A Learning Algorithm of Fuzzy Neural Networks Using a Shape Preserving Operation

Jun Jae Lee, Dug Hun Hong, and Seok Yoon Hwang

Abstract

We derive a back-propagation learning algorithm of fuzzy neural networks using fuzzy operations, which preserves the shapes of fuzzy numbers, in order to utilize fuzzy if-then rules as well as numerical data in the learning of neural networks for classification problems and for fuzzy control problems. By introducing the shape preseving fuzzy operation into a neural network, the proposed network simplifies fuzzy arithmetic operations of fuzzy numbers with exact results in learning the network. And we illustrate our approach by computer simulations on numerical examples.

I. Introduction

Most of the supervised learning methods of neural networks, for example the perceptron, the BP(back propagation) algorithm, and the RCE(reduced Coulomb energy) network, utilize only numerical data. On other hand, fuzzy control is one of the most useful approaches for utilizing expert knowledge. Many hybrid approaches of fuzzy control systems and neural networks proposed for utilizing numerical data. Recently, learning methods of neural networks have been proposed in order to utilize not only numerical data but also expert knowledge represented by fuzzy if-then rules[1-9]. The main contribution of those papers is to propose an idea for integrating human knowledge and numerical data into a single information processing system(fuzzy control system or classification system). In the learning of neural networks for constructing classification systems, fuzzy if-then rules such as "If x_{p1} is large and x_{p2} is small, then $x_p=(x_{p1},x_{p2})$ belongs to class 1" are utilized as well as numerical data that are usually employed in conventional supervised learning methods, On other hand, for constructing fuzzy control systems, fuzzy if-then rules such as "If x_{p1} is small and x_{p2} is large, then y is small" are utilized in the learning of neural networks as well as numerical data. In order to deal with linguistic values such as "large," "small," and "medium," an architecture of neural networks that can handle fuzzy input and output vectors is needed.

Multilayer feedforward neural networks can be fuzzified by replacing real inputs and real targets with fuzzy numbers[1,2]. Several approaches have been proposed for learning by fuzzified neural networks[3], Hayashi et al.[1] proposed a fuzzy back-propagation algorithm, which can be viewed as direct fuzzzfication of the standard back-propagation algorithm[4]. The algorithm was obtained by replacing real numbers used for inputs outputs, targets, and weights in the standard back-propagation algorithm with fuzzy numbers. It was reported in [3] that the algorithm converged to the wrong weights. Hayashi et al. [1] also discussed a back-propagation algorithm for individual α-cuts of fuzzy weights. Because the algorithm independently updates α-cuts of fuzzy weights, one is not sure that the updated fuzzy weight in fact forms a fuzzy set(see [1,3]). Ishibuchi et al. proposed an architecture of multilayer feedforward neural network for fuzzy input vectors[5], and that architecture was applied to the implementation of fuzzy if then rules in [6,7]. An α-cut based backpropagation algorithm for the supports of symmetric tringular fuzzy weights was derived in [8] and was also extended to the learning algorithm that could be applied to the learning of fuzzy weights of various shapes such as nonsymmetric triangular and trapezoidal types[9]. The learning algorithm derived in [9] is a generalization of the former work that was applicable only to symmetric triangular fuzzy weights or nonsymmetric trapezoidal fuzzy weights. The input-output relation of each unit was defined by the extension, principle of Zadeh[10]. Outputs from the fuzzy neural networks are numerically calculated by interval

Manuscript received March 24, 1997; accepted January 30, 1998.
Jun Jae Lee is with the Dept. of Computer Engineering, Dongseo University.
Duk Hun Hong is with the School of Automotive Engineering, Taegu Hyosung Catholic University.
Seok Yoon Hwang is with the Dept. of Mathematics, Taegu University.

arithmetic[11] for level sets(i.e., α-cuts) of fuzzy weigths and fuzzy inputs. The above operations on fuzzy numbers are performed numerically on level sets(i. e., α-cuts). The h -level set of a fuzzy number A is defined as $[A]_h = \{x \in R \mid \mu_A(x) \geq h\}$ for $0 < h \leq 1$, where R is the set of all real numbers. Because level sets of fuzzy numbers are closed intervals, the above operations of fuzzy numbers can be computed for h-level sets from interval arithmetic[11]. Thus for each h, h-level sets of each input-output pair are used in the learning by the fuzzified neural network. But for each h, the learning time of the algorithm for fuzzy data is about twice as long as that of the BP algorithm for numerical data. For obtaining more accurate results, it also requires a lot of levels. Therefore, it is complicated and much time-consuming than the neural network for numerical data. In practical computation, it is natural to require the preserving the shape of fuzzy numbers during the multiplication. Unfortunately, there are no results about multiplication.

In this paper, we propose the simple computing method, which is based on the shape preserving operations of fuzzy numbers, without using h-levels. By introducing T_u-based fuzzy arithmetic operation inducing a shape preserving operation of $L-R$ fuzzy numbers, we derive a learning algorithm for fuzzy neural networks. T_u is known as a shape preserving operation under addition and subtraction. Likewise, it is also proved in our paper to satisfy a shape preserving under multiplication(see the proof in Appendix). Therefore, it should be noted that this is simple and fast, since it can be peformed only on a center point and two spreads without doing fuzzy operations on many level sets. This method also does fuzzy arithmetic operations of fuzzy numbers with exact results. Another important feature is that it provides means of controlling the growth of uncertainty during calculations. Namely, shape preserving arithmetic operations of $L-R$ fuzzy numbers allow to controll the resulting spread. Our fuzzified neural networks are three-layer feedforward networks with multiple inputs and multiple outputs. And we define a cost function that measures the difference between a fuzzy target vector and an actual fuzzy output vector. Then, a crisp learning algorithms is derived from the cost function for adjusting center, and spreads of each fuzzy number of $L-R$ type. Finally we illustrate our approach by computer simulations on numerical examples.

II. Fuzzified neural network

1. *Fuzzified neural networks architecture*

The inputs, weights, and biases of the standard feedforward neural network can be extended to fuzzy numbers. The fuzzification of neural networks means this extension[9]. Therefore the fuzzification does not change the neural network architecture. That is, the fuzzified neural network has the same network architecture as the standard neural network.

Let us denote fuzzy numbers and real numbers by uppercase letters(e.g., A,B,..) and lowercase letters(e.g., a,b,...), respectively. Then the input-output relation of the fuzzified neural network can be written for a fuzzy input vector $X_p = (X_{p1}, X_{p2}, \ldots, X_{pn_I})$ as follows:

Input units :

$$O_{pi} = X_{pi},\ i = 1, 2, \cdots, n_I. \tag{1}$$

Hidden units :

$$O_{pj} = f(Net_{pj}),\quad Net_{pj} = \sum_{i=1}^{n_I} W_{ji} \otimes O_{pi} \oplus \Theta_j,\quad j = 1, 2, \cdots, n_H. \tag{2}$$

Output units :

$$O_{pk} = f(Net_{pk}),\quad Net_{pk} = \sum_{i=1}^{n_H} W_{kj} \otimes O_{pj} \oplus \Theta_k,\quad k = 1, 2, \cdots, n_O. \tag{3}$$

where X_{pi} is a fuzzy input, W_{ji} and W_{kj} are fuzzy weights, Θ_j and Θ_k are biases, $\otimes$ and $\oplus$ are fuzzy multiplication and addition operators, respectively. The architecture of the fuzzified neural network is shown in Fig. 1.

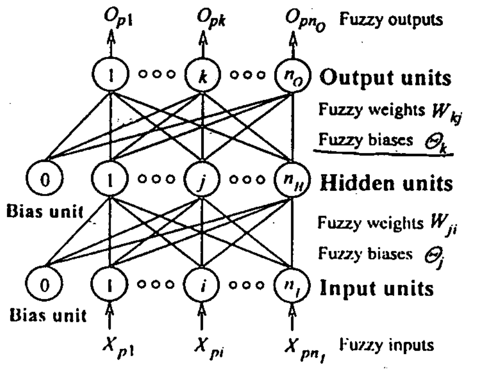


Fig. 1. Architecture of a three-layer feedforward fuzzified neural network

2. *Calculation of the input-output relation under a shape preserving operation*

Before describing the input-output relation in our fuzzified neural network, we briefly mention the previous approach[8,9]. The input-output relation (1)-(3) is defined by the extension principle of Zadeh[10]. This means that fuzzy arithmetic is employed for calculating the input-output relation of the fuzzified neural network. In [9], the following addition, multiplication, and nonlinear mapping of fuzzy members were used for defining fuzzified neural network:

$$\mu_{A+B}(z) = \max\{\mu_A(x) \wedge \mu_B(y) \mid z = x+y\}, \tag{4}$$

$$\mu_{AB}(z) = \max\{\mu_A(x) \wedge \mu_B(y) \mid z = xy\}, \tag{5}$$

$$\mu_{f(Net)}(z) = \max\{\mu_{Net}(x) \mid z = f(x)\}, \tag{6}$$

where A, B, Net are fuzzy numbers, $\mu_*(\cdot)$ denotes the membership function of each fuzzy number, and $\wedge$ is the minimum operator. These operations are illustrated in Fig. 2. The shape of fuzzy numbers is preserved under the addition, i.e., in Fig. 2, the shape of (c) after operation is identical to that of (b) and (c), but under multiplication it does not. Therefore, it is complicated and time-consuming since the operation of fuzzy numbers must be numerically performed on level sets(i.e., α-cuts). For obtaining more accurate results, it requires a lot of levels.

In this paper, to preseve the shape of fuzzy number, T_w -based addition and multiplication of fuzzy numbers of $L-R$ type are introduced. Since the shapes of most membership functions is of type $L-R$, we are concerned about fuzzy numbers of type $L-R$. A fuzzy number is a convex subset of the real line R with a normalized membership function. A triangular fuzzy number $\tilde{a}$ denoted by (a, α, β) is defined as

$$\tilde{a}(t) = \begin{cases} 1 - \dfrac{|a-t|}{\alpha} & \text{if } a-\alpha \le t \le a, \\ 1 - \dfrac{|a-t|}{\beta} & \text{if } a \le t \le a+\beta, \\ 0 & \textit{otherwise}, \end{cases}$$

where $a \in R$ is the center, $\alpha > 0$ is the left spread, and $\beta > 0$ is the right spread of $\tilde{a}$. If $\alpha = \beta$, then the triangular fuzzy number is called a symmetric triangular fuzzy number and denoted by (a, α). A fuzzy number $\tilde{a} = (a, \alpha, \beta)_{LR}$ of type $L-R$ is a function the reals into the interval [0,1] satisfying

$$\tilde{a}(t) = \begin{cases} R\left(\dfrac{t-a}{\beta}\right) & \text{for } a \le t \le a+\beta, \\ L\left(\dfrac{a-t}{\alpha}\right) & \text{for } a-\alpha \le t \le a, \\ 0 & \textit{else}, \end{cases}$$

where L and R are non-increasing and continuous functions from [0,1] to [0,1] satisfying $L(0) = R(0) = 1$ and $L(1) = R(1) = 0$. A binary operation T on the unit interval is said to be a triangular norm [12] (t-norm for short) iff T is associative, commutative, non-decreasing and $T(x, 1) = x$ for each $x \in [0,1]$. Moreover, every t-norm satisfies the inequality,

$$T_w(a, b) \le T(a, b) \le \min(a, b) = T_M(a, b)$$

where,

$$T_w(a, b) = \begin{cases} a & \text{if } b = 1, \\ b & \text{if } a = 1, \\ 0 & \textit{otherwise}. \end{cases}$$

The usual arithmetical operation of reals can be extended to the arithmetical operations on fuzzy numbers by means of Zadeh's extension principle[10] based on a triangular norm T. Let $\tilde{A}$, $\tilde{B}$ be fuzzy numbers of the real line R. The fuzzy number arithmetic operations are summarized as follows :
Fuzzy number addition $\oplus$:

$$(\tilde{A} \oplus \tilde{B})(z) = sup_{x+y=z}\, T(\tilde{A}(x), \tilde{B}(y))$$

Fuzzy number multiplication $\otimes$:

$$(\tilde{A} \otimes \tilde{B})(z) = sup_{x \cdot y=z}\, T(\tilde{A}(x), \tilde{B}(y))$$

The addition(subtraction) rule for $L-R$ fuzzy numbers is well known in the case of T_M - based addition and then the resulting sum is again on $L-R$ fuzzy numbers, i. e., the shape is preserved. It is also known that T_w - based addition preseves the shape of $L-R$ fuzzy numbers [13,14]. In practical computation, it is natural to require the shape preservation of fuzzy numbers during the multiplication. Unfortunately, there are no results about multiplication. Of course, we know that T_M - based multiplication does not preserve the shape of $L-R$ fuzzy numbers. In this section we show that, for a given shapes L and R, T_w induces a shape preserving multiplication of $L-R$ fuzzy numbers and we use T_w - based arithmetic operations to study fuzzy neural networks. This method simplifies fuzzy arithmetic operations of fuzzy numbers with exact results.

Another important feature is that it provides a mean of controlling the growth of uncertainty during calculations. Namely, shape preserving arithmetic operations of $L-R$ fuzzy numbers allow to controll the resulting spread. Now, let $T = T_w$ be the weakest t-norm and, let $\tilde{A} = (a, \alpha_A, \beta_A)_{LR}$, $\tilde{B} = (b, \alpha_B, \beta_B)_{LR}$ be two $L-R$ fuzzy numbers. By [13,14],

$$\begin{aligned} \tilde{A} \oplus \tilde{B} &= (a, \alpha_A, \beta_A)_{LR} \oplus (b, \alpha_B, \beta_B)_{LR} \\ &= (a+b, \max(\alpha_A, \alpha_B), \max(\beta_A, \beta_B))_{LR.} \end{aligned} \tag{7}$$

If $L = R$, we can easily show that

$$\begin{aligned} \tilde{A} \ominus \tilde{B} &= (a, \alpha_A, \beta_A)_{LR} \ominus (b, \alpha_B, \beta_B)_{LR} \\ &= (a, \alpha_A, \beta_A)_{LR} \oplus (-b, \beta_B, \alpha_B)_{LR} \\ &= (a-b, \max(\alpha_A, \beta_B), \max(\alpha_B, \beta_A))_{LR} \end{aligned} \tag{8}$$

From these two equations, we know that T_w is a shape preserving operation under addition and subtraction. Likewise, it is also shape preserving under multiplication(See the proof in Appendix). Therefore, it should be noted that this is simple and fast, since it can be pefomed only on center point and spreads without doing fuzzy operations on level sets.

Let $f(x) = \frac{1}{1+e^{-x}}$ be the activation function of hidden units and output units of our fuzzy neural network. For a fuzzy number $\tilde{A} = (a, \alpha_A, \beta_A)_{LR}$, we define a fuzzy number $f(\tilde{A})$ of $L-R$ type by

$$f(\tilde{A}) = (f(a), f(a) - f(a-\alpha_A), f(a+\beta_A) - f(a))_{LR}.$$

To simplify the computation, we assume in the paper that the fuzzy inputs, fuzzy weights, and fuzzy biases are fuzzy numbers of $L-R$ type with $L=R$, and the center of fuzzy inputs are nonnegative. These operations are illustrated in Fig. 3.

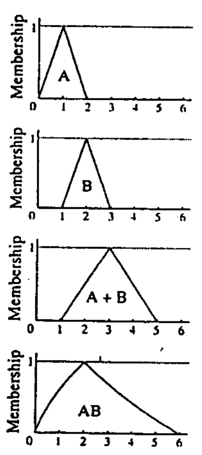


Fig. 2. Fuzzy number operation : addition and multiplication in [6,7]. The shape of fuzzy numbers is not preserved during the multiplication.

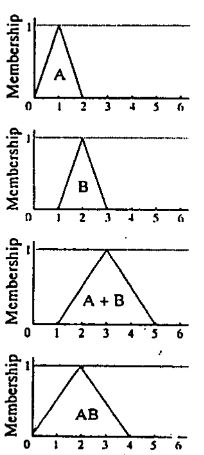


Fig. 3. T_w-based fuzzy number operation: addition and multiplication. The shape of fuzzy numbers is preserved.

Let us fuzzify a three-layer feedforward neural network with n_I input units, n_H hidden units and n_O output units. And let input vectors, target vectors, connection weights and biases be fuzzy numbers of $L-R$ type. Then the input-output relation of each unit of the fuzzified neural network can be written as follows:

By using the operations in (7), (A.1) and (A.2), the above relations are rewritten as follows :

Input units :

$$O_{pi} = (o_{pi}, o^L_{pi}, o^R_{pi})_{LR}$$

Hidden units :

$$O_{pj} = (o_{pj}, o^L_{pj}, o^R_{pj})_{LR} = f(Net_{pj})$$

$$\begin{aligned} Net_{pj} &= (W_{j1} \otimes O_{p1}) \oplus \cdots \oplus (W_{jn_I} \otimes O_{pn_I}) \oplus \Theta_j \\ &= \Big(\sum_{i=1}^{n_I} w_{ji} o_{pi} + \theta_j, \max_i (\max_{w_{ji} \geq 0} (w^L_{ji} o_{pi}, o^L_{pi} w_{ji}), \max_{w_{ji} < 0} (w^L_{ji} o_{pi}, o^R_{pi} |w_{ji}|), \\ &\quad \theta^L_j), \max_i (\max_{w_{ji} \geq 0} (w^R_{ji} o_{pi}, o^R_{pi} w_{ji}), \max_{w_{ji} < 0} (w^R_{ji} o_{pi}, o^L_{pi} |w_{ji}|), \theta^R_j) \Big)_{LR} \\ &= (net_{pj}, net^L_{pj}, net^R_{pj})_{LR}. \end{aligned} \quad \textbf{(9)}$$

where $W_{ji} = (w_{ji}, w^L_{ji}, w^R_{ji})_{LR}$.

Output units:

$$O_{pk} = (o_{pk}, o^L_{pk}, o^R_{pk})_{LR} = f(Net_{pk})$$

$$\begin{aligned} Net_{pk} &= (W_{k1} \otimes O_{p1}) \oplus \cdots \oplus (W_{kn_H} \otimes O_{pn_H}) \oplus \Theta_k \\ &= \Big(\sum_{j=1}^{n_H} w_{kj} o_{pj} + \theta_k, \max_j (\max_{w_{kj} \geq 0} (w^L_{kj} o_{pj}, o^L_{pj} w_{kj}), \max_{w_{kj} < 0} (w^L_{kj} o_{pj}, o^R_{pj} \\ &\quad |w_{kj}|), \theta^L_k), \max_j (\max_{w_{kj} \geq 0} (w^R_{kj} o_{pj}, o^R_{pj} w_{kj}), \max_{w_{kj} < 0} (w^R_{kj} o_{pj}, o^L_{pj} |w_{kj}|), \theta^R_k) \Big)_{LR} \\ &= (net_{pk}, net^L_{pk}, net^R_{pk})_{LR} \end{aligned} \quad \textbf{(10)}$$

Thus, one unit in the conventional neural network consists of three units of center, left spread and right spread neurons.

3. Learning algorithm

To derive a learning algorithm of our fuzzy neural network, we define a cost function for the fuzzy output O_{pk} from the k-th output unit and the corresponding fuzzy target

$$T_{pk} = (t_{pk}, t^L_{pk}, t^R_{pk})_{LR}$$

as follows :

$$\begin{aligned} e_p &= \sum_{k=1}^{n_O} e_{pk}, \\ e_{pk} &= \frac{1}{2} \{ (t_{pk} - o_{pk})^2 + (t^L_{pk} - o^L_{pk})^2 + (t^R_{pk} - o^R_{pk})^2 \}. \end{aligned} \quad \textbf{(11)}$$

By using the cost function e_p, the fuzzy weights W_{kj} of the fuzzified neural network are adjusted as follows:

$$\begin{aligned} w_{kj}(t+1) &= w_{kj}(t) + \triangle w_{kj}(t), \\ w^L_{kj}(t+1) &= w^L_{kj}(t) + \triangle w^L_{kj}(t), \\ w^R_{kj}(t+1) &= w^R_{kj}(t) + \triangle w^R_{kj}(t), \end{aligned}$$

$$\triangle w_{kj}(t) = -\eta \frac{\partial e_p}{\partial w_{kj}} + \alpha \cdot \triangle w_{kj}(t-1), \quad \textbf{(12)}$$

$$\triangle w^L_{kj}(t) = -\eta \frac{\partial e_p}{\partial w^L_{kj}} + \alpha \cdot \triangle w^L_{kj}(t-1), \quad \textbf{(13)}$$

$$\triangle w^R_{kj}(t) = -\eta \frac{\partial e_p}{\partial w^R_{kj}} + \alpha \cdot \triangle w^R_{kj}(t-1), \quad \textbf{(14)}$$

where η is a learning constant, α is a momentum constant and t indexes the number of adjustments. The explicit calculation of each derivative in (12), (13) and (14) is shown in the Appendix. The fuzzy weights W_{ji} and the fuzzy biases Θ_k, Θ_j are changed in the same manner as the fuzzy weights W_{kj}. (see the Appendix.). The learning algorithm in our fuzzifified neural netwok is the same as the standard back-propagation algorithm execept updating three fuzzy number values(center, left spread, right spread) instead of one real value. On the other hand, the method in [8,9] is calculated on all h-levels. Therefore, it is complicated and time-consuming.

III. Simulation Results

In this subsection, we illustrate the derived learning algorithm by two simple numerical examples. In these two examples, we used the following specifications of the learning algorithm.

(1) Number of hidden units:six units

(2) Stopping condition: 100,000 iterations of the learning algorithm.

(3) Learning constant: η = 0.5

(4) Momentum constant: α = 0.9

(5) Initial values of the fuzzy weights and the fuzzy biases: random real numbers in the closed interval[0,1].

1. Example 1

In this example, we apply the proposed method to the approximate realization of a nonlinear mapping of fuzzy numbers. Let us assume that both the input space and the output space of this mapping are the unit interval [0,1]. Therefore, we can depict each fuzzy input-output pair (X_p, T_p) in the input output space as shown in Fig. 4. The rectangle in Fig. 4 shows the cartesian product of supports of the fuzzy input X_p and the fuzzy target T_p . Let us assume that three fuzzy input-output pairs in Fig. 5 are given as training data. A fuzzy neural network with a single input unit, six hidden units and a single output unit was trained by the proposed learning algorithm. Fuzzy outputs from the trained fuzzy neural network are shown in Fig. 6 for the three fuzzy inputs used in the learning and two new fuzzy inputs. Calculation for center(dot), left and right spread(boundary) point is perforemed to be depicted while that for all h-level sets is in[8,9]. From the comparison between Fig. 5 and 6, we can observe the good generalization for the new fuzzy inputs as well as the good fitting to the fuzzy training data. The number of iteration and epoch are 39597 and 10754, respectively.

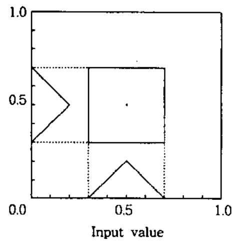


Fig. 4. Illustration of the input-output pair (X_p, T_p) in the input-output space

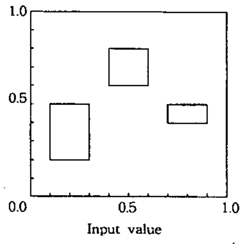


Fig. 5. Fuzzy training data in Example 1.

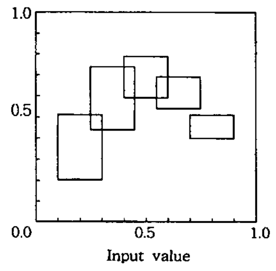


Fig. 6. Fuzzy outputs from the trained fuzzy neural network in Example 1.

2. Example 2

Since real numbers can be viewed as a special case of fuzzy numbers, the proposed fuzzy neural network can handle real inputs as well as fuzzy inputs. In this example, we apply the proposed method to the approximate realization of a nonlinear fuzzy function that maps a real number to a fuzzy number. Training data in this example consist of pairs of real inputs x_p's and fuzzy targets T_p's. Each pair is depicted in the input-output space as shown in Fig. 7. The triangle in Fig. 7 shows the membership function of the triangular fuzzy target T_p. Let us assume that six pairs of real inputs and fuzzy outputs in Fig. 8 are given as training data. Using these training data, we trained a fuzzy neural network with a single input unit, six hidden units and a single output unit by the proposed learning algorithm. The real input x_p is treated as a fuzzy number with the following membership function:

$$\mu_{x_p}(x)=\begin{cases} 1 & \text{if } x=x_p, \\ 0 & \text{if } x\neq x_p. \end{cases}$$

Fuzzy outputs from the trained fuzzy neural network are shown in Fig. 9 for 11 real inputs: x = 0.0, 0.1, ... , 1.0. From the comparison between Fig. 8 and Fig. 9, we can observe the good generalization for the new real inputs as well as the good fitting to the fuzzy training data. The number of iteration and epoch are 89904 and 14983, respectively.

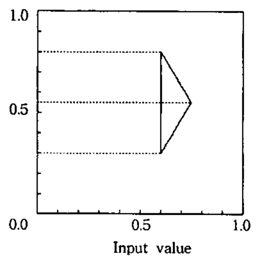


Fig. 7. Illustration of the input-output pair

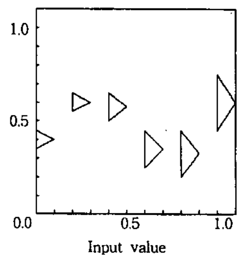


Fig. 8. Fuzzy training data in Example 2.

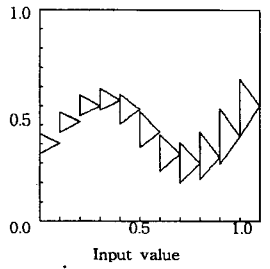


Fig. 9. Fuzzy outputs from the trained fuzzy neural network in Example 2.

3. Example 3

In this example, we apply the proposed method to the approximate realization of fuzzy if-then rules by a fuzzy neural network. Let us assume that the following three if-then rules are given:

if x is small then y is small.
if x is medium then y is medium.
if x is large then y is large

The membership function of linguistic values such as "small", "medium" and "large" are given Fig. 10. From the above fuzzy if-then rules, we can obtain the following training data:

$\{ \{(X_p, T_p)\} = \{(small, small), (medium, medium), (large, large)\}$

It should be noted that nonsymmetric fuzzy targets are include in these training data. They are depicted in Fig. 11 in the same manner as in Fig 4 and 5.

Using the training data in Fig. 11, we trained a fuzzy neural network with a single input unit, six hidden units and a single output unit. Fuzzy outputs from the trained fuzzy neural network are shown in Fig. 12 for new fuzzy inputs "medium small" and "medium large". From the comparison between Figs. 11 and 12, we can observe the good generalization to the new fuzzy inputs. Therefore, we can obtain the following two fuzzy if-then rulses:

If x is medium small then y is medium small.
If x is medium large then y is medium large.

It should be noted that these two fuzzy if-then rules coincide with our intuitive interpolation of the given three fuzzy if-then rules. Therefore, we can conclude that the trained fuzzy neural network found valid fuzzy if-then rules in this example.

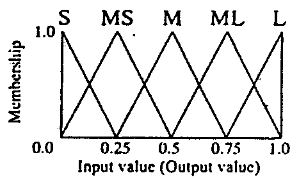


Fig. 10. Membership function of five linguistic values (S:small ; MS : medium small; M:medium ; ML : medium large; L:large).

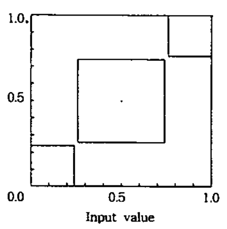


Fig. 11. Fuzzy training data in Example 3.

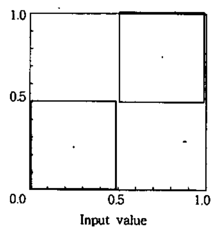


Fig. 12. Fuzzy outputs from the trained fuzzy neural network for new fuzzy inputs in Example 3

IV. Conclusion

In this paper, we derived a learning algorithm of fuzzy neural networks with fuzzy numbers of $L-R$ type by using T_u-based operation. If the inputs and targets are of $L-R$ type, application of the proposed method is simple and also derives the good approximation. The effectiveness of the derived learning algorithm was demonstrated by computer simulations. While the derived learning algorithm for fuzzy numbers of $L-R$ type worked in this paper, the extensions to the case of more general fuzzy numbers are left for future studies.

Acknowledgement

This research was supported by KOSEF Grant No. 951-0101-004-2.

References

[1] Y. Hayashi, J. J. Buckley and E. Cazogala, “Fuzzy neural network with fuzzy signals and weights,” *Internat. J. Intelligent Systems* vol. 8, pp. 527-537, 1993.

[2] H. Ishibuchi, H. Okada and H. Tanaka, “Fuzzy neural networks with fuzzy weights and fuzzy biases,” *Proc. ICNN '93* (San Francisco, 28 March-1), pp. 1650-1655, April 1993.

[3] J. J. Buckley and Y. Hayashi, “Fuzzy neural networks: a survey,” *Fuzzy Sets and Systems* vol. 66, pp. 1-13, 1991.

[4] D. E. Rumelhart, J. L. McClelland and the PDP Reserch Group, *Parallel Distributed Processing*, vol. 1, MIT Press, Cambridge, MA, 1986.

[5] H. Ishibuchi, R. Fujioka and H. Tanaka, “An architecture of neural network for input vectors of fuzzy numbers,” *Proc FUZZY-IEEE '92* San Diego, 8-12, pp. 643-650 march 1992.

[6] H. Ishibuchi, R. Fujioka, and H. Tanaka, “Neural networks that learn from fuzzy if-then rules,” *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 2, pp. 85-97, May 1993.

[7] H. Ishibuchi, H. Okada and H. Tanaka, “Interpolation of fuzzy if-then rules by neural networks,” *Internat. J. Appox. Reason.*, vol. 10(1), pp. 3-27, 1994.

[8] 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.

[9] H. Ishibuchi, K. Morioka and I. B. Turksen, "Learning by fuzzified neural networks," *Inter. J. Approx. Reason.* vol. 13, pp. 327-358, 1995.
[10] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning : Parts 1-3," *Inform. Sci.* vol. 8, pp. 199-249, 301-357 : 9, 43-80, 1975.
[11] A. Kaufmann and M. M. Gupta, *Introduction to fuzzy arithmetic*, Van Nostrand Reinhold, 1985.
[12] B. Schweizer and A. Sklar, "Associative functions and abstract semigroups," *Publ. Math. Debrecen,* vol. 10, pp. 69-81, 1963.
[13] A. Kolesarova, "Triangular norm-based addition of linear fuzzy numbers," *Tatra Mountains Math. Publ.* pp. 675-81, 1995.
[14] R. Mesiar, "Shape preserving additions of fuzzy intervals" (to appear in *Fuzzy Sets and Systems*).

Appendix

A.1 T_{w}-based Multiplication

T_w - based multiplication is as follows:

Case I) *Let* $a, b > 0$. *Then* for $z \le ab$ it is

$$\begin{aligned}(\tilde{A}\otimes\tilde{B})(z) &= \sup_{x\cdot y=z} T_w(\tilde{A}(x), \tilde{B}(y))\\ &= \max\left(A\left(\frac{z}{b}\right), B\left(\frac{z}{a}\right)\right)\\ &= \max\left(L\left(\frac{a-z/b}{\alpha_A}\right), L\left(\frac{b-z/a}{\alpha_B}\right)\right)\\ &= \max\left(L\left(\frac{ab-z}{\alpha_A b}\right), L\left(\frac{ab-z}{\alpha_B a}\right)\right)\\ &= L[(ab-z)/\max(\alpha_A b, \alpha_B a)]\end{aligned}$$

for $z \ge ab - \max(\alpha_A b, \alpha_B a)$, it is 0 otherwise.

Similarly, for z > ab we get

$(\tilde{A}\otimes\tilde{B})(z) = R((z-ab)/\max(\beta_A b, \beta_B a))$

for $z \le ab - \max(\beta_A b, \beta_B a)$ and it is zero otherwise.

It follows

$$\tilde{A}\otimes\tilde{B} = (ab, \max(\alpha_A b, \alpha_B a), \max(\beta_A b, \beta_B a))_{LR.} \tag{A-1}$$

In a similarly manner, we have the following cases;

Case II) For a < 0, b < 0 it is

$\tilde{A}\otimes\tilde{B} = (ab, \max(-\beta_A b, -\beta_B a), \max(-\alpha_A b, -\alpha_B a))_{LR}$

Case III) For a = 0, b ≠ 0 it is

$\tilde{A}\otimes\tilde{B} = (0, \alpha_A b, \beta_A b)_{LR}$

Case IV) For a ≠ 0, b = 0 it is

$\tilde{A}\otimes\tilde{B} = (0, \alpha_B a, \beta_B a)_{LR.}$

Case V) For a = 0, b = 0 it is

$\tilde{A}\otimes\tilde{B} = (0, 0, 0)_{LR} = I_{\{0\}}(x)$

where,

$$I_{\{0\}}(x) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \textit{otherwise.} \end{cases}$$

Case VI) For a < 0, b > 0, we assume $L = R$, additionally, then it is

$$\tilde{A}\otimes\tilde{B} = (ab, \max(\alpha_A b, -\beta_B a), \max(\beta_A b, -\alpha_B a))_{RR} \tag{A-2}$$

A.2. Calculation of Derivatives

1) Derivatives in output layer : $\frac{\partial e_p}{\partial w_{kj}}, \frac{\partial e_p}{\partial w_{kj}^L}, \frac{\partial e_p}{\partial w_{kj}^R}$

$$\begin{aligned}\frac{\partial e_p}{\partial w_{kj}} &= \frac{\partial e_p}{\partial\, net_{pk}}\frac{\partial\, net_{pk}}{\partial w_{kj}} + \frac{\partial e_p}{\partial\, net_{pk^L}}\frac{\partial\, net_{pk^L}}{\partial w_{kj}} + \frac{\partial e_p}{\partial\, net_{pk^R}}\frac{\partial\, net_{pk^R}}{\partial w_{kj}}\\ \frac{\partial e_p}{\partial w_{kj}^L} &= \frac{\partial e_p}{\partial\, net_{pk}}\frac{\partial\, net_{pk}}{\partial w_{kj}^L} + \frac{\partial e_p}{\partial\, net_{pk^L}}\frac{\partial\, net_{pk^L}}{\partial w_{kj}^L} + \frac{\partial e_p}{\partial\, net_{pk^R}}\frac{\partial\, net_{pk^R}}{\partial w_{kj}^L}\\ \frac{\partial e_p}{\partial w_{kj}^R} &= \frac{\partial e_p}{\partial\, net_{pk}}\frac{\partial\, net_{pk}}{\partial w_{kj}^R} + \frac{\partial e_p}{\partial\, net_{pk^L}}\frac{\partial\, net_{pk^L}}{\partial w_{kj}^R} + \frac{\partial e_p}{\partial\, net_{pk^R}}\frac{\partial\, net_{pk^R}}{\partial w_{kj}^R}\end{aligned} \tag{A-3}$$

And

$$\begin{aligned}\frac{\partial e_p}{\partial net_{pk}} &= (o_{pk} - t_{pk})o_{pk}(1 - o_{pk}),\\ \frac{\partial e_p}{\partial net_{pk}^L} &= (o_{pk}^L - t_{pk}^L)f(net_{pk} - net_{pk}^L)(1 - f(net_{pk} - net_{pk}^L))\\ \frac{\partial e_p}{\partial net_{pk}^R} &= (o_{pk}^R - t_{pk}^R)f(net_{pk} + net_{pk}^R)(1 - f(net_{pk} + net_{pk}^R))\end{aligned} \tag{A-4}$$

From net_{pk} in (10), we obtain

$$\frac{\partial net_{pk}}{\partial w_{kj}} = o_{pj}, \quad \frac{\partial\, net_{pk}}{\partial w_{kj}^L} = 0 = \frac{\partial\, net_{pk}}{\partial w_{kj}^R} \tag{A-5}$$

From net_{pk}^L in (10), we obtain

$$\begin{aligned}\frac{\partial net_{pk}^L}{\partial w_{kj}} &= \begin{cases} o_{pj}^L & \text{if } w_{kj} \ge 0 \text{ and } o_{pj}^L \cdot w_{kj} = net_{pk}^L, \\ o_{pj}^R & \text{if } w_{kj} < 0 \text{ and } o_{pj}^R \cdot |w_{kj}| = o_{pk}^L, \\ 0 & \textit{otherwise,} \end{cases}\\ \frac{\partial net_{pk}^L}{\partial w_{kj}^L} &= \begin{cases} o_{pj} & \text{if } o_{pj} \cdot w_{kj}^L = net_{pk}^L, \\ 0 & \textit{otherwise,} \end{cases}\\ \frac{\partial net_{pk}^L}{\partial W_{kj}^R} &= 0.\end{aligned} \tag{A-6}$$

And also from net_{pk}^R in (10), we obtain

$$\begin{aligned}\frac{\partial net_{pk}^R}{\partial w_{kj}} &= \begin{cases} o_{pj}^R & \text{if } w_{kj} \ge 0 \text{ and } o_{pj}^R \cdot w_{kj} = net_{pk}^R, \\ -o_{pj}^L & \text{if } w_{kj} < 0 \text{ and } o_{pj}^L \cdot |w_{kj}| = net_{pk}^R, \\ 0 & \textit{otherwise,} \end{cases}\\ \frac{\partial net_{pk}^R}{\partial w_{kj}^L} &= 0,\\ \frac{\partial net_{pk}^R}{\partial w_{kj}^R} &= \begin{cases} o_{pj} & \text{if } o_{pj} \cdot w_{kj}^R = net_{pk}^R, \\ 0 & \textit{otherwise.} \end{cases}\end{aligned} \tag{A-7}$$

By the above equations, we can calculate $\frac{\partial e_p}{\partial w_{kj}}, \frac{\partial e_p}{\partial w_{kj}^L}, \frac{\partial e_p}{\partial w_{kj}^R}$. Since the fuzzy bias Θ_k can be viewed as the fuzzy weight W_{kj} with O_{pn} = (1, 0, 0)LR, the dervatives $\frac{\partial e_p}{\partial \theta_k}, \frac{\partial e_p}{\partial \theta_k^L}, \frac{\partial e_p}{\partial \theta_k^R}$ for the learning of Θ_k can be obtained in the same manner as $\frac{\partial e_p}{\partial W_{kj}}, \frac{\partial e_p}{\partial W_{kj}^L}, \frac{\partial e_p}{\partial W_{kj}^R}$, respectively.

2) Derivatives in hidden layer $\frac{\partial e_p}{\partial w_{ji}}, \frac{\partial e_p}{\partial w_{ji}^L}, \frac{\partial e_p}{\partial w_{ji}^R}$:

$$\begin{aligned}\frac{\partial e_p}{\partial w_{ji}} &= \frac{\partial e_p}{\partial net_{pj}}\frac{\partial net_{pj}}{\partial w_{ji}} + \frac{\partial e_p}{\partial net^L_{pj}}\frac{\partial net^L_{pj}}{\partial w_{ji}} + \frac{\partial e_p}{\partial net^R_{pj}}\frac{\partial net^R_{pj}}{\partial w_{ji}},\\ \frac{\partial e_p}{\partial w^L_{ji}} &= \frac{\partial e_p}{\partial net_{pj}}\frac{\partial net_{pj}}{\partial w^L_{ji}} + \frac{\partial e_p}{\partial net^L_{pj}}\frac{\partial net^L_{pj}}{\partial w^L_{ji}} + \frac{\partial e_p}{\partial net^R_{pj}}\frac{\partial net^R_{pj}}{\partial w^L_{ji}},\\ \frac{\partial e_p}{\partial w^R_{ji}} &= \frac{\partial e_p}{\partial net_{pj}}\frac{\partial net_{pj}}{\partial w^R_{ji}} + \frac{\partial e_p}{\partial net^L_{pj}}\frac{\partial net^L_{pj}}{\partial w^R_{ji}} + \frac{\partial e_p}{\partial net^R_{pj}}\frac{\partial net^R_{pj}}{\partial w^R_{ji}}\end{aligned} \tag{A-8}$$

On the other hand,

$$\begin{aligned}\frac{\partial e_p}{\partial net_{pj}} &= \sum_{k=1}^{no}\left(\frac{\partial e_p}{\partial net_{pk}}\frac{\partial net_{pk}}{\partial net_{pj}} + \frac{\partial e_p}{\partial net^L_{pk}}\frac{\partial net^L_{pk}}{\partial net_{pj}} + \frac{\partial e_p}{\partial net^R_{pk}}\frac{\partial net^R_{pk}}{\partial net_{pj}}\right)\\ \frac{\partial e_p}{\partial net^L_{pj}} &= \sum_{k=1}^{no}\left(\frac{\partial e_p}{\partial net_{pk}}\frac{\partial net_{pk}}{\partial net^L_{pj}} + \frac{\partial e_p}{\partial net^L_{pk}}\frac{\partial net^L_{pk}}{\partial net^L_{pj}} + \frac{\partial e_p}{\partial net^R_{pk}}\frac{\partial net^R_{pk}}{\partial net^L_{pj}}\right)\\ \frac{\partial e_p}{\partial net^R_{pj}} &= \sum_{k=1}^{no}\left(\frac{\partial e_p}{\partial net_{pk}}\frac{\partial net_{pk}}{\partial net^R_{pj}} + \frac{\partial e_p}{\partial net^L_{pk}}\frac{\partial net^L_{pk}}{\partial net^R_{pj}} + \frac{\partial e_p}{\partial net^R_{pk}}\frac{\partial net^R_{pk}}{\partial net^R_{pj}}\right)\end{aligned} \tag{A-9}$$

And from (10), we obtain

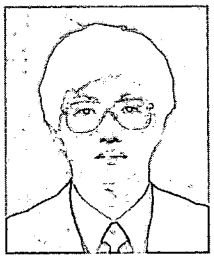
$$\begin{aligned}\frac{\partial net_{pk}}{\partial net_{pj}} &= w_{kj}\cdot o_{pj}(1-o_{pj}),\\ \frac{\partial net_{pk}}{\partial net^L_{pj}} &= 0 = \frac{\partial net_{pk}}{\partial net^R_{pj}},\end{aligned} \tag{A-10}$$

$$\frac{\partial net^L_{pk}}{\partial net_{pj}} = \begin{cases} w^L_{kj}o_{pj}(1-o_{pj}) & \text{if } net^L_{pk} = w^L_{kj}o_{pj},\\ 0 & \textit{otherwise},\end{cases}$$

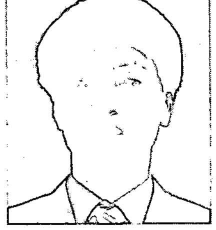
$$\begin{aligned}\frac{\partial net^L_{pk}}{\partial net^L_{pj}} &= \begin{cases} w_{kj}f(net_{pj}-net^L_{pj})(1-f(net_{pj}-net^L_{pj})) & \\ \qquad \text{if } w_{kj}\geq 0 \text{ and } net^L_{pk} = o^L_{pj}w_{kj}, & \\ 0 \qquad \textit{otherwise}, & \end{cases}\\ \frac{\partial net^L_{pk}}{\partial net^R_{pj}} &= \begin{cases} |w_{kj}|f(net_{pk}+net^R_{pk})(1-f(net_{pk}+net^R_{pk})) & \\ \qquad \text{if } w_{kj}<0 \text{ and } net^L_{pk} = o^R_{pj}|w_{kj}|, & \\ 0 \qquad \textit{otherwise}. & \end{cases}\end{aligned} \tag{A-11}$$

$$\begin{aligned}\frac{\partial net^R_{pk}}{\partial net_{pj}} &= \begin{cases} w^R_{kj}o_{pj}(1-o_{pj}) & \text{if } net^R_{pk} = w^R_{kj}o_{pj},\\ 0 & \textit{otherwise},\end{cases}\\ \frac{\partial net^R_{pk}}{\partial net^L_{pj}} &= \begin{cases} |w_{kj}|f(net_{pj}-net^L_{pj})(1-f(net_{pj}-net^L_{pj})) & \\ \qquad \text{if } w_{kj}<0 \text{ and } net^R_{pk} = o^L_{pj}\cdot|w_{kj}|, & \\ 0 \qquad \textit{otherwise}, & \end{cases}\\ \frac{\partial net^R_{pk}}{\partial net^R_{pj}} &= \begin{cases} w_{kj}f(net_{pk}+net^R_{pk})(1-f(net_{pk}+net^R_{pk})) & \\ \qquad \text{if } w_{kj}\geq 0 \text{ and } net^R_{pk} = o^R_{pj}w_{kj}, & \\ 0 \qquad \textit{otherwise}. & \end{cases}\end{aligned} \tag{A-12}$$

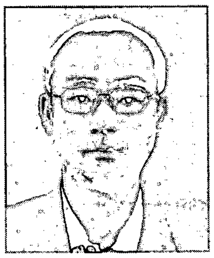
By applying the equations (A-5), (A-6) and (A-7) to the hidden layer, we can obtain $\frac{\partial e_p}{\partial w_{ji}}$, $\frac{\partial e_p}{\partial w^L_{ji}}$ and $\frac{\partial e_p}{\partial w^R_{ji}}$ from the equations (A-8) ~ (A-12).



Jun Jae Lee received the B.S. and M.S. in electronic engineering from Kyungpook National University, Taegu, Korea, in 1986 and 1991, respectively, and is currently a Ph.D. student in the Department of Electronic Engineering Kyungpook National University. He worked for Kyungpook National Univerrsity as a research assistance from September 1991 to July 1993. Since 1995, he has joined an assistant professor in Computer Engineering at Dongseo University. His main interests are in image processing, pattern recognition, computer vision and neuro-fuzzy systems.



Seok Yoon Hwang received the B.S. degree in mathematics from Kyungpook University in 1978, M.S. degree from Taegu University in 1983, and Ph.D. degree from Keimyung University in 1989. He is an associate professor of the Department of Mathemathics, Taegu University. His research interests include fuzzy systems, neural networks, image processing and pattern analysis.



Dug Hun Hong received the B.S. and M.S. degrees in Mathematics from Kyungpook National university, Taegu Korea in 1981 and 1983, respectively. He received the M.S. and Ph.D degrees from the University of Minnesota in 1988 and 1990, respectively. From 1991 to 1996 he was a professor in the Department of Statistics at Catholic University of Taegu-Hyosung. Since 1997 he has joined a professor in the school of Automotive Engineering at the same university. His research interests include probability theory and fuzzy theory with applications.