the 10 genes. Though this is not a perfect match in data fitting, it can be observed that the behavior of the trained network is similar to the original one, in which many of them have almost identical data sequences. It shows the efficiency of the proposed approach.

The second data set is a real experimental data set Rat central nervous system (CNS), taken from Ref. [25]. This data set includes expression data of 112 genes collected

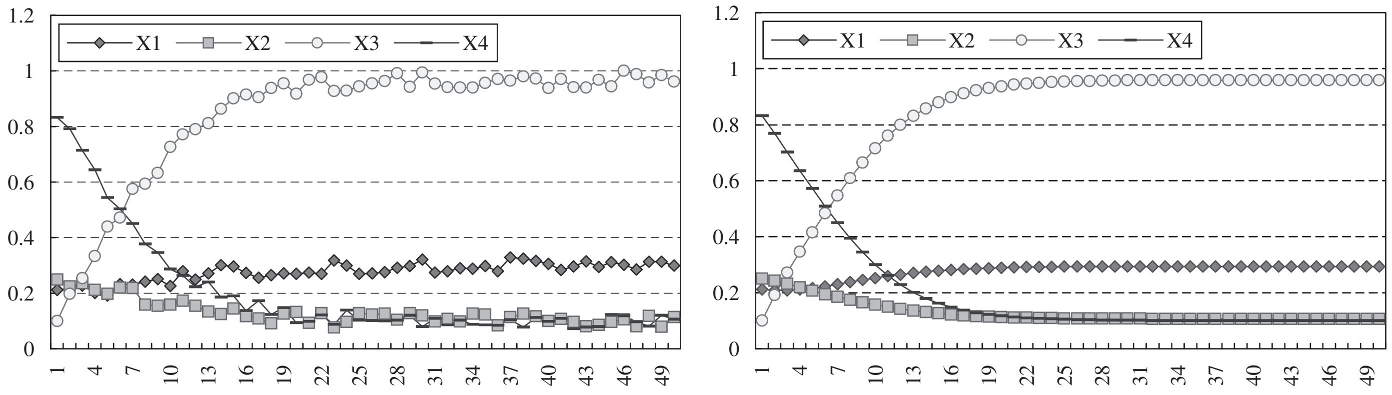
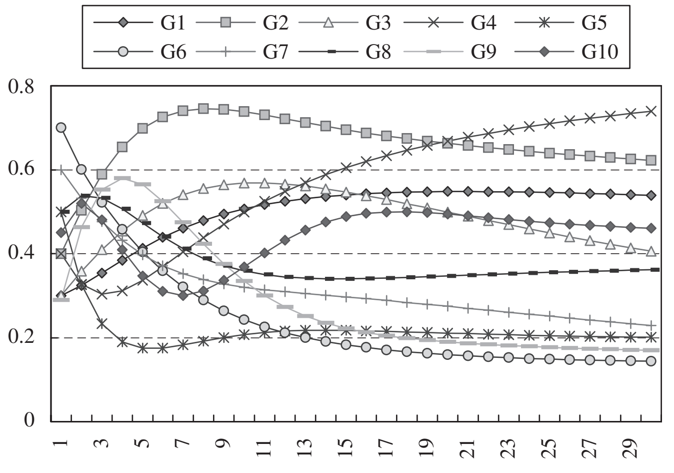


Fig. 6. The expression data of data set with random noises (left) and the expression data produced from the reconstructed network (right).



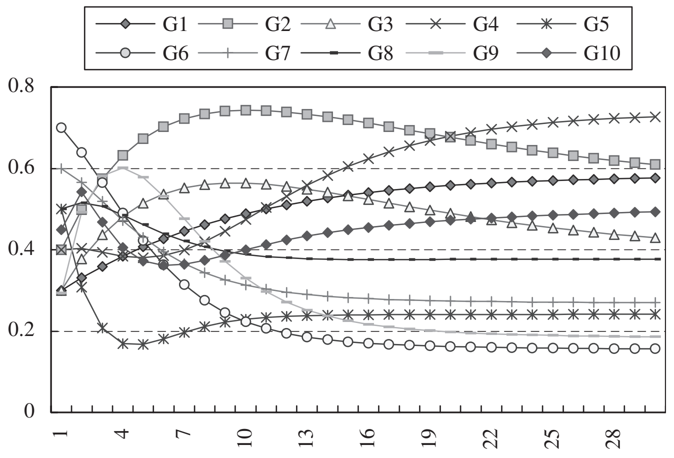


Fig. 7. The expression data of the original (up) and inferred (down) networks.

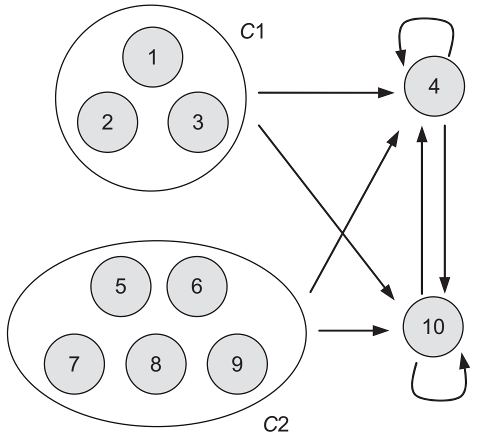


Fig. 8. The result after clustering.

from nine time points of different phases (embryonic, postnatal, and adult). As with the first data set, we performed WT to extract data features and then conducted cluster analysis to group genes. Among the 112 genes data, 103 of them were categorized into six different clusters and 9 genes did not belong to any cluster. One of our clusters consisting of 19 genes is very similar to the one reported in

Table 1
The details of each sub-group

Sub-group	Gene names	# Genes
sg1	NFH, NFM, MOG, GRg1, NGF, Afgf, GFAP, cfos	8
sg2	S100beta, mGluR1, CNTF, GFAP, cfos	5
sg3	ChAT, NMDA2A, Bfgf, MOG	4
sg4	mAChR4, cjun, IP3R2, GFAP, cfos	5

a previous study dealing with rat CNS data [25] (containing the 17 genes cluster in fact). To be consistent with the previous study and preserving the meaning of the cluster as the original work, we decided to use the 17 genes cluster reported in Ref. [25] as the target network to be reconstructed.

As the genes within the same cluster have been closely related, it is not practical to group them by the same clustering method again. Therefore, once the above target network (i.e., the one with 17 genes) has been determined, the genes were decomposed into four sub-groups according to the mutual information between them (that is often used to distinguish the close relationships between genes), in which some genes belonged to more than one sub-group. Table 1 lists the details of the sub-groups. Then the training method applied on the first data set was employed to build the four subnets and afterward, the target network. Fig. 9 shows four sets of behaviors of the desired (left) and modeled (right) networks group by group. Again, very similar behaviors can be obtained and it indicates that our approach can be efficiently and successfully used to model networks with relatively large size.

5. Conclusions and future work

The construction of genetic regulatory networks is one of the most important issues in conducting any system biology research. Many models have been proposed to simulate GRNs, and computational methods have also been developed to reconstruct networks. Instead of subjectively arguing which model and method are most suitable for network reconstruction, our work emphasizes the importance of establishing a practical approach that can model GRNs and is scalable for inferring complex networks. As recurrent networks can work as dynamical systems as GRNs do, therefore, our approach has adopted the RNN model to represent GRNs. To simulate how a GRN operates and to learn a network from expression data, we have developed a system with an enhanced version of RNN learning algorithm. In addition, in order to deal with the scalability problem, a clustering method with several data analysis techniques for feature extraction has been implemented to infer large networks hierarchically. The importance of network robustness has also been pointed out, and the criterion of adding noises to the original data has been applied to train a robust network. To verify the presented approach, three series of

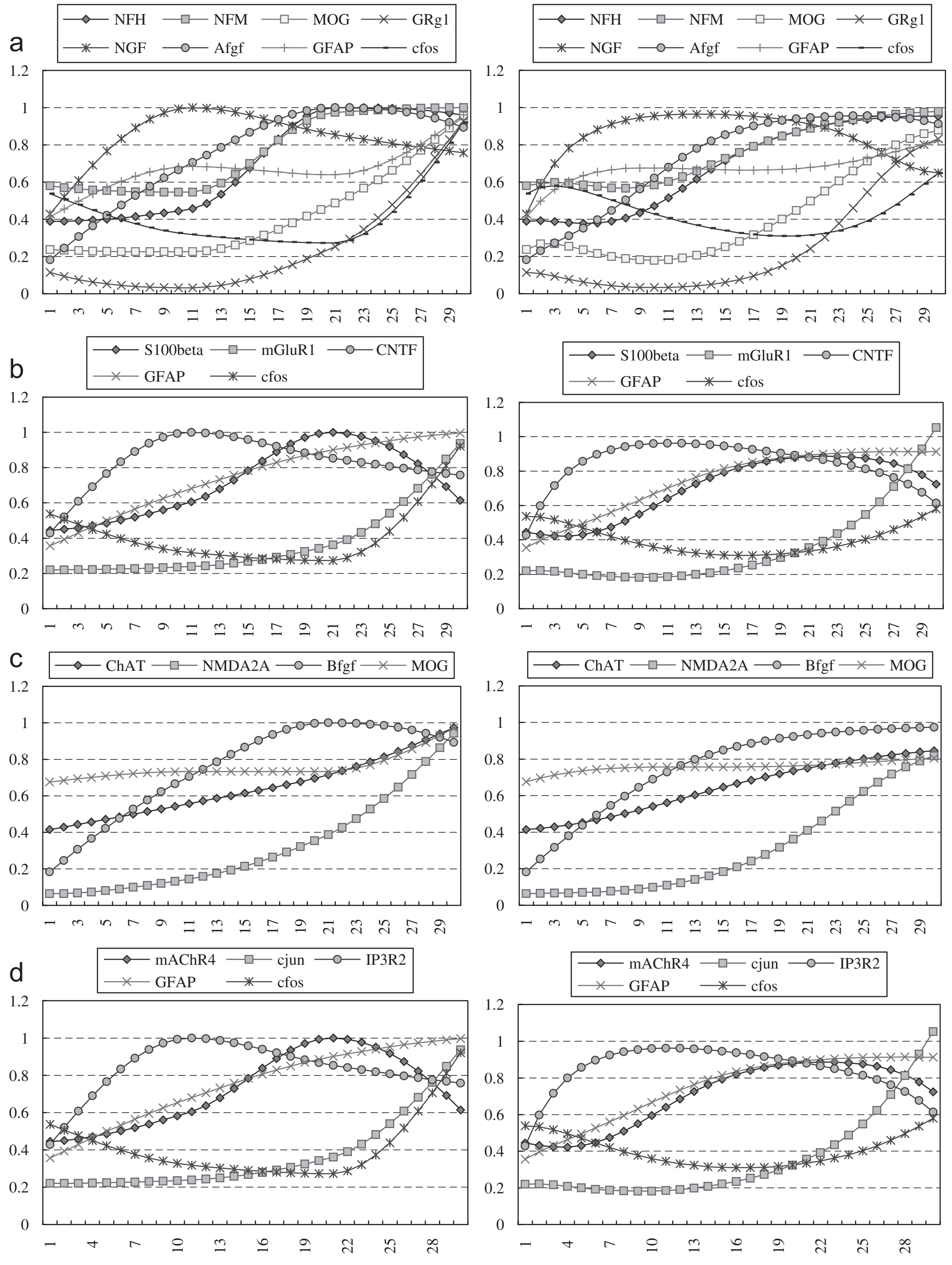


Fig. 9. The behaviors of the four sets of genes. (a) Sub-group 1; (b) sub-group 2; (c) sub-group 3; (d) sub-group 4.

experiments have been conducted to demonstrate how it works for inferring small and large networks. Experimental results have shown that our approach can be successfully used to infer networks from measured expression data.

Our work presented here directs to some prospects of future research. The first is to incorporate biological knowledge into our approach to construct GRNs in an even more efficient way. Because biological knowledge about the general properties of genetic networks can alleviate some of the data requirements, if we take it into account in network reconstruction, it can thus reduce computational effort and obtain more accurate model. Also it is worthwhile to investigate how to employ other types of learning algorithms to improve the modeling performance. We are currently implementing a framework of evolving neural networks and evaluating its corresponding performance. Another direction is to develop a gene clustering method that can consider more characteristics of gene regulation at the same time in feature extraction, and is suitable for gene regulatory network modeling in particular. It should be helpful in reconstructing networks in a hierarchical way.

Acknowledgment

This work was supported by National Science Council under contract NSC-95-2221-E-110–143.

References

[1] S. Ando, E. Sakamoto, H. Iba, Evolutionary modeling and inference of gene network, Inf. Sci. 145 (3–4) (2002) 237–259.

[2] R.D. Beer, Dynamical approaches to cognitive science, Trends Cogn. Sci. 4 (3) (2000) 91–99.

[3] C.M. Bishop, Training with noise is equivalent to tikhonov regularization, Neural Comput. 7 (1) (1994) 108–116.

[4] M.F. Blasi, I. Casorelli, A. Colosimo, F.S. Blasi, M. Bignami, A. Giuliani, A recursive network approach can identify constitutive regulatory circuits in gene expression data, Physica A 348 (2005) 349–370.

[5] M.E. Csete, J.C. Doyle, Reverse engineering of biological complexity, Science 295 (5560) (2002) 1664–1669.

[6] I. Daubechies, The wavelet transform time-frequency localization and signal analysis, IEEE Trans. Inf. Theory 36 (5) (1990) 961–1005.

[7] H. DeJong, Modeling and simulation of genetic regulatory systems: a literature review, J. Comput. Biol. 9 (1) (2002) 67–103.

[8] P. D'haeseleer, S. Liang, R. Somogyi, Genetic network inference: from co-expression clustering to reverse engineering, Bioinformatics 16 (8) (2000) 707–726.

[9] A.J. Hartemink, D.K. Gifford, T.S. Jaakkola, R.A. Young, Bayesian methods for elucidating genetic regulatory networks, IEEE Intell. Syst. 17 (2) (2002) 37–43.

[10] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, NJ, 1998.

[11] D. Heckerman, A Bayesian approach to learning causal networks, in: Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence, 1995, pp. 285–295.

[12] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Netw. 1 (4) (1988) 295–307.

[13] J. Koza, W. Mydlowec, G. Lanza, J. Yu, M.A. Keane, Automated reverse engineering of metabolic pathways from observed data by means of genetic programming, in: H. Kitano (Ed.), Foundations of System Biology, MIT Press, 2001, pp. 95–121.

[14] S. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Trans. Pattern Anal. Mach. Intell. 11 (7) (1989) 674–693.

[15] S. Mehra, W. Hu, G. Karypis, A Boolean algorithm for reconstructing the structure of regulatory networks, Metab. Eng. 6 (2004) 326–339.

[16] I.M. Ong, J.D. Glasner, D. Page, Modeling regulatory pathways in *E. coli* from time series expression profiles, Bioinformatics 18 (2002) s241–s248.

[17] B.A. Pearlmutter, Learning state space trajectories in recurrent neural networks, Neural Comput. 1 (1989) 263–269.

[18] F. Ren, J. Cao, Asymptotic and robust stability of genetic regulatory networks with time-varying delays, Neurocomputing, doi:10.1016/j.neucom.2007.03.011.

[19] M. Schena, D. Shalon, R.W. Davis, P.O. Brown, Quantitative monitoring of gene expression patterns with a complementary DNA microarray, Science 270 (1995) 467–470.

[20] M. Tomita, K. Hashimoto, K. Takahashi, T.S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J.C. Venter, R. Hutchison, E-Cell: software environment for whole-cell simulation, Bioinformatics 15 (1) (1999) 72–84.

[21] E.P. van Someren, L. Wessels, M. Reinders, E. Backer, Robust genetic network modeling by adding noisy data, in: Proceedings of IEEE Workshop on Non-linear Signal and Image Processing, 2001.

[22] J. Vohradsky, Neural network model of gene expression, FASEB J. 15 (2001) 846–854.

[23] E. Voit, Computational Analysis of Biochemical Systems, Cambridge University Press, 2000.

[24] T. Vu, J. Vohradsky, Genexp: a genetic network simulation environment, Bioinformatics 18 (2002) 1400–1401.

[25] X. Wen, S. Fuhrman, G. Michaels, D. Carr, S. Smith, J. Barker, R. Somogyi, Large-scale temporal gene expression mapping of central nervous system development, PNAS 95 (1998) 334–339.

[26] P.J. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (10) (1990) 1550–1560.

Wei-Po Lee received his Ph.D. in artificial intelligence from University of Edinburgh, United Kingdom. He is currently an associate professor at the department of Information Management, National Sun Yat-sen University, Taiwan. He is interested in intelligent autonomous systems, biologically inspired robotics, machine learning, and artificial life.

Kung-Cheng Yang received his M.Sc. degree from the Department of Management Information Systems, National Pingtung University, Taiwan. He is interested in system modeling and simulation, computational intelligence, and bioinformatics.