Mathematical Biosciences 242 (2013) 153–160

Contents lists available at SciVerse ScienceDirect

Mathematical Biosciences

journal homepage: www.elsevier.com/locate/mbs

Adaptive neural-based fuzzy modeling for biological systems

Shinq-Jen Wu [a,*], Cheng-Tao Wu [b], Jyh-Yeong Chang [b]

[a] Department of Electrical Engineering, Da-Yeh University, Chang-Hwa, Taiwan, ROC
[b] Department of Electrical and Control Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan, ROC

ARTICLE INFO

Article history:
Received 5 June 2012
Received in revised form 1 January 2013
Accepted 11 January 2013
Available online 31 January 2013

Keywords:
Reverse engineering
T–S fuzzy system
Neural-fuzzy modeling
System identification

ABSTRACT

The inverse problem of identifying dynamic biological networks from their time-course response data set is a cornerstone of systems biology. Hill and Michaelis–Menten model, which is a forward approach, provides local kinetic information. However, repeated modifications and a large amount of experimental data are necessary for the parameter identification. S-system model, which is composed of highly nonlinear differential equations, provides the direct identification of an interactive network. However, the identification of skeletal-network structure is challenging. Moreover, biological systems are always subject to uncertainty and noise. Are there suitable candidates with the potential to deal with noise-contaminated data sets? Fuzzy set theory is developed for handing uncertainty, imprecision and complexity in the real world; for example, we say "driving speed is high" wherein *speed* is a fuzzy variable and *high* is a fuzzy set, which uses the membership function to indicate the degree of a element belonging to the set (words in Italics to denote fuzzy variables or fuzzy sets). Neural network possesses good robustness and learning capability. In this study we hybrid these two together into a neural-fuzzy modeling technique. A biological system is formulated to a multi-input-multi-output (MIMO) Takagi–Sugeno (T–S) fuzzy system, which is composed of rule-based linear subsystems. Two kinds of smooth membership functions (MFs), Gaussian and Bell-shaped MFs, are used. The performance of the proposed method is tested with three biological systems.

1. Introduction

Computational systems biology is an active research area. Various modeling techniques were proposed to analyze and identify a broad range of biological networks such as gene regulation networks, metabolic pathways and signal transduction cascades. The first motivation of biological-network modeling is to get a synthetic view of biological knowledge existing in a network. The second motivation is to achieve the prediction of the dynamic behavior of a system. However, a biological system always contains many components. It is difficult to get a mathematical model to describe the biological system. Hill and Michaelis–Menten rate modeling is a forward approach to provide local kinetic information. However, repeated modifications and a large amount of experimental data are necessary for the parameter identification, partially for the analysis of a system with many substances or reactions. Chou and Voit [1] proposed a systematic approach of dynamic flux estimation to achieve the parameter and function-form identification. S-system structure is another popular nonlinear dynamic model. The model uniquely maps the dynamic interaction onto its parameters and possesses good generalization characteristics. Wang et al. [2] considered two extreme cases and proposed a two-step method to determine the ranges and the mean values of the parameters. Marino and Voit [3] wrote an algorithm to gradually increase the complexity of the model. Chou et al. [4] proposed an alternating regression method. Recently, many researchers have adopted various stochastic-search intelligent technologies, such as genetic programming [5], hybrid differential evolution [6,7], hybrid genetic algorithm and simulated annealing [8], and a neural network with particle swarm optimization [9]. However, as the number of system states increases the parameter identification becomes increasing difficult and finding a solution becomes increasingly problematic. Furthermore, the pruning of the redundant kinetic orders to infer a suitable network structure is a large challenge. Various penalty terms were introduced [10–12]. Chou et al. [13] and Sun et al. [14] provided a comprehensive review of various approaches that been developed for the identification of S-system parameters. These researchers also indicated some important issues and suggested possible search directions.

Fuzzy-based approaches are better candidates in dealing with uncertainty-rich and noise-contaminated systems [7]. T–S fuzzy systems were demonstrated to be the universal approximations of smooth nonlinear systems, and have been applied to various physical systems. This system, which is basically a locally

* Corresponding author. Tel.: +886 4 8511888; fax: +886 4 8511245.
E-mail address: jen@mail.dyu.edu.tw (S.-J. Wu).

0025-5564/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.mbs.2013.01.004

linearized fuzzy model, describes the global behavior of a nonlinear system by fuzzily blending the linear subsystems. Two theoretically modeling methods (locally linear approximation and sector-nonlinearity approaches) were proposed. However, these two approaches are only suitable for model-based systems (systems with mathematical models). Furthermore, if the nonlinear system is too complex, then theoretically converting its mathematical model into a T–S fuzzy system is also impractical. In our previous studies [15,16], we proposed two neural-fuzzy modeling techniques for model-free physical systems. In this study, a fuzzy-based neural framework, which integrates fuzzy reasoning with neural-network learning, is used to approximate MIMO biological systems. Noise, uncertainty and the interactive information are all implied in fuzzy if-then rules. The rules are inferred through step-wise neural learning.

In this study, an adaptive T–S type neural-fuzzy modeling (ATNM) network is presented in Section 2. This network possesses both fuzzy-reasoning and neural-learning abilities. Section 3 describes the performance of ATNM in the identification of parameters and structure of three biological systems. The comparison of performance for different types of MFs and different numbers of input-space partition are shown in this section. Section 4 summarizes the results of our research and suggests areas for future work.

2. Adaptive T–S type neural-fuzzy modeling (ATNM)

The dynamic behavior of a physical or biological system is often described as nonlinear differential equations:

$$\frac{dx_i}{dt} = f_i(x,p) \quad x = (x_1,\ldots,x_n), \quad p = (p_1,\ldots,p_n), \quad i = 1\ldots n, \qquad (1)$$

where the state variable x_i denotes the concentration or the activity level of the ith component, f_i describes how the rate change of x_i relates to other components, and p_i, $i = 1,\ldots,n$ are the constants denoting kinetic orders, rate constants, pH value, temperature... etc. The purpose of T–S type neural-fuzzy modeling is to map the time-series data sets onto a network in which a T–S type fuzzy system (composed of fuzzy rules with linear state-dependent consequences) is implied. After learning, the rule-based neural fuzzy scheme is able to simulate the dynamic behavior of the real system. The following equation describes the lth rule (the lth subsystem, $l = 1,\ldots,r$) of the T–S fuzzy system:

$$\begin{aligned} &\text{Rule } R^l: \textbf{IF } x_1 \text{ is } A_l, x_2 \text{ is } B_l, \ldots, x_n \text{ is } Z_l, \\ &\textbf{THEN } \dot{x}_1 = f_{l1} = p_{l11}x_1 + p_{l12}x_2 + \cdots + p_{l1n}x_n + p_{l10}, \\ &\qquad \ldots \\ &\qquad \dot{x}_n = f_{ln} = p_{ln1}x_1 + p_{ln2}x_2 + \cdots + p_{lnn}x_n + p_{ln0}, \end{aligned} \qquad (2)$$

where $A_l, B_l, \ldots, Z_l$ (e.g., *high*, *medium*,...,*very high*) are the term sets of the linguistic variables $x_1, x_2, \ldots, x_n$, respectively, and p_{lij}, $i = 1,\ldots,n$ and $j = 0,1,\ldots,n$ are the consequence parameters. The T–S fuzzy model is basically a locally linearized fuzzy model. The dynamic behavior of the whole system is obtained by *fuzzily blending* the rule-based linear subsystems in Eq. (2). In other words, the dynamic behavior of the fuzzy system is

$$\dot{X}(t) = \sum_{l=1}^{r} h_l(X(t)) \cdot (P_l \cdot X(t) + P_{l0}), \qquad (3)$$

where the normalized firing strength $h_l(X(t))$ is the measure of the contribution of the lth fuzzy rule to the whole system, $X = [x_i]$, $i = 1,\ldots,n$ is the state vector, and $P_l = [p_{lij}]$ and $P_{l0} = [p_{li0}]$, $i = 1,\ldots,n$, $j = 1,\ldots,n$ denote the consequence parameters.

The learning starts at the network (system) with given connection factors and adaptively adjusts the factors such that the inferred system exhibits the dynamic behavior of the real system. Values of system variables (inputs, outputs or states) are real numbers (points in a real line, $x_i = 0.3$) for the crisp system, but are fuzzy sets (points in a fuzzy space, $x_i = medium$) for the fuzzy system. The membership function of the fuzzy set *medium* at point 0.3 (denoting the degree of 0.3 belonging to *medium*) is described as $\mu_{medium}(x = 0.3)$ or simply $\mu_{medium}(0.3)$. Notice that the membership function values are between 0 and 1 instead of 0 or 1. In other words, fuzzy sets introduce vagueness by eliminating the sharp boundary, which divides members from nonmembers in a group [17]. For the biological system subject to noise and uncertainty the measured values of x_i cannot really reflect the true concentration. Therefore, the expression of "$x_i = medium$" is much more suitable than that of "$x_i = 0.3$".

We now explain fuzzy reasoning in brief. We regard the measured crisp data $(x_1,\ldots,x_n) = (0.05,\ldots,0.91)$ as a FACT. We then estimate the matching degree of the FACT to the lth fuzzy rule; i.e., the matching degree of $(x_1,\ldots,x_n) = (0.05,\ldots,0.91)$ to $(A_l,\ldots,Z_l) = (low,\ldots,very\ high)$, which is the precondition of the lth fuzzy rule. The matching degree of the FACT to the lth rule equals 0.23 implies that the possibility of the FACT to fire the lth rule is 23%; i.e., the normalized firing strength $h_i(x(t)) = 0.23$. By using the Larsen-product implication rule [17], the contribution of the lth rule to the rate change $\dot{X}(t)$ is 0.23 $(P_l \cdot X(t) + P_{l0})$, where $X = (0.05,\ldots,0.91)^T$. Finally, we *union* the contributions from all of the rules to get total concentration change of the state vector X.

Fig. 1 describes a five-layer neural-fuzzy inference scheme for realizing a two-input-two-output T–S fuzzy system, wherein x_1, x_2 are the input variables and $f_1 = \dot{x}_1$, $f_2 = \dot{x}_2$ are the output variables. For clarity only two fuzzy rules with two input-space partitions are shown. The crisp data are fed into the net and are

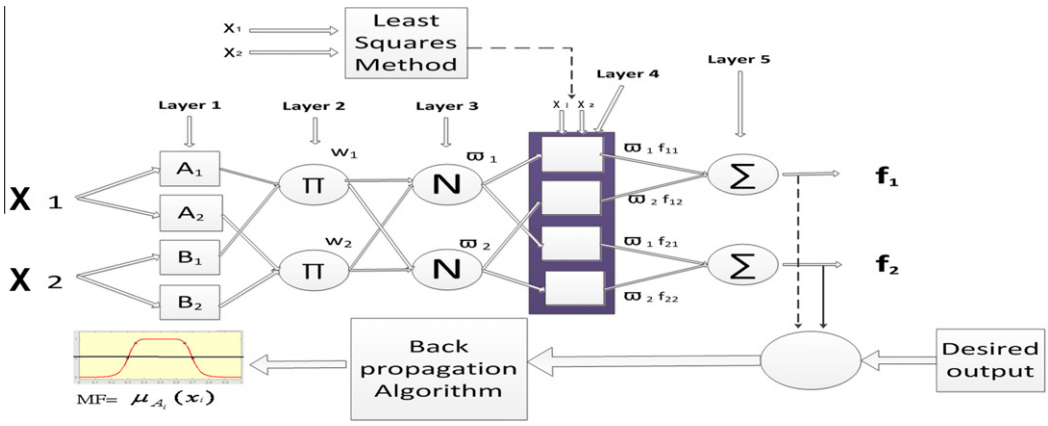

Fig. 1. The adaptive T–S type neural-fuzzy modeling (ATNM) scheme.

fuzzified in Layer 1. Each node in this layer denotes a fuzzy set. In this report we adopted Gaussian- and Bell-shaped functions as the membership functions, and compared their performance in identifying three biological systems. Each node in Layer 2 corresponds to a fuzzy rule. All of the Layer-2 nodes form a fuzzy system. The signal function w_i of the node corresponds the firing strength of the rule. In Layer 3, the signal function is normalized to $\bar{w}_i$ (the normalized firing strength $h_i(x(t))$, $i = 1,\ldots,r$, where r is the number of fuzzy rules). The fan-in links of the node in Layer 2 define the precondition of the rule, and the fan-out links of the respective node in Layer 3 define the consequence of the rule. Layers 4 and 5 execute, respectively, the Larsen-product implication operation and the *union* operation (a simply summation operation). The input signals grow forward to generate the output signals. Conversely, the error signals (showing the deviation of the estimated outputs from the desired outputs) go backward to adjust the parameters in the network (the parameters of the fuzzy rules). The least-square method is used to identify the consequence-parameter matrices (P_l and P_{l0}). The back-propagation algorithm is used to identify the parameters of the Gaussian or the Bell-shaped membership functions. We now describe the connected structure of ATNM scheme in more detail. For scheme simplification, only two of the fuzzy rules are shown in Fig. 1.

Layer 1: The input variables (x_1 and x_2) are fuzzified in this layer. Each node (A_i or B_i, $i = 1,2$) denotes a fuzzy set. A_1 and A_2 are the term sets of x_1. B_1 and B_2 are the term sets of x_2. If we choose the Bell-shaped distribution as the membership function, then the output of the node ji ($O^1_{ji} = \mu_j(x_i)$) is estimated by Eq. (4). $\mu_j(x_i)$ is the membership function of the jth term set of the fuzzy variable x_i; i.e., $\mu_1(x_1) = \mu_{A_1}(x_1)$, $\mu_2(x_1) = \mu_{A_2}(x_1)$; $\mu_1(x_2) = \mu_{B_1}(x_2)$, $\mu_2(x_2) = \mu_{B_2}(x_2)$. If the Gaussian distribution is used, then the output of the node becomes Eq. (5). The unknown parameters (a_{ji}, b_{ji}, c_{ji}) and (σ_{ji}, m_{ji}) are the premise parameters of the fuzzy rules

$$O^1_{ji} = \mu_j(x_i) = \frac{1}{1 + \left(\frac{x_i - c_{ji}}{a_{ji}}\right)^{2b_{ji}}}. \quad (\text{Bell} - \text{shaped}) \tag{4}$$

$$O^1_{ji} = \mu_j(x_i) = \exp\left\{-\left(\frac{x_i - m_{ji}}{\sigma_{ji}}\right)^2\right\}. \quad (\text{Guassian}) \tag{5}$$

Layer 2: Each node in this layer denotes a fuzzy logic rule. This layer performs the intersection operation of the fuzzy sets A_i and B_j; *i.e.*, $A_i \cap B_j$. We use the simple algebraic-product operation in Eq. (6) as the intersection operation. Notice that all of the fan-in connection weights equal to one. The node output w_l denotes the firing strength of the fuzzy rule:

$$O^2_l = w_l = \prod_{i=1}^{n} \mu_j(x_i), \quad l = 1,\ldots,r. \tag{6}$$

Layer 3: Normalization is performed in this layer. The resulting signal function of the lth node $\bar{w}_l$ is the normalized fire strength of the lth rule $h_l(x(t))$:

$$O^3_l = \bar{w}_l = \frac{w_l}{\Sigma_{i=1}^{r} w_i}, \quad l = 1, \ldots, r. \tag{7}$$

Layer 4: This layer is the consequence layer. The consequent parameters p_{lji}, $j = 1,\ldots,n$, $i = 0,1,\ldots,n$ are included in node parameters f_{lj}, which denotes the jth consequence of the lth rule. We adopt the Larsen-product implication operation [17] to infer the output of the rule:

$$O^4_{lj} = \bar{w}_l f_{lj} = \bar{w}_l(\Sigma_{i=1}^{n} p_{lji} x_i + p_{lj0}). \tag{8}$$

Layer 5: In this layer, the output signal of the node corresponds to the system output; i.e., $O^5_j = \dot{x}_j$, $j = 1,\ldots,n$. This layer performs the integration of the fan-out streams of Layer 4. We use the summation operation for the rule-union operation ($R_1 \cup \ldots \cup R_r$):

$$O^5_j = f_j = \sum_{l=1}^{r} \bar{w}_l f_{lj}, \quad j = 1,\ldots,n. \tag{9}$$

The network architecture consists of five layers in which the unknown parameters of the node are identified through a hybrid algorithm. The identification of parameters includes two-phase learning: Phase one is to identify the consequence parameters of the fuzzy rules (p_{lij} and p_{0ij}, $i = 1,\ldots,n$, $j = 1,\ldots,n$). Phase two is to identify the parameters associated with the membership functions (a_{ji}, b_{ji}, c_{ji}) or (σ_{ji}, m_{ji}). In the case of the premise parameters fixed, the system outputs are simply a linear combination of the consequence parameters. We can get the optimal consequent parameters by the least-square method: Given K pairs of input-and-output training data ($\mathbf{x}^k, \mathbf{y}^k_d$), $k = 1,\ldots,$K, where the input $\mathbf{x}^k = (x^k_1,\ldots,x^k_n)^T$ and the desired output $\mathbf{y}^k_d = (y^k_{1,d},\ldots,y^k_{n,d})^T$, we have n-times-$K$-set linear equations in terms of the consequence parameters. The desired outputs become

$$\begin{aligned} y^k_{j,d} &= \sum_{l=1}^{r} \bar{w}^k_l \left(\sum_{i=1}^{n} p_{lji} x^k_i + p_{lj0}\right) = \sum_{l=1}^{r} \bar{w}^k_l \left(\sum_{i=0}^{n} p_{lji} x^k_i\right) \\ &= \sum_{l=1}^{r} \sum_{i=0}^{n} (\bar{w}^k_l x^k_i) \cdot p_{lji} \end{aligned} \tag{10}$$

for $j = 1,\ldots,n$, where $x^k_0 = 1$, $k = 1,\ldots,$K. In other words,

$$\mathbf{Y}^d = [Y_{1,d} \cdots Y_{n,d}]^T = \mathbf{\Lambda} \times \mathbf{\Pi} = \begin{bmatrix} A & 0 & \ldots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & A & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & A \end{bmatrix} \times \begin{bmatrix} P_1 \\ \vdots \\ P_j \\ \vdots \\ P_n \end{bmatrix}, \tag{11}$$

where $Y_{j,d} = [y^1_{j,d} \cdots y^K_{j,d}]^T = A \cdot P_j$, $j = 1,\ldots,n$ and

$$\begin{aligned} A &= \begin{bmatrix} \bar{w}^1_1 & \bar{w}^1_1 x^1_1 \cdots & \bar{w}^1_1 x^1_n & \cdots\cdots & \bar{w}^1_r & \bar{w}^1_r x^1_1 \cdots & \bar{w}^1_r x^1_n \\ \vdots & & & \cdots\cdots & & & \vdots \\ \vdots & & & \cdots\cdots & & & \vdots \\ \bar{w}^K_1 & \bar{w}^K_1 x^K_1 \cdots & \bar{w}^K_1 x^K_n & \cdots\cdots & \bar{w}^K_r & \bar{w}^K_r x^K_1 \cdots & \bar{w}^K_r x^K_n \end{bmatrix}, \\ P_j &= \begin{bmatrix} p_{1j0} & p_{1j1} \cdots & p_{1jn} & \cdots & \cdots & p_{rj0} & p_{rj1} \cdots & p_{rjn} \end{bmatrix}^T. \end{aligned} \tag{12}$$

The sizes of the desired output vector $\mathbf{Y}^d$ and the unknown parameter matrix $\mathbf{\Pi}$ are nK $\times$ 1 and $nr(n+1) \times 1$, respectively. Usually the number of input–output patterns used for training (K) is larger than that of the unknown consequence parameters ($r(n+1)$). The optimal solution Π^* which is obtained from minimizing the square error $\|\mathbf{Y}^d - \mathbf{\Lambda} \times \mathbf{\Pi}\|^2$ by using the pseudo-inverse operation, is

$$\mathbf{\Pi}^* = (\mathbf{\Lambda}^T \mathbf{\Lambda})^{-1} \mathbf{\Lambda}^T \mathbf{Y}^d. \tag{13}$$

We now proceed to the second-phase learning to adjust the parameters of the membership functions. We estimate the network output $y^k_j = \sum_{l=1}^{r} \sum_{i=0}^{n} (\bar{w}^k_l x^k_i) \cdot p^*_{lji}$ first, and then the error $e^k_j = y^k_{j,d} - y^k_j$, $j = 1,\ldots,n$, $k = 1,\ldots,$K. The error signals are fed backward to update the premise parameters by using the chain rule. The goal of the learning in phase two is to minimize the mean-square error (MSE),

$$\mathbf{E} = \frac{1}{2nK} \sum_{j=1}^{n} \sum_{k=1}^{K} (e^k_j)^2 = \frac{1}{2nK} \sum_{j=1}^{n} \sum_{k=1}^{K} (y^k_{j,d} - y^k_j)^2. \tag{14}$$

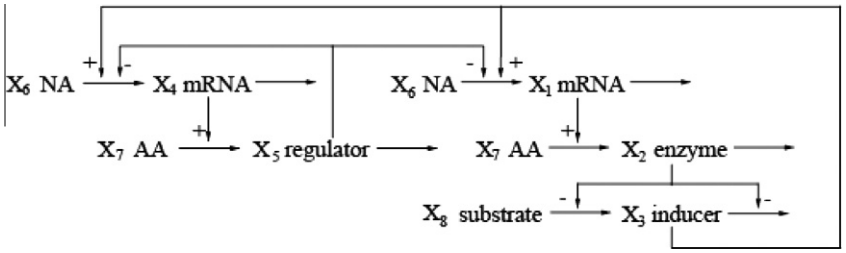

Fig. 2. The small-scale genetic network [18].

Table 1
Comparison for the small-scale gene network.

Case	MF	No. of input-space partitions	No. of premise-para./rule	Rule No.	Total of premise para.	MSE (training)	MSE (testing)
B_2	Bell	2	3^5	2^5	7776	0.014590	0.204038
B_3	Bell	3	3^5	3^5	59049	0.009538	0.330193
G_2	Gaussian	2	2^5	2^5	1024	0.016477	0.245342
G_3	Gaussian	3	2^5	3^5	7776	0.006545	0.232094

Let α denote one of the premise parameters (a_{ji}, b_{ji}, c_{ji}) or (σ_{ji}, m_{ji}). By using the gradient-descent method we have

$$\begin{aligned}\Delta\alpha &= -\eta\cdot\tfrac{\partial \mathbf{E}}{\partial\alpha},\\ \alpha(t+1) &= \alpha(t)+\eta(-\tfrac{\partial \mathbf{E}}{\partial\alpha}),\end{aligned} \tag{15}$$

where η is a learning rate. By using the chain rule, $\partial\mathbf{E}/\partial\alpha = \partial\mathbf{E}/\partial a(\cdot)\times\partial a(\cdot)/\partial\alpha$, we get

$$\Delta\alpha = \frac{1}{nK}\sum_{k=1}^{K}(y_{j,d}^k - y_j^k)\frac{\partial y_j^k}{\partial\alpha} = \frac{1}{nK}\sum_{k=1}^{K}(y_{j,d}^k - y_j^k)\sum_{i=1}^{n}p_{\ell ji}^* x_i^k\frac{\partial\bar{\omega}_\ell}{\partial\alpha}$$

$$\frac{\partial\bar{\omega}_l}{\partial\alpha} = \frac{\partial w_\ell}{\partial\alpha}\left(\frac{1}{\sum_{\ell=1}^{r}w_\ell}\right) - \frac{w_\ell}{(\sum_{\ell=1}^{r}w_\ell)^2}\cdot\frac{\partial(\sum_{\ell=1}^{r}w_\ell)}{\partial\alpha} \tag{16}$$

If $w_l = \mu_{A_1}(x_1)\cdot\mu_{B_2}(x_2)$ and α is one of the unknown parameters of the membership function $\mu_{A_1}(x_1)$ (a bell-shape function denoted as $bell(a_{11}, b_{11}, c_{11})$), then

$$\begin{aligned}\tfrac{\partial w_\ell}{\partial\alpha} &= \mu_{B_2}(x_2)\cdot\tfrac{\partial\mu_{A_1(x_1)}}{\partial\alpha}\\ \tfrac{\partial(\sum_{\ell=1}^{r}w_\ell)}{\partial\alpha} &= [\mu_{B_1(x-2)}+\mu_{B_2(x_2)}]\tfrac{\partial\mu_{A_1(x_1)}}{\partial\alpha}\\ \tfrac{\partial\mu_{A_1(x_1)}}{\partial a_{11}} &= 2a_{11}^{-2b_{11}^{-1}}\cdot b_{11}(x_1-c_{11})^{2b_{11}}(1+|\tfrac{x_1-c_{11}}{a_{11}}|^{2b_{11}})^{-2}\\ \tfrac{\partial\mu_{A_1(x_1)}}{\partial b_{11}} &= 2\cdot|\tfrac{x_1-c_{11}}{a_{11}}|^{2b_{11}}\cdot\ell n|\tfrac{x_1-c_{11}}{a_{11}}|\cdot(1+|\tfrac{x_1-c_{11}}{a_{11}}|^{2b_{11}})^{-2}\\ \tfrac{\partial\mu_{A_1(x_1)}}{\partial c_{11}} &= 2a_{11}^{-2b_{11}}\cdot b_{11}(x_1-c_{11})^{2b_{11}^{-1}}(1+|\tfrac{x_1-c_{11}}{a_{11}}|^{2b_{11}})^{-2}.\end{aligned} \tag{17}$$

3. Dry-lab experiments

We used the adaptive neural-fuzzy-modeling technique to identify three biological systems. The performance of the membership functions (bell-shaped and Gaussian) with the different numbers of input-space partition is discussed. The true time-series data sets were generated by solving the S-systems in [18–20]. The ode45 solver in Matlab toolbox was used to get the time-series data for training and testing. The eight-set training data sets were generated from eight different initial conditions, and the testing data was generated from another initial condition. The simulation time was set from $t = 0$ s to $t = 5$ s or to $t = 8$ s.

3.1. A small-scale genetic network

The used small-scale genetic network is shown in Fig. 2 [18]. This is a typical gene-interaction system, which consists of two genes (genes 1 and 4). X_1 is the mRNA generated from gene 1. X_2 is the respective enzyme protein. X_3 (an inducer protein) is catalyzed by X_2. X_4 is another mRNA, which is generated from gene 4. X_5 is the respective regulator protein. Both positive- and negative-feedback signals exist in the mRNA production of genes 1 and 4.

We considered four different cases, as shown in Table 1. Case B_2 is the case of the bell-shaped membership function with two input-space partitions. For this five-state (five-gene) system, the number of premise parameter for each rule is 3^5 and the number of fuzzy rules is 2^5. Therefore, the total number of premise parameters to be identified (denoted by $\emptyset$) is $3^5\times 2^5$ for Case B_2, $3^5\times 3^5$ for Case B_3, $2^5\times 2^5$ for Case G_2, and $2^5\times 3^5$ for Case G_3; i.e., $\emptyset(B\_3) > \emptyset(B\_2) = \emptyset(G\_3) > \emptyset(G\_2)$ The mean-square error $E(G\_3) < E(B\_3) < E(B\_2) < E(G\_2)$ in the training phase, but $E(B\_2) < E(G\_3) < E(G\_2) < E(B\_3)$ in the testing phase. Although Case B_3 uses the most number of premise parameters $(\emptyset(B\_3) = 59,049)$ and shows good learning performance in the

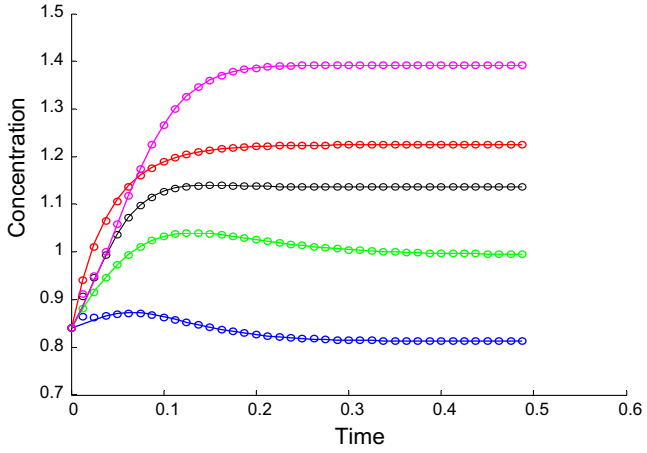

Fig. 3. The dynamic behavior of the small-scale gene network for Case B_2 (inputs with two fuzzy partitions and bell-shaped MFs). The solid curves are the profiles of the data generated from S-system in [18]. "o" denotes the estimated data.

Table 2
The term sets and the estimated membership functions for Case B_2.

Term set	Bell-shaped MFs
A_1	$\mu_{A_1}(x_1) = \text{bell}(a_{11}, b_{11}, c_{11}) = \text{bell}(0.3097191084, 1.9819508769, 0.0967951710)$
A_2	$\mu_{A_2}(x_1) = \text{bell}(a_{21}, b_{21}, c_{21}) = \text{bell}(0.0513495425, 2.0333314758, 0.6710566149)$
B_1	$\mu_{B_1}(x_2) = \text{bell}(a_{12}, b_{12}, c_{12}) = \text{bell}(0.2029263954, 1.9739584207, 0.0802034926)$
B_2	$\mu_{B_1}(x_2) = \text{bell}(a_{22}, b_{22}, c_{22}) = \text{bell}(1.7014551266, 1.9729088617, 3.2196524445)$
C_1	$\mu_{C_1}(x_3) = \text{bell}(a_{13}, b_{13}, c_{13}) = \text{bell}(0.2107807831, 1.9876189708, 0.1140792596)$
C_2	$\mu_{C_1}(x_3) = \text{bell}(a_{23}, b_{23}, c_{23}) = \text{bell}(0.1755185005, 2.0282764361, 0.6715300071)$
D_1	$\mu_{D_1}(x_4) = \text{bell}(a_{14}, b_{14}, c_{14}) = \text{bell}(0.2664561431, 1.9752868187, 0.1185672835)$
D_2	$\mu_{D_1}(x_4) = \text{bell}(a_{24}, b_{24}, c_{24}) = \text{bell}(0.1442955986, 2.0416071274, 0.7004080389)$
E_1	$\mu_{E_1}(x_5) = \text{bell}(a_{15}, b_{15}, c_{15}) = \text{bell}(0.2073412894, 1.9222169628, 0.0917616417)$
E_2	$\mu_{E_1}(x_5) = \text{bell}(a_{25}, b_{25}, c_{25}) = \text{bell}(0.1912147910, 2.0955691554, 0.5671200835)$

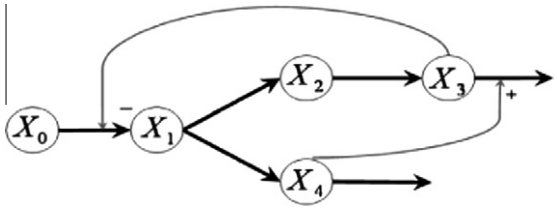

Fig. 4. The branch pathway [19].

training phase, it performs badly in the testing phase. Therefore, in the limited computation time the number of parameters to be identified should not be too large. Below, we compare the performance of Cases G_3, B_2 and B_2 only. As comparing the cases of the same number of unknown parameters (Cases B_2 and G_3), we observe that even if Case B_2 performs worse than Case G_3 in the training phase, it shows better in the testing phase. For the same number of input partitions, Case B_2 performs better than Case G_2 in both training and testing phases. In both training and testing phases Case G_3 performs better than Case G_2. Fig. 3 shows the dynamic behavior of Case B_2, where the circle points are the estimated data and the solid curves are the profiles of the data generated from the S-system in [18]. Table 2 shows the estimated membership functions. *Due to space limitation* Table 3 only shows three of the inferred thirty-two fuzzy rules. The complete results of the rules and the simulation results of Cases B_3, G_2 and G_3 are shown in supplement file.

The inferred fuzzy system is composed of thirty-two fuzzy rules. The 17-th rule in Table 3 denotes the seventeen fuzzy subsystem: If $\boldsymbol{x_1}$ is A_2, $\boldsymbol{x_2}$ is B_1, $\boldsymbol{x_3}$ is C_1, $\boldsymbol{x_4}$ is D_1 and $\boldsymbol{x_5}$ is E_1, then

$$\dot{\mathbf{X}} = \begin{bmatrix} 31.792603 & 4.996830 & 6.172297 & 6.623522 & 8.456148 \\ 23.093542 & 2.808661 & 0.868960 & 0.680079 & 20.396518 \\ -0.326283 & -0.134161 & -1.665026 & -1.056774 & 0.995313 \\ -1.917613 & 0.040022 & -1.403534 & -1.038280 & -1.299481 \\ 0.454389 & -1.915773 & 0.849511 & -3.283238 & 1.363533 \end{bmatrix} \cdot \mathbf{X} + \begin{bmatrix} 45.632534 \\ 35.545749 \\ -1.200182 \\ -0.722646 \\ 1.760645 \end{bmatrix}. \tag{18}$$

3.2. A branch-pathway system

We now consider the branch pathway in Fig. 4, wherein one constant source X_0 and two regulatory signals are included. The production rate of X_1 depends on the concentration of the source variable X_0. The rate is inhibited by X_3, which is generated from X_1 through the intermediate X_2. X_1 generates X_4 which, in turn,

Table 3
The estimated fuzzy rules for Case B_2 (32 rules).

Rule #	Premise condition	Consequence	
1	If ($\boldsymbol{x_1}$ is A_1)	$\dot{x}_1$	$= (-1.805402)\boldsymbol{x_1} + (0.374501)\boldsymbol{x_2} + (-0.571667)\boldsymbol{x_3} + (-2.390966)\boldsymbol{x_4} + (1.597605)\boldsymbol{x_5} + (1.237362)$
	and ($\boldsymbol{x_2}$ is B_1)	$\dot{x}_2$	$= (-6.585688)\boldsymbol{x_1} + (-5.244748)\boldsymbol{x_2} + (-3.067193)\boldsymbol{x_3} + (-6.027011)\boldsymbol{x_4} + (3.123262)\boldsymbol{x_5} + (8.742235)$
	and ($\boldsymbol{x_3}$ is C_1)	$\dot{x}_3$	$= (0.978450)\boldsymbol{x_1} + (3.776151)\boldsymbol{x_2} + (3.343020)\boldsymbol{x_3} + (10.201972)\boldsymbol{x_4} + (5.148152)\boldsymbol{x_5} + (16.074961)$
	and ($\boldsymbol{x_4}$ is D_1)	$\dot{x}_4$	$= (-6.596762)\boldsymbol{x_1} + (-2.548520)\boldsymbol{x_2} + (-2.916527)\boldsymbol{x_3} + (-5.073449)\boldsymbol{x_4} + (-3.200010)\boldsymbol{x_5} + (-1.444041)$
	and ($\boldsymbol{x_5}$ is E_1)	$\dot{x}_5$	$= (-17.650697)\boldsymbol{x_1} + (-6.419498)\boldsymbol{x_2} + (16.186847)\boldsymbol{x_3} + (-23.540106)\boldsymbol{x_4} + (-3.710285)\boldsymbol{x_5} + (35.275563)$
17	If ($\boldsymbol{x_1}$ is A_2)	$\dot{x}_1$	$= (31.792603)\boldsymbol{x_1} + (4.996830)\boldsymbol{x_2} + (6.172297)\boldsymbol{x_3} + (6.623522)\boldsymbol{x_4} + (8.456148)\boldsymbol{x_5} + (45.632534)$
	and ($\boldsymbol{x_2}$ is B_1)	$\dot{x}_2$	$= (23.093542)\boldsymbol{x_1} + (2.808661)\boldsymbol{x_2} + (0.868960)\boldsymbol{x_3} + (0.680079)\boldsymbol{x_4} + (20.396518)\boldsymbol{x_5} + (35.545749)$
	and ($\boldsymbol{x_3}$ is C_1)	$\dot{x}_3$	$= (-0.326283)\boldsymbol{x_1} + (-0.134161)\boldsymbol{x_2} + (-1.665026)\boldsymbol{x_3} + (-1.056774)\boldsymbol{x_4} + (0.995313)\boldsymbol{x_5} + (-1.200182)$
	and ($\boldsymbol{x_4}$ is D_1)	$\dot{x}_4$	$= (-1.917613)\boldsymbol{x_1} + (0.040022)\boldsymbol{x_2} + (-1.403534)\boldsymbol{x_3} + (-1.038280)\boldsymbol{x_4} + (-1.299481)\boldsymbol{x_5} + (-0.722646)$
	and ($\boldsymbol{x_5}$ is E_1)	$\dot{x}_5$	$= (0.454389)\boldsymbol{x_1} + (-1.915773)\boldsymbol{x_2} + (0.849511)\boldsymbol{x_3} + (-3.283238)\boldsymbol{x_4} + (1.363533)\boldsymbol{x_5} + (1.760645)$
32	If ($\boldsymbol{x_1}$ is A_2)	$\dot{x}_1$	$= (-13.394363)\boldsymbol{x_1} + (-15.489171)\boldsymbol{x_2} + (-14.522549)\boldsymbol{x_3} + (-10.179259)\boldsymbol{x_4} + (-3.172033)\boldsymbol{x_5} + (-12.303465)$
	and ($\boldsymbol{x_2}$ is B_2)	$\dot{x}_2$	$= (9.140159)\boldsymbol{x_1} + (18.039619)\boldsymbol{x_2} + (15.730383)\boldsymbol{x_3} + (11.776771)\boldsymbol{x_4} + (-39.428628)\boldsymbol{x_5} + (10.841829)$
	and ($\boldsymbol{x_3}$ is C_2)	$\dot{x}_3$	$= (12.172604)\boldsymbol{x_1} + (-18.911194)\boldsymbol{x_2} + (-4.967920)\boldsymbol{x_3} + (30.727985)\boldsymbol{x_4} + (-9.948022)\boldsymbol{x_5} + (6.052937)$
	and ($\boldsymbol{x_4}$ is D_2)	$\dot{x}_4$	$= (-15.099643)\boldsymbol{x_1} + (27.807653)\boldsymbol{x_2} + (14.595713)\boldsymbol{x_3} + (6.408267)\boldsymbol{x_4} + (-35.690673)\boldsymbol{x_5} + (-5.714309)$
	and ($\boldsymbol{x_5}$ is E_2)	$\dot{x}_5$	$= (7.316944)\boldsymbol{x_1} + (-4.871945)\boldsymbol{x_2} + (-5.849911)\boldsymbol{x_3} + (-14.834883)\boldsymbol{x_4} + (-7.203929)\boldsymbol{x_5} + (37.116727)$

Table 4
Comparison for the branch-pathway system.

Case	MF	No. of input-space partitions	No. of premise-para./rule	Rule No.	Total of premise para.	MSE (training)	MSE (testing)
B_2	Bell	2	3^4	2^4	1296	0.001674	0.642328
G_2	Gauss	2	2^4	2^4	256	0.001378	0.321211
G_3	Gauss	3	2^4	3^4	1296	0.000876	0.483230

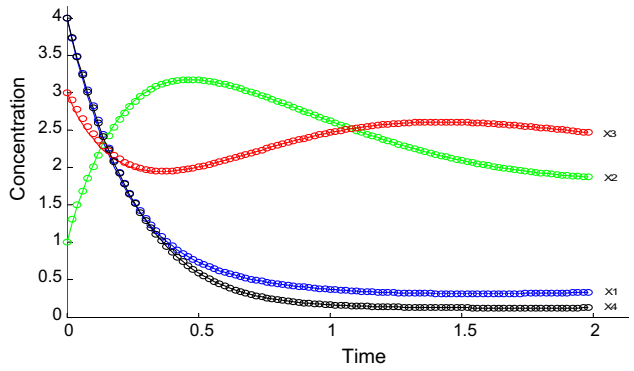

Fig. 5. The dynamic behavior of the branch pathway system for Case G_2 (inputs with two fuzzy partitions and Gaussian MFs). The solid curves are the profiles of the data generated from the S-system in [18]. "o" denotes the estimated data.

Table 5
The term sets and the estimated membership functions for Case G_2.

Term set	Gaussian MFs
A_1	$\mu_{A_1}(x_1) = N(m_{11},\sigma_{11}) = N(0.4484497445, -0.3698028420)$
A_2	$\mu_{A_2}(x_1) = N(m_{21},\sigma_{21}) = N(0.3358241071, 1.1569335970)$
B_1	$\mu_{B_1}(x_2) = N(m_{12},\sigma_{12}) = N(1.3556163212, 0.4388491336)$
B_2	$\mu_{B_2}(x_2) = N(m_{22},\sigma_{22}) = N(1.1968851967, 3.5675579132)$
C_1	$\mu_{C_1}(x_3) = N(m_{13},\sigma_{13}) = N(0.8472455821, 0.3862060982)$
C_2	$\mu_{C_2}(x_3) = N(m_{23},\sigma_{23}) = N(1.0329170905, 2.6702505223)$
D_1	$\mu_{D_1}(x_4) = N(m_{14},\sigma_{14}) = N(0.7192191057, -0.9281075647)$
D_2	$\mu_{D_2}(x_4) = N(m_{24},\sigma_{24}) = N(1.3702190503, 2.4501362904)$

inhibits the degradation of X_3. We shall further discuss the performance of Cases B_2, G_2 and G_3 in this system.

Table 4 shows the number of input partition, premise parameters and fuzzy rules. The total number of premise parameters to be identified is $3^4 \times 2^4$ for Cases B_2 and G_3, and $2^4 \times 2^4$ for Case G_2. In the training phase the mean-square error $E(G\_3) < E(G\_2) < E(B\_2)$, but $E(G\_2) < E(G\_3) < E(B\_2)$ in the testing phase. In this system Case G_3 performs always better than Case B_2, and Case G_2 performs better than Case B_2. These results are different from those in last subsection. We also observe that even using only 256 premise parameters, the performance of Case G_2 is still good. Fig. 5 shows the dynamic behavior of Case G_2, where the solid curves are the profiles of the data generated from the S-system in [19]. The circle points are the estimated data. Table 5 shows the estimated membership functions. *Due to space limitation* Table 6 only shows three of the sixteen fuzzy rules. Fig. 5 only shows the simulation results of Case G-2. The complete results of the rules and the simulation results of Cases B_2 and G_3 are shown in supplement file.

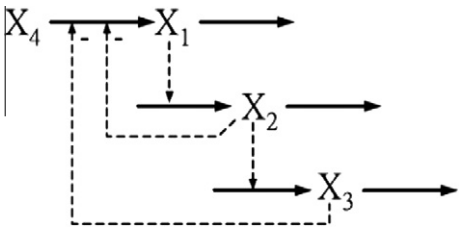

Fig. 6. The cascade network [20].

3.3. A cascade-network system

We further consider a small system in Fig. 6, which is a cascade network with three steps and two feedback signals. The dependent variable X_1 is generated from the precursors X_4, which is an independent variable. X_1 inhibits the production of X_2, which, in turn, inhibits the production of X_3. Both X_2 and X_3 inhibit the production of X_1.

From the results in the last subsection, we observe that the performance of Case G_2 is the best in the testing phase. In this case only 256 premise parameters to be identified. To further realize the performance of these two kinds of membership functions, we now consider the cases with the premise-parameter numbers higher than 200 but lower than 1000; i.e., Cases B_2, G_3, B_3 and G_4, where Cases B_2 and G_3 have the same number of the premise parameters ($\emptyset = 216$).

The mean-square error $E(G\_3) < E(G\_4) < E(B\_3) < E(B\_2)$ in the training phase and $E(B\_3) < E(G\_4) < E(G\_3) < E(B\_2)$ in the testing phase, as shown in Table 7. In both training and testing phases, Case G_3 performs better than Case B_2, and Case B_3 better than Case B_2. Fig. 7 shows the dynamic behavior of Case B_3, where the solid curves are the profiles of the data generated from the S-system in [20]. The circle points are the estimated data. Table 8 shows the estimated membership functions. *Due to space limitation* Table 9 only shows three of the estimated twenty-seven fuzzy rules. The complete results of the rules and the simulation results of Cases B_2, G_3 and G_4 are shown in supplement file.

3.4. Robustness and discussion

To realize the influence of the membership functions and the premise-parameter numbers to the accuracy, we first consider a five-state (the small-gene network), then a four-state (the branch pathway) and finally a three-state (the cascade pathway) systems.

Table 6
The estimated fuzzy rules for Case G_2 (16 rules).

Rules #	Premise condition	Consequence	
1	If ($\boldsymbol{x_1}$ is A_1)	$\dot{x}_1$	$= (0.872838)\boldsymbol{x_1} + (1.370118)\boldsymbol{x_2} + (2.463601)\boldsymbol{x_3} + (-4.308165)\boldsymbol{x_4} + (0.813599)$
	and ($\boldsymbol{x_2}$ is B_1)	$\dot{x}_2$	$= (7.058914)\boldsymbol{x_1} + (-1.040160)\boldsymbol{x_2} + (2.732890)\boldsymbol{x_3} + (3.790796)\boldsymbol{x_4} + (-3.882646)$
	and ($\boldsymbol{x_3}$ is C_1)	$\dot{x}_3$	$= (-1.593277)\boldsymbol{x_1} + (-0.497749)\boldsymbol{x_2} + (-2.045406)\boldsymbol{x_3} + (1.450548)\boldsymbol{x_4} + (4.787881)$
	and ($\boldsymbol{x_4}$ is D_1)	$\dot{x}_4$	$= (-12.142588)\boldsymbol{x_1} + (-0.035636)\boldsymbol{x_2} + (1.851351)\boldsymbol{x_3} + (-0.858458)\boldsymbol{x_4} + (1.450552)$
11	If ($\boldsymbol{x_1}$ is A_2)	$\dot{x}_1$	$= (0.037176)\boldsymbol{x_1} + (2.080029)\boldsymbol{x_2} + (-1.772855)\boldsymbol{x_3} + (0.119057)\boldsymbol{x_4} + (-0.793813)$
	and ($\boldsymbol{x_2}$ is B_1)	$\dot{x}_2$	$= (0.423635)\boldsymbol{x_1} + (1.783810)\boldsymbol{x_2} + (-2.130905)\boldsymbol{x_3} + (-0.077072)\boldsymbol{x_4} + (-1.377067)$
	and ($\boldsymbol{x_3}$ is C_2)	$\dot{x}_3$	$= (-0.150248)\boldsymbol{x_1} + (2.207348)\boldsymbol{x_2} + (-0.023670)\boldsymbol{x_3} + (3.259519)\boldsymbol{x_4} + (-0.417050)$
	and ($\boldsymbol{x_4}$ is D_1)	$\dot{x}_4$	$= (1.898811)\boldsymbol{x_1} + (2.039294)\boldsymbol{x_2} + (-1.889381)\boldsymbol{x_3} + (-0.472064)\boldsymbol{x_4} + (-4.762252)$
16	If ($\boldsymbol{x_1}$ is A_2)	$\dot{x}_1$	$= (0.796587)\boldsymbol{x_1} + (0.197774)\boldsymbol{x_2} + (-3.522612)\boldsymbol{x_3} + (-7.957953)\boldsymbol{x_4} + (4.309952)$
	and ($\boldsymbol{x_2}$ is B_2)	$\dot{x}_2$	$= (0.073142)\boldsymbol{x_1} + (-0.198299)\boldsymbol{x_2} + (1.059254)\boldsymbol{x_3} + (-5.341454)\boldsymbol{x_4} + (1.519781)$
	and ($\boldsymbol{x_3}$ is C_2)	$\dot{x}_3$	$= (2.458432)\boldsymbol{x_1} + (0.882399)\boldsymbol{x_2} + (-1.382544)\boldsymbol{x_3} + (-9.020635)\boldsymbol{x_4} + (-0.002416)$
	and ($\boldsymbol{x_4}$ is D_2)	$\dot{x}_4$	$= (-0.138391)\boldsymbol{x_1} + (-0.779898)\boldsymbol{x_2} + (0.582258)\boldsymbol{x_3} + (-3.785568)\boldsymbol{x_4} + (2.429161)$

Table 7
Comparison for the cascade-network system.

Case	MF	No. of input-space partitions	No. of premise para./rule	Rule No.	Total of premise para.	MSE (training)	MSE (testing)
B_2	Bell-shaped	2	3^3	2^3	216	0.127334	0.094908
B_3	Bell-shaped	3	3^3	3^3	729	0.000607	0.030100
G_3	Gaussian	3	2^3	3^3	216	0.000467	0.045322
G_4	Gaussian	4	2^3	4^3	512	0.000587	0.032074

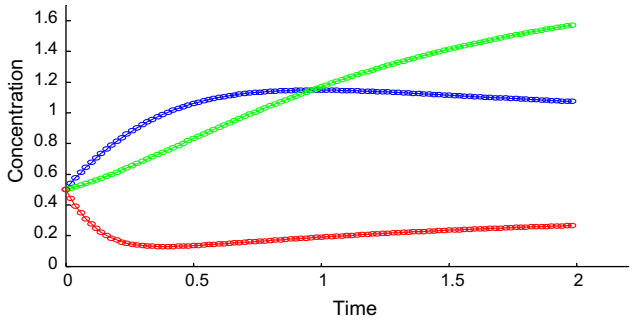

Fig. 7. The dynamic behavior of the cascade-network system for Case B_3 (inputs with three fuzzy partitions and Gaussian MFs). The solid curves are the profiles of the data generated from S-system in [18]. "o" denotes the estimated data.

Table 8
The term sets and the estimated membership functions for Case B_3.

Term set	Bell-shaped MFs
A_1	$\mu_{A_1}(x_1) = \text{bell}(a_{11}, b_{11}, c_{11})$ = bell(0.6496206720, 2.0631628436, 0.0328476062)
A_2	$\mu_{A_2}(x_1) = \text{bell}(a_{21}, b_{21}, c_{21})$ = bell (0.7636297791, 2.0618033038, 1.8481211554)
A_3	$\mu_{A_3}(x_1) = \text{bell}(a_{31}, b_{31}, c_{31})$ = bell (0.8757370711, 2.0027074037, 3.6194493740)
B_1	$\mu_{B_1}(x_2) = \text{bell}(a_{12}, b_{12}, c_{12})$ = bell (1.2405200557, 1.9630070208, 0.0821521074)
B_2	$\mu_{B_2}(x_2) = \text{bell}(a_{22}, b_{22}, c_{22})$ = bell (1.4040693335, 2.0220579489, 2.6854316838)
B_3	$\mu_{B_3}(x_2) = \text{bell}(a_{32}, b_{32}, c_{32})$ = bell (1.3399644665, 1.9838826452, 5.3884285322)
C_1	$\mu_{C_1}(x_3) = \text{bell}(a_{13}, b_{13}, c_{13})$ = bell (1.0132736653, 1.7070975995, −0.1696217798)
C_2	$\mu_{C_2}(x_3) = \text{bell}(a_{23}, b_{23}, c_{23})$ = bell (1.0318122863, 2.2118054698, 1.3334398913)
C_3	$\mu_{C_3}(x_3) = \text{bell}(a_{33}, b_{33}, c_{33})$ = bel1(0.8725660382, 2.0778530171, 2.7477785579)

The used premise-parameter numbers are, respectively, (1024, 7776, 7776, 59,049) for cases (G_2, G_3, B_2, B_3) in the five-state system, (256, 1296, 1296) for cases (G_2, G_3, B_2) in the four-state system, and (216, 216, 512, 729) for cases (G_3, B_2, G_4, B_3) in the three-state system. Cases G-3 and B-2 have the same number of the premise parameters. From the simulation results we observe three phenomena: (1) Except the five-state system, the performance of Case G_3 is always better than that of Case B_2 in both training and testing phases. In the five-state system the number of fuzzy rules and consequence parameters for Case G_3 is up to 243 and 7290, respectively. Therefore, the influence of consequence-parameter numbers to the accuracy should be considered for a fair comparison. In other words, the comparison of Cases B_2 and G_3 of Table 1 in this five-state system is not suitable. (2) Case B_2 performs better than G_2 in the five-state system, but worse in the four-state system. Therefore, for the same number of input-space partitions the Bell-type MFs, even with more parameters, does not always perform better than the Gaussian MFs. (3) Increasing the number of input-space partitions cannot improve the performance. In the four-state system $E(G\_3)$ is not always smaller than $E(G\_2)$. In the three-state system $E(G\_4)$ is not smaller than $E(G\_3)$. Based on these three phenomena and considering various limitations (computation time, rule numbers... etc.), Case G-3 is a good choice for neural-fuzzy modeling of biological systems.

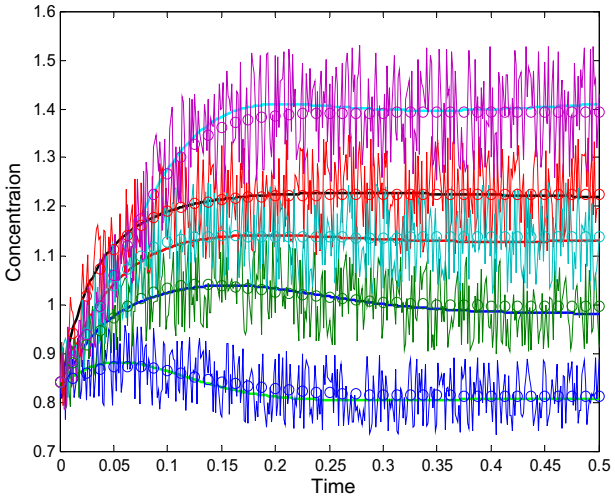

Fig. 8. Robustness examination for the identification of the small-scale genetic network (Case G_3). The dot points are the data subject to 10% random noise, the **solid curves** are the estimated profiles, and the **circle points** are the noise-free data.

We further discuss the scalability of the proposed method (the ability of ANTM when applied to a large situation). If there are a large number of data sets, then we can adopt pattern update to release the burden of the computation. The CPU time depends on the dimension of the system and the partition of the input space: The larger the dimension or the finer the partition, the more the CPU time is. When the partition is two, or the system dimension is two (except Case G-4), CPU time is less than thirty minutes. For other situations CPU time is around several hours. For an n-dimensional system with m output variables, the number of input variables is n. If we divided the input space into three, then the number of fuzzy rules to be identified is 3^n. In other words, the number of premise parameters to be identified is $3^n \times 2^n$, and that of the consequence parameters is $n(n+1)$. The latter issue is easy to be solved. As for the former issue, we should develop self-organization techniques to reduce the number of fuzzy rules. We also simulate for the tendency of the small genetic system to be subject to 10% external noise, the results of which are shown in Fig. 8 for Cases G_3 and in supplement file for Case B_3. We observe that ATNM is robust to the external noise, even testing at 20% beyond a training range.

4. Conclusion

In this study we present an adaptive neural-fuzzy modeling technique to identify biological systems. The proposed approach

Table 9
The estimated fuzzy rules for Case B_3 (27 rules).

Rule #	Premise condition	Consequence	
1	If ($\boldsymbol{x_1}$ is A_1) and ($\boldsymbol{x_2}$ is B_1) and ($\boldsymbol{x_3}$ is C_1)	$\dot{x}_1$	$= (3.675062)x_1 + (-1.876954)x_2 + (5.122481)x_3 + (0.543429)$
		$\dot{x}_2$	$= (-6.953465)x_1 + (-1.118006)x_2 + (0.204854)x_3 + (4.621695)$
		$\dot{x}_3$	$= (-3.904580)x_1 + (0.097342)x_2 + (2.374119)x_3 + (-0.738029)$
19	If ($\boldsymbol{x_1}$ is A_3) and ($\boldsymbol{x_2}$ is B_1) and ($\boldsymbol{x_3}$ is C_1)	$\dot{x}_1$	$= (5.426150)x_1 + (1.522649)x_2 + (-4.016111)x_3 + (-4.103842)$
		$\dot{x}_2$	$= (-1.844097)x_1 + (1.907740)x_2 + (-3.749317)x_3 + (-1.271477)$
		$\dot{x}_3$	$= (-1.875846)x_1 + (3.604842)x_2 + (-3.167104)x_3 + (-2.983358)$
27	If ($\boldsymbol{x_1}$ is A_3) and ($\boldsymbol{x_2}$ is B_3) and ($\boldsymbol{x_3}$ is C_3)	$\dot{x}_1$	$= (-0.040836)x_1 + (-0.047499)x_2 + (-0.014949)x_3 + (-0.038240)$
		$\dot{x}_2$	$= (0.004175)x_1 + (0.031465)x_2 + (0.035390)x_3 + (0.007242)$
		$\dot{x}_3$	$= (-0.000150)x_1 + (0.007043)x_2 + (-0.007488)x_3 + (0.001523)$

maps a biological network onto a fuzzy system in which the relation of variables is included in the fuzzy rules. As the time-series data are fed into the net, the parameter values of the fuzzy rules are updated through the hybrid learning algorithm. Our approach is tested with three biological systems. The simulation results show the inferred fuzzy models effectively simulate the dynamic behavior of these systems. However, when dealing with a large system too many fuzzy rules are needed such that the implementation is limited. In the future we shall develop self-organizing techniques to reduce the number of fuzzy rules while keeping the high accuracy. Self-organizing approaches can achieve not only the parameter identification but also the structure identification of a T–S fuzzy system. Even if the proposed technique is robust to external noise it is still not able to analyze microarray data because the noise contamination in these data sets may be as high as fifty percent. In the future we shall develop new fuzzy estimators (fuzzy filters) for data that are seriously contaminated by white and colored noise, and for systems with bias and uncertainty.

Acknowledgment

This research was supported by grant number NSC-101-2221-E-212-011 from the National Science Council, Taiwan, ROC.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.mbs.2013.01.004.

References

[1] I.C. Chou, E.O. Voit, Estimation of dynamic flux profiles from metabolic time series data, BMC Syst. Biol., 2012.
[2] H. Wang, L. Qian, E. Dougherty, Inference of gene regulatory networks using S-system: a unified approach, IET Syst. Biol. 4 (2) (2010) 145.
[3] S. Marino, E.O. Voit, An automated procedure for the extraction of metabolic network information from time series data, Bioinform. Comput. Biol. 4 (665) (2006).
[4] I.C. Chou, H. Martens, E.O. Voit, Parameter estimation in biochemical systems models with alternating regression, Theor. Biol. Med. Model 3 (25) (2006).
[5] D.Y. Cho, K.H. Cho, B.T. Zhang, Identification of biochemical networks by S-tree based genetic programming, Bioinformatics 22 (2006) 1631.
[6] P.K. Liu, F.S. Wang, Hybrid differential evolution with geometric mean mutation in parameter estimation of bioreaction systems with large parameter search space, Comput. Chem. Eng. 33 (2009) 1851.
[7] F.S. Wang, P.K. Liu, Inverse problems of biochemical systems using hybrid differential evolution and data collocation, Int. J. Syst. Synthetic Biol. 1 (2010) 21.
[8] C.M. Chen, C. Lee, C.L. Chuang, C.C. Wang, G.S. Shieh, Inferring genetic interactions via a nonlinear model and an optimization algorithm, BMC Syst. Biol. 4 (16) (2010).
[9] R. Xu, D.C. Wunsch II, R.L. Frank, Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization, IEEE Trans. Comput. Biol. Bioinform. 4 (2007) 1545.
[10] C.L. Ko, E.O. Voit, F.S. Wang, Estimating parameters for generalized mass action models with connectivity information, BMC Bioinform. 7 (230) (2009).
[11] P.K. Liu, F.S. Wang, Inference of biochemical network models in S-system using multiobjective optimization approach, Bioinformatics 24 (2008) 1085.
[12] S. Kimura, K. Ide, A. Kashihara, M. Kano, H. Mariko, R. Masui, et al., Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm, Bioinformatics 21 (2005) 1154.
[13] I.C. Chou, E.O. Voit, Recent developments in parameter estimation and structure identification of biochemical and genomic systems, Math. Biosci. 219 (2009) 57.
[14] J. Sun, J.M. Garibaldi, C. Hodgman, Parameter estimation using metaheuristics in Systems biology: A comprehensive review, IEEE/ACM Trans. Comput. Biol. Bioinform. 9 (1) (2012) 185.
[15] S.J. Wu, H.H. Chiang, H.T. Lin, T.T. Lee, Neural-network-based optimal fuzzy controller design for nonlinear systems, Fuzzy Sets Syst. 154 (2005) 182.
[16] S.J. Wu, C.T. Wu, Yeh-Chen Chang, Neural-fuzzy gap control for a current/voltage-controlled 1/4-vehicle magLev system, IEEE Trans. Intell. Transp. Syst. 9 (1) (2008) 122–136.
[17] C.T. Lin, C.S.G. Lee, Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems, Prentice-Hall International, Inc., 1996.
[18] W.S. Hlavacek, M.A. Savageau, Rules for coupled expression of regulator and effector genes in inducible circuits, J. Mol. Biol. 255 (1996) 121.
[19] E.O. Voit, J. Almeida, Decoupling dynamical systems for pathway identification from metabolic profiles, Bioinformatics 20 (2004) 1670.
[20] K.Y. Tsai, F.S. Wang, Evolutionary optimization with data collocation for reverse engineering of biological networks, Bioinformatics 21 (2005) 1180.