NN-Driven Fuzzy Reasoning

Hideyuki Takagi and Isao Hayashi
Matsushita Electric Industrial Co., Ltd., Moriguchi, Osaka, Japan

ABSTRACT

A new fuzzy reasoning that can solve two problems of conventional fuzzy reasoning by combining an artificial neural network (NN) and fuzzy reasoning is proposed. These problems are (1) the lack of design for a membership function except a heuristic approach and (2) the lack of adaptability for possible changes in the reasoning environment. The proposed fuzzy reasoning approach solves these problems by using the learning function and nonlinearity of an NN. First, the problems involved in conventional fuzzy reasoning and the NN used in this paper are identified. Then a proposed algorithm is formulated and a concrete explanation using realistic data is developed. An example structure of an NN-driven fuzzy reasoning system is given, and two applications of this method are presented. This new fuzzy reasoning is capable of automatic determination of inference rules and adjustment according to the time-variant reasoning environment because of the use of NN in fuzzy reasoning. This proposed method can be applied to NN modeling and AI and is considered from the standpoint of the explicit incorporation of knowledge into the NN structure.

KEYWORDS: ***fuzzy reasoning, neural network, membership functions***

INTRODUCTION

Fuzzy reasoning can express the qualitative aspect of human logic. Since it realizes the flexible reasoning corresponding to human logical reasoning, extensive research has been conducted into fuzzy reasoning. Its practical applications are now being seen not only in various control fields but also in AI and operation research. However, two problems of conventional fuzzy reasoning have not been solved yet: the method of determining membership functions and the adaptation of reasoning to the environment. The method proposed here is to solve these problems by using an artificial neural network, which we denote by NN. Since this method is an elemental technology, extensive application fields including not only control but also estimation,

Address correspondence to H. Takagi, Central Research Laboratories, Matsushita Electric Industrial Co., Ltd., 3-15 Yagumo-Nakamachi, Moriguchi, Osaka 570, Japan.

inference, prediction, and so on can be expected. The aim of this method and its formulation are described, and its effectiveness is explained through several applications.

FUZZY REASONING

Strong Points of Fuzzy Reasoning

The following describes the typical control rules using two-value logic.

AN EXAMPLE OF A TWO-VALUE LOGIC CONTROL RULE

IF the cornering angle is with 20°-40° and
the loading weight is more than 50 kg,
THEN the moving velocity = 3 − cornering angle/20.

The problem with this control rule is that it is inapplicable even if the loading weight is 49.9999 kg. Moreover, the control for a cornering of $20 \pm \alpha$ degrees becomes uncomfortable to drive because more than two control rules would be irregularly and alternately applied. There is no way to solve these problems except to chop up the given rules to accomplish smooth control. However, it is impossible to achieve control of perfect smoothness except with infinitely many control rules.

Fuzzy control employs the following rule:

AN EXAMPLE OF A FUZZY CONTROL RULE

IF the cornering angle is a *moderate* degree and
the loading weight is *heavy*,
THEN the moving velocity should be *gradually* decreased.

This is a control rule of human concept expressed in human words. Moreover, the control based on this fuzzy reasoning not only makes the control rules easier to use but also substantially decreases the number of rules.

To execute this inference rule, the corresponding relationship between the input numerical data obtained from sensors and the fuzzy sets such as "moderate," "heavy," and "gradually" must be determined. This mapping relationship is a membership function such as the one shown in Figure 1.

The main strong point of fuzzy reasoning is the accomplishment of distinctive separation between the "logic" and the "membership function." This logic constitutes the backbone of the rule, and the membership function deals with the fuzziness entangled in the logic. When an expert system based on

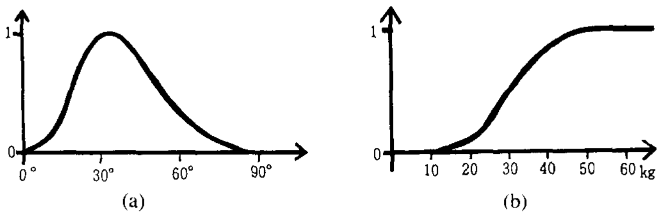


Figure 1. Examples of membership functions. (a) Membership function of "*moderate* cornering angle"; (b) membership function of "*heavy* loading weight."

two-value logic is constructed, it very often needs to reconstruct or adjust the rules. The main reason for this lies in the difficulty of fuzziness separation, and this calls for an adjustment of the logic itself. This is one of the reasons that rule construction based on two-value logic is difficult.

Problems with Fuzzy Reasoning

How should we design the membership function that is essential for fuzzy reasoning? The ordinary way is to assume an original membership function to be triangular or trapezoidal at the beginning. If this membership function is found to be unfit, heuristic tuning has to be tried. This means there is no straightforward method for designing a membership function. This is the first problem to overcome.

Since input/output variables are supposed to be independent in fuzzy reasoning, the membership function is designed variable by variable (Figure 2). However, in a case such as *IF the temperature is slightly higher and the humidity is lower, THEN the power has to be lowered slightly*, the temperature and humidity cannot be said to be completely independent. In this case, the membership function requires a curved surface in three-dimensional space consisting of the membership value axis and the temperature–humidity plane (Figure 3). Therefore, when a variable number increases, it is nearly impossible to design a membership function in multidimensional space by using experience and intuition.

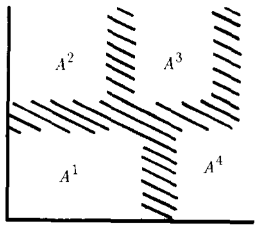


Figure 2. Conventional fuzzy rule partitions.

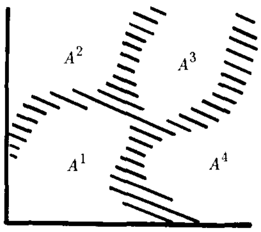


Figure 3. Proposed fuzzy rule partitions.

The second problem of fuzzy reasoning is the lack of learning functions. This makes the optimization of reasoning for a time-variant environment such as seasonal changes impossible. In the application of fuzzy reasoning to mass-produced home appliances, it is hard for the producer to adjust them to individual consumer preferences and environments and then ship them. The mass producer has no choice but to adjust them to an average value at the factory. If fuzzy reasoning could have a low-cost learning function, the consumer could tune fuzzy inference rules to fit individual preferences and environments. This could result in an "appliance like a pet that can be trained to read your mind."

PROBLEMS WITH FUZZY REASONING

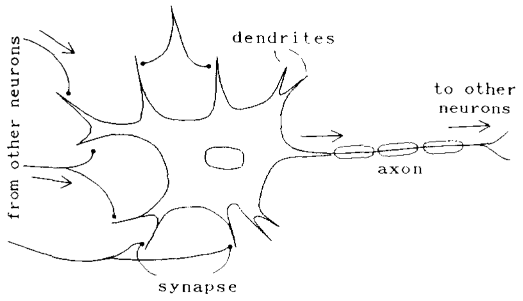
1. The lack of a definite method to determine the membership function
2. The lack of a learning function or adaptability

COMBINATION OF NEURAL NETWORKS

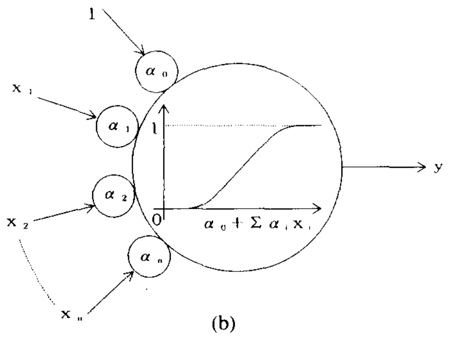
Introduction to Neural Networks

The brain's mechanism of information processing has been analyzed from the point of view of mathematics and engineering (McCulloch and Pitts [1], Rosenblatt [2]). It is known as associative memory or a learning machine. Recently developed "back-propagation" (Rumelhart et al. [3]) is an effective learning algorithm of multilayer perceptron and has directed attention to the information processing capability of artificial neural networks. This algorithm has been widely employed for various pattern classifications or inference problems expressed in terms of nonlinear functions. Reported in this paper is the application of reasoning and the learning function of the NN for nonlinear problems to fuzzy reasoning. This is started with mathematical analysis made on the NN.

The biological neuron receives signals from various other neurons through each synapse. It fires neural pulses if these input signals exceed a certain



(a)



(b)

Figure 4. (a) Biological neuron; (b) neuron model.

threshold value and will not fire for signals below the threshold. An example of engineering models expressing this biological concept is the multi-input, single-output nonlinear circuit shown in Figure 4b. Equation (1) expresses the relationship between the input **x** and the output y of this mathematical model.

$$y_i = f\left(\sum_{j}^{N} \alpha_{ji} x_j + \alpha_{0i}\right) \tag{1}$$

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

where α is a connectional weight for reflecting the synapse to the model and $f()$ is a sigmoid function expressed in Eq. (2). The reason for employing a sigmoid function is that it is an on/off function and is differentiable; that is the essential condition for back-propagation.

The NN is then a number of these neural models connected to each other to constitute a network. The k-layer perceptron trained by the back-propagation algorithm is employed as the NN in this paper. However, the proposed method does not necessarily limit the application of the NN to the k-layer perceptron.

The following is the definition of NN employed in this paper.

Definition

(*a*) *The k-layer perceptron is a model consisting of one output layer and k − 1 hidden layers.*

(*b*) *The I/O relationship of NN is expressed by the equation*

$$\mathbf{y} = NN(\mathbf{x}) \tag{3}$$

(*c*) *The term k-layers*$[u_0 \times u_1 \times \cdots \times u_k]$ *expresses the model size, where* u_i *is the number of neurons in the input layer, hidden layer(s), and output layer, and k is the number of layers.*

(*d*) *Each input layer and hidden layer has an extra unit of constant 1 in addition to those specified by (c). This unit has no connection to the lower-layer neurons and has connections to the upper-layer neurons.*

(*e*) *All neuron models in the neighboring layers are connected, but there should be no connections within layers nor jump connections between layers.*

Term (d) in the above definition is to express α_{0i} in Eq. (1). Term (e) is an explanation for the model used in the following section on formulation and is not essential for the NN. As an example, Figure 5 shows a 3-layers[3 × 2 × 2 × 2] NN.

An NN can attain the model identification well suited for training data because of its nonlinearity and learning function. This may in turn cause a case where the above condition is very well suited to the learning data but not to the evaluation data. Therefore, it is important to endow the learning data with enough deviation to derive a model that performs well.

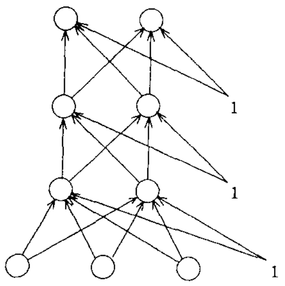


Figure 5. Example structure of neural network.

Introduction of Neural Network into Fuzzy Reasoning

Two problems inherent to fuzzy reasoning can be handled by the NN. The difference between these two technologies has to do with whether the logic is explicit or implicit. For fuzzy reasoning, stable reasoning can always be attained despite data deviations because the backbone logic is manifested as a rule in IF–THEN form. On the other hand, however, the rule cannot be expressed unless the logic is identified. Since the NN self-organizes the mapping relationship by learning, it is applicable even for unknown logical relationships. Moreover, it is capable of expressing any nonlinear relationship because it is itself nonlinear. However, such a nonparametric method requires a large amount of data. The NN might output a deviated answer if deviated learning data were supplied. From these differences, fuzzy reasoning is employed mainly for well-identified logic cases such as the control, and the NN is used mainly for unidentified recognition rules such as pattern recognition.

Consider now the problem of designing a membership function. Even if it is not clear, the inference rule can be automatically derived from the fuzzy rule partition of learning data using the learning function of the NN. Furthermore, an adaptive modification of the membership function is possible, because the NN has a learning function.

NN-driven fuzzy reasoning is fuzzy reasoning that is driven by an artificial neural network. The fundamental concept is the employment of fuzzy reasoning for the fundamental reasoning and the NN for the determination. Furthermore, the adaptive modification of the membership function becomes possible, replacing conventional experience and intuition.

FORMULATION OF NN-DRIVEN FUZZY REASONING

Outline

The outline of the proposed method is explained by taking as an example the control carried out by two inputs, x_1 and x_2, derived from two sensors. The algorithm consists of three major parts:

1. The partition of inference rules
2. The identification of IF parts (the determination of a membership function)
3. The identification of THEN parts (the determination of the amount of control for each rule)

The first part is the determination of the number of fuzzy inference rules and the combination of data belonging to each rule. These data are grouped by a

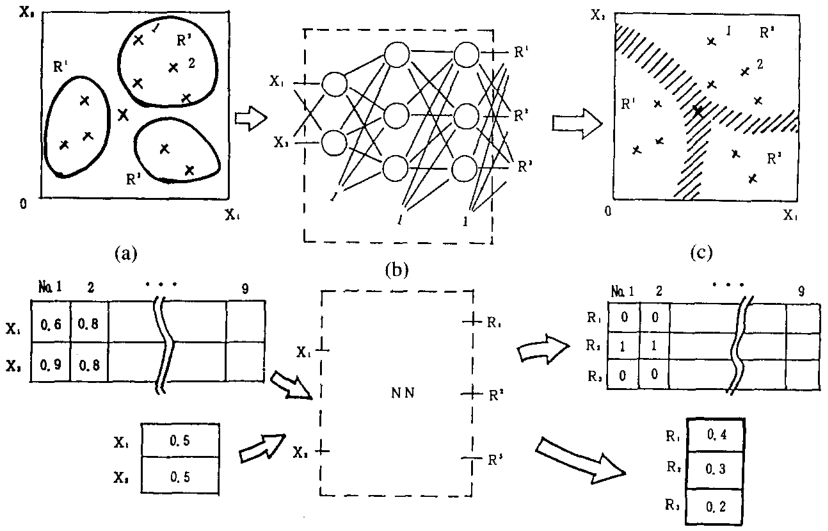


Figure 6. Design of membership functions using an NN. (a) Input data space; (b) NN that determines membership function; (c) data space that is partitioned for fuzzy rules.

clustering method, and the number of groups equals the number of inference rules (Figure 6a).

The second part is the attribution of arbitrary input for each rule (corresponding to the upper group) to each rule. This derives the membership function for each rule and corresponds to the identification of IF parts (the condition parts) of the rule. It should be noticed that this procedure combines all the fuzzy variables (x_1 and x_2 in Figure 6's case) in the IF part and constitutes a hypersurface membership function. We have used the NN for this determination. The NN can form a continuous membership function governing entire rules internally. It was proved that an arbitrary continuous function can be constituted by an NN having at least one hidden layer (Funahashi [4]).

The third part of the algorithm is the determination of THEN parts (the conclusion parts). The NN is supervised by the learning data and the control value for each rule as in (2). Fundamentally, another control method could be used, but the NN has better adaptability considering the later described systems.

Formulation

The detailed formulation of the NN is explained in the following example of fuzzy modeling included in fuzzy control (Hayashi and Takagi [5, 6]).

The THEN part in the system control is responsible for inferring the exact control value. One of these methods is the fuzzy modeling taking the form of "IF $\mathbf{x} \in A^s$, THEN $\mathbf{y} = u(\mathbf{x})$," wherein $\mathbf{x}$ is an input vector, A^s is the fuzzy set of the sth partitioned rule spaces (Figure 2), and $u(\,)$ appoints an inference function for the control operation. In NN-driven fuzzy reasoning, A^s is partitioned by an NN as shown in Figure 3, and $u(\,)$ is the NN itself. The following shows each step for this.

STEP 1 Define the observed value as y_i, the output, and the input variables as x_j, $j = 1, 2, \ldots, k$. In this step the x_j, $j = 1, 2, \ldots, m$, $m \le k$, related to the observed value y are selected by the NN. This is done by the backward elimination method using the sum of squared errors as a cost function. This is to eliminate the input variables attributed to noise and to select only those input variables that have significant correlations to the observed values.

STEP 2 The input/output data $(\mathbf{x}_i, y_i)$ are then divided into training data (TRD of n_t) and checking data (CHD of n_c) for model estimation, where $n = n_t + n_c$.

STEP 3 The partition of the TRD is found by a clustering method. The best number of partitions is decided in view of the distance between the clusters in a clustering dendrogram. Each of the TRD divided into r groups is expressed by R^s, $s = 1, 2, \ldots, r$, and the TRD of R^s are expressed by $(\mathbf{x}_i^s, y_i^s)$, where $i = 1, 2, \ldots, (n_t)^s$, and $(n_t)^s$ are TRD numbers in each R^s. The division of m-dimensional space into r here means that the number of inference rules is set to be r.

STEP 4 This step is the identification of the constitution of each IF part in $\mathrm{NN}_{\mathrm{mem}}$ (NN generating the membership functions). If $\mathbf{x}_i$ are the values for the input layer, w_i^s in the following are assigned as the supervised data for the output layer:

$$w_i^s = \begin{cases} 1, & \mathbf{x}_i \in R^s \\ 0, & \mathbf{x}_i \notin R^s \end{cases} \qquad i = 1, \ldots, n_t;\ s = 1, \ldots, r$$

The learning of $\mathrm{NN}_{\mathrm{mem}}$ is conducted so that these w_i^s can be inferred from the input $\mathbf{x}$. Thus, the $\mathrm{NN}_{\mathrm{mem}}$ becomes capable of inferring the degree of attribution $\hat{w}_i^s$ of each training data item $\mathbf{x}_i$ to R^s. We have defined the membership function of the IF part as the inferred value $\hat{w}_i^s$ that is the output of the learned $\mathrm{NN}_{\mathrm{mem}}$, that is,

$$\mu_{A^s}(\mathbf{x}_i) \equiv \hat{w}_i^s, \qquad i = 1, 2, \ldots, n$$

STEP 5 This step is the identification of each THEN part. The structure of the THEN part of each inference rule is expressed by the input/output relationship. The TRD input $x_{i1}^s, \ldots, x_{im}^s$ and the output value y_i^s, $i = 1, 2, \ldots, (n_t)^s$, are assigned to the input and output of the NN_s. This NN_s is the NN of the THEN part in R^s. The training of NN_s is so conducted that the control value can be inferred. The CHD input values $x_{i1}, \ldots, x_{im}$, $i = 1, 2, \ldots, n_c$, are substituted in the NN thus obtained to obtain the sum Θ_m^s of squared errors.

$$\Theta_m^s = \sum_{i=1}^{n_c} \{y_i - u_s(\mathbf{x}_i) \cdot \mu_{A^s}(\mathbf{x}_i)\}^2 \tag{4}$$

This estimated value $u_s(\mathbf{x}_i)$ is obtained as the output of NN_s. There is another idea to calculate Θ^s with the weight; this is

$$\Theta_m^s = \sum_{i=1}^{n_c} \mu_{A^s}(\mathbf{x}_i)\{y_i - u_s(\mathbf{x}_i) \cdot \mu_{A^s}(\mathbf{x}_i)\}^2 \tag{4$'$}$$

We have used the following index to decide the best iteration number of NN learning and prevent overlearning.

$$I^s = \frac{n_c}{(n_t)^s + n_c} \sum_{i=1}^{(n_t)^s} \{y_i - u_s(\mathbf{x}_i)\}^2 + \frac{(n_t)^s}{(n_t)^s + n_c} \sum_{j=1}^{n_c} \{y_j - u_s(\mathbf{x}_j) \cdot \mu_{A^s}(\mathbf{x}_j)\}^2$$

If the NN_s has overlearned data, the error of the TRD becomes small but the error of the CHD becomes large. Therefore the number of iterations that gives the smallest I^s, is the best.

STEP 6 This is the simplification of THEN parts by a backward elimination method. Among the m input variables of an NN that infers the control values of the THEN parts for each inference rule, one input variable x^p is arbitrarily eliminated, and the NN of each THEN part is trained by using the TRD as in step 5. Equation (5) gives the squared error Θ_{m-1}^{sp} of the control value of the sth rule in the case of eliminating x^p. This Θ_{m-1}^{sp} can be estimated using the CHD:

$$\Theta_{m-1}^{sp} = \sum_{i=1}^{n_c} \{y_i - u_s(\mathbf{x}_i) \cdot \mu_{A^s}(\mathbf{x}_i)\}^2, \qquad p = 1, 2, \ldots, m \tag{5}$$

By comparing Eqs. (4) and (5), if

$$\Theta_m^s > \Theta_{m-1}^{sp} \tag{6}$$

the significance of the eliminated input variables x^p can be considered minimal, and x^p can be discarded.

STEP 7 The same operations as those performed in step 5 are carried out for the remaining $m - 1$ input variables. Steps 5 and 6 are cyclically repeated in the succeeding step until Eq. (6) would not hold for any remaining input variables. The model that gives the minimum Θ^s value is the best NN.

Thus, steps 1–7 determine the IF parts and THEN parts of each inference rule. The system identification process for the fuzzy model is then completed.

STEP 8 The following equation can derive the final control value y_i^*:

$$y_i^* = \frac{\sum_{s=1}^{r} \mu_{A^s}(\mathbf{x}_i) \cdot \overline{u_s(\mathbf{x}_i)}}{\sum_{s=1}^{r} \mu_{A^s}(\mathbf{x}_i)}, \qquad i = 1, 2, \ldots, n \tag{7}$$

where $\overline{u_s(\mathbf{x}_i)}$ is an inferred value obtained when CHD is substituted in the best NN obtained in step 7.

Detailed Supplemental Explanations

Among the steps shown above, a concrete explanation is supplemented here for step 2 by which the nonlinear membership function can be automatically determined.

It is natural to consider the application of the same control rule to input data that share high similarity. Therefore, the training data are clustered in this way in step 3 (Figure 6a). Each of these clustered groups corresponds to one of the rules such as "**IF** x_1 is small and x_2 is large, **THEN** . . . ." In the case shown in Figure 6a, three rules including R^1 to R^3 exist. These rules make perfect fits at the typical points (training data points) in these clusters. A gradual fitting to multiple rules has to be conducted as one nears the boundary region. The degree of attribution is a membership value. This shape could be the one shown in Figure 6c, which is a top view; the crosshatched area is a region in which the membership functions intersect.

The procedure to derive such a membership function corresponds to step 4. Here we propose to use the NN shown in Figure 6b. The variables of the IF parts (x_1 and x_2 in this case) are assigned to the input. At the output layer, the rule number neuron belonging to input variables and the others are expressed by 1 and 0, respectively. A strong point of the NN is the correspondence of an analogous output to an analogous input. Only the $\times$ points in Figure 6a are used for the training, but the similar output (i.e., the similar control rule) is obtained for the input data that is neighbor to the training data. Moreover, an

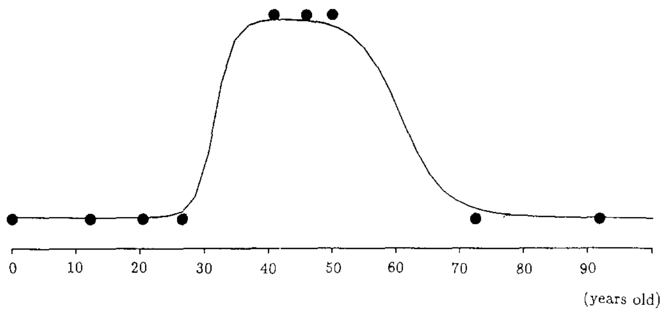


Figure 7. Membership function of "*middle age.*"

NN that has finished its training outputs the best balanced attribution value for each rule when those input data are in the intersecting rules. This is the value of membership functions; in other words, this NN comprehends all the membership functions for each rule it contains.

Figure 7 shows an example of membership functions internally formulated by an NN. As a result of NN learning from nine data consisting of "*apparently not middle-aged*" and "*apparently middle-aged*," this NN outputs the membership function "*middle-aged*" (Figure 7). The variable is "age," which is one-dimensional. The membership function shown in Figure 7 can be easily designed by calling on experience and intuition, which would be useless for two-dimensional or multidimensional hypercurved surfaces.

Examples of Concrete Application

Following the above procedures, this formulated NN-driven fuzzy reasoning is applied to the simple previously reported numerical data (Kang and Sugeno [7], Kondo [8]). These data were made from $y = (1.0 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5})^2$ and random noise x_4 (Kondo [8]).

STEPS 1, 2 Table 1 shows the input/output data. Items 1-20 are the training data (TRD) and 21-40 are the checking data (CHD); thus $n_t = n_c = 20$, and $k = 4$. Table 2 shows the result of the training for 15,000 iterations with a 3-layers[$4 \times 3 \times 3 \times 1$] model that uses all variables and a 3-layers[$3 \times 3 \times 3 \times 1$] model for the selection of input variables. Both training and checking data were used for these learning models.

The estimation performance of the model that eliminated x_4 was similar to that of the model that used all variables relatively. This means that input variable x_4 is negligible. We abandoned x_4 in the succeeding experiment.

STEP 3 The TRD are partitioned by using a conventional clustering method. The training data thus partitioned are shown in Table 3.

Table 1. Example Input/Output Data

Training Data (TRD)						Checking Data (CHD)					
No.	y	x_1	x_2	x_3	x_4	No.	y	x_1	x_2	x_3	x_4
1	11.110	1	3	1	1	21	9.545	1	1	5	1
2	6.521	1	5	2	1	22	6.043	1	3	4	1
3	10.190	1	1	3	5	23	5.724	1	5	3	5
4	6.043	1	3	4	5	24	11.250	1	1	2	5
5	5.242	1	5	5	1	25	11.110	1	3	1	1
6	19.020	5	1	4	1	26	14.360	5	5	2	1
7	14.150	5	3	3	5	27	19.610	5	1	3	5
8	14.360	5	5	2	5	28	13.650	5	3	4	5
9	27.420	5	1	1	1	29	12.430	5	5	5	1
10	15.390	5	3	2	1	30	19.020	5	1	4	1
11	5.724	1	5	3	5	31	6.380	1	3	3	5
12	9.766	1	1	4	5	32	6.521	1	5	2	5
13	5.8700	1	3	5	1	33	16.000	1	1	1	1
14	5.406	1	5	4	1	34	7.219	1	3	2	1
15	10.190	1	1	3	5	35	5.724	1	5	3	5
16	15.390	5	3	2	5	36	19.020	5	1	4	5
17	19.680	5	5	1	1	37	13.390	5	3	5	1
18	21.060	5	1	2	1	38	12.680	5	5	4	1
19	14.150	5	3	3	5	39	19.610	5	1	3	5
20	12.680	5	5	4	5	40	15.390	5	3	2	5

Source: Kang and Sugeno [7]; Kondo [8].

Table 2. Results of Backward Elimination Using a Neural Network

	Sum of Squared Errors
When all variables are used	0.0007
When x_1 is eliminated	0.3936
When x_2 is eliminated	0.1482
When x_3 is eliminated	0.0872
When x_4 is eliminated	0.0019

Table 3. Rule Partition of Training Data

Control Rule	Training Data Numbers
R^1	1, 2, 3, 4, 5, 11, 12, 13, 14, 15
R^2	6, 7, 8, 9, 10, 16, 17, 18, 19, 20

Table 4. Membership Value for Rule R^s

Training Data				Membership Value	
No.	x_1	x_2	x_3	Rule 1	Rule 2
1	1	3	1	0.9970	0.0031
2	1	5	2	0.9972	0.0028
3	1	1	3	0.9972	0.0028
4	1	3	4	0.9973	0.0027
5	1	5	5	0.9974	0.0026
6	5	1	4	0.0028	0.9971
7	5	3	3	0.0028	0.9972
8	5	5	2	0.0027	0.9972
9	5	1	1	0.0027	0.9973
10	5	3	2	0.0027	0.9973
11	1	5	3	0.9973	0.0028
12	1	1	4	0.9973	0.0027
13	1	3	5	0.9974	0.0026
14	1	5	4	0.9973	0.0027
15	1	1	3	0.9972	0.0028
16	5	3	2	0.0027	0.9973
17	5	5	1	0.0027	0.9973
18	5	1	2	0.0027	0.9973
19	5	3	3	0.0028	0.9972
20	5	5	4	0.0029	0.9971

STEP 4 The 3-layers[3 $\times$ 3 $\times$ 3 $\times$ 2] model is trained for 5000 times to infer $w_i^s \in \{0, 1\}$, that is, the degree of attribution of the training data $\mathbf{x}_i$, $i = 1, 2, \ldots, 20$, to A^s, by the value of $\hat{w}_i^s \in [0, 1]$. By this training, the fuzzy number A^s in the IF parts is derived. Table 4 shows the membership values of the fuzzy number A^s of IF parts for the control rule R^s.

STEP 5 The inference formula for determining the control value for the THEN parts in various control rules is identified. Table 5 shows the output errors Θ_3^s derived after the 20,000 iterations of training of the 3-layers[3 $\times$ 8 $\times$ 8 $\times$ 1] model.

STEPS 6 AND 7 The sum of the squared errors Θ_2^{sp} when one of the arbitrary

Table 5. Output Errors

Control rule 1: Θ_3^1	27.86
Control rule 2: Θ_3^2	1.93

Table 6. Output Errors After Elimination of Variables

	Rule 1	Rule 2
From Table 5	$\Theta_3^1 = 27.86$	$\Theta_3^2 = 1.93$
When x_1 is eliminated	$\Theta_2^{11} = 42.84$	$\Theta_2^{21} = 0.93$
When x_2 is eliminated	$\Theta_2^{12} = 74.71$	$\Theta_2^{22} = 119.61$
When x_3 is eliminated	$\Theta_2^{13} = 55.27$	$\Theta_2^{23} = 73.28$

input variables is removed from the IF-parts model with control rule R^s is derived. This sum, shown in Table 6, was obtained for control rules R^1 and R^2 after the learning of the 3-layers[$2 \times 8 \times 8 \times 1$] model for 10,000-20,000 iterations.

Comparing step 5 and step 6 for each control rule, the following conditions are seen to exist:

$$\text{All of } \Theta_2^{1p} > \Theta_3^1(=27.86)$$

and

$$\Theta_2^{21}(=0.93) < \Theta_3^2(=1.93)$$

Therefore, the NN of step 5 is designated as the conclusion parts model for control rule 1. The computation is continued for control rule 2, and is terminated by the repeated computations in step 2. Thus, the resulting NN, which has the input (x_2, x_3), is designated as the conclusion model. Therefore, the obtained fuzzy model is expressed by the following:

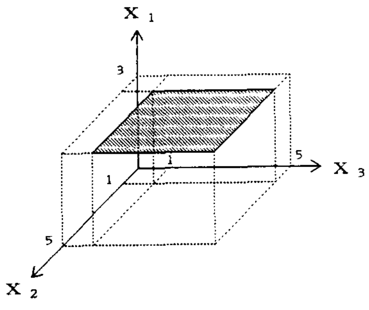
$$R^1: \quad \textbf{IF } \mathbf{x} = (x_1, x_2, x_3) \textit{ is } A^1, \textbf{ THEN } y^1 = NN_1(x_1, x_2, x_3)$$

$$R^2: \quad \textbf{IF } \mathbf{x} = (x_1, x_2, x_3) \textit{ is } A^2, \textbf{ THEN } y^2 = NN_2(x_2, x_3)$$

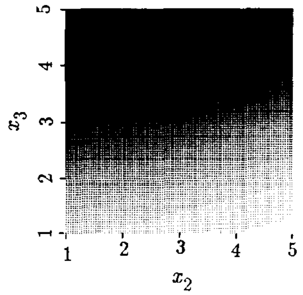
Figure 8 shows the plot of rule clustering of R^s in the $x_2 x_3$ plane, and Table 7 lists the y_i^* of Eq. (7).

SYSTEM CONSTRUCTION

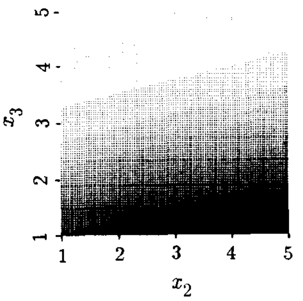
We now can execute the NN-driven fuzzy reasoning using the above formulation. Figure 9 shows an example of this system executing this reasoning. The $\mathrm{NN}_{\mathrm{mem}}$ at the left end is the NN of Figure 6b generating the membership functions corresponding to the IF parts derived in step 4. NN_1-NN_r are the NNs of the THEN parts prepared in steps 6-8. This system weighs the output of the THEN part by the membership values of the IF parts and computes the final output value (step 9).



(a)



(b)



(c)

Figure 8. Hypercurved plane of membership function. (a) Display area of membership functional plane for (b) and (c). (b) Membership function of rule 1. (c) Membership function of rule 2.

Table 7. Output of Trained NN-Driven Fuzzy Reasoning

Training Data					Checking Data				
No.	y	y^*	W_i^1	W_i^2	No.	y	y^*	W_i^1	W_i^2
1	11.110	11.136	0.9970	0.0031	21	9.545	8.882	0.9974	0.0027
2	6.521	6.534	0.9972	0.0028	22	6.043	6.140	0.9973	0.0027
3	10.190	10.210	0.9972	0.0028	23	5.724	5.712	0.9973	0.0028
4	6.043	6.140	0.9973	0.0027	24	11.250	10.547	0.9971	0.0030
5	5.242	5.370	0.9974	0.0026	25	11.110	11.136	0.9970	0.0031
6	19.020	18.995	0.0028	0.9971	26	14.360	14.334	0.0027	0.9972
7	14.150	14.134	0.0028	0.9972	27	19.610	19.061	0.0028	0.9972
8	14.360	14.334	0.0027	0.9972	28	13.650	13.918	0.0029	0.9971
9	27.420	27.373	0.0027	0.9973	29	12.430	12.293	0.0030	0.9969
10	15.390	15.383	0.0027	0.9973	30	19.020	18.995	0.0028	0.9971
11	5.724	5.712	0.9973	0.0028	31	6.380	7.178	0.9972	0.0028
12	9.766	9.791	0.9973	0.0027	32	6.521	6.534	0.9972	0.0028
13	5.8700	5.747	0.9974	0.0026	33	16.000	11.239	0.9969	0.0032
14	5.406	5.450	0.9973	0.0027	34	7.219	9.018	0.9971	0.0029
15	10.190	10.210	0.9972	0.0028	35	5.724	5.712	0.9973	0.0028
16	15.390	15.383	0.0027	0.9973	36	19.020	18.995	0.0028	0.9971
17	19.680	19.652	0.0027	0.9973	37	13.390	13.892	0.0030	0.9970
18	21.060	21.046	0.0027	0.9973	38	12.680	12.672	0.0029	0.9971
19	14.150	14.134	0.0028	0.9972	39	19.610	19.061	0.0028	0.9972
20	12.680	12.672	0.0029	0.9971	40	15.390	15.383	0.0027	0.9973

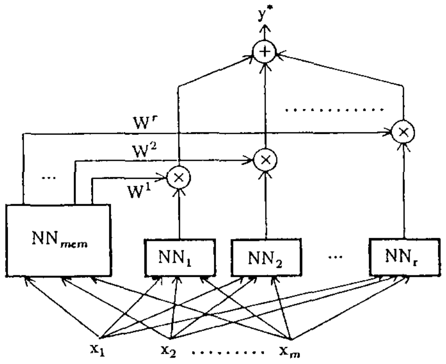


Figure 9. Block diagram of NN-driven fuzzy reasoning system. NN_{mem}, NN that decides membership values of all rules; NN_1-NN_r, NNs that determine control values and output y_i for ith rule; y^*, final control value; x_j, input variable; W^s, membership value.

This fundamental system can be developed and expanded. For instance, if the input is not a single defined value but is a fuzzy number, the plural inputs of the NN can be adopted for one of the fuzzy number inputs. Likewise, if the output value is a fuzzy number, the plural outputs can be applied.

This construction is highly suitable for parallel processing, and this is another strong point of this system. Each NN corresponding to the IF part and THEN part of each ruel can be independently processed; furthermore, the NN itself is suitable for parallel processing.

APPLICATIONS

The effectiveness of NN-driven fuzzy reasoning is demonstrated by two reasoning problems. One is the estimation of COD (chemical oxygen demand) density in Osaka Bay, and the other is the estimation of the roughness of a ceramic surface finished by a cup-shaped diamond whetstone.

Estimation of COD Density in Osaka Bay

The results of COD density measurements in Osaka Bay conducted between April 1976 and March 1979 were used as input/output data for inference conducted by NN-driven fuzzy reasoning. The input/output variables are as follows:

y	COD density, ppm
x_1	water temperature, °C
x_2	transparency, m
x_3	dissolved oxygen density, ppm
x_4	salty density, %
x_5	filtered COD density, ppm

Fujita and Koi [9] previously reported on the estimation of COD density y by using the GMDH method. They first estimated the filtered COD density x_5 from a diffusion simulation model. Next they estimated output COD density y by using this x_5 and other input variables employed for the GMDH (Group Method of Data Handling) model.

We employed the same data for estimating the COD density y by NN-driven fuzzy reasoning. The model was obtained by the experiments conducted under the following conditions.

1. Thirty-two data points acquired during a period from April 1976 to December 1978 were used for the estimation. Twelve more data points obtained during the period from January to December 1979 were employed for the evaluation. The result of the backward-elimination experi-

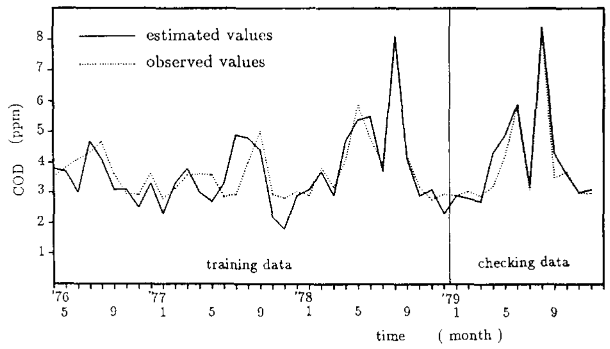


Figure 10. Estimation of COD density in Osaka Bay by NN-driven fuzzy reasoning.

ment showed that it was necessary to employ all input variables for estimation.

2. The structure of the NN model employed in the succeeding experiment was as follows:

For the determination of IF-part structure: 3-layers$[5 \times 12 \times 12 \times 2]$
For the determination of THEN-part structure:
3-layers$[m \times 12 \times 12 \times 1]$, $m = 5, 4, \ldots,$

3. There were 1500–2000 iterations of learning. The resulting NN-driven fuzzy reasoning was as follows:

$$R^1: \quad \textbf{IF } \mathbf{x} = (x_1, \ldots, x_5) \textit{ is } A^1, \textbf{ THEN } y^1 = NN_1(x_1, x_2, x_3, x_4, x_5)$$
$$R^2: \quad \textbf{IF } \mathbf{x} = (x_1, \ldots, x_5) \textit{ is } A^2, \textbf{ THEN } y^2 = NN_2(x_1, x_2, x_3, x_5)$$

Figure 10 shows the estimated value y_i^* and the observed data.

Table 8 shows the comparison between the estimated COD density derived by GMDH and the estimated COD density obtained by the proposed method using the model evaluation index $D = \left[\sum_i (y_i - y_i^*)^2\right]^{1/2}$.

Table 8. Results of Evaluation

	Training Data	Checking Data
GMDH [8]	3.63	2.04
Proposed	3.52	1.58

Estimation of Roughness of a Finished Ceramic Surface

The next is an example of application of NN-driven fuzzy reasoning for estimating the roughness of a surface finished by a diamond grinding wheel. The input/output variables were

y	roughness of ceramic surface, μm
x_1	rim velocity of diamond grinding wheel, m/min
x_2	moving velocity of ceramic plane, mm/min
x_3	cutting depth of diamond grinding wheel, mm
x_4	diamond size
x_5	concentration of diamond grinding wheel

Various whetted surface conditions were assumed by using the above input variables, and the ceramic plane was whetted by using a machining center. The finished ceramic surface was measured by using surface-roughness-measuring equipment. On the other hand, the roughness y was estimated by our proposed method.

Experiments were conducted under the following conditions for obtaining an inference model.

1. Thirteen data points with the following large deviations were used for the estimation:

$$S^2 = \sum_{i,j} \left(\frac{x_{ij} - \overline{x}_j}{\sigma_j} \right)^2$$

$$\sigma_j^2 = \frac{1}{n} \sum_{i}^{n} (x_{ij} - \overline{x}_j)^2$$

2. The remaining eight data points were used for the evaluation.
3. The structure of the NN and the number of learning iterations were the same as those in the Osaka Bay problem.

From the experimental results we obtained the following NN-driven fuzzy reasoning:

R^1: **IF** $\mathbf{x} = (x_1, \ldots, x_5)$ *is* A^1, **THEN** $y^1 = NN_1(x_1, x_2, x_4, x_5)$

R^2: **IF** $\mathbf{x} = (x_1, \ldots, x_5)$ *is* A^2, **THEN** $y^2 = NN_2(x_1, x_2, x_3, x_4, x_5)$

R^3: **IF** $\mathbf{x} = (x_1, \ldots, x_5)$ *is* A^3, **THEN** $y^3 = NN_3(x_2, x_4)$

Figure 11 shows the estimated value y_i^* and the observed data.

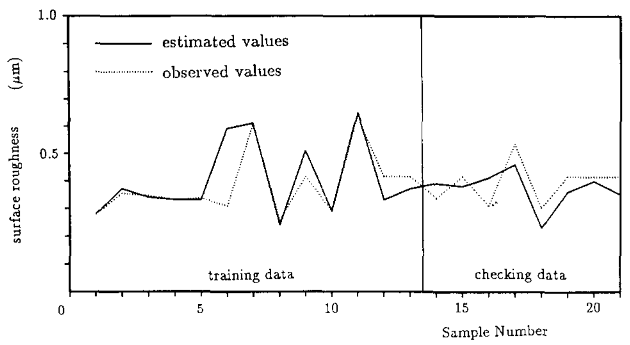


Figure 11. Estimation of roughness of whetted ceramic surface by NN-driven fuzzy reasoning.

Despite the limited quantity of available data, these two applications consistently show results better than those obtained by conventional methods.

CONCLUSION

This paper reports the formulation of the determination of fuzzy inference rules and the method of fuzzy reasoning using NN models. The most significant features are the feasibility of (1) automatic partition of a fuzzy rule and the best design of membership function and (2) the automatic adjustment of a membership function for a change of environment achieved by using the learning function of an NN. The experiments proved that the proposed system performed better than conventional reasoning systems.

Problems for future consideration include the verification of these results in extensive application fields. For example, the first point could be the automatic preparation of a fuzzy inference rule from the control data manually derived by an expert. The other example could be the automated adaptation of a fuzzy reasoning rule when fuzzy reasoning developed for environment A is to be applied to environment B.

The above considerations were made from the standpoint of fuzzy reasoning. From the standpoint of NNs, this proposed method corresponds to the explicit incorporation of knowledge into the NN structure as a form of fuzzy inference rule. Ways in which knowledge can be incorporated into the NN to improve its capability are being discussed. The conventional methods to incorporate knowledge are (1) the reflection of those on the input pattern, (2) the influence on performance by prewiring the network, and (3) the combination of NNs having different functions. The NN described here includes knowledge expressed in IF-THEN form. This expression is of far higher grade than conventional methods, so there is a possibility to realize powerful NN capability. Furthermore, development of the NN in the AI field as a neural expert system can be also expected.

ACKNOWLEDGMENT

We wish to acknowledge the help of Mr. H. Nomura in our laboratories. The experiments for evaluation were done with his help. We would also like to thank Dr. K. Nagasaka of the University of Osaka Prefecture for his offer of the raw data in the ceramics problem.

References

1. McCulloch, W. S., and Pitts, W., A logical calculus of the ideas imminent in nervous activity, *Bull. Math. Biophys.* 5, 115–133, 1943.
2. Rosenblatt, F., The perception: a probabilistic model for information storage and organization in the brain, *Psychol. Rev.* 65(6), 386–408, 1958.
3. Rumelhart, D. E., Hinton, G. E., and Williams, R. J., Learning representations by back-propagating errors, *Nature* 323(9), 533–536, 1986.
4. Funahashi, K., On the approximate realization of continuous mappings by neural networks, *Neural Networks* 2(3), 183–192, 1989.
5. Hayashi, I., and Takagi, H., Formulation of fuzzy reasoning by neural network, *Proceedings of the 4th Fuzzy System Symposium*, 55–60, May 1988 (in Japanese).
6. Takagi, H., and Hayashi, I., Artificial_neural_network-driven fuzzy reasoning, *Proceedings of an International Workshop on Fuzzy System Applications* (IIZUKA-88), 217–218, August 1988.
7. Kang, G. T., and Sugeno, M., Fuzzy modelling, *Trans. Soc. Instrum. Control Eng.* 23(6), 650–652, 1987 (in Japanese).
8. Kondo, T., Revised GMDH algorithm estimating degree of the complete polynomial, *Trans. Soc. Instrum. Control Eng.* 22(9), 928–934, 1986 (in Japanese).
9. Fujita, S., and Koi, H., Application of GMDH to environmental system modelling and management, in *Self-Organizing Methods in Modeling: GMDH Type Algorithms* (S. J. Farlow, Ed.), Statistics Textbooks and Monographs Ser., Vol. 54, Marcel Dekker, New York, 257–275, 1984.