A Learning Algorithm for Tuning Fuzzy Rules Based on the Gradient Descent Method

Yan Shi*, Masaharu Mizumoto*, Naoyoshi Yubazaki** and Masayuki Otani**

*Division of Information and Computer Sciences
Osaka Electro-Communication University, Neyagawa, Osaka 572, Japan
shi@mzlab.osakac.ac.jp mizumoto@mzlab.osakac.ac.jp
**Department of Research and Development, Mycom Inc., Kyoto 616, Japan
yubazaki@mycom-japan.co.jp otani@mycom-japan.co.jp

Abstract

In this paper, we suggested an utility learning algorithm for tuning fuzzy rules by using training input-output data, based on the gradient descent method. The major advantage of this method is that fuzzy rules or membership functions can be learned without changing the form of the fuzzy rule table used in usual fuzzy controls, so that the case of weak-firing can be avoided well, which is different from the conventional learning algorithm. Furthermore, we illustrated the efficiency of the suggested learning algorithm by means of several numerical examples.

1. Introduction

As the development of fuzzy applications in industry, it is getting more important to generate or tune fuzzy rules by using some of learning techniques such as neuro-fuzzy learning algorithms. Based on the back-propagation method of neural networks [1], Ichihashi [2], Wang and Mendel [3] have proposed a neuro-fuzzy learning algorithm for generating or tuning fuzzy rules for a given fuzzy model with Gaussian-type membership functions. While the fitting for training data is fast and good by using the method, the number of tuning parameters increases quickly as the number of input variables increases, and that, the representation of the fuzzy rule base in the form fuzzy rule table [4] becomes hard or impossible. Thus, it is possible to occur the case of weak-firing for unknown data, so that it will affect the fuzzy inference result. To improve the above problems, in this paper, we suggest a new learning algorithm for tuning fuzzy rules by using training input-output data, based on the gradient descent method. The major advantage of this method is that fuzzy rules or membership functions can be tuned without changing the form of the fuzzy rule table, so that the case of weak-firing can be avoided well, which is different from the conventional learning algorithm [2,3]. Moreover, the properties of the conventional methods and the new method are discussed in detail. Finally, we illustrate the efficiency of the suggested learning algorithm by numerical examples.

2. The conventional learning algorithm

First, we introduce the conventional learning algorithm for learning fuzzy rules or membership functions, which is widely used in recent fuzzy controls and fuzzy expert systems [2,3,5,6].

For a given fuzzy system model with n input variables $x_1, x_2, ..., x_n$ and one output variable y, the fuzzy rule base is defined as follows [2,3]:

$$\text{Rule i: IF } x_1 \text{ is } A_{1i} \text{ and } x_2 \text{ is } A_{2i} \text{ and ... and } x_n \text{ is } A_{ni}, \quad \text{THEN } y \text{ is } y_i \quad (i = 1,2,...,m) \tag{1}$$

where A_{ji} (j=1,2,...,n; i=1,2,...,m) is a Gaussian-type membership function for the input variable x_j which is expressed in Eq.(2), y_i is a real number on the output universe Y, and m is the number of the fuzzy rules.

$$A_{ji}(x_j) = \exp(-(x_j-a_{ji})^2/b_{ji}) \quad j=1,2...,n;\ i=1,2,...,m \tag{2}$$

where a_{ji} is the center of A_{ji}, b_{ji} means the width of A_{ji}.

When an observation $(x_1, x_2, ..., x_n)$ is given, according to simplified fuzzy reasoning method [2], a fuzzy inference conclusion y can be obtained in the following way:

First, for i =1, 2, ..., m, the agreement of the i-th antecedent part is calculated by using the product operator:

$$h_i = A_{1i}(x_1)A_{2i}(x_2)...A_{ni}(x_n) = \prod_{j=1}^{n} A_{ji}(x_j) \quad i=1,2,...,m \tag{3}$$

Then, a consequence y is calculated by using the center of gravity method as follows:

$$y = (\sum_{i=1}^{m} h_i y_i) / (\sum_{i=1}^{m} h_i) \tag{4}$$

It should be noted that there is another form for Eq.(4) in [2], that is, a fuzzy consequence is obtained without operating the center of gravity, so the denominator is omitted. We shall adopt the form of Eq.(4) throughout our discussions.

When a training input-output datum $(x_1, x_2, \ldots, x_n; y^*)$ is given for the fuzzy system model, it is well known to use the following objective function E for evaluating an error between y^* and y, which can be regarded as an optimum problem:

$$E = (y^* - y)^2 / 2 \tag{5}$$

where y^* is a desired output value, and y is a fuzzy inference result.

In order to minimize the objective function E, a learning algorithm for updating the parameters a_{ji}, b_{ji}, and y_i has been proposed in [2,3] based on the gradient descent method [1,4], which is described as follows:

$$\begin{aligned} a_{ji}(t+1) &= a_{ji}(t) - \alpha\partial E/\partial a_{ji}(t) \\ &= a_{ji}(t) - \alpha(\partial E/\partial y)(\partial y/\partial h_i)(\partial h_i/\partial A_{ji})(\partial A_{ji}/\partial a_{ji}(t)) \\ &= a_{ji}(t) + 2\alpha(y^*-y)(y_i-y)h_i(x_j-a_{ji}) / (b_{ji}\sum_{i=1}^{m} h_i) \end{aligned} \tag{6}$$

$$\begin{aligned} b_{ji}(t+1) &= b_{ji}(t) - \beta\partial E/\partial b_{ji}(t) \\ &= b_{ji}(t) - \beta(\partial E/\partial y)(\partial y/\partial h_i)(\partial h_i/\partial A_{ji})(\partial A_{ji}/\partial b_{ji}(t)) \\ &= b_{ji}(t) + \beta(y^*-y)(y_i-y)h_i(x_j-a_{ji})^2 / (b_{ji}^2\sum_{i=1}^{m} h_i) \end{aligned} \tag{7}$$

$$\begin{aligned} y_i(t+1) &= y_i(t) - \gamma\partial E/\partial y_i(t) \\ &= y_i(t) - \gamma(\partial E/\partial y)(\partial y/\partial y_i(t)) \\ &= y_i(t) + \gamma(y^*-y)h_i / (\sum_{i=1}^{m} h_i) \end{aligned} \tag{8}$$

where α, β and γ are the learning rates that are the constants in the learning process, and t means the learning iteration.

The main advantages of the above learning algorithm are that: the training data are fitted fast and well because each membership function depends on only one fuzzy rule, and the iterative formula is simple for the computation.

However, as analyzed in the following, we shall point out some of problems on the learning algorithm of Eqs.(6)-(8).

Property (1): According to the arrangement Eq.(1) and the algorithm Eqs. (6)-(8), we can prove easily that each membership function A_{ji} (j=1,2,...,n; i=1,2,...,m) exists in the fuzzy rule base independently, only corresponding to i-th fuzzy rule [8]. This means that, first, the fuzzy partitions for all of input variables must be same and equal to the number of the fuzzy rules, although each input variable may have different behaviors in the fuzzy system model potentially; second, for each membership function A_{ji}, it is only used one time in the fuzzy rule base; third, for a multiple-inputs fuzzy system, the number of tuning parameters is great because of the equivalence of the fuzzy partition for each input variable and the number of the fuzzy rules.

For example, for a fuzzy system model with 4 input variables and one output variable, if we assume that 100 fuzzy rules are necessary, then the number of fuzzy sets for each input variable is also 100. In this case, the number of tuning parameters can be calculated as: 100 (the number of membership functions for each input variable) ×2 (the center and the width of the membership function) ×4 (the number of input variables) +100 (the number of the consequent parts) = 900. Need less to say, expressing 100 fuzzy sets by using linguistic variables for a fixed input variable is too hard. Obviously, these stronger restrictions for membership functions of antecedent parts are inconvenient for constructing a fuzzy system model in real-world applications.

Property (2): Based on **Property (1)**, the representation of the fuzzy rule base in the form of the fuzzy rule table used in usual fuzzy control and other fuzzy application fields becomes very difficult or impossible under the conventional method, in generally.

For example, for a fuzzy system model with two variables x_1, x_2 and one output variable y, it is well known that the following representation of the fuzzy rule base in the form of the fuzzy rule table is a most common thing in fuzzy controls as shown in Figure 1 [4], where A_{1i} (i=1,2,3,4) is a fuzzy set for x_1, A_{2j} (j=1,2,3) is a fuzzy set for x_2, and $\boldsymbol{R}_k$ (k=1,2,..,12) denotes a fuzzy rule.

From Figure 1, one can see that, first, the fuzzy partition for x_1 and the fuzzy partition for x_2 may be different because of the behaviors of input variables; second, it allows that each membership function is used more than one time (for example, $A_{11}, A_{21} \Rightarrow \boldsymbol{R}_1$, $A_{11}, A_{22} \Rightarrow \boldsymbol{R}_2$, etc.); third, the input space is covered by the fuzzy rules enough.

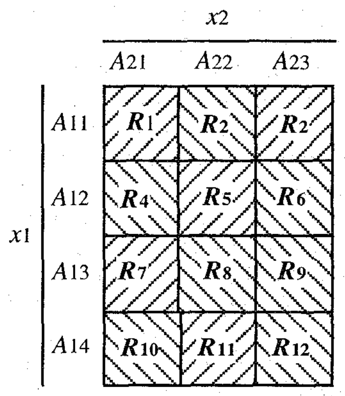


Figure 1. Representation of fuzzy rule base in the form of the fuzzy rule table

On the other hand, according to **Property (1)**, the representation of the fuzzy rule base in the form of the fuzzy rule table by using the conventional method must be the following form as shown in Figure 2, if we assume 9 fuzzy rules to be necessary for the fuzzy system model. Because the support of each Gaussian-type membership function is infinite, for any α ($\in [0,1]$), we can obtain all of α-cut sets, corresponding to the membership functions as shown in Figure 2. In this case, one can understand that all of the agreements of antecedent parts will be $h_i = A_{1i}(x_1)A_{2i}(x_2) < \alpha$ (i = 1,2,...,9), if (x_1,x_2) is given in out of $\boldsymbol{R}_i$ (i = 1,2,...,9). Clearly, if α is enough small, the fuzzy inference rules are very weak for unknown data given in the blank places as shown in Figure 2. This shows that expressing fuzzy inference rules in the form of the fuzzy rule table is very hard or impossible.

Figure 2 does not only restrict the fuzzy partition for each input variable, but also makes each fuzzy set to be only used one time in the fuzzy rule base. Especially, since the input space can not be covered by the fuzzy rules enough, the case of weak-firing will occur when we input (x_1,x_2) to be out of $\boldsymbol{R}_i$ (i=1,2,...,9) as shown in Figure 2.

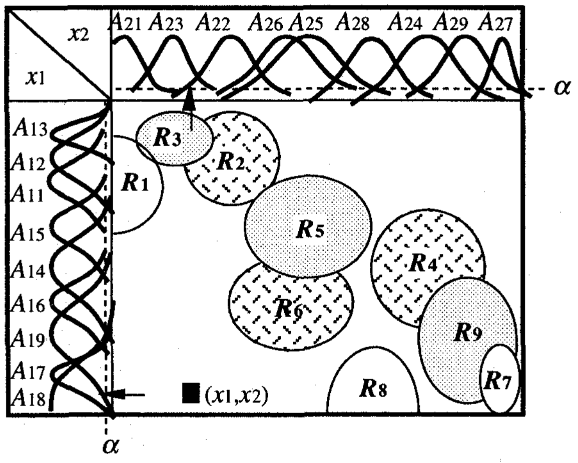


Figure 2. Representation of fuzzy rule base in the form of the fuzzy rule table under the conventional method

In Figure 2, for example, let an observation (x_1,x_2) be given as shown in the $\leftarrow$, $\uparrow$, then one can see that no suitable fuzzy rule exists corresponding to (x_1,x_2). In this case, all of grades $A_{1i}(x_1)$, $A_{2i}(x_2)$ (i=1,2,...,9) can be calculated as follows:

$$A_{1i}(x_1) \geq \alpha,\ i = 7,8,9;\ \ A_{1i}(x_1) < \alpha, \text{ otherwise;}$$

$$A_{2i}(x_1) \geq \alpha,\ i = 2,3;\ \ A_{1i}(x_1) < \alpha, \text{ otherwise.}$$

Thus, all of agreements become $h_i = A_{1i}(x_1)A_{2i}(x_2) < \alpha$ (i=1,2,...,9) to be very small if α is small enough, that is, there exists a weak-firing state. Clearly, the fuzzy consequence will be influenced when a weak-firing state occurs.

Therefore, we can say that the conventional learning algorithm sometimes is not suitable to a multiple-input fuzzy system, or is inconvenient to construct a fuzzy rule base in a form of the fuzzy rule table because of the above problems.

In order to improve above problems, we shall suggest a new learning algorithm for tuning fuzzy rules by using training input-output data, based on the gradient descent method.

3. A new learning algorithm for tuning fuzzy rules

For convenience, we shall derive our new learning algorithm in which the fuzzy system model has only two input variables and one output variable. It is not difficult to extend our idea to the case of multiple input variables in the same way.

First, like Eq.(2), the Gaussian-type membership functions A_{1i}, A_{2j} (i=1,2,...,r; j=1,2,...,k) for input variables x_1 and x_2 are defined as follows [8]:

$$A_{1i}(x_1) = \exp(-(x_1-a_{1i})^2/b_{1i}) \qquad i=1,2,...,r \qquad (9)$$

$$A_{2j}(x_2) = \exp(-(x_2-a_{2j})^2/b_{2j}) \qquad j=1,2,...,k \qquad (10)$$

where r means the number of fuzzy partitions for x_1, k means the number of fuzzy partitions for x_2. Obviously, it allows the case of r $\neq$ k, which is different from the conventional method [4,5].

Then, we assume that a fuzzy rule base is defined based on all of the combinations of A_{1i} and A_{2j} (i=1,...,r; j=1,...,k), which is different from Eq.(1) as shown in Eq.(11):

$$\begin{array}{ll}
\text{Rule 1:} & A_{11}, A_{21} \Rightarrow y_1 \\
\text{Rule 2:} & A_{11}, A_{22} \Rightarrow y_2 \\
 & \cdots\cdots \\
\text{Rule k:} & A_{11}, A_{2k} \Rightarrow y_k \\
\text{Rule k+1:} & A_{12}, A_{21} \Rightarrow y_{k+1} \\
 & \cdots\cdots \\
\text{Rule 2k:} & A_{12}, A_{2k} \Rightarrow y_{2k} \\
 & \cdots\cdots \\
\text{Rule (i-1)k+j:} & A_{1i}, A_{2j} \Rightarrow y_{(i-1)k+j} \\
 & \cdots\cdots \\
\text{Rule r×k:} & A_{1r}, A_{2k} \Rightarrow y_{r\times k}
\end{array} \qquad (11)$$

where $y_{(i-1)k+j}$ (i=1,2,...,r; j=1,2,...,k) is a real number on the output universe Y.

Clearly, we can express the above fuzzy rule base Eq.(11) in the form of the fuzzy rule table as the same as Figure 2, which is shown in Table 1 [7,8].

Table 1. Fuzzy rule table for Eq.(11)

x1 \ x2	A21	A22	...	A2j	...	A2k
A11	y1	y2	···	yj	···	yk
A12	yk+1		······			y2k
			··········			
A1i	···			y(i-1)k+j		···
			··········			
A1r	y(r-1)k+1		······			yrk

According to the above assumption, when an observation (x_1,x_2) is given, a fuzzy inference conclusion y can be obtained by using the simplified fuzzy reasoning method as follows:

First, the agreement of A_{1i} and A_{2j} at (x_1,x_2) is calculated by using the product operator:

$$h_{(i-1)k+j} = A_{1i}(x_1)A_{2j}(x_2) \quad i=1,2,...,r;\ j=1,2,...,k \tag{12}$$

Then, a consequence y is calculated by using the center of gravity method:

$$y = \left(\sum_{i=1}^{r}\sum_{j=1}^{k} h_{(i-1)k+j}y_{(i-1)k+j}\right) / \left(\sum_{i=1}^{r}\sum_{j=1}^{k} h_{(i-1)k+j}\right)$$
$$= \left(\sum_{i=1}^{r}\sum_{j=1}^{k} A_{1i}(x_1)A_{2j}(x_2)y_{(i-1)k+j}\right)/\left(\sum_{i=1}^{r}\sum_{j=1}^{k} A_{1i}(x_1)A_{2j}(x_2)\right) \tag{13}$$

In order to minimize the objective function E (see Eq.(5)), a new learning algorithm for updating the parameters a_{1i}, b_{1i}, a_{2j}, b_{2j} and $y_{(i-1)k+j}$ ($i=1,2,...,r$; $j=1,2,...,k$) is derived based on the gradient descent method as follows:

$$a_{1i}(t+1) = a_{1i}(t) - \alpha\partial E/\partial a_{1i}(t)$$
$$= a_{1i}(t) - \alpha(\partial E/\partial y)(\partial y/\partial h_{(i-1)k+j})(\partial h_{(i-1)k+j}/\partial A_{1i})\times(\partial A_{1i}/\partial a_{1i}(t))$$
$$= a_{1i}(t) + \frac{2\alpha(y^*-y)(x_1-a_{1i})\sum_{j=1}^{k} h_{(i-1)k+j}\,(y_{(i-1)k+j}-y)}{b_{1i}\sum_{i=1}^{r}\sum_{j=1}^{k} h_{(i-1)k+j}} \tag{14}$$

$$b_{1i}(t+1) = b_{1i}(t) - \beta\partial E/\partial b_{1i}(t)$$
$$= b_{1i}(t) - \beta(\partial E/\partial y)(\partial y/\partial h_{(i-1)k+j})(\partial h_{(i-1)k+j}/\partial A_{1i})\times(\partial A_{1i}/\partial b_{1i}(t))$$
$$= b_{1i}(t) + \frac{\beta(y^*-y)(x_1-a_{1i})^2\sum_{j=1}^{k} h_{(i-1)k+j}\,(y_{(i-1)k+j}-y)}{b_{1i}^2\sum_{i=1}^{r}\sum_{j=1}^{k} h_{(i-1)k+j}} \tag{15}$$

$$a_{2j}(t+1) = a_{2j}(t) - \alpha\partial E/\partial a_{2j}(t)$$
$$= a_{2j}(t) - \alpha(\partial E/\partial y)(\partial y/\partial h_{(i-1)k+j})(\partial h_{(i-1)k+j}/\partial A_{2j})\times(\partial A_{2j}/\partial a_{2j}(t))$$
$$= a_{2j}(t) + \frac{2\alpha(y^*-y)(x_2-a_{2j})\sum_{i=1}^{r} h_{(i-1)k+j}\,(y_{(i-1)k+j}-y)}{b_{2j}\sum_{i=1}^{r}\sum_{j=1}^{k} h_{(i-1)k+j}} \tag{16}$$

$$b_{2j}(t+1) = b_{2j}(t) - \beta\partial E/\partial b_{2j}(t)$$
$$= b_{2j}(t) - \beta(\partial E/\partial y)(\partial y/\partial h_{(i-1)k+j})(\partial h_{(i-1)k+j}/\partial A_{2j})\times(\partial A_{1i}/\partial b_{2j}(t))$$
$$= b_{2j}(t) + \frac{\beta(y^*-y)(x_2-a_{2j})^2\sum_{i=1}^{r} h_{(i-1)k+j}\,(y_{(i-1)k+j}-y)}{b_{2j}^2\sum_{i=1}^{r}\sum_{j=1}^{k} h_{(i-1)k+j}} \tag{17}$$

$$y_{(i-1)k+j}(t+1) = y_{(i-1)k+j}(t) - \gamma\partial E/\partial y_{(i-1)k+j}(t)$$
$$= y_{(i-1)k+j}(t) - \gamma(\partial E/\partial y)(\partial y/\partial y_{(i-1)k+j}(t))$$
$$= y_{(i-1)k+j}(t) + \gamma(y^*-y)h_{(i-1)k+j} / \left(\sum_{i=1}^{r}\sum_{j=1}^{k} h_{(i-1)k+j}\right) \tag{18}$$

where α, β and γ are the learning rates, and t means the learning iteration.

Obviously, the iterative formulas Eqs.(14)-(18) derived under the assumption Eq.(11) are different from the iterative formulas Eqs.(6)-(8). The differences between the new method and the conventional method lead to that the problems of the conventional method analyzed in last section have been improved well by the new method, namely, first, the fuzzy partitions for each input variable is allowed to be different from other ones; second, a membership function can be used more than one time in the fuzzy rule base; third, the representation of the fuzzy rule base in Table 1 is never destroyed even after the learning process, so that the case of weak-firing can be avoided well by the new method, in general.

For example, if we take r = 3, k = 4 in Eq.(9) and Eq.(10), then by using the new method we get the representation of the fuzzy rule base in the form of the fuzzy rule table as shown in Figure 3, corresponding to Figure 2.

Unlike the conventional method, the fuzzy rules can tuned without changing the form of the fuzzy rule table by

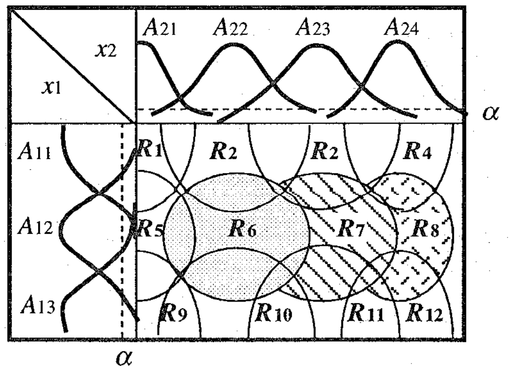


Figure 3. Representation of fuzzy rule base in the form of the fuzzy rule table under the new method

using the new method, the problems in the conventional method have been solved, and that, the expression of fuzzy rules is intuitive and convenient for a practical application.

4. Numerical examples

In the following, we apply our method to the following two nonlinear functions with two input variables and one output variable to compare it with the conventional learning algorithm for identifying and evaluating the problems, and show the efficiency of the new method. The two nonlinear functions are given as follows:

$$\text{Func.I:} \quad y = (2x_1+4x_2+0.1)^2 / 37.21 \tag{19}$$

$$\text{Func.II:} \quad y = [4\sin(\pi x_1)+2\cos(\pi x_2)]/12 + 0.5 \tag{20}$$

where $x_1, x_2 \in [-1,1]$ are input variables, and $y \in [0,1]$ is a normalized output variable.

First, we can arrange two kinds of initial fuzzy rules under the above two learning algorithms respectively, as shown in Table 2 and Table 3.

In Table 2, there exist five membership functions for each input variable, where -1, -0.5, 0, 0.5 and 1 are the centers of the membership functions, b = 0.09 means a width of each membership function, and 0 denotes a real number of the consequent part.

Table 2. Initial fuzzy rules under the new method

$A2j$ \ $A1i$	(-1, b)	(-0.5, b)	(0, b)	(0.5, b)	(1, b)
(-1,b)	0	0	0	0	0
(-0.5,b)	0	0	0	0	0
(0,b)	0	0	0	0	0
(0.5,b)	0	0	0	0	0
(1,b)	0	0	0	0	0

On the other hand, for a fuzzy model with 25 fuzzy rules, the initial fuzzy rules must be given in the form of Table 3 under the conventional method, otherwise, a case of weak-firing will occur before the learning [8].

Then, 49 training data are employed for identifying Func.I and Func.II, respectively. In our case, the learning rates are taken as $\alpha = 0.01$, $\beta = 0.05$ and $\gamma = 0.65$. Each method stops the learning process when the inference error D for identifying data is less than the threshold δ. In this case, δ is taken as 0.005 for both Func.I and Func.II. Here, D is defined as follows:

$$D = \sum_{d=1}^{49} (y^*_d - y_d)^2 / 2 \tag{21}$$

were y^*_d (d=1,2,...,49) is a desired output value, and y_d is a fuzzy inference value.

Table 3. Initial fuzzy rules under the conventional method

No.	$A1i$	$A2i$	yi
1	(-1,b)	(-1,b)	0
2	(-1,b)	(-0.5,b)	0
3	(-1,b)	(0,b)	0
4	(-1,b)	(0.5,b)	0
5	(-1,b)	(1,b)	0
6	(-0.5,b)	(-1,b)	0
7	(-0.5,b)	(-0.5,b)	0
8	(-0.5,b)	(0,b)	0
9	(-0.5,b)	(0.5,b)	0
10	(-0.5,b)	(1,b)	0
11	(0,b)	(-1,b)	0
12	(0,b)	(-0.5,b)	0
13	(0,b)	(0,b)	0
14	(0,b)	(0.5,b)	0
15	(0,b)	(1,b)	0
16	(0.5,b)	(-1,b)	0
17	(0.5,b)	(-0.5,b)	0
18	(0.5,b)	(0,b)	0
19	(0.5,b)	(0.5,b)	0
20	(0.5,b)	(1,b)	0
21	(1,b)	(-1,b)	0
22	(1,b)	(-0.5,b)	0
23	(1,b)	(0,b)	0
24	(1,b)	(0.5,b)	0
25	(1,b)	(1,b)	0

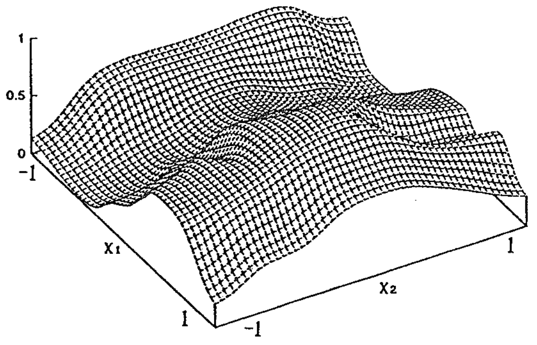
Table 4. Comparison between the new method (B) and the conventional method (A) for Func.I

No.	Iteration of learning		Error of evaluation		Maximum absolute error	
	(A)	(B)	(A)	(B)	(A)	(B)
1	63	515	0.0084	0.0037	0.8952	0.7710
2	51	306	0.0069	0.0050	0.9020	0.8180
3	62	118	0.0020	0.0015	0.2350	0.1892
4	89	783	0.0177	0.0117	0.9725	0.9022
5	49	95	0.0009	0.0009	0.1558	0.1200

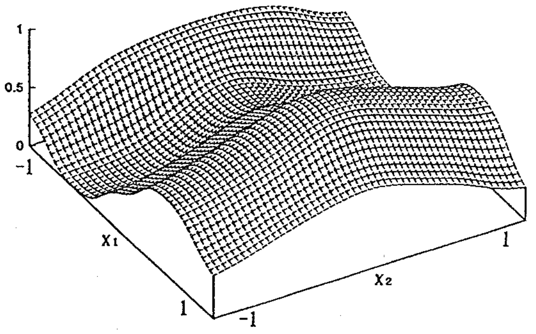
Table 4 and Table 5 show the iteration of the learning for training data, the error of evaluation and the maximum absolute error for checking data for identifying Func.I and Func.II respectively, where (A) denotes the case of the conventional method and (B) means the case of the new method. Here, the error of evaluation denotes a mean square error for the checking data. In our case, 2601 checking data (x_1,x_2) are employed from (-1,-1) to (1,1), equally.

Table 5. Comparison between the new method (*B*) and the conventional method (*A*) for Func.II

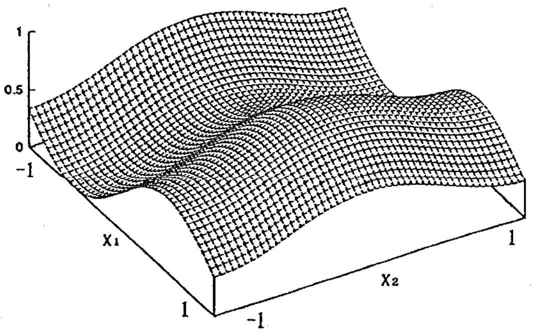
No.	Iteration of learning (*A*)	Iteration of learning (*B*)	Error of evaluation (*A*)	Error of evaluation (*B*)	Maximum absolute error (*A*)	Maximum absolute error (*B*)
1	54	70	0.0041	0.0026	0.3560	0.2136
2	216	87	0.0059	0.0011	0.3559	0.0974
3	59	142	0.0108	0.0044	0.4452	0.3064
4	42	69	0.0021	0.0010	0.2417	0.1536
5	31	208	0.0094	0.0043	0.5620	0.3688



(a) Fuzzy inference results by the conventional method

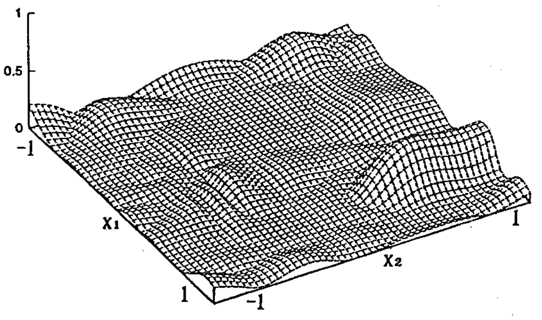


(b) Fuzzy inference results by the new method

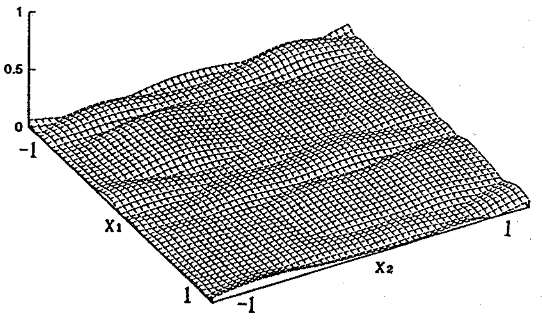


(c) The desired output results

Figure 4. Comparison between two kinds of fuzzy inference results



(a) Fuzzy inference errors by the conventional method



(b) Fuzzy inference errors by the new method

Figure 5. Comparison between two kinds of fuzzy inference errors

One can see from Table 4 and Table 5 that, the iteration of the learning by using the conventional algorithm is smaller than the new method, because each membership function is independent with others, so that, the training data is usual fitted fast. However, one should note a fact that the number of tuning parameters under the new method is smaller than the conventional method (in this case, the number of the new method is 45, the number of the conventional method is 125). The learning times of both methods are not so different.

On the other hand, one can also see the rather finer evaluation for both Func.I and Func.II by the new method than the conventional method, according to Table 4 and Table 5.

As an illustration, Figure 4 (a), (b) show the two fuzzy inference results for Func.II using the fuzzy rule bases *No*.2 in Table 5, which were generated by the conventional method and the new method, respectively, Figure 4 (c) shows the corresponded desired output results for Func.II. Moreover, Figure 5 (a), (b) show the errors of evaluation corresponding to Figure 4 (a), (b), receptively.

From Figure 4 and Figure 5, one can see that the new method has a good approximation. Since the input space is covered by the fuzzy rule base even after the learning process, so that, the fuzzy inference results are smooth.

On the other hand, while the fitting for training data is often good under the conventional method, the input space can not be covered by the fuzzy rule base enough after the learning process, so that the fitting for unknown data must be not always well.

5. Conclusion

We have suggested a new learning algorithm for tuning fuzzy rules, which is different from the conventional method. The best advantage of the new method is that the fuzzy rules can be tuned without changing the form of the fuzzy rule table, so that the case of weak-firing occurring in the conventional method can be avoided well.

However, the speed of training data is sometimes fitted slow under the new method if the number of the fuzzy partitions is too small for each input variable, because of the dependency of the membership functions. Therefore, it should be considered to give suitable fuzzy partitions for a practical fuzzy system model.

References

[1] J.E.Dayhoff, *Neural Network Architectures: An Introduction*, Van Nostrand Reinhold, New York (1990)

[2] H.Ichihashi, Iterative fuzzy modeling and a hierarchical network, *Proceedings of the 4th IFSA Congress*, Vol. Engineering, Brussels, 49-52 (1991)

[3] B.Kosko, *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ 07632 (1992)

[4] L.X. Wang and J.M. Mendel, Back-propagation fuzzy system as nonlinear dynamic system identifiers, *Proceedings of the IEEE International Congress on Fuzzy Systems*, San Diego, 1409-1416 (1992)

[5] R.R.Yager and D.P.Filev, Generation of fuzzy rules by mountain clustering, *International Journal of Intelligent and Fuzzy Systems*, Vol. 2, 209-219 (1994)

[6] R.R.Yager and D.P.Filev, *Essentials of Fuzzy Modeling and Control*, John Wiley & Sons (1994)

[7] Y. Shi, M. Mizumoto, N. Yubazaki and M. Otani, A neuro-fuzzy learning algorithm adapted to a fuzzy rule table, *Proceedings of the 40th Annual Conference of the Institute of Systems, Control and Information Engineers*, Kyoto, 131-132 (1996, in Japanese)

[8] Y.Shi, M.Mizumoto, N.Yubazaki and M.Otani, A method of generating fuzzy rules based on the neuro-fuzzy learning algorithm, *Journal of Japan Society for Fuzzy Theory and Systems* (in press, in Japanese)