Function Approximation Based on Fuzzy Rules Extracted from Partitioned Numerical Data

Ruck Thawonmas, *Senior Member, IEEE,* and Shigeo Abe, *Senior Member, IEEE*

***Abstract*—We present an efficient method for extracting fuzzy rules directly from numerical input–output data for function approximation problems. First, we convert a given function approximation problem into a pattern classification problem. This is done by dividing the universe of discourse of the output variable into multiple intervals, each regarded as a class, and then by assigning a class to each of the training data according to the desired value of the output variable. Next, we partition the data of each class in the input space to achieve a higher accuracy in approximation of class regions. Partition terminates according to a given criterion to prevent excessive partition. For class region approximation, we discuss two different types of representations using hyperboxes and ellipsoidal regions, respectively. Based on a selected representation, we then extract fuzzy rules from the approximated class regions. For a given input datum, we convert, or in other words, defuzzify, the resulting vector of the class membership degrees into a single real value. This value represents the final result approximated by the method. We test the presented method on a synthetic nonlinear function approximation problem and a real-world problem in an application to a water purification plant. We also compare the presented method with a method based on neural networks.**

***Index Terms*—Ellipsoidal representation, function approximation, fuzzy rules, hyperboxes representation.**

I. Introduction

In theory, fuzzy system and neural network approaches have a potential for use in function approximation because both approaches have been shown to be universal approximators in [1] and [2], respectively. In practice, however, each approach holds both advantages and disadvantages. In the neural network approach, knowledge is automatically acquired by learning algorithms such as the backpropagation (BP) algorithm [3]. But, when the desired performance is not attained, it is difficult to analyze the trained networks so that they can be tuned. In addition, use of neural networks is prohibited for some applications due to computational expense in training the networks.

Fuzzy systems, in general, have more analysis tractability [4]. However, for constructing rules in fuzzy systems, knowledge acquisition is difficult if it is performed through interviewing experts. Automatically acquiring knowledge from numerical data for fuzzy systems has thus been the focus of much interest in recent years [5]–[9]. In [5] and [6], fuzzy rules are defined for multiple subspaces obtained by partitioning the input space. The methods proposed in these papers, therefore, have a large rule-acquisition (RA) time for applications with high-dimensional input space. In [7], fuzzy rules are defined for clusters of input–output data. Though this method can effectively handle applications with high-dimensional input space, its RA time still grows according to the order of the number of data used for extracting rules, i.e., the training data. In [8] and [9], fuzzy rules are defined for hyperboxes of input data. A hyperbox is represented by the minimum and maximum values of each input variable of the corresponding data. Due to a compact representation of the hyperboxes, the methods presented in [8] and [9] have an extremely low RA time, and have been shown to have promising performance when applied to pattern classification problems and function approximation problems, respectively. For the former type of problems, we have recently shown in [10] that the classification performance can be significantly improved by partitioning data prior to generation of hyperboxes.

In this paper, we introduce the partition concept in [10] to the method for function approximation in [9], with the aim of improving the approximation ability. In addition to the hyperbox representation, we discuss another type of representation that approximates class regions using ellipsoidal regions.

The method presented in this paper consists of four processes:

1) problem conversion;
2) data partition;
3) rule extraction;
4) defuzzification.

In 1), as was done in [9], a given function approximation problem is converted into a pattern classification problem by dividing the universe of discourse of the output variable into multiple intervals, each regarded as a class, and then by assigning a class to each of the training data according to the desired value of the output variable. In 2), before fuzzy rules are extracted in 3), the data of each class are partitioned for a higher accuracy in approximation of class regions. The basic idea behind this process is that partitioning of data prevents large hyperboxes or ellipsoidal regions from being defined which causes less accuracy in the approximation, due to under-fitting of the training data. For unknown data, however, the approximation ability may drop if excessive partition, which leads to over-fitting of the training data, is performed. The partition process is therefore governed in

Manuscript received April 15, 1996; revised April 19, 1997. This paper was recommended by Associate Editor A. Kandel.

R. Thawonmas is with the Department of Information Systems Engineering, Kochi University of Technology, Kochi 782-8502, Japan.

S. Abe is with the Department of Electrical and Electronics Engineering, Kobe University, Kobe 657-8501, Japan.

Publisher Item Identifier S 1083-4419(99)05390-X.

such a way that it terminates, according to a given criterion, before over-fitting occurs. In 3), either the hyperbox or ellipsoidal representation can be selected for approximating the partitioned class regions. Based on a selected representation, corresponding fuzzy rules are then extracted from the approximated regions. If the hyperbox representation is selected, the fuzzy rules and the inference mechanism discussed in [10] are used. Different fuzzy rules and the inference mechanism, proposed in this paper, are used for the ellipsoidal representation. In 4), as done in [9], for a given input datum, the resulting vector of the class membership degrees is converted into a single real value. The converted value is then used as the result approximated by the presented method.

This paper is organized as follows. Section II contains a brief description of the problem conversion and defuzzification processes. Section III describes the partitioning process and discusses how fuzzy rules are extracted based on either the hyperbox or ellipsoidal representation. Section IV shows the effectiveness of the presented method, compared with fuzzy systems based on rules extracted from nonpartitioned training data and with neural networks trained with the BP algorithm, using a synthetic nonlinear function approximation problem and an approximation problem for a water purification plant. Finally the results of these experiments are discussed.

II. Approximation and Prediction Problems as Classification Problems

In this section, we describe the two underlying processes discussed in [9] for treating a given function approximation problem as a classification problem.

A. Problem Conversion

Assume that we are given a set of N multiple-input, single-output training data points $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)$ of a function approximation problem. To convert the given problem into the corresponding classification problem, the range of the output is partitioned into K intervals, each regarded as a class, $[z_0, z_1], \cdots, [z_{K-1}, z_K]$ of the same width. Class i is assigned to $\boldsymbol{x}_j$ if $z_{i-1} < y_j \le z_i$. The resulting classified data are thereby given by a set $(\boldsymbol{x}_1, C_1), \cdots, (\boldsymbol{x}_N, C_N)$ where $C_i \in \{1, \cdots, K\}$. These data are then partitioned in the input space and the partitioned data are used for extracting fuzzy rules, according to the methods to be described in Section III. Note that, though considering only single-output problems, the method presented in this paper can be extended to multiple-output problems in a straightforward fashion.

B. Defuzzification

Assume now that we have a set of fuzzy rules extracted, based on a selected representation, using the training data. For a given unknown datum $\boldsymbol{x}$ in the input space, its output value $\hat{y}$ can be approximated by defuzzifying the resulting vector of the class membership degrees $\boldsymbol{d}(\boldsymbol{x}) = (d_1(\boldsymbol{x}), \cdots, d_K(\boldsymbol{x}))$, derived according to the fuzzy inference mechanisms to be described in Section III, as follows:

$$\hat{y} = \frac{\sum_{i=1}^{K} \mu_i \sigma_i d_i(\boldsymbol{x})}{\sum_{i=1}^{K} \sigma_i d_i(\boldsymbol{x})} \tag{1}$$

where μ_i and σ_i are, respectively, the center and the width of the output interval i.

To increase the approximation accuracy, if necessary, the values of μ_i and σ_i can be tuned using the steepest descent method. The tuning objective is to minimize the error measure E (e.g., the average absolute error $(1/K)\ \Sigma_{i=1}^{K} |y_i - \hat{y}_i|$) on the training data. Namely, after E of the training data is computed, μ_i and σ_i are then moved along the direction that minimizes E. Let one presentation of the training data be called one iteration. At iteration t, μ_i, and σ_i are therefore updated, respectively, by

$$\begin{aligned} \mu_i(t+1) &= \mu_i(t) - \alpha \frac{\partial E(t)}{\partial \mu_i(t)} \\ \sigma_i(t+1) &= \sigma_i(t) - \alpha \frac{\partial E(t)}{\partial \sigma_i(t)} \end{aligned} \tag{2}$$

where

$\mu_i(1)$	$= (z_{i-1} + z_i)/2;$
$\sigma_i(1)$	$= z_i - z_{i-1};$
α	tuning rate.

Note that the derivatives in (2) are easily obtained.

III. Fuzzy Rule Extraction

In this section, we discuss how to approximate class regions using the training data and how to extract fuzzy rules from the approximated class regions. For this, we consider two types of representations: hyperbox and ellipsoidal. The former uses a hyperbox to approximate a class region, while the latter uses an ellipsoidal region. In Section III-A, we first show how the training data are partitioned, before approximation of class regions, so as to improve the approximation ability. We then describe in Sections III-B and III-C the fuzzy rules and membership functions associated with the hyperbox and ellipsoidal representations, respectively.

A. Data Partition

We describe in the following a partition process for coping with the following problems. Namely, under-fitting of the training data occurs if we use a small number of hyperboxes (or ellipsoidal regions), each covering a relatively large portion in the input space. On the other hand, over-fitting of the training data takes place if we use a large number of hyperboxes (or ellipsoidal regions), each covering a relatively small portion in the input space. It is known that these problems cause degradation in the approximation ability of unknown data.

To prevent under-fitting for a given class, we first partition the data of this class into two groups. Suppose that each input variable is scaled so that its values lie in the range [0, 1]. We perform partition of these data based on their values in the input dimension that maximizes the difference between the

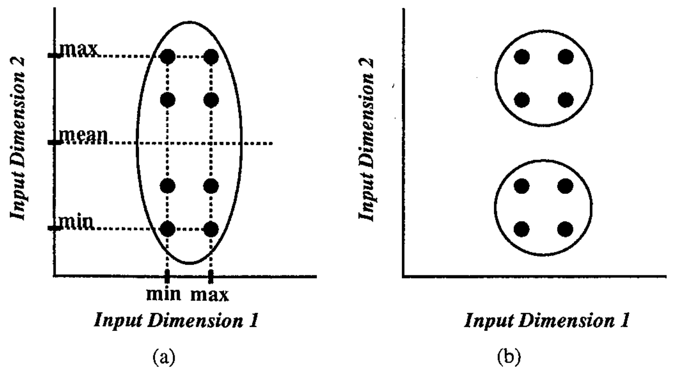

Fig. 1. Concept of partition in two input dimensions. Data of a given class (a) before partition and the data groups of the same class (b) after partition based on input dimension 2.

maximum and minimum values of the data. Though simple, this heuristic selects for partition the dimension whose data are apparently most sparse. In addition, it is also suited for the case where more complicated, second-order statistic information, such as the covariance of each input variable, does not work well due to lack of data. In the resulting dimension, we assign the data whose values are less than the mean value to one group and the remaining data to another group. We then recursively perform partition of these new data until a terminating criterion to be discussed below is met. Fig. 1 illustrates partition of data in the two-dimensional input space.

To prevent over-fitting, we do not perform additional partition for a data group that has the number of data less than or equal to a threshold value. The threshold value for a data group of class i is defined by $\beta \max(N_{\text{avg}}, |X_i|)$ where $\beta \leq 1$ is a ratio, called the terminating parameter, $N_{\text{avg}} = (1/K)\ \Sigma_{i=1}^{K}\ |X_i|$ is the average number of data for one class, X_i denotes the set of training input data of class i, and $|X|$ is the cardinality of X. We use this definition because a data group with a considerably smaller number of data, as compared to the total number of data in the corresponding class or the average number of data for one class, might contain no meaningful information. Due to the maximum operation in the threshold value, at large β, partitioning is not performed for data in the classes with a smaller number of data than the average number of data for one class. Partitioning of such data starts after β is decreased to a proper smaller value. Deriving the optimal value of the terminating parameter is of course *problem dependent*. In this paper, for a given problem, we approximately determine it by selecting the one that yields the minimum average absolute error using an error estimation technique called *random subsampling* [11] on the training data. Finally, we note here that in pattern classification, there exist many clustering algorithms [7], [12] that can be considered for partitioning of data. However, these algorithms not only need more extensive iterative computations, compared to our heuristic partition, but similarly they require some experimental error estimation techniques for determining the optimal number of clusters. As a result, we do not use them in our work.

Now, we formulate the above partitioning process for the pth data group of class i, G_i^p, as follows. Let x_m denote the mth element of M-dimensional input vector $\boldsymbol{x} \in G_i^p$, and v_{iim} and V_{iim} the minimum and maximum values of x_m, respectively. Initially, each class has one data group, hence $G_i^1 = X_i$. For a given terminating parameter β, the following steps are repeated for $i = 1, \cdots, K$.

1) Select G_i^p that satisfies
$$|G_i^p| > \beta \max(N_{\text{avg}}, |X_i|).$$
2) Determine m^* that satisfies
$$V_{iim^*} - v_{iim*} = \max(V_{iim} - v_{iim}) \quad \text{for } m = 1, \cdots, M.$$
3) Compute $\mu_{m^*}(G_i^p) = \text{mean}(x_{m*}) \quad \forall \boldsymbol{x} \in G_i^p$.
4) Divide G_i^p into G_i^l and G_i^r based on $\mu_{m^*}(G_i^p)$ as follows:
$$\begin{aligned} G_i^l &= \{\boldsymbol{x} | \boldsymbol{x} \in G_i^p \quad \text{and} \quad x_{m^*} < \mu_{m^*}(G_i^p)\} \\ G_i^r &= \{\boldsymbol{x} | \boldsymbol{x} \in G_i^p \quad \text{and} \quad x_{m^*} \geq \mu_{m^*}(G_i^p)\}. \end{aligned} \tag{3}$$
5) If there is no G_i^p that satisfies
$$|G_i^p| > \beta \max(N_{\text{avg}}, |X_i|)$$
go to Step 6); otherwise, return to Step 1).
6) Terminate partition process for class i.

B. Hyperbox Representation

Here, we approximate class regions of the partitioned data groups using only the minimum and maximum values of the corresponding data in each input variable. Previous works using this type of axis-parallel representation include [13] and [14]. The methods used in these papers take into account one training datum at a time for approximation. Their performance hence varies depending on the order of presenting the training data to the systems. In the method presented in this paper, such a problem does not occur because hyperboxes are defined after all training data are presented to the system. To increase the approximation accuracy, if hyperboxes of different classes overlap, new hyperboxes are defined for the data residing in the overlapping regions. Namely, hyperboxes are generated according to the following procedure.

1) Define an activation hyperbox for each partitioned data group of each class by finding the minimum and maximum values of the input data of the data group under consideration.
2) Resolve each overlap between activation hyperboxes of different classes by 1) defining the overlapping region as an inhibition hyperbox. Then 2) generate a new activation hyperbox in the inhibition hyperbox for either of the corresponding two classes whose data exist in that inhibition hyperbox. Carry out steps 1) and 2) recursively until the overlapping is resolved.

In the following, we describe the formulation of the hyperboxes (see Fig. 2), as well as the associated fuzzy rules and membership function.

Setting at this moment overlapping level l to 1, we define a hyperbox, called the activation hyperbox, $A_{ij'}^{rs'}(l)$ for the rth partitioned data group of class i, G_i^r, as
$$A_{ij'}^{rs'} = \{\boldsymbol{x} | v_{ij'm}^{rs'}(l) \leq x_m \leq V_{ij'm}^{rs'}(l), m = 1, \cdots, M\} \tag{4}$$

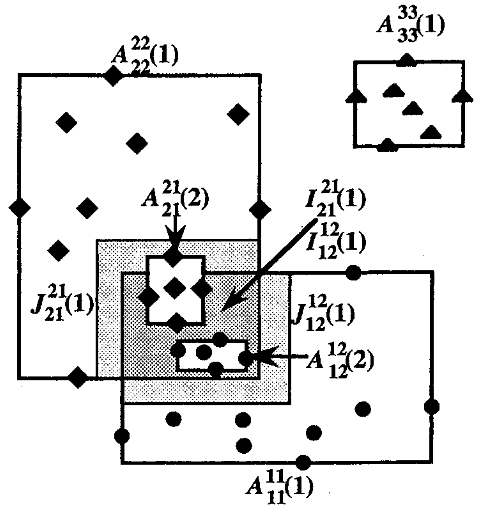

Fig. 2. Hyperbox representation for approximation of class regions in a two-dimensional three-class case.

where $s' = r, j' = i$ for $l = 1$; and $s' = s, j' = j$ for $l \geq 2$; x_m is the mth element of input vector $\boldsymbol{x}$; $\boldsymbol{x} \in G_i^r$ for $l = 1$; and $\boldsymbol{x} \in G_i^r$ and $\boldsymbol{x}$ is in $J_{ij}^{rs}(l-1)$, to be defined below, for $l \geq 2$; $v_{ij'm}^{rs'}(l)$ is the minimum value of x_m; $V_{ij'm}^{rs'}(l)$ is the maximum value of x_m.

If $A_{ij'}^{rs'}(l)$ and $A_{ji'}^{sr'}(l)$ overlap, we define an overlapping region, called the inhibition hyperbox, $I_{ij}^{rs}(l)$ as

$$I_{ij}^{rs} = \{\boldsymbol{x} | w_{ijm}^{rs}(l) \leq x_m \leq W_{ijm}^{rs}(l), m = 1, \cdots, M\} \quad (5)$$

where $v_{ij'm}^{rs'}(l) \leq w_{ijm}^{rs}(l) \leq W_{ijm}^{rs}(l) \leq V_{ij'm}^{rs'}(l)$. Note that $I_{ij}^{rs}(l) = I_{ji}^{sr}(l)$.

With this definition, data of classes i and/or j may reside on the same edges of $I_{ij}^{rs}(l)$, which degrades the accuracy in approximation of class regions of the training data. We therefore expand $I_{ij}^{rs}(l)$ to the expanded inhibition hyperbox $J_{ij}^{rs}(l)$ defined as

$$J_{ij}^{rs} = \{\boldsymbol{x} | u_{ijm}^{rs}(l) \leq x_m \leq U_{ijm}^{rs}(l), m = 1, \cdots, M\} \quad (6)$$

where $v_{ij'm}^{rs'}(l) \leq u_{ijm}^{rs}(l) \leq w_{ijm}^{rs}(l) \leq W_{ijm}^{rs}(l) \leq U_{ijm}^{rs}(l) \leq V_{ij'm}^{rs'}(l)$. The expanded inhibition hyperbox $J_{ji}^{sr}(l)$ can be defined in the same fashion. We use a parameter called the expansion parameter η to regulate expansion. In the case shown in Fig. 2, the expansion is performed for a pair of classes 1 and 2 according to the following definitions:

$$\begin{aligned} &\text{For } v_{ji'm}^{sr'}(l) \leq v_{ij'm}^{rs'}(l) \leq V_{ji'm}^{sr'}(l) \leq V_{ij'm}^{rs'}(l) \\ &\quad u_{ijm}^{rs}(l) = v_{ij'm}^{rs'}(l) \\ &\quad U_{ijm}^{rs}(l) = V_{ji'm}^{sr'}(l) + \eta(V_{ij'm}^{rs'}(l) - V_{ji'm}^{sr'}(l)). \quad (7) \\ &\text{For } v_{ij'm}^{rs'}(l) \leq v_{ji'm}^{sr'}(l) \leq V_{ij'm}^{rs'}(l) \leq V_{ji'm}^{sr'}(l) \\ &\quad u_{ijm}^{rs}(l) = v_{ji'm}^{sr'}(l) - \eta(v_{ji'm}^{sr'}(l) - v_{ij'm}^{rs'}(l)) \\ &\quad U_{ijm}^{rs}(l) = V_{ij'm}^{rs'}(l). \quad (8) \end{aligned}$$

Now, if there exist data of classes i and/or j in $J_{ij}^{rs}(l)$, we repeatedly define $A_{ij'}^{rs'}(l)$, $I_{ij}^{rs}(l)$, and $J_{ij}^{rs}(l)$, with l being incremented, using (4)–(6), respectively. We terminate when either there is no overlap between $A_{ij'}^{rs'}(l)$ and $A_{ji'}^{sr'}(l)$ or the condition $A_{ij'}^{rs'}(l) = A_{ji'}^{sr'}(l) = I_{ij}^{rs}(l-1)$ holds. In the latter condition, we terminate because the overlap cannot be resolved by recursively defining hyperboxes. In this case, for each datum residing in such $I_{ij}^{rs}(l-1)$, we define an activation hyperbox that includes only that datum.

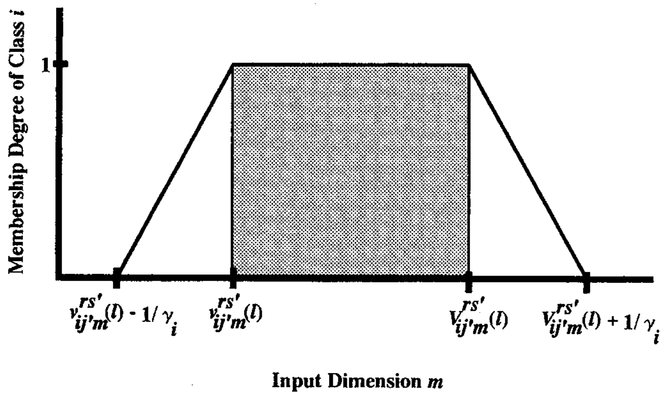

Fig. 3. One-dimensional membership function used in the hyperbox representation. Membership degrees in the shaded region, i.e., inside the corresponding activation hyperbox, are always 1. Membership degrees outside the shaded region are regulated by the trapezoidal sensitivity parameter.

Assume an unknown datum $\boldsymbol{x}$ is given. To label, i.e., classify, $\boldsymbol{x}$, we apply either of the following fuzzy rules $r_{ij'}^{rs'}(l)$ for each partitioned data group G_i^r

$$\text{If } \boldsymbol{x} \text{ is in } A_{ij'}^{rs'}(l), \text{ then } \boldsymbol{x} \text{ belongs to class } i. \quad (9)$$

$$\begin{aligned} &\text{If } \boldsymbol{x} \text{ is in } A_{ij'}^{rs'}(l) \text{ and } \boldsymbol{x} \text{ is not in } J_{ij}^{rs}(l) \\ &\quad \text{then } \boldsymbol{x} \text{ belongs to class } i. \quad (10) \end{aligned}$$

We use rule (9) when $J_{ij}^{rs}(l)$ does not exist, i.e., when there is no inhibition in the rule; otherwise, we use rule (10).

Next, we discuss the membership function associated with $A_{ij'}^{rs'}(l)$. If $\boldsymbol{x}$ is inside $A_{ij'}^{rs'}(l)$ in (9) or $\boldsymbol{x}$ is inside $A_{ij'}^{rs'}(l)$ but not inside $J_{ij}^{rs}(l)$ in (10), we assign the membership degree of $\boldsymbol{x}$ to 1. If $\boldsymbol{x}$ is outside this region, we assign the membership degree of $\boldsymbol{x}$ to a lower value. As the distance increases, the membership degree of $\boldsymbol{x}$ decreases and vice versa. To implement these characteristics, we use the following function:

$$m_{A_{ij'}^{rs'}(l)}(\boldsymbol{x}) = \min(m_{A_{ij'}^{rs'}(l)}(\boldsymbol{x}, m)) \quad (11)$$

and

$$\begin{aligned} &m_{A_{ij'}^{rs'}(l)}(\boldsymbol{x}, m) \\ &\quad = [1 - \max(0, \min(1, \gamma_i(v_{ij'm}^{rs'}(l) - x_m)))] \\ &\quad \times [1 - \max(0, \min(1, \gamma_i(x_m - V_{ij'm}^{rs'}(l))))] \quad (12) \end{aligned}$$

for $m = 1, \cdots, M$. In (12), the right term is of a trapezoidal shape and the parameter γ_i, called the trapezoidal sensitivity parameter of class i, is used for regulating the class membership degree (see Fig. 3).

Finally, we derive the membership degree of $\boldsymbol{x}$ with respect to a particular class through at most five levels of inference. We then use the resulting vector of the membership degrees for defuzzification according to the process described in Section II-B. We give the details of the fuzzy-rule-inference mechanism of each level in the Appendix.

C. Ellipsoidal Representation

Now, for approximation of class regions, we discuss another type of representation, the shape of which is not oriented to the axes of the input variables.

Based on the ellipsoidal representation, class regions of the partitioned data groups are approximated using ellipsoidal regions. Unlike hyperboxes, ellipsoidal regions are not actually defined. Here the regions are instead represented by contour surfaces of the membership degrees. To compute the membership degrees, we use the Gaussian membership function which is controlled by the mean vector and the covariance matrix of the corresponding data in the input space. Assuming that a high approximation accuracy is obtained[1] due to this form of axis-nonparallel representation when incorporated with partition, we can apply the following simple fuzzy rule for each partitioned data group G_i^r to classify an unknown datum $\boldsymbol{x}$

$$\text{If } \boldsymbol{x} \text{ is } \boldsymbol{c}_{ir}, \text{ then } \boldsymbol{x} \text{ belongs to class } i. \tag{13}$$

In (13), $\boldsymbol{c}_{ir} = (c_{ir1}, \cdots, c_{irM})^t$ represents the center of G_i^r and is calculated by

$$c_{irm} = \frac{1}{|G_i^r|} \sum_{\boldsymbol{x} \in G_i^r} x_m. \tag{14}$$

We then use the following functions to compute the membership degree of $\boldsymbol{x}$

$$d_i^r(\boldsymbol{x}) = \exp(-h_{ir}^2(\boldsymbol{x})) \tag{15}$$

$$h_{ir}^2(\boldsymbol{x}) = \frac{d_{ir}^2(\boldsymbol{x})}{\lambda_i} \tag{16}$$

$$d_{ir}^2(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{c}_{ir})^t \boldsymbol{Q}_{ir}^{-1} (\boldsymbol{x} - \boldsymbol{c}_{ir}) \tag{17}$$

where $d_{ir}^2(\boldsymbol{x})$ is the weighted square distance between $\boldsymbol{x}$ and $\boldsymbol{c}_{ir}$; $h_{ir}^2(\boldsymbol{x})$, the regulated square distance; $\lambda_i > 0$, the Gaussian sensitivity parameter of class i for regulating the membership degree; $\boldsymbol{Q}_{ir}$, the $M \times M$ covariance matrix of G_i^r; the superscript t denotes the transpose of a matrix, and the superscript -1, the inverse of a matrix. The covariance matrix $\boldsymbol{Q}_{ir}$ is calculated by

$$\boldsymbol{Q}_{ir} = \frac{1}{|G_i^r|} \sum_{\boldsymbol{x} \in G_i^r} (\boldsymbol{x} - \boldsymbol{c}_{ir})(\boldsymbol{x} - \boldsymbol{c}_{ir})^t. \tag{18}$$

Compared to the hyperbox representation, we derive the membership degree of $\boldsymbol{x}$ with respect to a particular class, say class i, $d_i(\boldsymbol{x})$ through a much simpler inference mechanism which is given below as

$$d_i(\boldsymbol{x}) = \max(d_i^r(\boldsymbol{x})) \quad \text{for } r = 1, \cdots, g_i \tag{19}$$

where g_i is the number of partitioned data groups of class i. We then use the resulting vector of the class membership degrees to approximate the output value of $\boldsymbol{x}$, according to the defuzzification process in Section II-B.

[1] See [15] for a tuning mechanism.

IV. Results

We illustrate the application of the presented method using two problems. First a problem of modeling a highly nonlinear function is used. Then an application of the presented method to a real-world problem is investigated. The second problem is an approximation problem from a water purification plant, discussed in [9]. For comparison, we use the average absolute (AA) error and the maximum absolute (MA) error as the performance measures. We compare our results with those of neural networks trained with the BP algorithm (NN-BP).

A. Synthetically Generated Data

We consider here the problem of modeling the highly nonlinear function

$$f(x, y, z) = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2 \tag{20}$$

where x, y, z are random variables uniformly generated in the range [1, 5]. After generation of 200 data, we use the first 100 data as training data and the last 100 as test data. To facilitate comparison with NN-BP, the output f of each datum is normalized to lie in the range [0, 1].

For the hyperbox representation (HR) and the ellipsoidal representation (ER), we use *random subsampling* to determine the number of intervals (i.e., classes) K, the tuning rate α, the number of tuning iterations, the value of β, the value of the trapezoidal sensitivity parameter γ_i, and the value of the Gaussian sensitivity parameter λ_i. Namely, for each representation, we choose the set of parameters that minimizes the AA error from a set of ten runs of *random subsampling*. In each run, two-thirds of the training data are randomly selected and used for extracting fuzzy rules. The extracted fuzzy rules are then tested on the remaining one-third. We summarize the chosen parameters in the following:

$$\begin{aligned} K &= 10, \quad \alpha = 0.01, 1000 \text{ iterations}, \beta = 1.0 \\ \eta &= 0.001, \quad \gamma_i = 1 \quad \text{for } i = 1, \cdots, K, \quad \text{and} \\ \lambda_i &= 10 \quad \text{for } i = 1, \cdots, K. \end{aligned}$$

The results from the NN-BP are the average value of ten runs of a network with three hidden neurons, learning rate = 0.1, training epochs = 1000, and initial connection weights randomly assigned between −0.1 and 0.1. Computer simulations of the NN-BP, HR, and ER are carried out on a 20 MIPS workstation.

Figs. 4 and 5 show the actual outputs as well as the approximated outputs from the NN-BP, HR, and ER for the training data and the test data, respectively. Differences in the approximated results from the three methods can be hardly seen in these figures. Table I gives performance comparisons of the HR and ER with the NN-BP. These results indicate that the approximation abilities of the three methods are comparable to each other for this synthetic problem.

B. Real-World Data from a Water Purification Plant

The task of this approximation problem is to calculate the desired amount of a particular coagulant to add as part of a treatment in improving water quality. The decision on the

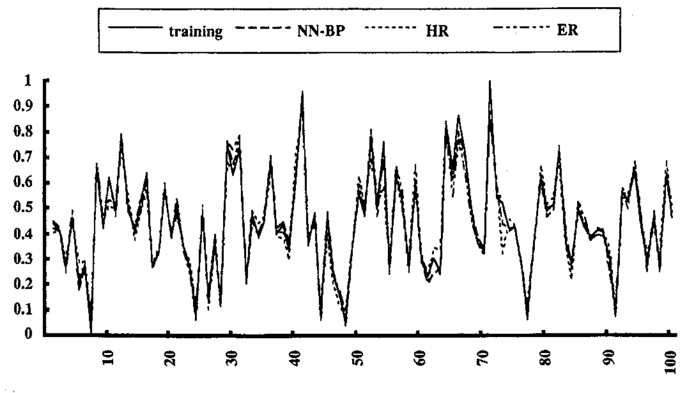

Fig. 4. Synthetic nonlinear function approximation problem. Comparison of the actual outputs on the training data with the approximated outputs by the NN-BP, HR, and ER.

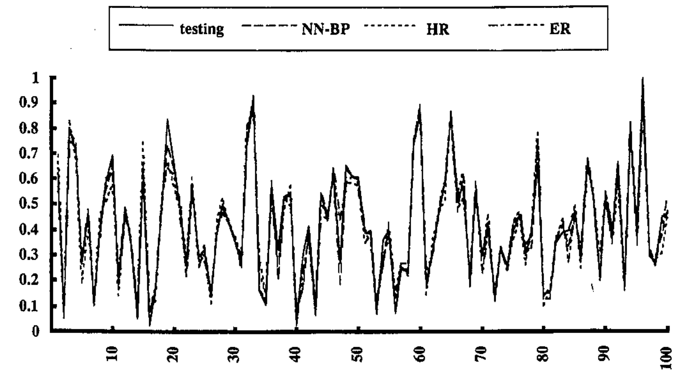

Fig. 5. Synthetic nonlinear function approximation problem. Comparison of the actual outputs on the test data with the approximated outputs by the NN-BP, HR, and ER.

TABLE I
THE SYNTHETIC NONLINEAR FUNCTION APPROXIMATION PROBLEM. APPROXIMATION ERRORS (MA - AA) FOR COMPARISON OF DIFFERENT METHODS

Methods	Training Data	Test Data
NN-BP	0.137 - 0.027	0.175 - 0.035
HR($\beta = 1.0$)	0.198 - 0.036	0.164 - 0.039
ER($\beta = 1.0$)	0.129 - 0.023	0.189 - 0.034

amount of the coagulant to add is based on a set of ten observed variables, each indicating a particular property of the water such as turbidity, water temperature, and so on. Two sets of training and test data, gathered over a one-year period, are available: stationary data set and nonstationary data set. The former consists of 241 training data and 237 test data for which the value of turbidity is less than a specific value, while the latter is composed of 45 training data and 40 test data having the turbidity larger than the specific value.

For each set of data, we use the number of intervals K, the tuning rate α, and the number of tuning iterations that were used in [9]. To obtain results for our method based on the hyperbox representation (HR), we set the expansion parameter η to 0.001 and use the value of the trapezoidal sensitivity parameter γ_i mentioned in [9], where it was called the sensitivity parameter, that gives the most

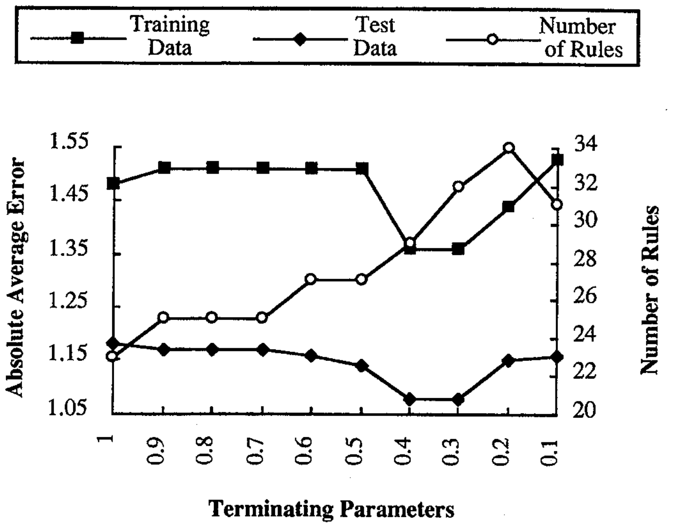

Fig. 6. Water purification plant approximation problem. Performance of the hyperbox representation applied to stationary data. The AA errors on the training data using random subsampling, the AA errors on the test data, and the numbers of rules are plotted for various values of the terminating parameter.

promising performance without partition of data. For our method based on the ellipsoidal representation (ER), we use the value of the Gaussian sensitivity parameter λ_i that gives us the minimum AA error from a set of ten runs of *random subsampling*. We perform the same procedure of *random subsampling* to find the optimal value of β. We summarize in the following the parameters that we use for the stationary data and nonstationary data, respectively. These parameters are used in computer simulations carried out on a 20 MIPS workstation.

Stationary Data:

$$K = 7, \quad \alpha = 0.01, 100 \text{ iterations}$$
$$\eta = 0.001, \quad \gamma_i = 4.0 \quad \text{for } i = 1, \cdots, K \quad \text{and}$$
$$\lambda_i = 3 \quad \text{for } i = 1, \cdots, K.$$

Nonstationary Data:

$$K = 5, \quad \alpha = 0.01, 100 \text{ iterations}$$
$$\eta = 0.001, \quad \gamma_i = 20.0 \quad \text{for } i = 1, \cdots, K \text{ and}$$
$$\lambda_i = 19 \quad \text{for } i = 1, \cdots, K.$$

1) Stationary Data: Figs. 6 and 7 plot the AA errors on the training data using *random subsampling*, the AA errors on the test data, and the numbers of rules against various values of β for the HR and ER, respectively. For the HR, the minimum value of the AA errors on the training data using *random subsampling*, i.e., 1.36, is obtained at $\beta = 0.4$. At $\beta = 1.0$, i.e., the nonpartition case, the AA error on the test data is 1.18, the number of fuzzy rules is 23, and the RA time is 5.53×10^{-1} s. As β is decreased, the AA error on the test data decreases while the number of fuzzy rules increases. The AA error on the test data hits the bottom of 1.08 at $\beta = 0.4$ where the number of fuzzy rules increases to 29 and the RA time to 7.17×10^{-1} s. From $\beta = 0.2$, the AA error on the test data increases.

For the ER, the minimum value of the AA errors on the training data using *random subsampling*, i.e., 1.29, is obtained

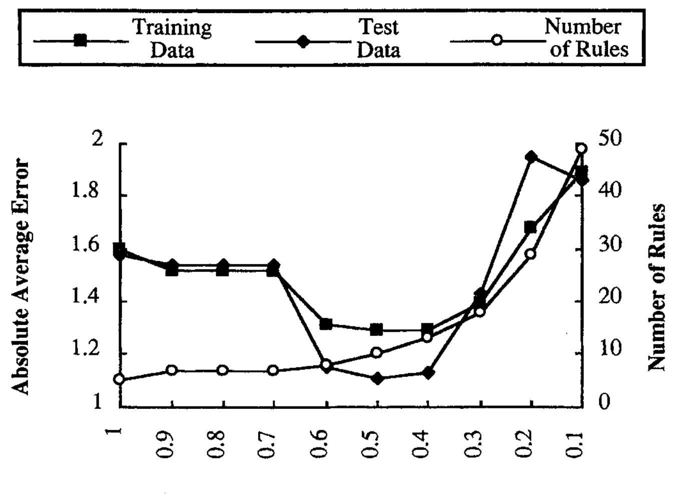

Fig. 7. Water purification plant approximation problem. Performance of the ellipsoidal representation applied to stationary data. The AA errors on the training data using random subsampling, the AA errors on the test data, and the numbers of rules are plotted for various values of the terminating parameter.

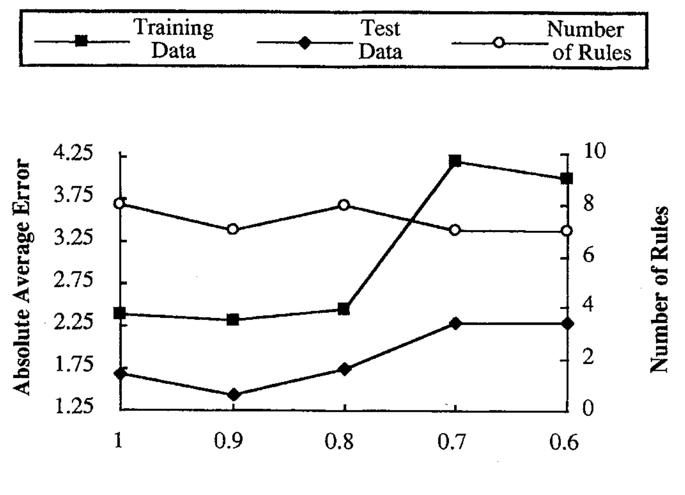

Fig. 8. Water purification plant approximation problem. Performance of the hyperbox representation applied to nonstationary data. The AA errors on the training data using random subsampling, the AA errors on the test data, and the numbers of rules are plotted for various values of the terminating parameter.

TABLE II
WATER PURIFICATION PLANT APPROXIMATION PROBLEM. STATIONARY DATA. APPROXIMATION ERRORS ON THE TEST DATA FOR COMPARISON OF DIFFERENT METHODS

Methods	Average Absolute Error	Maximum Absolute Error
NN-BP	0.99	6.95
HR($\beta = 0.4$)	1.08	5.85
ER($\beta = 0.5$)	1.11	5.38

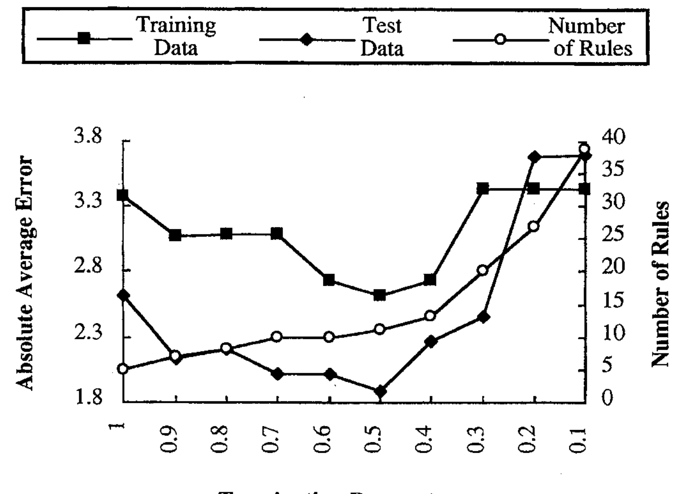

Fig. 9. Water purification plant approximation problem. Performance of the ellipsoidal representation applied to nonstationary data. The AA errors on the training data using random subsampling, the AA errors on the test data, and the numbers of rules are plotted for various values of the terminating parameter.

at $\beta = 0.5$. At $\beta = 1.0$, the AA error on the test data is 1.58, the number of fuzzy rules is five, and the RA time is 8.33×10^{-2} s. Similar to the HR, the number of fuzzy rules increases as β is decreased. At β in the interval [1.0, 0.5], the AA error on the test data decreases as β is decreased. The minimum AA error of 1.11 on the test data is obtained at $\beta = 0.5$ where the number of fuzzy rules increases to 10 and the RA time to 1.50×10^{-1} s. The AA error on the test data increases from $\beta = 0.4$.

Table II gives performance comparisons of the HR and ER with the NN-BP. In Table II errors of the NN-BP are from [9] at epochs = 2000, where the performance of the NN-BP is most promising. Though the NN-BP has the lowest AA error, its MA error is larger than those of the HR and ER. Compared to the ER, the HR has the lower AA error, but higher MA error. As a result, the performance of the NN-BP, HR, and ER, with respect to the approximation ability for unknown data, is comparable for this set of data. As for the time cost, the averaged training time of the NN-BP is one minute using a 31 MIPS mainframe computer [9]. Hence, the RA time of the HR and ER is much less than the training time of the NN-BP. And the RA time of the ER is less than that of the HR.

2) Nonstationary Data: Figs. 8 and 9 plot the AA errors on the training data using *random subsampling*, the AA errors on the test data, and the numbers of rules against various values of β for the HR and ER, respectively. For the HR, the minimum value of the AA errors on the training data using *random subsampling*, i.e., 2.08, is obtained at $\beta = 0.9$. At $\beta = 1.0$, the AA error on the test data is 1.68, the number of fuzzy rules is eight, and the RA time is 2.33×10^{-1} s. When partition is performed at $\beta = 0.9$, the AA error on the test data significantly decreases to 1.43, the number of fuzzy rules is seven, and the RA time is 2.50×10^{-1} s. From $\beta = 0.8$, the AA error on the test data increases.

For the ER, the minimum value of the AA errors on the training data using *random subsampling*, i.e., 2.61, is obtained at $\beta = 0.5$. At $\beta = 1.0$, the AA error on the test data is 2.61, the number of fuzzy rules is five, and the RA time is 8.33×10^{-2} s. As β is decreased, the AA error on the test data decreases at β in the interval [1.0, 0.5] while the number of fuzzy rules always increases. The minimum AA error of

TABLE III
WATER PURIFICATION PLANT APPROXIMATION PROBLEM. NONSTATIONARY DATA. APPROXIMATION ERRORS ON THE TEST DATA FOR COMPARISON OF DIFFERENT METHODS

Methods	Average Absolute Error	Maximum Absolute Error
NN-BP	1.74	6.78
HR($\beta = 0.9$)	1.43	4.81
ER($\beta = 0.5$)	1.88	5.85

1.88 on the test data is obtained at $\beta = 0.5$. Here, the number of fuzzy rules increases to ten and the RA time to 1.17×10^{-1} s. From $\beta = 0.4$, the AA error on the test data increases.

Table III gives performance comparisons of the HR and ER with the NN-BP. In Table III errors of the NN-BP are from [9] at epochs = 2000, where the performance of the NN-BP is most promising. The HR is superior to the other two regarding both the AA error and the MA error. Compared to the NN-BP, the ER has the higher AA error, but lower MA error.

C. Discussion

For the synthetically generated data of the highly nonlinear function in (20), the best performances of the HR and ER can be obtained without partitioning of the training data. However, the approximation ability of the presented method for unknown data in the water purification plant approximation problem can be greatly improved by partitioning of the training data, i.e., by resolving under-fitting of data. Namely, for the stationary data, the AA error on the test data of the HR (ER) is decreased from 1.18 (1.58) at $\beta = 1$, where partitioning of the training data is not performed, to 1.08 (1.11) when partitioning is done with $\beta = 0.4$ (0.5). For the nonstationary data, the AA error on the test data of the HR (ER) is reduced from 1.68 (2.61) at $\beta = 1$ to 1.43 (1.88) at $\beta = 0.9$ (0.5).

The value of β that minimizes the AA error on the training data using *random subsampling* represents well the point at which the partition process should be terminated to prevent over-fitting of data, due to excessive partition. Namely, in the water purification plant approximation problem, $\beta = 0.4$ and 0.5 when the HR and ER are applied to the stationary data, respectively. And in the nonstationary data case, $\beta = 0.9$ and 0.5 for the HR and ER, respectively. As can be seen, performing partition beyond this point not only leads to having more fuzzy rules but also causes an increase in the AA error on the test data, i.e., degradation in the approximation ability for unknown data. The same tendency is also found for the test data in the synthetic nonlinear function problem.

We note here that, since the RA time of the presented method based on both representations is extremely low, compared to other methods such as neural networks, use of *random subsampling* for finding the optimal parameters is feasible for a real-world problem. For a method based on neural networks to be able to use *random subsampling* to find the optimal network structure, much more powerful computational resources are needed due to the relatively large training time of the networks.

In the synthetic nonlinear function problem, the ER has better approximation ability on the training data, compared to that of the HR, while it has worse performance on the test data. In the water purification function approximation problem, the approximation ability on the test data of the ER is comparable to that of the HR for the stationary data while it is worse for the nonstationary data. From these results, we conjecture that the characteristic of the ER that it is not oriented to the axes of the input variables contributes to higher approximation ability on the training data, as shown in Table I. In order to achieve high approximation ability on unknown data (test data), the ER needs a larger number of training data, compared to the HR. This is because it uses more complicated information, i.e., covariance matrixes, from the training data to approximate class regions. However, the RA time of the ER is much lower than that of the HR. To achieve a high approximation ability on unknown data of a given application, we suggest use of the HR if there are only a small number of training data, as in the case of the synthetically generated data in the nonlinear function approximation problem or the nonstationary data in the water purification plant approximation problem; otherwise, the ER should be used. In practice, a decision on selection of the proper representation can be done based on the AA error on the training data using *random subsampling*. Namely, the selected representation is the one that has a lower value for the minimum of errors over various β. For example, for the stationary data in the water purification plant approximation problem, we select the ER because its minimum error is 1.29, which is lower than that of the HR, i.e., 1.36. On the other hand, we select the HR for the nonstationary data because the minimum error of the HR is 2.08 while that of the ER is 2.61.

V. CONCLUSIONS

In this paper, we discussed a function approximation method based on fuzzy rules extracted from partitioned numerical data. Our previous paper had shown that partition of the training data prior to extraction of fuzzy rules can improve the performance in classification of unknown data. We applied the concept of partition to function approximation in this paper, with the aim to improve the approximation ability of the output of given unknown data. We presented two representations, the hyperbox and ellipsoidal representations. Fuzzy rules were then extracted based on these representations. We showed the results of the presented method based on the two representations for a synthetic nonlinear function approximation problem and an application to a water purification plant. We demonstrated that the approximation ability can be significantly improved with partition, especially for the latter real-word problem. We compared our results with those of a method based on neural networks. We also gave a practical guide for selecting a more proper representation for a given function approximation problem.

APPENDIX

Based on the hyperbox representation, the membership degree of a given unknown x with respect to a particular class, say class i, is derived through at most five levels of inference.

First Inference Level: First, at this level, the membership degrees with respect to each activation hyperbox and each expanded inhibition hyperbox are derived. The membership degree with respect to $A^{rs'}_{ij'}(l)$, $m_{A^{rs'}_{ij'}(l)}(\boldsymbol{x})$, is already given in (11). We define the membership degree with respect to $J^{rs}_{ij}(l)$, $m_{J^{rs}_{ij}(l)}(\boldsymbol{x})$, as

$$m_{J^{rs}_{ij}(l)}(\boldsymbol{x}) = \max(m_{J^{rs}_{ij}(l)}(\boldsymbol{x}, m)) \qquad \text{for } m = 1, \cdots, M \tag{21}$$

where the definition of $m_{J^{rs}_{ij}(l)}(\boldsymbol{x}, m)$ has a form similar to (12) [8].

Second Inference Level: Next, the membership degrees with respect to fuzzy rules (9) and (10) are derived at this level. We define the membership degree of $\boldsymbol{x}$ with respect to the fuzzy rule $r^{rs'}_{ij'}(l)$ given by (9), $d_{r^{rs'}_{ij'}(l)}(\boldsymbol{x})$, as

$$d_{r^{rs'}_{ij'}(l)}(\boldsymbol{x}) = m_{A^{rs'}_{ij'}(l)}(\boldsymbol{x}). \tag{22}$$

From (10), if $\boldsymbol{x}$ is outside the activation hyperbox or $\boldsymbol{x}$ is inside the expanded inhibition hyperbox, $\boldsymbol{x}$ has a membership degree lower than 1. Its value lies on a contour surface of membership degrees that is parallel to, and lies at, an equal distance from the surface of $A^{rs'}_{ij'}(l) - J^{rs}_{ij}(l)$. To realize this, we calculate the membership degree with respect to $r^{rs'}_{ij'}(l)$ given by (10) according to whether or not $\boldsymbol{x}$ is included in the region $H^{rs}_{ij}(l)$, to be defined below. We define this membership, $d_{r^{rs'}_{ij'}(l)}(\boldsymbol{x})$, as

$$\begin{aligned} d_{r^{rs'}_{ij'}(l)}(\boldsymbol{x}) &= m_{A^{rs'}_{ij'}(l)}(\boldsymbol{x}) \\ &\qquad \text{for } \boldsymbol{x} \notin H^{rs}_{ij}(l) \\ &= \min(m_{A^{rs'}_{ij'}(l)}(\boldsymbol{x}), m_{J^{rs}_{ij}(l)}(\boldsymbol{x})) \\ &\qquad \text{for } \boldsymbol{x} \in H^{rs}_{ij}(l). \end{aligned} \tag{23}$$

In (23), $H^{rs}_{ij}(l)$ is associated with $A^{rs'}_{ij'}(l)$ as well as $J^{rs}_{ij}(l)$, and it defines an input region where the expanded inhibition hyperbox has an effect on the membership degree with respect to the rule given by (10). We define $H^{rs}_{ij}(l)$ as

$$\begin{aligned} H^{rs}_{ij}(l) = \{\boldsymbol{x} | \\ & x_m \le U^{rs}_{ijm}(l) \\ &\quad \text{for } v^{sr'}_{ji'm}(l) \le v^{rs'}_{ij'm}(l) \le V^{sr'}_{ji'm}(l) \le V^{rs'}_{ij'm}(l) \\ & x_m \ge u^{rs}_{ijm}(l) \\ &\quad \text{for } v^{rs'}_{ij'm}(l) \le v^{sr'}_{ji'm}(l) \le V^{rs'}_{ij'm}(l) \le V^{sr'}_{ji'm}(l) \\ & -\infty \le x_m \le \infty \\ &\quad \text{for } v^{sr'}_{ji'm}(l) \le v^{rs'}_{ij'm}(l) \le V^{rs'}_{ij'm}(l) \le V^{sr'}_{ji'm}(l), \\ & u^{rs}_{ijm}(l) \le x_m \le U^{rs}_{ijm}(l) \\ &\quad \text{for } v^{rs'}_{ij'm}(l) \le v^{sr'}_{ji'm}(l) \le V^{sr'}_{ji'm}(l) \le V^{rs'}_{ij'm}(l) \\ &\qquad m = 1, \cdots, M\} \end{aligned} \tag{24}$$

where $H^{rs}_{ij}(l)$ and $H^{sr}_{ji}(l)$ are in general different.

Third Inference Level: The membership degrees calculated at the second inference level are used here to derive the membership degrees with respect to aggregation of fuzzy rules (9) and (10) for all overlapping levels. We define the membership degree of $\boldsymbol{x}$ with respect to a set of fuzzy rules $\{r^{rs'}_{ij'}(l) | l = 1, \cdots, l^{rs}_{ij}\}$, $d_{r^{rs}_{ij}}(\boldsymbol{x})$, as

$$d_{r^{rs}_{ij}}(\boldsymbol{x}) = \max(d_{r^{rs'}_{ij'}(l)}(\boldsymbol{x})) \tag{25}$$

where l^{rs}_{ij} is the deepest level of the rules involved in both $A^{rr}_{ii}(1)$ and $A^{ss}_{jj}(1)$.

In (25), we take the maximum because each fuzzy rule in $\{r^{rs'}_{ij'}(l) | l = 1, \cdots, l^{rs}_{ij}\}$ is exclusive of any others due to inclusion of $A^{rs}_{ij}(l+1)$ in $J^{rs}_{ij}(l)$.

Fourth Inference Level: Here the membership degree with respect to each partitioned data group is derived. We define the membership degree of $\boldsymbol{x}$ with respect to the fuzzy rules related to G^r_i (i.e., $A^{rr}_{ii}(1)$), $d^r_i(\boldsymbol{x})$, as

$$d^r_i(\boldsymbol{x}) = \min(d_{r^{rs}_{ij}}(\boldsymbol{x})) \tag{26}$$

where $j \ne i$; $j = 1, \cdots, K$; $s = 1, \cdots, g_j$; $A^{rr}_{ii}(1) \cap A^{ss}_{jj}(1) \ne \emptyset$; and g_i is the number of partitioned data groups of class i.

When $A^{rr}_{ii}(1)$ overlaps with $A^{ss}_{jj}(1)$ and $A^{tt}_{kk}(1)$, we resolve the conflict, independently, first between $A^{rr}_{ii}(1)$ and $A^{ss}_{jj}(1)$ for all s satisfying $A^{rr}_{ii}(1) \cap A^{ss}_{jj}(1) \ne \emptyset$, then between $A^{rr}_{ii}(1)$ and $A^{tt}_{kk}(1)$ for all t satisfying $A^{rr}_{ii}(1) \cap A^{tt}_{kk}(1) \ne \emptyset$. We thereby implement this by taking the minimum in (26).

Fifth Inference Level: Finally, the membership degree with respect to each class is derived. We define the membership degree of $\boldsymbol{x}$ with respect to class i, $d_i(\boldsymbol{x})$, as

$$d_i(\boldsymbol{x}) = \max(d^r_i(\boldsymbol{x})) \qquad \text{for } r = 1, \cdots, g_i. \tag{27}$$

In (27), we take the maximum because each $A^{rr}_{ii}(l)$, where $r = 1, \cdots, g_i$, is exclusive of any others, and so is rule $d^r_i(\boldsymbol{x})$.

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for their helpful comments.

REFERENCES

[1] L.-X. Wang, "Fuzzy systems are universal approximators," in *Proc. IEEE Int. Conf. Fuzzy Systems*, San Diego, CA, 1992, pp. 1163–1170.

[2] K. Funahashi, "On the approximate realization of continuous mapping by neural networks," *Neural Networks*, vol. 2, no. 3, pp. 183–192, 1989.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, vol. 1, pp. 318–362.

[4] R. Thawonmas and S. Abe, "A novel approach to feature selection based on analysis of class regions," *IEEE Trans. Syst., Man, Cybern. B*, vol. 27, pp. 196–207, Apr. 1997.

[5] L.-X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1414–1427, Nov./Dec. 1992.

[6] J.-S. R. Jang "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, May/June 1993.

[7] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.

[8] S. Abe and M.-S. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 18–28, 1995.

[9] ——, "Fuzzy rules extraction directly from numerical data for function approximation," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 119–129, Jan. 1995.

[10] R. Thawonmas and S. Abe, "Extraction of fuzzy rules for classification based on partitioned hyperboxes," *J. Intell. Fuzzy Syst.*, vol. 4, no. 3, pp. 215–226, 1996.

[11] S. M. Weiss and C. A. Kulikowski, *Computer Systems That Learn.* San Mateo, CA: Morgan Kaufmann, 1991, pp. 30–36.
[12] J. C. Bezdek, "A review of probabilistic, fuzzy, and neural models for pattern recognition," *J. Intell. Fuzzy Syst.*, vol. 1, no. 1, pp. 1–25, 1993.
[13] S. Salzberg, "A nearest hyperrectangle learning method," *Mach. Learn.*, vol. 6, pp. 251–276, 1991.
[14] P. K. Simpson, "Fuzzy min-max neural networks—Part 1: Classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 776–786, 1992.
[15] S. Abe and R. Thawonmas, "A fuzzy classifier with ellipsoidal regions," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 358–368, 1997.

Ruck Thawonmas (M'97–SM'99) received the B.Eng. degree in electrical engineering from Chulalongkorn University, Thailand, in 1987, the M.Eng. degree in information science from Ibaraki University, Japan, in 1990, and the Dr.Eng. degree in information engineering from Tohoku University, Japan, in 1994.

Since April 1999, he has been Associate Professor, Department of Information Systems Engineering, Kochi University of Technology (Kochi-Tech), Japan. From August 1998 to March 1999 he was BSI Researcher at the Brain-Style Information Systems Research Group, the Brain Science Institute, RIKEN. From October 1997 to July 1998, he was Assistant Professor with the Department of Computer Hardware, University of Aizu. From April 1996 to September 1997, he was a Special Postdoctoral Researcher at the Brain Information Processing Group, Frontier Research Program, Institute of Physical and Chemical Research (RIKEN). From January 1994 to March 1996, he was Visiting (HIVIPS) Researcher at Hitachi Research Laboratory, Hitachi, Ltd. His research interests are data analysis using soft computing techniques such as neural networks and fuzzy logic (soft data analysis).

Dr. Thawonmas was a recipient of the Japanese Government (Monbusho) Scholarship and received a grant from the Tohoku Kaihatsu Memorial Foundation to be a Research Fellow at Tohoku University.

Shigeo Abe (M'79–SM'83) received the B.S. degree in electronics engineering, the M.S. degree in electrical engineering, and the Dr.Eng. degree, all from Kyoto University, Kyoto, Japan, in 1970, 1972, and 1984, respectively.

He is a Professor of Department of Electrical and Electronics Engineering, Kobe University, Kobe, Japan. His research interests are system modeling, system control, and pattern recognition using neural networks and fuzzy systems. From 1972 to 1997, he was with Hitachi Research Laboratory, Hitachi, Ltd., and was engaged in power system analysis, development of a vector processor, a Prolog processor, neural network theories, and fuzzy system models. From 1978 to 1979, he was a Visiting Research Associate at the University of Texas, Arlington. He is the author of *Neural Networks and Fuzzy Systems: Theory and Applications* (Norwell, MA: Kluwer).

Dr. Abe was awarded an outstanding paper prize from the Institute of Electrical Engineers of Japan in 1984 and 1995. He is a member of The International Neural Network Society, The Institute of Electrical Engineers of Japan, The Information Processing Society of Japan, The Institute of Electronics, Information and Communication Engineers of Japan, The Society of Instrument and Control Engineers of Japan, and The National Geographic Society.