Overview of Fuzzified Neural Networks with Comparison of Learning Mechanism

Ching-Yi Kuo and Hsiao-Fan Wang

Abstract

A fuzzified neural network copes with fuzzy signals and/or weights so that the information about the uncertainty of input and output can be served in the training process. Since learning process is the main function of fuzzy neural networks, in this study, we focus on review and comparison of the existing learning algorithms, so that the theoretical achievement and the application agenda of each considered algorithm can be clarified from the aspects of computation complexity and accuracy. Two numerical examples of nonlinear mapping of fuzzy numbers and realization of fuzzy IF-THEN rules are used for illustration and analysis.

Keywords: Fuzzy neural networks, Fuzzy arithmetic, Learning algorithm, Back-propagation.

1. Introduction

Construction of an artificial neural network is to mimic the learning capability of human brain on pattern recognition. While neural network techniques can solve various types of problems by its learning ability; fuzzy logic formalized by Zadeh [27] in 1965 is capable of representing vague concepts. Incorporating the concept of fuzzy sets into artificial neural networks possesses the benefits of both paradigms.

Following Zadeh's notion, there are two categories of the hybrid fuzzy neural systems: neuro-fuzzy system and fuzzy-neural system. While a neuro-fuzzy system is to apply a neural network for modifying a fuzzy system [15, 17, 18, 25, 26], a fuzzy-neural system is a fuzzified neural network (FNN) with fuzzy signals and/or weights, so that the information about the uncertainty of the input and output can be processed in a learning procedure. Since learning process is the basis for data mining and pattern recognition, in this study, we focus on analysis and comparison of the existing learning algorithms used in fuzzified neural networks.

Many issues related to fuzzy neural network have been discussed extensively in the literatures. On theoretical studies, most research focused on how a fuzzy neural network is developed to approximate a fuzzy function [2, 6, 20]. Among them, Buckley and Hayashi [2] evaluated its approximation capability and concluded that fuzzy neural networks can not be used as a universal approximator, for which Liu [20] proposed some extended functions to support the argument. Besides, Feuring and Lippe [6] also defined a class of fuzzy functions and proved that they can be approximated by a certain fuzzy neural networks. As regards the applications, developing an effective learning algorithm has been the core topic. In 1994, Buckley and Hayashi [1] have given a thorough survey on fuzzy neural networks and suggested to use more general fuzzy sets in order to facilitate wider applications; for instance, using generalized fuzzy numbers for fuzzy signals/weights. However, this would mean to deal with complex computations of the fuzzy arithmetic. Thus, the research on fuzzy neural networks has put forth for investigating learning algorithms with fuzzy arithmetic and obtained many significant results.

Based on the types of fuzzy arithmetic adopted in the learning process, there are two categories of the algorithms. One, which is the most commonly adopted approach, is using approximate interval arithmetic on level sets of fuzzy numbers [5, 11-14]. The other is using approximate arithmetic on three parameters which are used to represent the fuzzy numbers [16, 24]. Both categories have existing algorithms. To clarify their capabilities in learning, in this article, we shall discuss them in details.

This paper is organized as follows: we first describe the basic structures, notations, and operations of an FNN in Section 2. Then in Sections 3 and 4, we discuss the properties of different types of fuzzy number representation used in the forward and backward operations of FNN learning process respectively. Furthermore, in Section 5, we discuss the computation complexity of these learning algorithms and compare their efficiency and accuracy by two numerical examples of nonlinear mapping of fuzzy numbers and the realization of fuzzy IF-THEN rules. Finally, we conclude this study in Section 6 and discuss the possible aspects for further research.

Corresponding Author: Hsiao-Fan Wang, Department of Industrial Engineering and Engineering Management, National Tsing Hua University, 101, Sec. 2, Kuang Fu Rd., Hsinchu, Taiwan, 300.
E-mail: hfwang@ie.nthu.edu.tw
Manuscript received 10 Jan. 2008; accepted 16 May. 2008.
2. Fuzzy Neural Networks (FNNs)

A fuzzy neural network is a multi-layer feed forward neural network in which the signals and/or the weights are fuzzy sets. Recently, there were different structures of fuzzy neural networks being proposed. Table 1 lists the basic types in use [1]. Since a crisp number can be viewed as a special fuzzy number, our review focuses on FNN3 whose architecture is shown in Fig. 1 that is a fully fuzzified neural network and is considered as the most general form of FNN with fuzzy inputs, weights and outputs.

<table>
<thead>
<tr>
<th>Types</th>
<th>Inputs</th>
<th>Weights</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional NN</td>
<td>Crisp number</td>
<td>Crisp number</td>
<td>Crisp number</td>
</tr>
<tr>
<td>FNN1</td>
<td>Crisp number</td>
<td>Fuzzy number</td>
<td>Fuzzy number</td>
</tr>
<tr>
<td>FNN2</td>
<td>Fuzzy number</td>
<td>Fuzzy number</td>
<td>Fuzzy number</td>
</tr>
<tr>
<td>FNN3</td>
<td>Fuzzy number</td>
<td>Fuzzy number</td>
<td>Fuzzy number</td>
</tr>
</tbody>
</table>

Because the learning mechanism of an FNN is to mimic human’s learning process, therefore it normally is so designed to constantly calibrate along the process in order to target a desired goal. This leads to two constituent parts of learning process in FNN. In other words, the learning mechanism of an FNN consists of two passes of computation through the network: a forward pass and a backward pass. The function of a forward pass is to derive a response from the input by propagating the input vector through the network layer by layer. Then, by comparing the deviation of the response with the desired output, the backward pass provides a calibration function of which the synaptic fuzzy weights are all adjusted in accordance to an error-corrected rule.

![Fig. 1. Architecture of a Fully Fuzzified Neural Network with One Hidden Layers.](Image)

Let us first describe some basic notations used throughout the article: A fuzzy set is denoted by $\bar{\alpha}$ of which the membership function evaluated at $x$ is written as $\mu(x)$, which is in $[0, 1]$ for all $x \in \mathbb{R}$. The level set at membership degree $h$, $h \in [0, 1]$, is called $h$-cut of fuzzy set $\bar{\alpha}$ and is denoted by $\bar{\alpha}_h(x) = \{ x \mid \mu(x) \geq h, 0 \leq h \leq 1 \}$. If $\bar{\alpha}$ represents a fuzzy number which is defined by a normal and convex fuzzy set, $\bar{\alpha}$ can be presented by an interval set of $[\bar{\alpha}_L, \bar{\alpha}_U]$ for any $h \in [0, 1]$, where $\bar{\alpha}_L$ and $\bar{\alpha}_U$ denote the lower limit and upper limit, respectively, of the level set $\bar{\alpha}$. Accordingly, for a fuzzy neural network with fuzzy inputs, weights, and outputs represented by fuzzy numbers, the following notations are defined for each layer $s$, where $s = 1, \ldots, S$. Assuming that there are total $n_i$ neurons in each $s^{th}$ layer and total $S$ layers in the network.

(a) $\bar{\mathbf{y}}_i$, $\bar{\mathbf{y}}_j$, and $\bar{\mathbf{y}}_k$ are the fuzzy signals of neuron $i, j$, and $k$ at the $(s-1)^{th}$, $s^{th}$, and $(s+1)^{th}$ layer, respectively, where $i = 1, \ldots, n_{s-1}; j = 1, \ldots, n_s; k = 1, \ldots, n_{s+1}$. The fuzzy signals are propagated through the network from neurons $i$ to neurons $j$ and then neurons $j$ to neurons $k$. Therefore, the fuzzy inputs $\bar{x}$ at the input layer are propagated to the fuzzy signal $\bar{y}_i$ at the first hidden layer as $s = 1$.

(b) $\bar{\alpha} = [\bar{\alpha}_L, \bar{\alpha}_U]$ of which $i = 1, \ldots, n_0$ propagated to $\bar{y}_i$ at the first hidden layer as $s = 1$ is the fuzzy inputs in the input layer of network.

(c) $\bar{\mathbf{w}} = [\bar{\mathbf{w}}_j]$ and $\bar{\mathbf{b}} = [\bar{\mathbf{b}}]$ are the synaptic fuzzy weights and fuzzy biases for all $i$ and $j$.

Thus, the computations of both-way passes can be described as below:

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not add any kind of pagination anywhere in the paper. All tables and figures will be processed as images.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

2.1 Forward pass

Based on the definitions above, the operations of neuron $j$ at $s^{th}$ layer along a forward pass are

$$\bar{v}_j = \sum_{i=1}^{n_{s-1}} \bar{\mathbf{y}}_i \cdot \bar{w}_{ij} + \bar{\mathbf{b}}_j, j = 1, \ldots, n_s;$$

$$\bar{y}_j = f(\bar{v}_j) = \frac{1}{1 + e^{-\bar{v}_j}}, j = 1, \ldots, n_s.$$  \hspace{1cm} (1)

First, the fuzzy signals are weighted sum to $\bar{v}_j$ by fuzzy weights and biases. Then, $\bar{v}_j$ is transferred to $\bar{y}_j = f(\bar{v}_j)$ by an activation function as usually defined by a logistic function as shown in Eq. (2). The fuzzy inputs of the network, $\bar{x}$, are propagated via the first hidden layer as $\bar{v}_i = \sum_{j=1}^{n_s} \bar{\mathbf{y}}_j \cdot \bar{w}_{ij} + \bar{\mathbf{b}}_i$ and $\bar{y}_i = f(\bar{v}_i)$ where $n_0$ is the number of nodes in input layer.

2.2 Backward pass

To calibrate the inference results from the forward pass, in backward pass, the learning algorithm is developed
from an error back-propagation algorithm [9]. The error is defined by a cost function as
\[ \xi(\hat{\mathbf{w}}) = \frac{1}{2} \sum_{i=0}^{N} (\hat{y}^{(i)} - y^{(i)})^2, \]
where \( \hat{y}^{(i)}, y^{(i)} \) is the desired output, \( \mathbf{d}^{(i)} \) is the input. The forward and backward propagation in Eqs. (1) – (2) and Eqs. (4) – (5) involves computations of fuzzy numbers. Therefore, we shall focus on this issue for our overview.

In the literature, there are two categories of fuzzy number representation used in FNNs, which lead to two types of fuzzy arithmetic: interval arithmetic on level sets of fuzzy numbers [5, 11-14] and approximate arithmetic on three parameters [16, 24]. When level-set is operated, the fuzzy number is presented by an interval with respective to that level of membership degree. Hence, denoting \( h \) level-set of Eqs. (1) and (2) in the forward pass as

\[ h \mathbf{y} = \sum_{j=1}^{n} y_{j} \mathbf{w}_{j} + \mathbf{\theta}, j = 1, \ldots, n, \]
\[ h \mathbf{y} = f(h \mathbf{y}) = \frac{1}{1 + e^{-h \mathbf{y}}}, j = 1, \ldots, n, \]
the interval arithmetic on addition of fuzzy number at \( h \) level-set is
\[ h \mathbf{A} + h \mathbf{B} = [\mathbf{A}^h + \mathbf{B}^h], \]
where assuming of \( h \mathbf{A} = \sum_{i=1}^{n} h_i \mathbf{w}_i \). Besides, the multiplication in Eq. (6) and activation function in Eq. (7) can be obtained from
\[ h \mathbf{y} \times h \mathbf{w} = \min\{h_i \mathbf{y}_i \times h_i \mathbf{w}_i, \mathbf{y}_i \times \mathbf{w}_i, \mathbf{y}_i, \mathbf{w}_i \}, \]
\[ \max\{h_i \mathbf{y}_i \times h_i \mathbf{w}_i, h_i \mathbf{y}_i \times \mathbf{w}_i, \mathbf{y}_i \times h_i \mathbf{w}_i, \mathbf{y}_i \times \mathbf{w}_i \} \]
\[ h \mathbf{y} = f(h \mathbf{y}) = \frac{1}{1 + e^{-h \mathbf{y}}}, \]
where \( \mathbf{\theta} \) is the responsive output in the output layer of FNN when \( h \)-step learning is performed.

2.3 Summary
The learning process of an FNN with iterations from Eq. (1) to Eq. (5) involves computations of fuzzy arithmetic, which, in turn, rely on the representation of the involved fuzzy numbers. Apart from the parameters set a priori for any specific problem of an FNN, the factors which may affect learning capability of FNN in general largely attribute to the representation of the involved fuzzy numbers. Therefore, we shall focus on this issue for our overview.

In this section, based on two types of fuzzy number representation, we shall describe how the fuzzy input signals propagate layer-by-layer to the fuzzy outputs with two types of fuzzy arithmetic in the forward pass of FNN.

3. Fuzzy Arithmetic on Forward Pass

3.1 Level-set approach
Performing interval arithmetic on level sets of fuzzy numbers is a commonly adopted approach to approximate the fuzzy outputs in a fuzzy neural network [5, 11-14]. When level-set is operated, the fuzzy number is presented by an interval with respective to that level of membership degree. Hence, denoting \( h \) level-set of Eqs. (1) and (2) in the forward pass as

\[ \mathbf{\mu}_y(x) = \left\{ \begin{array}{ll}
L & x \leq y \\
R & y \leq x
\end{array} \right., \]

The fuzzy addition [4] of two fuzzy numbers is

\[ \mathbf{\mu}_{y_1 + y_2}(x) = \left\{ \begin{array}{ll}
L & x \leq (y_1 + y_2) \\
R & x \geq (y_1 + y_2)
\end{array} \right., \]

For fuzzy multiplication, by assuming all fuzzy signals being positive and having triangular membership functions, Li, et al. [16] proposed a formulae of \( \mu_{y_1 y_2}(x) \) in which

\[ \mu_{y_1 y_2}(x) = \left\{ \begin{array}{ll}
L & x \leq y_1 y_2 \\
R & x \geq y_1 y_2
\end{array} \right., \]

In which \( \alpha_{y_1 y_2} \) and \( \beta_{y_1 y_2} \) are the non-negative left and right spreads, respectively. L is a monotonically increasing function toward 1, and R is a monotonically decreasing function from 1 with \( L(0) = R(0) = 1 \) and \( L(1) = R(1) = 0 \). Therefore, both of L and R functions have inverse functions denoted by \( L^{-1} \) and \( R^{-1} \).

Ching-Yi Kuo and Hsiao-Fan Wang: Overview of Fuzzified Neural Networks with Comparison of Learning Mechanism 73
and defined the activation function of Eq. (2) by
\[ \mu_{\hat{y}_j}(z) = \max_{i} \left\{ \mu_{\hat{y}_j}(x_i) = \frac{1}{1 + e^{-z_i}} \right\} . \] (14)

Dubios and Prade have considered partially positive or negative numbers to define the following approximate formulas for fuzzy multiplication [3, 4].
\[ \tilde{y}_i \cdot \tilde{w}_j = \left( y_{i1}, \alpha_{i1}, \beta_{i1} \right) \cdot \left( w_{j1}, \alpha_{j1}, \beta_{j1} \right) =
\begin{cases}
\left( y_{i1} \cdot y_{j1}, \alpha_{i1} + w_{j1} \cdot \alpha_{i1}, \beta_{i1} + w_{j1} \cdot \beta_{i1} \right) & \text{when } \tilde{y}_i > 0, \tilde{w}_j > 0; \\
\left( y_{i1} \cdot y_{j1}, -w_{j1} \cdot \alpha_{i1}, -y_{i1} \cdot w_{j1} \cdot \beta_{i1} \right) & \text{when } \tilde{y}_i > 0, \tilde{w}_j < 0; \\
\left( y_{i1} \cdot y_{j1}, y_{i1} \cdot \alpha_{i1} - w_{j1} \cdot \beta_{i1}, y_{i1} \cdot \beta_{i1} - w_{j1} \cdot \alpha_{i1} \right) & \text{when } \tilde{y}_i < 0, \tilde{w}_j > 0; \\
\left( y_{i1} \cdot y_{j1}, -y_{i1} \cdot \beta_{i1} - w_{j1} \cdot \alpha_{i1}, y_{i1} \cdot \alpha_{i1} - w_{j1} \cdot \beta_{i1} \right) & \text{when } \tilde{y}_i < 0, \tilde{w}_j < 0.
\end{cases} \] (15)

However, it can be noted that this formulas cannot allow the fuzzy numbers defined across the origin, and also the triangular membership assumption remains. Therefore, a more general model called three-parameter fuzzy arithmetic approximation method was further proposed by Wang and Kuo [24]. By adopting an L-R fuzzy number denoted as \( \tilde{y}_i = \left( y_{i1}, \alpha_{i1}, \beta_{i1} \right) \), Wang and Kuo’s approach defined the forward operations in Eq. (1) by the three-parameter fuzzy arithmetic approximation as:

(i) The mode
\[ v_i = \sum_{j=1}^{n_j} y_{ij} \cdot w_{ij} + \theta_j; \] (16)

(ii) The left spread
\[ \alpha_v = \sum_{i=1}^{n_i} \left( y_{ij} \cdot \alpha_{ij} + w_{ij} \cdot \alpha_{ij} \right) + \sum_{i=1}^{n_i} \left( y_{ij} \cdot \alpha_{ij} + w_{ij} \cdot \alpha_{ij} \right); \] (17)

(iii) The right spread
\[ \beta_v = \sum_{i=1}^{n_i} \left( y_{ij} \cdot \beta_{ij} + w_{ij} \cdot \beta_{ij} \right) + \sum_{i=1}^{n_i} \left( y_{ij} \cdot \beta_{ij} + w_{ij} \cdot \beta_{ij} \right). \] (18)

The activation function of Eq. (2) was also a logistic function and approximated by
\[ \begin{align*}
\hat{y}_{i1} &= \frac{1}{1 + e^{-v_i}}, \\
\alpha_{ij} &= \frac{1}{1 + e^{-\alpha_{ij}}}, \quad j = 1, \ldots, n_{si}, \\
\beta_{ij} &= \frac{1}{1 + e^{-\beta_{ij}}}. \end{align*} \] (19)

For a single layer perceptron, they have shown that the approximate errors in Eqs. (16) – (19) will not cause any effect in the learning process [24].

3.3 Summary
For those learning algorithms using the level-set approach in fuzzy arithmetic, the output fuzzy number is also represented as interval sets. Because this approach is operated on level sets, the forms of membership functions of the fuzzy signals are not restricted; thus, this approach is applicable for all types of fuzzy numbers conducted in FNNs. However, because its outputs are level sets, we cannot describe the specific membership functions for its outputs. On the other hand, three-parameter approach has overcome this shortage on fuzzy arithmetic, yet the approach requires more efforts on calculation.

Moreover, although in different research disciplines, there were research which developed different methods for approximating fuzzy arithmetic [7, 8, 21, 23], their high complexity has prevented them from applying to the backward learning rule defined in Eq. (4) of an FNN.

In next section, we shall introduce the backward learning processes of FNNs as Eqs. (3) – (5) when approximate fuzzy arithmetic discussed in sections 3.1 and 3.2 are used. Then, these learning algorithms with different calculations of the approximate fuzzy arithmetic will be compared on both efficiency and accuracy by computer simulations.

4. Fuzzy Arithmetic on Backward Pass

The backward learning rule is designed to update the synaptic fuzzy weights so that the cost function of \( \xi(\tilde{w}) \) can be minimized. Because different approximations of fuzzy numbers contribute to different definitions of cost functions, Table 2 summarizes the cost functions, \( \xi(\tilde{w}) \), used in FNNs based on different categories of approximate fuzzy arithmetic. In this table, \( Q \) is total number of learning iterations, \( H \) is the number of level sets, and \( n_S \) is the number of neurons in output layers of FNN.

In the following sections, we compare the basic properties of these approaches. The detailed processes can be referred to Appendix A.

4.1 Level-set approach
By performing the interval arithmetic on the level sets of fuzzy numbers, Ishibuchi et al. [11] defined a cost function as shown in Table 2 in which \( k \) was the neuron in output layer; \( q \) was the learning pattern performed in fuzzy neural network, \( y_{k}^{(q)} = \left[ y_{k}^{(q)}(1), y_{k}^{(q)}(2) \right] \) was the network output, and \( d_{k}^{(q)} = \left[ d_{k}^{(q)}(1), d_{k}^{(q)}(2) \right] \) was the target (desired) output for the \( h \)-level sets. The squared errors of
To relax the symmetric triangular assumption, Ishibuchi and Nii extended the learning algorithm with the cost function defined in [11] so that the asymmetric triangular and trapezoid fuzzy weights can be applied [13, 14]. But, because the fitting of fuzzy outputs to fuzzy targets was no good for the small $h$ values, the authors adopted Krishnamraju et al. ‘cost function [10] showed in Table 2 with no weight of $h$, and showed the improvement in simulations with asymmetric fuzzy signals.

Moreover, Dunyak and Wunsch defined a cost function by allowing a relative weighting for lower and upper limits of level sets [5]. This concept was different from that used in Ishibuchi et al.’s objective function with weights of $h$ on level sets, and the lower and upper limits of level sets were weighted by the positive weights of $\omega_{h1}$ and $\omega_{h2}$ in the object function. Besides, to avoid the distortion on membership functions of fuzzy weights which were caused by the interval arithmetic of level sets (Fig. 2), Dunyak and Wunsch [5] indicated that the level-cut constraint of Eq. (20) should be satisfied.

$$h_{1}w_{j}^{h_{1}} \leq h_{2}w_{j}^{h_{2}} \leq \cdots \leq h_{n-1}w_{j}^{h_{n-1}} \leq h_{n}w_{j}^{h_{n}} \leq \cdots \leq h_{m}w_{j}^{h_{m}}, \quad (20)$$

where $h_{1}, h_{2}, \ldots, h_{m}$ described the $H$ cutting levels used in interval arithmetic on fuzzy numbers. Then, they proposed a learning algorithm to satisfy this constraint by a transformation showed in Eq. (A.17). However, the transformation proposed does not simplify the representation of the fuzzy weights, because they use more parameters in representing fuzzy weights than those of Ishibuchi et al.’s algorithms. Thus their algorithm requires more computation time.

These algorithms [5, 11-14] have shown their learning capabilities for FNNs, however, the performed fuzzy outputs could not be described by specific membership functions. Therefore, learning algorithms in which more general fuzzy weights are used in order to facilitate wider applications are needed to be further investigated.

### 4.2 Three-parameter approach

Different from the level-set approach, Li et al. [16] proposed a learning algorithm of fuzzy neural network based on three-parameter operations by their defined cost function shown in Table 2. Because their algorithm was developed for all positive fuzzy numbers, a simple heuristic method was also proposed as below to shift the negative to positive fuzzy numbers. That is, by Eqs. (3) – (5) the new fuzzy weights $\tilde{w}_{j} = \left(w_{j}, \alpha_{w_{j}}, \beta_{w_{j}}\right)$ were obtained. The details learning can be referred to Appendix A-(II), then if $\tilde{w}_{j} = \left(w_{j}, \alpha_{w_{j}}, \beta_{w_{j}}\right)$, then let

$$\tilde{w}_{j} = \left(w_{j} - \alpha_{w_{j}}, \beta_{w_{j}}\right)$$

in which $r_{1}$, $r_{2}$ and $r_{3}$ were positive random numbers. Besides, if $w_{j}^{h_{1}} \leq \alpha_{w_{j}}$ or $w_{j}^{h_{1}} + \beta_{w_{j}} \leq w_{j}^{h_{2}}$, $\tilde{w}_{j}$ will be reassigned into

$$\tilde{w}_{j} = \left(\min_{w_{j}} \left(w_{j} - \alpha_{w_{j}}\right), \max_{w_{j}} \left(w_{j} + \beta_{w_{j}}\right)\right)$$

This learning algorithm showed its capability and was less time-consuming when comparing with the learning algorithms based on level-set approximation. The capability of this algorithm was that only three parameters of the mode, the left and the right values of fuzzy numbers were considered. However, the restriction of using triangular fuzzy numbers in models limits the

<table>
<thead>
<tr>
<th>Fuzzy Arithmetic</th>
<th>Cost Functions used in Learning Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level-set approximation</td>
<td>Ishibuchi et al. [11, 12], Ishibuchi and Nii [13, 14]</td>
</tr>
<tr>
<td>$z(\tilde{w}) = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{d} h_{i} L_{j}^{(i-1)} L_{j}^{(i)} + \left(h_{i} R_{j}^{(i-1)} R_{j}^{(i)}\right)$</td>
<td></td>
</tr>
<tr>
<td>$z(\tilde{w}) = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{d} \left(h_{i} L_{j}^{(i-1)} L_{j}^{(i)}\right) + \left(h_{i} R_{j}^{(i-1)} R_{j}^{(i)}\right)$</td>
<td></td>
</tr>
<tr>
<td>Dunyak and Wunsch [5]</td>
<td></td>
</tr>
<tr>
<td>$z(\tilde{w}) = \frac{1}{2H} \sum_{j=1}^{n} \sum_{i=1}^{d} \left(w_{i} \alpha_{w_{i}} L_{j}^{(i-1)} L_{j}^{(i)}\right) + \left(w_{i} \beta_{w_{i}} R_{j}^{(i-1)} R_{j}^{(i)}\right)$</td>
<td></td>
</tr>
<tr>
<td>Three-parameter approximation</td>
<td>Li et al. [16]</td>
</tr>
<tr>
<td>$\xi_{l}(\tilde{w}) = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{d} \left(h_{i} L_{j}^{(i-1)} L_{j}^{(i)}\right)$</td>
<td></td>
</tr>
<tr>
<td>$\xi_{r}(\tilde{w}) = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{d} \left(h_{i} L_{j}^{(i-1)} L_{j}^{(i)} + \left(h_{i} \alpha_{w_{i}} L_{j}^{(i-1)} L_{j}^{(i)}\right)\right)$</td>
<td></td>
</tr>
<tr>
<td>$\xi_{w}(\tilde{w}) = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{d} \left(h_{i} L_{j}^{(i-1)} L_{j}^{(i)} + \left(h_{i} \beta_{w_{i}} R_{j}^{(i-1)} R_{j}^{(i)}\right)\right)$</td>
<td></td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td></td>
</tr>
<tr>
<td>$\xi(\tilde{w}) = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{d} \left(h_{i} L_{j}^{(i-1)} L_{j}^{(i)}\right) + \left(h_{i} \alpha_{w_{i}} L_{j}^{(i-1)} L_{j}^{(i)}\right) + \left(h_{i} \beta_{w_{i}} R_{j}^{(i-1)} R_{j}^{(i)}\right)$</td>
<td></td>
</tr>
</tbody>
</table>
applications of this algorithm in practice. Therefore, in order to have broader applications, Wang and Kuo [24] proposed an approximate fuzzy arithmetic of L-R fuzzy numbers for the learning algorithm of a fully fuzzified neural network.

4.3 Summary

In this section, we have discussed two basic approaches of the backward learning in FNN; one is based on level-set approximation, and the other is based on three-parameter approximation in fuzzy arithmetic. For those algorithms based on level-set approximation which is the most commonly used method, the core issue is how to avoid the distortion of the resultant membership functions in fuzzy weights as shown in Fig. 2. On the other hand, the learning algorithms based on three-parameter approximation can provide the definite membership functions for the fuzzy outputs so that the applicability of the model is broadened.

In the next section, we shall further compare the efficiency and accuracy of the salient algorithms resulted from the above analysis.

5. Comparison of The Performances

First, the time complexity of two representative categories of the algorithms will be analyzed. Second, two numerical examples will be used to compare the accuracy of the algorithms while the efficiency will also be presented.

5.1 Time complexity of the algorithms

(i) With level-set operations.

For an $S$-layers fuzzy neural network with fully interconnections, it requires

$$\sum_{i=1}^{S} n(s) \cdot n(s_{i+1})$$

multiplications to generate a network output from an input. Also, the leaning in the backward pass requires the same complexity as in the forward pass. In most of the existing learning algorithms of the fuzzy neural network, the operations of the fuzzy numbers were based on the $h$-level operations [5, 11-14]. Hence, the complexity with $Q$ learning iterations is

$$Q \cdot 2 \cdot 4H \cdot \sum_{i=1}^{S} n(s) \cdot n(s_{i+1}) = 8QH \cdot \sum_{i=1}^{S} n(s) \cdot n(s_{i+1}),$$

where $H$ is the number of the $h$-levels.

(ii) With three-parameter operations.

Such as the operations of fuzzy numbers adopted in [16] and [24] which were operated with three parameters, the computation complexity is

$$Q \cdot 2 \cdot 3 \cdot \sum_{i=1}^{S} n(s) \cdot n(s_{i+1}) = 6Q \cdot \sum_{i=1}^{S} n(s) \cdot n(s_{i+1}).$$

5.2 Accuracy and efficiency of the algorithms

In this subsection, we compared the learning algorithms on both accuracy and efficiency of three basic models, one developed by Ishibuchi et al. [11] was the type performing interval arithmetic on level sets; and two proposed by Li et al. [16] and Wang & Kuo [24] respectively were based on three-parameter approximation for fuzzy arithmetic. In addition, the cost function proposed by Krishnamraju et al. [10] with no weighting value $h$ was also considered when testing Ishibuchi et al.’s learning algorithm in [11]; this consideration was denoted as [11]$^*$ when showing the learning results. On the other hand, Dunyak and Wunsch’s algorithm [5] has shown more complicated in computation than Ishibuchi et al.’s algorithms from its presentation of the fuzzy weights. Therefore, their learning algorithms were not considered in our numerical comparison.

In order to compare these algorithms, two cases of nonlinear mapping of fuzzy number and the realization of fuzzy IF-THEN rules were conducted by computer simulations. When the network converged to the required level of $10^{-9}$, the accuracy was measured by the errors between learning outputs and desired outputs of the network. In the meantime, the efficiency was measured by the number of learning iterations required when the training procedure stopped.

First, these two cases are illustrated as follows:

**Case 1: Nonlinear mapping of fuzzy numbers**

Adopting the case given by Ishibuchi et al. [11] with three input-output pairs of

$$(\bar{x}_1, \bar{d}_1) = (0.2, 0.1, 0.1_{\text{LT}}, 0.35, 0.15, 0.15_{\text{LT}}),$$

$$(\bar{x}_2, \bar{d}_2) = (0.5, 0.1, 0.1_{\text{LT}}, 0.7, 0.1, 0.1_{\text{LT}}),$$

$$(\bar{x}_3, \bar{d}_3) = (0.8, 0.1, 0.1_{\text{LT}}, 0.45, 0.05, 0.05_{\text{LT}}),$$

where the symmetric triangular membership functions were assumed as shown in Fig. 3, the case was conducted by using a fuzzy neural network with single input and single output.
Case 2: Realization of fuzzy IF-THEN rules

This case was also adopted from Ishibuchi et al. [11] where a fuzzy system with nine fuzzy if-then rules was given for training, in which two linguistic values in the antecedent and single fuzzy output in the consequence were summarized in Table 3 with their membership functions defined in Fig. 4.

IF \( x_1 \) is small and \( x_2 \) is small, THEN \( y \) is large.
IF \( x_1 \) is small and \( x_2 \) is medium, THEN \( y \) is medium large.
IF \( x_1 \) is small and \( x_2 \) is large, THEN \( y \) is medium.
IF \( x_1 \) is medium and \( x_2 \) is small, THEN \( y \) is medium large.
IF \( x_1 \) is medium and \( x_2 \) is medium, THEN \( y \) is small.
IF \( x_1 \) is medium and \( x_2 \) is large, THEN \( y \) is small.
IF \( x_1 \) is large and \( x_2 \) is small, THEN \( y \) is medium.
IF \( x_1 \) is large and \( x_2 \) is medium, THEN \( y \) is small.
IF \( x_1 \) is large and \( x_2 \) is large, THEN \( y \) is small.

Table 3. Given Nine Fuzzy IF-THEN Rules in Case 2.

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>S</th>
<th>MS</th>
<th>M</th>
<th>ML</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>L</td>
<td>ML</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>L</td>
<td>ML</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>ML</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>ML</td>
<td>M</td>
<td>S</td>
<td>S</td>
<td>M</td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

When implementing the learning algorithms on both cases, the settings of parameters of the FNN were:
(a) The number of hidden layer = 1;
(b) The number of hidden neurons = 6;
(c) The learning constant \( \gamma = 0.9 \);
(d) The momentum constant \( \gamma = 0.9 \);
(e) The stopping criterion was
\[
\Delta \xi_{\text{w}}(\tilde{w}) = \xi_{\text{w}}(\tilde{w}^{(t)}) - \xi_{\text{w}}(\tilde{w}^{(t-1)}) \leq 10^{-9},
\]
where \( \xi_{\text{w}}(\tilde{w}^{(t)}) = \xi(\tilde{w}^{(t)})/q \).

By checking the analysis of variance by the F-test with 95% confidence (Appendix B), the p-values of the analyses in learning errors and numbers of iterations required were less than 0.05 for both cases. Hence, the null hypothesis of that there was no significant difference between these four algorithms was rejected. That is, the significant evidences have shown that there existed differences within these algorithms. Then, in order to further compare their accuracy and efficiency, Tukey’s Honestly Significant Differences (HSD) test [22] was used to do the multiple significance tests. The results were analyzed in the next section.

5.3 Analysis of the comparison

For the first case of nonlinear mapping of fuzzy numbers, Table 6 to Table 9 showed the results of multiple comparisons of the learning errors and numbers of learning iterations required in convergence of the FNN by Tukey’s HSD test at 95% confidence, i.e. the significant level is 0.05. In Table 6, for example, the mean difference between Ishibuchi et al. [11] and Li et al.’s [16] algorithms was -1.046 \( \times 10^{-1} \); the significance of 0.000 was less than 0.05 so that the difference was significant. Therefore, it can be conclude that Ishibuchi et al.’s algorithm was more accurate than that of Li et al.’s.
Besides, Table 7 showed homogeneous subsets of these algorithms based on the learning errors. It can be noted that Li et al.’s algorithm was the least accurate in these four learning algorithms; Wang and Kuo’s algorithm [24] was more accurate than the algorithm in [11]. Moreover, there was no significant difference at 0.05 significant level in the homogeneous subsets of algorithms [(24), (11)*] and [(11)*, (11)].

Table 6. Case 1: Multiple Comparisons of The Learning Errors by Tukey’s HSD Test.

<table>
<thead>
<tr>
<th>Method</th>
<th>Difference (A-B)</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ishibuchi et al. [11]</td>
<td>1.086 × 10^-1</td>
<td>0.000*</td>
<td>3.823 × 10^-2</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>-1.046 × 10^-1</td>
<td>-1.020 × 10^-1</td>
<td>-8.897 × 10^-2</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>2.262 × 10^-2</td>
<td>0.000*</td>
<td>7.011 × 10^-2</td>
</tr>
<tr>
<td>Ishibuchi et al. [11]*</td>
<td>1.176 × 10^-2</td>
<td>0.000*</td>
<td>3.823 × 10^-2</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>0.105 × 10^-1</td>
<td>0.000*</td>
<td>8.897 × 10^-2</td>
</tr>
<tr>
<td>Ishibuchi et al. [11]*</td>
<td>0.115 × 10^-1</td>
<td>0.000*</td>
<td>3.823 × 10^-2</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>1.272 × 10^-1</td>
<td>0.000*</td>
<td>3.823 × 10^-2</td>
</tr>
</tbody>
</table>

* The mean difference is significant at the 0.05 level.

Table 7. Case 1: Homogeneous Subsets of Algorithms in the Learning Errors.

<table>
<thead>
<tr>
<th>Method</th>
<th>N</th>
<th>Subset for 0.05 significant level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less</td>
<td>0.561 × 10^-2</td>
<td>1 2 3</td>
</tr>
<tr>
<td>Ishibuchi et al. [11]*</td>
<td>0.237 × 10^-1</td>
<td>1 2 3</td>
</tr>
<tr>
<td>Ishibuchi et al. [11]</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>More</td>
<td>0.383 × 10^-1</td>
<td>1 2 3</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>


On the comparison of efficiency, Table 8 and Table 9 showed that Li et al.’s algorithm [16] could converge with the least number of learning iterations, and there was significant evidence showing that this algorithm was more efficient than those in Ishibuchi et al.’s algorithm [11] and Wang and Kuo’s algorithm [24]. However, Li et al.’s algorithm was the least accurate (Table 6 and 7). Besides, based on the number of learning iterations required in convergence; Table 9 showed that [(16), [11]*] and [(11)*, (11), [24]) were two homogeneous subsets of these algorithms. It is clear that Wang and Kuo’s algorithm is comparatively more accurate, but requires more iterations than Li et al.’s algorithm in convergence. However, these two algorithms are both based on the three-parameter approximation; their computational complexities described in Section 5.1 are the same and less than that of Ishibuchi et al.’s algorithm.

On the other hand, in the second case of realization of fuzzy IF-THEN rule, Table 10 to Table 13 showed the results of multiple comparisons by Tukey’s HSD test. Comparing the mean difference of learning errors, the results of these four methods shown in Table 10 and Table 11 were very similar to those in Case 1. The accuracy of Wang and Kuo’s algorithm and Ishibuchi et al.’s algorithm were similar and better than Li et al.’s algorithm (Table 11). The efficiency of Wang and Kuo’s algorithm was comparatively the best because the model could converge with the least number of learning iterations (Table 12 and 13). Based on the number of learning iterations required to stop the learning process, Table 13 showed that the algorithms of [(11)*, (11], and [24] were in the homogeneous subset.

Table 8. Case 1: Multiple Comparisons of the Numbers of Learning Iterations by Tukey’s HSD Test.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Difference (A-B)</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ishibuchi et al. [11]*</td>
<td>3699</td>
<td>-2.404</td>
<td>9802</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>6235</td>
<td>0.039*</td>
<td>12248</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>12855</td>
<td>0.947</td>
<td>7388</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>4985</td>
<td>0.150</td>
<td>11087</td>
</tr>
<tr>
<td>Ishibuchi et al. [11]*</td>
<td>-2626</td>
<td>0.677</td>
<td>8728</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>-7610</td>
<td>0.008*</td>
<td>15071</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>1285</td>
<td>0.947</td>
<td>7388</td>
</tr>
<tr>
<td>Ishibuchi et al. [11]*</td>
<td>4985</td>
<td>0.150</td>
<td>11087</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>7610</td>
<td>0.008*</td>
<td>15071</td>
</tr>
</tbody>
</table>

* The mean difference is significant at the 0.05 level.


<table>
<thead>
<tr>
<th>Method</th>
<th>N</th>
<th>Subset for 0.05 significant level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less</td>
<td>0.561 × 10^-2</td>
<td>1 2 3</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Ishibuchi et al. [11]</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Ishibuchi et al. [11]*</td>
<td>0.237 × 10^-1</td>
<td>1 2 3</td>
</tr>
<tr>
<td>More</td>
<td>0.383 × 10^-1</td>
<td>1 2 3</td>
</tr>
</tbody>
</table>


Table 10. Case 2: Multiple Comparisons of the Learning Errors by Tukey’s HSD Test.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Difference (A-B)</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ishibuchi et al. [11]*</td>
<td>-3699</td>
<td>0.394</td>
<td>-2.404</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>-3699</td>
<td>0.394</td>
<td>-2.404</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>-3699</td>
<td>0.394</td>
<td>-2.404</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>-3699</td>
<td>0.394</td>
<td>-2.404</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>0.150</td>
<td>-11087</td>
<td>11087</td>
</tr>
<tr>
<td>Ishibuchi et al. [11]*</td>
<td>4985</td>
<td>0.150</td>
<td>11087</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>7610</td>
<td>0.008*</td>
<td>15071</td>
</tr>
</tbody>
</table>

* The mean difference is significant at the 0.05 level.

When testing Ishibuchi et al.’s learning algorithm of [11], an alternative cost function proposed by Krishnamurju et al. [10] with no weighting value h was also considered.

functions of the learning results of fuzzy outputs of which sixteen missing fuzzy IF-THEN rules were generated from the trained fuzzy neural network. By comparing the fuzzy outputs with each of the five linguistic values shown in Fig. 4; Table 15 lists the best-fit results. Both results of these two numerical examples confirm the learning and predictive abilities of Wang and Kuo’s approach in [24].

Table 15. The Complete Fuzzy IF-THEN Rule Table Generated by the FNN of Case 2.

<table>
<thead>
<tr>
<th>Method</th>
<th>S</th>
<th>MS</th>
<th>M</th>
<th>ML</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

79

Because Wang & Kuo’s approach was comparatively more accurate and efficient; further investigation on the prediction abilities of their algorithm was carried out from these two cases.

First, Case 1 of mapping the nonlinear fuzzy input–output relations was employed to show the learning ability. Table 14 and Fig. 5 showed the learning results of fuzzy outputs after 10000 iterations, of which the three fuzzy patterns of \((\bar{x}_1, \bar{d}_1)\), \((\bar{x}_2, \bar{d}_2)\) and \((\bar{x}_3, \bar{d}_3)\) were used for learning, and two new fuzzy patterns \(\bar{x}_2 = (0.35, 0.1, 0.1)\) and \(\bar{x}_4 = (0.65, 0.1, 0.1)\) were used for testing.

Table 14. Learning Results and the New Fuzzy Patterns in Case 1.

<table>
<thead>
<tr>
<th>Fuzzy inputs</th>
<th>Desired outputs</th>
<th>Fuzzy outputs</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>([0.5, 0.1, 0.1])</td>
<td>([0.35, 0.1, 0.1])</td>
<td>([0.350, 0.132, 0.132])</td>
<td>([0.000, 0.002, 0.000])</td>
</tr>
<tr>
<td>([0.5, 0.1, 0.1])</td>
<td>([0.7, 0.1, 0.1])</td>
<td>([0.700, 0.097, 0.100])</td>
<td>([0.000, -0.003, 0.000])</td>
</tr>
<tr>
<td>([0.8, 0.1, 0.1])</td>
<td>([0.45, 0.05, 0.05])</td>
<td>([0.450, 0.056, 0.051])</td>
<td>([0.000, 0.006, 0.001])</td>
</tr>
</tbody>
</table>

Second, for prediction ability, the nine if-then rules given in Table 3 of Case 2 were used to train the fuzzy neural network. Then, Fig. 6 showed the membership
mimic a human learning process, the learning mechanism of FNN was designed to consist of forward and backward passes with respective to derivation and calibration purposes. The operations designed for these two passes rely on the representation of a fuzzy number assumed for preceding these operations by fuzzy arithmetic.

From the literature, such representations were classified into two categories of the level-set approximation and three-parameter approximation. Different approximations on fuzzy arithmetic operated in FNNs lead to different learning performances. Therefore, the representative algorithms from each category were reviewed and compared by their efficiency and accuracy in performance in this study.

When performing the level-set approximation in operating fuzzy neural networks, the calculated fuzzy outputs are described by interval sets; whereas when performing the three-parameter approximation, the outputs can be directly presented with definite membership functions. Because of this difference, the learning algorithms adopting three-parameter approximation of fuzzy arithmetic have less computation complexity than those with level-set computations. Two numerical cases were provided for illustration in this study. For those algorithms adopting level-set approximation on fuzzy arithmetic [5, 11-14], Dunyak and Wunsch's algorithm [5] was more complicated than Ishibuchi et al.'s algorithms [11-14] in representing the fuzzy weights. Hence, this two learning procedures was not considered in the numerical illustration. On the other hand, for the algorithms adopting three-parameter approximation, Li et al. [16] and Wang & Kuo [24] were the two existing algorithms. Therefore, these three algorithms were discussed and compared intensively on their methodologies and performance with the learning accuracy and efficiency. With computer simulations of these two numerical cases, the results have shown that Wang and Kuo's learning algorithm is comparatively more efficient and accurate.

In the future, representation of a generalized fuzzy number will remain an issue for general applications. For instance, development of four-parameter fuzzy arithmetic approximation should be considered so that the trapezoidal type of fuzzy numbers can be applied. Also, improvement of the learning algorithm by better approximation of the activation function is another possibility.

### 7. Appendix

**Appendix A: Learning Rules of Backward Pass in FNNs**

The derivative \( \frac{\partial \xi(\tilde{w})}{\partial \tilde{w}_j^{(t)}} \) in Eq. (4) can be calculated by chain rules,

\[
\frac{\partial \xi(\tilde{w})}{\partial \tilde{w}_j^{(t)}} = \frac{\partial \xi(\tilde{w})}{\partial \hat{y}_j^{(t)}} \frac{\partial \hat{y}_j^{(t)}}{\partial y_j^{(t)}} = -\hat{y}_j^{(t)} \cdot \frac{\partial \xi(\tilde{w})}{\partial \hat{y}_j^{(t)}}
\]

where \( \hat{y}_j^{(t)} = \frac{\partial \xi(\tilde{w})}{\partial \hat{y}_j^{(t)}} \).

Based on the cost functions showed in Table 2 of the learning algorithms in FNNs, the derivatives are briefly described as follows:

1) **Level-set approach**

\[
\Delta w_j^{(t)} = \left[ \Delta w_j^{L(t)}; \Delta w_j^{U(t)} \right]
\]

\[
\frac{\partial \xi(\tilde{w})}{\partial \tilde{w}_j^{(t)}} = -\eta \left[ \frac{\partial \xi(\tilde{w})}{\partial \hat{y}_j^{(t)}} \hat{y}_j^{(t)} \left[ \Delta w_j^{L(t)} + \Delta w_j^{U(t)} \right] \right]
\]

\[
\frac{\partial \xi(\tilde{w})}{\partial \tilde{w}_j^{(t)}} = \left[ w_j^{U(t)} - w_j^{L(t)} \right]
\]

2) **Ishibuchi et al.'s algorithm** [11]

\[
\xi(\tilde{w}) = \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{m} H_{ij} \left( d_{ij}(t) - y_{ij}(t) \right)^2 + \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{i=1}^{n} \left( \theta_{ijk} d_{ij}(t) - y_{ij}(t) \right)^2
\]

\[
\frac{\partial \xi(\tilde{w})}{\partial \tilde{w}_j^{(t)}} = \frac{\partial \xi(\tilde{w})}{\partial \hat{y}_j^{(t)}} \hat{y}_j^{(t)} \left[ \Delta w_j^{L(t)} + \Delta w_j^{U(t)} \right]
\]

If \( \tilde{y}_j \) is the output of the neuron in the output layer, then

\[
\delta_j^{(t)} = \frac{1}{2} \sum_{j=1}^{n} \left( \theta_{ijk} d_{ij}(t) - y_{ij}(t) \right)^2
\]

If \( \tilde{y}_j \) is the output of the neuron in the s\(^{th}\) hidden layer, then

\[
\delta_j^{(t)} = \frac{1}{2} \sum_{j=1}^{n} \left( \theta_{ijk} d_{ij}(t) - y_{ij}(t) \right)^2
\]

3) **Dunyak and Wunsch's algorithm** [5]

\[
\xi(\tilde{w}) = \frac{1}{2N} \sum_{i=1}^{N} \sum_{k=1}^{m} \sum_{j=1}^{n} \theta_{ijk} \left( d_{ij}(t) - y_{ij}(t) \right)^2 + \theta_{ijk} \left( d_{ij}(t) - y_{ij}(t) \right)^2
\]

If \( \tilde{y}_j \) is the output of the neuron in the output layer, then

\[
\frac{\partial \xi(\tilde{w})}{\partial \tilde{w}_j^{(t)}} = -H \cdot \theta_{ijk} \left( d_{ij}(t) - y_{ij}(t) \right) \left( 1 - y_{ij}(t) \right) \frac{\partial \xi(\tilde{w})}{\partial \hat{y}_j^{(t)}}
\]

If \( \tilde{y}_j \) is the output of the neuron in the s\(^{th}\) hidden layer,
then
$$\frac{\partial \xi}{\partial \theta_{ji}} = \sum_{t} (d_{jt} - y_{jt}) y_{jt} (1 - y_{jt}) \sum_{k} h_{ik} y_{kt} \frac{\partial y_{kt}}{\partial \theta_{ji}}, \quad (A.15)$$

$$\frac{\partial \xi}{\partial \theta_{ji}} = \sum_{t} (d_{jt} - y_{jt}) y_{jt} (1 - y_{jt}) \sum_{k} h_{ik} y_{kt} \frac{\partial y_{kt}}{\partial \theta_{ji}}. \quad (A.16)$$

The transformation of updated $\tilde{w}_{ji}^{(n+1)}$ is
$$\tilde{w}_{ji}^{(n+1)} = \tilde{w}_{ji}^{(n)} + \sum_{t} (d_{jt} - y_{jt}) y_{jt} (1 - y_{jt}) \sum_{k} h_{ik} y_{kt} \frac{\partial y_{kt}}{\partial \theta_{ji}}. \quad (A.17)$$

Thus, the new fuzzy weights represented with level sets
$$\tilde{w}_{ji}^{(n+1)} = \tilde{w}_{ji}^{(n)} + \sum_{t} (d_{jt} - y_{jt}) y_{jt} (1 - y_{jt}) \sum_{k} h_{ik} y_{kt} \frac{\partial y_{kt}}{\partial \theta_{ji}}. \quad (A.18)$$

**A-(II). Three-parameter approach**

*(1) Li et al.'s algorithm [16]*

$$\xi_{\tilde{w}} = \frac{1}{2} \sum_{j=1}^{N} \left( d_{j} - y_{j} \right)^{2} \quad (A.19)$$

$$\xi_{\alpha} = \frac{1}{2} \sum_{j=1}^{N} \left( \left| d_{j} - y_{j} \right| - \alpha_{j} \right) \quad (A.20)$$

$$\xi_{\beta} = \frac{1}{2} \sum_{j=1}^{N} \left( \left| d_{j} - y_{j} \right| + \beta_{j} \right) \quad (A.21)$$

$$\Delta \alpha_{j} = -\eta \frac{\partial \xi}{\partial \alpha_{j}} + \gamma \Delta \alpha_{j}, \quad (A.22)$$

$$\Delta \beta_{j} = -\eta \frac{\partial \xi}{\partial \beta_{j}} + \gamma \Delta \beta_{j}, \quad (A.23)$$

$$\Delta \alpha_{j} + \beta_{j} = -\eta \frac{\partial \xi}{\partial \alpha_{j} + \beta_{j}} + \gamma \Delta \alpha_{j} + \beta_{j}, \quad (A.24)$$

$$\tilde{w}_{j}^{(n+1)} = \left( \tilde{w}_{j}^{(n)} + \alpha_{j}^{(n)} + \beta_{j}^{(n)} \right) \quad (A.25)$$

If $\tilde{y}_{j}$ is the output of the neuron in the output layer, then
$$\frac{\partial \xi_{\tilde{w}}}{\partial \tilde{w}_{ji}} = \left( d_{ji} - y_{j} \right) y_{j} (1 - y_{j}) y_{j}, \quad (A.26)$$

$$\frac{\partial \xi_{\alpha}}{\partial \alpha_{j}} = \left( d_{ji} - y_{j} \right) y_{j} (1 - y_{j}) \left( y_{j} - \alpha_{j} \right), \quad (A.27)$$

If $\tilde{y}_{j}$ is the output of the neuron in the $n^{th}$ hidden layer, then
$$\frac{\partial \xi_{\tilde{w}}}{\partial \tilde{w}_{ji}} = \left( d_{ji} - y_{j} \right) y_{j} (1 - y_{j}) \sum_{t} h_{it} y_{jt} \frac{\partial y_{jt}}{\partial \theta_{ji}}. \quad (A.28)$$

$$\frac{\partial \xi_{\alpha}}{\partial \alpha_{j}} = \left( d_{ji} - y_{j} \right) y_{j} (1 - y_{j}) \sum_{t} h_{it} y_{jt} \frac{\partial y_{jt}}{\partial \theta_{ji}}. \quad (A.29)$$

Thus,
$$\alpha_{j} = \left( \tilde{w}_{j}^{(n+1)} + \alpha_{j}^{(n)} \right) \quad (A.30)$$

$$\beta_{j} = \left( \tilde{w}_{j}^{(n+1)} + \beta_{j}^{(n)} \right) \quad (A.31)$$

$$\tilde{w}_{j}^{(n+1)} = \left( \tilde{w}_{j}^{(n)} + \alpha_{j}^{(n)} + \beta_{j}^{(n)} \right) \quad (A.32)$$

*(2) Wang and Kuo’s algorithm [24]*

$$\xi_{\tilde{w}} = \frac{1}{2} \sum_{j=1}^{N} \left( d_{j} - y_{j} \right)^{2} \quad (A.33)$$

$$\Delta \alpha_{j} = \left( \Delta \tilde{w}_{j}^{(n)} + \Delta \alpha_{j}^{(n)} \right) \quad (A.34)$$

$$\tilde{w}_{j}^{(n+1)} = \left( \tilde{w}_{j}^{(n)} + \Delta \tilde{w}_{j}^{(n)} + \alpha_{j}^{(n)} + \beta_{j}^{(n)} + \Delta \alpha_{j}^{(n)} + \Delta \beta_{j}^{(n)} \right) \quad (A.35)$$

If $\tilde{y}_{j}$ is output of the network, then
\[ \frac{\partial z(w)}{\partial W_{ji}} = \left( d^{(e)}_{ji} - y_{ji}^{(e)} \right) y_{ji}^{(e)} \cdot \left( 1 - y_{ji}^{(e)} \right) \alpha^{(e)}_{ji} \cdot \left( 1 - \alpha^{(e)}_{ji} \right), \]  

(A.38)

\[ \frac{\partial z(w)}{\partial A^{(e)}_{ji}} = \left( d^{(e)}_{ji} - y_{ji}^{(e)} \right) y_{ji}^{(e)} \cdot \left( 1 - y_{ji}^{(e)} \right) \alpha^{(e)}_{ji} \cdot \left( 1 - \alpha^{(e)}_{ji} \right), \]  

(A.39)

\[ \frac{\partial z(w)}{\partial \beta_{ji}} = \rho_{ji} \cdot \left( 1 - \beta_{ji} \right) \cdot \left( \sum_{k=1}^{n_{out}} \delta^{(e)}_{ik} \cdot \rho_{ik} \right) y_{ji}^{(e)} \cdot \left( 1 - y_{ji}^{(e)} \right) \alpha^{(e)}_{ji} \cdot \left( 1 - \alpha^{(e)}_{ji} \right), \]  

(A.40)

If \( y_{ji} \) is output of \( j \)th hidden layer, then

\[ \frac{\partial z(w)}{\partial W_{ji}} = y_{ji}^{(e)} \cdot \left( 1 - y_{ji}^{(e)} \right) \cdot \left( \sum_{k=1}^{n_{out}} \delta^{(e)}_{ik} \cdot \alpha^{(e)}_{ik} \right) y_{ij}^{(e)} \cdot \left( 1 - y_{ij}^{(e)} \right), \]  

(A.41)

\[ \frac{\partial z(w)}{\partial A^{(e)}_{ji}} = \alpha^{(e)}_{ji} \cdot \left( 1 - \alpha^{(e)}_{ji} \right) \cdot \left( \sum_{k=1}^{n_{out}} \delta^{(e)}_{ik} \cdot \alpha^{(e)}_{ik} \right) y_{ij}^{(e)} \cdot \left( 1 - y_{ij}^{(e)} \right), \]  

(A.42)

\[ \frac{\partial z(w)}{\partial \beta_{ji}} = \rho_{ji} \cdot \left( 1 - \beta_{ji} \right) \cdot \left( \sum_{k=1}^{n_{out}} \delta^{(e)}_{ik} \cdot \beta^{(e)}_{ik} \right) y_{ij}^{(e)} \cdot \left( 1 - y_{ij}^{(e)} \right), \]  

(A.43)

where \( \delta^{(e)}_{ik}, \delta^{(e)}_{ik}, \delta^{(e)}_{ik} \) are the propagated local gradients of the mode, left spread and right spread in neuron \( k \), respectively.

Appendix B. Analysis of Variance

Case 1: Nonlinear mapping of fuzzy numbers

(1) The learning errors

One-way ANOVA: Ishibuchi et al. [11], [1]a, Li et al. [16], Wang and Kuo [24]

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>3</td>
<td>0.039112</td>
<td>0.103037</td>
<td>191.50</td>
<td>0.000</td>
</tr>
<tr>
<td>Error</td>
<td>116</td>
<td>0.006416</td>
<td>0.000358</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>119</td>
<td>0.371528</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Individual 95% CIs For Mean Based on Pooled SD/Dev

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>Mean</th>
<th>SD/Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ishibuchi et al. [11]</td>
<td>30</td>
<td>0.353923</td>
<td>0.01412</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>30</td>
<td>0.144211</td>
<td>0.003367</td>
</tr>
<tr>
<td>Wang and Kuo [24]</td>
<td>30</td>
<td>0.01541</td>
<td>0.00792</td>
</tr>
<tr>
<td>Pooled SD/Dev = 0.02320</td>
<td></td>
<td>0.040</td>
<td>0.080</td>
</tr>
</tbody>
</table>

Case 2: Realization of fuzzy IF-THEN rules

(1) The learning errors

One-way ANOVA: Ishibuchi et al. [11], [1]a, Li et al. [16], Wang and Kuo [24]

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>3</td>
<td>1.087E+09</td>
<td>3.62E+08</td>
<td>4.41</td>
<td>0.006</td>
</tr>
<tr>
<td>Error</td>
<td>116</td>
<td>9.33E+08</td>
<td>8.22E+05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>119</td>
<td>1.062E+10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Individual 95% CIs For Mean Based on Pooled SD/Dev

<table>
<thead>
<tr>
<th>Level</th>
<th>N</th>
<th>Mean</th>
<th>SD/Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ishibuchi et al. [11]</td>
<td>30</td>
<td>16125</td>
<td>670</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>30</td>
<td>9801</td>
<td>319</td>
</tr>
<tr>
<td>Pooled SD/Dev = 9908</td>
<td></td>
<td>1000</td>
<td>12000</td>
</tr>
</tbody>
</table>

Acknowledgment

The authors are grateful to the anonymous referees for their helpful comments and suggestions to improve the presentation of the paper. This work was financially supported by the National Science Council of the Republic of China with Project Number NSC93-2213-E007-062.

References


Ching-Yi Kuo received the B.S. degree in Industrial Engineering from Chung Yuan Christian University (CYCU), Chungli, Taiwan, R.O.C., in 1998, and the M.S. degree in Industrial Engineering and Engineering Management from National Tsing Hua University (NTHU), Hsinchu, Taiwan, R.O.C., in 2000. She is currently working toward the Ph.D. degree at the same university, NTHU. Her research interests include pattern recognition, fuzzy neural networks, and fuzzy systems.

Hsiao-Fan Wang is the Distinguished Chair Professor of National Tsing Hua University, Taiwan, Republic of China. She has been teaching at the Department of Industrial Engineering and Engineering Management at the same university, NTHU after she graduated from Cambridge University, UK in 1981, and now is also the Vice Dean of the College of Engineering. She used to be the Head of the Department of IEEM, NTHU, President of Chinese Fuzzy Systems Association, Vice President of International Fuzzy Systems Association and Erskine Fellow of Canterbury University, NZ. Also, she has been awarded the Distinguished Research Award from National Science Council of Taiwan, R.O.C.; Distinguished Contracted Research Fellow of NSC and Distinguished Teaching Award of Engineering College, NTHU. She used to be the editor-in-chief of the Journal of Chinese Industrial Engineering Association; also the Journal of Chinese Fuzzy Set and Theories and now is the area editor of several international journals. Her research interests are in Multicriteria Decision Making, Fuzzy Set Theory and Operations Research.