Original Paper

Function Approximation Approach to the Inference of Neural Network Models of Genetic Networks

Shuhei Kimura,† Katsuki Sonoda,†† Soichiro Yamane,††
Koki Matsumura† and Mariko Hatakeyama†††

A model based on a set of differential equations can effectively capture various dynamics. This type of model is therefore ideal for describing genetic networks. Several genetic network inference algorithms based on models of this type have been proposed. Most of these inference methods use models based on a set of differential equations of the fixed form to describe genetic networks. In this study, we propose a new method for the inference of genetic networks. To describe genetic networks, the proposed method does not use models of the fixed form, but uses neural network models. In order to interpret obtained neural network models, we also propose a method based on sensitivity analysis. The effectiveness of the proposed methods is verified through a series of artificial genetic network inference problems.

1. Introduction

The mathematical modeling of biochemical networks, such as genetic networks, metabolic networks, and signal transduction cascades, is a central theme in systems biology and a point of intensifying focus in the post-genomic era [24]. Many recent studies have sought to develop computational methods for inferring genetic networks from gene expression data obtained by DNA microarrays [4]. The inferred model of the genetic network is conceived as an ideal tool to help biologists generate hypotheses and facilitate the design of their experiments. It may also shed light on the biological functions of genes.

Numerous models to describe biochemical networks have been proposed [1),5),7),13),17),22),26),28)]. Among them, we focus on the models based on a set of differential equations in this study because these models have an ability to capture the dynamics. In a genetic network inference problem based on a set of differential equations, the genetic network can be described as

$$\frac{dX_i}{dt} = G_i(X_1, X_2, \cdots, X_N),\ (i = 1, \cdots, N), \tag{1}$$

where X_i is the expression level of the i-th gene, N is the number of genes in the network, and G_i is a function of an arbitrary form. The purpose of the genetic network inference problem based on a set of differential equations is to identify the function G_i from the observed gene expression data.

When we try to infer genetic networks, the function G_i is generally approximated using a set of differential equations of the fixed form. One of well-studied models based on a set of differential equations of the fixed form is a linear model [6),28)]. The computational time for inferring linear models of genetic networks is short. However, the linear model is not suitable for analyzing time-series of gene expression data because the model requires that the system is operating near a steady state [28)]. Another well-studied model based on a set of differential equations of the fixed form is an S-system model [23)]. This model possesses a rich structure capable of capturing various dynamics, and can be analyzed by several available methods. Because of these advantages, a number of methods for the inference of S-system models of genetic networks have been proposed [2),8),10),11),20),21)]. However, these methods are time-consuming because they require to solve a set of differential equations many times.

In this paper, we propose a new method for the inference of genetic networks. The proposed method uses a neural network model to approximate the function G_i. As the neural network model is a powerful function approximator, it has an ability to provide a good approximation of the function G_i. Moreover, as it is not necessary for the learning of the neural network models to solve a set of differential equations, the proposed method requires the shorter computational time. For the interpretation of the neural network models obtained, this study also proposes a method based on sensitivity analysis.

† Faculty of Engineering, Tottori University
†† JFE R&D Corporation
††† RIKEN Genomic Sciences Center

Finally, we verify the effectiveness of the proposed inference methods by applying them to several artificial genetic network inference problems.

2. Genetic Network Inference Problem

The purpose of the genetic network inference problem is to find a good approximation of the function G_i $(i = 1, \cdots, N)$, as mentioned in the previous section. This problem can be defined as a function approximation problem [12]. The function approximation problem is a problem where an unspecified function f is approximated only on the basis of observations at T points

$$(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2)), \cdots, (\mathbf{x}_T, f(\mathbf{x}_T)).$$

When we try to infer genetic networks in this study, we approximate the function G_i on the basis of observations at T points

$$\left(\mathbf{X}|_{t_1}, \left.\frac{dX_i}{dt}\right|_{t_1}\right), \cdots, \left(\mathbf{X}|_{t_T}, \left.\frac{dX_i}{dt}\right|_{t_T}\right),$$

where $\mathbf{X}|_{t_k} = \left(X_1|_{t_k}, X_2|_{t_k}, \cdots, X_N|_{t_k}\right)$ is the gene expression levels of all of the genes at time t_k, and $\left.\frac{dX_i}{dt}\right|_{t_k}$ is the differential coefficient of the expression level (rate of transcription) of the i-th gene at time t_k.

As described just above, our approach requires both the gene expression levels and their differential coefficients to solve the genetic network inference problem. Though DNA microarray technologies allow us to measure the levels of gene expression, we have yet to find a biological technique capable of measuring the differential coefficients of gene expression levels. As an alternative, the data we obtain by measuring the time-series of the gene expression levels allow us to estimate the differential coefficients using interpolation techniques, such as the spline interpolation [15] or the local linear regression [3]. Taking basically the same approach, Voit and Almeida [25] proposed a method that uses the estimated differential coefficients of gene expression levels to infer the S-system model of a genetic network. Their method used a neural network to estimate the differential coefficients.

3. Learning of Neural Network Model

Several techniques should be available for solving the function approximation problem defined above. In this study, however, as a neural network is known to be a powerful function approximator, we use it to approximate the function G_i [12]. We should note here that one neural network in this study corresponds to one gene. When we try to solve a genetic network inference problem consisting of N genes, we therefore must obtain N neural networks, each corresponding to one of the genes.

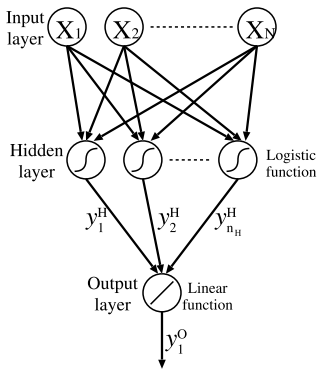

Fig. 1 Three-layer feed-forward neural network model.

In the remainder of this section, we will explain the method to obtain the neural network corresponding to the i-th gene.

3.1 Model Description

We obtain our function approximation using a typical multi-layer feed-forward neural network. The neural network used here consists of three layers (the input, hidden and output layers), each containing several neurons (the input, hidden and output neurons) (**Fig. 1**).

The input layer contains N neurons, where N is the number of genes in the genetic network. Each input neuron works as an input for the expression level of each gene. The input neurons connect only with the hidden neurons.

The hidden layer consists of n_H neurons. The k-th hidden neuron $(k = 1, \cdots, n_H)$ receives the values from the input neurons, and its output y_k^H is calculated as

$$y_k^H = f\left(\sum_{j=1}^{N} w_{j,k}^{IH} X_j - \theta_k^H\right), \qquad (2)$$

where X_j is the expression level of the j-th gene, $w_{j,k}^{IH}$ is the weight parameter between the j-th input neuron and the k-th hidden neuron, θ_k^H is the threshold parameter of the k-th hidden neuron, and $f(x) = \frac{1}{1+\exp(-x)}$.

The output layer has one neuron that receives the outputs of the hidden neurons. This neuron

calculates its output y_1^O through

$$y_1^O = g\left(\sum_{k=1}^{n_H} w_{k,1}^{HO} y_k^H - \theta_1^O\right), \quad (3)$$

where $w_{k,1}^{HO}$ is the weight parameter between the k-th hidden neuron and the output neuron, θ_1^O is the threshold parameter of the output neuron, and $g(x) = x$. y_1^O is the final output of this neural network and the differential coefficient of the expression level of the i-th gene.

3.2 Problem Definition

When trying to obtain a neural network that outputs Y_t $(t = 1, \cdots, T)$ against a corresponding input $\mathbf{x}_t$, we formulate the learning of the neural network as a function optimization problem. Our object in this problem is to find the model parameters ($w_{j,k}^{IH}$, θ_k^H, $w_{k,1}^{HO}$ and θ_1^O) which minimize the function

$$E = \frac{1}{2}\sum_{t=1}^{T}\left[y_1^O(\mathbf{x}_t) - Y_t\right]^2, \quad (4)$$

where $y_1^O(\mathbf{x}_t)$ is the output of the neural network against the input $\mathbf{x}_t$. The variable $\mathbf{x}_t$ in the genetic network inference problem represents the observed expression levels of all of the genes at time t (i.e., $\mathbf{X}|_t$). The variable Y_t represents the differential coefficient of the expression level of the i-th gene at time t (i.e., $\frac{dX_i}{dt}\big|_t$), estimated from the observed time-series data. We seem to find, however, that this function optimization problem has multiple optima. This drawback stems from the high degree-of-freedom of the neural network model and the generally insufficient amounts of time-series data observed. To increase the probability of finding a reasonable solution, we introduce a priori knowledge about the genetic network into the objective function (4).

Genetic networks are known to be sparsely connected [19]. When the i-th gene is unaffected by the j-th gene, the weight parameters associated with the j-th input neuron, i.e., $w_{j,k}^{IH}$ $(k = 1, \cdots, n_H)$, should be zero. When this condition holds, the sum of the squared weight parameters $\omega_j = \sum_{k=1}^{n_H}\left(w_{j,k}^{IH}\right)^2$ will also be zero. We incorporate this knowledge into the objective function (4) using a penalty term, as shown below.

$$F = \frac{1}{2}\sum_{t=1}^{T}\left[y_1^O(\mathbf{x}_t) - Y_t\right]^2 + c\sum_{j=1}^{N-I} W_j, \quad (5)$$

where W_j is given by rearranging ω_j in descending order of their values (i.e., $W_1 \le W_2 \le \cdots \le W_N$). The variable c is a penalty coefficient and I is a maximum indegree. The maximum indegree determines the maximum number of genes that directly affect the i-th gene.

The penalty term is the second term on the right-hand side of Eq. (5). When the penalty term is applied, most of the genes are disconnected from each other. When the number of genes that directly affect the i-th gene is smaller than the maximum indegree I, this penalty term, however, will not actually penalize. Here, we use Eq. (5) as an objective function to be minimized. Several earlier studies have used penalty terms to reduce the degree-of-freedom in the genetic network inference problem [8),10),11)].

3.3 Learning Algorithm

Though any type of function optimizer can be applied to the neural network learning problem, the back-propagation is by far the most popular algorithm. The back-propagation algorithm optimizes the objective function (4) using the computed gradient values [16)]. As the new objective function (5) is also differentiable, we can design an algorithm similar to the back-propagation even when trying to optimize the function (5).

The back-propagation algorithm has been successfully applied in various areas, but its use of gradient descent has drawbacks [27)]. Most importantly, the algorithm often gets trapped into a local optimum of the objective function and fails to find a global optimum when the objective function is multimodal. To overcome the shortcomings of the gradient-descent-based learning algorithms, evolutionary algorithms have been often used [27)].

Evolutionary algorithms generally use no gradient information of the objective function. However, when it is available, we should use it for the effective search. Therefore, our group uses an evolutionary algorithm, GLSDC [9)], for the learning of the neural network model. GLSDC applied here uses the back-propagation algorithm as its search operator in order to utilize the gradient information. The detailed information on GLSDC is given in Refs. 9), 12).

4. Model Refinement

As the obtained neural network models approximate the function G_i's given in Eq. (1), we can simulate the gene expression of the target system by solving the following set of differential equations.

$$\frac{dX_i}{dt} = \hat{G}_i(X_1, X_2, \cdots, X_N),\ (i=1, \cdots, N), \tag{6}$$

where $\hat{G}_i$ is the obtained neural network model corresponding to the i-th gene, and N is the number of genes in the network. The computed time-courses of the gene expression levels, however, do not always resemble the observed data because of the training errors of the neural network models. Therefore, we refine the obtained neural network models by solving function optimization problems described below.

Since the high-dimensionality makes it difficult to refine all of the obtained neural network models simultaneously, we utilize the problem decomposition strategy [10),14)]. This strategy divides the refinement problem into N function optimization problems, each of which corresponding to each gene. In the i-th refinement problem corresponding to the i-th gene, we search for the model parameters of the i-th neural network model, which minimize

$$f_i = \sum_{k=1}^{T} \left(X_i|_{t_k}^{\exp} - X_i|_{t_k}^{\mathrm{cal}} \right)^2 + d \sum_{j=1}^{N-I} W_j, \tag{7}$$

where the second term on the right-hand side is the same penalty term as that we introduced into Eq. (5), and d is a penalty coefficient. $X_i|_{t_k}^{\exp}$ is an experimentally observed gene expression level at time t_k of the i-th gene, and $X_i|_{t_k}^{\mathrm{cal}}$ is a numerically calculated one. In this study, $X_i|_{t_k}^{\mathrm{cal}}$ is obtained not by solving the set of differential Eq. (6), but by solving the following differential equation.

$$\frac{dX_i}{dt} = \hat{G}_i(Y_1, Y_2, \cdots, Y_N), \tag{8}$$

where

$$Y_j = \begin{cases} X_j, & \text{if } j = i, \\ \hat{X}_j, & \text{otherwise}, \end{cases} \tag{9}$$

and $\hat{X}_j$ is an estimated time-course of the j-th gene's expression level acquired by making a direct estimation from the observed time-series data. Two methods are used to estimate $\hat{X}_j$'s in this study. When the data are assumed to be observed with no measurement error, $\hat{X}_j$'s are estimated using the spline interpolation [15)]. When the absence of measurement error cannot be confirmed, the local linear regression [3)] is used to estimate the values.

We used the modified Powell's method [15)] to solve the function optimization problems defined here. These problems require to solve the differential Eq. (8) many times, in contrast to the neural network learning problem described in the previous section. These computational costs are, however, not high because the modified Powell's method is a sophisticated local search method.

5. Model Interpretation

When working on the genetic network inference, we must know whether the i-th gene is affected by the j-th gene. In this study, we propose a method based on sensitivity analysis to extract this information from the neural network model obtained [12)].

The change in X_i is unaffected by X_j when the i-th gene is unaffected by the j-th gene. Thus, on this condition, the following equations always hold.

$$\frac{\partial}{\partial X_j}\left(\frac{dX_i}{dt}\right) = \frac{\partial G_i(X_1, \cdots, X_N)}{\partial X_j} = 0, \tag{10}$$

where G_i is the function that gives the mapping from the expression levels of all of the genes to the differential coefficient of the expression level of the i-th gene, as described in the section 1. $\frac{\partial G_i}{\partial X_j}$ is generally called the sensitivity coefficient. The absolute value of the sensitivity coefficient represents the impact of the j-th gene upon the i-th gene. The large absolute value of the sensitivity coefficient indicates that the j-th gene strongly affects the i-th gene. We can also know, from the sign of the sensitivity coefficient, whether the regulation of the i-the gene from the j-th gene is positive or negative; when the sign is positive (negative), the j-th gene positively (negatively) regulates the i-th gene.

As the sensitivity coefficient generally varies according to time, we use the positive and negative sensitivity coefficients averaged over time, $S_i^p(j)$ and $S_i^m(j)$ respectively, to infer interactions between genes. To cope with the difficulty of calculating these values precisely, this paper approximates them as

$$\begin{aligned} S_i^p(j) &= \frac{1}{t_T - t_1} \int_{t_1}^{t_T} p\left(\frac{\partial G_i}{\partial X_j}\right) dt \\ &\simeq \frac{1}{T} \sum_{k=1}^{T} p\left(\left.\frac{\partial \hat{G}_i}{\partial X_j}\right|_{t_k}\right), \end{aligned} \tag{11}$$

and

$$S_i^m(j) = \frac{1}{t_T - t_1} \int_{t_1}^{t_T} m\left(\frac{\partial G_i}{\partial X_j}\right) dt \simeq \frac{1}{T} \sum_{k=1}^{T} m\left(\left.\frac{\partial \hat{G}_i}{\partial X_j}\right|_{t_k}\right), \quad (12)$$

where

$$p(x) = \begin{cases} x, & \text{if } x > 0, \\ 0, & \text{otherwise,} \end{cases} \quad (13)$$

$$m(x) = \begin{cases} x, & \text{if } x < 0, \\ 0, & \text{otherwise,} \end{cases} \quad (14)$$

T is the number of sampling points of the measured time-series data, and $\left.\frac{\partial \hat{G}_i}{\partial X_j}\right|_t$ is the estimated sensitivity coefficient at time t calculated from the neural network model. As the neural network model is differentiable, we can calculate $\left.\frac{\partial \hat{G}_i}{\partial X_j}\right|_t$ analytically.

The large absolute values of $S_i^p(j)$ and $S_i^m(j)$ suggest the existence of the positive and negative regulations of the i-th gene from the j-th gene, respectively. Therefore, we conclude that the i-th gene is positively regulated by the j-th gene when $|S_i^p(j)| + |S_i^m(j)|$ exceeds a threshold $Thresh(i)$ and

$$\frac{|S_i^p(j)|}{|S_i^p(j)| + |S_i^m(j)|} > \alpha, \quad (15)$$

where α is a constant parameter. Similarly, when $|S_i^p(j)| + |S_i^m(j)| > Thresh(i)$ and

$$\frac{|S_i^m(j)|}{|S_i^p(j)| + |S_i^m(j)|} > \alpha, \quad (16)$$

the proposed method infers the negative regulation of the i-th gene from the j-th gene. As the threshold, we use

$$Thresh(i) = \beta \max_j \left(|S_i^p(j)| + |S_i^m(j)|\right). \quad (17)$$

We set the threshold parameters α and β to 0.3 and 0.05, respectively. When α is set to less than 0.5, our method has an ability to infer both positive and negative regulations of the i-th gene from the j-th gene simultaneously.

6. Numerical Experiments

To confirm its effectiveness, we applied the proposed approach to two genetic network inference problems. Artificial genetic network problems were used in each case.

6.1 S-system Network

In this experiment, we confirmed that our approach has an ability to infer a genetic network model correctly.

6.1.1 Experimental Setup

As a target network that we attempt to infer, we used an S-system model consisting of 30 genes ($N = 30$). The S-system model [23] is often used to describe biochemical networks (see, e.g., Refs. 2), 11), 18), 20), 25)). The model is structured as a set of non-linear differential equations of the form

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{N} X_j^{g_{i,j}} - \beta_i \prod_{j=1}^{N} X_j^{h_{i,j}}, \quad (i = 1, 2, \cdots, N), \quad (18)$$

where X_i is the expression level of the i-th gene, α_i and β_i are multiplicative parameters called rate constants, and $g_{i,j}$ and $h_{i,j}$ are exponential parameters called kinetic orders. The network structure and the S-system parameters of the target network are shown in **Fig. 2** and **Table 1**, respectively [13].

15 sets of time-series data, each covering all 30 genes, were given as the observed gene expression patterns. The sets began from randomly generated initial values in $[0.0, 2.0]$ and were obtained by solving the differential

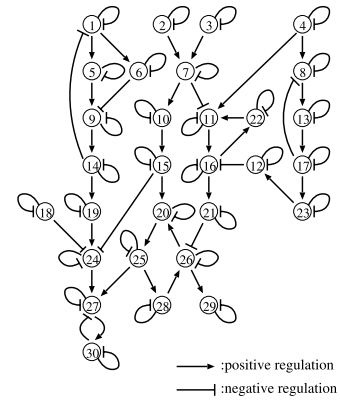

Fig. 2 The network structure of the target model.

Table 1 S-system parameters of the target model.

α_i	1.0
β_i	1.0
$g_{i,j}$	$g_{1,14} = -0.1$, $g_{5,1} = 1.0$, $g_{6,1} = 1.0$, $g_{7,2} = 0.5$, $g_{7,3} = 0.4$, $g_{8,4} = 0.2$, $g_{8,17} = -0.2$, $g_{9,5} = 1.0$, $g_{9,6} = -0.1$, $g_{10,7} = 0.3$, $g_{11,4} = 0.4$, $g_{11,7} = -0.2$, $g_{11,22} = 0.4$, $g_{12,23} = 0.1$, $g_{13,8} = 0.6$, $g_{14,9} = 1.0$, $g_{15,10} = 0.2$, $g_{16,11} = 0.5$, $g_{16,12} = -0.2$, $g_{17,13} = 0.5$, $g_{19,14} = 0.1$, $g_{20,15} = 0.7$, $g_{20,26} = 0.3$, $g_{21,16} = 0.6$, $g_{22,16} = 0.5$, $g_{23,17} = 0.2$, $g_{24,15} = -0.2$, $g_{24,18} = -0.1$, $g_{24,19} = 0.3$, $g_{25,20} = 0.4$, $g_{26,21} = -0.2$, $g_{26,28} = 0.1$, $g_{27,24} = 0.6$, $g_{27,25} = 0.3$, $g_{27,30} = -0.2$, $g_{28,25} = 0.5$, $g_{29,26} = 0.4$, $g_{30,27} = 0.6$, other $g_{i,j} = 0.0$
$h_{i,j}$	1.0 if $i = j$, 0.0 otherwise.

Eq. (18) on the target model. In a practical application, these sets of time-series data were obtained by actual biological experiments under different experimental conditions. 11 sampling points for the time-series data were assigned on each gene in each set. Thus, the observed time-series data for each gene consisted of $15 \times 11 = 165$ sampling points. We estimated neural network models solely from these time-series data and their estimated differential coefficients. As the data contained no measurement noise, the spline interpolation [15] was used to estimate the differential coefficients of the gene expression levels from the given time-series data.

As described before, our most important task in solving this genetic network inference problem was to obtain N neural networks, each corresponding to one gene. Each neural network consisted of N input neurons, n_H hidden neurons and one output neuron. Given the condition $n_H = 5$ applied in this study, the learning problem of each neural network was defined as a $N \times n_H + n_H \times 1 + n_H + 1 = 161$ dimensional function optimization problem. 10 runs were carried out by changing the seed for the pseudo random number to solve each function optimization problem. Each run was continued until the number of generations reached 300 or the best candidate solution in the population had not been improved in the last 30 generations. The method described in the section 4 was then applied to refine the obtained neural network models. The search regions of the parameters were $[-10.0, 10.0]$, the maximum indegree I was 5, and the penalty coefficients c and d were 0.1. The following recommended parameters were used in GLSDC applied; the population size n_p is $3n$, where n was the dimension of the search space, the number of children generated by the crossover per selection n_c was 10, and the number of applied the converging operations N_0 was $2n_p$.

6.1.2 Result

We extracted regulations from the obtained neural network models using the sensitivity coefficients $S_i^p(j)$ and $S_i^m(j)$, as described in Section 5. **Figure 3** shows a typical genetic network inferred from obtained models. As the figure illustrates, most of the regulations were correctly inferred by the proposed methods. The inferred networks, however, contained several unnecessary regulations that were absent in the target network, i.e., false-positive regulations.

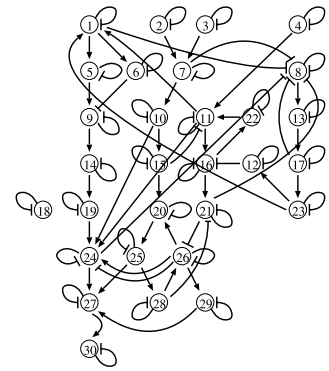

Fig. 3 A sample of the network structure inferred by the proposed approach.

The averaged numbers of the false-negative and false-positive regulations were 5.5 ± 2.1 and 12.9 ± 4.0, respectively, where the values subsequent to the '$\pm$' signs are the standard deviations. As the target network contains 68 regulations, the sensitivity and the specificity of the proposed approach were 0.919 ± 0.030 and 0.993 ± 0.002, respectively. The sensitivity and the specificity are defined as

$$\text{sensitivity} = \frac{TP}{TP + FN},$$

$$\text{specificity} = \frac{TN}{FP + TN},$$

where TP, FN, TN and FP are the numbers of true-positive, false-negative, true-negative and false-positive regulations, respectively. The sensitivity increases from 0 to 1 with decreasing the number of false-negative regulations, and the specificity increases from 0 to 1 with decreasing the number of false-positive regulations.

Even when the model refinement was not applied, the inference ability of the proposed approach was not degraded. The sensitivity and the specificity calculated from the neural network models without the model refinement were 0.919 ± 0.030 and 0.993 ± 0.002, respectively. Therefore, the application of the model refinement may be omitted when we are interested only in a network structure of the target system. The use of the model refinement, however, improved the sum of the squared error between the time-courses produced by the obtained model and the given time-series data, i.e., the value of the objective function (7). The averaged objective values (7) with and without the model refinement were $4.108 \times 10^{-3} \pm 9.112 \times 10^{-3}$ and $6.215 \times 10^{-3} \pm 1.421 \times 10^{-2}$, respectively. The

model refinement should be applied only when we try to perform the computational simulation.

When the method based on the S-system model [10] was applied to the same problem, the numbers of the true-positive, false-positive, true-negative and false-negative regulations were reportedly 68.0, 225.0, 1507.0 and 0.0, respectively. The number of the false-positive regulations inferred by the proposed approach was smaller than that of the S-system approach. However, most of the false-positive regulations inferred by the S-system approach were omissible because the absolute parameter values of these regulations were much smaller than those of the correct regulations [10]. Thus, the inference ability of the proposed approach was not always better than that of the S-system approach. The proposed approach, on the other hand, required shorter computational time. Our approach running on the single-CPU personal personal computer (Pentium IV 2.8 GHz) required about 47.1 and 46.2 minutes for the learning and the refinement of each neural network model, respectively. The proposed approach therefore required about $(47.1 + 46.2) \times 30$ minutes $\simeq 47$ hours to solve this genetic network inference problem while the S-system approach took about 73.8×30 hours on a single-CPU personal computer (Pentium III 1 GHz) [10]. When the linear model is used to infer the genetic network, much shorter computational time is expected. The linear model is not, however, suitable for analyzing the time-series data of this study because the model requires that the system is operating near a steady state [28].

We can change the function approximation capability of the neural network model using the number of hidden neurons n_H. When using a larger n_H, the neural network obtained should provide a model with a better fit to the observed data. However, an unduly large n_H produces a neural network model which overfits into the observed data. This is a result to be avoided at all costs, especially when the observed data are polluted by the measurement noise. An unduly small n_H, on the other hand, makes it difficult to infer a reasonable genetic network. Therefore, we must carefully choose n_H when we use the proposed approach. In order to determine the number of hidden neurons n_H, a k-fold cross-validation that estimates the generalization error of a neural network model is often used. This method divides the training data into k subsets first. Then, a neural network model is trained k times, each time leaving out one of the subsets from the training data, but using only the omitted subset to estimate the generalization error.

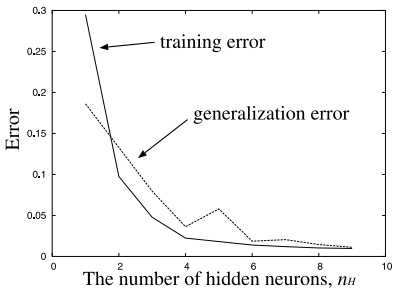

Fig. 4 The training error and the generalization error of the neural network models with different numbers of hidden neurons.

As the given data consisted of 15 sets of time-series data in this study, we applied the 15-fold cross-validation. The averaged training error and the averaged generalization error of the neural network models are plotted against the number of hidden neurons n_H in **Fig. 4**. The training error and the generalization error are the values of the function (5) for the training data (14 subsets) and the test data (one subset), respectively. Although an excessively large n_H generally makes the generalization error worse, the figure shows that it decreases with increasing n_H. As the penalty term introduced into the objective function (5) works to reduce the degree-of-freedom of the neural network models, the generalization error should be hard to get worse in this study. On the other hand, the learning of the neural network models with the large n_H requires high computational effort. Therefore, we used $n_H = 5$ in this study.

We performed an additional experiment without applying the penalty terms introduced into the objective functions (5) and (7) in order to confirm the effectiveness of them. When the penalty terms were omitted, the values of the objective function (7) averaged about $1.663 \times 10^{-4} \pm 8.290 \times 10^{-5}$. Though these objective function values were certainly smaller, network structures inferred were not always reasonable. While the number of false-negative regulations slightly decreases to an average of 3.2 ± 1.2, the number of false-positive regulations increased dramatically, reaching as many as 945.3 ± 17.0 on average. The sensitivity and the specificity without applying the penalty terms were there-

fore 0.953±0.018 and 0.454±0.010, respectively. Our findings indicate that the penalty terms are necessary to obtain reasonable genetic networks when using the proposed methods.

6.2 Random Genetic Network

In the second experiment, we checked the performance of our approach in a noisy real-world setting by conducting the experiment with sets of noisy time-series data.

6.2.1 Experimental Setup

We used the following set of differential equations to describe target networks [28].

$$\frac{dX_i}{dt} = -\lambda_i X_i + \frac{\alpha_i + \sum_{j\in\mathcal{A}_i} X_j^{\gamma_{i,j}}}{1+\sum_{j\in\mathcal{A}_i} X_j^{\gamma_{i,j}} + \sum_{k\in\mathcal{R}_i} X_k^{\beta_{i,k}}}, \quad (i = 1, 2, \cdots, N), \quad (19)$$

where λ_i is the degradation rate of the i-th mRNA, α_i is the synthesis rate of the i-th mRNA, $\gamma_{i,j}$ is the activation cooperativity of the j-th gene on the i-th gene, and $\beta_{i,k}$ is the repression cooperativity of the k-th gene on the i-th gene. The sets $\mathcal{A}_i$ and $\mathcal{R}_i$ specify the genes that activate and repress, respectively, the i-th gene.

As the target networks, we randomly constructed the systems of 10, 20 and 30 genes ($N = 10, 20, 30$). As the inference ability of the proposed approach may depend on the structure of the target network, we changed the network structure on every trial. We generated target networks of different structures by changing the model parameters described above. In order to construct the sets $\mathcal{A}_i$ and $\mathcal{R}_i$, we randomly picked an integer m from a power-law distribution with a cutoff 5. Then, m genes were randomly selected from all of the genes contained in the network. Finally, the indices corresponding to the selected genes were added to the set $\mathcal{A}_i$ or the set $\mathcal{R}_i$ with the probability 0.5. The rate parameters (λ_i and α_i) and the cooperativity parameters ($\gamma_{i,j}$ and $\beta_{i,k}$) are randomly selected from $[0.0, 1.0]$ and $[1.0, 2.0]$, respectively.

The performances of inference methods generally depend on the amount of given time-series data. Therefore, we performed the experiments with different numbers of sets of time-series data. We obtained the sets of time-series data by solving the set of differential Eq. (19). Each set of time-series data began from randomly generated initial values in $[0.0, 2.0]$, and consisted of the expression levels of all of the genes at 11 time points. The measurement noise was simulated by adding 10% Gaussian noise to the computed time-series data. As the given time-series data were polluted by the noise, we used the local linear regression [3] to estimate the differential coefficients of the gene expression levels. All of the other experimental conditions were the same as those used in the previous experiment.

6.2.2 Result

Figure 5 (a), (b) and (c) show the sensitivity and the specificity of the proposed approach on the experiments of the random genetic networks consisting of 10, 20 and 30 genes, respectively. As the figures illustrate, the specificity is almost

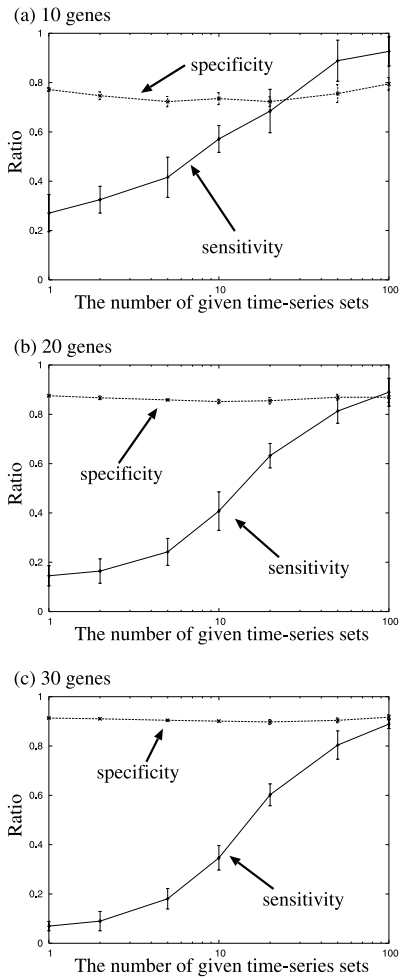

Fig. 5 Performances of the proposed approach on the experiments of random genetic networks consisting of (a) 10 genes, (b) 20 genes, and (c) 30 genes, respectively. Solid line: the sensitivity. Dotted line: the specificity.

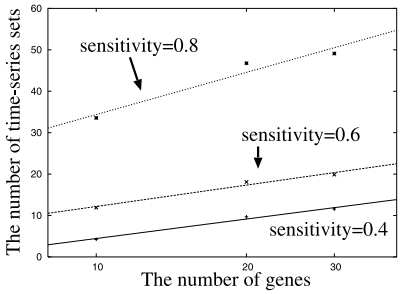

Fig. 6 The amount of time-series sets required by the proposed approach to achieve a certain sensitivity. Data points are estimated from the experimental results of the section 6.2, and lines are obtained by least-squares fitting applied to the data points.

independent of the amount of given time-series data. In order to improve the specificity, we must reduce the number of false-positive regulations. The number of false-positive regulations is, however, difficult to be reduced because of the maximum indegree I used in the penalty terms of the objective functions (5) and (7). Therefore, it should be difficult for our approach to enhance the specificity even when a larger number of time-series sets are available.

The figures also show that the sensitivity improves as the amount of given time-series data increases. When we try to achieve a certain sensitivity in a larger-scale problem, we should give a larger amount of observed data. The number of required time-series sets, however, seems not to be proportional to the number of genes contained in the network. From the experimental results, we estimated the numbers of time-series sets that are required to make the sensitivity 0.4, 0.6 and 0.8 (**Fig. 6**). The figure suggests that, when we try to achieve a certain sensitivity, the amount of time-series sets required by our approach was $O(\log N)$. Yeung and his colleagues [28] revealed that their inference method also requires $O(\log N)$ measurements to infer genetic networks consisting of N genes correctly. Although their method is based on a linear model, our experimental results were consistent with their findings.

7. Conclusion

In this study, we defined the genetic network inference problem as a function approximation problem. On the basis of this problem definition, we proposed a new method to infer neural network models of genetic networks. As the obtained models are not always suitable for computational simulation, we also proposed a method for the refinement of them. Then, we proposed a method based on sensitivity analysis for the interpretation of the obtained neural network models. Numerical experiments proved that our approach is capable of inferring genetic networks correctly. The proposed approach analyzed the gene expression data that cannot be analyzed by the methods based on the linear model, and its computational time was much shorter than those of the methods based on the S-system model. However, as the experimental results show, the proposed approach requires a large number of time-series sets of gene expression levels to infer a genetic network correctly. Therefore, our approach may be still impractical for analyzing actual genetic networks because biological experiments generally provide us with few sets of noisy time-series data. In a future work, we must reduce the amount of observed data needed to obtain a reasonable result.

There is probably no perfect model for the inference of genetic networks. Therefore, in order to extract reliable information from the observed gene expression patterns, it may be important to describe the genetic network using many different models. The neural network is one of the promising models for this purpose.

Acknowledgments We thank Mr. Hideki Terasawa from JFE SOLDEC Corporation for his fruitful discussions.

References

1) Akutsu, T., Miyano, S. and Kuhara, S.: Inferring Qualitative Relations in Genetic Networks and Metabolic Pathways, *Bioinformatics*, Vol.16, pp.727–734 (2000).
2) Cho, D.-Y., Cho, K.-H. and Zhang, B.-T.: Identification of Biochemical Networks by S-tree Based Genetic Programming, *Bioinformatics*, Vol.22, pp.1631–1640 (2006).
3) Cleveland, W.S.: Robust Locally Weight Regression and Smoothing Scatterplots, *J. of American Statistical Association*, Vol.79, pp.829–836 (1979).
4) DeRisi, J.L., Iyer, V.R. and Brown, P.O.: Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale, *Science*, Vol.278, pp.680–686 (1997).
5) D'haeseleer, P., Liang, S. and Somogyi, R.: Genetic network inference: From co-expression clustering to reverse engineering, *Bioinformatics*, Vol.16, pp.707–726 (2000).
6) Gardner, T.S., Bernardo, D., Lorenz, D. and

Collins, J.J.: Inferring Genetic Networks and Identifying compound Mode of Action via Expression Profiling, *Science*, Vol.301, pp.102–105 (2003).
7) Imoto, S., Goto, T. and Miyano, S.: Estimation of genetic networks and functional structures between genes by using Bayesian network and nonparametric regression, *Proc. Pacific Symposium on Biocomputing*, Vol.7, pp.175–186 (2002).
8) Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K. and Tomita, M.: Dynamic Modeling of Genetic Networks using Genetic Algorithm and S-system, *Bioinformatics*, Vol.19, pp.643–650 (2003).
9) Kimura, S. and Konagaya, A.: High Dimensional Function Optimization using a new Genetic Local Search suitable for Parallel Computers, *Proc.2003 Conference on Systems, Man and Cybernetics*, pp.335–342 (2003).
10) Kimura, S., Hatakeyama, M. and Konagaya, A.: Inference of S-system Models of Genetic Networks from Noisy Time-series Data, *Chem-Bio Informatics J.*, Vol.4, pp.1–14 (2004).
11) Kimura, S., Ide, K., Kashihara, A., Kano, M., Hatakeyama, M., Masui, R., Nakagawa, N., Yokoyama, S., Kuramitsu, S. and Konagaya, A.: Inference of S-system Models of Genetic Networks using a Cooperative Coevolutionary Algorithm, *Bioinformatics*, Vol.21, pp.1154–1163 (2005).
12) Kimura, S., Sonoda, K., Yamane, S., Matsumura, K. and Hatakeyama, M.: Inference of Genetic Networks using Neural Network Models, *Proc. 2005 Congress on Evolutionary Computation*, pp.1738–1745 (2005).
13) Maki, Y., Tominaga, D., Okamoto, M., Watanabe, S. and Eguchi, Y.: Development of a System for the Inference of Large Scale Genetic Networks, *Proc. Pacific Symposium on Biocomputing*, Vol.6, pp.446–458 (2001).
14) Maki, Y., Ueda, T., Okamoto, M., Uematsu, N., Inamura, Y. and Eguchi, Y.: Inference of Genetic Network Using the Expression Profile Time Course Data of Mouse P19 Cells, *Genome Informatics*, Vol.13, pp.382–383 (2002).
15) Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P.: *Numerical Recipes in C*, 2nd ed., Cambridge University Press (1995).
16) Rumelhart, D.E., McClelland, J.L. and PDP Research Group: *Parallel Distributed Processing*, MIT Press (1986).
17) Sakamoto, E. and Iba, H.: Inferring a System of Differential Equations for a Gene Regulatory Network by using Genetic Programming, *Proc. 2001 Congress on Evolutionary Computation*, pp.720–726 (2001).
18) Shiraishi, F. and Savageau, M.A.: The Tricarboxylic Acid Cycle in *Dictyostelium discoideum*, *J. of Biological Chemistry*, Vol.267, pp.22912–22918 (1992).
19) Thieffry, D., Huerta, A.M., Pérez-Rueda, E. and Collado-Vides, J.: From Specific Gene Regulation to Genomic Networks: a Global Analysis of Transcriptional Regulation in *Escherichia Coli, BioEssays*, Vol.20, pp.433–440 (1998).
20) Tominaga, D. and Horton, P.: Inference of Scale-free Networks from Gene Expression Time Series, *J. of Bioinformatics and Computational Biology*, Vol.4, pp.503–514 (2006).
21) Tsai, K.Y. and Wang, F.S.: Evolutionary Optimization with Data Collocation for Reverse Engineering of Biological Networks, *Bioinformatics*, Vol.21, pp.1180–1188 (2005).
22) Vance, W., Arkin, A. and Ross, J.: Determination of Causal Connectivities of Species in Reaction Networks, *Proc. Natl. Acad. Sci. USA*, Vol.99, pp.5816–5821 (2002).
23) Voit, E.O.: *Computational Analysis of Biochemical Systems*, Cambridge University Press (2000).
24) Voit, E.O.: Models-of-data and Models-of-processes in the Post-genomic Era, *Math. Biosci.*, Vol.180, pp.263–274 (2002).
25) Voit, E.O. and Almeida, J.: Decoupling Dynamical Systems for Pathway Identification from Metabolic Profiles, *Bioinformatics*, Vol.20, pp.1670–1681 (2004).
26) Weaver, D.C., Workman, C.T. and Stormo, G.D.: Modeling Regulatory Networks with Weight Matrices, *Proc. Pacific Symposium on Biocomputing*, Vol.4, pp.112–123 (1999).
27) Yao, X.: Evolving Artificial Neural Networks, *Proc. IEEE*, Vol.87, pp.1423–1447 (1999).
28) Yeung, M.K.S., Tegnér, J. and Collins, J.J.: Reverse engineering gene networks using singular value decomposition and robust regression, *Proc.Natl.Acad.Sci.USA*, Vol.99, pp.6163–6168 (2002).

(Received November 13, 2006)
(Accepted December 18, 2006)

(Communicated by *Masakazu Sekijima*)

Shuhei Kimura received his M.E. degree from Kyoto University in 1998 and Ph.D. degree from Tokyo Institute of Tech. in 2001. He had been in RIKEN Genomic Sciences Center as a research scientist since 2001. Since 2004, he has been in Tottori University as an associate professor. His current research interests are evolutionary algorithms and bioinformatics.

Katsuki Sonoda received his M.E. degree from Kyoto University in 1994. He had been working in JFE Engineering Corp. since 1994. Since 2006 he has been working in JFE R&D Corp. His current research interests are artificial intelligence technology and numerical simulation technology.

Soichiro Yamane received his M.E. degree from Kyoto University in 1992. He had been working in JFE Engineering Corp. since 1992. Since 2006 he has been working in JFE R&D Corp. His current research interests are artificial intelligence technology and numerical simulation technology.

Koki Matsumura received his Ph.D. degree from Osaka City University in 1978. He had been in Ibaraki University as an associate professor, and Konan University as a professor. Since 2002, he has been in Tottori University as a professor. His current research interests are evolutionary algorithms, knowledge engineering, study of management information and financial engineering.

Mariko Hatakeyama received Ph.D. from Tokyo University of Agriculture and Technology (major in Biochemistry). She is the Team Leader of the Cellular Systems Biology Team in RIKEN Genomic Sciences Center, Japan. She was previously a Visiting Researcher at the Institute of Toxicology and Environmental Health, University of California, Davis, USA. Prior to this she was Project Leader of Novo Nordisk Bioindustry (Japanese R&D branch of Novo Nordisk A/S, Denmark).