Japanese Journal of Fuzzy Theory and Systems
Volume 2, Number 4, 1990

ACQUISITION OF INFERENCE RULES BY NEURAL NETWORK DRIVEN FUZZY REASONING

Isao HAYASHI, Hiroyoshi NOMURA and Noboru WAKAMI

Central Research Laboratories, Matsushita Electric Industrial Co. Ltd.
Received March 2, 1990; Resubmitted June 21, 1990.

In the conventional fuzzy reasoning there is a tuning problem which requires slightly adjusting the inference rules so that the result of the fuzzy reasoning is same as the observed data. In other words, the shape of fuzzy numbers is determined and fuzzy reasoning rules are adjusted in order to minimize a sum of squares of the error between the result of fuzzy reasoning for input data and output data. On the other hand, we proposed neural network driven fuzzy reasoning which does not require this tuning procedure. In the neural network driven fuzzy reasoning a fuzzy set in the antecedent part and an input and output relationship in the consequent part can be identified by using back-propagating error learning of a neural network. In this paper an algorithm for constructing inference rules by neural network driven fuzzy reasoning is explained, and the validity of the algorithm is shown by constructing the operation rules of a tester who flings the pendulum of an inverted pendulum system.

Keywords: *fuzzy reasoning, tuning problem, neural network, back-propagating error learning, membership function*

1. INTRODUCTION

Fuzzy reasoning has recently been applied to the control field, and there are many application examples

[1] of fuzzy control. In fuzzy reasoning vague language can be described in the inference rules in the "If...Then..." form [2,3]. Since the inference rules can be described in the natural language, the designers of control models can design a system model easily. However, fuzzy reasoning has a tuning problem [4]. Tuning is a procedure which slightly adjusts the reasoning process in order to obtain an optimal result from fuzzy reasoning. In other words, the shape of fuzzy numbers, antecedent part in the fuzzy inference, and the label of fuzzy variables in the consequent part are modified in order to minimize the difference between the result of reasoning for input data and the output data.

As a method for solving the tuning problem, we proposed neural network driven fuzzy reasoning [7,8] which constructs inference rules by a neural network [5,6] learning function. The neural network driven fuzzy reasoning is a method which has a back-propagating error learning model [9] that express fuzzy sets in the antecedent part and has another back-propagating error learning models. that express the input and output relationship in the consequent part. In this paper an algorithm for constructing inference rules by neural network driven fuzzy reasoning is explained, and the validity of the algorithm is confirmed by an experiment using an inverted pendulum system. In the experiment a tester flings a pendulum up from a down position using the device. By using the algorithm of neural network driven fuzzy reasoning, fuzzy rules are determined from the data measured by the tester. The constructed inference rules can certainly fling up the pendulum. The one cycle reasoning time for the fling-up control is approximately 15 msec.

Since neural network driven fuzzy reasoning can determine a fuzzy set in the antecedent part and the input and output relationship in the consequent part by using the learning function of a neural network from the input and output data without adding a slight adjustment to the inference rules, it is considered as a method that solves the tuning problem in fuzzy reasoning.

2. BACK-PROPAGATING ERROR LEARNING

Back-propagating error learning is a learning algorithm for a network consisting of units and combined bodies between units. It can determine the structure of a network which can minimize the error between an output value and the estimate when input and output data are given to the network.

Figure 1 shows a hierarchical network consisting of M layers. The fist layer is called an input layer, the layer in the M order is called an output layer, and the other layers are called intermediate

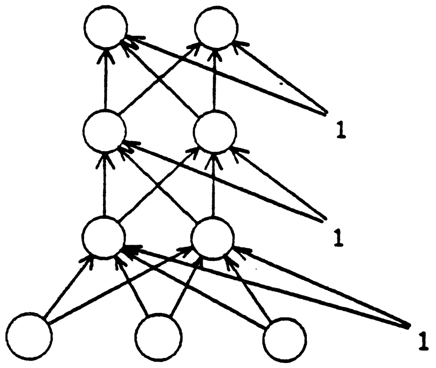


Fig. 1. M layer back-propagating error learning model.

layers. The neural network having M layers in the back-propagating error learning model used in this paper is defined as follows.

1) The M layered back-propagating error learning model is a models having one input layer, M−2 layers of intermediate layers, and one output layer.
2) The input layer and intermediate layers have a constant unit which outputs a constant, 1. The constant unit is not connected to the units in the lower layers but is connected to the upper layers.
3) The units in the adjoining layers, except for constant units, are connected to each other. There are no connections within same layer or between separate layers.

The following is the input and output relationship of each unit. Now, the units in the q order of the layer in the p order are expressed as $u(p, q)$. The total sum of the input of $u(p, q)$ is expressed as x_q^p, and the output is expressed as y_q^p. The output, y_k^{p-1}, of the unit, $u(p-1, k)$, in the k order in the layer in the $p-1$ order becomes the input for the unit, $u(p, q)$, by multiplying it by the degree, $\alpha_{k.\,q}^{p-1,p}$, of the connection between units called a synapse connection. The input and output relationship of $u(p,q)$ is expressed as follows.

$$x_q^p = \sum_{k=1}^{w} \alpha_{k.\,q}^{p-1,\,p} y_k^{p-1} - \theta_q^p \tag{1}$$

$$y_q^p = f(x_q^p) \tag{2}$$

$$f(x) = \frac{1}{1+\exp(-x)} \tag{3}$$

where, θ_q^p is the threshold value of $u(p, q)$, and formula (3) is called a sigmoid function [5]. It is also assumed that the layer in the $p-1$ order has w pieces of units.

Next, the learning algorithm of a back-propagating error learning model is explained.

Now, input and output data (O_i, I_i), $i = 1, 2, \ldots, N$ are given. The optimal output value of the unit in the q order in the output layer for input I_i is defined as O_{qi}, and the estimate of O_{qi} by the learning algorithm is defined as EO_{qi}. During the learning of the output layer a synapse connection α which minimizes s sum of squares, E_i, of the error between the output, O_{qi}, and the estimate, EO_{qi}, is obtained.

$$E_i = (\sum_q (O_{qi} - EO_{qi})^2)/2 \tag{4}$$

$$E = \sum_i E_i \tag{5}$$

The value of E_i in formula (4) is minimized in order to minimize E.

$$\frac{dE_i}{dEO_{qi}} = -(O_{qi} - EO_{qi}) \tag{6}$$

When considering the case where the layer in the p order is the output layer, $EO_q = y_q^p$ is defined. Therefore, the effect, $dEO_{qi}/d\alpha_{k,q}^{p-1,p}$, on the slight change in the degree of connection, $\alpha_{k,q}^{p-1,p}$, is defined as follows from formulas (1) and (2).

$$\frac{dEO_{qi}}{d\alpha_{k.\,q}^{p-1,\,p}} = \frac{dy_{qi}^p}{d\alpha_{k.\,q}^{p-1,\,p}} = f'(x_{qi}^p)\, y_{ki}^{p-1} \tag{7}$$

The effect, $dE_i/d\alpha_{k,q}^{p-1,p}$ on the square error, E_i, of the degree of connection, $\alpha_{k,q}^{p-1}$, is defined as follows.

$$\frac{dE_i}{d\alpha_{k,q}^{p-1,p}}=\frac{dE_i}{dEO_{qi}}\times\frac{dEO_{qi}}{d\alpha_{k,q}^{p-1,p}}=-\delta_q y_{ki}^{p-1} \tag{8}$$

where,

$$\delta_q=f'(x_{qi}^p)(O_{qi}-EO_{qi}) \tag{9}$$

On the other hand, when the layer in the p order is not the output layer, the following is defined by considering the output from unit $u(p, q)$ to unit $u(p+1, g)$.

$$\frac{dE_i}{d\alpha_{k,q}^{p-1,p}}=-\delta_q y_{qi}^p \tag{10}$$

where,

$$\delta_q=f'(x_{qi}^p)\sum_g \delta_g \alpha_{q,g}^{p,p+1} \tag{11}$$

In back-propagating error learning, all the δ values in the direction where the value of E_i in formula (4) is decreasing can be calculated by constantly applying formulas (9) and (11) to the output through the input in the multi-layered structure of back-propagating error learning models.

The changing quantity of the synapse connection, $\alpha_{k,q}$, for decreasing E_i can be calculated as follows.

$$\triangle\,\alpha_{k,q}=\eta\delta_q y_{ki}^{p-1} \tag{12}$$

where, η is a constant.

In this way, the value of the synapse connection is sequentially propagated from the output layer to input layer for decreasing the error between the true value and the estimate. Therefore, it is called back-propagating error learning.

Here, the input and output relationship of a back-propagating error learning model is expressed as follows. The input is defined as $x = (x_1, x_2, \ldots, x_n)$, and the output is defined as $y = (y_1, y_2, \ldots, y_l)$.

$$y=NN(x) \tag{13}$$

The M layer of the scale of a model is expressed as M layers $[u_1 \times u_2 \times \ldots \times u_M]$. u_1 is the number of units in the input layer. $u_i = 1, 2, \ldots, M-1$ are the number of units in the intermediate layers. u_M is the number of units in the output layer. For example, Figure 1 consists of four layers $[3 \times 2 \times 2 \times 2]$.

3. NEURAL NETWORK DRIVEN FUZZY REASONING

A tuning problem exists in fuzzy reasoning [4]. Now, it is assumed that the input and output data, $(y_i, x_i) = (y_i, x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, 2, \ldots, N$, are obtained using input variables, $x_1, x_2, \ldots, x_n$, and output variable, y. The tuning problem is to modify the shape and number of fuzzy numbers and the labels of fuzzy variables in the antecedent and consequent parts in order to minimize a sum of squares of the error between the result of fuzzy reasoning and output data. Neural network driven fuzzy reasoning which performs fuzzy reasoning by driving the learning function of the neural network is one method for solving this tuning problem. The neural network driven fuzzy reasoning is constructed based on fuzzy modeling [10]. The fuzzy modeling is a method which describes the dynamic property of an control object in the inference rules in the form, "If... then...," used in fuzzy control. The consequent part is described using linear formulas. The neural network driven fuzzy reasoning determines a fuzzy set in the antecedent part by using the learning function of a neural network and determines inference rules by considering the consequent part as a non-linear input and output relationship.

Here, an algorithm which constructs the basic idea of the neural network driven fuzzy reasoning and inference rules is explained. Now, the inference rules of the following modeling are considered.

$$\left.\begin{aligned} &R_1 \,;\text{ IF } x_1 \text{ is } F_{SL} \text{ and } x_2 \text{ is } F_{SL} \\ &\quad \text{THEN } y_1 = a_{10} + a_{11}x_1 + a_{12}x_2 \\ &R_2 \,;\text{ IF } x_1 \text{ is } F_{SL} \text{ and } x_2 \text{ is } F_{BG} \\ &\quad \text{THEN } y_2 = a_{20} + a_{21}x_1 + a_{22}x_2 \\ &R_3 \,;\text{ IF } x_1 \text{ is } F_{BG} \\ &\quad \text{THEN } y_3 = a_{30} + a_{31}x_1 \end{aligned}\right\} \tag{14}$$

where, x_1 and x_2 are input variables. y_1, y_2, and y_3 are output variables. a_{10}, a_{11}, $a_{12}, \ldots,$ a_{31} are parameters. F_{SL} and Fb_{BG} are fuzzy numbers that express "SL: small," and "BG: big" respectively. In R_1, $F_1 = F_{SL} \times F_{SL}$ is defined, where $\times$ denotes topological product. Since the antecedent part of R_1 means that "x_1 is small," and "x_2 is small," the fuzzy set, F_1, can be constructed in the partial space of input, R_1, as shown in Figure 2. The fuzzy sets in R_2 and R_3 can be constructed as well. In Figure 2, fuzzy sets are constructed on F_1, F_2 and F_3, respectively. Since each fuzzy sets are overlapped with other ones, the overlapped-area is described by hatched lines. Input space is divided into the number of inference rules, and sub-space are constructed by divided area. A fuzzy set in the antecedent part is constructed on each sub-space. Input data are scattered in the input space. Now, a case where input data are divided into several groups, C_s, $s = 1, 2, \ldots, r$, is considered. A fuzzy set in the antecedent part is constructed by using the divided input data by back-propagating error learning. Here, it is assumed that the degree, W_{is}, where input data x_i belongs to division C_s is given as 0 or 1 as follows.

$$\begin{aligned} &1\text{: In the case where } x_i \text{ belongs to } C_s \\ &0\text{: In the case where } x_i \text{ does not belong to } C_s \end{aligned} \tag{15}$$

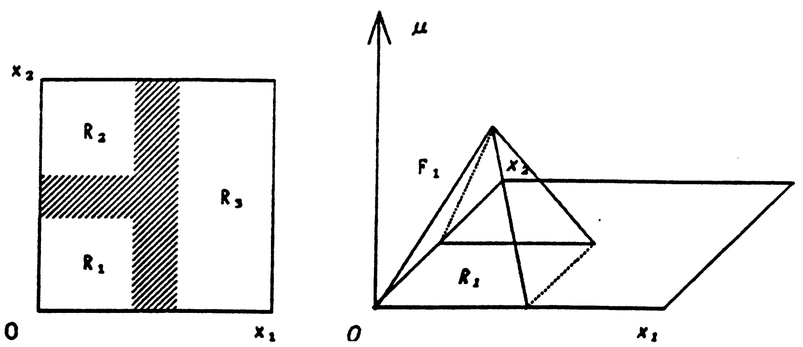


Fig. 2. Input space partition by fuzzy reasoning.

The given input data x_i satisfies $W_{is} = 1$ only for certain C_s, and for the other $C_j, j = 1, 2,\ldots, r$, $j \neq s$, $W_{ij} = 0$. Since the degree W_{is} shows whether input data x_i belongs to each division C_s or not, it can be said that it indicates a possibility that input data return to each C_s. In this paper, the degree of possibility for input data is obtained as the real number of $EW_{is} \in [0, 1]$ and not as the integer of $\{0, 1\}$ by learning using a back-propagating error learning model. Here, EW_{is} is the degree of possibility for the input data obtained by learning using a back-propagating error learning model.

Now, the degree of possibility that the given input data, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ belong to C_s is expressed using the symbol, $W_i = (W_{i1}, W_{i2}, \ldots, W_{is}, \ldots, W_{ir}) = (0, 0, \ldots, 1, \ldots, 0)$. The back-propagating error learning is performed by assigning $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ in the input layers and by assigning $(W_{i1}, W_{i2}, \ldots, W_{is}, \ldots, W_{ir})$ in the output layers. After back-propagating error learning, the data except for the given input data are placed in the input layers of the back-propagating error learning model. The obtained output value is not the integer of $\{0, 1\}$ but the real number of $EW_{is} \in [0, 1]$.

$$\begin{aligned} EW_v &= NN_{mem}(x_v),\ v = 1,2,\ldots,k \\ EW_v &= (EW_{v1}, EW_{v2}, \ldots, EW_{vs}, \ldots, EW_{vr}) \end{aligned} \tag{16}$$

where, NN_{men} shows a back-propagating error learning model for determining a fuzzy set in the antecedent part, and EW_v shows r pieces of output values of NN_{men} for input data x_v.

In formula (15) the degree of a possibility, W_{is}, for the given data shows a possibility that it belongs to C_s using the integer of $\{0, 1\}$. On the other hand, the possibility EW_{is} for new data except for the given input data is obtained in a real number within the division, [0, 1]. The membership value $\mu_{F_s}(x_i)$ of fuzzy set F_s for the antecedent part of the inference rule in the s order is defined as follows.

$$\mu_{F_s}(x_i) = EW_{is},\ i = 1,2,\ldots,k,\ s = 1,2,\ldots,r \tag{17}$$

The formula 17 defines a possibility [11] that input data belong to inference rules as the membership function of a fuzzy set in the antecedent part.

In the neural network driven fuzzy inference, for constructing a fuzzy set in the antecedent part input data are divided into s pieces according to the dispersion condition of the data. The number s of this dividing becomes the number of inference rules. A fuzzy set of each inference rule R_s in the antecedent part is determined from formulas (15), (16), and (17) by using the input data which belongs to each division C_s. This considers division C_s as R_s.

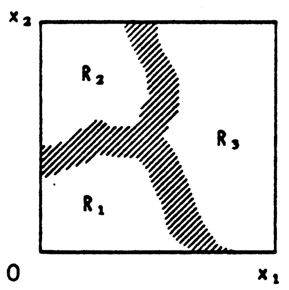


Fig. 3. Input space partition by neural network driven fuzzy reasoning.

As a study for determining a fuzzy set using a neural network, the method of Pao [12] is mentioned. Pao obtained a set of the sum of products using a neural network. However, this determines a set of the sum of products from the connecting patterns between units in the neural network and does not identify the shape of a fuzzy set from input and output data as in the neural network driven fuzzy reasoning.

Figure 3 shows an example of space division in the case where a membership value was determined by using back-propagating error learning models when there are two input variables. The division of the rules by the neural network driven fuzzy reasoning does not become a rectangle shape as shown in Figure 2, and the boundary of the division becomes a non-linear shape.

Next, a basic idea for the identification method of a consequent part is shown. In the neural network driven fuzzy reasoning a consequent part does not become a linear formula. In formula (14) the consequent part uses linear formula, but the back-propagating error learning model of formula (13) is used instead. Since a back-propagating error learning model can identify the non-linear relationship of input and output data, this becomes the same as tuning the consequent part of each inference rule into a non-linear formula. The input and output relationship of the consequent part of each inference rule is expressed as follows.

$$y_s = NN_s(\boldsymbol{x}), \quad s = 1, 2, \cdots, r \tag{18}$$

Here, NN_s is the back-propagating error learning model of the consequent part of the inference rule in the s order, and the number of the output variable of NN_s is one.

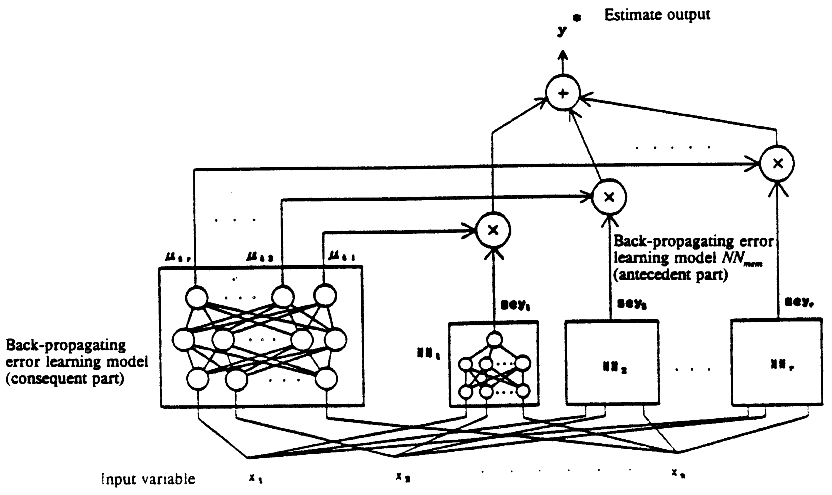
The input data, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, 2, \ldots, (N)^s$ are substituted for the input layer of the back-propagating error learning model, NN_s, and output data y_i for input data x_i are substituted for the output layer in order to perform back-propagating error learning. Here, $(N)^s$ represents the number of the input and output data of the inference rule in the s order. In this way the non-linear relationship of the entire input and output space can be identified by identifying the input and output relationship in the partial space of the divided input and output space in the number of inference rules by using a back-propagating error learning model in the consequence part.

Now, it is also possible to identify the entire non-linear relationship using one back-propagating error learning model. However, the identification of the entire input and output relationship using a back-propagating error learning model for each partial space can represent the property of the entire non-linear relationship more clearly than identification using one back-propagating error learning model.

Now, the best model is selected for a back-propagating error learning model in the consequent part. In the conventional regression analysis there is a stagewise procedure [13] for determining an optimal combination of input variables and a model formula by introducing or removing particular input variables among the combinations of input variables in order to obtain a model that outputs the best estimate. In this paper only the removal of input variables among the combinations of input variables is performed using back-propagating error learning models in order to obtain an optimal combination of input variables and model formula. Here, a sum of squares of errors is used as the evaluation for determining input variables.

In this way a fuzzy set in the antecedent part and an input and output relationship in the consequent part are determined. In the neural network driven fuzzy reasoning, inference rules are expressed as follows.

$$\left.\begin{array}{l} R_s \,;\ \text{IF } \boldsymbol{x} = (x_1, x_2, \cdots, x_n) \ \text{ is } \ A_s \\ \text{THEN } y_s = NN_s(x_1, x_2, \cdots, x_m), \\ \qquad s = 1, 2, \cdots, r, \quad m \leqq n \end{array}\right\} \tag{19}$$



y^* Estimate output
Back-propagating error learning model NN_{mem} (antecedent part)
Back-propagating error learning model (consequent part)
μ_{1r}
μ_{12}
μ_{11}
mey_1
mey_2
mey_r
NN_1
NN_2
NN_r
Input variable
x_1
x_2
x_n

Fig. 4. Structure of neural network driven fuzzy reasoning.

Where, there are r number of inference rules. A_s represents a fuzzy set in the input space in the antecedent part. The degree that input, $x = (x_1, x_2, \ldots, x_n)$, belongs to the inference rule in the s order is the membership value of fuzzy set A_s for input x. The quantity of the operations in the consequent part, y_s, is the estimate of output when a combination of input variables $(x_1, x_2, \ldots, x_m)$ is substituted for the input layer of the back-propagating error learning model, NN_s. Here, the number of the combinations of variables used for the consequent part is m by the best model selection using back-propagating error learning models.

The estimate, y_i^* in the case where input and output data, (y_i, x_i), $i = 1, 2, \ldots, N$ are given is obtained as follows.

$$y_i^* = \frac{\sum_{s=1}^{r} \mu_{As}(x_{i1}, x_{i2}, \cdots, x_{in}) \times mey_{is}}{\sum_{s=1}^{r} \mu_{As}(x_{i1}, x_{i2}, \cdots, x_{in})} \qquad i=1, 2, \cdots, N \tag{20}$$

where, $\mu_{As}(x_{i1}, x_{i2}, \ldots, x_{in})$ is the membership function of fuzzy set A_s, and mey_{is} is the estimate of output in the i order from the input data obtained from each NN_s.

Next, an algorithm for constructing inference rules in the neural network driven fuzzy reasoning is shown.

Figure 4 shows the outline of the algorithm. The inference rules and the quantity of the operation, y_i^*, for input data x_i can be obtained in the following procedure.

Procedure 1. Input variables, $x_1, x_2, \ldots, x_n$, which relate to output y are set. Here, it is assumed that input and output data, $(y_i, x_i) = (y_i, x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, 2, \ldots, N$, were obtained. The input data, x_{ij}, $j = 1, 2, \ldots, n$, show the data in the i order of the input variable in the j order.

Procedure 2. The input and output data are divided into r divisions, R_s, $s = 1, 2, \ldots, r$. As stated above, since division C_s is considered as R_s, each division means an inference rule. Here, the input and output data of each R_s is expressed in (y_i, x_i), $i = 1, 2, \ldots, N^s$. Here, N^s is the number of the input and output data of each R_s.

Procedure 3. The shape of a membership function in the antecedent part is determined using NN_{men} in Figure 4. The method for determining the shape of a membership function was described before. Figure 5 shows an example which determines the membership value of fuzzy numbers in the antecedent part by using 3 layers [$2 \times 3 \times 3$] of a back-propagating error learning model. In Figure 5 the first input data are $(x_{11}, x_{11}) = (0.2, 0.15)$, and $(R_1, R_2, R_3) = (1, 0, 0)$ is obtained. The structure of a back-propagating error learning model is learned by allocating these relationships to the input layer and output layer of back-propagating error learning models, respectively. Here, the repeating number of learning is approximately 1,000. After learning, for example, the membership value, $(R_1, R_2, R_3) = (0.4, 0.1, 0.2)$, is obtained for a fuzzy set in the antecedent part, which consists of the new data, $(x_1, x_2) = (0.5, 0.4)$, except for the given input data.

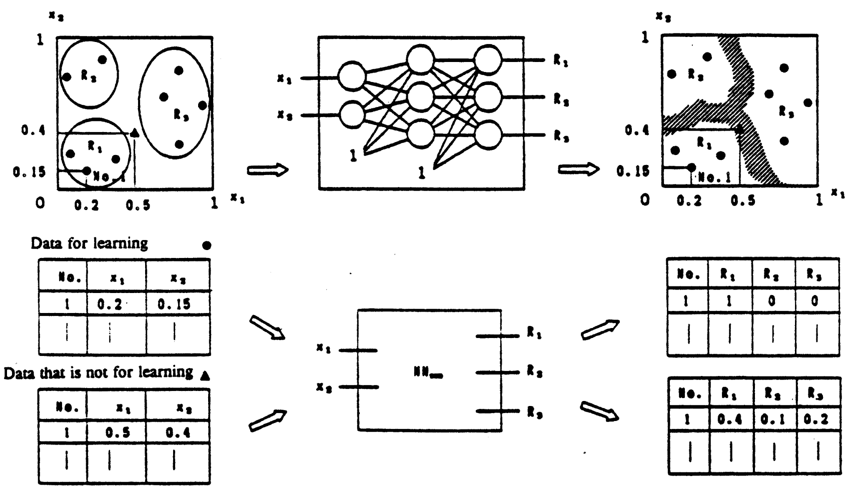
Procedure 4. The structure of the consequent part is identified using NN_1, NN_2, ..., NN_r in Figure 4. Each back-propagating error learning model NN_s describes the consequent part of each inference rule and defines that the structure has M layers [$k \times u_2 \times \ldots \times u_{M-1} \times 1$], $k = n, n - 1, \ldots, 1$. Then, the best model is selected for each NN_s. Now, the procedure for determining input variables using back-propagating error learning models and a method for identifying the structure of a consequent part are described.

First of all, $k = n$ is defined, and the input data, $x_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$, $i = 1, 2, \ldots, N$, are assigned to the input layer of the back-propagating error learning model, NN_s. The output data, y_i, is assigned to the output layer of NN_s. The input variables assigned to the input layer of NN_s and the output variables assigned to the output layer are expressed as follows.

$$\Lambda_s = \{x_1, x_2, \cdots, x_k\} \tag{21}$$

$$\Omega_s = \{y\} \tag{22}$$

where, Λ represents a set of input variables assigned to the input layer of each back-propagating error learning model NN_s, and Ω represents a set of output variables assigned to the output layer of NN_s.



No.	x_1	x_2
1	0.2	0.15
⋮	⋮	⋮

No.	x_1	x_2
1	0.5	0.4
⋮	⋮	⋮

No.	R_1	R_2	R_3
1	1	0	0
⋮	⋮	⋮	⋮

No.	R_1	R_2	R_3
1	0.4	0.1	0.2
⋮	⋮	⋮	⋮

Fig. 5. Decision of a membership function in the antecedent part of rules.

After the back-propagating error learning of NN_s, an estimate ey_i for the input data, $x_{i1}, \ldots, x_{ik}$ is obtained. The repeating learning number is approximately 3,000. A sum of squares for the error between output data y_i and estimate ey_i is calculated, and the evaluation value, Θ_{ks} for determining input variables is obtained.

$$\Theta_k{}^s = (\sum_{i=1}^{N} (y_i - ey_i)^2)/N, \quad s = 1, 2, \cdots, r \tag{23}$$

In order to measure the refrection of the relation between input variables x_j and output variable y, input variable x_j is temporally removed from a set of input variables, $\{x_1, x_2, \ldots, x_j, \ldots, x_k\}$. The input data without input variables, x_j, $x_{i1}, \ldots, x_{ij-1}, x_{ij+1}, \ldots, x_{ik}$, $i = 1, 2, \ldots, N$, are assigned to the input layer of the back-propagating error learning model in the M layer, $[k - 1 \times u_2 \times \ldots \times u_{M-1} \times 1]$, and output data y_i are assigned to the output layer. After back-propagating error learning, estimate ey_i' for the input data, $x_{i1}, \ldots, x_{ij-1}, x_{ij+1}, \ldots, x_{ik}$, is obtained. A sum of squares for the error between this estimate ey_i' and output data y_i is calculated, and the evaluation value, $\Theta_{k-1}{}^{sj}$, for determining input variables is obtained.

$$\Theta_{k-1}{}^{sj} = (\sum_{i=1}^{N} (y_i - ey_i')2)/N, \quad s = 1, 2, \cdots, r \tag{24}$$

The same operation is applied to the input variables besides x_j, and the evaluation values, $\Theta_{k-1}{}^{s1}$, $\Theta_{k-1}{}^{s2}, \ldots, \Theta_{k-1}{}^{sj}, \ldots, \Theta_{k-1}{}^{sk}$, are obtained. Now, among the evaluation values, the minimum $\Theta_{k-1}{}^{sc}$ is obtained.

$$\Theta_{k-1}{}^{sc} = \min_j \Theta_{k-1}{}^{sj}, \quad j = 1, 2, \cdots, k \tag{25}$$

Formula (25) shows that the evaluation value, $\Theta_{k-1}{}^{sc}$, in the case where input variable x_c was eliminated from a set of input variables becomes minimum among the evaluation values, $\Theta_{k-1}{}^{s1}, \Theta_{k-1}{}^{s2}, \ldots, \Theta_{k-1}{}^{sj}, \ldots, \Theta_{k-1}{}^{sk}$.

A set of input variables, Λ_s, which will be assigned to the input layer of each back-propagating error learning model is modified as follows by comparing $\Theta_{k-1}{}^{sc}$ in formula (25) and $\Theta_k{}^s$ in formula (23).

$$\begin{aligned} \Lambda_s = \ & \{x_1, x_2, \cdots, x_{c-1}, x_{c+1}, \cdots, x_k\} \\ \text{if} \ \ & \Theta_{k-1}{}^{sc} < \Theta_k{}^s \end{aligned} \tag{26}$$

$$\begin{aligned} \Lambda_s = \ & \{x_1, x_2, \cdots, x_k\} \\ \text{if} \ \ & \Theta_{k-1}{}^{sc} \geqq \Theta_k{}^s \end{aligned} \tag{27}$$

When formula (26) is obtained, a sum of squares of the error decreases by eliminating input variable x_c. This means that estimate ey_i' expresses y_i better than ey_i. Therefore, the relation between input variable x_c and output variable y is considered to be weak, and input variable x_c is eliminated from a set of input variables, Λ_s. As a result, a new set of input variables, Λ_s consists of $k - 1$ pieces of input variables. On the other hand, when formula (27) is obtained, the validity by eliminating the input variable is not obtained. Therefore, the relation between input variable x_c and output variable y is considered to be strong and the set, Λ_s, consisting of k pieces of input variables remains the same.

When the input variables can be reduced, k is changed to $n - 1, n - 2, \ldots, 1$, and procedure 4 is repeated until formula (27) can be obtained. When formula (27) is obtained, the procedure for decreasing the input variables of back-propagating error learning model NN_s is finished. When the procedure ends, the back-propagating error learning model, NN_s, having a set of final input variables,

$\Lambda_s = \{x_1, x_2, \ldots, x_m\}$, becomes an optimal back-propagating error learning model which expresses the structure of the consequent part of rule R_s. The same operation is repeated for each NN_s, and the structure of the consequent part of all the inference rules is determined. This operation for decreasing input variables is called a stagewise variable decreasing procedure by back-propagating error learning models.

Procedure 5. The estimate output, y_i^*, is obtained from the following formula.

$$y_i^* = \frac{\sum_{s=1}^{r} \mu_{As}(x_{i1}, x_{i2}, \cdots, x_{in}) \times mey_{is}}{\sum_{s=1}^{r} \mu_{As}(x_{i1}, x_{i2}, \cdots, x_{in})} \quad i=1, 2, \cdots, N \tag{28}$$

where, mey_{is} is the estimate by the optimal back-propagating error learning model obtained in procedure 4.

Figure 4 shows that a product operation, $\otimes$, is applied to a membership value in the antecedent part of each inference rule, $\mu_{As}(x_{i1}, x_{i2}, \ldots, x_{in})$, and to the estimate in the consequent part, mey_{is}. Consequently, the estimate value, y_i^*, is obtained from the result of sum operation, $\oplus$, between rules. In Figure 4, the expression, $\Sigma\mu_{As}(x_{i1}, x_{i2}, \ldots, x_{in}) = 1$, is used.

4. EXPERIMENT OF AN INVERTED PENDULUM BY NEURAL NETWORK DRIVEN FUZZY REASONING

A pendulum flinging experiment using an inverted pendulum system is performed in order to show the validity of neural network driven fuzzy reasoning.

The purpose of an inverted pendulum system is to perpendicularly control a freely rotating pendulum without bringing it down, and the truck of the pendulum is run by a motor. When an inverted pendulum system is used, problems that occur when the control theory is applied to actual equipment can be confirmed visually, therefore, the validity of a control algorithm can be verified.

Figure 6 shows the structure of the inverted pendulum system used in this experiment. The gradient of the pendulum is detected by potention meter b, and the pendulum is maintained perpendicularly by quickly moving the truck in the direction where the pendulum falls down. The inverted pendulum device consists of the following four parts.

1) Truck that runs on the rail
2) A pendulum which can rotate 360°. It is attached to the truck via a shaft.
3) A Pulley and a belt system which connect the above three parts.

As shown in Figure 6, the angle from the perpendicular pendulum and the displacement of the truck can be detected by potention meters a and b respectively. The values of angle, Θ, and of the distance of the truck, r, are converted to digitals using an A/D converter and entered in the personal computer, Panacom-M700. The angular speed, $\Delta\Theta$, is obtained by calculating the difference in the quantity between the present angle Θ and the angle before one sampling time, which is approximately 4 msec. The speed of the truck, Δr, is also obtained from the difference in the quantity between the present speed and the speed before one sampling time. At the same time, M700 reasons the torque signal of the motor by fuzzy reasoning. The torque command from M700 is converted to an analogue using a D/A converter and are outputted to the DC motor.

The following shows the specifications of the inverted pendulum experimental system and the attached parts.

1) Inverted pendulum experimental device body: 1,410 mm length, 400 mm width, 880 mm height.
2) Inverted pendulum: 400 mm length, 40 g weight, 4 mm diameter.

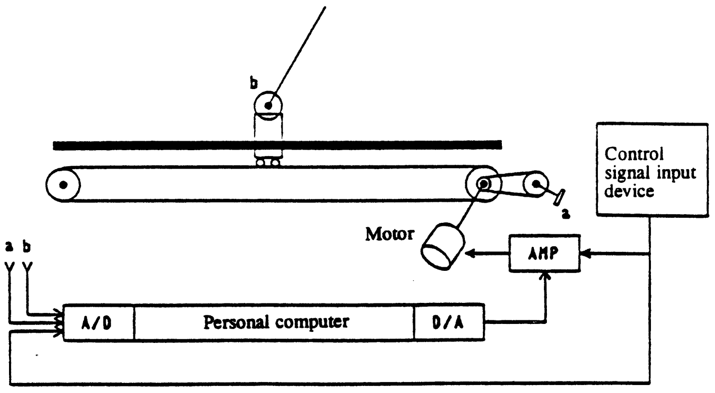


Fig. 6. Structure of an inverted pendulum device.

3) Motor: 25 W DC motor, 12.5:1 rotation ratio.

4) Computer: Panacom - M700 (CPU80286).

5) Control program: C language, 21 K bytes.

The inverted pendulum system used in this experiment has the following two balancing points.

1) Stable balancing point when the pendulum is down.

2) Unstable balancing point when the pendulum is inverted perpendicularly.

Here, the pendulum is flung from the stable balancing point by driving the truck horizontally, and the pendulum which comes close to the unstable balancing point is stopped perpendicularly.

Next, the control rules of the inverted pendulum is created according to the algorithm for constructing inference rules by neural network driven fuzzy reasoning.

Procedure 1. A method for creating input and output data is described here. A tester drives the truck horizontally and flings the pendulum by pressing two buttons in the control signal input device for flinging it and letting it invert. The operation is continued until the pendulum comes close to the perpendicular condition, and the following input and output data are measured in the approximately 4 msec sampling time.

y: motor control signal (V)

x_1: Distance of the truck from the original point (cm)

x_2: Speed of the truck (cm/sec)

x_3: Angle of the pendulum from the original point (deg)

x_4: Angular speed of the pendulum (deg/sec)

The input values, x_2 and x_4 are obtained by calculating the difference in the quantity of x_1 and x_3.

Approximately 1,000-3,000 pieces of data are measured from this operation, and 90 pieces of data are extracted as the input and output data of neural network driven fuzzy reasoning. Table 1 shows the extracted input and output data.

Procedure 2. The input and output data are defined in two rules according to the dispersion condition of the data.

Procedure 3. The shape of the membership function in the antecedent part is determined. The back-propagating error learning model for determining the structure of the antecedent part has three layers [$4 \times 6 \times 2$]. The number of learning is approximately 1,000.

Table 1 Input and output data for controlling an inverted pendulum.

No.	Output data				Input data
	x_1 [cm]	x_2 [cm/sec]	x_3 [deg]	x_4 [deg/sec]	y [v]
1	-1.1482	0.0000	178.5074	0.0000	0.7597
2	-0.0201	8.5486	180.9129	34.5660	0.7421
3	3.2197	29.9073	185.6439	34.5660	0.7617
4	7.8338	38.4472	188.4660	0.0000	0.0039
5	10.9510	4.2697	182.7386	-121.0536	-0.7168
6	9.3718	-21.3586	165.3085	-155.6554	-0.7968
7	5.3319	-38.4560	151.5283	-69.1678	-0.7519
8	0.1432	-42.7261	150.9487	51.8840	-0.7519
93	7.9980	42.7261	85.663	-380.4464	-0.0136
94	11.7713	4.2701	199.1743	-639.8375	-0.7265
95	10.6843	-17.0885	117.4961	-622.5536	-0.7519
96	7.0135	-42.7261	56.6514	-345.8786	-0.7519
97	1.9891	-38.4560	27.0170	-138.3357	0.0039
98	-2.9937	-34.1774	16.6419	-51.8822	-0.0019

Table 2 Elimination of the input variables of rule 1.

Case where variables are not eliminated	$\Theta_4{}^1=0.016$
Case where x_1 is eliminated	$\Theta_3{}^{11}=0.007$
Case where x_2 is eliminated	$\Theta_3{}^{12}=0.040$
Case where x_3 is eliminated	$\Theta_3{}^{13}=0.026$
Case where x_4 is eliminated	$\Theta_3{}^{14}=0.032$

Procedure 4. The structure of the consequent part is identified. The back-propagating error learning model for determining the structure of the consequent part has three layers $[k \times 6 \times 1]$, $k = 4, 3, \ldots, 1$. The number of the learning of each back-propagating error learning model is approximately 3,000. Table 2 is obtained in relation to Θ of rule 1 by the stagewise variable decreasing procedure using back-propagating error learning models. From Table 2 the following are obtained.

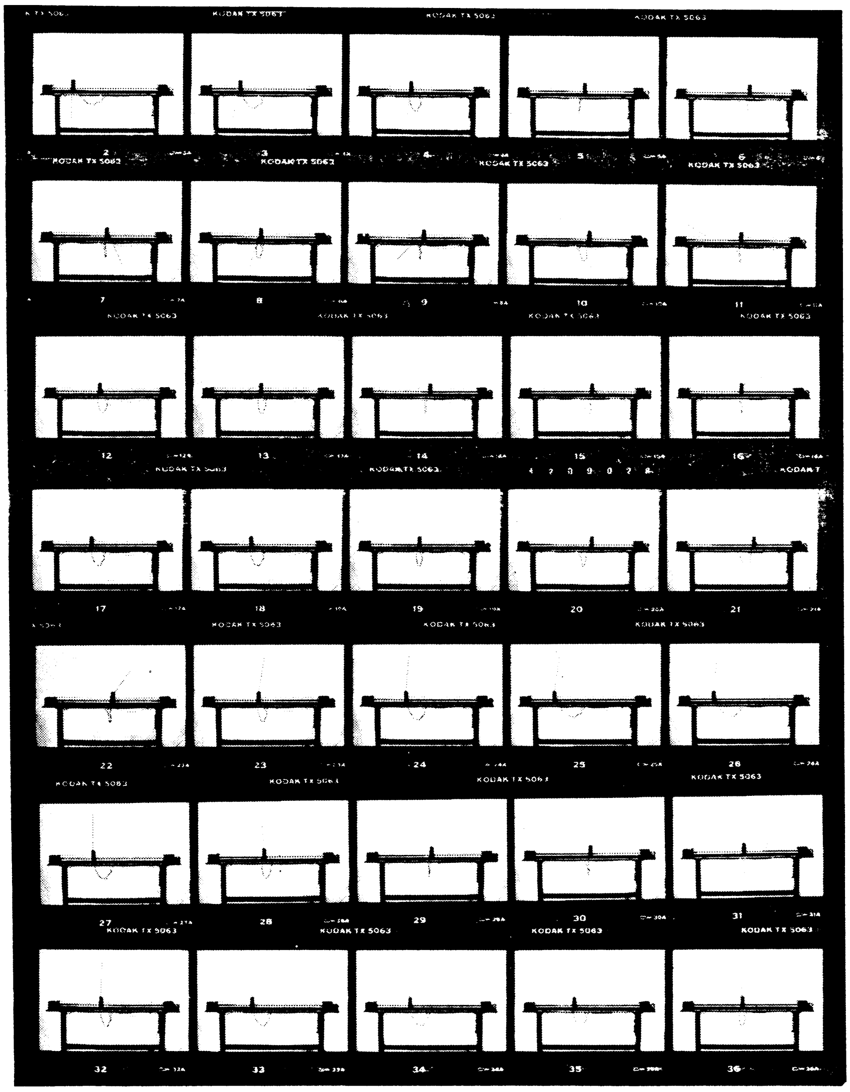
$$\Theta_4{}^1=0.016 \tag{29}$$

$$\Theta_3{}^{11}=\min_j \Theta_3{}^{1j}(=0.007),\ j=1,\ 2,\ \cdots,\ 4 \tag{30}$$

Therefore, the following is defined.

$$\Theta_3{}^{11}<\Theta_4{}^1 \tag{31}$$

The input variable, x_1, is eliminated, the Λ_s becomes $\Lambda_s = \{x_2, x_3, x_4\}$.

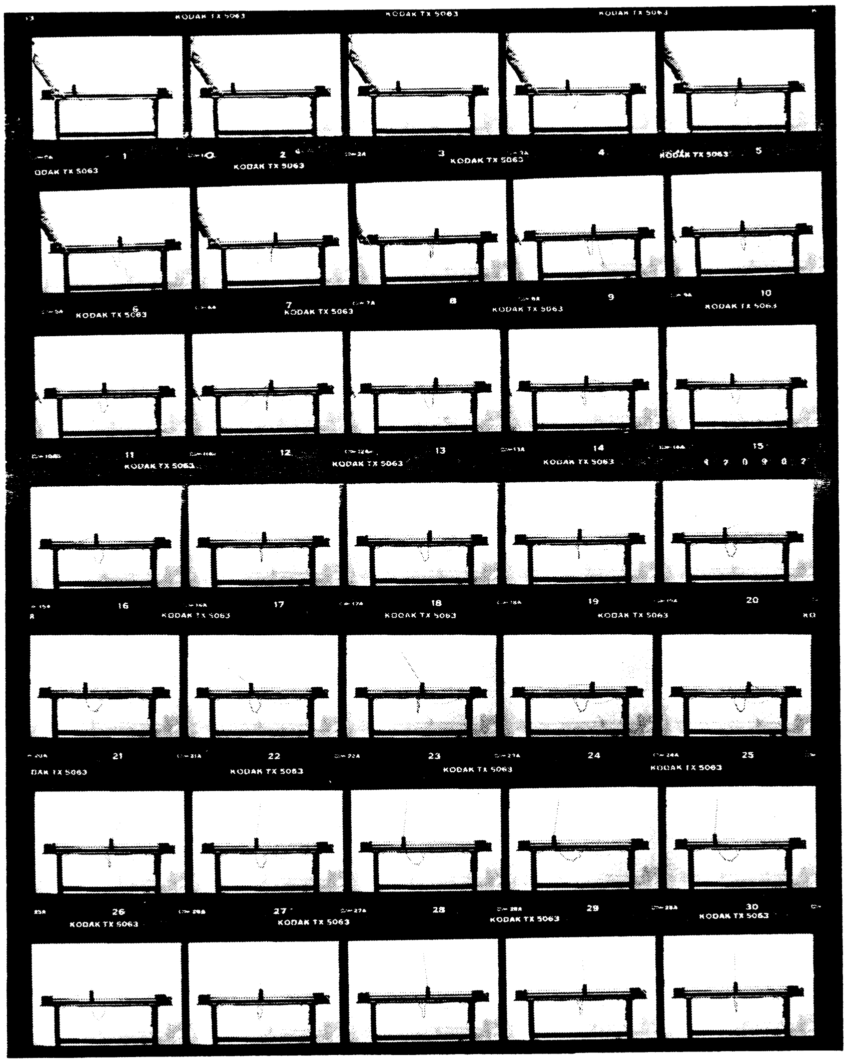


Photograph 1 Control of an inverted pendulum (1).

Next, the stagewise variable decreasing procedure using back-propagating error learning models is applied again to $\Lambda_s = \{x_2, x_3, x_4\}$. The following are obtained by using formulas (23), (24) and (25).

$$\Theta_3{}^{11} = 0.007 \tag{32}$$

$$\Theta_2{}^{13} = \min_j \Theta_2{}^{1j} (=0.021),\ j=2,3,4 \tag{33}$$



Photograph 2 Control of an inverted pendulum (2).

Therefore, the following is obtained.

$$\Theta_2^{13} > \Theta_3^{11} \tag{34}$$

The input variables are not eliminated, and the algorithm for rule 1 is completed in the second repeating calculation.

As for rule 2, the stagewise variable decreasing procedure using the same back-propagating error learning model is applied to eliminate input variable x_3, and the algorithm is completed in the second repeating calculation.

The inference rules obtained as a result are shown below.

$$\left.\begin{array}{l} R_1 : \text{IF } \boldsymbol{x}=(x_1, x_2, x_3, x_4) \text{ is } A_1 \\ \quad \text{THEN } y_1 = NN_1(x_2, x_3, x_4), \\ R_2 : \text{IF } \boldsymbol{x}=(x_1, x_2, x_3, x_4) \text{ is } A_2 \\ \quad \text{THEN } y_2 = NN_2(x_1, x_2, x_4) \end{array}\right\} \tag{35}$$

Photographs 1 and 2 show the control of a pendulum according to the inference rule in formula (35). Photograph 1 shows an action which flings a pendulum from the stable balancing point in approximately 10-15 msec/time of reasoning time and seizes it. The estimate, y_i^*, can be obtained from formula (28). The inverted pendulum certainly flings up where the truck is located on the rail. The pendulum can be flung by adding disturbance during the initial condition of control or during reasoning. Photograph 2 shows the control of the flinging of a pendulum where the angular of the pendulum was changed in the initial condition of the control.

5. CONCLUSION

Conventional fuzzy reasoning has problems. One of them is a tuning problem and the shape of fuzzy numbers and the labels of fuzzy variables in the antecedent and consequent parts of fuzzy reasoning rules must be changed so that the result of fuzzy reasoning becomes optimal. In the neural network driven fuzzy reasoning, when input and output data are given an optimal inference rule and membership function can be determined by using the non-linear characteristic and the learning characteristic of back-propagating error learning models. An experiment using an inverted pendulum was performed in order to confirm the usefulness of neural network driven fuzzy reasoning. The pendulum certainly flung up from the down position at the stable balancing point and came to a standstill. Since the inference rules of fuzzy control can be created by using the learning function of back-propagating error learning models in this method, a learning function can be applied to fuzzy control. In the future, it will be necessary to discuss a learning function which follows the movement of a reasoning environment that changes dynamically.

REFERENCE

1. K. Hirota: Movement of the Application of Fuzzy Control, Measure and Control, 28-11, pp.970-975, (1989).
2. E. H. Mamdani and S. Assilian: An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller, Int. J. Man-Mach. Stud., 17, pp.1-13, (1974).
3. Y. Tsukamoto: Fuzzy Reasoning Method, Measure and Control, 28-11, pp.946-952, (1989).
4. M. Maeda and S. Murakami: Self-Adjustment Fuzzy Controller, Proc. of the Society of Metrological and Automatic Control, 24-2, pp.191-197, (1988).
5. T. Amari: Mathematics of Neural Network, Industrial Library, (1978).
6. H. Asou: Neural Network Information Processing, Industrial Library, (1988).
7. H. Takagi and I. Hayashi: Artificial Neural Network Driven Fuzzy Reasoning, Int. J. Approximate Reasoning, (1991 in press).
9. D.E. Rumelhart, G.E. Hinton and R.J. Williams: Learning Representations by Back-propagating Errors, Nature, 323-9, pp.533-536, (1986).
10. M. Sugano: Fuzzy Control, Nikkan Industry Press, (1988).
11. D. Dubois and H. Prade: Fuzzy Sets and Systems, Theory and Applications, Academic Press, (1980).

12. Y.H. Pao: Adaptive Pattern Recognition and Neural Networks, Addison-Wesley, (1989).
13. M. Kawaguchi: Introduction to Multivariate Analysis 1, Morikita Press, (1973).