An Algorithm for Extracting Fuzzy Rules Based on RBF Neural Network

Wen Li and Yoichi Hori, *Fellow, IEEE*

***Abstract*—A four-layer fuzzy–neural network structure and some algorithms for extracting fuzzy rules from numeric data by applying the functional equivalence between radial basis function (RBF) networks and a simplified class of fuzzy inference systems are proposed. The RBF neural network not only expresses the architecture of fuzzy systems clearly but also maintains the explanative characteristic of linguistic meaning. The fuzzy partition algorithm of input space, inference algorithm, and parameter tuning algorithm are also discussed. Simulation examples are given to illustrate the validity of the proposed algorithms.**

***Index Terms*—Explanative characteristic, fuzzy rules, radial basis function (RBF) neural network.**

I. Introduction

ESSENTIALLY, system modeling is the task of building models from a combination of *a priori* knowledge and empirical data. When a complex system is to be modeled, usually, the only available information is a collection of empirical data, which are inherently imprecise and incomplete as obtained from the observation of the system behavior or the measurement of some system states. There are some types of modeling with imprecise and incomplete data, such as fuzzy modeling and rough modeling. Fuzzy modeling based on numerical data, which was first explored systematically by Takagi and Sugeno [7], has found numerous successful applications to complex system modeling.

Considerable works on hybrids between fuzzy inference and neural networks have been done to integrate the excellent learning capability of neural networks with fuzzy inference systems, resulting in neuro-fuzzy modeling approaches that combine the benefits of these two powerful paradigms into a single capsule and provide a powerful framework to extract good-quality fuzzy rules from numerical data [2], [8], [9].

The error backpropagation neural network (BP) is used widely because its learning scheme is visible and easy to understand. However, the classification capability is lower for those patterns away from the sample set or for new patterns. Hence, radial basis function networks (RBFNs) are recently adopted widely for fuzzy rules drawing and fuzzy inference system modeling because they possess simple structure, good local approximating performance, particular resolvability, and function equivalence with a simplified class of fuzzy inference systems [1]. Because of function equivalence, which made it possible to combine the features of these two systems, a powerful type of neuro-fuzzy systems was developed [2]. However, a fuzzy system that has been trained using learning algorithms may lose its interpretability or transparency, which is one of the most important features of fuzzy systems.

In this paper, a structure of RBFN, which can represent the interpretability of fuzzy systems efficiently, is proposed based on the analysis of RBFNs. Then, the learning algorithms of extracting fuzzy rules from this RBFN are discussed in detail.

Manuscript received August 30, 2004; revised October 18, 2004. Abstract published on the Internet May 18, 2006. This work was supported by the Science and Technology Fund of the Ministry of Education of China.

W. Li was with the University of Tokyo, Tokyo 153-8505, Japan. She is now with the Electrical Engineering School, Dalian Jiaotong University, Dalian 116028, China (e-mail: lw6017@vip.sina.com).

Y. Hori is with the Information and Electronics Division (Electrical Control System Engineering), Institute of Industrial Science, University of Tokyo, Tokyo 153-8505, Japan (e-mail: hori@iis.u-tokyo.ac.jp; y.hori@ieee.org).

Digital Object Identifier 10.1109/TIE.2006.878305

II. RBFN Construction and Classification Mechanism

RBFN belongs to a kind of forward networks that are structured based on the theory of functional approximation. Network learning is equivalent to searching the best surface matched with training data in multidimensional space. The activation function in each node of the hidden layers of the network forms a basis function of the matched surface from which the name of the network originates. It is known that a BP network is a typical global approximation network, whose network output is decided by all the neurons of the network. Compared with the BP network, RBFN is a type of local approximation network, i.e., the network output is decided by a few neurons existing in a certain local area in input space. Although the size of the RBFN is bigger than that of the BP network, its performance characteristics such as learning speed, ability for approximation, pattern recognition, and classification are better than the same characteristics of the BP network.

A. General RBFN Structure

A three-layer RBFN with N inputs, L nodes (neurons) in the hidden layer, and M neurons in the output layer is shown in Fig. 1. Although RBFNs belong to forward network models because of their structure, the method of initializing parameters is different from the BP model, in which parameters are initialized randomly. The parameters of RBFN such as the center and width of receptive fields are determined according to the distribution of sample data [3]. Radial basis functions (RBFs) are adopted as active functions of the hidden layer nodes, and there are three RBFs commonly used [4]. In this

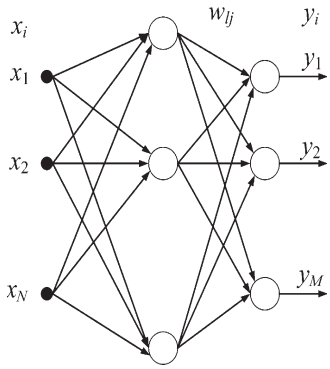

Fig. 1. RBF network structure.

paper, Gaussian function is chosen as the active functions of the hidden layer nodes

$$\varphi_l(X) = \exp\left[-\frac{\|X - C_l\|^2}{\sigma_l^2}\right], \qquad l = 1, \ldots, L \tag{1}$$

where X is an N-dimensional input vector, C_l is a vector (i.e., the center of the Gaussian function) with the same dimension as X, L is the number of nodes in the hidden layer, and σ_l, which is a scalar quantity, is the width of the Gaussian function. If the transfer functions of the output layer nodes are linear functions, then the output of each output node is

$$y_j(X) = \sum_{l=1}^{L} w_{lj}\varphi_l(X), \qquad j = 1, \ldots, M. \tag{2}$$

The normalized output is

$$y_j(X) = \frac{\sum_{l=1}^{L} w_{lj}\varphi_l(X)}{\sum_{l=1}^{L} \varphi_l(X)} \tag{3}$$

where w_{lj} denotes the weight between the jth output and the lth node in the hidden layer. The performance index function E is expressed as

$$E = \frac{1}{2}(y^t - y)^2 \tag{4}$$

where y is the output of the network and y^t is the target value.

The structural form of RBFNs, as shown in Fig. 1, is a whole linkage network. The next section will discuss other network forms.

B. RBFNs for Representing Interpretability of Fuzzy Systems

It has been proved that RBFN is functional equivalent with a simplified class of fuzzy inference systems [1]. However, the only difference between the two systems is interpretability,

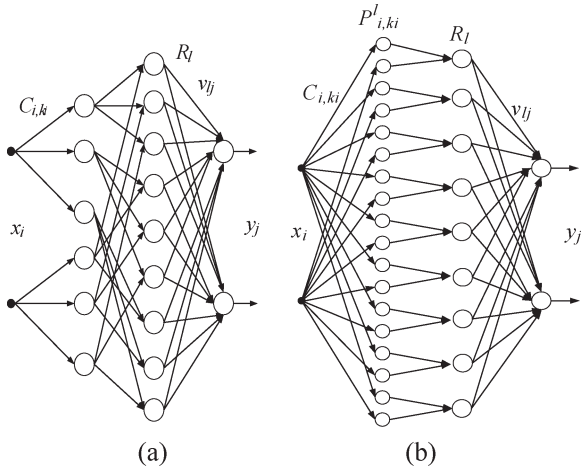

Fig. 2. Two kinds of network structures.

which makes fuzzy systems easy to understand [5]. Representing a fuzzy system with a general RBFN weakens the outstanding interpretability of fuzzy systems. Therefore, most RBFNs used to extract fuzzy rules or to implement fuzzy inference are the transformed ones of the whole linkage form in practice. Those transformed RBFNs can improve the interpretability of networks used to express fuzzy systems. In this section, two transformed RBFNs are given in Fig. 2 [5], [10].

The two networks have the same number of input variables. For the structure in Fig. 2(a), the number of nodes in hidden layer 1 is equal to the sum of the fuzzy partitions of each input variable, and the node linked with an input variable denotes a fuzzy subset of the input variable, which describes clearly the fuzzy partition of input space. Hidden layer 2 is used to implement the algorithm of fuzzy inference, and the number of nodes is the number of fuzzy rules, i.e., each node is associated with a fuzzy rule. The overall outputs are acquired from the output layer. In addition, for the structure in Fig. 2(b), the number of nodes in hidden layer 1 is equal to the product of premise-number and rule-number; the n nodes connected to a node in hidden layer 2 form the premise part of a fuzzy rule. Therefore, the components of fuzzy rules can be described clearly with the two hidden layers. Simultaneously, hidden layer 2 is also the fuzzy inference layer. In addition, the output layer has the same function as that of the structure in Fig. 2(a).

From the preceding comparison and analysis, it can be observed that the structure in Fig. 2(a) is not only simple but also capable of describing clearly the fuzzy partitions of input space. However, the more input variables, the more difficult to express fuzzy systems and the worse the interpretability. With regard to expressing fuzzy partitions in input space, although structure Fig. 2(b) is not clearer than the structure in Fig. 2(a), it has better representational power on the interpretability of fuzzy systems than the structure in Fig. 2(a), which makes the described fuzzy systems more comprehensible than conventional neural systems.

C. New Network Structure

According to the analysis and comparison of structures and natures of the two networks in Fig. 2, it can be observed that the

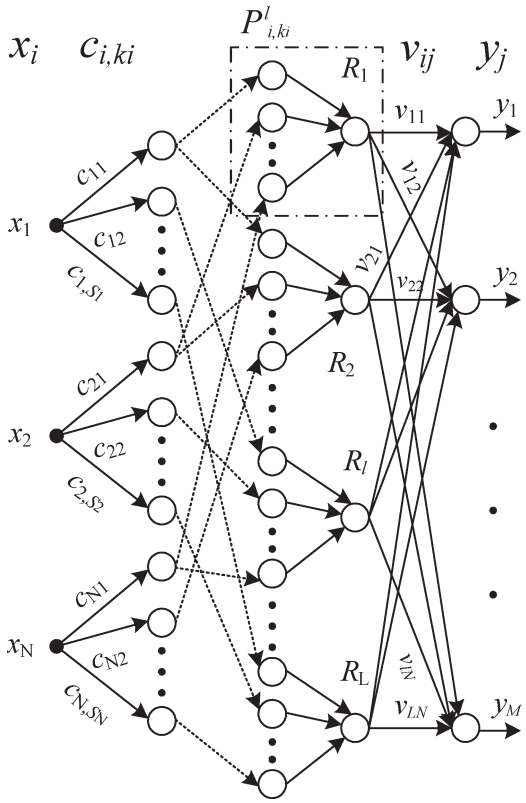

Fig. 3. New RBF neural network.

hidden layer 1 of the structure in Fig. 2(b) is only an extension of the hidden layer 1 of the structure in Fig. 2(a). Based on this idea, a new structure of RBFN is proposed for the purpose of improving the interpretability of fuzzy systems. Fig. 3 gives the new network structure, which integrates the natures of the two networks shown in Fig. 2. The new structure cannot only represent the fuzzy partitions of input space clearly but can also give the formal description of fuzzy systems intuitively. Compared with the structure in Fig. 2(a), for the new structure, the representational power of the interpretability is improved greatly, and the ability to clearly express fuzzy partitions is maintained. Because the weights between hidden layer 1 and hidden layer 2 are 1, the performance of the running network is not changed in nature. Although the structure increases memory units outwardly, the parameter set of hidden layer 2 is, in fact, equal to that of hidden layer 1. Therefore, a quick running speed of the network and the small memory units may be obtained by applying a suitable learning algorithm and storing method. The network structure and the method of selecting initial parameters have been discussed in the literature [6].

III. Learning Algorithm Design

Let $X = (x_1, x_2, \ldots, x_N)$ denote the N-dimensional input space, where $x_i \;\; i = 1, 2, \ldots, N$ is an input variable; and $Y = (y_1, y_2, \ldots, y_M)$ denote the M-dimensional output space, where $y_j \;\; j = 1, 2, \ldots, M$ is an output variable. According to the network structure, the learning algorithm is composed of a fuzzy partition algorithm of input space, a fuzzy inference algorithm, and an inference output algorithm, which are given in following subsections.

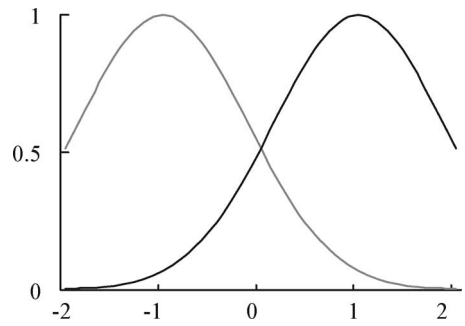

Fig. 4. Definition of overlap degree.

A. Fuzzy Partition Algorithm of Input Space

The input layer and hidden layer 1 of the network form the fuzzy partition part, and the corresponding algorithm is used to implement the fuzzy partition of input space. Each input node x_i is connected to the corresponding s_i nodes in hidden layer 1, and s_i denotes the number of fuzzy partition for variable x_i. In this paper, $s = (s_1, s_2, \ldots, s_N)$ denotes the number of all the fuzzy partitions , notation $c_{ik_i} \;\; k_i = 1, 2, \ldots, s_i$ denotes the weights from the ith input node to the k_ith node in hidden layer 1, and k_i denotes the k_ith fuzzy partition of the ith input variable. The fuzzy partition labels of N input variables are denoted by notation $k = (k_1, k_2, \ldots, k_N)$, where each k_i corresponds to $s_i \in (s_1, s_2, \ldots, s_N)$.

The Gaussian function (a kind of RBF) is adopted as the transfer functions of the nodes in hidden layer 1; hence, the output of the nodes can be written as

$$f_{i,k_i}(x_i)\, e^{-(x_i - c_{ik_i})^2/\sigma^2_{ik_i}} \tag{5}$$

where c_{ik_i} and σ_{ik_i} are the center and width of the Gaussian function, respectively. The initial value of c_{ik_i} is determined by the initial clustering center [6].

In order to determine parameter σ_{ik_i}, a conception of overlap degree is introduced. The overlap degree is the degree by which two fuzzy subsets overlap, which is measured by the maximum membership degree of intersection produced by the two subsets. As shown in Fig. 4, for example, the overlap degree is 0.5. In fuzzy control, overlap degree is an important factor that affects control performance. Generally, overlap degree should be around 0.5; a value that is too big or too small may result in an unexpected control effect. Considering this case, a formula for determining the initial width of the Gaussian function is deduced.

Let the distance between two adjacent clustering centers be $d = \|c_i - c_{i-1}\|$, corresponding to clustering center c_i, and the RBF be $f_i(x_i)\; e^{-(x_i - c_i)^2/\sigma_i^2}$. In the selection of width σ_i, the overlap degree of adjacent fuzzy subsets should remain to be around 0.5. Therefore, when $\|x_i - c_i\|(d/2)$,

$$0.3679 < f_i(x_i) < 0.7788 \tag{6}$$

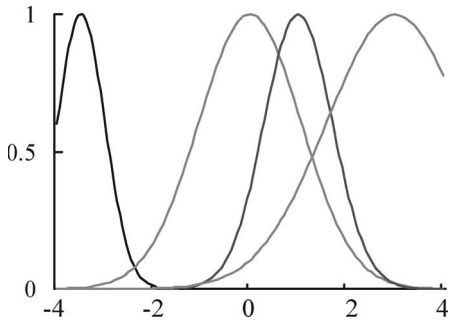

Fig. 5. Fuzzy partition when membership functions are symmetric.

i.e., $1/4 < ((x_i - c_i)^2/\sigma_i^2)((d/2)^2/\sigma_i^2) < 1$. Thus, for the RBFN shown in Fig. 3, width σ_i should be selected by the following equation:

$$\sigma_i = \frac{\|c_i - c_{i-1}\|}{\gamma}, \qquad 1 < \gamma < 2 \tag{7}$$

where γ is the overlap coefficient.

Usually, a clustering center corresponds to a fuzzy subset using an RBF as its membership function. Taking Fig. 5 as an example, there are four clustering centers $c_1 = -3.5$, $c_2 = 0$, $c_3 = 1$, and $c_4 = 3$; each corresponds to an RBF with $\sigma_i = \|c_i - c_{i-1}\|/\gamma$, $\gamma = 1.5$, which define the fuzzy subsets (fuzzy partitions) at region [-4, 4]. As shown in Fig. 5, because the different intervals of adjacent clustering centers result in different widths σ_{i-1} and σ_i from (7), (6) cannot always be satisfied. In order to keep the overlap degree of the adjacent fuzzy subsets to about 0.5, the membership functions should be reconstructed as

$$f_i(x_i) \begin{cases} e^{-(x_i - c_i)^2/\sigma_{il}^2}, & x_i \le c_i \\ e^{-(x_i - c_i)^2/\sigma_{ir}^2}, & x_i > c_i \end{cases} \tag{8}$$

where

$$\sigma_{il} = \frac{\|c_i - c_{i-1}\|}{\gamma} \tag{9}$$

$$\sigma_{ir} = \frac{\|c_{i+1} - c_i\|}{\gamma}. \tag{10}$$

Equation (8) gives the mathematical description of nonsymmetry membership function. As long as membership functions are defined by (8) and the selection of widths σ_{il} and σ_{ir} are defined by (9) and (10), it can be ensured that the overlap degree of two adjacent fuzzy subsets satisfies (6), no matter how different the two neighbored intervals of adjacent clustering centers are. Fig. 6 shows a fuzzy partition case that has the same conditions as that in Fig. 5, except that the membership functions are described by (8). With (8) reconstructing RBF, (5) is rewritten as

$$f_{i,k_i}(x_i) = \begin{cases} e^{-(x_i - c_{ik_i})^2/\sigma_{il,k_i}^2}, & x_i \le c_{ik_i} \\ e^{-(x_i - c_{ik_i})^2/\sigma_{ir,k_i}^2}, & x_i > c_{ik_i} \end{cases}. \tag{11}$$

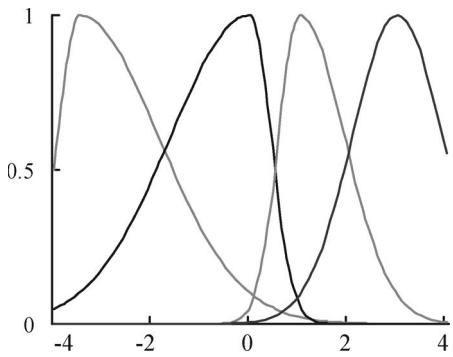

Fig. 6. Fuzzy partition when membership functions are nonsymmetric.

The input layer and hidden layer 1 describe clearly the fuzzy partition status of each dimension in input space for a control system, i.e., the definition of the fuzzy subsets (fuzzy linguistic values) of each input variable. Supposing the crisp input is $x^0(x_1^0, x_2^0, \ldots, x_N^0)$, the membership degree of each variable that belonged to various fuzzy subsets can be calculated using (11). Hence, the fuzzification process of the input variables is completed.

B. Forward Inference Algorithm

The inference task of fuzzy systems is implemented by hidden layer 2 and hidden layer 3. The L node groups denote L rules. In each group, there are N input nodes $P_{ik_i}^l$, $i = 1, 2, \ldots, N$ $k_i \in (1, 2, \cdots, s_i)$, which correspond to the N premises of the lth rule. Notation $P_{ik_i}^l$ denotes that the ith premise of the lth rule takes the k_ith linguistic value $A_{ik_i}^l$. When there is a crisp input $x^0(x_1^0, x_2^0, \ldots, x_N^0)$, $P_{ik_i}^l$ should be the membership degree of x_i^0 that belongs to $A_{ik_i}^l$. In other words, for l outputs in hidden layer 2 $P_{ik_i}^l$, $l = 1, \cdots, L$, the following equation exists:

$$P_{ik_i}^l = f_{i,k_i}\left(x_i^0\right) = e^{-\frac{\left(x_i^0 - c_{i,k_i}\right)^2}{\sigma_{i,k_i}^2}}, \qquad l = 1, \ldots, L. \tag{12}$$

The transfer function of each node R_l in hidden layer 3 (inference layer) is determined according to the operating method of the fuzzy implication relation selected. In this paper, the product operation is selected as the transfer function of node R_l; thus, the output of node R_l can be given by

$$R_l \prod_{i=1}^{N} P_{ik_i}^l. \tag{13}$$

The M outputs of system $y_j, j = 1, \ldots, M$, are composed of the M consequents of a fuzzy rule. Using weight v_{lj} between the inference layer and the output layer to denote the jth consequent of the lth rule, the initial v_{lj} is determined by the initial cluster center of sample data [6]. The output of the network may be represented with a general form given by

$$y_j = f(v_{lj}, R_l), \qquad j = 1, \ldots, M. \tag{14}$$

Summarizing the preceding discussion, if input x^0, singleton fuzzification, product inference mechanism, and center-average defuzzification are used, then the output of the network is

$$y_j^0 = \frac{\sum_{l=1}^{L} v_{lj} R_l}{\sum_{l=1}^{L} R_l} = \frac{\sum_{l=1}^{L} v_{lj} \prod_{i=1}^{N} P_{ik_i}^l(x^0)}{\sum_{l=1}^{L} \prod_{i=1}^{N} P_{ik_i}^l(x^0)}. \quad (15)$$

IV. Network Modifying Algorithm and Steps

A. Modifying Algorithm of Network Parameters

Obtaining network output y_j^0 from (15), the overall error of the network output can be calculated by performance index function (16)

$$E = \frac{1}{2} \sum_{j=1}^{M} \left(y_j^t - y_j^0 \right)^2. \quad (16)$$

The network parameters are modified via a gradient descent technique. The modifying algorithms of weights in each layer are derived as follows.

The expression of modifying weights between hidden layer 3 and the output layer is

$$\Delta v_{lj} = -\eta \frac{\partial E}{\partial v_{lj}} - \eta \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial v_{lj}} \quad (17)$$

where $0 < \eta < 1$ is the learning rate, and the partial derivative of E with respect to y_j, for a specific j (jth output), is

$$\frac{\partial E}{\partial y_j} = y_j^t - y_j. \quad (18)$$

Let

$$\delta_j = -\frac{\partial E}{\partial y_j}. \quad (19)$$

Because $y_j = (\sum_{l=1}^{L} R_l v_{lj}) / (\sum_{l=1}^{L} R_l)$, the following equation exists for a specific l:

$$\frac{\partial y_j}{\partial v_{lj}} = \frac{R_l}{\sum_{t=1}^{L} R_t}. \quad (20)$$

From (18)–(20), (17) becomes

$$\Delta v_{lj} = \eta \delta_j \frac{R_l}{\sum_{t=1}^{L} R_t}. \quad (21)$$

The outputs of the nodes in hidden layer 3 are computed by (13). All weights between hidden layer 2 and hidden layer 3 are 1 without modifying. Those nodes in hidden layer 2 are computed by (12).

The modifying formula of weights between the input layer and hidden layer 1 is derived as

$$\Delta c_{i,k_i} = -\eta \frac{\partial E}{\partial c_{i,k_i}} = -\eta \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial R_l} \frac{\partial R_l}{\partial P_{i,k_i}^l} \frac{\partial P_{i,k_i}^l}{\partial c_{i,k_i}}. \quad (22)$$

Each output y_j relates with all R_l. Hence,

$$\frac{\partial y_j}{\partial R_l} = \sum_{j=1}^{M} \frac{\partial}{\partial R_l} \left(\frac{\sum_{l=1}^{L} v_{lj} R_l}{\sum_{l=1}^{L} R_l} \right) = \sum_{j=1}^{M} \frac{\left(\sum_{l=1}^{L} v_{lj} R_l \right)'_{R_l} \left(\sum_{l=1}^{L} R_l \right) - \left(\sum_{l=1}^{L} v_{lj} R_l \right) \left(\sum_{l=1}^{L} R_l \right)'_{R_l}}{\left(\sum_{l=1}^{L} R_l \right)^2}.$$

For a specific l,

$$\frac{\partial y_j}{\partial R_l} = \sum_{j=1}^{M} \frac{v_{lj} - y_j}{\sum_{ll=1}^{L} R_{ll}} \quad (23)$$

$$\frac{\partial R_l}{\partial P_{i,k_i}^l} = \frac{\partial}{\partial P_{i,k_i}^l} \left(\prod_{i=1}^{N} P_{i,k_i}^l \right). \quad (24)$$

For a specific i, (24) is changed to

$$\frac{\partial R_l}{\partial P_{i,k_i}^l} = \prod_{\substack{ii=1 \\ ii \neq i}}^{N} P_{ii,k_i}^l \quad (25)$$

and for $\partial P_{i,k_i}^l / \partial c_{i,k_i}$ with specific i and k_i, there exists

$$\frac{\partial P_{i,k_i}^l}{\partial c_{i,k_i}} = \frac{\partial}{\partial c_{i,k_i}} \left(e^{-\frac{\left(x_i^0 - c_{i,k_i} \right)^2}{\sigma_{i,k_i}^2}} \right) = \frac{2 \left(x_i^0 - c_{i,k_i} \right)}{\sigma_{i,k_i}^2} P_{i,k_i}^l. \quad (26)$$

Considering (19), (23), (25), and (26), (22) is changed to

$$\begin{aligned}\Delta c_{i,k_i} &= \eta\delta_j \sum_{j=1}^{M} \frac{v_{lj}-y_j}{\sum_{ll=1}^{L} R_{ll}} \frac{2\left(x_i^0 - c_{i,k_i}\right)}{\sigma_{i,k_i}^2} \prod_{\substack{ii=1 \\ ii\neq i}}^{N} P_{ii,k_i}^l P_{i,k_i}^l \\ &= \eta\delta_l \frac{2(x_i^0 - c_{i,k_i})}{\sigma_{i,k_i}^2} \prod_{ii=1}^{N} P_{ii,k_i}^l \\ &= \eta\delta_l \frac{2\left(x_i^0 - c_{i,k_i}\right)}{\sigma_{i,k_i}^2} R_l \end{aligned} \tag{27}$$

where

$$\delta_l = \sum_{j=1}^{M} \delta_j \frac{v_{lj}-y_j}{\sum_{ll=1}^{L} R_{ll}}. \tag{28}$$

Similarly, the following equation can be derived:

$$\begin{aligned}\Delta\sigma_{i,k_i} &= -\eta \frac{\partial E}{\partial \sigma_{i,k_i}} \\ &= -\eta \sum_{j=1}^{M} \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial R_l} \frac{\partial R_l}{\partial P_{i,k_i}^l} \frac{\partial P_{i,k_i}^l}{\partial \sigma_{i,k_i}} \\ &= \eta\delta_l \frac{2\left(x_i^0 - c_{i,k_i}\right)^2}{\sigma_{i,k_i}^3} R_l. \end{aligned} \tag{29}$$

B. Modifying Steps of Network Parameters

According to Section III-A, the modifying steps of network parameters can be summarized in the following.

For Δv_{lj}, the modifying steps are given as follows:

Step 1) computing δ_j by (18) and (19);
Step 2) computing $\partial y_j/\partial v_{lj}$ by (20);
Step 3) computing Δv_{lj} by (21);
Step 4) updating v_{lj} by (30)

$$v_{lj}(t+1) = v_{lj}(t) + \Delta v_{lj}(t). \tag{30}$$

For $\Delta c_{i,k_i}$, the modifying steps are given as follows:

Step 1) computing δ_l by (28);
Step 2) computing $\Delta c_{i,k_i}$ by (27);
Step 3) updating c_{i,k_i} by (31)

$$c_{ik_i}(t+1) = c_{ik_i}(t) + \Delta c_{ik_i}(t). \tag{31}$$

For $\Delta\sigma_{i,k_i}$, the modifying steps can be obtained from (29), and to update σ_{i,k_i} by (32),

$$\sigma_{ik_i}(t+1) = \sigma_{ik_i}(t) + \Delta\sigma_{ik_i}(t). \tag{32}$$

It should be noticed that σ_{i,k_i} should satisfy (7).

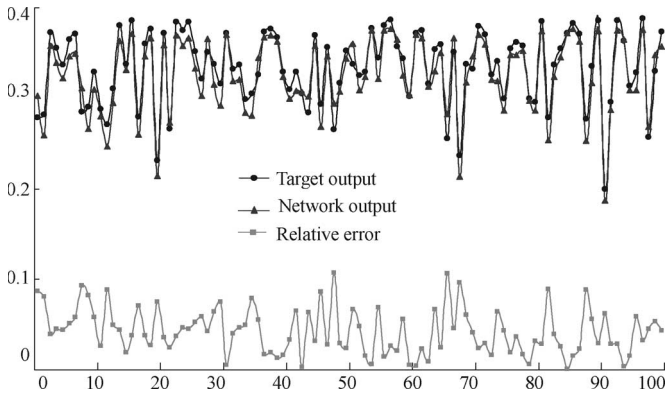

Fig. 7. Simulation curve using group 1 data.

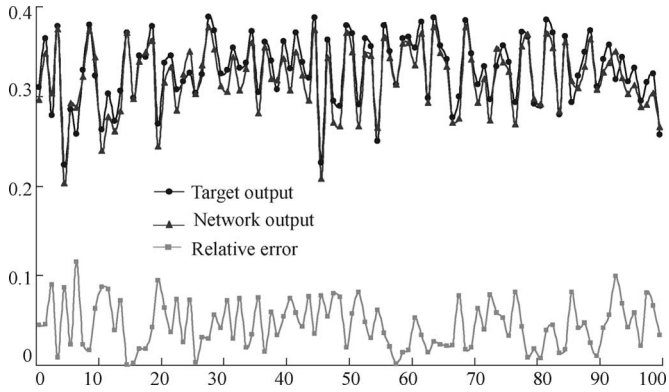

Fig. 8. Simulation curve using group 2 data.

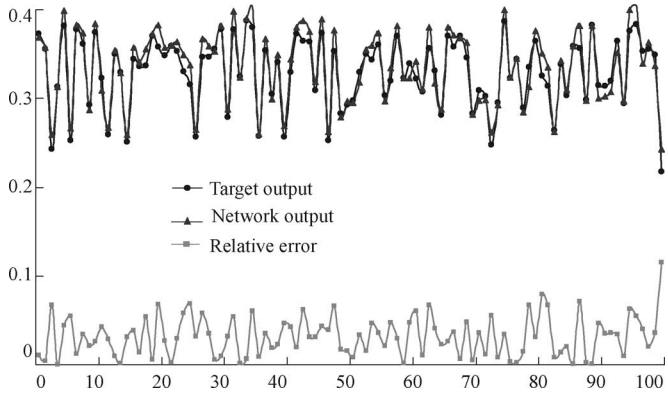

Fig. 9. Simulation curve using group 3 data.

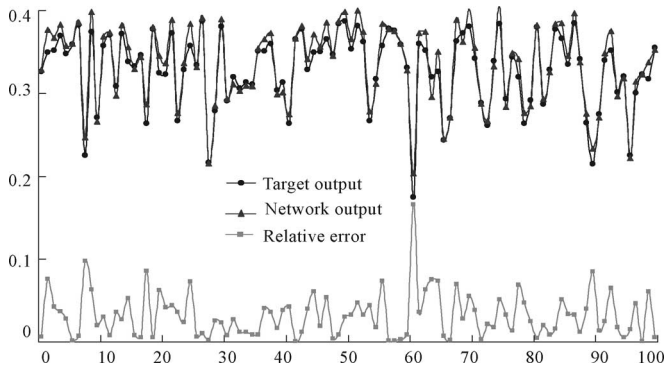

Fig. 10. Simulation curve using group 4 data.

TABLE I
SIMULATION RESULT FOR GROUP 1

j	x_{1_j}	x_{2_j}	y^0	y^t	relative error
1	0.301500	0.192900	0.278900	0.303062	0.086631
2	0.180000	0.453500	0.282000	0.259159	0.080998
3	0.716200	0.622600	0.372700	0.357951	0.039574
4	0.790300	0.647200	0.355700	0.339770	0.044786
5	0.331300	0.371000	0.337400	0.322333	0.044656
⋮					
99	0.712500	0.202400	0.330000	0.347702	0.053641
100	0.727000	0.393500	0.373300	0.357179	0.043185

Max. relative error: :0.106949
Min. relative error: 0.000518
Average relative error: 0.042900

TABLE II
SIMULATION RESULT FOR GROUP 2

j	x_{1_j}	x_{2_j}	y^0	y^t	relative error
1	0.265200	0.353000	0.310200	0.296334	0.044700
2	0.399000	0.451100	0.364500	0.347991	0.045292
3	0.303700	0.190300	0.278700	0.303651	0.089526
4	0.566800	0.367400	0.378300	0.374997	0.008732
5	0.214300	0.843900	0.223300	0.203894	0.086905
⋮					
99	0.918900	0.582500	0.325600	0.303892	0.066672
100	0.205000	0.254400	0.257500	0.266079	0.033315

Max. relative error: 0.115589
Min. relative error: 0.000846
Average relative error: 0.045410

TABLE III
SIMULATION RESULT FOR GROUP 3

j	x_{1_j}	x_{2_j}	y^0	y^t	relative error
1	0.716200	0.622600	0.372700	0.368850	0.010329
2	0.790300	0.647200	0.355700	0.357057	0.003816
3	0.200600	0.781100	0.242700	0.259044	0.067342
4	0.569700	0.857400	0.313300	0.313308	0.000024
5	0.545400	0.407100	0.382300	0.399063	0.043847
⋮					
99	0.444800	0.288500	0.349300	0.336842	0.035667
100	0.915300	0.919000	0.217800	0.242918	0.115328

Max. relative error: 0.115328
Min. relative error: 0.000024
Average relative error: 0.03181

TABLE IV
SIMULATION RESULT FOR GROUP 4

j	x_{1_j}	x_{2_j}	y^0	y^t	relative error
1	0.571400	0.826600	0.326200	0.328395	0.006729
2	0.477700	0.269500	0.349700	0.376374	0.076276
3	0.778500	0.320600	0.352100	0.367099	0.042597
4	0.540600	0.673000	0.369900	0.383493	0.036748
5	0.827900	0.361300	0.347900	0.357795	0.02844
⋮					
99	0.371000	0.762300	0.317800	0.337163	0.060929
100	0.834900	0.564300	0.354900	0.353001	0.005350

Max. relative error: 0.165802
Min. relative error: 0.000534
Average relative error: 0.031920

V. EXAMPLE AND CONCLUSION

In order to validate the validity of the algorithm, a simulation is made with the function

$$y = \sqrt{64 - 81((x_1 - 0.6)^2 + (x_2 - 0.5)^2)}/9 - 0.5.$$

Having training by 1000 samples, four groups of curves about the network output are shown in Figs. 7–10, and the corresponding data are given in Tables I–IV, in which x_1, x_2, y^t, and y^0 denote input 1, input 2, target output, and network output, respectively. Each group has 100 test points. The average relative error is less then 5%.

As shown in the figures and tables, the algorithm for extracting fuzzy rules based on RBFN is effective.

REFERENCES

[1] J. S. R. Jang and C. T. Sun, "Functional equivalence between radial basis functions and fuzzy inference systems," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 156–158, Jan. 1993.
[2] ——, "Neurofuzzy modeling and control," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 378–406, Mar. 1995.
[3] Q. Zhao and Z. Bao, "On the classification mechanism of a radial basis function network," *J. China Inst. Commun.*, vol. 17, no. 2, pp. 86–93, Mar. 1996.
[4] B.-T. Miao and F.-L. Chen, "Applications of radius basis function neural networks in scattered data interpolation," *J. China Univ. Sci. Technol.*, vol. 31, no. 2, pp. 135–142, Apr. 2001.
[5] Y. Jin and B. Sendhoff, "Extracting interpretable fuzzy rules from RBF networks," *Neural Process. Lett.*, vol. 17, no. 2, pp. 149–164, Apr. 2003.
[6] H. Sun and W. Li, "A method of selection initial cluster centers for cluster neural networks," *J. Syst. Simul.*, vol. 16, no. 4, pp. 775–777, Apr. 2004.
[7] T. Takagi and S. M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Jan./Feb. 1985.
[8] L. X. Wang and J. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, Nov./Dec. 1992.
[9] D. A. Linkens and M.-Y. Chen, "Input selection and partition validation for fuzzy modeling using neural network," *Fuzzy Sets Syst.*, vol. 107, no. 3, pp. 299–308, Nov. 1999.
[10] X. Lin, "Fast extracting fuzzy if–then rules based on RBF networks," *Syst. Eng.—Theory Methodology Appl.*, vol. 10, no. 2, pp. 145–149, 2001.

Wen Li received the B.S. degree in industry automation and the M.S. degree in railway traction electrization and automation from Dalian Jiaotong University, Dalian, China, in 1982 and 1992, respectively, and the Eng.D. degree in control theory and control engineering from Harbin Institute of Technology, Harbin, China, in 1999.

Since 1982, she has been with the Department of Electrical Engineering, Dalian Jiaotong University, where she has been a Professor since 2001. She was a Visiting Researcher with the University of Tokyo, Tokyo, Japan, from October 2004 to October 2005. Her research interests include intelligent control, fuzzy systems modeling, and control theory and its industrial application.

Prof. Li is a member of the China Railway Society and the China Electrotechnical Society.

Yoichi Hori (S'81–M'83–SM'00–F'05) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Tokyo, Tokyo, Japan, in 1978, 1980, and 1983, respectively.

In 1983, he joined the Department of Electrical Engineering, University of Tokyo, as a Research Associate and was later promoted to Assistant Professor, Associate Professor, and in 2000, Professor. In 2002, he joined the Institute of Industrial Science, University of Tokyo, as a Professor in the Information and Electronics Division (Electrical Control System Engineering). During 1991–1992, he was a Visiting Researcher with the University of California, Berkeley (UCB). His research interests include control theory and its industrial application to motion control, mechatronics, robotics, and electric vehicles.

Prof. Hori served as the Treasurer of the IEEE Japan Council and Tokyo Section during 2001–2002. He is currently the Vice President of the Institute of Electrical Engineers of Japan (IEE-Japan) Industry Applications Society. He is a member of IEE-Japan, the Japan Society of Mechanical Engineers, the Society of Instrument and Control Engineers, the Robotic Society of Japan, the Japan Society of Mechanical Engineers, and the Society of Automotive Engineers of Japan.