A Learning Algorithm of Fuzzy Neural Networks with Trapezoidal Fuzzy Weights

Kyu-Hee Lee and Sung-Bae Cho

Computer Science Department, Yonsei University
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea
Phone : +82-2-361-2720 Fax : +82-2-365-2579 Email : sbcho@csai.yonsei.ac.kr

Abstract

In this paper, we propose a learning algorithm of fuzzy neural networks with trapezoidal fuzzy weights. This fuzzy neural networks can use fuzzy numbers as well as real numbers, and represent linguistic information better than standard neural networks. We construct trapezoidal fuzzy weights by the composition of two triangles, and devise a learning algorithm using the two triangular membership functions. The results of computer simulations on numerical data show that the fuzzy neural networks have high fitting ability for target output.

Keywords: fuzzy neural networks, trapezoidal fuzzy weights, fuzzy inputs, fuzzy target outputs, learning algorithm

1. Introduction

It is very difficult to represent linguistic information from human experts using only numerical data. For better representation of linguistic knowledge, fuzzy if-then rules are used very often. Furthermore, neural networks using fuzzy numbers as well as numerical numbers as inputs and outputs have been proposed in order to integrate neural networks and fuzzy logic [2,4,5,7]. Fuzzy neural networks using triangular fuzzy numbers as weights were proposed by H. Ishibuchi et al. [3]. Extending these concepts, this paper proposes a learning algorithm and architecture of fuzzy neural networks which utilize not only triangular fuzzy numbers but also trapezoidal fuzzy numbers as weights.

We construct a trapezoid as composition of two triangles and devise a learning algorithm using the triangular membership functions. Hence, each trapezoidal fuzzy weight has six parameters and is modified through learning without destruction of its trapezoidal shape. This fuzzy neural network is three layer feedforward neural network. The fuzzy numbers used in each unit of this neural network are represented by closed intervals of the associate values of h levels, and numerically calculated for h-level sets. The interval arithmetic is used for the calculation of fuzzy numbers. It requires the modification of standard back propagation algorithm. To show the usefulness of the proposed fuzzy neural networks, we apply them to several examples.

2. Fuzzy Neural Networks

2.1 Architecture

The input-output relation of a three-layer feedforward neural network can be formulated as follows. This neural network has n_I input units, n_H output units, and n_O output units. The fuzzy weights and fuzzy biases are trapezoidal fuzzy numbers, and the inputs and target outputs are any shape of fuzzy numbers.

- Input units :

$$O_{pi} = X_{pi},\quad i=1,2,\ldots,n_I$$

- Hidden units :

$$O_{pj} = f(Net_{pj}),\quad j=1,2,\ldots n_H$$

$$Net_{pj} = \sum_{i=1}^{n_I} W_{ji}O_{pi} + \Theta_j$$

- Output units :

$$O_{pk} = f(Net_{pk}),\quad k = 1, 2, \ldots, n_O$$
$$Net_{pk} = \sum_{j=1}^{n_H} W_{kj} O_{pj} + \Theta_k$$

where X_{pi} is a fuzzy input, W_{ji} and W_{kj} are trapezoidal fuzzy weights, and Θ_j and Θ_k are trapezoidal fuzzy biases. Fig. 1 represents the architecture of this fuzzified neural network.

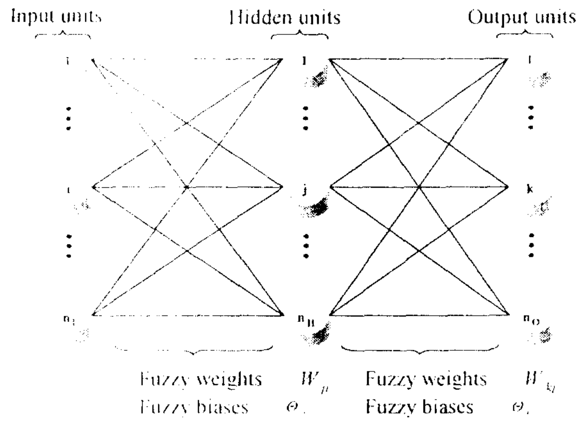

Fig. 1. Architecture of fuzzy neural network

2.2 Fuzzy Numbers

The operations of fuzzy numbers can be defined by the extension principle of Zadeh. The fuzzy outputs are numerically calculated by interval arithmetic for level sets of fuzzy weights and fuzzy inputs [1]. In addition, multiplication and function of fuzzy numbers are defined as following.

$$\mu_{A+B}(z) = \max\{\mu_A(x) \wedge \mu_B(y) | z = x + y\},$$
$$\mu_{AB}(z) = \max\{\mu_A(x) \wedge \mu_B(y) | z = xy\},$$
$$\mu_{f(X)}(z) = \max\{\mu_X(x) | z = f(x)\},$$

where A, B and X are fuzzy numbers. These operations are illustrated in Fig. 2.

The h-level set is defined as

$$[X]_h = \{x | \mu_X(x) \geq h, x \in \mathbb{R}\} \text{ for } 0 < h \leq 1,$$

where $\mu_X(x)$ is membership function of X. We denote h-level set of X as

$$[X]_h = [[X]_h^L, [X]_h^U],$$

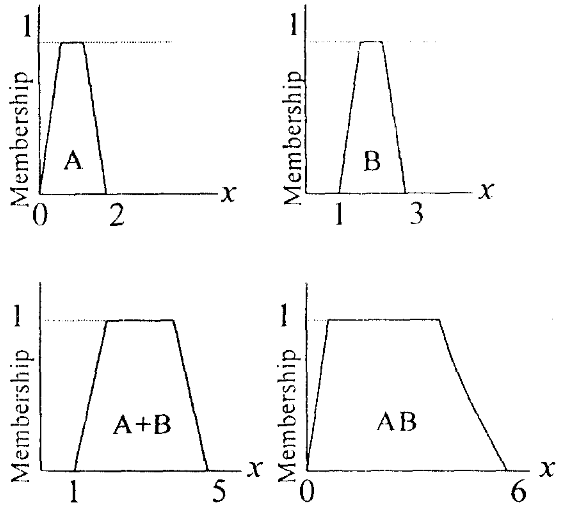

Fig 2. Illustration of fuzzy arithmetic

where $[X]_h^L$ and $[X]_h^U$ are the lower and upper limits of the h level set of X. Then we can write above operations of fuzzy numbers as

$$[A]_h + [B]_h = [[A]_h^L + [B]_h^L, [A]_h^U + [B]_h^U]$$
$$[A]_h \cdot [B]_h = [\min\{[A]_h^L \cdot [B]_h^L, [A]_h^L \cdot [B]_h^U\}, \max\{[A]_h^U \cdot [B]_h^L, [A]_h^U \cdot [B]_h^U\}]$$

where B is nonnegative. It is assumed that input numbers of the fuzzy neural network are nonnegative.

Our approach is based on the level sets of fuzzy numbers, and arbitrary number of h can be used. For example, Fig. 3 represents a fuzzy number denoted by 6 level sets for h=0.0, 0.2,...,1.0.

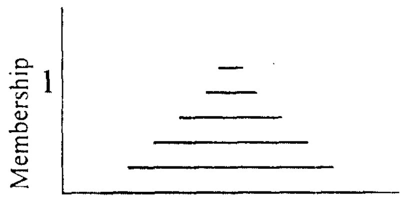

Fig 3. Fuzzy numbers in fuzzy neural network

3. Learning Algorithm

3.1 Cost Function

Let T_p be the n_O dimensional fuzzy target output, and O_{pk} be the actual output. Then the cost function for the kth output of level h is

$$e_{pkh} = e^L_{pkh} + e^U_{pkh},$$

$$e^L_{pkh} = h \cdot \frac{([T_{pk}]^L_h - [O_{pk}]^L_h)^2}{2}$$

$$e^U_{pkh} = h \cdot \frac{([T_{pk}]^U_h - [O_{pk}]^U_h)^2}{2}$$

Then cost function of the h level sets is

$$e_{ph} = \sum_{k=1}^{n_o} e_{pkh}$$

Hence we can write the cost function of (X_p, T_p) as

$$e_p = \sum_h e_{ph}.$$

3.2 Derivation of Learning Algorithm

In this section we derive a learning algorithm of the neural network from the cost function describd in the previous section. The fuzzy weights and biases are adjusted to minimize the value of cost function. Since the adjustment should not distort the trapezoidal shape of weights and biases, we do not change the h-level set of the fuzzy weights independently. In this paper, we construct a trapezoid as the composition of two symmetric triangles (shown in Fig. 4). We can update each trapezoidal fuzzy weights without destructing its shape by the movement of the two triangles and the change of their width.

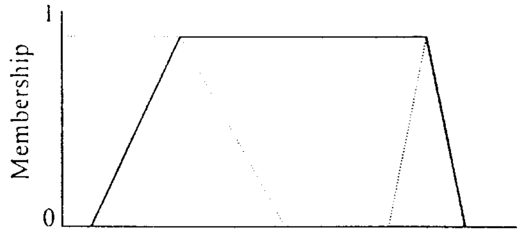

Fig 4. Trapezoidal fuzzy weights

Then, let the trapezoidal fuzzy weights have six parameters as

$$W_{kj} = (w^{L1}_{kj}, w^{C1}_{kj}, w^{U1}_{kj}, w^{L2}_{kj}, w^{C2}_{kj}, w^{U2}_{kj}),$$

$$W_{ji} = (w^{L1}_{ji}, w^{C1}_{ji}, w^{U1}_{ji}, w^{L2}_{ji}, w^{C2}_{ji}, w^{U2}_{ji}),$$
$$\Theta_k = (\theta^{L1}_k, \theta^{C1}_k, \theta^{U1}_k, \theta^{L2}_k, \theta^{C2}_k, \theta^{U2}_k),$$
$$\Theta_j = (\theta^{L1}_j, \theta^{C1}_j, \theta^{U1}_j, \theta^{L2}_j, \theta^{C2}_j, \theta^{U2}_j).$$

Fig. 5 shows the parameters of a trapezoidal fuzzy weights. Since the two triangles composing a trapezoidal fuzzy weight are symmetric, we can write the weights as follows,

$$w^{C1}_{kj} = \frac{w^{L1}_{kj} + w^{U1}_{kj}}{2}, \quad w^{C1}_{ji} = \frac{w^{L1}_{ji} + w^{U1}_{ji}}{2},$$
$$w^{C2}_{kj} = \frac{w^{L2}_{kj} + w^{U2}_{kj}}{2}, \quad w^{C2}_{ji} = \frac{w^{L2}_{ji} + w^{U2}_{ji}}{2},$$
$$\theta^{C2}_k = \frac{\theta^{L2}_k + \theta^{U2}_k}{2}, \quad \theta^{C2}_j = \frac{\theta^{L2}_j + \theta^{U2}_j}{2},$$
$$\theta^{C1}_k = \frac{\theta^{L1}_k + \theta^{U1}_k}{2}, \quad \theta^{C1}_j = \frac{\theta^{L1}_j + \theta^{U1}_j}{2},$$

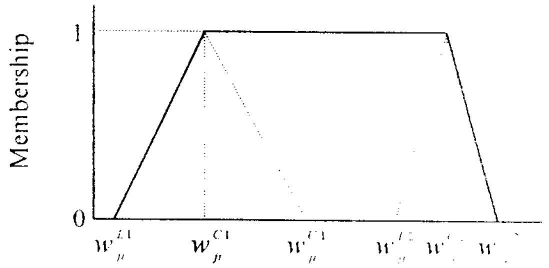

Fig 5. Parameters of trapezoidal fuzzy weights

We derive a rule of adjustment of the weights for each parameter. According to the standard back-propagation algorithm [6], parameters w^{L1}_{kj} and w^{U1}_{kj} are updated by the following formula, and the other weights are updated in the same way.

$$w^{L1}_{kj}(t+1) = w^{L1}_{kj}(t) + \Delta w^{L1}_{kj}(t),$$
$$w^{U1}_{kj}(t+1) = w^{U1}_{kj}(t) + \Delta w^{U1}_{kj}(t),$$
$$\Delta w^{L1}_{kj}(t) = -\eta \cdot \frac{\partial e_{ph}}{\partial w^{L1}_{kj}} + \alpha \cdot \Delta w^{L1}_{kj}(t-1),$$
$$\Delta w^{U1}_{kj}(t) = -\eta \cdot \frac{\partial e_{ph}}{\partial w^{U1}_{kj}} + \alpha \cdot \Delta w^{U1}_{kj}(t-1),$$

where t is index of the number of iterations through learning, η is positive real constant, and α is positive real constant less than 1.0. Then the derivatives are written as follows.

$$\frac{\partial e_{ph}}{\partial w_{kj}^{L1}} = \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{L1}} \cdot \frac{\partial [W_{kj}]_h^{L1}}{\partial w_{kj}^{L1}} + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{U1}} \cdot \frac{\partial [W_{kj}]_h^{U1}}{\partial w_{kj}^{L1}}$$

$$+ \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{L2}} \cdot \frac{\partial [W_{kj}]_h^{L2}}{\partial w_{kj}^{L1}} + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{U2}} \cdot \frac{\partial [W_{kj}]_h^{U2}}{\partial w_{kj}^{L1}},$$

$$\frac{\partial e_{ph}}{\partial w_{kj}^{U1}} = \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{L1}} \cdot \frac{\partial [W_{kj}]_h^{L1}}{\partial w_{kj}^{U1}} + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{U1}} \cdot \frac{\partial [W_{kj}]_h^{U1}}{\partial w_{kj}^{U1}}$$

$$+ \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{L2}} \cdot \frac{\partial [W_{kj}]_h^{L2}}{\partial w_{kj}^{U1}} + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{U2}} \cdot \frac{\partial [W_{kj}]_h^{U2}}{\partial w_{kj}^{U1}}.$$

In the meantime, followings are obvious for each h-level.

$$[W_{kj}]_h^{L1} = w_{kj}^{L1} \cdot (1-\frac{h}{2}) + w_{kj}^{U1} \cdot \frac{h}{2},$$

$$[W_{kj}]_h^{U1} = w_{kj}^{L1} \cdot \frac{h}{2} + w_{kj}^{U1} \cdot (1-\frac{h}{2}),$$

$$[W_{kj}]_h^{L2} = w_{kj}^{L2} \cdot (1-\frac{h}{2}) + w_{kj}^{U2} \cdot \frac{h}{2},$$

$$[W_{kj}]_h^{U2} = w_{kj}^{L2} \cdot \frac{h}{2} + w_{kj}^{U2} \cdot (1-\frac{h}{2}).$$

Hence.

$$\frac{\partial e_{ph}}{\partial w_{kj}^{L1}} = \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{L1}} \cdot (1-\frac{h}{2}) + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{U1}} \cdot \frac{h}{2},$$

$$\frac{\partial e_{ph}}{\partial w_{kj}^{U1}} = \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{L1}} \cdot \frac{h}{2} + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^{U1}} \cdot (1-\frac{h}{2}).$$

In this way, the amount of adjustment is calculated. Finally, the weights can be updated as follows.

$$w_{kj}^{L1}(t+1) = w_{kj}^{L1}(t) + \Delta w_{kj}^{L1}(t),$$

$$w_{kj}^{U1}(t+1) = w_{kj}^{U1}(t) + \Delta w_{kj}^{U1}(t).$$

$$w_{kj}^{C1}(t+1) = \frac{w_{kj}^{L1}(t+1) + w_{kj}^{U1}(t+1)}{2}.$$

If $w_{kj}^{L1}(t+1)$ is larger than $w_{kj}^{U1}(t+1)$. the two values are swaped.

w_{kj}^{L2} and w_{kj}^{U2} are modified in the same way. An example for the adjustment of weights is illustrated in Fig. 6. All the weights and biases of the fuzzy neural network are modified in this way.

4. Computer Simulations

In this section, we illustrate results of computer simulations on simple numerical examples. A fuzzy neural network is used with single input unit. 5 hidden

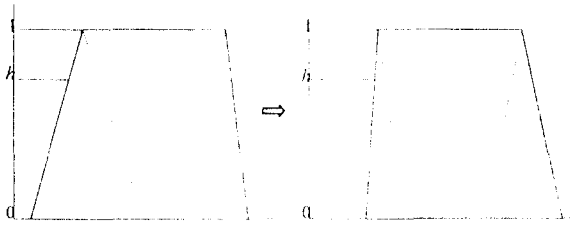

Fig 6. Modification of weights

units, single output unit. The specification of the neural network in the examples is shown in Table 1.

Table 1. Specifications of parameters

Learning constant η	0.5
Momentum constant α	0.9
Number of iterations	200
Initial value of weights	random
Value of h	0.1, 0.2, ... , 1.0

4.1 Example 1

Suppose that the next fuzzy if-then rules are given for the fuzzy neural network.

Fuzzy rules : If x is small then y is small.

If x is large then y is large.

The membership functions of "small" and "large" are shown in Fig. 7. The shape of membership functions in this example is triangle. The left triangular fuzzy number is "small". and the right one means "large" in this figure. Actual fuzzy outputs from the trained neural network are shown in Fig 8. We can see that the proposed neural network produces good performence with respect to the training data.

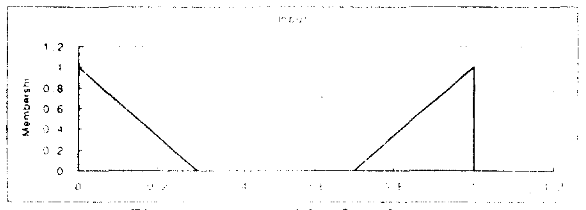

Fig 7. Membership functions

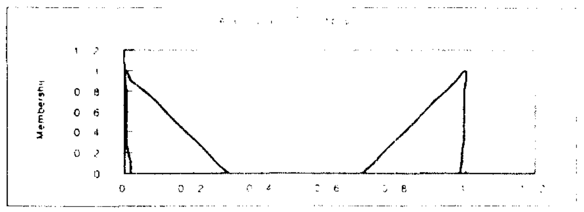

Fig 8. Fuzzy outputs of "small" and "large"

4.2 Example 2

Let us consider next fuzzy if-then rules.

Fuzzy rules :

If x is medium small then y is medium small.

If x is medium large then y is medium large.

Fig. 9 represents the membership functions of the fuzzy numbers, "medium small" and "medium large". We can see that the fuzzy numbers of this example are trapezoidal fuzzy numbers. The result is shown in Fig. 10.

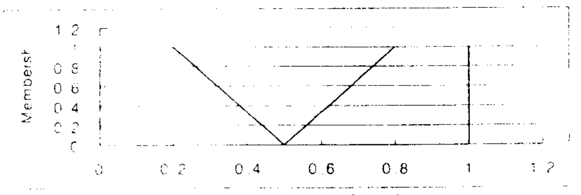
Fig 9. Membership functions

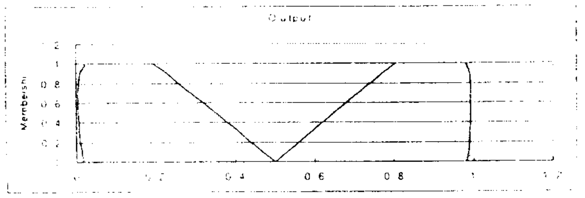

Fig 10. Actual fuzzy output and target output

4.3 Example 3

In this section, we take the fuzzy rules as follows.

Fuzzy rules :

If x is more or less small then y is medium small.

If x is more or less large then y is medium large.

The membership functions of each linguistic value are shown in Fig. 11 and 12. Fig. 11 represents "more or less small" and "more or less large", and Fig. 12 shows "medium small" and "medium large". The actual fuzzy output is shown in Fig. 13.

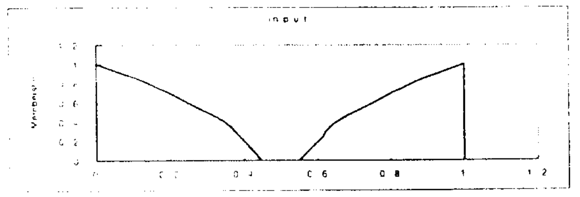

Fig 11. Input fuzzy numbers

The results indicate that the fuzziness of the fuzzy inputs is larger than that of the fuzzy targets. The actual result illustrates that the fitting ability is high.

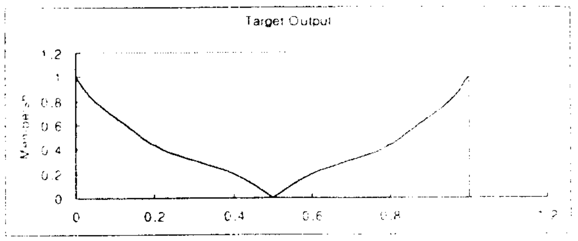

Fig 12. Target output

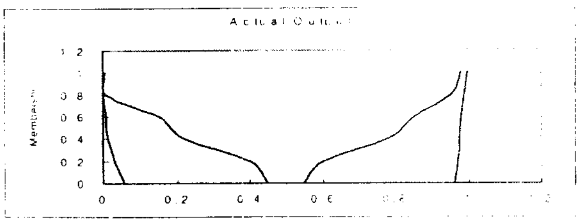

Fig 13. Actual output in Example 3

In order to examine the generalization ability of the trained neural network, we presented the fuzzy number "more or less medium" to the neural network. The membership fnction is shown in Fig. 14. and the output from the trained neural network is shown in Fig. 15.

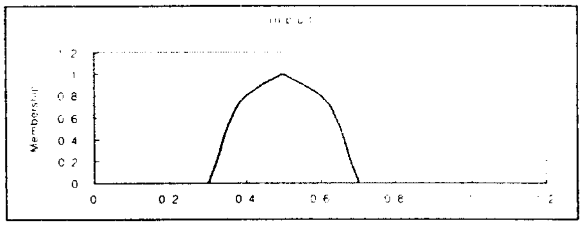

Fig. 14. Membership function

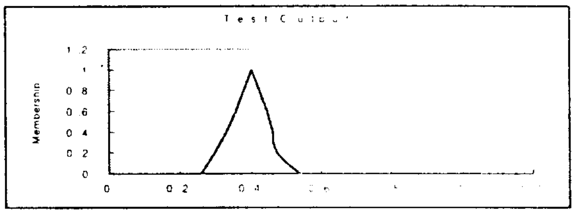

Fig. 15 Actual test output

4.4 Example 4

In this example, we show that the fuzzy neural network can handle real number inputs. A real number x can be considered as a fuzzy number as follows.

$$\mu_x(y)=\begin{cases}1 & if\ \ y=x\\ 0 & if\ \ y\neq x\end{cases}$$

Let us consider a single input and single output fuzzy mapping. We suppose that the input of the fuzzy mapping is real number while the output is a fuzzy number.

Fuzzy rules : If x is 0 then y is medium small.

If x is 1 then y is medium large.

The membership functions of "medium small" and "medium large" are show in Fig. 12. Fig. 16 represents the actual output for the example. It shows that the fuzzy neural network can treat real numbers as a special case of fuzzy numbers.

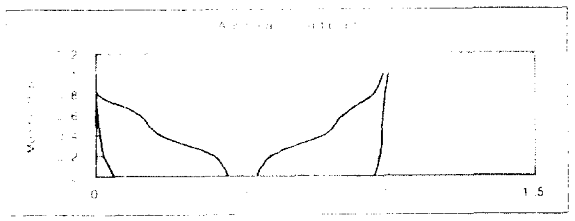

Fig 14. Actual fuzzy outputs

5. Concluding Remarks

In this paper, we have proposed a learning algorithm for three-layer feedforward fuzzy neural networks with trapezoidal fuzzy weights which are represented by combining two triangular fuzzy numbers. The learning algorithms derived can be viewed as an extention of standard back-propagation algorithm to the case of fuzzy input vectors and fuzzy target vectors. The neural network has worked well for several examples: It has high fitting ability for various fuzzy targets.

References

1. G. Alefeld and J. Herzberger, *Introduction to Interval Computations*. New York: Academic Press, 1983.
2. H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural networks that learn from fuzzy if-then rules," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 2, pp. 85-97, May 1993.
3. H. Ishibuchi, K. Kwon, and H. Tanaka, "A learning algorithm of fuzzy neural networks with triangular fuzzy weights," *Fuzzy Sets and Systems*, vol. 71, pp. 277-293, 1995.
4. H. Ishibuchi, K. Morioka, and I. B. Turksen, " Learning by fuzzified neural networks," *International Journal of Approximate Reasoning*, vol. 13, pp. 327-358, 1995
5. J. M. Keller and H. Tahani, "Back propagation neural networks for fuzzy logic," *Inf. Sci.*, Vol. 62, pp. 205-221, 1992.
6. D. E. Rumelhart, J. L. McClelland, and the PDP Reserch Group, *Parallel Distributed Processing*, vol. 1, Cambridge, MA: MIT Press, 1986.
7. K. Uehara and M. Fujise, "Learning of fuzzy-inference criteria with artificial neural network," *Proc. Int. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan, Vol. 1, pp. 193-198, 1990.