A Fuzzy Neural Network with Fuzzy Impact Grades

Song Hengjie 1, Miao Chunyan 1, Shen Zhiqi 2, Miao Yuan 3

1: School of Computing Engineering, Nanyang Tech University, Singapore
2: School of Electrical and Electronic Engineering, Nanyang Tech University, Singapore
3: School of Computer Science and Mathematics, Victoria University, Australia

Abstract. Fuzzy rule derivation is often difficult and time-consuming, and requires expert knowledge. This creates a common bottleneck in fuzzy system design. In order to solve this problem, many fuzzy systems that automatically generate fuzzy rules from numerical data have been proposed. In this paper, we propose a fuzzy neural network based on mutual subsethood (MSBFNN) and its fuzzy rule identification algorithms. In our approach, fuzzy rules are described by different fuzzy sets. For each fuzzy set representing a fuzzy rule, the universe of discourse is defined as the summation of weighted membership grades of input linguistic terms that associate with the given fuzzy rule. In this manner, MSBFNN fully considers the contribution of input variables to the joint firing strength of fuzzy rules. Afterwards, the proposed fuzzy neural network quantifies the impacts of fuzzy rules on the consequent parts by fuzzy connections based on mutual subsethood. Furthermore, to enhance the knowledge representation and interpretation of the rules, a linear transformation from consequent parts to output is incorporated into MSBFNN so that higher accuracy can be achieved. In the parameter identification phase, the backpropagation algorithm is employed, and proper linear transformation is also determined dynamically. To demonstrate the capability of the MSBFNN, simulations in different areas including classification, regression and time series prediction are conducted. The proposed MSBFNN shows encouraging performance when benchmarked against other models.

Keywords: Fuzzy neural network, mutual subsethood, fuzzy rule identification

1 Introduction

Fuzzy systems have been successfully used in a variety of applications, such as pattern recognition [1~4], automatic control [5~7] and fuzzy inference systems [8~12]. A fuzzy system consists of a set of fuzzy IF-THEN rules. Conventionally, the derivation of fuzzy IF-THEN rules often relies on a substantial amount of heuristic observation to express proper strategy’s knowledge. However, it is difficult for human experts to examine all the input-output data from a complex system to construct a number of proper rules for the fuzzy system. To cope with this problem, several fuzzy neural networks have been proposed and investigated [12~19]. As an integrated system of fuzzy logic and neuron networks, fuzzy neural networks combine the human inference style and natural language description of fuzzy systems with the learning and parallel processing of neural networks [20~22]. Recent results [23~26] show that fuzzy neural network is a promising approach to reap the benefits of both fuzzy systems and neural networks to solve their respective problems. Neural networks extract
information from the systems to be learned or controlled, while fuzzy logic techniques most often use linguistic information from experts. As a result, such hybrid intelligent systems alleviate the shortcomings of both techniques, including common problems encountered in the design of fuzzy rule-based system, such as the determination of the membership functions, the identification of the fuzzy rules as well as the fuzzy inference process. Such problems in fuzzy rule-based systems can be resolved by using neural network techniques. Analogically, the integration of fuzzy concepts in neural networks provides a more transparent interpretation for the inner works of neural networks. There are numerous approaches to integrate fuzzy systems and neural networks. The extensive review and categorization on fuzzy neural networks can be found in [27–30].

Two typical types of fuzzy neural networks are TSK-type fuzzy neural networks [31–33] and Mamdani-type [34–37]. These two type fuzzy neural networks have both advantages and disadvantages. Many researches [38–40] have shown that the TSK-type fuzzy neural network achieves superior performance in network size and learning accuracy than Mamdani-type fuzzy neural networks. However, in TSK-type fuzzy neural networks, the consequence layer of each rule is a function of input variable, which leads to vague interpretations for the inner works of neural networks and restricts its application domain. In addition, the traditional TSK-type fuzzy neural network does not take full advantage of the mapping capability that the consequent part offers. In contrast with TSK-type fuzzy neural network, Mamdani-type fuzzy neural networks provide more transparent interpretation on how the networks work. However, the T-norm operators commonly used in Mamdani-type fuzzy neural networks, such as minimum operator, tend to ignore necessary dimension information for fuzzy reasoning, thereby causing unsatisfactory reasoning results.

In order to improve the accuracy of Mamdani-type neural networks and to process linguistic information with fuzzy inputs, fuzzy outputs and fuzzy weights, Tanaka and his colleagues [41–43] proposed a series of approaches and applications with the capacity of processing linguistic input, linguistic output and linguistic weights. In their methods, the weights, inputs and outputs of fuzzy neural networks are fuzzified using fuzzy numbers represented by $\alpha$ - level sets. Since the $\alpha$ - level sets of fuzzy numbers are intervals, the operations in traditional neural networks are extended to the closed intervals. The calculations in the fuzzy neural network are performed by using interval arithmetic operations for $\alpha$ - level sets. However, the interval arithmetic operations on $\alpha$ - level sets made the computation of neural networks more complex. Moreover, since the fuzzy numbers are propagated through the whole neural networks, the time of computation and the required memory are $2h$ times of those in the traditional neural networks, where $h$ represents the number of $\alpha$ - level sets. In order to solve the above problems, models that employ interval arithmetic [41] [43] or mutual subsethood [45] [46] to process the linguistic information of fuzzy inputs, fuzzy outputs and fuzzy weights have been proposed. In addition, Hisao et al. [4] [23] examined the effect of fuzzy IF-THEN rules with certainty grades (i.e., rule weights) in fuzzy rule-based system and showed that the rule weights play important role when a fuzzy rule-based system is mixture of general rules. Based on these observations, they defined a series of methods to measure the rule weights. In their studies, the calculation of rule weights greatly depended on the predetermined fuzzy partition of the $n$ -dimensional pattern space and the predefined membership functions. Although acknowledging that learning is necessary for determining membership functions, and for fine tuning the weights in fuzzy neural networks, the studies in [4] [23] did not mention the learning algorithm involved in fuzzy-rule based system design.
To attack above problems, this paper proposes a five-layer fuzzy neural network and discusses its rule identification algorithms in detail. In contrast with other fuzzy neural networks, MSBFNN has the following novelities.

a) MSBFNN provides a better estimate of the joint firing strength of fuzzy rules. In MSBFNN, all fuzzy rules are defined as fuzzy sets with respect to the linguistic terms of input variables which associate with the given rule, and described as Gaussian membership functions with different centroids and spreads. Correspondingly, the discourse of the fuzzy rules is defined as the summation of weighted membership grades of inputs. As a consequence, the firing strength of fuzzy rules is calculated by using Gaussian membership function, rather than minimum fuzzy operator \([41-43]\) or product operator \([45-46]\), which are commonly used in most fuzzy neural networks. Therefore, MSBFNN neither ignores the important dimension information underlying input vectors, nor causes the firing strength of fuzzy rules to degrade rapidly as the input dimensions increase, as the min operator \([41-43]\) or product operator \([45-46]\) do. In this manner, MSBFNN fully considers the contribution of input variables to the joint firing strength of fuzzy rules and gives a more efficient estimate.

b) MSBFNN employs mutual subsethood \([2]\) to measure the impacts of fuzzy rules on the consequent parts, which propagate to the consequent nodes along fuzzy connections. In our proposed model, the application of mutual subsethood avoids using interval arithmetic operations for \(\alpha\) - level sets and provides a more intuitive interpretation of the impacts of rule weights on the consequent parts.

c) MSBFNN takes advantage of the mapping capability offered by the consequent parts more efficiently to approximate the desired output values. MSBFNN partitions each output into a set of linguistic terms (consequent parts). The consequent parts are then defuzzified by standard-volume-based centroid defuzzification \([2]\). Finally, MSBFNN generates the numeric output by a linear combination of the defuzzified consequent parts. In contrast with other fuzzy neural networks, this method takes advantage of the mapping capability that the consequent parts offer more efficiently.

The rest of this paper is organized as follows. Section 2 describes MSBFNN and discusses the functionalities and corresponding operations of each layer in MSBFNN in detail. Section 3 describes in details the learning algorithm employed to tune the related parameters in MSBFNN. In Section 4, the improved performance of MSBFNN is demonstrated by testing the model in 3 different applications, including classification for XOR, Iris data and Breast Cancer Wisconsin, regression for Servo data, and prediction for Mackey-Glass time series. The comparisons of the experiment results of MSBFNN with other models are also given in Section 4. Finally, Section 5 presents the conclusions.

2 Structure of the MSBFNN

Nauck and Kruse \([47]\) discussed the effect of rule weights in fuzzy rule-based systems for function approximation problems. Ishibuchi et al. \([4]\ [23]\) examined the effect of certainty grades on the performance of fuzzy IF–THEN rules for pattern classification problems and showed a similar relation between rule weights and membership functions. Our fuzzy IF–THEN rules with certainty grades in MSBFNN, shown below, follow the above research results \([4]\ [23]\ [47]\).
Rule $j$: IF $x_1$ is $A^{u_1}_j$ and... and $x_i$ is $A^{u_i}_j$ and... and $x_N$ is $A^{u_N}_j$

THEN $y$ is $y^k$ with $\varepsilon^j$

where $x_i$ and $y$ denote an input variable and output variable respectively; $i = 1, \ldots, N$ is the input index; $A_i$ represents the fuzzy set in the universe of discourse $x_i$, and $A^{u_i}_j$ represents a fuzzy subset in the input spaces $x_i$, which constitutes $A_i$ and associates with rule $j$; $y^k$ is the $k$th linguistic term of $y$, for example ‘Large’, ‘Medium’ or ‘Small’; Note that $\varepsilon^j$ is the certainty grade of the fuzzy IF–THEN rule $R_j$. Usually $\varepsilon^j$ is a real number in the unit interval $[0,1]$. In MSBFNN, $\varepsilon^j$ measures the impact of $R_j$ on consequent part ‘$y$ is $y^k$’. Therefore, the fuzzy rule-based inference mechanism in MSBFNN is as Table I:

**TABLE I**

<table>
<thead>
<tr>
<th>Inference Mechanism in MSBFNN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> $x_1$ is $A_1$ and... $x_N$ is $A_N$</td>
</tr>
<tr>
<td>Rule 1: IF $x_1$ is $A^{u_1}_1$ and... $x_N$ is $A^{u_N}_1$ THEN $y$ is $y^k$ is $\varepsilon^1$</td>
</tr>
<tr>
<td>Rule $j$: IF $x_1$ is $A^{u_1}_j$ and... $x_N$ is $A^{u_N}_j$ THEN $y$ is $y^k$ is $\varepsilon^j$</td>
</tr>
<tr>
<td>Rule $M$: IF $x_1$ is $A^{u_1}_M$ and... $x_N$ is $A^{u_N}_M$ THEN $y$ is $y^k$ is $\varepsilon^M$</td>
</tr>
<tr>
<td><strong>Outputs:</strong> $y = \sum_{k=1}^{K} \xi_k A(y^k)$</td>
</tr>
</tbody>
</table>

In Table I, $j = 1, \ldots, M$ is the fuzzy rule index; $k = 1, \ldots, K$ denotes the linguistic term index of $y$; $A(y^k)$ represents the defuzzification of linguistic term $y^k$, which will be discussed in Section 2.4: $\xi_k$ is the coefficient associated with defuzzified $y^k$ with output $y$. It is evident that the proposed MSBFNN encodes the fuzzy rule base in the form of “IF a set of conditions is satisfied, THEN a set of consequences is inferred”. In order to implement the above inference mechanism, MSBFNN adopts the five-layer structure shown in Figure 1.

In MSBFNN, the inputs and outputs are represented by non-fuzzy vectors $X^T = [x_1, x_2, \ldots, x_i, \ldots, x_N]$ and $Y^T = [y_1, y_2, \ldots, y_i, \ldots, y_M]$ respectively, where $N$ and $M$ denote the numbers of the input and output variables. In Figure 1, each input variable $x_i$ is represented by a set of fuzzy subsets $IL^i_n$, where $n_i$ ($n_i = 1, \ldots, N_i$) is the linguistic term index of input variable $x_i$.
Each $IL^n_i$ represents a fuzzy subset in the universe of discourse on $x_i$. The symbol $R_k$ in the rule layer denotes the $k$th rule in the investigated system. Similar to $IL^n_i$, symbol $OL^m_j$ ($m_j = 1, ..., M_j$) represents the $m_j$th linguistic term for the output $y_j$.

Each layer in MSBFNN performs a specific fuzzy operation. Detailed mathematical functions of each layer are introduced in as follows.

2.1 The Input Layer

This layer represents the input linguistic variables in a fuzzy rule. In this layer, nodes directly transmit crisp input values $X^T = [x_1, x_2, ..., x_i, ..., x_n]$ to next layer. The net input $f^{(1)}_i$ and net output $x^{(1)}_i$ of the $i$th node are given in Eq. (1).

$$f^{(1)}_i = x_i$$
$$x^{(1)}_i = f^{(1)}_i$$

(1)

where $x_i$ is the $i$th element of input vector $X^T$.

2.2 The Antecedent Layer

This layer represents the input linguistic terms. The $n_i$th linguistic term of input variable $x_i$ is denoted as $IL^n_i$, that is expressed as a semantic symbol, such as ‘Small (S)’, ‘Medium (M)’ or ‘Large (L)’ etc. As a matter of fact, each $IL^n_i$ represents a fuzzy subset in the universe of discourse on $x_i$. In MSBFNN, The fuzzy set $IL^n_i$ is
modeled by a symmetric Gaussian membership function with centroid \( C_{n_i} \) and spread \( \sigma_{n_i} \) and denoted by \( \mu_{n_i}^{\text{IL}} = (C_{n_i}, \sigma_{n_i}) \).

The application of the symmetric Gaussian membership function, instead of the triangular or trapezoidal function, is to ensure differentiability of these functions. This is necessary for the backpropagation algorithm employed in the learning process of the MSBFNN. The net input \( f_{n_i}^{(2)} \) and net output \( x_{n_i}^{(2)} \) of the \( IL_i^n \)th linguistic node are given in Eq. (2).

\[
\begin{align*}
f_{n_i}^{(2)} &= - \left( x_i^{(1)} - C_{n_i} \right)^2 / \sigma_{n_i}^2 \\
x_{n_i}^{(2)} &= e^{-\left( x_i^{(1)} - C_{n_i} \right)^2 / \sigma_{n_i}^2}
\end{align*}
\] (2)

It is obvious that the fuzzification process of the system is accomplished in the antecedent layer. The net output \( x_{n_i}^{(2)} \) quantifies the membership grade of the input linguistic variable \( x_i \) to the fuzzy subset \( IL_i^n \).

### 2.3 The Rule Layer

Each node in this layer represents a fuzzy IF-THEN rule in MSBFNN. With reference to Fig. 1, the \( k \)th IF-THEN rule is denoted as \( R_k \). In MSBFNN, each \( R_k \) has \( N \) inputs which are weighted outputs of nodes in the antecedent layer \( (x_{n_i}^{(2)}) \) with corresponding crisp weights \( \omega_{k,n_i} \). Therefore, the linear combination of membership grades of linguistic terms \( IL_i^n \) \((i = 1, 2, ..., N \text{ and } n_i \in N_i)\) constitutes the antecedent part of fuzzy rule \( R_k \). As a result for each fuzzy rule \( R_k \), the number and the locations of fuzzy subsets \( IL_i^n \) \((i = 1, 2, ..., N \text{ and } n_i \in N_i)\) lead to the partition of the antecedent space \( D = D_1 \times D_2 \times ... \times D_N \) and formulate a fuzzy region in \( D \) which can be regarded as a multidimensional fuzzy set. Hence, fuzzy rule \( R_k \) can be correspondingly modeled by a Gaussian membership function with the centroid \( C_k \) and spread \( \sigma_k \) and denoted by \( R_k = (C_k, \sigma_k) \). The universe of discourse of fuzzy rule \( R_k \) is defined as the summation of weighted membership grades of the input linguistic variables \( x_i \) \((i = 1, 2, ..., N)\). The net input \( f_k^{(3)} \) and net output \( x_k^{(3)} \) of the \( k \)th node in rule layer are given in Eq. (3).
In Eq. (3), \( x_k^{(3)} \) in essence describes the joint firing strength of fuzzy rule \( R_k \). Compared with other methods for computation of joint firing strength of fuzzy rules, MSBFNN replaces the minimum fuzzy operator \([41~43]\) or product operator \([45~46]\) with Eq. (3) to obtain the joint firing strength of fuzzy rules. By employing crisp \( \omega_{k,n}^i \) to measure the effect of linguistic terms \( IL_k^n \) on the activation of fuzzy rule \( R_k \), MSBFNN provides a more efficient method to describe the impacts of different linguistic terms \( IL_k^n \) \((i = 1, 2, ..., N \text{ and } n_i \in N_j)\) on the firing strength of fuzzy rule \( R_k \). More importantly, the joint firing strength of fuzzy rule \( R_k \) given in Eq. (3) neither ignores the necessary dimension information underlying in input vector, nor causes the firing strength of fuzzy rules to decrease drastically as the number of input dimensions increases, as the min operator \([41~43]\) or product operator \([45~46]\) do. Also, the algebraic expression of Eq. (3) is continuously differentiable, which allows the model to employ gradient-based learning in a straightforward way.

2.4 The Consequent Layer

Nodes in the consequent layer represent the output linguistic terms that form the consequent parts of fuzzy rules \( R_k \) \((k = 1, ..., K)\). With reference to Fig. 1, \( OL_{j}^{m_j} \) \((j = 1, 2, ..., M; m_j \in M_j)\) represents the \( m_j \)th linguistic term of the output variable \( y_j \) where \( M \) denotes the number of output linguistic variables; \( M_j \) denotes the number of linguistic terms of the output variable \( y_j \). In MSBFNN, the defuzzification process is implemented in this layer.

It should be noted that the consequent layers and the rule layer are fully connected by fuzzy weights \( V_{OL_{j}^{m_j},k} \) \((k = 1, 2, ..., K)\) where \( K \) denotes the number of fuzzy rules.

The fuzzy weight \( V_{OL_{j}^{m_j},k} \) from rule \( R_k \) to the output linguistic term \( OL_{j}^{m_j} \) is also described by a Gaussian function with the centroid \( V_{OL_{j}^{m_j},k}^c \) and spread \( V_{OL_{j}^{m_j},k}^\sigma \), which is denoted by \( V_{OL_{j}^{m_j},k} = (V_{OL_{j}^{m_j},k}^c, V_{OL_{j}^{m_j},k}^\sigma) \). In MSBFNN, the defuzzification process is performed using standard volume based centroid defuzzification [16].

As a consequence, the net input \( f_{OL_{j}^{m_j},k}^{(4)} \) and net output \( x_{OL_{j}^{m_j},k}^{(4)} \) of the output linguistic term \( OL_{j}^{m_j} \) are expressed in Eq. (4)
In Eq. (4), $\varepsilon(V_{OL_j}^m, R_k)$ is mutual subsethood [16] that measures the similarity of fuzzy sets. In MSBFNN, mutual subsethood $\varepsilon(V_{OL_j}^m, R_k)$ describes the fuzzy weights from fuzzy rule $R_k$ to the consequent part $OL_j^m$ and quantifies the impact of $R_k$ on $OL_j^m$. The detailed discussion about the calculation of mutual subsethood is presented in Section 3.2.2.

### 2.5 The Output Layer

Each node in output layer represent a output variable $y_j$. Hence, the net input $f_j^{(5)}$ and net output $x_j^{(5)}$ of the $j$th output are calculated using Eq. (5)

\[
\begin{align*}
  f_j^{(5)} &= \sum_{m_j=1}^{M_j} \xi_{f,m_j} \cdot x_j^{(4)}_{OL_j^m} \\
  x_j^{(5)} &= y_j = f_j^{(5)}
\end{align*}
\]

In Eq. (5), $\xi_{f,m_j}$ is the crisp weight from the output linguistic term $OL_j^m$ to the $j$th node in output layer. Eq. (5) indicates clearly that MSBFNN presents each crisp output $y_j$ as the linear combination of defuzzified linguistic terms $OL_j^m$. In this manner, MSBFNN takes full mapping capability offered by consequent parts.

This section discusses the functionalities and corresponding operations of each layer in MSBFNN in detail. It must be noted that the operations in rule layer, consequent layer and output layer are significantly different from those in existing fuzzy neural networks presented in [7, 10, 12, 15, 16, 21, 22, 43, 45]. We believe that these operations are important contributing factors to the high performance of our approach. The following section describes the supervised learning process which is used to tune the related parameters in MSBFNN.

## 3 Supervised Learning

The proposed MSBFNN is trained by supervised learning based on backpropagation algorithm. This involves repeated presentation of a set of input patterns drawn from the training set and compares the calculated value of the network outputs with the desired output value to compute the error. The network weights and parameters of membership
functions are adjusted in terms of error minimizing criterion. Hence, given a group of training data samples
\[
\left( X^T(\tau), D^T(\tau) \right) \quad \tau = 1, 2, ..., T
\]
\[
X^T(\tau) = (x_1(\tau), x_2(\tau), ..., x_i(\tau), ..., x_n(\tau))
\]
\[
D^T(\tau) = (d_1(\tau), d_2(\tau), ..., d_j(\tau), ..., d_M(\tau))
\]
where \( x_i(\tau) \) is the \( i \)th element of the input vector \( X^T \) which is presented to neural network at discrete time \( \tau \). Similarly, \( d_j(\tau) \) is the desired value of output variable \( y_j \) at discrete time \( \tau \).

### 3.1 Iterative Update Equations

The backpropagation algorithm employed in supervised learning process uses the gradient descent method. The squared error \( E(\tau) \) at iteration \( \tau \) is computed in the standard way
\[
E(\tau) = \frac{1}{2} \sum_{j=1}^{M} \left( d_j(\tau) - y_j(\tau) \right)^2
\]
where \( y_j(\tau) \) denotes the computed value of the \( j \)th element of output vector \( Y^T \) at iteration \( \tau \). In terms of the structure of MSBFNN and backpropagation algorithm, the parameters involved in our approach are tuned as following formulas,
\[
\xi_{j,m_j}(\tau+1) = \xi_{j,m_j}(\tau) - \eta \left( \frac{\partial E(\tau)}{\partial \xi_{j,m_j}(\tau)} \right)
\]
\[
\nu_{\sigma}^{\sigma}(\tau+1) = \nu_{\sigma}^{\sigma}(\tau) - \eta \left( \frac{\partial E(\tau)}{\partial \nu_{\sigma}^{\sigma}(\tau)} \right)
\]
\[
\nu_{\sigma}^{\nu}(\tau+1) = \nu_{\sigma}^{\nu}(\tau) - \eta \left( \frac{\partial E(\tau)}{\partial \nu_{\sigma}^{\nu}(\tau)} \right)
\]
\[
C_{\sigma}(\tau+1) = C_{\sigma}(\tau) - \eta \left( \frac{\partial E(\tau)}{\partial C_{\sigma}(\tau)} \right)
\]
\[
\sigma_{\sigma}(\tau+1) = \sigma_{\sigma}(\tau) - \eta \left( \frac{\partial E(\tau)}{\partial \sigma_{\sigma}(\tau)} \right)
\]
\[
\omega_{k,\xi_m}(\tau+1) = \omega_{k,\xi_m}(\tau) - \eta \left( \frac{\partial E(\tau)}{\partial \omega_{k,\xi_m}(\tau)} \right)
\]
\[
C_{\omega_m}(\tau+1) = C_{\omega_m}(\tau) - \eta \left( \frac{\partial E(\tau)}{\partial C_{\omega_m}(\tau)} \right)
\]
\[
\sigma_{\omega_m}(\tau+1) = \sigma_{\omega_m}(\tau) - \sigma \left( \frac{\partial E(\tau)}{\partial \sigma_{\omega_m}(\tau)} \right)
\]
where \( \eta \) is learning rate. For the sake of simplicity, the temporal identifier \( \tau \) is omitted in following discussion.
3.2 Evaluation of Partial Derivatives

The expression of partial derivatives in above update equations is derived as follows.

3.2.1 Layer 5-Output Layer

In this layer, the error signal is calculated. The crisp weight $\xi_{j,m}$ which connects the $m$th output linguistic term $OL_{j}^{m}$ with the $j$th crisp output $y_{j}$ is adjusted according to Eq. (7). For the error derivative with respect to $\xi_{j,m}$

$$\frac{\partial E}{\partial \xi_{j,m}} = \sum_{l=1}^{M} \frac{\partial E}{\partial y_{j}} \frac{\partial y_{j}}{\partial \xi_{j,m}}$$ (15)

From Eq. (5), we can see that $y_{l}$ is independent of given $\xi_{j,m}$ if $l \neq j$. Hence by substituting Eq. (5) and (6) into Eq. (15), $\frac{\partial E}{\partial \xi_{j,m}}$ is rewritten as

$$\frac{\partial E}{\partial \xi_{j,m}} = -(d_{j} - y_{j})\frac{\partial x_{OL}^{(e)}}{\partial \xi_{j,m}}$$ (16)

where $d_{j}$ and $y_{j}$ respectively represent desired value and computed value of the $j$th output variable.

3.2.2 Layer 4- Consequent Layer

In this layer, the centroids $\nu_{OL_{j},k}^{c}$ and widths $\nu_{OL_{j},k}^{\sigma}$ of fuzzy weights $\nu_{OL_{j},k}^{c}$ and $\nu_{OL_{j},k}^{\sigma}$ are tuned according to Eq. (8) and (9). For the error derivative with respect to $\nu_{OL_{j},k}^{c}$ and $\nu_{OL_{j},k}^{\sigma}$

$$\frac{\partial E}{\partial \nu_{OL_{j},k}^{c}} = \sum_{l=1}^{M} \frac{\partial E}{\partial y_{j}} \frac{\partial y_{j}}{\partial \nu_{OL_{j},k}^{c}} \frac{\partial x_{OL}^{(e)}}{\partial \nu_{OL_{j},k}^{c}}$$ (17)

$$\frac{\partial E}{\partial \nu_{OL_{j},k}^{\sigma}} = \sum_{l=1}^{M} \frac{\partial E}{\partial y_{j}} \frac{\partial y_{j}}{\partial \nu_{OL_{j},k}^{\sigma}} \frac{\partial x_{OL}^{(e)}}{\partial \nu_{OL_{j},k}^{\sigma}}$$ (18)

By substituting Eq. (4), (5) and (6) into Eq. (17) and (18), we have
\[
\frac{\partial E}{\partial v_{\omega j}^{\alpha}} = \sum_{i=1}^{u} \frac{\partial E}{\partial y_i} \frac{\partial x_{\omega j}^{\alpha}}{\partial v_{\omega j}^{\alpha}} = \frac{\partial E}{\partial y_j} \frac{\partial x_{\omega j}^{\alpha}}{\partial v_{\omega j}^{\alpha}}
\]

\[
= -(d - y_j) \cdot \xi_{j,m_j}
\]

\[
\frac{\partial E}{\partial v_{\omega j}^{\beta}} = \sum_{i=1}^{u} \frac{\partial E}{\partial y_i} \frac{\partial x_{\omega j}^{\beta}}{\partial v_{\omega j}^{\beta}} = \frac{\partial E}{\partial y_j} \frac{\partial x_{\omega j}^{\beta}}{\partial v_{\omega j}^{\beta}}
\]

\[
= -(d - y_j) \cdot \xi_{j,m_j}
\]

\[
(19)
\]
As mentioned previously, the fuzzy weights connecting fuzzy rule and consequent parts describe the impacts of rules on output linguistic terms. In MSBFNN, the impacts are measured by mutual subsethood \( E(\nu_{\alpha_{j'}^c,k}^c,R_k) \) [16] as shown in Eq. (19) and Eq. (20). Actually, \( E(\nu_{\alpha_{j'}^c,k}^c,R_k) \) measures the similarity between fuzzy rule \( R_k \) and fuzzy set \( \nu_{\alpha_{j'}^c,k}^c \). In the following subsections, we will discuss the calculation of \( E(\nu_{\alpha_{j'}^c,k}^c,R_k) \) and \( E(\nu_{\alpha_{j'}^c,k}^c,\nu_{\alpha_{j'}^c,k}^c) \), which are essential to Eq. (19) and (20).

For the given fuzzy sets \( \nu_{\alpha_{j'}^c,k}^c \) and \( R_k \) that are described by Gaussian membership functions \( \exp\left(-((x - v_{\alpha_{j'}^c,k}^c)/\sigma_{\alpha_{j'}^c,k}^c)^2\right) \) and \( \exp\left(-((x - c_k)/\sigma_k)^2\right) \) respectively, the cardinality \( C(\nu_{\alpha_{j'}^c,k}^c) \) of fuzzy set \( \nu_{\alpha_{j'}^c,k}^c \) and the cardinality \( C(R_k) \) of fuzzy set \( R_k \) can be defined by

\[
C(\nu_{\alpha_{j'}^c,k}^c) = \int_{-\infty}^{\infty} \exp\left(-((x - v_{\alpha_{j'}^c,k}^c)/\sigma_{\alpha_{j'}^c,k}^c)^2\right) dx
\]

\[
C(R_k) = \int_{-\infty}^{\infty} \exp\left(-((x - c_k)/\sigma_k)^2\right) dx
\]

Furthermore, the mutual subsethood \( E(\nu_{\alpha_{j'}^c,k}^c,R_k) \) is formulated as

\[
E(\nu_{\alpha_{j'}^c,k}^c,R_k) = \frac{C(\nu_{\alpha_{j'}^c,k}^c \cap R_k)}{C(\nu_{\alpha_{j'}^c,k}^c \cup R_k)} = \frac{C(\nu_{\alpha_{j'}^c,k}^c \cap R_k) - C(\nu_{\alpha_{j'}^c,k}^c \cap R_k)}{C(\nu_{\alpha_{j'}^c,k}^c \cup R_k) - C(\nu_{\alpha_{j'}^c,k}^c \cup R_k) - C(\nu_{\alpha_{j'}^c,k}^c \cap R_k)} \in [0,1]
\]
As shown in Eq. (19) and (20), \( \frac{\partial E}{\partial \nu_{\text{OL}}^c} \) and \( \frac{\partial E}{\partial \nu_{\text{OL}}^\sigma} \) greatly depend on the calculation of \( \frac{\partial}{\partial \nu_{\text{OL}}^c} \left( \nu_{\text{OL}}^c \right) \) and \( \frac{\partial}{\partial \nu_{\text{OL}}^\sigma} \left( \nu_{\text{OL}}^\sigma \right) \), which are closely related to the centroid and spread of fuzzy set \( \nu_{\text{OL}}^c \) and those of \( R_k \). Three cases of overlap can arise.

3.2.2.1 Case 1

In this case, let \( \nu_{\text{OL}}^c = C_k \). Then, the case includes three sub-cases:

i ) Case 1.1: \( \nu_{\text{OL}}^c = C_k \) and \( \nu_{\text{OL}}^\sigma = \sigma_k \); ii ) Case 1.2: \( \nu_{\text{OL}}^c = C_k \) and \( \nu_{\text{OL}}^\sigma > \sigma_k \); iii ) Case 1.3: \( \nu_{\text{OL}}^c = C_k \) and \( \nu_{\text{OL}}^\sigma < \sigma_k \).

These three sub-cases are shown in Fig. 2.

![Fig. 2. Three sub-cases included in case 1.](image)

Case 1.1 (a) \( \nu_{\text{OL}}^\sigma = \sigma_k \); Case 1.2 (b) \( \nu_{\text{OL}}^\sigma > \sigma_k \); Case 1.3 (c) \( \nu_{\text{OL}}^\sigma < \sigma_k \)

For the two fuzzy sets \( \nu_{\text{OL}}^c \) and \( R_k \) in case 1, either one fuzzy set belongs to the other, or the two fuzzy set are identical. In following sections, in order to facilitate description, the cardinality of fuzzy sets \( \nu_{\text{OL}}^c \) and \( R_k \) are expressed in terms of standard error function

\[
\text{erf}(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-\frac{t^2}{2}} dt
\]

which has limited values \( \text{erf}(-\infty) = \frac{1}{2}, \quad \text{erf}(\infty) = \frac{1}{2} \) and \( \text{erf}(0) = 0 \). Therefore, the cardinality of fuzzy sets \( \nu_{\text{OL}}^c \) and \( R_k \) are given
\[ C(V_{\text{OL},j,k}) = \int_{-\infty}^{+\infty} \exp \left( - \left( \frac{x - V_{\text{OL},j,k}}{\sigma_{\text{OL},j,k}} \right)^2 \right) dx \]
\[ = \sigma_{\text{OL},j,k} \sqrt{\pi} \left( \text{erf} (\infty) - \text{erf} (-\infty) \right) = \sigma_{\text{OL},j,k} \sqrt{\pi} \]

\[ C(R_k) = \int_{-\infty}^{+\infty} \exp \left( - \left( \frac{x - C_k}{\sigma_k} \right)^2 \right) dx \]
\[ = \sigma_k \sqrt{\pi} \left( \text{erf} (\infty) - \text{erf} (-\infty) \right) = \sigma_k \sqrt{\pi} \]  

(25)

For Case 1.1, Case 1.2 and Case 1.3, \( C(V_{\text{OL},j,k} \cap R_k) \) is calculated using Eq. (26), (27) and (28),

\[ C(V_{\text{OL},j,k} \cap R_k) = C(V_{\text{OL},j,k}) = C(R_k) \]  

(26)

\[ C(V_{\text{OL},j,k} \cap R_k) = C(R_k) = \sigma_k \sqrt{\pi} \]  

(27)

\[ C(V_{\text{OL},j,k} \cap R_k) = C(V_{\text{OL},j,k}) = \sigma_{\text{OL},j,k} \sqrt{\pi} \]  

(28)

Substituting Eq. (25), (26), (27) and (28) into Eq. (23), \( \varepsilon(V_{\text{OL},j,k}, R_k) \) can be calculated respectively,

\[ \partial \left( \varepsilon(V_{\text{OL},j,k}, R_k) \right) / V_{\text{OL},j,k} \text{, and } \partial \left( \varepsilon(V_{\text{OL},j,k}, R_k) \right) / \sigma_{\text{OL},j,k} \text{ can be calculated respectively,} \]

\[ \varepsilon(V_{\text{OL},j,k}, R_k) = \begin{cases} 1 & \text{if } V_{\text{OL},j,k} = C_k, \text{ and } \sigma_{\text{OL},j,k} = \sigma_k \\ \sigma_k / V_{\text{OL},j,k} & \text{if } V_{\text{OL},j,k} = C_k, \text{ and } \sigma_{\text{OL},j,k} > \sigma_k \\ V_{\text{OL},j,k} / \sigma_k & \text{if } V_{\text{OL},j,k} = C_k, \text{ and } \sigma_{\text{OL},j,k} < \sigma_k \end{cases} \]  

(29)

\[ \frac{\partial}{\partial V_{\text{OL},j,k}} \left( \varepsilon(V_{\text{OL},j,k}, R_k) \right) = 0, \text{ if } V_{\text{OL},j,k} = C_k \]  

(30)

\[ \frac{\partial}{\partial \sigma_{\text{OL},j,k}} \left( \varepsilon(V_{\text{OL},j,k}, R_k) \right) = \begin{cases} 0 & \text{if } V_{\text{OL},j,k} = C_k, \text{ and } \sigma_{\text{OL},j,k} = \sigma_k \\ -\sigma_k / (V_{\text{OL},j,k})^2 & \text{if } V_{\text{OL},j,k} = C_k, \text{ and } \sigma_{\text{OL},j,k} > \sigma_k \\ 1 / \sigma_k & \text{if } V_{\text{OL},j,k} = C_k, \text{ and } \sigma_{\text{OL},j,k} < \sigma_k \end{cases} \]  

(31)

3.2.2.2 Case 2

In this case, let \( V_{\text{OL},j,k} > C_k \). Then, the case contains three sub-cases
i) Case 2.1: $\nu^C_{oL_{j,k}^i} > C_k$ and $\nu^\sigma_{oL_{j,k}^i} = \sigma_k$; ii) Case 2.2: $\nu^C_{oL_{j,k}^i} > C_k$ and $\nu^\sigma_{oL_{j,k}^i} > \sigma_k$; iii) Case 2.3: $\nu^C_{oL_{j,k}^i} > C_k$ and $\nu^\sigma_{oL_{j,k}^i} < \sigma_k$.

These three sub-cases are portrayed in Fig. 3.

![Fig. 3. Three sub-cases included in case 2.](image)

Case 2.1 (a) $\nu^\sigma_{oL_{j,k}^i} = \sigma_k$; Case 2.2 (b) $\nu^\sigma_{oL_{j,k}^i} > \sigma_k$; Case 2.3 (c) $\nu^\sigma_{oL_{j,k}^i} < \sigma_k$

In these sub-cases, the two fuzzy sets $\nu_{oL_{j,k}^i}$ and $R_k$ cross over. To calculate the cross points, we set $\nu_{oL_{j,k}^i} = R_k$ to obtain the equal points. The cross points are,

$$\lambda_1 = \left(\nu^\sigma_{oL_{j,k}^i} C_k - \sigma_k \nu^C_{oL_{j,k}^i} \right) / \left(\nu^\sigma_{oL_{j,k}^i} - \sigma_k \right)$$

$$\lambda_2 = \left(\nu^\sigma_{oL_{j,k}^i} C_k + \sigma_k \nu^C_{oL_{j,k}^i} \right) / \left(\nu^\sigma_{oL_{j,k}^i} + \sigma_k \right)$$

(32)

According to above discussion, $C(\nu_{oL_{j,k}^i} \cap R_k)$ can be calculated as Eq. (33).
By substituting Eq. (25) and (33) to Eq. (23), the mutual subsethood $e(\nu_{OL_{M,k}}, R_k)$ is drawn as Eq. (34),

$$e(\nu_{OL_{M,k}}, R_k) =$$

$$= \text{if } \nu_{OL_{M,k}} > C_k \text{ and } \nu_{OL_{M,k}} \leq \sigma_k$$

$$\nu_{OL_{M,k}} \sqrt{2 \pi} \left[ \text{erf} \left( \frac{\sqrt{2} (\lambda - \nu_{OL_{M,k}})}{\nu_{OL_{M,k}}} \right) + \frac{1}{2} \right] \left\{ \begin{array}{ll} + \nu_{OL_{M,k}} \sqrt{2 \pi} \left[ \text{erf} \left( \frac{\sqrt{2} (\lambda - \nu_{OL_{M,k}})}{\nu_{OL_{M,k}}} \right) - \frac{1}{2} \right] & \text{if } \nu_{OL_{M,k}} > C_k \text{ and } \nu_{OL_{M,k}} \leq \sigma_k \\
\end{array} \right.$$
Then, $\frac{\partial}{\partial \nu_{\text{oL}_j k}} \left( \nu_{\text{OL}_j k} \right)$ and $\frac{\partial}{\partial \nu_{\text{oL}_j k}} \left( \nu_{\text{OL}_j k} \right)$ can be rewritten as:

\[
\frac{\partial}{\partial \nu_{\text{oL}_j k}} \left( \nu_{\text{OL}_j k} \right) = \frac{\partial}{\partial \nu_{\text{oL}_j k}} \left[ C(V_{\text{OL}_j k} \cap R_k) \right]
\]

\[
= \frac{\partial}{\partial \nu_{\text{oL}_j k}} \left[ \sqrt{\pi} \nu_{\text{OL}_j k} + \sigma_k \right] - C(V_{\text{OL}_j k} \cap R_k)
\]

\[
\frac{\partial}{\partial \nu_{\text{oL}_j k}} \left( \nu_{\text{OL}_j k} \right) = \frac{\partial}{\partial \nu_{\text{oL}_j k}} \left[ C(V_{\text{OL}_j k} \cap R_k) \right]
\]

\[
= \frac{\partial}{\partial \nu_{\text{oL}_j k}} \left[ \sqrt{\pi} \nu_{\text{OL}_j k} + \sigma_k \right] - C(V_{\text{OL}_j k} \cap R_k)
\]

For Case 2 $(\nu_{\text{oL}_j k} > c_k$ and $\nu_{\text{oL}_j k} = \sigma_k)$, $\frac{\partial}{\partial \nu_{\text{oL}_j k}} \left( C(V_{\text{OL}_j k} \cap R_k) \right)$ are derived respectively,

\[
\frac{\partial}{\partial \nu_{\text{oL}_j k}} \left( C(V_{\text{OL}_j k} \cap R_k) \right) = \begin{cases}
\text{if } \nu_{\text{oL}_j k} > c_k \text{ and } \nu_{\text{oL}_j k} = \sigma_k,
= \frac{\partial}{\partial \nu_{\text{oL}_j k}} \left( \nu_{\text{OL}_j k} \right) \right)
\end{cases}
\]
\[
\begin{aligned}
\frac{\partial}{\partial \nu_{\alpha_{ij}^n}} \left( C(\nu_{\alpha_{ij}^n} \cap R_1) \right) &= \begin{cases}
\frac{1}{2} + \operatorname{erf} \left( \frac{\lambda_i - \nu_{\alpha_{ij}^n}}{\nu_{\alpha_{ij}^n}} \right) - \frac{\lambda_i - \nu_{\alpha_{ij}^n}}{\nu_{\alpha_{ij}^n}} \exp \left( -\left( \lambda_i - \nu_{\alpha_{ij}^n} \right)^2 / \nu_{\alpha_{ij}^n}^2 \right) & \text{if } \nu_{\alpha_{ij}^n}^c > c_i \text{ and } \nu_{\alpha_{ij}^n}^{\sigma} > \sigma_i \\
\frac{1}{2} + \operatorname{erf} \left( \frac{\sqrt{2} \lambda_i - \nu_{\alpha_{ij}^n}}{\nu_{\alpha_{ij}^n}^{\sigma}} \right) - \frac{\sqrt{2} \lambda_i - \nu_{\alpha_{ij}^n}}{\nu_{\alpha_{ij}^n}^{\sigma}} \exp \left( -\left( \sqrt{2} \lambda_i - \nu_{\alpha_{ij}^n} \right)^2 / \nu_{\alpha_{ij}^n}^{\sigma}\right) & \text{if } \nu_{\alpha_{ij}^n}^c > c_i \text{ and } \nu_{\alpha_{ij}^n}^{\sigma} > \sigma_i \\
\frac{1}{2} + \operatorname{erf} \left( \frac{\sqrt{2} \lambda_i - \nu_{\alpha_{ij}^n}}{\nu_{\alpha_{ij}^n}^{\sigma}} \right) - \frac{\sqrt{2} \lambda_i - \nu_{\alpha_{ij}^n}}{\nu_{\alpha_{ij}^n}^{\sigma}} \exp \left( -\left( \sqrt{2} \lambda_i - \nu_{\alpha_{ij}^n} \right)^2 / \nu_{\alpha_{ij}^n}^{\sigma}\right) & \text{if } \nu_{\alpha_{ij}^n}^c > c_i \text{ and } \nu_{\alpha_{ij}^n}^{\sigma} < \sigma_i \\
\frac{1}{2} + \operatorname{erf} \left( \frac{\sqrt{2} \lambda_i - \nu_{\alpha_{ij}^n}}{\nu_{\alpha_{ij}^n}^{\sigma}} \right) - \frac{\sqrt{2} \lambda_i - \nu_{\alpha_{ij}^n}}{\nu_{\alpha_{ij}^n}^{\sigma}} \exp \left( -\left( \sqrt{2} \lambda_i - \nu_{\alpha_{ij}^n} \right)^2 / \nu_{\alpha_{ij}^n}^{\sigma}\right) & \text{if } \nu_{\alpha_{ij}^n}^c > c_i \text{ and } \nu_{\alpha_{ij}^n}^{\sigma} < \sigma_i \\
\end{cases}
\end{aligned}
\]
Based on above discussion, by substituting Eq. (33), (37), (38) into (35) and (36), 
\[
\frac{\partial}{\partial \epsilon} \left( \epsilon \frac{\partial v_{m,j,k}^c}{\partial \epsilon} \right) \text{ and } \frac{\partial}{\partial \epsilon} \left( \epsilon \frac{\partial v_{m,j,k}^\sigma}{\partial \epsilon} \right)
\] for case 2 could be calculated.

3.2.2.3 Case 3

In this case, \( v_{m,j,k}^c < C_k \). This case also includes three sub-cases as shown in Fig. 4.

i) Case 3.1: \( v_{m,j,k}^c < C_k \) and \( v_{m,j,k}^\sigma = \sigma_k \).

ii) Case 3.2: \( v_{m,j,k}^c < C_k \) and \( v_{m,j,k}^\sigma > \sigma_k \).

iii) Case 3.3: \( v_{m,j,k}^c < C_k \) and \( v_{m,j,k}^\sigma < \sigma_k \).

Similar to case 2, the cross points of fuzzy sets \( v_{m,j,k}^c \) and \( R_k \) are given in Eq. (32). Therefore, by substituting Eq. (25) and (32) into (23),

\[
\begin{align*}
C(v_{m,j,k}^c \cap R_k) &= \begin{cases} 
\int_{-\infty}^{\lambda_2} \exp \left[ \frac{-(x-C_k)}{\sigma_1} \right] dx + \int_{\lambda_2}^{\infty} \exp \left[ \frac{-(x-C_k)}{\sigma_1} \right] dx & \text{if } v_{m,j,k}^c < C_k \text{ and } v_{m,j,k}^\sigma = \sigma_k \\
\int_{-\infty}^{\lambda_2} \exp \left[ \frac{-(x-C_k)}{\sigma_1} \right] dx + \int_{\lambda_2}^{\infty} \exp \left[ \frac{-(x-C_k)}{\sigma_1} \right] dx & \text{if } v_{m,j,k}^c < C_k \text{ and } v_{m,j,k}^\sigma > \sigma_k \\
\int_{-\infty}^{\lambda_2} \exp \left[ \frac{-(x-C_k)}{\sigma_1} \right] dx + \int_{\lambda_2}^{\infty} \exp \left[ \frac{-(x-C_k)}{\sigma_1} \right] dx & \text{if } v_{m,j,k}^c < C_k \text{ and } v_{m,j,k}^\sigma < \sigma_k 
\end{cases}
\end{align*}
\]
By substituting Eq. (25) and (39) into Eq. (23), we can draw the corresponding $E(V_{\alpha_{OL},k}^*, R_k)$ for case 3,

$$E(V_{\alpha_{OL},k}^*, R_k) =$$

If $v_{\alpha_{OL},k}^* < C_i$ and $v_{\alpha_{OL},k}^a = \sigma_k$

$$\nu_{\alpha_{OL},k}^a \left[ \frac{1}{2} - \text{erf} \left( \frac{\nu_{\alpha_{OL},k}^a}{\nu_{\alpha_{OL},k}^*} \right) \right] + \sigma_i \left[ \frac{1}{2} + \text{erf} \left( \frac{\nu_{\alpha_{OL},k}^a}{\sigma_i} \right) \right]$$

If $v_{\alpha_{OL},k}^a > \sigma_k$

$$\nu_{\alpha_{OL},k}^a \left[ 1 - \text{erf} \left( \frac{\nu_{\alpha_{OL},k}^a}{\nu_{\alpha_{OL},k}^*} \right) \right] + \sigma_i \left[ 1 + \text{erf} \left( \frac{\nu_{\alpha_{OL},k}^a}{\sigma_i} \right) \right]$$

If $v_{\alpha_{OL},k}^a < C_i$ and $v_{\alpha_{OL},k}^a < \sigma_k$

$$\nu_{\alpha_{OL},k}^a \left[ 1 + \text{erf} \left( \frac{\nu_{\alpha_{OL},k}^a}{\nu_{\alpha_{OL},k}^*} \right) \right] + \sigma_i \left[ \text{erf} \left( \frac{\nu_{\alpha_{OL},k}^a}{\sigma_i} \right) \right]$$

(40)

Correspondingly, $\frac{\partial}{\partial v_{\alpha_{OL},k}^*} (C(v_{\alpha_{OL},k}^*, R_i), C_{(v_{\alpha_{OL},k}^*, R_k)})$ and $\frac{\partial}{\partial v_{\alpha_{OL},k}^*} (C(v_{\alpha_{OL},k}^*, R_i), C_{(v_{\alpha_{OL},k}^*, R_k)})$ are derived as follows,
\[
\left\{ \begin{array}{l}
\exp\left\{ -\left( \lambda - \nu^c_{\alpha(c)j,k}/\nu^\sigma_{\alpha(c)j,k} \right)^2 \right\} \quad \text{if } \nu^c_{\alpha(c)j,k} < c_k \text{ and } \nu^\sigma_{\alpha(c)j,k} = s_k \\
\exp\left\{ -\left( \lambda - \nu^c_{\alpha(c)j,k}/\nu^\sigma_{\alpha(c)j,k} \right)^2 \right\} - \exp\left\{ -\left( \lambda - \nu^c_{\alpha(c)j,k}/\nu^\sigma_{\alpha(c)j,k} \right)^2 \right\} \quad \text{if } \nu^c_{\alpha(c)j,k} < c_k \text{ and } \nu^\sigma_{\alpha(c)j,k} > s_k \\
\exp\left\{ -\left( \lambda - \nu^c_{\alpha(c)j,k}/\nu^\sigma_{\alpha(c)j,k} \right)^2 \right\} - \exp\left\{ -\left( \lambda - \nu^c_{\alpha(c)j,k}/\nu^\sigma_{\alpha(c)j,k} \right)^2 \right\} \quad \text{if } \nu^c_{\alpha(c)j,k} < c_k \text{ and } \nu^\sigma_{\alpha(c)j,k} < s_k
\end{array} \right.
\] (41)

\[
\frac{\partial \left( C(v_{\alpha(c)j,k} \cap R_k) \right)}{\partial v^\sigma_{\alpha(c)j,k}} = \begin{cases} 
\left[ \frac{\sqrt{\pi}}{2} \mathrm{erf} \left( \frac{\sqrt{2} \lambda - \nu^c_{\alpha(c)j,k}}{\nu^\sigma_{\alpha(c)j,k}} \right) \right] + \frac{\lambda - \nu^c_{\alpha(c)j,k} - \exp\left\{ -\left( \lambda - \nu^c_{\alpha(c)j,k}/\nu^\sigma_{\alpha(c)j,k} \right)^2 \right\}}{\nu^\sigma_{\alpha(c)j,k}} \quad \text{if } \nu_{\alpha(c)j,k}^c < c_k \text{ and } \nu_{\alpha(c)j,k}^\sigma = s_k \\
\frac{\lambda - \nu^c_{\alpha(c)j,k} - \exp\left\{ -\left( \lambda - \nu^c_{\alpha(c)j,k}/\nu^\sigma_{\alpha(c)j,k} \right)^2 \right\}}{\nu^\sigma_{\alpha(c)j,k}} \quad \text{if } \nu_{\alpha(c)j,k}^c < c_k \text{ and } \nu_{\alpha(c)j,k}^\sigma > s_k \\
\left[ \frac{\sqrt{\pi}}{2} \mathrm{erf} \left( \frac{\sqrt{2} \lambda - \nu^c_{\alpha(c)j,k}}{\nu^\sigma_{\alpha(c)j,k}} \right) \right] - \frac{\sqrt{2} \lambda - \nu^c_{\alpha(c)j,k} - \exp\left\{ -\left( \lambda - \nu^c_{\alpha(c)j,k}/\nu^\sigma_{\alpha(c)j,k} \right)^2 \right\}}{\nu^\sigma_{\alpha(c)j,k}} \quad \text{if } \nu_{\alpha(c)j,k}^c < c_k \text{ and } \nu_{\alpha(c)j,k}^\sigma < s_k \\
\left[ \frac{\sqrt{\pi}}{2} \mathrm{erf} \left( \frac{\sqrt{2} \lambda - \nu^c_{\alpha(c)j,k}}{\nu^\sigma_{\alpha(c)j,k}} \right) \right] - \frac{\sqrt{2} \lambda - \nu^c_{\alpha(c)j,k} - \exp\left\{ -\left( \lambda - \nu^c_{\alpha(c)j,k}/\nu^\sigma_{\alpha(c)j,k} \right)^2 \right\}}{\nu^\sigma_{\alpha(c)j,k}} \quad \text{if } \nu_{\alpha(c)j,k}^c < c_k \text{ and } \nu_{\alpha(c)j,k}^\sigma < s_k
\end{cases}
\] (42)
By substituting Eq. (39), (41), (42) into (35) and (36), \[ \frac{\partial}{\partial \epsilon} \left( \frac{\partial \epsilon}{\partial \nu} \right) \] and \[ \frac{\partial}{\partial \nu} \left( \frac{\partial \nu}{\partial \epsilon} \right) \] are calculated.

Summary: For case 1, case 2 and case 3, based on the calculated \[ \frac{\partial}{\partial \epsilon} \left( \frac{\partial \epsilon}{\partial \nu} \right) \] and \[ \frac{\partial}{\partial \nu} \left( \frac{\partial \nu}{\partial \epsilon} \right) \], by substituting Eq. (29)~(31), (34)~(36) and (40)~(42) into Eq. (19) and (20) respectively, the error derivatives with respect to the centroid and spread of fuzzy weights \[ V_{OL_j}^{m_k} \], \[ \frac{\partial E}{\partial V_{OL_j}^{m_k}} \] and \[ \frac{\partial E}{\partial \sigma_{OL_j}^{m_k}} \] can be calculated.

3.2.3 Layer 3- Rule Layer

In this layer, the centroid \[ C_k \] and spread \[ \sigma_k \] of fuzzy rule \[ R_k \] are adjusted according to Eq. (10) and (11). In addition, the crisp weight \[ \omega_{k,IL_i} \] from input linguistic term \[ IL_i \] to the fuzzy rule \[ R_k \] is modified according to Eq. (12). The error derivative with respect to \[ C_k \] and \[ \sigma_k \] are described as formula (43) and (44),

\[
\frac{\partial E}{\partial C_k} = \sum_{m=1}^{M} \frac{\partial E}{\partial y_j} \sum_{j=1}^{J} \frac{\partial y_j}{\partial C_k} \frac{\partial \epsilon^{(m)}}{\partial C_k} = \sum_{m=1}^{M} \sum_{j=1}^{J} \frac{\partial \epsilon^{(m)}}{\partial C_k} \frac{\partial y_j}{\partial C_k} + \sum_{m=1}^{M} \sum_{j=1}^{J} \frac{\partial \epsilon^{(m)}}{\partial C_k} \frac{\partial y_j}{\partial C_k} (\sigma^2)^2 \epsilon(V_{OL_j}^{m_k}, R_k) \]

\[
\frac{\partial E}{\partial \sigma_k} = \sum_{m=1}^{M} \frac{\partial E}{\partial y_j} \sum_{j=1}^{J} \frac{\partial y_j}{\partial \sigma_k} \frac{\partial \epsilon^{(m)}}{\partial \sigma_k} = \sum_{m=1}^{M} \sum_{j=1}^{J} \frac{\partial \epsilon^{(m)}}{\partial \sigma_k} \frac{\partial y_j}{\partial \sigma_k} + \sum_{m=1}^{M} \sum_{j=1}^{J} \frac{\partial \epsilon^{(m)}}{\partial \sigma_k} \frac{\partial y_j}{\partial \sigma_k} (\sigma^2)^2 \epsilon(V_{OL_j}^{m_k}, R_k) \]
Similarly, the error derivative with respect to crisp weight $\omega_{k,n_i}$ is calculated

$$\frac{\partial E}{\partial \omega_{k,n_i}} = \sum_{j=1}^{M} \frac{\partial E}{\partial y_j} \sum_{m=1}^{M_i} \frac{\partial y_j}{\partial x_{nm,j}} \frac{\partial x_{nm,j}}{\partial \omega_{k,n_i}} \frac{\partial f_j}{\partial \omega_{k,n_i}}$$

$$= -\sum_{j=1}^{M} (d_j - y_j) \sum_{m=1}^{M_i} \left( \sum_{j=1}^{M} \left( \frac{\partial E}{\partial \omega_{k,n_i}} \right) \right)\left( \frac{\partial f_j}{\partial \omega_{k,n_i}} \right)^2$$

$$= -\sum_{j=1}^{M} (d_j - y_j) \sum_{m=1}^{M_i} \left( \frac{\partial E}{\partial \omega_{k,n_i}} \right) \left( \frac{\partial f_j}{\partial \omega_{k,n_i}} \right)^2$$

In Eq. (43)~(45), the mutual subsethood $\epsilon(\nu \alpha_{ij}, R_k)$ is given in Eq. (40).

### 3.2.4 Layer 2- Antecedent Layer

In this layer, the centroid $C_{n_i}$ and width $\sigma_{n_i}$ of input-label node $I_{n_i}^C$ are adjusted according to Eq. (13) and (14). The error derivatives with respect to $C_{n_i}^C$ and $\sigma_{n_i}^C$ are formulated,

$$\frac{\partial E}{\partial C_{n_i}^C} = \sum_{j=1}^{M} \frac{\partial E}{\partial y_j} \sum_{m=1}^{M_i} \frac{\partial y_j}{\partial x_{nm,j}} \frac{\partial x_{nm,j}}{\partial C_{n_i}^C} \frac{\partial f_j}{\partial C_{n_i}^C}$$

$$\frac{\partial E}{\partial \sigma_{n_i}^C} = \sum_{j=1}^{M} \frac{\partial E}{\partial y_j} \sum_{m=1}^{M_i} \frac{\partial y_j}{\partial x_{nm,j}} \frac{\partial x_{nm,j}}{\partial \sigma_{n_i}^C} \frac{\partial f_j}{\partial \sigma_{n_i}^C}$$
\[
\frac{\partial E}{\partial \sigma_{\omega_j}} = -\frac{M}{\sum_{m,n} \xi_{m,n}} \sum_{m,n} \frac{f^{(i)}}{\sigma_j^2} \left( \frac{\sum_{k=1}^K f^{(i)} \left( \frac{x^{(i)} - C_j}{\sigma_j} \right) \xi_{k,m} e}{ \sum_{k=1}^K f^{(i)} \left( \frac{x^{(i)} - C_j}{\sigma_j} \right) \xi_{k,m}} \right)
\]

4 Illustrative Examples

To verify the performance of the proposed MSBFNN, we compare and contrast the performance of MSBFNN with other models on four applications:

1) XOR problem;
2) Classification of iris dataset and breast cancer (wisconsin) dataset;
3) Regression for servo dataset;
4) Time series modeling and prediction for Mackey-Glass dataset;

These experiments cover the areas of classification, regression and prediction. These datasets are available from UCI repository and website of IEEE working group on data modeling (http://neural.cs.nthu.edu.tw/jiang/benchmark). Training the MSBFNN model involves the standard application of gradient descent learning. In all applications, the centroids of antecedent parts are initialized randomly within the range of the minimum and maximum values of respective input data. The centroids of the fuzzy rule sets and the weight fuzzy sets are randomized in the range \([0,1]\), and the spreads of all fuzzy sets involved in MSBFNN are randomized in the range \((-1,1)\).

The performances of MSBFNN are subsequently benchmarked against other neural and neural fuzzy systems.

4.1 XOR problem

The XOR problem is a typical classification example that is used for evaluating the capability of fuzzy neural networks to handle non-partitionable problems. For this
problem, some fuzzy neural networks, such as Falcon-FKP [16], Falcon-PFKP [16] and POPFNN-CRI (S) [15] can not generate satisfactory results because the clustering techniques used in these models are not sufficiently flexible to handle non-partitionable problems [16] [24]. In order to solve this problem, C. Quek and Tung proposed GenSoFNN [24]; Mantas and Puche discussed the application of zero-order TSK fuzzy systems in XOR problem [40]. In this section, we use XOR problem to test the performance of the proposed MSBFNN for non-partitionable problems.

The XOR logic function has two inputs and one output. It produces an output only if either one of the inputs is on, but not if both are off or both are on.

**TABLE II**

<table>
<thead>
<tr>
<th>Input-Output Mapping for the XOR Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

In this experiment, each input and the single output in MSBFNN are described by two linguistic terms, namely 'large' (L) and 'Small' (S). Because XOR problem only consists of four class patterns, the constructed MSBFNN with same learning rate ($\eta = 0.65$) are executed repeatedly (5 runs) to validate its performance on XOR. The input-output mapping target of XOR and experiment results are given in Table II. From Table II, we can clearly see that MSBFNN can effectively solve XOR problem. It indicates that MSBFNN can solve the non-partitionable problems which are difficult for other fuzzy neural networks.

In MSBFNN, each fuzzy rule is described with respect to the linguistic terms of inputs which are associated with the given fuzzy rule. Therefore for this example, the four fuzzy rules derived by MSBFNN can be described by Fig. 5. Correspondingly, the membership functions of input $X_1$ and $X_2$ are shown in Fig. 6.

![Fig. 5. Firing strength of fuzzy rules by using MSBFNN with $\eta = 0.65$.](image1)

![Fig. 6. Membership functions of inputs by using MSBFNN with $\eta = 0.65$.](image2)
Fig. 5 clearly demonstrates that, for a fuzzy rule, every input label associated with the rule contributes to the firing strength of the given rule. Therefore for fuzzy neural network, the impacts of all input labels on firing strength of fuzzy rules should be considered for providing an appropriate estimation on firing strength of fuzzy rules. From Fig. 5, we can see that the mechanism, which is used in MSBFNN to calculate firing strength of fuzzy rules, truly reflects the underlying knowledge structure in training dataset and explicitly describes the relationship between firing strengths of fuzzy rules and the relevant input linguistic terms. According to the experiment results, we can summarize four fuzzy rules generated by MSBFNN and compare these rules with Zero-Order TSK as shown in Table III.

<table>
<thead>
<tr>
<th>Rule Index</th>
<th>Zero-Order TSK</th>
<th>MSBFNN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Antecedent Part</td>
<td>Output</td>
</tr>
<tr>
<td>R 1</td>
<td>If '1' (x_1 \leq 0.25) and '2' (x_2 \leq 0.25) Then (y = 0)</td>
<td>If &quot;1(x_1) is S&quot; and &quot;2(x_2) is S&quot;</td>
</tr>
<tr>
<td>R 2</td>
<td>If '1' (x_1 = 0.5) and '2' (x_2 = 0.5) Then (y = 1)</td>
<td>If &quot;1(x_1) is S&quot; and &quot;2(x_2) is L&quot;</td>
</tr>
<tr>
<td>R 3</td>
<td>If '1' (x_1 \geq 0.75) and '2' (x_2 \geq 0.75) Then (y = 0)</td>
<td>If &quot;1(x_1) is L&quot; and &quot;2(x_2) is S&quot;</td>
</tr>
<tr>
<td>R 4</td>
<td>'x'</td>
<td>If &quot;1(x_1) is L&quot; and &quot;2(x_2) is L&quot;</td>
</tr>
</tbody>
</table>

In Table III, \(\lambda(OL_1^S)\) and \(\lambda(OL_1^L)\) represent the defuzzified values of output linguistic terms “Small (S)” and “Large (L)” respectively. By comparing the fuzzy rules extracted from MSBFNN against other models, we also see that the coefficients in ‘Impacts on Consequence’ quantify the impacts of rules on consequences. Additionally as shown in ‘Output’ part, MSBFNN explicitly describes the outcome as linear function of defuzzified values of output linguistic terms. In this manner, MSBFNN provides more transparent interpretation on fuzzy rules. This form of description is intuitive to human cognition.

4.2 Classification

4.2.1 Iris Classification

Anderson’s Iris data [48] was used as the reference experimental dataset. This dataset contains 50 vectors each for the three iris subspecies (Iris setosa, Iris versicolor, and Iris virginica). Each instance of the iris dataset contains four physical attributes, namely sepal length, sepal width, petal length and petal width. Iris dataset has been extensively used to illustrate various clustering and classifier designs. The properties of this dataset are discussed in details in [49] and [50].

An experiment was conducted using the entire 150 data vector set of Anderson’s Iris data. In this simulation, the Iris dataset was partitioned into two parts: Training Set and Test Set. The training set contains 60% data points and there are 30 instances from each of the three iris subspecies. The training and test sets are randomly selected. For this experiment, we employed two outputs to represent the three classes of irises. They are...
namely: Setosa (Class 1 and coded as “01”), Virginica (Class 2 and coded as “10”) and Versicolor (Class 3 and coded as “11”). Each input and output variable are represented by three semantics (“Large: L”, “Medium: M” and “Small: S” respectively).

For each attribute of the Iris dataset, the corresponding membership functions are identified by MSBFNN. Fig. 7 shows the membership functions of sepal length, sepal width, petal width, and petal length by using a leaning rate $\eta = 0.8725$. Table IV presents the classification results of the proposed MSBFNN for the iris experiment. Its performances are subsequently benchmarked against that of Falcon-ART [51], Falcon-FKP [16], Falcon-PFKP [16], Native Bayes classifier (NB), 9-NN classifier and 9-NN$^2$ classifier [52].

![Fig. 7. Membership functions identified by using MSBFNN with $\eta = 0.8725$](image)

Table IV clearly shows that the proposed MSBFNN has significant improvement on classification accuracy over the Falcon-ART, Falcon-PFKP, 9-NN classifier and 9-NN$^2$ classifier. This is essentially due to the different calculation mechanisms to describe the firing strength of fuzzy rules and to measure the impacts of rules on consequent part in MSBFNN.

<table>
<thead>
<tr>
<th>Network</th>
<th>Classification Rates (%)</th>
<th>Mean Classification Rates (%)</th>
<th>Num of Training Epoch</th>
<th>Training Time (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
<td>Class 2</td>
<td>Class 3</td>
<td></td>
</tr>
<tr>
<td>Falcon-ART†</td>
<td>69.70</td>
<td>83.84</td>
<td>73.74</td>
<td>75.76</td>
</tr>
<tr>
<td>Falcon-FKP†</td>
<td>92.93</td>
<td>90.91</td>
<td>87.88</td>
<td>90.57</td>
</tr>
<tr>
<td>Falcon-PFKP†</td>
<td>90.91</td>
<td>84.85</td>
<td>84.85</td>
<td>86.87</td>
</tr>
<tr>
<td>NB classifier *</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>94.0</td>
</tr>
<tr>
<td>9-NN classifier *</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>88.7</td>
</tr>
<tr>
<td>9-NN$^2$ classifier *</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>87.3</td>
</tr>
<tr>
<td>MSBFNN</td>
<td>100</td>
<td>87</td>
<td>100</td>
<td>95.6</td>
</tr>
</tbody>
</table>

As mentioned in section 2.3, MSBFNN takes full advantage of the dimension information contained in the input vector. Hence, MSBFNN provides a better estimation of the joint strength of the various inputs on the given rule. In comparison, the classification accuracy of MSBFNN is comparable with that of NB, which is
slightly higher than that of Falcon-FKP. However, the training times of Falcon-FKP and Falcon-PFKP are much shorter than that of MSBFNN and other methods. On the one hand, the back-propagation algorithm used in MSBFNN is a time-consuming process, but on the other hand this is because that Falcon-PFKP and Falcon-FKP might have not count in the execution time of clustering algorithms used in Falcon-PFKP and Falcon-FKP.

Taking the definition of MBSFNN and the attribute number of Iris dataset into account, for this experiment, the joint firing strength of each rule is a surface of four-dimension space with respect to the membership grades of sepal length, sepal width, petal length and petal width (Y axis in Fig. 8). Fig. 8 gives the projections of the fifth rule on all 3-dimensional space which consists of two attributes associated with the fifth rule and the firing strength of the given rule.

Fig. 8. The projection of the fifth rule with respect to membership grades of attributes in Iris dataset

Fig. 8 demonstrates that the membership grades of each input linguistic terms (‘L’, ‘M’ and ‘S’) contribute to the firing strength of a given fuzzy rule. From this perspective, taking full advantage of the dimension information that underlies the input vectors could provide a better estimate of firing strength of fuzzy rules, as well as improve the performance of classification.

In addition, one of main purposes of complex neural fuzzy system is to generate fuzzy rules and to interpret such rules. Hence, a consistent and accurate fuzzy rule-base is essential for any neural fuzzy system. For this experiment, the constructed MSBFNN generates at most 81 fuzzy rules. In most existing neural fuzzy systems, fuzzy rules are described in simple if-then form. However, in this experiment, MSBFNN employs two outputs to represent the three classes of irises, and each output contains three semantics (Consequent parts, namely $\alpha^{L}_{t_{1}}$, $\alpha^{M}_{t_{2}}$ and $\alpha^{M}_{t_{3}}$). Consequently according to the experiment results and the definition of MSBFNN, the descriptions of fuzzy rules in MSBFNN have significantly different forms. The comparison is given in Table V.

In Table V, the crisp coefficients in ‘Impacts on Consequent Part’ quantify the impacts of fuzzy rules on output linguistic term. In addition, MSBFNN takes full
advantage of the mapping capability that the consequent part offers, which are represented by the crisp coefficients, associated with $\Lambda(OL_{m}^{M_j})$ in ‘Output 1’ and ‘Output 2’ ($m$ denotes the output index; $M_j$ is the linguistic term index; $\Lambda(OL_{m}^{M_j})$ is the defuzzified value of linguistic term $OL_{m}^{M_j}$). The application of the mapping capability provided by the consequent part makes the interpretation of the fuzzy rules more transparent and explicit. The comparison clearly shows that the fuzzy rules derived by MSBFNN are easier to be understood than those generated by other methods.

<table>
<thead>
<tr>
<th>Network</th>
<th>Rule</th>
<th>Description</th>
<th>Impacts on Consequent Parts</th>
<th>Output 1</th>
<th>Output 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Falcon-FKP†</td>
<td>1</td>
<td>If sepal length (SL) is $S$ and sepal width (SW) is $L$ and petal length (PL) is $S$ and petal width is $S$, then iris is Setosa</td>
<td>$0.36034$ on ‘L’ output1 ( $OL_1^{M_1}$)</td>
<td>$0.14608$ on ‘M’ output 1 ( $OL_2^{M_1}$) then $y_1 = -0.04955\Lambda(OL_1^{M_1}) - 0.76815\Lambda(OL_2^{M_1}) - 0.87324L(\Lambda(OL_3^{M_1}))$</td>
<td>$0.13396$ on ‘L’ output 2 ( $OL_1^{M_2}$) $0.86147$ on ‘M’ output 2 ( $OL_2^{M_2}$) then $y_2 = 0.50034\Lambda(OL_1^{M_2}) + 3.31898\Lambda(OL_2^{M_2}) + 0.26303L(\Lambda(OL_3^{M_2}))$</td>
</tr>
<tr>
<td>Falcon-PFKP†</td>
<td>16</td>
<td>If SL is $S$ and SW is $L$ and PL is $S$ and PW is $S$.</td>
<td>$0.67189$ on ‘S’ output 1 ( $OL_3^{M_1}$)</td>
<td>$0.50034$ on ‘L’ output1 ( $OL_1^{M_1}$) $0.86147$ on ‘M’ output 1 ( $OL_2^{M_1}$) then $y_1 = -0.04955\Lambda(OL_1^{M_1}) - 0.76815\Lambda(OL_2^{M_1}) - 0.87324L(\Lambda(OL_3^{M_1}))$</td>
<td>$0.13396$ on ‘L’ output 2 ( $OL_1^{M_2}$) $0.86147$ on ‘M’ output 2 ( $OL_2^{M_2}$) then $y_2 = 0.50034\Lambda(OL_1^{M_2}) + 3.31898\Lambda(OL_2^{M_2}) + 0.26303L(\Lambda(OL_3^{M_2}))$</td>
</tr>
</tbody>
</table>

$\Lambda(OL_{m}^{M_j})$ represents the defuzzified value of linguistic term $OL_{m}^{M_j}$. The application of the mapping capability provided by the consequent part makes the interpretation of the fuzzy rules more transparent and explicit. The comparison clearly shows that the fuzzy rules derived by MSBFNN are easier to be understood than those generated by other methods.

4.2.2 Breast Cancer Wisconsin Classification

In order to further validate the classification capability of MSBFNN in multidimensional space, MSBFNN is used in this section to diagnose cancer based on the breast cancer Wisconsin dataset available in UCI repository. This dataset contains 699 instances with nine attributes. In this dataset, 16 instances have missing values. To be consistent with the literature [52] [53] and [54], we removed the instances with missing values from the dataset and constructed a new dataset with 683 instances. The detailed information about the dataset has been discussed in [53]. In this experiment, 400 instances in the new dataset were randomly chosen as the training set, and the remaining 283 as the test set. When MSBFNN was applied for this experiment, each attribute was divided in to 2 linguistic terms, and the output was represented by 3 linguistic terms.

For each input variable, Fig. 9 shows the membership functions identified by using MSBFNN with $\eta = 0.85$. The experiment result and the comparisons between MSBFNN and the others algorithms are presented in Table VI. Table VI shows that MSBFNN achieves slightly better performance over the test set. In this experiment, MSBFNN
significantly reduces the epoch number for the high performance compared with the traditional BP algorithm for training ANNS. Additionally, the experiment result generated by MSBFNN has a much lower standard deviation.

Since standard deviation reflects the consistency of the classification performance of models, the lower standard deviation obtained by using MSBFNN (3 runs) shows that MSBFNN has stronger tolerance to data variations and better generalization capability.

### TABLE VI

<table>
<thead>
<tr>
<th>Network</th>
<th>Mean Classification Rates (%) (average)</th>
<th>Num of Training Epoch</th>
<th>Standard Deviation (average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAD classifier</td>
<td>96.9/97.2</td>
<td>×</td>
<td>0.9/1.3</td>
</tr>
<tr>
<td>Other approach</td>
<td>96.2</td>
<td>×</td>
<td>0.3</td>
</tr>
<tr>
<td>Abbass et al. (ANN) †</td>
<td>97.5</td>
<td>10000</td>
<td>1.8</td>
</tr>
<tr>
<td>NB classifier *</td>
<td>97.4</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>9-NN classifier †</td>
<td>97.3</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>MSBFNN</td>
<td>98.81</td>
<td>400</td>
<td>0.0164</td>
</tr>
</tbody>
</table>

### 4.3 Servo Regression

This simulation was conducted to evaluate the effectiveness of the proposed MSBFNN in regression using the servo dataset available in UCI repository. Servo dataset contains 167 instances. Each instance contains four physical attributes, namely motor, screw, pgain and vgain. For this simulation, four cross-validation groups of training and test sets were used. They are CV1, CV2, CV3 and CV4. For each cross-validation group, the training set and test set consist of 117 instances and 50 instances respectively. Additionally for this
experiment, each input variable is divided into 5 linguistic terms. The training windows and corresponding test results by using a leaning rate $\eta = 0.75$ are shown in Fig. 10.

![Fig. 10. Test results of servo data by using MSBFNN with $\eta = 0.75$.](image)

In this example, the performance of MSBFNN is evaluated by means and standard deviation of absolute error, which are summarized in Table VII and compared with the classical regression models, such as least squares regression methods (i.e. LR and LRC), ridge regression (RIDGE) and principal components approach (PCA).

<table>
<thead>
<tr>
<th>Network</th>
<th>Means of absolute error</th>
<th>Standard deviation of absolute error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR†</td>
<td>0.363</td>
<td>0.05</td>
</tr>
<tr>
<td>LRC†</td>
<td>0.363</td>
<td>0.05</td>
</tr>
<tr>
<td>RIDGE†</td>
<td>0.362</td>
<td>0.05</td>
</tr>
<tr>
<td>PCR†</td>
<td>0.362</td>
<td>0.05</td>
</tr>
<tr>
<td>CV1 CV2 CV 3 CV 4 average</td>
<td></td>
<td>CV1 CV2 CV 3 CV 4 average</td>
</tr>
<tr>
<td>MSBFNN</td>
<td>0.043 0.122 0.086 0.117 0.092</td>
<td>0.064 0.154 0.01 0.133 0.06</td>
</tr>
</tbody>
</table>

The experiment results show that though MSBFNN has the similar standard deviation of absolute error with others, the means of absolute error is much smaller than that of other regression models. For this experiment, the smaller means of absolute error indicates that MSBFNN has better accuracy for servo dataset over others.

### 4.4 Mackey-Glass Prediction

In this section, MSBFNN is applied for modeling and predicting the future values of a chaotic time series, namely the Mackey-glass (MG) dataset. This problem has been used as a benchmark problem in the areas of neural networks, fuzzy systems and
hybrid systems. The dataset of MG is generated from the following delay differential equation:

\[
\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^2(t-\tau)} - 0.1x(t)
\]

where \( \tau > 17 \). The task of this time series prediction problem is to predict future value \( x(t+\Delta t) \) (\( \Delta t \) being the prediction time step) based on a set of values of \( x(t) \) at certain times. To obtain values at integer time points and to facilitate comparison with previous works, we obtain the raw dataset from website of IEEE working group on data modeling benchmarks (http://neural.cs.nthu.edu.tw/jiang/benchmark). For this example, four past valued are used to predict \( x(t) \), and the input-output data is

\[
[x(t-24), x(t-18), x(t-12), x(t-6); x(t)]
\]

Assuming \( x(0) = 1.2 \), the following experiment was conducted: 1000 data patterns were generated, from \( t = 124 \) to \( t = 1123 \). In these patterns, the first 500 patterns were taken as the training data and the other 500 patterns were used for testing. For this application, each input variable and the single output in MSBFNN are respectively represented by two linguistic terms. Fig. 11 and Table VIII shows the performance of MSBFNN with a learning rate \( 0.75 \). Fig. 11 shows the target data, test results and the mean absolute error. Table VIII compares the performance of MSBFNN with other methods on the basis of NDEI-nondimensional error index which is defined as the root mean square error (RMSE).

Note from Table VIII that MSBFNN outperforms all other models in terms of NDEI, training epochs and training time. Additionally, the rule number of SEIT2FNN and MSBFN is much less in comparison with others. This means that the rule set generated by SEIT2FNN and MSBFN are more efficient to solve MG prediction problem.

5 Conclusions

Neural fuzzy network is the realization of the functionality of fuzzy system using neural network techniques. In recent years, the approach of automatically extracting fuzzy rules from the numerical training data has been an active research topic. In this
paper, we propose a novel fuzzy neural network named MSBFNN, which is applicable to
the systems where the prior knowledge is unavailable in the form of rules. The
proposed model permits the fuzzy rule and its corresponding impact on consequent part
to be modeled by Gaussian fuzzy sets. In MSBFNN, the firing strength of each fuzzy
rule is defined as a Gaussian membership function with respect to the summation of
weighted membership grades of input linguistic terms which associate with the given
fuzzy rule. In this manner, MSBFNN fully considers the contribution of input variables
to the joint firing strength of fuzzy rules. Additionally, MSBFNN employs mutual
subsethood to measure the impacts of fuzzy rules on linguistic parts of outputs. In
training process, the parameters involved in MSBFNN are tuned using a data-driven
gradient descent approach.

The performance of the proposed MSBFNN model is demonstrated by tests
performed on four different benchmarking problems: the XOR problem, classification
of Iris data and breast cancer data, servo regression, and prediction of chaotic time
series. For all the problems selected, the detailed comparisons of performance between
MSBFNN and various other models are reported in the literature. From the
comparisons, we can see that MSBFNN does not only generate satisfactory experiment
results for these problems, but also objectively derives the fuzzy rules, as well as
provides more transparent interpretations to describe the problem domain.

Due to the application of gradient descent learning, the MSBFNN also suffers from
the inherent computation complexity. This makes it unsuitable for online modeling.
Hence, extensive efforts have been devoted to integrate more efficient learning
algorithm with MSBFNN to support capability of online modeling. This will be
reported in our future work.

References

1. W. Pedrycz, “Fuzzy sets in pattern recognition: Methodology and methods,”
2005.
5. C. C. Lee, “Fuzzy logic in control systems: fuzzy logic controller, Parts I
6. C. T. Lin and C. S. G. Lee, “Neural-network based fuzzy logic control and
2000.


38. R. Dvorakova, and P. Husek, “Comment on Computing Extreme Values in -
Stability Issues on Takagi-Sugeno Fuzzy Model-Parametric Approach,” IEEE
39. M. Mahfouf, M. F. Abbod and D.A. Linkens, “Online elicitation of Mamdani-
40. C. J. Mantas and J. M. Puche, “Artificial Neural Networks are Zero-Order
41. H. Ishibuchi, R. Fujikawa, and H. Tanaka, “Neural networks that learn from
42. H. Ishibuchi and H. Tanaka, “Fuzzy regression analysis using neural
weights and fuzzy biases,” in Proc. Int. Joint Cont Neural Networks, San
Francisco, CA, 1993, pp. 1650-1655.
44. Z. Q. Lin, K. Vojislav and I. Akira, “Fuzzified neural network based on fuzzy
45. S. Paul and S. Kumar, “Subsethood product fuzzy neural inference system
2002.
47. D. Nauck and R. Kruse, “How the learning of rule weights affects the
interpretability of fuzzy systems,” in Proc. 7th IEEE Int. Conf. Fuzzy Systems,
49. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis. New 
50. M. Nadler and E. P. Smith, Pattern Recognition Engineering. New York: 
52. K. Petri, L. Jussi, M. Petri and T. Henry, “Unsupervised bayesian
visualization of high-dimensional data,” in Proc. the sixth ACM SIGKDD 
international conference on Knowledge discovery and data mining, Boston, 
Massachusetts, 2000, pp. 325–329.
53. E. Boros, L. P. Hammer and A. Kogan, “An implementation of logical
analysis of data,” IEEE Trans. Knowledge and Data Eng., vol. 12, pp. 292- 
