Fuzzy Rules Extraction Directly from Numerical Data for Function Approximation

Shigeo Abe, *Senior Member, IEEE*, and Ming-Shong Lan

Abstract— **In our previous work we developed a method for extracting fuzzy rules directly from numerical input-output data for pattern classification. In this paper we extend the method to function approximation. For function approximation, first, the universe of discourse of an output variable is divided into multiple intervals, and each interval is treated as a class. Then the same as for pattern classification, using the input data for each interval, fuzzy rules are recursively defined by activation hyperboxes which show the existence region of the data for the interval and inhibition hyperboxes which inhibit the existence region of data for that interval. The approximation accuracy of the fuzzy system derived by this method is empirically studied using an operation learning application of a water purification plant. Additionally, we compare the approximation performance of the fuzzy system with the function approximation approach based on neural networks.**

I. Introduction

FUZZY systems and neural networks have recently been proposed for function approximation [1]–[3]. When comparing these two technologies, fuzzy systems are more favorable in that their behavior can be explained based on fuzzy rules and thus their performance can be adjusted by tuning the rules. But since, in general, knowledge acquisition is difficult and also the universe of discourse of each input variable needs to be divided into several intervals, applications of fuzzy systems are restricted to the fields where expert knowledge is available and the number of input variables is small.

To overcome the problem of knowledge acquisition, several methods for extracting fuzzy rules from numerical data have been developed. But most methods assume the divisions of input variables are fixed regions [4], [5]. In [6], fuzzy rules with variable fuzzy regions (hyperboxes) are extracted for classification problems. This approach has a potential applicability to problems having a high-dimensional input space. But because the overlap of hyperboxes of different classes must be resolved by dynamically expanding, splitting and contracting hyperboxes, the approach is difficult to apply to the problems in which several classes overlap.

In [7], we discussed a method for extracting fuzzy rules for pattern classification. The fuzzy rules with variable fuzzy regions were defined by activation hyperboxes which show the existence region of data for a class and inhibition hyperboxes which inhibit the existence of the data for that class. These rules were extracted directly from numerical data by recursively resolving overlaps between two classes. Performance comparison with neural networks for a license plate recognition system showed that training time of our method was negligible compared with that of neural networks, while the recognition rates were almost the same.

In this paper we extend our method to function approximation. First, we divide the range of an output variable into multiple intervals and using the input data belonging to each interval we define fuzzy rules recursively in the same manner as we discussed previously [7]. Each rule is composed of an activation hyperbox which defines the existence region of an interval and, if necessary, an inhibition hyperbox which inhibits the existence of data in that activation hyperbox. Then, we discuss their inference and defuzzification mechanisms. Furthermore, we discuss how to delete redundant input variables based on the number of fuzzy rules created. Finally, we apply our method to an operation learning application of a water purification plant by which generalization ability and training time are compared with that of neural networks.

II. Function Approximation by Fuzzy Rules With Variable Fuzzy Regions

A. Fuzzy System Architecture

Since the extension of the method to multiple-output problems is straightforward, here we consider approximating functions which have a one-dimensional output y and an m-dimensional input vector $\mathbf{x}$. First we divide the universe of discourse of y into n intervals as follows:

$$\begin{aligned} &[y_0, y_1]\colon\ y_0 \le y \le y_1 \\ &(y_1, y_2]\colon\ y_1 < y \le y_2 \\ &\quad\cdots \\ &(y_{n-1}, y_n]\colon\ y_{n-1} < y \le y_n. \end{aligned} \tag{1}$$

We call the i-th interval the output interval i. Using the input data whose outputs are in the output interval i, we recursively define the input region that generates output in the output interval i. Namely, first we determine activation hyperboxes, which define the input region corresponding to the output interval i, by calculating the minimum and maximum values of input data for each output interval. If the activation hyperbox for the output interval i overlaps with the activation hyperbox for the output interval j, the overlapped region is defined as an inhibition hyperbox. If the input data for output intervals i or/and j exist in the inhibition hyperbox, within

Manuscript received April 16, 1993; revised February 12, 1994.

S. Abe is with Hitachi Research Laboratory, Hitachi, Ltd., 7-1-1 Omika, Hitachi, Japan.

M.-S. Lan is with the Science Center, Rockwell International Corporation, Thousand Oaks, CA 91358 USA.

IEEE Log Number 9404975.

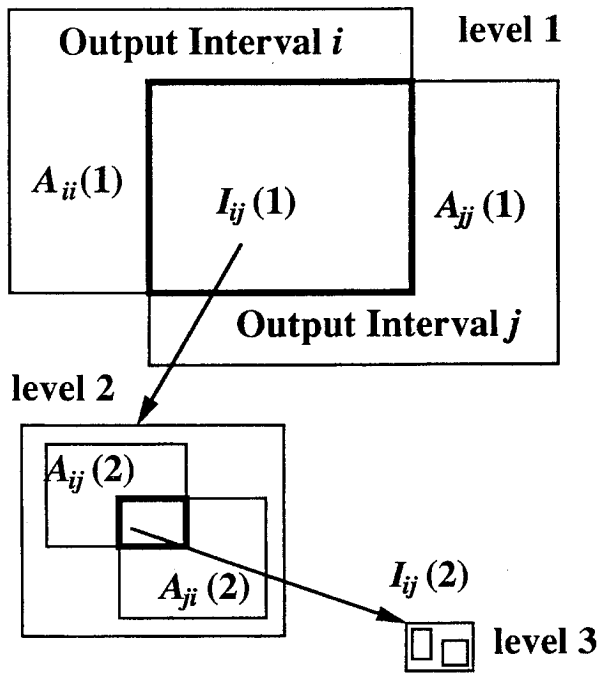

Fig. 1. Recursive definition of activation and inhibition hyperboxes.

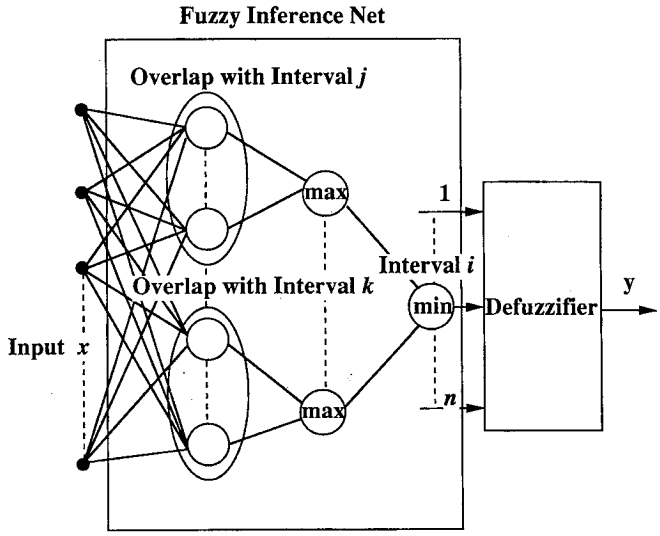

Fig. 2. Architecture of a fuzzy system.

this inhibition hyperbox, we define one or two additional activation hyperboxes; moreover, if two activation hyperboxes are defined and they overlap, we further define an additional inhibition hyperbox: this process is repeated until the overlap is resolved. Fig. 1 illustrates this process schematically.

Based on an activation hyperbox or based on an activation hyperbox and its corresponding inhibition hyperbox (if generated), a fuzzy rule is defined. Fig. 2 shows a fuzzy system architecture, including a fuzzy inference net which calculates degrees of membership for output intervals and a defuzzifier. For an input vector $\mathbf{x}$, degrees of membership for output intervals 1 to n are calculated in the inference net and then the output y is calculated by the defuzzifier using the degrees of membership as inputs.

The fuzzy inference net consists of four layers at most. The inference net is sparsely connected. Namely, different output intervals have different units for the second to fourth layers and there is no connection among units of different output intervals. The second-layer units consist of fuzzy rules which calculate the degrees of membership for an input vector $\mathbf{x}$. The third-layer units take the maximum values of inputs from the second layer, which are the degrees of membership generated by resolving overlaps between two output intervals. The number of third-layer units for the output interval i is determined by the number of output intervals whose input spaces overlap with that of the output interval i. Therefore, if there is no overlap between the input space of the output interval i and that of any other output intervals, the network for the output interval i is reduced to two layers. The fourth-layer unit for the output interval i takes the minimum value among the maximum values generated by the preceding layer, each of them is associated with an overlap between two output intervals. Therefore, if the output interval i overlaps with only one output interval, the network for the output interval i is reduced to three layers. Calculation of a minimum in the fourth layer resolves overlaps among more than two output intervals. Thus in the process of generating hyperboxes, we need to resolve only an overlap between two output intervals at a time.

B. Fuzzy Rule Extraction

Let a set of input data for the output interval i be X_i, where $i = 1, \ldots, n$. First, using X_i, an activation hyperbox of level 1, denoted as $A_{ii}(1)$, is defined as follows:

$$A_{ii}(1) = \{\mathbf{x} | v_{iik}(1) \leq x_k \leq V_{iik}(1), k = 1, \ldots, m\}, \quad (2)$$

where x_k: the k-th element of input vector $\mathbf{x}$;

$v_{iik}(1)$: the minimum value of x_k of $\mathbf{x} \in X_i$; and
$V_{iik}(1)$: the maximum value of x_k of $\mathbf{x} \in X_i$.

If there is no overlap between activation hyperboxes $A_{ii}(1)$ and $A_{jj}(1)(j \neq i, j = 1, \ldots, n)$, we obtain a fuzzy rule of level 1 for the output interval i as follows:

$$\text{If } \mathbf{x} \text{ is } A_{ii}(1) \text{ then } \mathbf{x} \text{ is in output interval } i. \quad (3)$$

If there are some overlaps, we resolve them recursively. If an overlap exists between the activation hyperboxes $A_{ii}(1)$ and $A_{jj}(1)$, we define the overlapped region as the inhibition hyperbox of level 1 denoted as $I_{ij}(1)$:

$$I_{ij}(1) = \{\mathbf{x} \mid w_{ijk}(1) \leq x_k \leq \mathbf{W}_{ijk}(1), k = 1, \ldots, m\} \quad (4)$$

where $v_{iik}(1) \leq w_{ijk}(1) \leq W_{ijk}(1) \leq V_{iik}(1)$. Then we define a fuzzy rule of level 1 with inhibition by

$$\text{If } \mathbf{x} \text{ is } A_{ii}(1) \text{ and } \mathbf{x} \text{ is not } I_{ij}(1)$$
$$\text{then } \mathbf{x} \text{ is in output interval } i. \quad (5)$$

If some data belonging to X_i exist in $I_{ij}(1)$, we define the activation hyperbox of level 2 denoted as $A_{ij}(2)$ within the inhibition hyperbox $I_{ij}(1)$. Similarly, we define fuzzy rules of levels higher than 2 until we resolve an overlap. In a general

from, we define the fuzzy rule $r_{ij}(l)$ of level $l(\geq 1)$ without inhibition as follows:

$$\text{If } \mathbf{x} \text{ is } A_{ij}(l) \text{ then } \mathbf{x} \text{ is in output interval } i, \quad (6)$$

where $i = j$ for $l = 1$ and $i \neq j$ for $l \geq 2$, and we define the activation hyperbox $A_{ij}(l)(l > 2)$ as

$$A_{ij}(l) = \{\mathbf{x} \mid v_{ijk}(l) \leq x_k \leq V_{ijk}(l), k = 1, \ldots, m\} \quad (7)$$

where for $\mathbf{x} \in X_i$ and $\mathbf{x}$ is in $I_{ij}(l-1)$

$$w_{ijk}(l-1) \leq v_{ijk}(l) \leq x_k \leq V_{ijk}(l) \leq W_{ijk}(l-1). \quad (8)$$

Or we define the fuzzy rule $r_{ij}(l)$ of level l with inhibition as follows:

$$\text{If } \mathbf{x} \text{ is } A_{ij'}(l) \text{ and } \mathbf{x} \text{ is not } I_{ij}(l) \text{ then } \mathbf{x} \text{ is in output interval } i, \quad (9)$$

where $j' = i$ for $l = 1$ and $j' = j$ for $l \geq 2$, and the inhibition hyperbox $I_{ij}(l)$ is

$$I_{ij}(l) = \{\mathbf{x} \mid w_{ijk}(l) \leq x_k \leq W_{ijk}(l), k = 1, \ldots, m\} \quad (10)$$

for $v_{ijk} \leq w_{ijk}(l) \leq W_{ijk}(l) \leq V_{ijk}(l)$.

C. Fuzzy Rule Inference

The degree of membership of the fuzzy rule (6) for a given input $\mathbf{x}$ is determined by the membership function of the activation hyperbox $A_{ij}(l)$, while the degree of membership of the fuzzy rule (9) for a given input $\mathbf{x}$ is determined by the difference between the membership function of the activation hyperbox $A_{ij}(l)$ and that of the inhibition hyperbox $I_{ij}(l)$. Thus first we define the membership functions for the activation and inhibition hyperboxes. We assume that the degree of membership of $\mathbf{x}$ for an activation hyperbox $A_{ij}(l)$ is 1 if $\mathbf{x}$ is in the activation hyperbox and it decreases as $\mathbf{x}$ moves away from the activation hyperbox. To realize a membership function with this characteristic we use the following function which takes the minimum value among the degrees of membership of all the input variables. Here, the membership function for each input variable is a trapezoidal shape, as shown in Fig. 3.

$$m_X(\mathbf{x}) = \min_{k=1,\ldots m} m_X(\mathbf{x}, k), \quad (11)$$

$$m_X(\mathbf{x}, k) = \begin{cases} 1 & \text{for } u_k \leq x_k \leq U_k \\ 1 - \max(0, \min(1, \gamma(u_k - x_k)) & \text{for } x_k < u_k \\ 1 - \max(0, \min(1, \gamma(x_k - U_k)) & \text{for } x_k > U_k \end{cases} \quad (12)$$

where $X = A_{ij}(l)$, $u_k = v_{ijk}(l)$, $U_k = V_{ijk}(l)$, and γ is a sensitivity parameter. The minimum value in (11) is taken so that the degree of membership within the hyperbox and on the surface of the hyperbox becomes 1.

The membership function of the inhibition hyperbox $I_{ij}(l)$ with respect to $A_{ij}(l)$ is different from that with respect to $A_{ji}(l)$. So hereafter, when we say the membership function of $I_{ij}(l)$, we mean that of $I_{ij}(l)$ with respect to $A_{ij}(l)$, and the membership function of $I_{ji}(l)$, we mean that of $I_{ji}(l)$ with respect to $A_{ji}(l)$; here $I_{ij}(l) = I_{ji}(l)$.

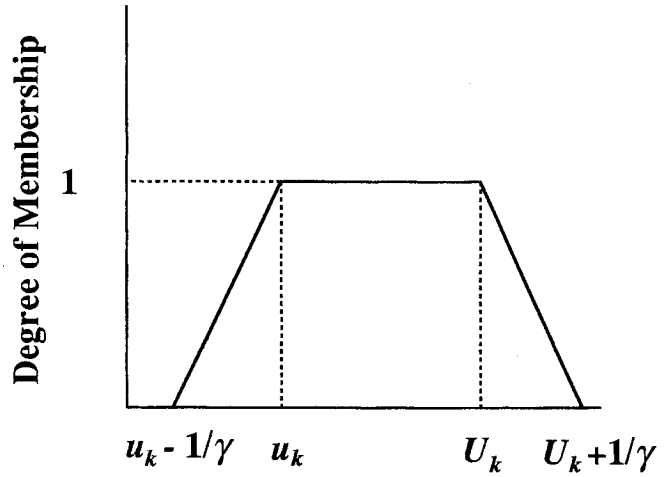

Fig. 3. One-dimensional membership function for the activation hyperbox.

The membership function of the inhibition hyperbox $I_{ij}(l)$ is the same as (11) although the definition of membership function $m_{I_{ij}^{(l)}}(\mathbf{x}, k)$ for each dimension is different. We assume that $m_{I_{ij}(l)}(\mathbf{x}, k)$ is 1 when input x_k is on the boundary of $I_{ij}(l)$ and when this boundary is also the boundary of $A_{ij}(l)$; namely, if $x_k = w_{ijk}(l) = v_{ij'k}(l)$ or $W_{ijk}(l) = V_{ij'k}(l)$ holds where $j' = i$ for $l = 1$, and $j' = j$ for $l \geq 2$. When x_k moves away from this type of boundary of $I_{ij}(l)$, the value of $m_{I_{ij}^{(l)}}(\mathbf{x}, k)$ decreases with the same slope as that of $m_{A_{ij}(l)}(\mathbf{x}, k)$. This is to ensure that the degree of membership of $\mathbf{x}$ for the fuzzy rule (9) becomes 0 on this boundary. When input x_k is on the boundary of $I_{ij}(l)$ and this boundary is not the boundary of $A_{ij}(l)$, in other words $I_{ij}(l)$ is within $A_{ij}(l)$, the value of $m_{I_{ij}(l)}(\mathbf{x}, k)$ is 0 and increases as x_k moves inside $I_{ij}(l)$ from the boundary. This is to ensure that the degree of membership is 1 when $\mathbf{x}$ is in $A_{ij}(l)$ and not in $I_{ij}(l)$.

Let $u_k = w_{ijk}(l)$ and $U_k = W_{ijk}(l)$. Then the membership function $m_{I_{ij}(l)}(\mathbf{x}, k)$ is given by (see top of next page):

Thus the degree of membership of a fuzzy rule $r_{ij}(l)$ represented by (6) for a given $\mathbf{x}$ is

$$d_{r_{ij}(l)}(\mathbf{x}) = m_{A_{ij}(l)}(\mathbf{x}). \quad (19)$$

And the degree of membership of a fuzzy rule $r_{ij}(l)$ represented by (9) for a given $\mathbf{x}$ is

$$d_{r_{ij}(l)}(\mathbf{x}) = \max(0, m_{A_{ij}(l)}(\mathbf{x}) - m_{I_{ij'}(l)}(x)) \quad (20)$$

where $j' = i$ for $l = 1$ and $j' = j$ for $l \geq 2$ (as illustrated in Fig. 7). Here, the maximum operation assures the value will be non-negative.

Thus, the final degree of membership of $\mathbf{x}$ for a set of fuzzy rules $\{r_{ij}(l) \mid l = 1, \ldots\}$ denoted as $d_{r_{ij}}(\mathbf{x})$ is given by

$$d_{r_{ij}}(\mathbf{x}) = \max_{l=1,\ldots}(d_{r_{ij}(l)}(\mathbf{x})). \quad (21)$$

We take the maximum because the activation hyperbox $A_{ij}(l+1)$, if it exists, is included in the inhibition hyperbox $I_{ij}(l)$, and thus each fuzzy rule in $\{r_{ij}(l) \mid l = 1, \ldots\}$ is exclusive of one another.

(i) For $u_k = w_{ijk}(l) \neq v_{ij'k}(l)$ and $U_k = W_{ijk}(l) = V_{ij'k}(l)$ where $j' = i$ for $l = 1$, and $j' = j$ for $l \geq 2$ (see Fig. 4)

(a) For $u_k + 1/\gamma \leq U_k$

$$m_{I_{ij}(l)}(x,k) = \begin{cases} 1 & \text{for } u_k + 1/\gamma \leq x_k \leq U_k \\ 1 - \max(0, \min(1, \gamma(u_k + 1/\gamma - x_k))) & \text{for } x_k < u_k + 1/\gamma \\ 1 - \max(0, \min(1, \gamma(x_k > U_k))) & \text{for } x_k > U_K \end{cases} \tag{13}$$

(b) For $u_k + 1/\gamma > U_k$

$$m_{I_{ij}(l)}(x,k) = \begin{cases} 1 - \max(0, \min(1, \gamma(u_k + 1/\gamma - x_k))) & \text{for } x_k < U_k \\ 1 - \max(0, \min(1, \gamma(x_k - U_k))) & \text{for } x_k \geq U_k \end{cases} \tag{14}$$

(ii) For $u_k = w_{ijk}(l) = v_{ij'k}$ and $U_k = W_{ijk}(l) \neq V_{ij'k}(l)$ where $j' = i$ for $l = 1$, and $j' = j$ for $l \geq 2$ (see Fig. 5)

(a) For $u_k \leq U_k - 1/\gamma$

$$m_{I_{ij}(l)}(x,k) = \begin{cases} 1 & \text{for } u_k \leq x_k \leq U_k - 1/\gamma \\ 1 - \max(0, \min(1, \gamma(u_k - x_k))) & \text{for } x_k < u_k \\ 1 - \max(0, \min(1, \gamma(x_k - U_k + 1/\gamma))) & \text{for } x_k > U_k - 1/\gamma \end{cases} \tag{15}$$

(b) For $u_k > U_k - 1/\gamma$

$$m_{I_{ij}(l)}(x,k) = \begin{cases} 1 - \max(0, \min(1, \gamma(u_k - x_k))) & \text{for } x_k < u_k \\ 1 - \max(0, \min(1, \gamma(x_k - U_k + 1/\gamma))) & \text{for } x_k \geq u_k \end{cases} \tag{16}$$

(iii) For $u_k = w_{ijk}(l) \neq v_{ij'k}(l)$ and $U_k = W_{ijk}(l) \neq V_{ij'k}(l)$ where $j' = i$ for $l = 1$, and $j' = j$ for $l \geq 2$ (see Fig. 6)

(a) For $u_k + 1/\gamma \leq U_k - 1/\gamma$

$$m_{I_{ij}(l)}(x,k) = \begin{cases} 1 & \text{for } u_k + 1/\gamma \leq x_k \leq U_k - 1/\gamma \\ 1 - \max(0, \min(1, \gamma(u_k + 1/\gamma - x_k))) & \text{for } x_k < u_k + 1/\gamma \\ 1 - \max(0, \min(1, \gamma(x_k - U_k + 1/\gamma))) & \text{for } x_k > U_k - 1/\gamma \end{cases} \tag{17}$$

(b) For $u_k + 1/\gamma > U_k - 1/\gamma$

$$m_{I_{ij}(l)}(x,k) = \begin{cases} 1 - \max(0, \min(1, \gamma(u_k + 1/\gamma - x_k))) & \text{for } x_k < u_k + 1/\gamma \\ 1 - \max(0, \min(1, \gamma(x_k - U_k + 1/\gamma))) & \text{for } x_k \geq u_k + 1/\gamma \end{cases} \tag{18}$$

Now the degree of membership of $\mathbf{x}$ for the output interval i denoted as $d_i(\mathbf{x})$ is given by

$$d_i(\mathbf{x}) = \min_{\substack{j \neq i, j=1,\ldots,n, \\ A_{ii}(1) \cap A_{jj}(1) \neq \emptyset}} (d_{r_{ij}}(\mathbf{x})). \tag{22}$$

When the activation hyperbox of the output interval i overlaps with those of more than one output interval e.g., j and k, we resolve the conflict, independently, first between output intervals i and j, then between output intervals i and k. This process is implemented by taking the minimum in (22). For example, if $d_{r_{ij}}(\mathbf{x}) = 1$ and $d_{r_{ik}}(\mathbf{x}) = 0$, this means that $\mathbf{x}$ is in the region inhibited by the inhibition hyperbox between output intervals i and k and thus $\mathbf{x}$ should not be classified as the output interval i at all.

D. Defuzzification

To defuzzify a fuzzy set into a nonfuzzy value in the output space, the method based on the center of gravity is often used. But here we use the method discussed in [4], which uses a bell-shaped function as the membership function of the output variable and approximates the center of gravity as follows:

$$\hat{y} = \frac{\sum_{i=1}^{n} m_i \sigma_i d_i(\mathbf{x})}{\sum_{i=1}^{n} \sigma_i d_i(\mathbf{x})} \tag{23}$$

where m_i and σ_i are, respectively, the center (or mean) and the width (or variance) of a bell-shaped function of the input interval i. We choose this expression for conveniently tuning m_i and σ_i, if necessary for increasing approximation accuracy.

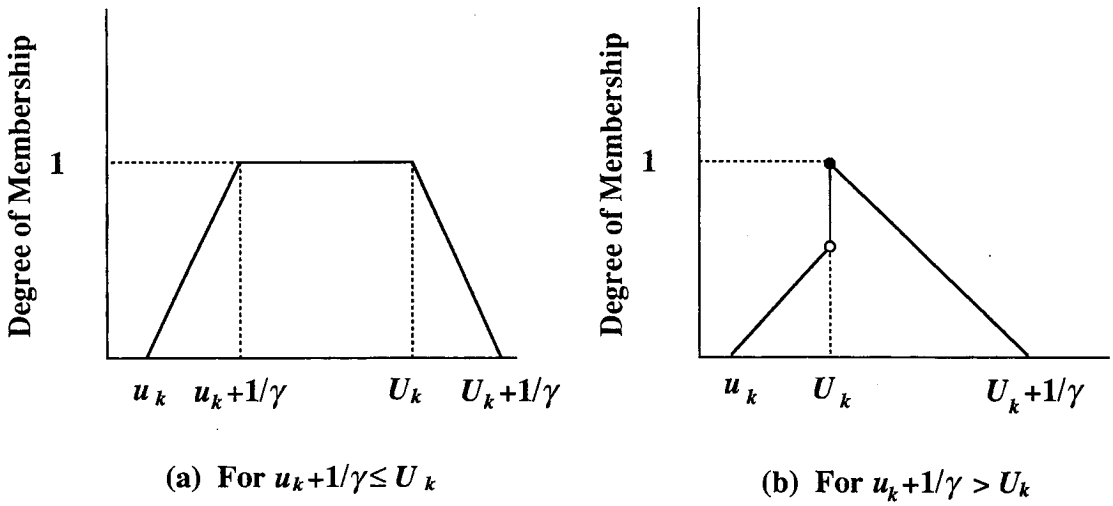

Fig. 4. Membership function for the inhibition hyperbox. The minimum value of the k-th input variable for the inhibition hyperbox is within the corresponding activation hyperbox, while the maximum value is on the surface of the activation hyperbox.

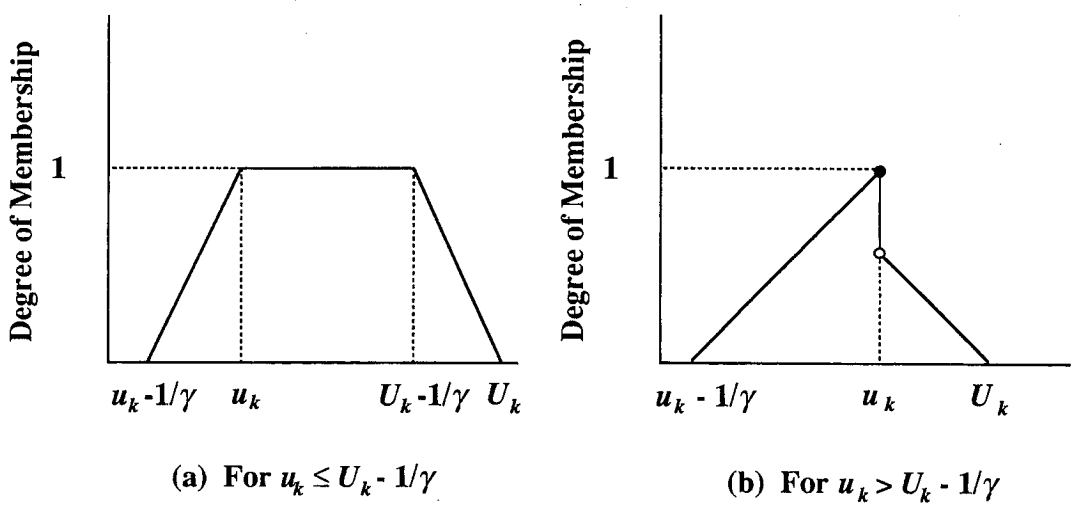

Fig. 5. Membership function for the inhibition hyperbox. The maximum value of the k-th input variable for the inhibition hyperbox is within the corresponding activation hyperbox, while the minimum value is on the surface of the activation hyperbox.

Good initial estimates of m_i and σ_i are, respectively,

$$m_i = (y_i + y_{i-1})/2 \text{ and}$$
$$\sigma_i = (y_i - y_{i-1})/2. \tag{24}$$

To increase the approximation accuracy of fuzzy systems, we can tune m_i and σ_i using the method as follows [4]:

$$m_i^{(n+1)} = m_i^{(n)} + \alpha \Delta m_i^{(n)}$$
$$\Delta m_i^{(n)} = [y(\mathbf{x}) - \hat{y}(\mathbf{x})] \frac{\sigma_i d_i(\mathbf{x})}{\sum_{i=1}^{n} \sigma_i d_i(\mathbf{x})}$$
$$\sigma_i^{(n+1)} = \sigma_i^{(n)} + \alpha \Delta \sigma_i^{(n)},$$

and (see (25) below) where n and α denote the number of epochs and a tuning parameter which controls the rate of change, respectively, and $\Delta m_i^{(n)}$ and $\Delta \sigma_i^{(n)}$ are changes determined by the steepest descent method.

E. Selection of Input Variables

For a given number of divisions in the universe of discourse of the output variable, the larger the number of rules generated for each output interval, the more difficult function approximation becomes. Thus if the total number of rules generated for output intervals is the same even if we delete one input variable, we can consider that this variable is redundant. Namely, we can delete input variables in the following manner.

$$\Delta \sigma_i(n) = [y(\mathbf{x}) - \hat{y}(\mathbf{x})] \times \frac{m_i d_i(\mathbf{x})(\sum_{i=1}^{n} \sigma_i d_i(\mathbf{x})) - d_i(\mathbf{x})(\sum_{i=1}^{n} m_i \sigma_i d_i(\mathbf{x}))}{(\sum_{i=1}^{n} \sigma_i d_i(\mathbf{x}))^2} \tag{25}$$

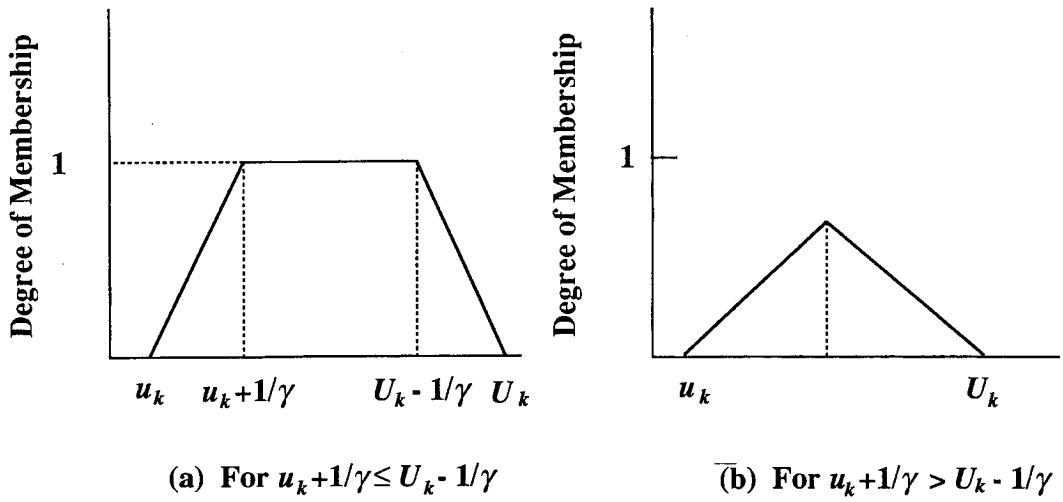

Fig. 6. Membership Function for the Inhibition Hyperbox. The minimum and maximum values of the k-th input variable for the inhibition hyperbox are within the corresponding activation hyperbox.

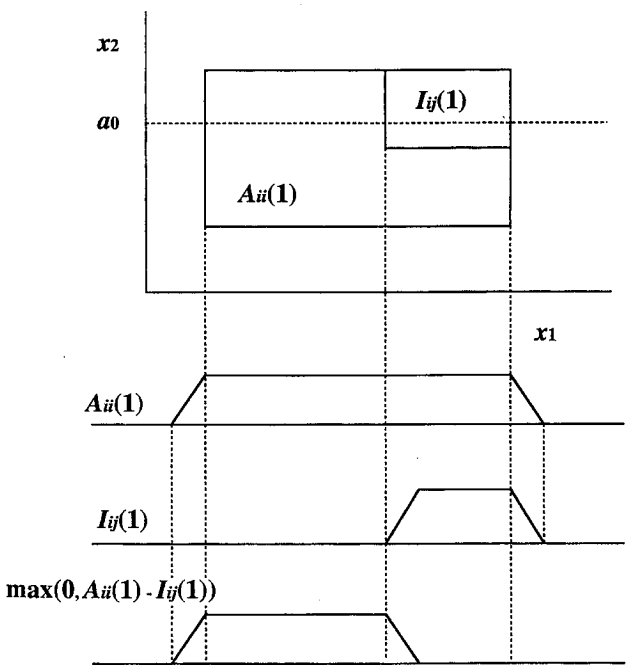

Fig. 7. Degree of membership of output interval i at $x_2 = a_0$ with activation and inhibition hyperboxes for $u_1 + 1/\gamma \leq U_1$. Here u_1 and U_1 are the minimum and maximum values of variable x_1 of an inhibition hyperbox.

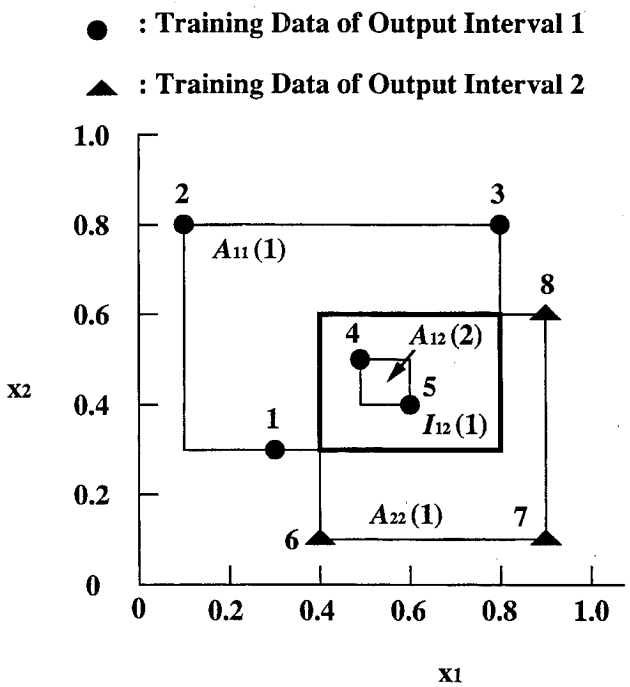

Fig. 8. An example illustrating the procedure of defining activation hyperboxes and an inhibition hyperbox.

Let M be the set of input variables, and $r_i(M)$ be the number of fuzzy rules obtained for the output interval i with M and a given set of data. Let M' be the set in which one input variable is deleted from M. Then acquire new fuzzy rules using numerical data with M'. If

$$\sum_{i=1}^{n} r_i(M) = \sum_{i=1}^{n} r_i(M') \tag{26}$$

holds, then remove the deleted variable from M permanently; otherwise restore the deleted variable. The above procedure is repeated until testing is finished for all the input variables included in the initial set of M.

F. An Illustrative Example

To illustrate the procedures of fuzzy rule extraction, inference and defuzzification step by step, we consider a fuzzy system with two input variables and two output intervals and assume we have training data listed below (See Fig. 8): For output interval 1:

1: (0.3, 0.3)
2: (0.1, 0.8)
3: (0.8, 0.8)
4: (0.5, 0.5)
5: (0.6, 0.4)

For output interval 2:

6: (0.4, 0.1)
7: (0.9, 0.1)
8: 0.9, 0.6)

For simplicity we do not illustrate the tuning procedure, therefore the output values for the training data are omitted.

Fuzzy rule extraction Using the above training data, the activation hyperboxes of level 1 are defined as follows:

$$A_{11}(1) = \{\mathbf{x} \mid 0.1 \le x_1 \le 0.8, 0.3 \le x_2 \le 0.8\} \text{ (use(2))}, \tag{27}$$

$$A_{22}(l) = \{\mathbf{x} \mid 0.4 \le x_1 \le 0.9, 0.1 \le x_2 \le 0.6\} \text{ (use(2))}. \tag{28}$$

Since $A_{11}(1)$ overlaps with $A_{22}(1)$, we define the following inhibition hyperbox:

$$I_{12}(1) = \{\mathbf{x} \mid 0.4 \le x_1 \le 0.8, 0.3 \le x_2 \le 0.6\} \text{ (use(4))}. \tag{29}$$

Thus the fuzzy rules of level 1 are:

$$\begin{aligned} &\text{If } \mathbf{x} \text{ is } A_{11}(1) \text{ and } \mathbf{x} \text{ is not } I_{12}(1) \\ &\text{then } \mathbf{x} \text{ is in output interval 1} \\ &\qquad \text{(use(5))}, \end{aligned} \tag{30}$$

$$\begin{aligned} &\text{If } \mathbf{x} \text{ is } A_{22}(1) \text{ and } \mathbf{x} \text{ is not } I_{12}(1) \\ &\text{then } \mathbf{x} \text{ is in output interval 2} \\ &\qquad \text{(use(5))}. \end{aligned} \tag{31}$$

Since data 4 and 5 belonging to output interval 1 are in $I_{12}(1)$, we define the activation hyperbox of level 2 for output interval 1 as follows:

$$A_{12}(2) = \{\mathbf{x} \mid 0.5 \le x_1 \le 0.6, 0.4 \le x_2 \le 0.5\} \quad \text{(use(7))}. \tag{32}$$

But since no data belonging to output interval 2 are in $I_{12}(1)$, the fuzzy rule extraction process is stopped. And we obtain the following fuzzy rule of level 2:

$$\text{If } \mathbf{x} \text{ is } A_{12}(2) \text{ then } \mathbf{x} \text{ is in output interval 1} \quad \text{(use (6))}. \tag{33}$$

Fuzzy rule inference We assume $\gamma = 1$. Then, the membership functions of $A_{11}(1)$ are as follows:

$$m_{A_{11}(1)}(\mathbf{x}, 1) = \begin{cases} 1 & \text{for } 0.1 \le x_1 \le 0.8 \\ 1 - \max(0, \min(1, (0.1 - x_1))) & \text{for } x_1 < 0.1 \\ 1 - \max(0, \min(1, (x_1 - 0.8))) & \text{for } x_1 > 0.8 \\ & \text{(use (12))}, \end{cases} \tag{34}$$

$$m_{A_{11}(1)}(\mathbf{x}, 2) = \begin{cases} 1 & \text{for } 0.3 \le x_2 \le 0.8 \\ 1 - \max(0, \min(1, (0.3 - x_2))) & \text{for } x_2 < 0.3 \\ 1 - \max(0, \min(1, (x_2 - 0.8))) & \text{for } x_2 > 0.8 \\ & \text{(use (12))}. \end{cases} \tag{35}$$

The membership functions of $I_{12}(1)$ with respect to $A_{12}(1)$ are

$$m_{I_{12}(1)}(\mathbf{x}, 1) = \begin{cases} 1 - \max(0, \min(1, (1.4 - x_1))) & \text{for } x_1 < 0.8 \\ 1 - \max(0, \min(1, (x_1 - 0.8))) & \text{for } x_1 \ge 0.8 \\ & \text{(use (14))}, \end{cases} \tag{36}$$

$$m_{I_{12}(1)}(\mathbf{x}, 2) = \begin{cases} 1 - \max(0, \min(1, (0.3 - x_2))) & \text{for } x_2 < 0.3 \\ 1 - \max(0, \min(1, (x_2 + 0.4))) & \text{for } x_2 \ge 0.3 \\ & \text{(use (16))}. \end{cases} \tag{37}$$

The membership functions $A_{22}(1)$ are

$$m_{A_{22}(1)}(\mathbf{x}, 1) = \begin{cases} 1 & \text{for } 0.4 \le x_1 \le 0.9 \\ 1 - \max(0, \min(1, (0.4 - x_1))) & \text{for } x_1 < 0.4 \\ 1 - \max(0, \min(1, (x_1 - 0.9))) & \text{for } x_1 > 0.9 \\ & \text{(use (12))}, \end{cases} \tag{38}$$

$$m_{A_{22}(1)}(\mathbf{x}, 2) = \begin{cases} 1 & \text{for } 0.1 \le x_2 \le 0.6 \\ 1 - \max(0, \min(1, (0.1 - x_2))) & \text{for } x_2 < 0.1 \\ 1 - \max(0, \min(1, (x_2 - 0.6))) & \text{for } x_2 > 0.6 \\ & \text{(use (12))}. \end{cases} \tag{39}$$

The membership functions of $I_{21}(1)$ with respect to $A_{21}(1)$ are

$$m_{I_{21}(1)}(\mathbf{x}, 1) = \begin{cases} 1 - \max(0, \min(1, (0.4 - x_1))) & \text{for } x_1 < 0.4 \\ 1 - \max(0, \min(1, (x_1 + 0.2))) & \text{for } x_1 \ge 0.4 \\ & \text{(use (16))}, \end{cases} \tag{40}$$

$$m_{I_{21}(1)}(\mathbf{x}, 2) = \begin{cases} 1 - \max(0, \min(1, (1.3 - x_2))) & \text{for } x_2 < 0.6 \\ 1 - \max(0, \min(1, (x_2 - 0.6))) & \text{for } x_2 \ge 0.6 \\ & \text{(use (14))}. \end{cases} \tag{41}$$

The membership functions of $A_{12}(2)$ are as follows:

$$m_{A_{12}(2)}(\mathbf{x}, 1) = \begin{cases} 1 & \text{for } 0.5 \le x_1 \le 0.6 \\ 1 - \max(0, \min(1, (0.5 - x_1))) & \text{for } x_1 < 0.5 \\ 1 - \max(0, \min(1, (x_1 - 0.6))) & \text{for } x_1 \ge 0.6 \\ & \text{(use (12))}, \end{cases} \tag{42}$$

$$m_{A_{12}(2)}(\mathbf{x}, 2) = \begin{cases} 1 & \text{for } 0.1 \le x_4 \le 0.5 \\ 1 - \max(0, \min(1, (0.4 - x_2))) & \text{for } x_2 < 0.4 \\ 1 - \max(0, \min(1, (x_2 - 0.5))) & \text{for } x_2 > 0.5 \\ & \text{(use (12))}. \end{cases} \tag{43}$$

Now calculate the degrees of membership at $\mathbf{x} = (0.65, 0.5)$. The degree of membership of $\mathbf{x}$ for $A_{11}(1)$ is

$$m_{A_{11}(1)}(0.65, 0.5) = \min(1, 1) = 1 \quad \text{(use (34), (35) and (11))}. \tag{44}$$

The degree of membership of $\mathbf{x}$ for $I_{12}(1)$ with respect to $A_{11}(1)$ is

$$m_{I_{12}(1)}(0.65, 0.5) = \min(0.25, 0.9) = 0.25 \quad \text{(use (36), (37), and (11))} \tag{45}$$

Thus the degree of membership for fuzzy rule (30) is

$$d_{r_{11}(1)}(0.65, 0.5) = \max(0, 1 - 0.25) = 0.75 \quad \text{(use (44), (45) and (20))}. \tag{46}$$

Likewise the degree of membership of $\mathbf{x}$ for fuzzy rule (33) is

$$d_{r_{12}(2)}(0.65, 0.5) = m_{A_{12}(2)}(0.65, 0.5) = \min(0.95, 1) = 0.95 \quad \text{(use (42), (43) and (11))}. \tag{47}$$

Thus, the degree of membership of $\mathbf{x}$ for fuzzy rules (30) and (33) is

$$d_1(0.65, 0.5) = \max(0.75, 0.95) = 0.95 \quad \text{(use (46), (47) and (21))}. \tag{48}$$

Since we consider only two intervals, we need not use (22). Likewise the degree of membership of $\mathbf{x}$ for $A_{22}(1)$ is

$$m_{A_{22}(1)}(0.65, 0.5) = \min(1, 1) = 1 \quad \text{(use (39) and (11))}. \tag{49}$$

The degree of membership of $\mathbf{x}$ for $I_{21}(1)$ with respect to $A_{22}(1)$ is

$$m_{I_{21}(1)}(0.65, 0.5) = \min(0.15, 0.2) = 0.15 \quad \text{(use (40), (41) and (11))}. \tag{50}$$

Thus the degree of membership for fuzzy rule (31) is

$$d_{r_{22}(1)}(0.65, 0.5) = d_2(0.65, 0.5) = \max(0, 1 - 0.15) = 0.85 \quad \text{(use (49), (50) and (21))}. \tag{51}$$

Difuzzification Let the output interval is divided as [0, 0.5) and [0.5, 1]. Then using (24),

$$m_1 = 0.25, m_2 = 0.75 \text{ and } \sigma_1, \sigma_2 = 0.25. \tag{52}$$

Substituting (48), (51) and (52) into (23) gives

$$\hat{y} = 0.486 \tag{53}$$

III. Performance Evaluation Using a Water Purification Plant

In this section we apply the proposed method to a water purification plant in which the amount of coagulant injection needs to be estimated [8]. The amount of coagulant injection is determined by ten variables on water qualities, such as turbidity and temperature of water, and on floc image properties such as a floc diameter. For evaluation we divided 563 input-output data which were gathered over a one-year period into 478 stationary data and 95 nonstationary data according to whether the value of turbidity was or was not smaller than a specified value. Then we further divided each type of data into two groups to generate training and test data under the restriction that training and test data have a similar distribution of output values. The resulting training and test data are (1) 241 training data and 237 test data, and (2) 45 training data and 40 test data.

Using each group of the training and test data, we empirically studied the effects of the number of divisions of the output variable, tuning of the mean values and variances of the output membership function, and the sensitivity parameter of input membership functions on approximation accuracy. First, we studied the approximation errors of the fuzzy system by varying the number of divisions of the output variable without tuning mean values m_i's and variances σ_i's, i.e., they were set by using (24). In this study, the sensitivity parameter was set to four. Then we studied whether the approximation accuracy could be improved by tuning the mean values and variances for 100 epochs with the tuning parameter $\alpha = 0.01$. Furthermore, using the optimum number of divisions, we studied the effect of the sensitivity parameter on the approximation accuracy; and we compared the performance of the fuzzy approximator with that of the neural network. Finally, we investigated whether eliminating some input variables, according to the method described in Section II.E would improve the approximation accuracy of the fuzzy system.

A. Performance Evaluation Using the 241 Training Data and 237 Test Data (Stationary Data)

Table I shows the average and maximum approximation errors of the fuzzy system based on 241 training data and 237 test data for different numbers of divisions, varying from 3 to 8, of the universe of discourse for the output variable. The fuzzy system could not determine an output on one or two data in the test data set because the inputs of these data were outside the ranges defined by the training data; to overcome this problem, for these data we changed the sensitivity parameter γ from four to a smaller value by which the fuzzy region in the input space was enlarged. As seen in the table, dividing the universe of discourse of the output variable into seven divisions gave the best performance. To check the effect of tuning the parameters m_i and σ_i on approximation errors, we set α of (25) to be 0.01 and the parameters were tuned for 100 epochs using the training data. The results are listed in Table II. Again the seven-division gave the best performance. In some cases, tuning the parameters for 10 epochs was sufficient because good initial values of m_i and σ_i were used; in most cases, tuning the parameters for more than 100 epochs did not further improve the performance.

TABLE I
APPROXIMATION ERRORS (IN MG/L) OF FUZZY SYSTEMS FOR THE 241 TRAINING DATA AND 237 TEST DATA ($\gamma = 4$)

No. of Divisions	Training Data		Test Data	
	Ave. Error	Max. Error	Ave. Error	Max. Error
3	2.01	6.69	2.16	5.50
4	1.47	7.61	1.53	6.57
5	1.15	7.35	1.28	5.71
6	1.24	5.48	1.40	5.73
7	**1.13**	**6.72**	**1.20**	**5.19**
8	1.21	6.99	1.29	5.18

Note: In this and subsequent tables, the set giving the best performance is indicated by the bold-faced type.

TABLE II
APPROXIMATION ERRORS (IN MG/L) OF FUZZY SYSTEM WITH PARAMETER TUNING FOR THE 241 TRAINING DATA AND 237 TEST DATA (($\gamma = 4$), 100 EPOCHS, $\alpha = 0.01$)

No. of Divisions	Training Data		Test Data	
	Ave. Error	Max. Error	Ave. Error	Max. Error
3	1.75	6.47	1.90	8.33
4	1.60	5.54	1.67	5.76
5	1.12	4.42	1.25	5.31
6	1.12	4.16	1.28	5.53
7	**1.07**	**4.75**	**1.18**	**5.57**
8	1.15	4.09	1.25	5.90

TABLE III
APPROXIMATION ERRORS (IN MG/L) OF NEURAL NETWORKS FOR THE 241 TRAINING DATA AND 237 TEST DATA (10 HIDDEN UNITS, 100 TRIALS)

No. of Epochs	Training Data		Test Data	
	Ave. Error	Max. Error	Ave. Error	Max. Error
500	1.03	12.5	1.09	5.72
1000	0.95	5.20	1.03	7.08
2000	**0.84**	**4.75**	**0.99**	**6.95**
3000	0.83	4.61	1.00	6.95
4000	0.83	4.40	1.01	6.97

TABLE IV
APPROXIMATION ERRORS (IN MG/L) OF FUZZY SYSTEMS FOR THE 241 TRAINING DATA AND 237 TEST DATA WHEN INPUT VARIABLE ARE DELETED (TUNED FOR 100 EPOCHS, 7 DIVISIONS FOR THE OUTPUT RANGE, $\gamma = 4$)

Input Deleted	No. of Rules	Training Data		Test Data	
		Ave. Error	Max. Error	Ave. Error	Max. Error
none	**33**	**1.07**	**4.75**	**1.18**	**5.57**
1	34	1.09	5.82	1.20	5.65
2	35	1.15	4.78	1.22	5.63
3	36	1.07	4.77	1.18	5.45
4	38	1.36	5.37	1.47	5.51
5	42	1.09	4.77	1.20	5.58
6	34	1.09	4.75	1.20	5.59
7	**33**	**1.07**	**4.75**	**1.18**	**5.81**
8	35	1.08	4.75	1.19	5.50
9	37	1.08	4.75	1.21	5.56
10	**33**	**1.06**	**4.73**	**1.17**	**5.62**
7,10	**33**	**1.07**	**4.74**	**1.18**	**5.76**

To compare the results with those generated by using neural networks as function approximators, we trained three-layer networks with five, ten and fifteen hidden units, respectively; and for each of the three different networks, 100 trials were performed and each trial used different initial values. From this, we found that the performance of the three networks did not differ very much; therefore here we show only the results from the network with ten hidden units, which are listed in Table III. When training the network for more than 2000 epochs, e.g., 3000 and 4000 epochs, the average approximation error for the test data increased because the neural approximator overfitted the training data, which resulted in losing the generality. Thus in this case the best performance of the neural approximator was obtained when it was trained for about 2000 epochs. Meanwhile, the average approximation error of the neural approximator was slightly better than that of the fuzzy system, but the maximum error of the former was worse than that of the latter. Therefore the performance of these two types of approximators was considered comparable for our current application.

Training the neural network using a 31 MIPS mainframe computer took an average one minute, while it only took less than one second using a 16 MIPS workstation to extract fuzzy rules. Although tuning the parameters of the fuzzy system took time, it was still less than the training time of the neural network.

Table IV shows the approximation errors as well as the numbers of rules generated when different input variables were deleted. When the input variables 7 and/or 10 were deleted, the number of rules generated was the same as that when none of them was deleted, and the average approximation error and maximum error of these cases were also about the same. When other variables were deleted the approximation error became worse except when the input variable 3 was deleted.

B. Performance Evaluation Using the 45 Training Data and 40 Test Data (Nonstationary Data)

Table V shows the approximation errors of the fuzzy system based on 45 training data and 40 test data, respectively, without tuning the parameters of the defuzzification method. Table VI shows the approximation errors of the fuzzy system based on 45 training data and 40 test data, respectively, with tuning the parameters of the defuzzification method; the optimal number of divisions was 5 or 7. We further checked the effect of the sensitivity parameter γ on the approximation error based on five divisions; Tables VII and VIII show the results without and with tuning, respectively. For both cases, the best results were obtained for $\gamma = 20$. Table IX shows the average approximation errors of the neural network. The best performance was obtained when the network was trained for 1000 epochs; comparing this result with the best result shown in Table VII, the average approximation errors for the training data were about the same, whereas the average approximation error of the fuzzy system for the test data was better than that of the neural network.

TABLE V
APPROXIMATION ERRORS (IN MG/L) OF FUZZY SYSTEMS FOR THE 45 TRAINING DATA AND 40 TEST DATA ($\gamma = 4$)

No. of Divisions	Training Data		Test Data	
	Ave. Error	Max. Error	Ave. Error	Max. Error
3	2.85	7.55	3.11	10.2
4	2.29	7.50	2.65	7.97
5	2.38	8.23	2.79	8.43
6	2.79	6.65	3.12	8.87
7	2.03	7.54	2.33	10.0
8	1.96	8.23	2.34	9.97

TABLE VI
APPROXIMATION ERRORS (IN MG/L) OF FUZZY SYSTEMS WITH PARAMETER TUNING FOR THE 45 TRAINING DATA AND 40 TEST DATA ($\gamma = 4$, 100 EPOCHS, $\alpha = 0.01$)

No. of Divisions	Training Data		Test Data	
	Ave. Error	Max. Error	Ave. Error	Max. Error
3	2.35	8.41	2.88	11.9
4	1.94	7.91	2.34	8.35
5	**1.74**	**8.50**	**2.18**	**8.36**
6	1.83	7.26	2.70	10.6
7	**1.73**	**6.86**	**1.97**	**8.86**
8	1.82	7.23	2.09	9.05

TABLE VII
APPROXIMATION ERRORS (IN MG/L) OF FUZZY SYSTEMS FOR THE 45 TRAINING DATA AND 40 TEST DATA (5 DIVISIONS OF THE OUTPUT RANGE)

γ	Training Data		Test Data	
	Ave. Error	Max. Error	Ave. Error	Max. Error
4	2.38	8.23	2.80	8.43
8	1.73	8.16	2.16	6.37
12	1.75	8.08	1.89	6.75
16	1.70	7.99	1.81	6.89
20	**1.72**	**7.91**	**1.62**	**5.45**

TABLE VIII
APPROXIMATION ERRORS (IN MG/L) OF FUZZY SYSTEMS WITH PARAMETER TUNING FOR THE 45 TRAINING DATA AND 40 TEST DATA (5 DIVISIONS OF THE OUTPUT RANGE, 100 EPOCHS, $\alpha = 0.01$)

γ	Training Data		Test Data	
	Ave. Error	Max. Error	Ave. Error	Max. Error
4	1.74	8.50	2.18	8.36
8	1.60	7.29	1.88	6.39
12	1.60	7.75	1.66	6.28
16	1.57	7.64	1.64	5.90
20	**1.56**	**7.20**	**1.46**	**4.97**

As shown in Table X, the number of rules created based on 10 input variables was 15. When we deleted input variable 1 in addition to variables 7 and 10, as for the 241 training data, the number of rules created remained 15; but if we deleted any of the remaining input variables, the number of rules exceeded 15. Furthermore, the performance of the fuzzy system did not deteriorate when some input variables were deleted according to the proposed method.

TABLE IX
APPROXIMATION ERRORS (IN MG/L) OF NEURAL NETWORKS FOR THE 45 TRAINING DATA AND 40 TEST DATA (10 HIDDEN UNITS, 100 TRIALS)

No. of Epochs	Training Data		Test Data	
	Ave. Error	Max. Error	Ave. Error	Max. Error
500	1.76	7.03	1.84	6.62
1000	**1.59**	**6.83**	**1.74**	**6.78**
2000	1.34	5.52	2.04	6.30
3000	1.11	3.41	2.15	8.49
4000	1.00	2.57	2.12	9.08

TABLE X
APPROXIMATION ERRORS (IN MG/L) OF FUZZY SYSTEMS FOR THE 45 TRAINING DATA AND 40 TEST DATA WHEN INPUT VARIABLES ARE DELETED (TUNED FOR 100 EPOCHS, 5 DIVISIONS OF THE OUTPUT RANGE, $\gamma = 4$)

Input Deleted	No. of Rules	Training Data		Test Data	
		Ave. Error	Max. Error	Ave. Error	Max. Error
none	15	1.74	8.50	2.18	8.36
7,10	15	1.83	7.83	2.00	8.09
1,7,10	15	1.92	8.02	2.14	8.36

IV. DISCUSSION OF RESULTS

The empirical studies described above indicate that the approximation accuracy of the type of fuzzy system proposed in this paper is comparable with that of neural networks. Nevertheless, our fuzzy system approach to function approximation has several advantages over the approach using neural networks. These advantages are listed as follows.

1) The fuzzy system architecture shown in Fig. 2 is automatically determined through the acquisition of fuzzy rules according to the overlap between the input spaces of output intervals. Namely, the network has two layers if there is no overlap between the input spaces of output intervals; or it has three layers if the input space of one output interval overlaps with the input space of at most one output interval; or it has four layers if the input space of one output interval overlaps with the input spaces of more than one output interval.
2) Knowledge acquisition or training is very fast. Also tuning of defuzzification parameters is not a problem because the optimal values are not too far off the initial values.
3) System performance can be easily analyzed by fuzzy rules. Thus modification of rule is also possible.
4) By varying the sensitivity parameter γ, we can easily check whether a test datum is in a region where no activation hyperbox is in its neighborhood.

5) Implementation is relatively easy since the activation and inhibition hyperboxes can be determined recursively.

There are two advantages of our fuzzy systems over conventional fuzzy systems.

(1) Fuzzy rules can be easily obtained directly from numerical data.
(2) The fuzzy systems can be generated even for a large number of input variables.

V. Conclusion

In this paper, we discussed a method for extracting fuzzy rules directly from numerical input-output data for function approximation. First, the universe of discourse of an output variable was divided into multiple intervals, and each interval was treated as a class. Then as for pattern classification, using the input data for each interval, fuzzy rules were recursively defined by activation hyperboxes which show the existence region of the data for the interval and inhibition hyperboxes which inhibit the existence region of data for that interval. The approximation accuracy of the fuzzy system derived by this method was empirically studied using an operation learning application of a water purification plant. Meanwhile, the approximation performance of the fuzzy system was compared with the function approximation approach based on neural networks.

References

[1] B. Kosko, "Fuzzy Systems as Universal Approximators," *Proceedings of the 1992 IEEE International Conference on Fuzzy Systems*, pp. 1153–1162, Mar. 1992.
[2] L.-X. Wang, "Fuzzy Systems are Universal Approximators," *Proceedings of the 1992 IEEE International Conference on Fuzzy Systems*, pp. 1163–1170, Mar. 1992.
[3] K. Hornik, M. Stinchcombe and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.
[4] C. T. Lin and C. S. G. Lee, "Neural Network-based Fuzzy Logic Control and Decision System," *IEEE Trans. Computers*, vol. 40, no. 12, pp. 1320–1336, 1991.
[5] L.-X. Wang and J. M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, Nov./Dec. 1992.
[6] P. K. Simpson, "Fuzzy Min-Max Neural Networks-Part 1: Classification," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, pp. 776–786, 1992.
[7] S. Abe and M.-S. Lan, "A Classifier Using Fuzzy Rules Extracted Directly from Numerical Data," *Proceedings of Second IEEE International Conference on Fuzzy Systems*, pp. 1191–1198, Mar. 1993.
[8] K. Baba, I. Enbutsu and M. Yoda, "Explicit Representation of Knowledge Acquired from Plant Historical Data Using Neural Network," *Proc. IJCNN-90*, San Diego, vol. 3, pp. 115–160, Jun. 17–21, 1990.

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

Since 1972, he has been with Hitachi Research Laboratory, Hitachi, Ltd. and has been 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 at Arlington, Arlington, TX. He is now a chief researcher at Hitachi Research Laboratory. Dr. Abe was awarded an outstanding paper prize from the Institute of Electrical Engineers of Japan in 1984. He is a member of the The International Neural Network Society, The Institute of Electrical Engineers of Japan, The Institute of Electronics, Information and Communication Engineers of Japan, and The Society of Instrument and Control Engineers of Japan.

Ming-Shong Lan is a technical staff member at Rockwell International's Science Center, Thousand Oaks, CA. He received his B.S. in mechanical engineering from National Taiwan University (1976) and his Ph.D. in mechanical engineering from the University of California at Berkeley (1983). He has conducted research in the in-process monitoring and control of manufacturing processes, and applications of AI technology in design and manufacturing. His current research interests include neural networks, fuzzy logic, neurofuzzy systems, adaptive control and intelligent control.