

A simple but powerful heuristic method for generating fuzzy rules from numerical data

Ken Nozaki, Hisao Ishibuchi*, Hideo Tanaka

Department of Industrial Engineering, University of Osaka Prefecture, Gakuen-cho 1-1, Sakai, Osaka 593, Japan

Received March 1995; revised November 1995

Abstract

In this paper, we propose a simple but powerful heuristic method for automatically generating fuzzy if–then rules from numerical data. Fuzzy if–then rules with nonfuzzy singletons (i.e., real numbers) in the consequent parts are generated by the proposed heuristic method. The main advantage of the proposed heuristic method is its simplicity, i.e., it involves neither time-consuming iterative learning procedures nor complicated rule generation mechanisms. We also suggest a linguistic representation method for deriving linguistic rules from fuzzy if–then rules with consequent real numbers. The proposed linguistic approximation method consists of two linguistic rule tables, which can realize exactly the same nonlinear mapping as an original system based on fuzzy if–then rules with consequent real numbers. Using computer simulations on rice taste data, we demonstrate the high performance of the proposed heuristic method and illustrate the proposed linguistic representation method. © 1997 Elsevier Science B.V.

Keywords: Rule generation; Function approximation; Linguistic modeling; Rice taste analysis

1. Introduction

Fuzzy rule-based systems have been successfully applied to many control problems [6,10]. Basically, a fuzzy rule-based system provides an effective way to capture the approximate and inexact nature of the real world. In particular, fuzzy rule-based systems appear useful when the processes are too complex for analysis by conventional quantitative techniques or when the available information from the processes is qualitative, inexact or uncertain. From a theoretical point of view, it was proven that fuzzy systems are universal approximators of any real continuous function on a compact domain to arbitrary accuracy [5,16]. In many applications of fuzzy rule-based systems, fuzzy if–then rules have been obtained from human experts. Recently, various methods were proposed for **automatically generating fuzzy if–then rules from numerical data**. Most of these methods have involved iterative learning procedures or complicated rule generation mechanisms such as gradient descent learning methods [1–3,8], genetic-algorithm-based methods [4,9], least-squares methods [11,14], a fuzzy c-means method [12] and a fuzzy-neuro method [13]. In Wang and Mendel [15], an efficient rule generation method with no time-consuming iterative procedure was proposed and its high performance was demonstrated.

* Corresponding author. Fax: +81-722-59-3340; e-mail: hisaoi@ie.osakafu-u.ac.jp.

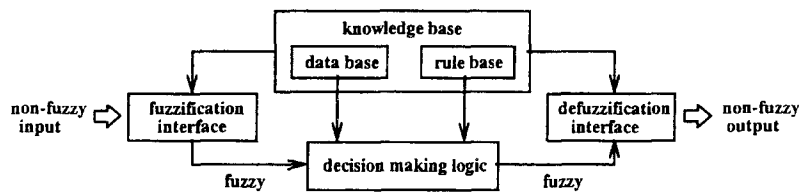


Fig. 1. Basic configuration of a fuzzy rule-based system.

In this paper, we propose a simple but powerful heuristic method for automatically generating fuzzy if–then rules from numerical data. Fuzzy if–then rules with nonfuzzy singletons (i.e., real numbers) in the consequent parts are generated by the proposed heuristic method. This type of fuzzy if–then rules has been also used by Ichihashi and Watanabe [1] and Nomura et al. [8,9]. The main advantage of these fuzzy if–then rules is the simplicity of a fuzzy reasoning procedure because no defuzzification step is required. In the proposed heuristic method, the consequent real number of each fuzzy if–then rule is determined as the weighted mean value of given numerical data. Thus, the proposed heuristic method does not require neither time-consuming iterative learning procedures nor complicated rule generation mechanisms.

In real world applications, it may be desired that linguistic rules are generated from numerical data. However, fuzzy if–then rules generated by the proposed heuristic method would have real numbers in the consequent parts. In general, it is not easy to linguistically represent such fuzzy rules. Sugeno and Yasukawa [12] proposed an approach for deriving linguistic rules from fuzzy if–then rules with fuzzy sets in the consequent parts. In this paper, we suggest a linguistic representation method for deriving linguistic rules from fuzzy if–then rules with consequent real numbers. The linguistic representation method generates two linguistic rule tables, i.e., a main rule table and a secondary rule table. Using these two linguistic rule tables, the linguistic representation method can realize exactly the same nonlinear mapping as an original system based on fuzzy if–then rules with real numbers in the consequent parts.

This paper is organized as follows. Section 2 gives a description of a fuzzy rule-based system used in this paper. In Section 3, we propose a heuristic method for determining the consequent real number of each fuzzy if–then rule and illustrate this method by computer simulations. Section 4 provides a linguistic representation method for deriving linguistic rules from fuzzy if–then rules with consequent real numbers. Section 5 gives computer simulations on rice taste data [7] and discusses the performance of the proposed heuristic method in comparison with that of a gradient descent learning method [2]. We also address knowledge acquisition about the rice taste analysis by a linguistic representation method. Section 6 finds conclusions.

2. Fuzzy rule-based systems

In this section, we present a fuzzy rule-based system used in this paper. Fuzzy rule-based systems are also known as fuzzy inference systems, fuzzy models, fuzzy associative memories (FAM) or fuzzy controllers. Basically, such fuzzy rule-based systems are composed of four principal components: a fuzzification interface, a knowledge base, a decision-making logic and a defuzzification interface. Fig. 1 shows the basic configuration of a fuzzy rule-based system.

In this paper, we consider a single-output fuzzy rule-based system in the n -dimensional input space $[0, 1]^n$. Let us assume that the following m input–output pairs are given as training data for constructing a fuzzy rule-based system:

$$\{(x_p; y_p) | p = 1, 2, \dots, m\}, \quad (1)$$

where $x_p = (x_{p1}, x_{p2}, \dots, x_{pn})$ is the input vector of the p th input–output pair and y_p is the corresponding output.

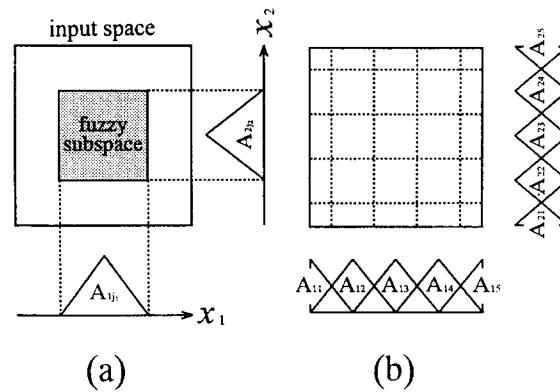


Fig. 2. (a) Fuzzy subspace and (b) fuzzy partition for $K_1 = 5$ and $K_2 = 5$.

2.1. Fuzzification interface

The fuzzification interface performs a mapping that converts crisp values of input variables into fuzzy singletons. Basically, a fuzzy singleton is a precise value and hence no fuzziness is introduced by fuzzification in this case. This strategy, however, has been widely used in fuzzy control applications because it is easily implemented. In this paper, we employ fuzzy singletons in the fuzzification interface.

2.2. Knowledge base

The knowledge base of a fuzzy rule-based system consists of two components, i.e., a data base and a rule base.

2.2.1. Data base

There are two factors that determine a data base, i.e., a fuzzy partition of an input space and membership functions of antecedent fuzzy sets. In this paper, we assume that the domain interval of the i th input variable x_i is evenly divided into K_i fuzzy sets labeled as $A_{i1}, A_{i2}, \dots, A_{iK_i}$ for $i = 1, 2, \dots, n$. Then the n -dimensional input space is divided into $K_1 K_2 \dots K_n$ fuzzy subspaces:

$$(A_{1j_1}, A_{2j_2}, \dots, A_{nj_n}), \quad j_1 = 1, 2, \dots, K_1; \dots; j_n = 1, 2, \dots, K_n. \quad (2)$$

For example, in the case of a two-dimensional input space, the fuzzy subspace (A_{1j_1}, A_{2j_2}) corresponds to the region shown in Fig. 2(a).

Though any type of membership functions (e.g., triangle-shaped, trapezoid-shaped and bell-shaped) can be used for antecedent fuzzy sets, we employ the symmetric triangle-shaped fuzzy sets A_{ij} with the following membership functions:

$$\mu_{ij_i}(x) = \max\{1 - |x - a_{j_i}^{K_i}|/b^{K_i}, 0\}, \quad j_i = 1, 2, \dots, K_i, \quad (3)$$

where

$$a_{j_i}^{K_i} = (j_i - 1)/(K_i - 1), \quad j_i = 1, 2, \dots, K_i, \quad (4)$$

$$b^{K_i} = 1/(K_i - 1). \quad (5)$$

Fig. 2(b) shows an example of the fuzzy partition for $K_1 = 5$ and $K_2 = 5$ in the case of a two-input-single-output fuzzy rule-based system.

2.2.2. Rule base

The rule base consists of a set of fuzzy if-then rules in the form of “IF a set of conditions are satisfied, THEN a set of consequences can be inferred”. In this paper, we assume that the rule base is composed of fuzzy if-then rules of the following form:

$$\text{Rule } R_{j_1 \dots j_n}: \text{ If } x_1 \text{ is } A_{1j_1} \text{ and } \dots \text{ and } x_n \text{ is } A_{nj_n} \text{ then } y \text{ is } b_{j_1 \dots j_n}, \\ j_1 = 1, 2, \dots, K_1; \dots; j_n = 1, 2, \dots, K_n, \quad (6)$$

where $R_{j_1 \dots j_n}$ is the label of each fuzzy if-then rule and $b_{j_1 \dots j_n}$ is the consequent real number. These fuzzy if-then rules are referred to as simplified fuzzy if-then rules and have been used in Ichihashi and Watanabe [1] and Nomura et al. [8,9]. In Jang [3], it was proven that fuzzy rule-based systems with simplified fuzzy if-then rules can approximate any nonlinear function on a compact set to arbitrary accuracy under certain conditions.

A single fuzzy if-then rule is generated in each fuzzy subspace in (2). Thus the number of fuzzy if-then rules in the fuzzy rule-based system is equal to that of the fuzzy subspaces in (2). In the case of the two-input-single-output fuzzy rule-based system with $K_1 = 5$ and $K_2 = 5$ in Fig. 2(b), 25 fuzzy if-then rules are generated.

We will propose a method for determining consequent real numbers of fuzzy if-then rules from numerical data in the next section.

2.3. Decision making logic

The decision making logic is the kernel of a fuzzy rule-based system, which employs fuzzy if-then rules from the rule base to infer the output by a fuzzy reasoning method. In this paper, we employ the following fuzzy reasoning method to calculate the inferred output of the fuzzy rule-based system. Given an input vector $\mathbf{x}_p = (x_{p1}, x_{p2}, \dots, x_{pn})$, the inferred output $y(\mathbf{x}_p)$ is defined by

$$y(\mathbf{x}_p) = \frac{\sum_{j_1=1}^{K_1} \dots \sum_{j_n=1}^{K_n} \mu_{j_1 \dots j_n}(\mathbf{x}_p) \cdot b_{j_1 \dots j_n}}{\sum_{j_1=1}^{K_1} \dots \sum_{j_n=1}^{K_n} \mu_{j_1 \dots j_n}(\mathbf{x}_p)}, \quad (7)$$

where $\mu_{j_1 \dots j_n}(\mathbf{x}_p)$ is the degree of compatibility of the input vector $\mathbf{x}_p = (x_{p1}, x_{p2}, \dots, x_{pn})$ to the fuzzy if-then rule $R_{j_1 \dots j_n}$ in (6), which is given by

$$\mu_{j_1 \dots j_n}(\mathbf{x}_p) = \mu_{1j_1}(x_{p1}) \times \dots \times \mu_{nj_n}(x_{pn}). \quad (8)$$

From (7), we can see that the inferred output $y(\mathbf{x}_p)$ is the weighted average of the consequent real numbers $b_{j_1 \dots j_n}$'s of the $K_1 K_2 \dots K_n$ fuzzy if-then rules. This method was also employed in [1,2,8].

2.4. Defuzzification interface

Basically, the defuzzification interface performs a mapping from the fuzzy output of a fuzzy rule-based system to a nonfuzzy output. The fuzzy rule-based system employed in this paper, however, does not require a defuzzification interface because it uses simplified fuzzy if-then rules in the rule base. This is one of the advantages of employing simplified fuzzy if-then rules.

3. Generating fuzzy rule from numerical data

We first propose a method to generate fuzzy if-then rules from numerical data and then illustrate the proposed method by computer simulations for single-input-single-output fuzzy rule-based systems. In this

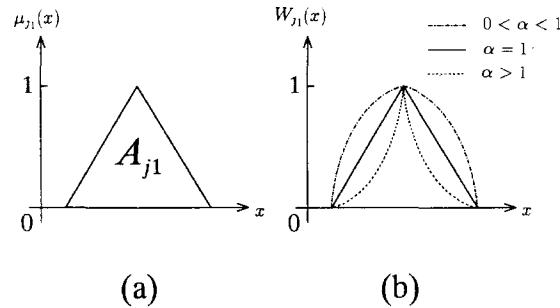


Fig. 3. (a) Membership function and (b) weights for input–output pairs.

section, we also address a least-squares method for determining consequent real numbers of fuzzy if–then rules, which is compared with the proposed heuristic method.

3.1. Rule generation method

Let us suppose that m input–output pairs $(x_p; y_p)$, $p = 1, 2, \dots, m$ are given as training data for an n -input–single-output fuzzy rule-based system described previously. Our problem here is to derive a consequent real number of each fuzzy if–then rule from the given training data.

For determining the consequent real number $b_{j_1 \dots j_n}$ of the fuzzy if–then rule $R_{j_1 \dots j_n}$ in (6), let us define the weight of the p th input–output pair $(x_p; y_p)$ as

$$W_{j_1 \dots j_n}(x_p) = \{\mu_{j_1 \dots j_n}(x_p)\}^\alpha, \quad (9)$$

where α is a positive constant. Fig. 3 illustrates the weights $W_{j_1}(x)$ in the case of one-dimensional input space with $0 < \alpha < 1$, $\alpha = 1$ and $\alpha > 1$, respectively. As seen from Fig. 3, we make much of only input–output pairs with high degrees of compatibility in the case of $\alpha > 1$, whereas we make much of even input–output pairs with low degrees of compatibility as well as those with high degrees of compatibility in the case of $0 < \alpha < 1$. The role of the positive constant α will be demonstrated by computer simulations later.

Using the weight $W_{j_1 \dots j_n}(x_p)$ of each input–output pair, we propose the following heuristic method for determining the consequent real number:

$$b_{j_1 \dots j_n} = \sum_{p=1}^m W_{j_1 \dots j_n}(x_p) \cdot y_p \Big/ \sum_{p=1}^m W_{j_1 \dots j_n}(x_p). \quad (10)$$

In (10), the consequent real number $b_{j_1 \dots j_n}$ is determined as the weighted mean value of y_p 's. We can generate a fuzzy if–then rule $R_{j_1 \dots j_n}$ by (10) if at least one of the weights $W_{j_1 \dots j_n}(x_p)$'s for $p = 1, 2, \dots, m$ is positive. If all the weights $W_{j_1 \dots j_n}(x_p)$'s in (10) are zero, we cannot generate the fuzzy if–then rule $R_{j_1 \dots j_n}$. This means that we cannot generate the fuzzy if–then rule $R_{j_1 \dots j_n}$ if there are no input–output pairs with positive degrees of compatibility to the corresponding fuzzy subspace $(A_{1j_1}, A_{2j_2}, \dots, A_{nj_n})$. In this case, the corresponding fuzzy subspace should be enlarged (i.e., we should use a coarser fuzzy partition).

Though we generate fuzzy if–then rules by the heuristic method above, it is possible to determine consequent real numbers of fuzzy if–then rules by a mathematical method such as a least-squares method. Provided that a sufficient number of noiseless input–output pairs are given, the least-squares method can find the consequent real numbers of fuzzy if–then rules that minimize a performance index of a fuzzy rule-based system. However, the fuzzy rule-based system based on the least-squares method is sensitive to noise and outliers. Therefore, it is not suitable for being applied to real-world problems, which usually include noisy data. This will be

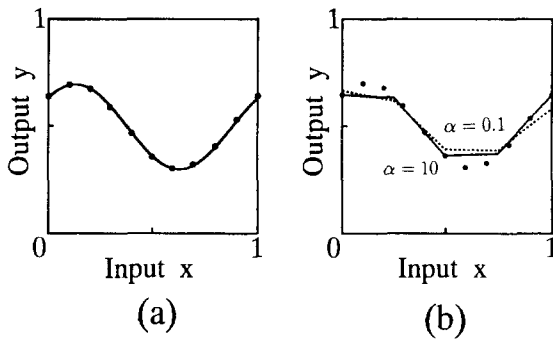


Fig. 4. (a) Input–output relation of the nonlinear system of (12) and (b) outputs of the proposed method for $K = 5$.

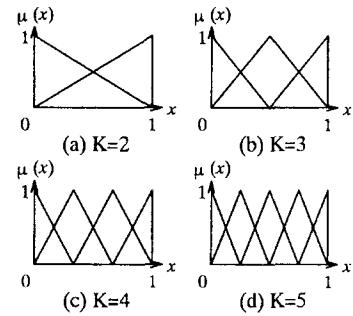


Fig. 5. Four fuzzy partitions.

illustrated by computer simulations below. The algorithm of the least-squares method for generating fuzzy rules will be shown in Appendix A.

3.2. Illustrative examples

We show four illustrative examples in this section. The first example illustrates the proposed heuristic method. The second example demonstrates how the proposed method can generate fuzzy if–then rules insensitive to noise. The third example discusses the fitting ability for training data and the generalization ability for test data of the proposed method and the least-squares method. The last example examines the performance of the proposed method with simple numerical data which were used in [13].

Example 1. To illustrate the proposed heuristic method, we apply it to a simple numerical example. As a performance index of a fuzzy rule-based system, we use the summation of square errors between the desired output y_p and the inferred output $y(x_p)$ for each input–output pair $(x_p; y_p)$:

$$PI = \sum_{p=1}^m \{y(x_p) - y_p\}^2 / 2. \quad (11)$$

Let us consider the following nonlinear static system with a single input x and a single output y :

$$y = 0.2 \sin(2\pi x + \pi/4) + 0.5, \quad 0 \leq x \leq 1. \quad (12)$$

Fig. 4(a) shows the input–output relation of this nonlinear static system. In Fig. 4(a), the closed circles denote the eleven input–output pairs shown in Table 1. We used those eleven input–output pairs $(x_p; y_p)$, $p = 1, 2, \dots, 11$ as training data for a single-input–single-output fuzzy rule-based system. Our problem here is to approximate the nonlinear static system of (12) by using the given eleven input–output pairs in Fig. 4(a).

For dividing the input space $[0, 1]$ into fuzzy subspaces, we employed the four fuzzy partitions shown in Fig. 5 where K means the number of fuzzy sets in each fuzzy partition. The fuzzy if–then rules corresponding to each fuzzy partition in Fig. 5 can be written as

$$\text{If } x \text{ is } A_j \text{ then } y = b_j, \quad j = 1, 2, \dots, K \quad (13)$$

where A_j is the antecedent fuzzy set, b_j is the consequent real number and K is the number of A_j 's; in this case, K is also the number of fuzzy if–then rules.

Table 1
Input–output pairs of the nonlinear system of (12)

p	x	y
1	0.000	0.641
2	0.100	0.698
3	0.200	0.678
4	0.300	0.591
5	0.400	0.469
6	0.500	0.359
7	0.600	0.302
8	0.700	0.322
9	0.800	0.409
10	0.900	0.531
11	1.000	0.641

Table 2
Performance index of fuzzy rule-based systems

α	Performance index			
	$K = 2$	$K = 3$	$K = 4$	$K = 5$
0.1	0.108	0.070	0.041	0.020
0.5	0.101	0.060	0.032	0.015
1	0.095	0.051	0.025	0.011
5	0.126	0.044	0.015	0.006
10	0.165	0.049	0.014	0.005
50	0.200	0.052	0.016	0.005
100	0.200	0.052	0.016	0.005

To investigate the relation between the performance of fuzzy rule-based systems and the values of α , we employed the proposed heuristic method with the following parameter specifications:

Fuzzy partitions. $K = 2, 3, 4$ and 5 ,

Values of α . $\alpha = 0.1, 0.5, 1, 5, 10, 50$ and 100

Table 2 shows the values of the performance index of the resulting fuzzy rule-based systems with various parameter specifications. From Table 2, we can see the following:

(i) Large values of K lead to good fitting to the given input–output pairs.

(ii) The performance of fuzzy rule-based systems can be improved by choosing an appropriate value of α .

Fig. 4(b) shows two examples of the inferred outputs by the fuzzy rule-based systems with the generated fuzzy if–then rules for $K = 5$. In Fig. 4(b), the dotted and the solid lines denote the inferred outputs by the fuzzy rule-based systems generated with $\alpha = 0.1$ and 10 , respectively. From this figure, we can see that better fitting to the given input–output pairs is obtained by the fuzzy rule-based system generated with $\alpha = 10$. Fig. 6 shows the generated fuzzy if–then rules in the fuzzy rule-based system for $K = 5$ and $\alpha = 10$.

Example 2. In real-world applications, available input–output data may be noisy or include some outliers. This example illustrates how the proposed heuristic method can generate fuzzy if–then rules insensitive to noise and outliers.






If x_1 is  then y is 0.642
 If x_1 is  then y is 0.634
 If x_1 is  then y is 0.359
 If x_1 is  then y is 0.366
 If x_1 is  then y is 0.641

Fig. 6. Fuzzy if–then rules generated by the proposed method for $K = 5$ and $\alpha = 10$.

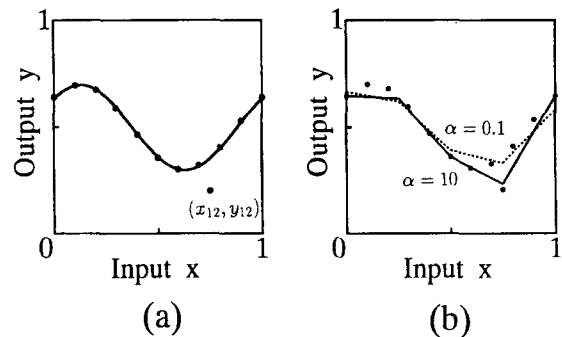


Fig. 7. (a) Input–output relation of the nonlinear system of (12) with one outlier (x_{12}, y_{12}) and (b) outputs by the proposed method for $K = 5$.

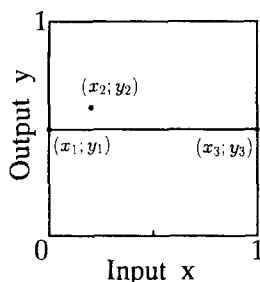


Fig. 8. Input-output relation of the system of (14).

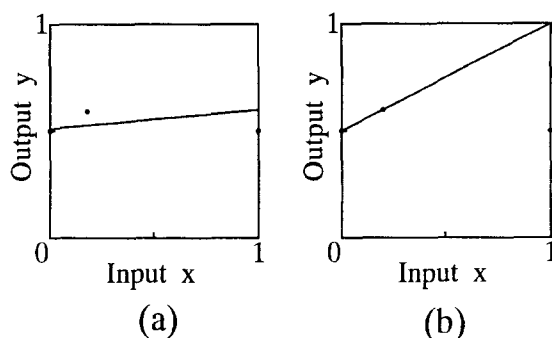


Fig. 9. Outputs by (a) the proposed method for $K = 2$ and (b) the least squares method for $K = 2$.

Let us assume that one outlier $(x_{12}; y_{12}) = (0.75; 0.2)$, which does not satisfy the nonlinear function in (12), is added to the previous input–output pairs as shown in Fig. 7(a). Fig. 7(b) shows the inferred outputs by the proposed heuristic method with the same parameter specifications in Fig. 4(b). From Fig. 7(b), we can see the following:

(i) The inferred output is over-fitting to the outlier $(x_{12}; y_{12})$ in the case of the fuzzy rule-based system generated with a large value of α (in this case, $\alpha = 10$).

(ii) On the contrary, the inferred output by the fuzzy rule-based system generated with a small value of α (in this case, $\alpha = 0.1$) is not sensitive to the outlier, i.e., the two inferred outputs in Figs. 4(b) and 7(b) are similar to each other in the case of $\alpha = 0.1$.

Example 3. In this example, we address the fitting ability for training data and the generalization ability for test data by the proposed heuristic method and the least-squares method.

Let us consider the following static system with a single input x and a single output y :

$$y = 0.5, \quad 0 \leq x \leq 1. \quad (14)$$

Fig. 8 shows the input–output relation of this system. We obtain the three input–output pairs shown in Fig. 8, i.e., $(x_1; y_1) = (0; 0.5)$, $(x_2; y_2) = (0.2; 0.6)$ and $(x_3; y_3) = (1; 0.5)$. Suppose that $(x_1; y_1)$ and $(x_2; y_2)$ are given as training data and $(x_3; y_3)$ as test data. In this case, $(x_2; y_2)$ can be viewed as noise.

Fig. 9(a) and (b) shows the inferred outputs by the proposed heuristic method for $K = 2$ and $\alpha = 1$ and the least-squares method for $K = 2$, respectively. The fitting ability of the least-squares method is better than that of the proposed heuristic method, i.e., we have $PI = 0.0$ for the least-squares method and $PI = 0.003$ for the proposed method. As stated previously, we can see that the least-squares method enables us to find the consequent real numbers that minimize the performance index of the fuzzy rule-based system. For the generalization ability for test data, however, we have $PI = 0.125$ for the least-squares method and $PI = 0.005$ for the proposed method. As seen from Fig. 9(b), the inferred output by the least-squares method overfitted to the training data $(x_1; y_1)$ and $(x_2; y_2)$. On the contrary, we can see from Fig. 9(a) the inferred output by the proposed heuristic method did not overfit to the training data. It is because the proposed heuristic method determines the consequent real number of each fuzzy rule as the weighted mean value of the given data. Thus, we can conclude that the proposed heuristic method is more suitable for determining consequent real numbers of fuzzy if–then rules than the least-squares method in the sense that the proposed heuristic method is less sensitive to noisy data than the least-squares method.

Example 4. In order to examine the performance of the proposed heuristic method, we applied our method to simple numerical data which were used in Takagi and Hayashi [13].

The numerical data consisted of 40 input–output pairs that were divided into training data (20 pairs) and checking data (20 pairs). In our computer simulation, eight fuzzy if–then rules were generated from the training data by using two fuzzy sets (small and large) for each of three input variables x_1 , x_2 and x_3 . As in Takagi and Hayashi, we excluded x_4 from the antecedent part of fuzzy if–then rules. The sum of squared errors for the checking data was 37.4, which is larger than the result of Takagi and Hayashi (14.0). While our method seems worse than the NN-driving fuzzy reasoning method by Takagi and Hayashi, we should note the following:

(i) In our method, the checking data were not used for generating fuzzy if–then rules at all. On the contrary, the checking data were used for determining the number of learning iterations of each neural network and the structure of fuzzy if–then rules in Takagi and Hayashi.

(ii) Our method involves no time-consuming procedure for generating fuzzy if–then rules. The NN-driven fuzzy reasoning method involves clustering of training data, iterative learning of multiple neural networks and structure determination of each fuzzy if–then rule.

4. Deriving linguistic rules

Fuzzy if–then rules in the previous section have real numbers in the consequent parts. Therefore, the linguistic interpretation of these fuzzy if–then rules is not easy. In real-world applications, however, it may be desired that linguistic rules are generated from numerical data. In this section, we propose an approach for **translating fuzzy if–then rules with consequent real numbers into linguistic rules**. In this connection, this approach can derive linguistic rules from fuzzy if–then rules with consequent real numbers, which may be generated by other rule generation methods (for example, [1,8,9]) as well as the proposed heuristic method in the previous section.

4.1. Linguistic representation

Let us assume that fuzzy if–then rules in (6) are given. Our problem here is to generate fuzzy if–then rules with linguistic labels in the consequent parts from the given fuzzy if–then rules with consequent real numbers in (6). To translate consequent real numbers into linguistic labels, suppose that the **domain interval of an output y is divided into N fuzzy sets (i.e., linguistic labels) B_1, B_2, \dots, B_N , which are associated with the membership functions $\mu_{B_1}, \mu_{B_2}, \dots, \mu_{B_N}$, respectively**. For example, these fuzzy sets may have linguistic labels such as S: small; MS: medium small; M: medium; ML: medium large and L: large. Fig. 10 shows an example of linguistic labels associated with fuzzy sets.

We propose the following method to linguistically represent fuzzy if–then rules with consequent real numbers. In this method, the given fuzzy if–then rules in (6) are represented by two linguistic rule tables, i.e., a main rule table and a secondary rule table.

Main rule table. The main rule table consists of the following **fuzzy if–then rules, which are mainly used to infer the output by the fuzzy rule-based system**.

$$\text{Rule } R_{j_1 \dots j_n}^* : \text{ If } x_1 \text{ is } A_{1j_1} \text{ and } \dots \text{ and } x_n \text{ is } A_{nj_n} \text{ then } y \text{ is } B_{j_1 \dots j_n}^* \text{ with } CF_{j_1 \dots j_n}^*, \quad (15)$$

$$j_1 = 1, 2, \dots, K_1; \dots; j_n = 1, 2, \dots, K_n,$$

where $B_{j_1 \dots j_n}^*$ is the consequent fuzzy set characterized by the following membership function:

$$\mu_{B_{j_1 \dots j_n}^*}(b_{j_1 \dots j_n}) = \max\{\mu_{B_i}(b_{j_1 \dots j_n}) \mid i = 1, 2, \dots, N\}, \quad (16)$$

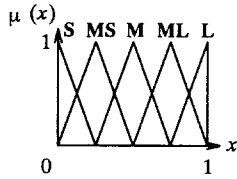
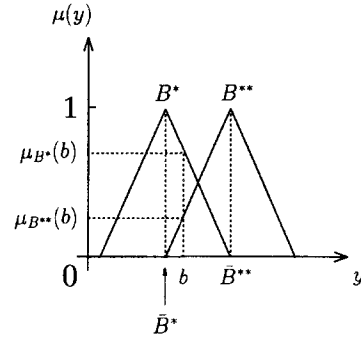


Fig. 10. Linguistic labels associated with fuzzy sets.

Fig. 11. Consequent fuzzy sets B^* and B^{**} .

and $CF_{j_1 \dots j_n}^*$ is the degree of certainty defined as

$$CF_{j_1 \dots j_n}^* = \mu_{B^*}(b_{j_1 \dots j_n}). \quad (17)$$

Secondary rule table. The secondary rule table is composed of the following fuzzy if–then rules, which are subordinately used to infer the output by the fuzzy rule-based system.

$$\text{Rule } R_{j_1 \dots j_n}^{**}: \text{ If } x_1 \text{ is } A_{1j_1} \text{ and } \dots \text{ and } x_n \text{ is } A_{nj_n} \text{ then } y \text{ is } B_{j_1 \dots j_n}^{**} \text{ with } CF_{j_1 \dots j_n}^{**}, \\ j_1 = 1, 2, \dots, K_1; \dots; j_n = 1, 2, \dots, K_n, \quad (18)$$

where $B_{j_1 \dots j_n}^{**}$ is the consequent fuzzy set with the second best fitting to the consequent real number $b_{j_1 \dots j_n}$ and $CF_{j_1 \dots j_n}^{**}$ is defined in the same manner as in (17).

Let us consider the case where the following fuzzy if–then rule is to be linguistically represented by linguistic fuzzy if–then rules.

$$\text{If } x_1 \text{ is } A_{1j_1} \text{ and } \dots \text{ and } x_n \text{ is } A_{nj_n} \text{ then } y \text{ is } b, \quad (19)$$

where b is a consequent real number. In this case, the fuzzy if–then rule in (19) is linguistically represented by the following two fuzzy if–then rules:

$$\text{If } x_1 \text{ is } A_{1j_1} \text{ and } \dots \text{ and } x_n \text{ is } A_{nj_n} \text{ then } y \text{ is } B^* \text{ with } CF = \mu_{B^*}(b), \quad (20)$$

and

$$\text{If } x_1 \text{ is } A_{1j_1} \text{ and } \dots \text{ and } x_n \text{ is } A_{nj_n} \text{ then } y \text{ is } B^{**} \text{ with } CF = \mu_{B^{**}}(b), \quad (21)$$

where B^* and B^{**} are consequent fuzzy sets with linguistic labels. Fig. 11 shows the two consequent fuzzy sets B^* and B^{**} to linguistically represent the consequent real number b in (19). In Fig. 11, $\mu_{B^*}(b)$ and $\mu_{B^{**}}(b)$ ($\mu_{B^*}(b) > \mu_{B^{**}}(b)$) denote the membership values of the consequent fuzzy sets B^* and B^{**} at the consequent real number b , respectively; \bar{B}^* and \bar{B}^{**} are the center values of B^* and B^{**} , respectively. Under the assumption that we use the triangle-shaped fuzzy sets whose membership functions are defined by (3)–(5) in the consequent parts of fuzzy if–then rules, the following equations always hold for all y 's satisfying $\bar{B}^* \leq y \leq \bar{B}^{**}$:

$$y = \mu_{B^*}(y) \cdot \bar{B}^* + \mu_{B^{**}}(y) \cdot \bar{B}^{**}, \quad (22)$$

$$\mu_{B^*}(y) + \mu_{B^{**}}(y) = 1. \quad (23)$$

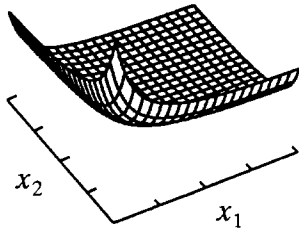


Fig. 12. Input-output relation of the nonlinear system of (25).

x_2	L	0.465	0.182	0.040	0.005	0.001
	ML	0.540	0.160	0.067	0.038	0.017
	M	0.825	0.254	0.063	0.099	0.088
	MS	0.757	0.346	0.259	0.204	0.250
	S	0.986	0.800	0.628	0.363	0.383
		S	MS	M	ML	L
		x_1				

Fig. 13. Generated fuzzy if-then rules for $K = 5$ and $\alpha = 5$.

From (22) and (23), we can see that any real number y in the interval $[\bar{B}^*, \bar{B}^{**}]$ can be represented by the two fuzzy sets B^* and B^{**} . This means that the consequent real number b can also be linguistically represented by two fuzzy sets B^* and B^{**} . Thus we can translate the fuzzy if-then rule with the consequent real number in (19) into the two fuzzy if-then rules with the consequent fuzzy sets in (20) and (21). The **fuzzy if-then rule with a larger degree of certainty** (in this case, the fuzzy if-then rule in (20)) is **added to the main rule table** and the **other fuzzy if-then rule is added to the secondary rule table**. This procedure is applied to all fuzzy if-then rules with consequent real numbers in (6) and then fuzzy if-then rules with consequent fuzzy sets in (15) and (18) are derived.

Given an input vector x_p , the inferred output by the fuzzy rule-based system based on these two linguistic rule tables is calculated as

$$y(x_p) = \frac{\sum_{j_1=1}^{K_1} \cdots \sum_{j_n=1}^{K_n} \{\mu_{j_1 \dots j_n}(x_p) \cdot \bar{B}_{j_1 \dots j_n}^* \cdot CF_{j_1 \dots j_n}^* + \mu_{j_1 \dots j_n}(x_p) \cdot \bar{B}_{j_1 \dots j_n}^{**} \cdot CF_{j_1 \dots j_n}^{**}\}}{\sum_{j_1=1}^{K_1} \cdots \sum_{j_n=1}^{K_n} \{\mu_{j_1 \dots j_n}(x_p) \cdot CF_{j_1 \dots j_n}^* + \mu_{j_1 \dots j_n}(x_p) \cdot CF_{j_1 \dots j_n}^{**}\}}, \quad (24)$$

where $\bar{B}_{j_1 \dots j_n}^*$ and $\bar{B}_{j_1 \dots j_n}^{**}$ are the center values of the consequent fuzzy sets $B_{j_1 \dots j_n}^*$ and $B_{j_1 \dots j_n}^{**}$, respectively. From (24), we can see that the inferred output by the linguistic representation method in (24) is exactly the same as that in (7).

4.2. Illustrative example

To illustrate the linguistic representation method, let us consider the following nonlinear two-input-single-output system:

$$y = (1 + x_1^{-2} + x_2^{-1.5})^2, \quad 1 \leq x_1, x_2 \leq 5, \quad (25)$$

which was also used in [12]. Fig. 12 shows the input-output relation of this nonlinear system. We obtain 50 input-output pairs $(x_{p1}, x_{p2}; y_p)$, $p = 1, 2, \dots, 50$ from (25), which are the same as those used in [12].

We first applied the proposed rule generation method described previously to this example. In computer simulations, we normalized the input-output pairs in the interval $[0, 1]$ and used the same fuzzy partition for each of the two inputs, i.e., we specified the values of K_i 's as $K_1 = K_2 = K$ where $K = 2, 3, 4$ and 5 . Table 3 summarizes the values of the performance index of the resulting fuzzy rule-based systems. From Table 3, we can get the best performance index of the fuzzy rule-based system for $K = 5$ and $\alpha = 5$. Fig. 13 shows the generated fuzzy if-then rules for $K = 5$ and $\alpha = 5$. In Fig. 13, S, MS, M, ML and L denote the linguistic

Table 3
Performance index of fuzzy rule-based systems

α	Performance index			
	$K = 2$	$K = 3$	$K = 4$	$K = 5$
0.1	1.598	0.757	0.566	0.417
0.5	1.180	0.632	0.462	0.324
1	0.925	0.525	0.375	0.252
5	0.730	0.408	0.236	0.175
10	0.988	0.576	0.240	0.204
50	1.636	0.886	0.268	0.230
100	1.716	0.893	0.276	0.231

x_2	L	M	MS	S	S	S
	ML	M	MS	S	S	S
	M	ML	MS	S	S	S
	MS	ML	MS	MS	MS	MS
	S	L	ML	ML	MS	M
		S	MS	M	ML	L
		x_1				

Fig. 14. Main rule table.

x_2	L	MS	S	MS	MS	MS
	ML	ML	S	MS	MS	MS
	M	L	M	MS	MS	MS
	MS	L	M	M	S	M
	S	ML	L	M	M	MS
		S	MS	M	ML	L
		x_1				

Fig. 15. Secondary rule table.

labels associated with the antecedent fuzzy sets in Fig. 10. For example, we have the fuzzy if–then rule such that “If x_1 is S and x_2 is S then y is 0.986” in the fuzzy subspace where ‘0.986’ is attached in Fig. 13.

Then we applied the linguistic representation method to the generated fuzzy if–then rules in Fig. 13. We here employed the linguistic labels in Fig. 10 as the consequent fuzzy sets to linguistically represent the fuzzy if–then rules in Fig. 13. Figs. 14 and 15 show the main rule table and the secondary rule table derived by the linguistic representation method, respectively. From the comparison of Fig. 12 with Fig. 14, we can say that the main rule table linguistically approximates the nonlinear system of (25). The derived linguistic rules in the main rule table are almost the same as those generated by Sugeno and Yasukawa’s method [12].

5. Simulation results on rice taste data

This section presents simulation results by the proposed methods on a real-world problem. To this aim, we deal with the rice taste data, which was also used by Ishibuchi et al. [2] and Matsuda and Kameoka [7]. First, we discuss the performance of the proposed heuristic method in comparison with a gradient descent learning method [2] and a least-squares method. Then we perform the linguistic representation method to acquire linguistic knowledge about rice taste analysis.

The rice taste data consists of five inputs and a single output whose values are associated with subjective evaluations as follows:

x_1 : flavor, x_2 : appearance, x_3 : taste, x_4 : stickiness, x_5 : toughness, y : overall evaluation.

Sensory test data shown in Appendix B have been obtained by such subjective evaluations for 105 kinds of rice (e.g., Sasanishiki, Akita-Komachi, etc.). The input–output pairs can be written as

$$(x_{p1}, x_{p2}, x_{p3}, x_{p4}, x_{p5}; y_p), \quad p = 1, 2, \dots, 105. \quad (26)$$

Our problem here is to approximate the following unknown nonlinear mapping f by a fuzzy rule-based system.

$$y_p = f(x_{p1}, x_{p2}, x_{p3}, x_{p4}, x_{p5}), \quad p = 1, 2, \dots, 105. \quad (27)$$

Since the unknown mapping in (27) to be realized by a fuzzy rule-based system is a five-input–single-output system, fuzzy if–then rules for the rice taste data can be written as

$$\text{If } x_1 \text{ is } A_{1j_1} \text{ and } \dots \text{ and } x_5 \text{ is } A_{5j_5} \text{ then } y \text{ is } b_{j_1 \dots j_5}, \\ j_1 = 1, 2, \dots, K_1; \dots; j_5 = 1, 2, \dots, K_5. \quad (28)$$

In computer simulations, the input–output pairs of the rice taste data were normalized in the interval $[0, 1]$. We used the four fuzzy partitions shown in Fig. 5 for dividing the domain interval of each input. The same fuzzy partition was used for each of the five inputs, i.e., we specified the values of K_i 's as $K_1 = K_2 = K_3 = K_4 = K_5 = K$ where $K = 2, 3, 4$ and 5 .

To investigate the fitting ability for training data and the generalization ability for test data, we randomly divided the 105 input–output pairs into 75 pairs (training data) and 30 pairs (test data). The average values of the performance index over ten trials with different sets of training and test data were calculated. We also show the simulation results by other methods such as the gradient descent learning method [2] and the least squares method.

5.1. Fitting ability for training data

Tables 4 and 5 summarize the average values of the performance index for training data by the proposed heuristic method and the gradient descent learning method, respectively. From Tables 4 and 5, we can see the following:

- (i) Large values of K (i.e., fine fuzzy partitions) lead to good fitting to the training data in both methods.
- (ii) The fitting ability of the gradient descent learning method is superior to that of the proposed heuristic method.

We also employed the least-squares method on the same data sets. For high-dimensional problems, it is extremely memory-consuming to calculate the matrix equations in the least-squares method. In this case, the matrix equations involve matrices of dimension $K^5 \times K^5$. Therefore, we only used small values of K (i.e., $K = 2$ and 3). The average values of the performance index of the least-squares method were $PI = 0.041$ for $K = 2$ and $PI = 0.004$ for $K = 3$. From the values of the performance index of each method, we can see that the fitting ability of the least-squares method is better than the other methods. It is theoretically proven that the least-squares method finds an optimal solution and the performance index of the gradient descent method converges to the optimal solution.

5.2. Generalization ability for test data

Tables 6 and 7 summarize the average values of the performance index for test data by the proposed heuristic method and the gradient descent learning method, respectively. From these tables, we can see the following:

- (i) The performance of the proposed heuristic method for test data is less sensitive to the value of K than that of the gradient descent learning method.
- (ii) Large values of K lead to bad fitting to the test data (i.e., overfitting to the training data) in the gradient descent learning method.
- (iii) The performance of the proposed heuristic method for test data is comparable to that of the gradient descent learning method.

Table 4
Fitting ability of the proposed method

α	Performance index			
	$K = 2$	$K = 3$	$K = 4$	$K = 5$
0.1	1.348	0.385	0.158	0.110
0.5	0.930	0.245	0.117	0.080
1	0.634	0.178	0.096	0.064
5	0.216	0.110	0.069	0.052
10	0.213	0.110	0.067	0.054
50	0.212	0.119	0.069	0.055
100	0.215	0.119	0.069	0.055

Table 5
Fitting ability of the gradient descent learning method

No. epochs	Performance index			
	$K = 2$	$K = 3$	$K = 4$	$K = 5$
100	0.070	0.043	0.037	0.016
200	0.066	0.036	0.023	0.010
300	0.063	0.033	0.017	0.008
400	0.061	0.031	0.014	0.006
500	0.060	0.030	0.012	0.006

Table 6
Generalization ability of the proposed method

α	Performance index			
	$K = 2$	$K = 3$	$K = 4$	$K = 5$
0.1	0.610	0.193	0.077	0.099
0.5	0.429	0.134	0.064	0.091
1	0.302	0.104	0.058	0.088
5	0.120	0.074	0.054	0.087
10	0.118	0.073	0.055	0.088
50	0.114	0.074	0.058	0.089
100	0.115	0.073	0.059	0.142

Table 7
Generalization ability of the gradient descent learning method

No. epochs	Performance index			
	$K = 2$	$K = 3$	$K = 4$	$K = 5$
100	0.047	0.040	0.211	0.534
200	0.047	0.046	0.210	0.532
300	0.048	0.052	0.211	0.533
400	0.051	0.059	0.214	0.534
500	0.056	0.065	0.216	0.535

Table 8
CPU times (s)

	$K = 2$	$K = 3$	$K = 4$	$K = 5$
Heuristic	1.2	6.8	23.9	66.8
Learning	260.5	1995.9	8448.1	25846.8
LSM	6.6	3216.6	N/A	N/A

The average values of the performance index of the least-squares method were $PI = 5.976$ for $K = 2$ and $PI = 39.338$ for $K = 3$. Whereas the least-squares method can yield the best fitting to the training data, the generalization ability of the least-squares method is not so good because of overfitting to the training data.

5.3. Comparison of computation times

To compare the required computation times of each method, we generated fuzzy if-then rules from all the given 105 input-output pairs by each method. Table 8 shows the required CPU times on a 17 MIPS Sony workstation for each method. The simulation results by the gradient descent learning method in Table 8 are the CPU times for 500 epochs. From Table 8, we can see that the proposed heuristic method can generate fuzzy if-then rules in much less CPU time than the gradient descent learning method and the least-squares method (LSM). This is one advantage of the proposed heuristic method.

Table 9
Main rule table for rice taste analysis

Flavor (x_1)	Appearance (x_2)	Taste (x_3)	Stickiness (x_4)	Toughness (x_5)	Overall evaluation (y)
Good	Good	Good	Sticky	Tough	High
Good	Good	Good	Sticky	Tender	High
Good	Bad	Good	Sticky	Tough	Medium high
Bad	Good	Good	Sticky	Tender	Medium high
Bad	Bad	Good	Sticky	Tender	Medium high
Good	Bad	Good	Sticky	Tender	Medium high
Good	Good	Good	Not sticky	Tender	Medium high
Good	Bad	Bad	Sticky	Tender	Medium high
Good	Good	Bad	Sticky	Tough	Medium
Good	Good	Bad	Not sticky	Tender	Medium
Good	Good	Bad	Sticky	Tender	Medium
Bad	Bad	Bad	Sticky	Tender	Medium
Good	Bad	Good	Not sticky	Tender	Medium
Good	Bad	Bad	Sticky	Tough	Medium
Bad	Bad	Good	Sticky	Tough	Medium
Good	Good	Good	Not sticky	Tough	Medium
Bad	Good	Good	Not sticky	Tender	Medium
Bad	Good	Bad	Sticky	Tender	Medium
Bad	Good	Good	Not sticky	Tough	Medium
Bad	Good	Bad	Sticky	Tough	Medium
Bad	Good	Good	Sticky	Tough	Medium
Good	Good	Bad	Not sticky	Tough	Medium
Bad	Bad	Good	Not sticky	Tough	Medium low
Good	Bad	Good	Not sticky	Tough	Medium low
Good	Bad	Bad	Not sticky	Tender	Medium low
Bad	Good	Bad	Not sticky	Tough	Medium low
Bad	Bad	Bad	Sticky	Tough	Medium low
Bad	Good	Bad	Not sticky	Tender	Medium low
Bad	Bad	Bad	Not sticky	Tender	Medium low
Bad	Bad	Good	Not sticky	Tender	Medium low
Good	Bad	Bad	Not sticky	Tough	Medium low
Bad	Bad	Bad	Not sticky	Tough	Low

5.4. Knowledge acquisition

We performed the linguistic representation method to derive linguistic rules from fuzzy if–then rules generated by the heuristic method. In this computer simulation, we first generated fuzzy if–then rules from the 105 input–output pairs by the heuristic method for $K = 2$ and $\alpha = 50$. Then we obtained two linguistic rule tables from the generated fuzzy if–then rules by the linguistic representation method. Table 9 shows the main rule table consisting of 32 fuzzy if–then rules whose antecedent and consequent fuzzy sets are characterized by the linguistic labels in Fig. 16.

From Table 9, we can acquire linguistic knowledge about rice taste analysis. For example, the following linguistic knowledge is obtained:

- (i) The factor x_3 (taste) with *good* leads to *high* overall evaluation.
- (ii) The factor x_4 (stickiness) with *not sticky* leads to *low* overall evaluation.
- (iii) Rice with “ x_1 (flavor), x_2 (appearance) and x_3 (taste) are *good* and x_4 (stickiness) is *sticky*” obtains *high* overall evaluation.

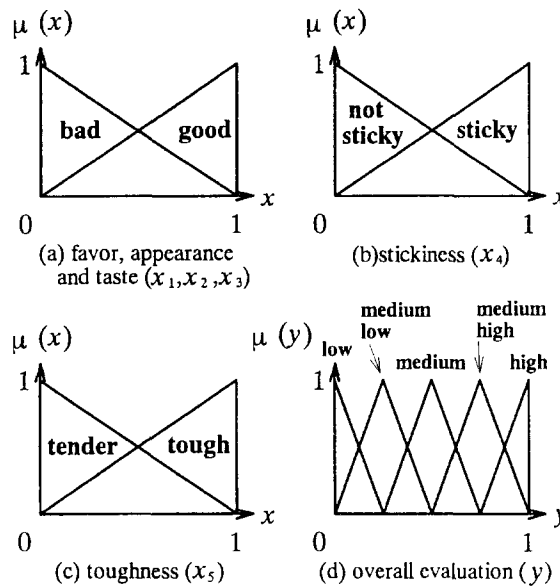


Fig. 16. Linguistic labels for rice taste analysis.

(iv) Rice with “ x_1 (flavor), x_2 (appearance) and x_3 (taste) are *bad* and x_4 (stickiness) is *not sticky* and x_5 (toughness) is *tough*” obtains *low* overall evaluation.

As is shown above, we can acquire linguistic knowledge about the rice taste analysis from the main rule table derived by the linguistic representation method. This is one advantage of the proposed linguistic representation method.

6. Conclusions

We have proposed a simple but powerful heuristic method for generating fuzzy if–then rules from numerical data. The main advantage of the proposed heuristic method is its simplicity. It can easily be implemented as a computer program and requires much less computation time than the gradient descent learning method as demonstrated by computer simulations. We have also suggested a linguistic representation method for deriving linguistic rules from fuzzy if–then rules with real numbers in the consequent parts. This linguistic representation method consists of two linguistic rule tables, i.e., a main rule table and a secondary rule table. Using these two linguistic rule tables, we can realize exactly the same nonlinear mapping as an original system based on fuzzy if–then rules with consequent real numbers. In computer simulations on rice taste data, we can acquire linguistic knowledge about the rice taste analysis from the main rule table derived by the linguistic representation method. From this point of view, the linguistic representation method can be viewed as a knowledge acquisition tool by which we can obtain linguistic knowledge.

Acknowledgements

This work was partially supported by Grant-in-Aid for Scientific Research (No. 06680404) from the Ministry of Education, Science and Culture in Japan.

Appendix A. Generating fuzzy rules by least-squares method

Our problem here is to find consequent real numbers $b_{j_1 \dots j_n}$'s that minimize the performance index PI defined by (11).

Introducing matrix notation, our problem can be written as

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} c_{1\dots 1}^1 & \dots & c_{K_1 \dots K_n}^1 \\ \vdots & \vdots & \vdots \\ c_{1\dots 1}^m & \dots & c_{K_1 \dots K_n}^m \end{bmatrix} \begin{bmatrix} b_{1\dots 1} \\ \vdots \\ b_{K_1 \dots K_n} \end{bmatrix}, \quad (\text{A.1})$$

where

$$c_{j_1 \dots j_n}^p = \mu_{j_1 \dots j_n}(\mathbf{x}_p) / \sum_{j_1=1}^{K_1} \dots \sum_{j_n=1}^{K_n} \mu_{j_1 \dots j_n}(\mathbf{x}_p), \quad p = 1, 2, \dots, m. \quad (\text{A.2})$$

For simplicity, we rewrite (A.1) as

$$Y = CB, \quad (\text{A.3})$$

where

$$Y = [y_1 \ y_2 \ \dots \ y_m]^T, \quad (\text{A.4})$$

$$B = [b_{1\dots 1} \ \dots \ b_{K_1 \dots K_n}]^T, \quad (\text{A.5})$$

and

$$C = \begin{bmatrix} c_{1\dots 1}^1 & \dots & c_{K_1 \dots K_n}^1 \\ \vdots & \vdots & \vdots \\ c_{1\dots 1}^m & \dots & c_{K_1 \dots K_n}^m \end{bmatrix}. \quad (\text{A.6})$$

Using the pseudo inverse of C , we can obtain the consequent parameter vector B as

$$B = (C^T C)^{-1} C^T Y. \quad (\text{A.7})$$

Since it is computationally expensive to directly calculate the pseudo inverse $(C^T C)^{-1} C^T$, we apply the following sequential least-squares method to calculate the consequent parameter vector B . Let c_i be the i th row vector of the matrix C defined in (A.6), $i = 0, 1, \dots, m$. Then B is recursively calculated as follows:

$$B_{i+1} = B_i + S_{i+1} \cdot c_{i+1}^T \cdot (y_{i+1} - c_{i+1} \cdot B_i), \quad i = 0, 1, \dots, m-1, \quad (\text{A.8})$$

$$S_{i+1} = S_i - \frac{S_i \cdot c_{i+1}^T \cdot c_{i+1} \cdot S_i}{1 + c_{i+1} \cdot S_i \cdot c_{i+1}^T}, \quad i = 0, 1, \dots, m-1. \quad (\text{A.9})$$

The initial values of B_0 and S_0 are defined as $B_0 = 0$ and $S_0 = \gamma I$, respectively, where γ is a positive large number; I is the identity matrix of dimension $K_1 \dots K_n \times K_1 \dots K_n$. Thus, we can obtain the least-squares estimate $B = B_m$.

Appendix B. Rice taste data

Favor (x_1)	Appearance (x_2)	Taste (x_3)	Stickiness (x_4)	Toughness (x_5)	Overall evaluation (y)
0.523	0.913	1.571	1.6	0	1.784
0.699	1.543	1.76	1.944	-0.875	1.706
-0.72	-2.022	-1.733	-1.864	1.2	-2.217
-0.309	-0.377	-0.766	-0.96	0.326	-0.875
0.192	0.885	-0.166	-0.106	0.234	0.218
-0.163	-0.094	-0.315	0.083	0.034	-0.655
-0.274	0.044	-0.194	-0.401	1.57	-0.545
0.291	0.984	0.935	0.68	-0.652	1.033
-2.687	-2.853	-3.067	-1.75	2.392	-3.101
-0.704	-0.424	-0.518	-0.779	0.052	-0.652
1.094	1.987	1.613	1.514	-1.156	1.552
-0.247	0.334	0.122	-0.705	0.694	-0.416
-0.673	0.032	-0.838	-0.837	0.354	-0.784
-1.103	-1.046	-1.678	-1.335	-0.46	-1.71
-0.033	0.162	-0.541	-0.159	0.616	-0.529
-0.351	0.119	-0.786	-0.435	0.129	-0.635
0.594	0.266	0.885	1	-0.548	1.105
-0.156	0.173	0.272	0.092	0.163	0.145
-0.256	-0.593	-1.012	-1.403	1.165	-1.901
-0.183	-0.496	-0.77	-1.538	1.78	-1.188
-0.871	-1.27	-1.655	-1.307	1.281	-2.037
-0.06	-0.11	-0.95	-0.719	1.159	-0.788
-1.135	0.163	-0.47	0.285	-0.023	-0.911
-0.609	-0.215	-0.681	-0.534	0.963	-0.612
0.393	0.388	-0.232	0.154	-0.42	-0.127
-2.163	-0.911	-1.61	-1.328	0.145	-1.952
0.831	0.6	0.953	0.953	-0.163	1.141
0.081	1.712	0.946	0.678	-0.257	1.723
0.836	0.799	0.782	1.529	-0.769	1.038
0.214	0.544	0.679	0.314	-0.347	0.51
-0.827	-0.759	-1.012	-0.517	0.197	-0.788
0.221	0.708	0.737	0.923	-0.495	0.899
-0.558	-0.343	-0.331	0.266	-0.756	-0.157
-0.684	-0.33	-0.185	0.349	-0.545	-0.155
-1.816	-0.952	-1.409	-1.882	0.639	-1.783
0.013	1.01	0.927	1.135	-0.387	0.937
0.344	0.856	0.372	0.327	0.227	0.36
-0.323	-0.772	-0.895	-1.002	0.517	-1.286
-0.82	0.608	-0.288	-0.311	0.386	-0.837
0.24	-0.906	-0.715	-1.467	1.626	-1.646
-0.571	-0.151	-0.635	-0.093	0.956	-0.73
0.789	0.821	-0.769	0.428	-1.815	-0.508
-0.353	0.115	-0.394	0.722	-0.089	-0.731
-0.467	-0.622	-0.661	-0.302	0.698	-0.986
-0.491	0.021	-0.221	-0.563	0.385	-0.672
0.848	0.837	1.54	1.112	-0.475	1.113
0.099	0.12	0.387	1.252	-0.638	0.655
-0.738	0.989	0.415	0.574	-0.647	0.093
-0.82	-0.702	1.203	1.579	-1.212	0.483
-0.629	-1.049	-1.271	-1.116	1.637	-1.407
-0.593	-0.898	-0.883	-0.647	0.323	-1.235

Favor (x_1)	Appearance (x_2)	Taste (x_3)	Stickiness (x_4)	Toughness (x_5)	Overall evaluation (y)
0.14	0.181	0.06	0.136	−0.234	0.146
−1.022	−0.827	−1.294	−0.583	0.437	−1.357
−0.649	−0.288	−0.257	0.327	−0.412	−0.653
−0.413	0.026	−0.108	0.393	−0.291	0.019
−0.358	−0.106	−1.275	−1.059	0.505	−1.168
−1.825	−2.018	−1.867	−1.517	0.114	−2.136
−0.002	−0.144	0.545	0.674	−0.111	0.641
0.158	0.163	0.03	0.359	−0.128	0.135
0.516	0.73	1.218	1.662	−0.366	1.44
−0.235	−0.566	−0.289	−0.763	0.048	−0.749
−1.085	−0.614	−1.24	−1.12	0.958	−0.65
−1.283	0.09	−0.651	−0.19	−0.131	−1.067
0.25	0.232	0.378	1.983	−1.231	0.519
0.288	0.205	0.147	0.399	0.365	0.352
0.567	1.502	0.824	0.53	−0.387	0.842
0.011	0.26	0.102	0.487	−0.402	0.067
−0.912	−0.911	−1.448	−1.666	1.375	−1.524
−0.588	0.792	1.023	0.995	−0.489	0.901
−0.686	−0.682	−0.514	−0.088	−0.43	−0.466
−2.871	0.59	−0.117	0.195	0.051	−1.084
−1.208	−0.257	−0.426	−0.237	0.24	−0.628
−1.819	−0.856	−1.758	−1.014	1.094	−1.925
0.713	1.172	0.648	1.065	−0.886	1.053
−0.248	0.185	0.204	−0.41	0.588	−0.186
−0.638	0.462	−0.149	−0.214	0.389	−0.23
0.255	0.98	0.837	1.64	0.028	1.747
−1.212	−0.074	−0.317	−0.144	0.18	−0.658
−0.093	0.712	0.594	1.27	−0.236	0.88
−0.315	−0.352	−0.741	−0.288	0.201	−0.964
0.021	0.805	0.367	0.19	0.33	0.056
0.158	0.265	0.053	0.373	−0.101	0.164
0.372	0.388	0.448	0.291	−0.402	0.489
−0.47	−0.301	−0.481	0.173	0.263	−0.546
−0.244	−0.044	−0.466	−0.464	0.469	−0.804
−0.049	0.401	−0.097	0.062	0.282	−0.223
−0.639	0.63	−0.089	0.166	−0.293	−0.394
0.039	0.524	0.207	1.082	−1.303	0.537
0.079	1.231	0.373	0.281	−1.273	0.218
0.104	−0.194	0.261	0.838	0.085	0.434
0.477	0.707	−0.172	0.097	−0.247	−0.246
0.468	0.333	0.511	1.188	−0.646	0.78
−0.465	−0.195	−0.491	−0.671	0.837	−0.614
0.125	0.127	−0.264	0.461	0.197	−0.046
−0.491	0.525	0.419	0.78	−0.512	0.291
−1.388	−0.114	−0.894	−0.168	0.028	−0.821
−0.955	−1.146	−1.107	−0.922	0.106	−1.367
−0.255	0.232	0.517	0.398	−0.247	0.424
−0.686	−0.252	−0.749	−0.428	0.048	−0.704
−0.016	0.674	−0.046	0.662	−0.694	0.326
−0.618	−0.425	0.098	0.471	0.158	0.104
0.57	1.542	1.302	1.842	−0.563	1.712
−1.134	−1.281	−1.031	−1.477	1.345	−1.624
0.124	0.588	0.053	−0.405	−0.322	−0.517
0.163	1.088	1.074	0.683	−0.135	0.913

References

- [1] H. Ichihashi and T. Watanabe, Learning control system by a simplified fuzzy reasoning model, *Proc. IPMU'90* (1990) 417–419.
- [2] H. Ishibuchi, K. Nozaki, H. Tanaka, Y. Hosaka and M. Matsuda, Empirical study on learning in fuzzy systems by rice taste analysis, *Fuzzy Sets and Systems* **64** (1994) 129–144.
- [3] J.S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, *IEEE Trans. Systems, Man Cybernet.* **23** (1993) 665–685.
- [4] C.L. Karr and E.J. Gentry, Fuzzy control of pH using genetic algorithms, *IEEE Trans. Fuzzy Systems* **1** (1993) 46–53.
- [5] B. Kosko, Fuzzy systems as universal approximators, *Proc. FUZZ-IEEE '92* (1992) 1153–1162.
- [6] C.C. Lee, Fuzzy logic in control systems, Fuzzy logic controller—Part I and Part II, *IEEE Trans. Systems, Man Cybernet.* **20** (1990) 404–435.
- [7] M. Matsuda and T. Kameoka, Application of fuzzy measure, fuzzy integral and neural network to the system which estimate taste by using industrial analysis, *Proc. IIZUKA '92* (1992) 601–606.
- [8] H. Nomura, I. Hayashi and N. Wakami, A learning method of fuzzy inference rules by descent method, *Proc. FUZZ-IEEE '92* (1992) 203–210.
- [9] H. Nomura, I. Hayashi and N. Wakami, A self-tuning method of fuzzy reasoning by genetic algorithm, *Proc. 1992 Int. Fuzzy Systems and Intelligent Control Conf.* (1992) 236–245.
- [10] M. Sugeno, An introductory survey of fuzzy control, *Inform. Sci.* **36** (1985) 59–83.
- [11] M. Sugeno and G.T. Kang, Structure identification of fuzzy model, *Fuzzy Sets and Systems* **28** (1988) 15–33.
- [12] M. Sugeno and T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, *IEEE Trans. Fuzzy Systems* **1** (1993) 7–31.
- [13] H. Takagi and I. Hayashi, NN-driven fuzzy reasoning, *Approximate Reasoning* **5** (1991) 191–212.
- [14] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *IEEE Trans. Systems, Man Cybernet.* **15** (1985) 116–132.
- [15] L.X. Wang and J.M. Mendel, Generating fuzzy rules by learning from examples, *IEEE Trans. Systems, Man Cybernet.* **22** (1992) 1414–1427.
- [16] L.X. Wang, Fuzzy systems are universal approximators, *Proc. FUZZ-IEEE '92* (1992) 1163–1170.