한국퍼지 및 지능 시스템학회 논문지 1998, Vol. 8, No. 6. (1998) 58~69

A Neuro-Fuzzy Approach to Integration and Control of Industrial Processes: Part I

Sungshin Kim

Department of Electrical Engineering, Pusan National University

ABSTRACT

This paper introduces a novel neuro-fuzzy system based on the polynomial fuzzy neural network (PFNN) architecture. The PFNN consists of a set of if-then rules with appropriate membership functions whose parameters are optimized via a hybrid genetic algorithm. A polynomial neural network is employed in the defuzzification scheme to improve output performance and to select appropriate rules. A performance criterion for model selection, based on the Group Method of Data Handling, is defined to overcome the overfitting problem in the modeling procedure. The hybrid genetic optimization method, which combines a genetic algorithm and the Simplex method, is developed to increase performance even if the length of a chromosome is reduced. A novel coding scheme is presented to describe fuzzy systems for a dynamic search range in the GA. For a performance assessment of the PFNN inference system, three well-known problems are used for comparison with other methods. The results of these comparisons show that the PFNN inference system outperforms the other methods while it exhibits exceptional robustness characteristics.

1. Introduction

An industrial process is generally composed of several constituent subprocesses that produce an output. The three primary difficulties in industrial processes are: integrating all of the constituent subprocesses, determining the optimal operating conditions in each subprocess that maximize productivity, and controlling the system. A model of a complex process may be required for that purpose. For example, it could be used to simulate the real process or it could be used to design a control system. However, if the process is too complex to be described by a few idealized physical laws, or there is not enough knowledge available, then an accurate model may not be possible to develop. Indeed, considerable experimental and theoretical efforts are often required to derive even the most basic representation. This is especially the case when the process is highly nonlinear and has a large number of variables.

Fuzzy systems and neural networks (NNs) have been investigated along different lines since they are derived from different fields. Both can best be described mathematically as function approximators [13,20]. A neuro-fuzzy system can be viewed as a feed-forward network that employs a back- propagation algorithm for training purposes. It is used to identify nonlinear dynamic systems. The method in [33] employs a back-propagation algorithm in order to adjust its parameters. The inference system in [11] is based on the Takagi, Sugeno, and Kang fuzzy model as described in [26,28], also known as the TSK fuzzy model. The method in [16] is similar to radial basis function (RBF) neural networks[3]. The approach in [15] is based on the TSK fuzzy model and adopts a genetic algorithm (GA) to determine the premise and consequent parameters as well as the number of fuzzy rules.

In this paper, a polynomial fuzzy neural network (PFNN) modeling method is proposed to improve the performance of the fuzzy inference system. A performance criterion is defined to overcome the overfitting problem in the modeling procedure. The hybrid genetic optimization method is developed to increase performance even if the length of a chromosome is reduced. A novel coding scheme is presented to describe fuzzy systems for a flexible search range in the GA. For a performance assessment of the PFNN inference system, three well-known

*This paper was supported by the Pusan National University Research Fund, 1998.

problems are used for comparison with other methods.

This paper is organized as follows: Section 2 introduces the PFNN based upon the performance criterion for model selection. Section 3 describes the hybrid genetic optimization method combined with the dynamic coding in a genetic algorithm. Section 4 presents the performance test and evaluation for the proposed modeling approach. Finally, conclusions are summarized in Section 5.

2. Neuro-Fuzzy Modeling

Fuzzy model identification that is based on fuzzy implications and reasoning is one of the most important aspects of fuzzy system theory because of its simple form as a tool and its power for representing highly nonlinear relations. Using qualitative expressions, graded numbering, and experimental data, the performance index or yield can be incorporated into a fuzzy model, a fuzzy objective function, or a decision making model[12,30,31].

In this section first, a neuro-fuzzy inference system based on the PFNN architecture is presented. Then a performance criterion for model selection is introduced to prevent the overfitting problem.

2.1 Polynomial Fuzzy Neural Network

The proposed neuro-fuzzy inference system is based on the PFNN architecture. The PFNN consists of a set of *if-then* rules with appropriate membership functions whose parameters are optimized via a hybrid genetic algorithm. A polynomial neural network [5,25] is employed in the defuzzification scheme to improve output performance and to select rules.

Fig. 1 depicts the basic structure of the polynomial fuzzy neural network. A basic assumption is that the neuro-fuzzy system under consideration has n inputs, x_i, i=1, ⋯, n, and one output $\hat{y}$. The same procedure described for single-output systems can be applied to multi-output systems. Each input has m_i membership functions. The total number of rules, r, is equal to $r=\Pi_{i=1}^{n} m_i$. A typical rule with a fuzzy if-then structure is expressed by

$$Rule_{(i)}: \text{ If } x_1 \text{ is } A_1^{i_1} \text{ and } \cdots \text{ andd } x_n \text{ is } A_n^{i_n}, \text{ then } y_i \text{ is } p_i \quad (1)$$

where x_j, j=1, ⋯, n, are non-fuzzy input variables, $A_k^{i_k}$, $1<i_k<m_k$, are fuzzy variables related to the k-th input, and p_i, i=1, ⋯, r, are constants in the consequent.

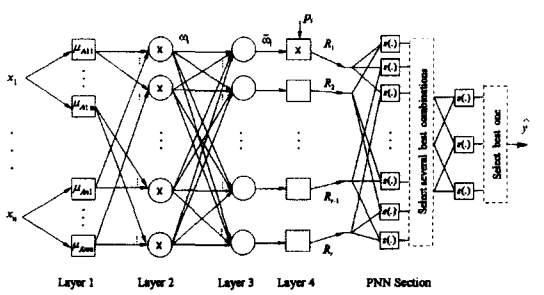


Fig. 1. Polynomial fuzzy neural network (PFNN)

Layer 1: The nodes in this layer are adaptive nodes with node outputs defined by

$$O_{1,p} = \mu_{A_i^j}(x_i), \; i=1, \cdots, n, \; j=1, \cdots, m_i \quad (2)$$

where $O_{1,p}$ represents the output of node p in layer 1, x_i is the non-fuzzy input to the node, and A_i^j is associated with the j-th fuzzy variable in the i-th input. The number of nodes is $\sum_{i=1}^{n} m_i$. The outputs of this layer, $O_{1,p}$, are the membership values of the premise part. Here, the membership functions for A_i^j, $\mu_{A_i^j}(x_i)$, can be any appropriately parameterized membership functions such as triangular, trapezoidal, Gaussian, etc. For example, A_i^j can be characterized by a Gaussian membership function:

$$\mu_{A_i^j}(x_i; c_j, \sigma_j) = \exp\left\{\frac{-(x_i - c_j)^2}{2\sigma_j^2}\right\} \quad (3)$$

Layer 2: The nodes in layers 2 and 3 are computational nodes that do not include adjustable parameters. The nodes in layer 2 are T-norm operators for the fuzzy AND. T-norm product operators multiply the incoming signals and send the result to the next layer. The operation in this layer is as follows:

$$O_{2,i} = w_i = \prod_{k=1}^{n} \mu_{A_k^j}(x_k) \quad (4)$$

where r is the number of rules and $1 \le j \le m_k$, $1 \le i \le r$. The output in layer 2, $O_{2,i}$, represents the firing strength or degree of fulfillment of a rule.

Layer 3: The i-th node in this layer calculates a normalized firing strength, which is the ratio of the i-th rule's firing strength to the sum of all rules' firing strengths:

$$O_{3,i} = \overline{w}_i = \frac{w_i}{\sum_{j=1}^{r} w_j} \tag{5}$$

Layer 4: The i-th node in this layer is an adaptive node with parameters p_i, i=1, ···, n. The outputs in this layer are calculated by

$$O_{4,i} = R_i = \overline{w}_i \cdot y_i = \overline{w}_i \cdot p_i \tag{6}$$

The outputs in this layer are used as inputs to the polynomial neural network (PNN). In fuzzy systems, the next step following layer 4 is defuzzification, where the overall output is computed as the sum of the outputs from layer 4; i.e., the final output of the fuzzy system is $\sum_{i=1}^{r} O_{4,i}$. In this case, the output of each rule is considered as an equally important part in the defuzzification. However, in the PFNN model, the outputs in layer 4, R_i, are not directly used to produce the overall output. These are considered as inputs to the PNN.

The parameters in the adaptive nodes in layer 1 and layer 4 will be optimized by the method discussed in Section 3.

A PNN is a feed-forward network that computes a polynomial function of a set of parallel inputs to generate an output. Ivakhnenko[9] describes an approximation that is determined by successively fitting polynomials and uses the mean squared error at each layer to determine the best fit at that layer. The right side of Fig. 1 illustrates how a PNN is developed using the basic group method of data handling (GMDH)[5].

A least-squares fit of the training data for the function to be learned is determined for all pairs of input variables, R_is, using polynomials of up to second degree. A decision rule is used after the first level of computations to choose only those polynomials that provide the best fit to the training data. The outputs of this reduced set of polynomials are then used as inputs to a second layer of the model. Here again, all pairs of polynomials up to second degree are tested, and only those that provide the best fit are kept as inputs to the next layer. This procedure is continued until a stopping criterion is satisfied, e.g., the present layer of polynomial outputs provides a worse fit to the training data than the previous one. At this point, for n layers, the output polynomial contains terms of up to degree 2^n in the input variables, but the polynomial network's typical combinatorial explosion of coefficients has been avoided.

The polynomial network shown in Fig. 1 has r inputs and two layers, and each two-input module forms a quadratic polynomial. For example, the output of the bottom left module is a function of the inputs R_{r-1} and R_r:

$$s(R_{r-1}, R_r; a_i) = a_0 + a_1 R_{r-1} + a_2 R_r + a_3 R_{r-1} R_r + a_4 R_{r-1}^2 + a_5 R_r^2 \tag{7}$$

The decision rule module chooses the three best fitting polynomials from the first layer and sends them on to the second layer, where another set of quadratic least-squares fitting operations is performed. Choosing the best fitting polynomial from the second layer gives rise to the final output $\hat{y}$ of degree four. The proposed PFNN algorithm may provide the following benefits:

(1) It reduces the number of adjustable parameters in the consequent to r that is the number of rules;

(2) It employs the output of layer 4 for the unbiased input candidate;

(3) It increases the accuracy through a new defuzzification method. The output is more accurate than the sum of the R_i since the PNN gives a better approximation than the sum of its inputs;

(4) It uses the dominant rules to infer the outputs. The PNN selects several R_is from among all of the R_is, thereby reducing the number of rules.

Example 1 – The simulation in [33] and [11] is repeated here. The plant to be identified is governed by the difference equation

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + g(u(k)) \tag{8}$$

where the unknown function has the form $g(u)$= $0.6\sin(\pi u)+0.3\sin(3\pi u)+0.1\sin(5\pi u)$. To identify the plant, a series-parallel model described by the difference equation

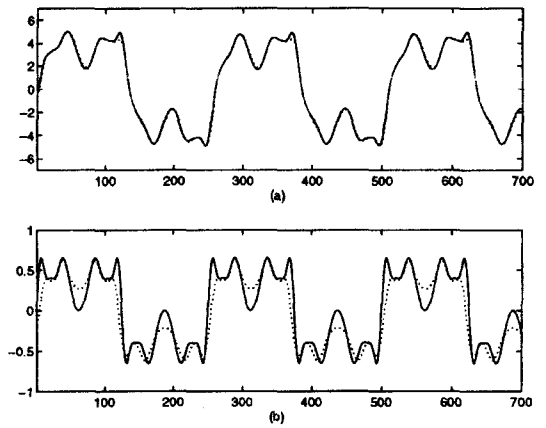


Fig. 2. Back-propagation fuzzy system in Example 1: (a) Outputs of the plant (solid line) nd the identification model (dotted line). (b) The unknown function $g(\cdot)$ (solid line) and the estimated function $f(\cdot)$ (dotted line).

$$\hat{y}(k+1) = 0.3y(k)+0.6y(k-1)+f(u(k)) \quad (9)$$

is used, where $f(\cdot)$ is the function implemented by the PFNN. The identified output, $\hat{y}(\cdot)$, and $f(\cdot)$ are compared with example 1 of [33] that adopts a back-propagation fuzzy system (BPFS) identifier.

Figs. 2(a)~(b) show the output of the identified model in BPFS with mean squared error (MSE)= 0.2148. Figs. 3(a)~(b) show the output of the identified model in the PFNN scheme with MSE= 0.0361. Fig. 4 shows the output of layer 4, R_i, in (6). In this example, the number of rules is 20, but only

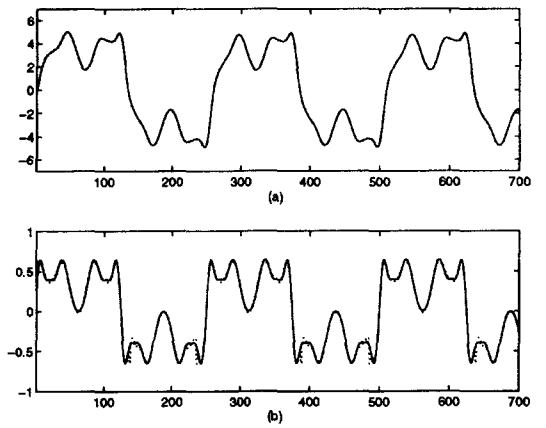


Fig. 3. PFNN in Example 1: (a) Outputs of the plant (solid line) and the identification model (dotted line). (b) The unknown function $g(\cdot)$ (solid line) and the estimated function $f(\cdot)$ (dotted line).

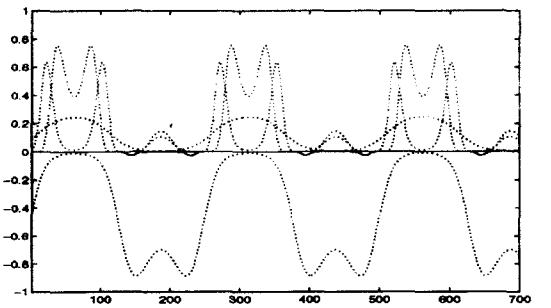


Fig. 4. Plot of R_i, $(1 \le i \le 10)$, in Example 1. The values of R_i at i=1, 2, 6 and 8, are almost zeros.

R_1 to R_{10} are displayed. It shows that the several rules are not fired even though their parameters are optimized. In the PFNN, these rules are filtered, and the output is improved by the PNN.

2.2 Performance Criterion

One of the most common and difficult problems in the empirical modeling arena is the question of when to stop searching the free parameters or adding terms to a model. The fitting error on the finite data set is decreased by simply increasing the number of free parameters in the model or the model order.

The goal of modeling is to obtain a predictive model that generalizes across many such samples to the universe at large, and not merely to the sample at hand. We want our model to be as effective as possible when we use it on future data, and not just on the sample on which it was based.

The proposed performance criterion (*PC*) for a model selection is based on the GMDH to minimize the error and at the same time to prevent overfitting of the empirical data set. As a proper criterion for the verification of a fuzzy model, the observed data are divided into two sets, N_A for training and N_B for testing purposes.

The performance criterion is calculated as

$$e_1^2 = \sum_{i=1}^{n_A} (y_i^A - f_A(x_i^A))^2 / n_A ,$$

$$e_2^2 = \sum_{i=1}^{n_B} (y_i^B - f_A(x_i^B))^2 / n_B ,$$

$$PC = e_1^2 + e_2^2 + \eta(e_1^2 - e_2^2)^2 \quad (10)$$

where n_A is the number of data points in the data set

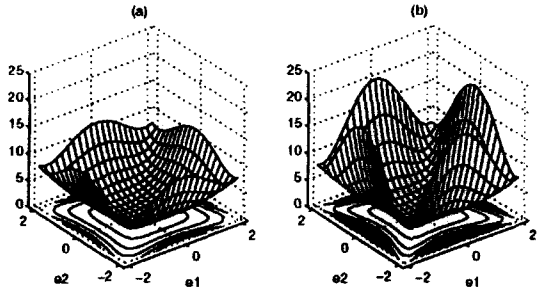


Fig. 5. Surface of the performance criterion: (a) η=0.5 and (b) η=1.0.

N_A, and y_i^A is the real output of data set N_A. $f_A(x_i^B)$ is the estimated output for the data set N_B from the model identified using the data set N_A. η is a weight factor on the difference between e_1^2 and e_2^2. We want to minimize the PC to find the best model for N_A and N_B. The partial derivatives of PC with respect to e_1 and e_2 are as follows:

$$\frac{\partial PC}{\partial e_1} = 4\eta e_1^3 + 2(1-2\eta e_2^2)e_1 \tag{11}$$

$$\frac{\partial PC}{\partial e_2} = 4\eta e_2^3 + 2(1-2\eta e_1^2)e_2 \tag{12}$$

Eq. (11) shows that the derivative of PC with respect to e_1 is a function of e_1 and e_1. Also, Eqs. (11) and (12) are symmetric. The final model based on the PC is not overfitting the training data set but satisfying both data sets. Figs. 5(a) and (b) are surfaces of the performance criterion in (10) with η= 0.5 and η=1, respectively.

3. Fuzzy Optimization

Fuzzy optimization is defined as the process of finding the conditions that give the maximum or minimum value of a cost function that consists of a rule-based fuzzy model with its membership functions as constraints. Fuzzy optimization is performed based on a fuzzy model instead of a mathematical model. A fuzzy objective function is not differentiable by its own nature.

3.1 Hybrid Genetic Optimization

The advantages of a genetic algorithm are: it searches from a set of designs and not from a single design; it is not derivative-based; it works with discrete and continuous parameters; and it both explores and exploits the parameter space[6]. The major disadvantage of a genetic algorithm is the excessive number of runs of the design code required for convergence.

The search space in a GA is discretized by its resolution. In the binary coding method, the bit length L_i and the corresponding resolution R_i are related by

$$R_i = \frac{UB_i - LB_i}{2^{L_i} - 1} \tag{13}$$

where UB_i and LB_i are the upper and lower bounds of the parameter x_i, respectively. As a result, the parameter set can be transformed into a binary string (chromosome) with length $l=\sum_{i=1}^{n} L_i$, where n is the number of parameters to be optimized and L_i is different for each parameter. By adding k bits to the parameters, the resolution is improved by approximately $2^k \times 100\%$. On the other hand, the search space is dramatically increased to $(2^k)^n$ times. To overcome this problem, a hybrid genetic optimization method that combines a GA with Nelder and Mead's simplex method[22] is introduced. First, a GA is used to search the coarse search space predefined by (13) and to find the basin of attraction of the global solution. Second, the result of the GA is passed to a simplex method as an initial condition. The simplex method is a fast and effective search methodology for finding local minima. By defining the resolution in each variable, the length of bits is calculated from (13):

$$L_i = int\ [\log_2(\frac{UB_i - LB_i}{R_i} + 1)] \tag{14}$$

Example 2 – Find the maximum values of the function: $f(x)=1+\cos(x)/(1+0.01x^2)$. Notice that the optimal value of this function occurs at x=0, but notice also that there are many local maxima which are suboptimal. Represent a chromosome as a seven bit binary string and scale the values between -20 and 20. The resolution is 0.31496. Therefore, the nearest points to zero are $\pm$0.15748 with $f(\cdot)$= 1.98738, and these points are the best solution for

the given resolution found by the GA. After employing the simplex method, the maximum value is found at x=0, and $f(\cdot)$=2.

3.2 Coding in Genetic Algorithms

The chromosome representation of potential problem solutions is a crucial aspect in determining the success or failure of a GA. In a fuzzy system, the coding of a fuzzy rule base implies coding the membership functions. The main questions are: How is the chromosome to be structured? How does this structure encode string vectors?

A rule consists of fuzzy variables in which parameters need to be optimized. For example, a Gaussian membership function has two parameters, c_j and σ_j in (3). In a n-input, one-output system, its genotype, that is the complete sets of chromosomes, is illustrated in Fig. 6. An n-bit binary number v can represent a number w that falls in the range [LB, UB] using the mapping

$$w = LB + (UB - LB)\frac{v}{2^n - 1} \tag{15}$$

The range of p_i in (1) can be replaced with the output range of the system to be modeled. From (3), the upper bound of the variance, σ_i^{UB}, in the MF can be confined within a predefined range by

$$\sigma_i^{UB} = \frac{|x_o - c_i|}{\sqrt{2 \cdot |\ln(\varepsilon)|}} \tag{16}$$

where x_0, which is the almost zero point in the MF, satisfies

$$\mu_{A_i}(x_0; c_i, \sigma_i^{UB}) = \varepsilon. \tag{17}$$

For example, for the degree of a membership function to equal 10^{-2} at a distance d from its center, the upper bound of the variance is determined by $d/(2\sqrt{\ln(10)})$ from (16).

The range of p_i and σ_i will be determined from the

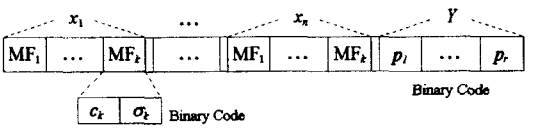


Fig. 6. Coding scheme for a fuzzy system.

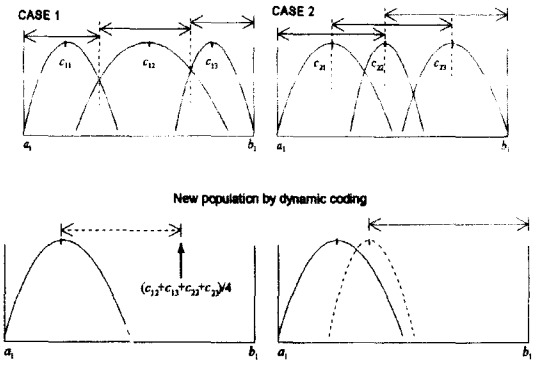


Fig. 7. Dynamic coding method for fuzzy rule bases.

GA, but the range of the center value, c_i, in the MF cannot be fixed. A fixed range for c_i will degrade the performance of the fuzzy system because it reduces the search space in the GA or sometimes permits the center values of the MFs to be swapped. These are shown in Fig. 7 as Case 1 and Case 2, respectively. In Case 1, the range of c_i is fixed to produce a new population in each generation without overlapping LB_i and UB_i in (15). In Case 2, the range of c_i is fixed but LB_i and UB_i are overlapping.

To maximize the search space and prevent the change of meaning in the MFs during the GA operation, a dynamic coding method for c_i will be adopted. The range of c_i in each generation is dynamically varied from parents to children. Fig. 7 shows the dynamic coding method graphically.

We can generate the center values in MFs of the new population from that of parents. We assume that c_{ij} is the center value of the i-th present population with j-th MF and c_j^{new} is the center value of a new population with j-th MF. In the dynamic coding method, the range of the new population is defined as follows:

$$\begin{aligned}
&a_1 \le c_1^{new} \le (\textstyle\sum_{i=1}^{2}\sum_{j=1}^{2} c_{ij})/4 \\
&c_1^{new} < c_2^{new} \le (\textstyle\sum_{i=1}^{2}\sum_{j=2}^{3} c_{ij})/4 \\
&\quad\vdots \\
&c_{k-1}^{new} < c_k^{new} \le (\textstyle\sum_{i=1}^{2}\sum_{j=k}^{k+1} c_{ij})/4 \\
&c_{n-1}^{new} < c_n^{new} \le b_1
\end{aligned} \tag{18}$$

where $[a_1, b_1]$ is the universe of discourse in the fuzzy variable of c_j in (3). The dynamic coding method maximizes the search space and does not

permit any overlap behavior between MFs.

4. Performance Test and Evaluation

4.1 Box and Jenkins's Gas Furnace

This section presents an example of fuzzy modeling of a dynamic process using gas furnace data. The data set includes 296 successive input-output pairs of observations, (u_t, y_t), with a sampling interval of 9 seconds. The performance index (PI) of the model in (19) is used for comparison with other identification algorithms,

$$\Pi = \frac{1}{N}\sum_{k=1}^{N}(y(k)-\hat{y}(k))^2. \quad (19)$$

The fuzzy model is assumed to take the form $y(t) = F(y(t\text{-}1),u(t\text{-}4))$ in order to compare it with other methods. The initial values used in the GA are as follows: number of populations=100, number of generations=300, crossover rate=0.75, and mutation rate=0.001. Our method produces final membership functions for the two inputs as shown in Fig. 8. The output can be inferred from the complete rule base shown in Table 1. Fig. 9 shows the model behavior compared with the actual process. The PI

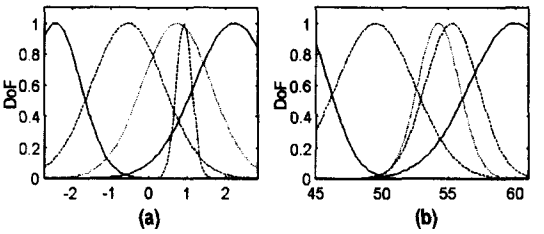


Fig. 8. (a) Final MFs on $u(t\text{-}4)$ and (b) Final MFs on $y(t\text{-}1)$.

Table 1. Fuzzy rule base: bold face are identified after PNN

		$y(t\text{-}1)$				
		NB	***NS***	***ZE***	***PS***	***PB***
$u(t\text{-}4)$	***NB***	47.5	57.1	**59.4**	64.1	60.1
	NS	56.6	55.7	**47.5**	**57.5**	51.1
	ZE	45.4	**49.7**	**53.4**	**57.8**	59.7
	PS	46.6	31.8	57.7	66.7	**75.0**
	PB	**44.5**	**52.5**	49.0	66.6	27.2

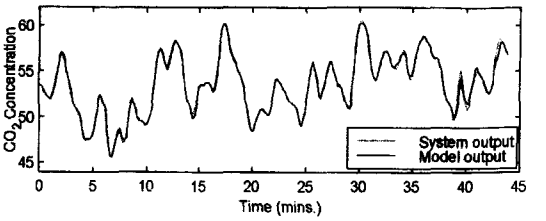


Fig. 9. The PFNN model for the Box-Jenkins data.

Table 2. Comparison of fuzzy model with other models

Model Name	Inputs	Number of rule	Model Error
Box nd Jenkins[1][2]	6	ARMA model	0.202
Tong's model[29]	$y(t\text{-}1),u(t\text{-}4)$	19	0.469
Pedrycz's model[21]	$y(t\text{-}1),u(t\text{-}4)$	25 49 81	0.776 0.478 0.320
Xu's model[34]	$y(t\text{-}1),u(t\text{-}4)$	25	0.328
TSK model[27]	$y(t\text{-}1),u(t\text{-}4)$	2	0.359
Linear model[1][28]	$y(t\text{-}1),u(t\text{-}4),$ $u(t\text{-}3),u(t\text{-}4),$ $u(t\text{-}5)$	–	0.193
Modified TSK model[28]	$y(t\text{-}1),u(t\text{-}3),$ $u(t\text{-}4)$	6	0.190
Kupper's model[14]	$y(t\text{-}1),u(t\text{-}4)$	25	0.166
Saleem *et al.*[23]	$y(t\text{-}1),u(t\text{-}2),$ $y(t\text{-}3),u(t\text{-}4)$	4-4-1 network[2]	0.417
	$y(t\text{-}1),u(t\text{-}3),$ $u(t\text{-}4)$	–	0.403
Wang *et al.*[32]	$y(t\text{-}1),u(t\text{-}4)$	5	0.158
Present model	$y(t\text{-}1),u(t\text{-}4)$	11	0.108

[1]Non fuzzy model. [2]Neural network model.

of the proposed method is computed as 0.108. The excellent performance of the proposed model is highlighted when compared with the other fuzzy models in Table 2.

Table 3 shows a comparison of performance with different optimization methods under the same fuzzy model. From this table, the performance of a hybrid optimization method combined with a PFNN is better than that exhibited by other methods.

Table 3. Comparison with optimization methods

	Fuzzy model with			PNN without fuzzy
	GA	Simplex Method	Hybrid Method	
without PNN	00.125	0.287	0.121	–
with PNN	0.159	0.177	**0.108**	0.186

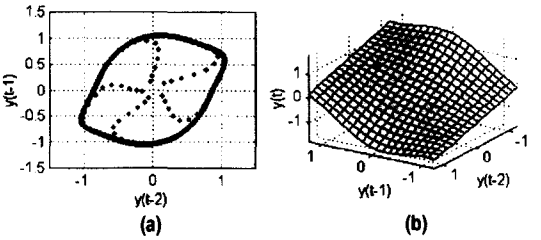


Fig. 10. (a) Iterated time series mapping from the initial condition $(0.1,\ 0.1)^T$. (b) Time series output surface $y(t)$ vs. $y(t\text{-}1)\times y(t\text{-}2)$.

4.2 Nonlinear Time Series Modeling

A second benchmark problem is the time series function described by the following second-order nonlinear difference equation:

$$y(t)=(0.8-0.5\exp(-y^2(t-1)))y(t-1)$$
$$-(0.3+0.9\exp(-y^2(t-1)))y(t-2)$$
$$+\sin(\pi y(t-1))+\eta(t) \qquad (20)$$

where $\eta(t)$ denotes the additive noise at time t. When $\eta(t)=0$, this difference equation has an unstable equilibrium at the origin and a globally attracting limit cycle, as shown in Fig. 10. Different modeling methods, such as CMAC and B-splines in [7], and RBF and MLP in [1,3], have been applied to the same problem.

In this problem, $\eta(t)$ has been assumed as a zero-mean Gaussian white noise sequence with variance 0.01. A set of noisy training samples are collected from (20). From an initial condition $x(1)=(y(0),\ y(-1))=\mathbf{0}$, 300 iterated noisy time series samples are generated for training inputs and shown in Fig. 11.

After training with the proposed method, the identified PFNN model is evaluated using two different tests. In the first one, the normalized output error autocorrelation function, $\phi(k)$, is used to evaluate the approximation ability over the training set, and is computed as follows:

$$\phi(k)=\frac{\sum_{t=1+k}^{N}\varepsilon(t)\varepsilon(t-k)}{\sum_{t=1}^{N}\varepsilon^2(t)} \qquad (21)$$

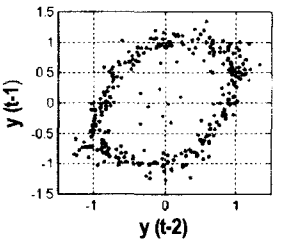


Fig. 11. The noisy iterated dynamics of the time series from an initial position at the origin.

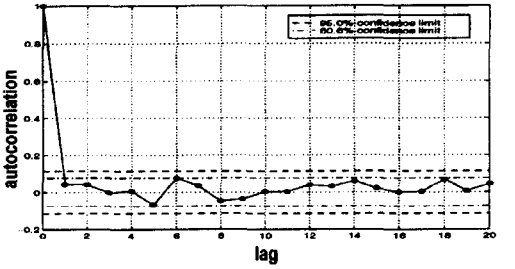


Fig. 12. The autocorrelation of the prediction errors after the 20th training cycle. The 9.5% confidence band is shown by the dashed lines.

where k is the time lag, N is the training set size, and $\varepsilon(t)$ is the output error at sample time t over the training set. When the identified model reproduces the underlying function exactly, $\phi(k)$ becomes an impulse function at $k=0$; i.e., $\phi(k)$ is equal to one at $k=0$, and zero otherwise. Generally, when the training set is large, the standard deviation of the correlation estimate is $1/\sqrt{N}$, and the 95% confidence limits are $\pm 1.96/\sqrt{N}$. The autocorrelation plot for the PFNN model is shown in Fig. 12, and the values lie within the 80.6% confidence band rather than the 95%. Thus, there has been an approximate improvement of 14.4%, whereas other methods are barely inside or outside the 95% confidence band.

The second test for assessing the model's performance is to iterate the identified model from an initial condition $\mathbf{x}(0)=(y(0),\ y(-1))=(0.1,\ 0.1)$, and then plot the resulting dynamic behavior. The phase plot and the time series surface plot are shown in Fig. 13. The basic shape of the limit cycle is a good approximation to the true shape, and a five armed interior spiral can be seen clearly.

4.3 Predicting Chaotic Dynamics

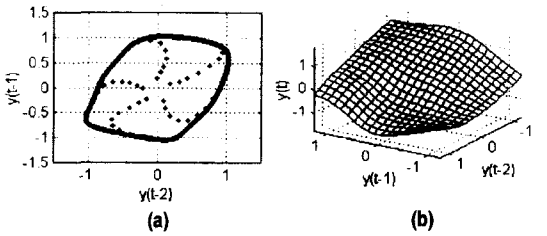


Fig. 13. (a) The iterated dynamical behavior of the PFNN from the initial condition $x(1)=(0.1, 0.1)^T$. (b) The PFNN approximation of the true time series surface.

The time series used in this simulation is generated by the chaotic Mackey-Glass differential delay equation defined as:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{22}$$

The Mackey-Glass equation was first proposed as a model of white blood cell production[16] and subsequently popularized in the nonlinear field due to its richness in structure. The prediction of future values of this time series is a benchmark problem which has been considered by a number of new adaptive computing techniques: neural network approaches[8,24] radial basis function approaches[18], and fuzzy reasoning algorithms[10,19].

The goal of the task is to predict known values at some point in the future $x=t+P$. The standard method for this type of prediction is to create a mapping from D points of the time series spaced Δ apart, that is, $x(t-(D-1)\Delta), \cdots, x(t-\Delta), x(t))$, to a predicted future value $x(t+P)$. To allow comparison with earlier work (Moody[18], Crowder[4], Jang[10], the values $D=4$ and $\Delta=P=6$ were used. All other simulation settings in this example were purposely arranged to be as close as possible to those reported in [4,10].

To obtain the time series values at each integer point, we applied the fourth-order Runge-Kutta method. From the Mackey-Glass time series $x(t)$, 1000 input-output data pairs were extracted with the following format:

$$[x(t-18), x(t-12), x(t-6), x(t);\ x(t+6)] \tag{23}$$

where t=124 to 1123. The first 500 pairs (training data set) were used for training the PFNN while the remaining 500 pairs (validation data set) were used for validating the identified model. The number of membership functions assigned to each input is arbitrarily set to 2, so the number of rules is 16. Note that the PFNN used here contains a total of 24 fitting parameters, of which 8 are premise parameters and 16 are consequent parameters. The initial values used in the GA are: population size=100, number of generations= 200, crossover rate=0.75, and mutation rate=0.001.

The identified final model has $RMSE_{trn}$=0.0046 and $RMSE_{chk}$=0.0060, which is included in the top group when compared with other approaches as explained below. The desired and predicted values for both training and checking data are essentially the same in

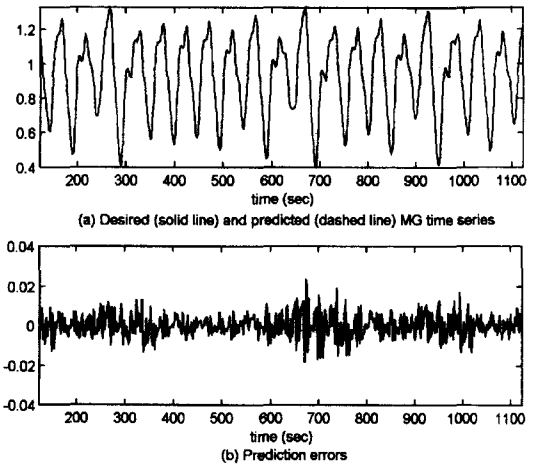


Fig. 14. (a) Mackey-Glass time series from t=124 to 1123 and one-step ahead prediction; (b) prediction errors.

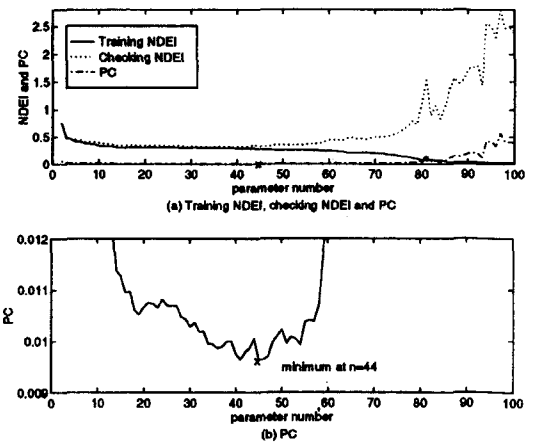


Fig. 15. Training and checking errors and PC of AR model with different parameter numbers.

Fig. 14(a); the differences are shown in Fig. 14(b).

As a comparison, we performed the same prediction by using the auto-regressive (AR) model as follows:

$$x(t) = a_0 + a_1 x(t-6) + a_2 x(t-2\times 6) + \cdots + a_n x(t-n\times 6) \quad (24)$$

where there are $n+1$ fitting parameters a_k, $k=0$ to n. To search for the best AR model in terms of its generalization capability, the number of parameters was varied from 2 to 100. The AR model, based on the *PC* in (10), was obtained when the parameter number is 45 ($n=44$), as shown in Fig. 15. The best AR model according to the *PC* shows the result in Fig. 16 where there is no overfitting at the price of larger training error. For $n=44$, 1000 data pairs were extracted from $t=364$ to 1363, of which the first 500 pairs were used to identify a_k and the remaining 500 pairs were used for checking. The results obtained through the standard least squares estimates are $RMSE_{trn}=0.063$ and $RMSE_{chk}=0.075$, which are much worse than those of the PFNN.

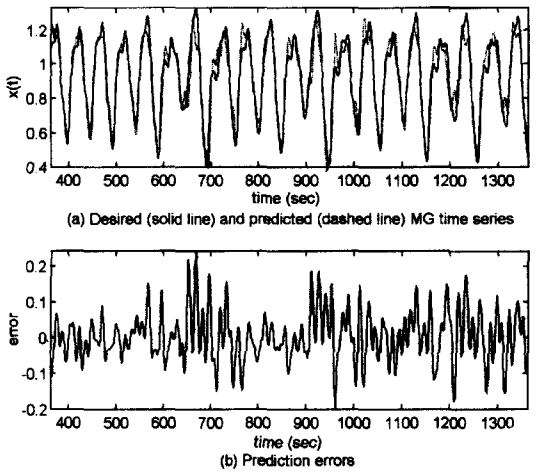


Fig. 16. (a) Mackey-Glass time series from $t=364$ to 1363 and one-step ahead prediction by the best AR model ($n=44$); (b) prediction errors.

Table 4. Generalization result comparisons for $P=6$

Method	Training Cases	Error Index (NDEI)
PENN	500	0.02
AR Model	500	0.32
ANFIS[10]	500	0.01
Cascde-Correlation NN[1]	500	0.06
Back-Prop NN[1]	500	0.02
6th-order Polynomial[1]	500	0.04
Linear Predictive Method[1]	2000	0.55

[1]These are taken from [4].

Table 4 summarizes the generalization capabilities of several other methods; each method is called upon to predict 500 points immediately following the training set. Here, the *non-dimensional error index* (NDEI) is defined as the root mean squared error divided by the standard deviation of the target series[4].

Table 5 lists the results of the more challenging generalization tests when $P=84$ (the first seven rows)

Table 5. Generalization result comparisons for $P=84$ (the first seven rows) and 85 (the last six rows). Results for the seven methods are generated by iterating the solution at $P=6$. Results for localized receptive fields (LRF) and multi-resolution hierarchies (MRH) are for networks trained for $P=85$

Method	Training Cases	Error Index (NDEI)
PENN	500	0.06
AR Model	500	0.53
ANFIS[10]	500	0.04
Cascade-Correlation NN[1]	500	0.32
Back-Prop NN[1]	500	0.05
6th-order Polynomial[1]	500	0.84
Linear Predictive Method[1]	2000	0.60
LRF[1]	500	0.10~0.25
LRF[1]	10000	0.03~0.05
MRH[1]	500	0.05
MRH[1]	10000	0.02
Locally Tuned Network[18]	500	0.08
Fuzzy-neural approach[19]	500	0.21

[1]These are taken from[4].

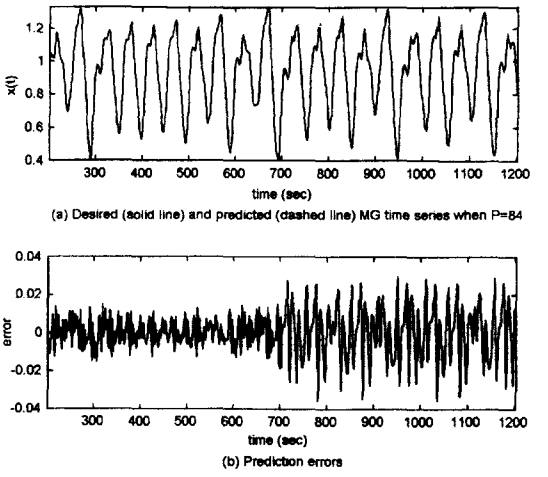


Fig. 17. Generalization test of PFNN for $P=84$.

and P=85 (the last six rows). The results of the first seven rows are obtained by iterating the prediction of P=6 till P=84. The PFNN still shows good performance. Fig. 17 illustrates the results for P=84, where the first 500 points are the desired outputs of the training set while the last 500 are the predicted outputs.

The PFNN method has been shown to accurately predict the behavior of the Mackey-Glass equation. We have obtained predictions that match the accuracy of previous approaches as in Tables 4 and 5. If the number of membership functions in the input variables is increased, the accuracy can be improved further.

5. Conclusions

This paper introduces a new neuro-fuzzy system based on the PFNN architecture. An effective optimization method through a hybrid genetic algorithm is introduced to optimize parameters. A new performance criterion is defined to minimize the output error while preventing overfitting of the empirical data set. A novel coding scheme is presented to describe fuzzy systems for a dynamic search range in the GA. For a performance assessment of the PFNN inference system, three well-known problems are used for comparison with other methods. The results of these comparisons show that the PFNN inference system outperforms and is more robust than the other methods. A PFNN modeling and hybrid genetic optimization method can be employed to provide optimum set points for low-level control activity.

References

[1] P. C. An, M. Brown, C. J. Harris and S. Chen, "Comparative aspects of neural network algorithms for on-line modelling of dynamic processes", *IMechE, Proc. Instn. Mech. Engrs., Part I, J. Sys. Cont. Eng.*, **207**, pp. 223-241, 1993.

[2] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*, San Francisco, Holden Day, 1976.

[3] S. Chen and S. A. Billings, "Neural networks for nonlinear dynamic system modelling and identification", in *Advances intelligent Control* (C. J. Harris, ed.), ch. 4, Taylor and Francis, 1994.

[4] R. S. Crowder, "Predicting the Mackey-Glass timeserieswith cascade-correlation learning", In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proc. of the 1990 Connectionist Models Summer School*, Morgan Kaufmann Pub., pp. 117-123, 1990.

[5] S. Farlow, ed., *Self Organizing Methods in Modeling: GMDH-Type Algorithm*, Marcel Dekker, Inc., New York, 1984.

[6] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[7] C. J. Harris, *Neurofuzzy Adaptive Modelling and Control*, ch. 6 and 8. Prentice Hall, 1994.

[8] W. Hsu, L. S. Hsu and M. F. Tenorio, "A ClusNet architecture for prediction", *IEEE Int. Conf. on Neural Networks*, **1**, pp. 329-334, 1993.

[9] A. G. Ivakhnenko, "Polynomial theory of complex systems", *IEEE Trans. Syst. Man and Cybern.*, **12**, pp. 364-378, 1971.

[10] J. S. R. Jang and C. T. Sun, "Predicting chaotic time series with fuzzy if-then rules", *IEEE Int. Conf. on Fuzzy Systems*, **2**, pp. 1079-1084, 1993.

[11] J. S. R. Jang. "ANFIS: Adaptive-network-based fuzzy inference systems", *IEEE Trans. Syst. Man Cybern.*, **23**(3), pp. 665-685, May 1993.

[12] S. Kim, A. Kumar, J. L. Dorrity and G. Vachtsevanos, "Fuzzy modeling, control and optimization of textile processes", *Proc. of NAFIPS/IFIS/NASA*, San Antonio, TX, pp. 32-38, Dec. 1994.

[13] B. Kosko, "Fuzzy systems as universal approximators", *IEEE Trans. on Computers*, **43**(11), pp. 1329-1333, 1994.

[14] K. Kupper, "Self learning fuzzy models using stochastic approximation", *Proc. of the 3rd IEEE Conf. on Control Applications*, **3**, pp. 1723-1728, 1994.

[15] M. A. Lee and H. Takagi, "Integration design stages of fuzzy systems using genetic algorithm," *2nd IEEE Int. Conf. on Fuzzy Systems*, **1**, pp. 612-617, Mar. 1993.

[16] Y. Lin and G. A. Cunningham III, "A new approach to fuzzy-neural system modeling," *IEEE Trans. on Fuzzy Systems*, **3**(2), pp. 190-198, 1995.

[17] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems", *Science*, **197**, pp. 287-289, July 1977.

[18] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units", *Neural Computation*, **1**, pp. 281-294, 1989.

[19] J. Nie, "A fuzzy-neural approach to time series prediction", *IEEE Int. Conf. on Neural Networks*, **5**, pp. 3164-3169, July 1994.

[20] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks", *Neural Computation*, **5**(2), pp. 305-316, 1993.

[21] W. Pedrycz, "An identification algorithm in fuzzy relational systems", *Fuzzy Sets and Systems*, **13**, pp.

153-167, 1984.

[22] S. S. Rao, *Optimization: Theory and Applications*. New York, John Wiley & Sons, pp. 209-300, 1984.

[23] R. M. Saleem and B. E. Postlethwaite, "A comparison of neural networks and fuzzy relational systems in dynamic modelling", *Int. Conf. on Control '94*, **2**, pp. 1448-1452, 1994.

[24] T. D. Sanger, "A tree-structured adaptive network for function approximate in high-dimensional space", *IEEE Trans. on Neural Networks*, **2**, pp. 285-293, March 1991.

[25] S. Shrier, R. L. Barron and L. O. Gilstrap, "Polynomial and neural networks: analogies and engineering applications", *IEEE First Int. Conf. on Neural Networks*, **2**, pp. 431-439, 1987.

[26] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model", *Fuzzy Sets and Systems*, **28**, pp. 15-33, 1988.

[27] M. Sugeno and K. Tanaka, "Successive identification of a fuzzy model and its application to prediction ofa complex system", *Fuzzy Sets and Systems*, **42**, pp. 315-334, 1991.

[28] M. Sugeno and T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, *IEEE Trans. Fuzzy Systems*, **1**(1), pp. 7-31, 1993.

[29] R. M. Tong, "The evaluation of fuzzy models derived from experimental data", *Fuzzy, Sets and Systems*, **4**, pp. 1-12, 1980.

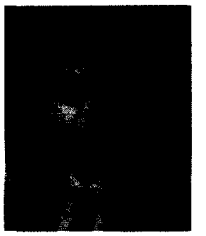
[30] G. J. Vachtsevanos, S. S. Kim, J. R. Echauz and V. K. Ramani, "Neuro-Fuzzy Approaches to Decision Making: A Comparative Study with an Application to Check Authorization", *Journal of Intelligent and Fuzzy Systems*, **6**, pp. 259-278, 1998.

[31] G. Vachtsevanos, J. L. Dorrity, A. Kumar and S. Kim, "Advanced application of statistical and fuzzy control to textile processes", *IEEE Trans. on Industry Applications*, **30**(3), pp. 510-516, May-June 1994.

[32] L. Wang and R. Langari, "Complex systems modeling via fuzzy logic", *Proc. of IEEE Conf. on Decision and Control*, **4**, pp. 4136-4141, 1994.

[33] L. X. Wang and J. M. Mendel, "Back-propagation fuzzy systems as nonlinear dynamic system identifiers", *Proc. IEEE Int. conf. on Fuzzy Systems*, San Diego, pp. 1409-1418, 1992.

[34] C. W. Xu, "Fuzzy systems identification", *IEEE Proceedings*, **136**, Part. D, no. 4, 1989.



김 성 신(Sungshin Kim)

1984년 : 연세대학교 전기공학과 공학사

1986년 : 연세대학교 전기공학과 공학석사

1996년 : Georgia Inst. of Tech. 공학박사

1996년~1998년 : Georgia Inst. of Tech. 연구원

1997년~1998년 : Appalachian Electronic Instruments Inc. 연구원

1998년~현재 : 부산대학교 전기공학과 조교수