Membership function learning in fuzzy classification

HON KEUNG KWAN†, YALING CAI† and BIN ZHANG‡

A novel three-layer fuzzy neural network (FNN) is proposed which possesses the structure and learning ability of artificial neural networks, and the classification ability of fuzzy algorithms for pattern classification problems. During learning, the proposed FNN learns the membership function of each fuzzy class from training samples and adaptively organizes its hidden layer. The learning and recall times of the FNN are fast. Simulation results are also presented.

1. Introduction

In a pattern classification problem, a given pattern is to be classified into one of the \( P \) possible classes based on its feature values. If the boundaries between the \( P \) classes are non-overlapping, the pattern can only belong to one class and the classification is 'non-fuzzy'. If the boundaries are overlapping, a pattern can belong to several classes, each to a different degree, and the classification is 'fuzzy'. When given a pattern, a non-fuzzy classifier decides which class this pattern belongs to, while a fuzzy classifier computes the membership values of all the classes for the pattern. The design of membership functions is the key point of a fuzzy classification algorithm.

Artificial neural networks (ANNs) (Lippmann 1987, Lau 1991, Sanchez-Sinencio and Lau 1992) and fuzzy algorithms (Keller et al. 1985, Pal and Majumder 1986, Bezdek and Pal 1992, Kosko 1992) are powerful tools for pattern classifications. An ANN has the advantage of being trainable, but it is a non-fuzzy network that has difficulty in classifying patterns of overlapping characteristics. Traditional fuzzy classification algorithms (Keller et al. 1985, Pal and Majumder 1986, Bezdek and Pal 1992) have the ability to classify such overlapping patterns but they cannot learn the membership functions of the fuzzy classes directly from training samples.

In this paper, a novel fuzzy neural network and its associated learning algorithm is proposed. The proposed FNN can learn arbitrarily shaped membership functions of all fuzzy classes from input–output data (training samples) by adjusting the weights and parameters of its fuzzy neurons (FNs). When a test input sample is provided, the FNN can estimate the membership values of all classes of this test input sample. The learning speed of our proposed FNN is fast.

2. Fuzzy neurons and fuzzy neural networks

A neuron is called a fuzzy neuron (FN) if it has a state \( s \), \( n \) weighted inputs \( (w_i x_i) \) for \( i = 1, \ldots, n \), and \( m \) outputs \( (y_j) \) for \( j = 1, \ldots, m \). The state \( s \) and all the inputs and outputs are real values. Moreover, we have

---

Received 15 December 1992; accepted 8 January 1993.

†Department of Electrical Engineering, University of Windsor, 401 Sunset Avenue, Windsor, Ontario, Canada N9B 3P4.

‡Department of Industrial Engineering, University of Windsor, 401 Sunset Avenue, Windsor, Ontario, Canada N9B 3P4.

0020-7217/93 $10.00 © 1993 Taylor & Francis Ltd.
\[ z = h[w_1x_1, w_2x_2, \ldots, w_nx_n] \quad (1) \]
\[ s = f[z - t] \quad (2) \]
\[ y_j = g_f[s] \quad \text{for } j = 1, \ldots, m \quad (3) \]

where \( z \) is the net input of the FN; \( h[\cdot] \) is the aggregation function; \( t \) is the activation threshold; \( f[\cdot] \) is the activation function; and \( \{g_f[\cdot]\}_{j = 1, 2, \ldots, m} \) represent the \( m \) output functions of the FN.

The structure of the proposed FNN is shown in Fig. 1. There are \( N \) FNs in the first layer for a \( N \)-dimensional classifier. Each FN has one input and \( M \) outputs \( (M \) is the number of FNs in the second layer). The algorithm of the \( i \)th FN \( (i = 1, \ldots, N) \) in the first layer is

\[ s_i^{[1]} = z_i^{[1]} = x_i \quad (4) \]

where \( x_i \) is the input of the \( i \)th FN which represents the \( i \)th feature value of an input pattern; \( z_i^{[1]} \) and \( s_i^{[1]} \) are, respectively, the net input and the state of the \( i \)th FN. Defining \( y_j^{[1]} \) as the output of the \( i \)th FN which is to be passed into the \( j \)th FN in the second layer, we have

\[ y_j^{[1]} = g_{ij}[s_i^{[1]}] = \begin{cases} 1 - 2|s_i^{[1]} - \theta_{ij}|/\alpha & \text{if } \alpha \geq 2|s_i^{[1]} - \theta_{ij}| \\ 0 & \text{otherwise} \end{cases} \quad \text{for } j = 1, \ldots, M \quad (5) \]

where \( \theta_{ij} \) is the central point and \( \alpha \) is a positive parameter of the \( j \)th output function \( g_{ij}[s_i^{[1]}] \). \( \alpha \) and \( \theta_{ij} \) \( (i = 1, \ldots, N \) and \( j = 1, \ldots, M) \) are to be determined by the learning algorithm. We call such an FN the INPUT-FN.

There are \( M \) FNs in the second or hidden layer. Each FN in this layer has \( N \) inputs and one output. The algorithm of the \( j \)th FN in the second layer is

\[ y_j^{[2]} = \begin{cases} \max\{y_j^{[1]} \} & \text{for } j = 1, \ldots, M \\ \min\{y_j^{[1]} \} & \text{for } j = 1, \ldots, M \end{cases} \quad (6) \]

\[ y_1^{[2]} \quad y_2^{[2]} \quad \ldots \quad y_M^{[2]} \quad \text{MAX-FN} \]
\[ y_1^{[2]} \quad y_2^{[2]} \quad \ldots \quad y_M^{[2]} \quad \text{MIN-FN} \]
\[ y_1^{[1]} \quad y_2^{[1]} \quad \ldots \quad y_M^{[1]} \quad \text{INPUT-FN} \]

Figure 1. Proposed fuzzy neural network.
\[ z_j^{[2]} = \min_{i=1}^N |y_i^{[1]}| \]  
\[ y_j^{[2]} = s_j^{[2]} = \begin{cases} 
z_j^{[2]} & \text{if } z_j^{[2]} > t_j \\
0 & \text{otherwise} 
\end{cases} \]  
(7)

where \( t_j \) is the activation threshold of the \( j \)th FN. \( M \) and \( t_j \) (for \( j = 1, \ldots, M \)) are to be determined by the learning algorithm. We call such an FN the MIN-FN.

The third layer consists of \( P \) FNs that represent \( P \) fuzzy classes. Each FN in the output layer has \( M \) inputs and one output. The algorithm of the \( p \)th FN is

\[ y_p^{[3]} = s_p^{[3]} = z_p^{[3]} = \max_{j=1}^M |w_{jp}y_j^{[2]}| \]  
(8)

where \( w_{jp} \) (\( j = 1, \ldots, M \) and \( p = 1, \ldots, P \)) is the connection weight between the \( j \)th FN in the second layer and the \( p \)th FN in the third layer which is to be determined by the learning algorithm. We call such an FN the MAX-FN.

3. Adaptive-organizing learning algorithm

Define \( d_{pk} \) as the desired membership value of the \( p \)th fuzzy class of the \( k \)th training sample (\( k = 1, \ldots, K \)); \( E_t \) as the error limit of the FNN; and \( \delta_t \) as the decreasing step for \( E_t \).

Step 1. Create \( N \) INPUT-FNs in the first layer and \( P \) MAX-FNs in the third layer.

Set a value for \( \alpha \), an initial value for \( E_t \), and a value for \( \delta_t \). Set \( m = 1 \) and \( k = 1 \).

Step 2. Create the \( m \)th FN in the second layer. Set \( \theta_{im} = x_{ik} \) (for \( i = 1, \ldots, N \)), \( t_m = 0 \) and \( w_{mp} = d_{pk} \) (for \( p = 1, \ldots, P \)).

Step 3. For \( p = 1 \) to \( P \), repeat this step: input the \( k \)th training sample to the network and compute the output errors: \( e_p = d_{pk} - y_p^{[3]} \). If \( e_p < -E_t \), find any hidden FN (assuming the \( j \)th MIN-FN) which satisfies \( y_p^{[3]} = w_{jp}y_j^{[2]} \), set \( t_j = y_j^{[2]} \). Repeat this procedure until \( e_p \geq -E_t \).

Step 4. If \( |e_p| \leq E_t \) (for \( p = 1, \ldots, P \)), set \( k = k + 1 \). If \( k \leq K \), go to Step 3. If \( k > K \), go to Step 6.

Step 5. If there is one or more \( e_p > E_t \) (for \( p = 1, \ldots, P \)), set \( m = m + 1 \). Go to Step 2.

Step 6. Input all the \( K \) training samples to the network and compute the maximum absolute error of the FNN as

\[ E_m = \max_{k=1}^K \left( \max_{p=1}^P |d_{pk} - y_p^{[3]}| \right) \]  
(9)

If \( E_m > E_t \), set \( E_t = E_t - \delta_t \), \( k = 1 \) and go to Step 3. If \( E_m \leq E_t \), then the learning procedure is finished.

4. Simulation results

The proposed FNN was simulated on a 486-p.c. (33 MHz) using the C language (Turbo C version 2.0). Two data sets of training samples (see Tables 1 and 2), in
which the input and output values of each sample are known, were used for training. The input values of the training samples are distributed between 0:00 and 1:00. In the simulation experiments, different values of $\alpha$, $E_r$ and $\delta_r$ were used. The training procedure for all the experiments were finished in one epoch and the training times were about 0:03 CPU seconds and about 0:06 CPU seconds, respectively, for the 10 samples in Table 1 (Data Set 1) and the 20 samples in Table 2 (Data Set 2). The FNN can learn the membership functions from training samples. By scanning the input $x$ from 0:00 to 1:00 at intervals of 0:01, we can get the complete estimated membership functions. The recalling time for these 101 points was about 0:12 CPU seconds. Figures 2 and 3 give the estimated membership functions when the FNN was trained by Data Sets 1 and 2, respectively, with different $\alpha$, $E_r$ and $\delta_r$. In Figs 2 and 3, $N$, $M$ and $P$ represent, respectively, the numbers of FNs in the first, second and third layers.

From the simulation results, we observe that: (a) if more training samples are learned, the estimated functions will be smoother; (b) the smaller the $E_r$ is, the more hidden FNs are needed but the number of the hidden FNs will not exceed the number of training samples; (c) an increase in the value of $\alpha$ will help to smooth out the shape of the estimated membership function, but it may increase the number of hidden FNs.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$1$</th>
<th>$2$</th>
<th>$3$</th>
<th>$4$</th>
<th>$5$</th>
<th>$6$</th>
<th>$7$</th>
<th>$8$</th>
<th>$9$</th>
<th>$10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{1k}$</td>
<td>0:10</td>
<td>0:20</td>
<td>0:30</td>
<td>0:40</td>
<td>0:50</td>
<td>0:60</td>
<td>0:70</td>
<td>0:80</td>
<td>0:90</td>
<td>1:00</td>
</tr>
</tbody>
</table>
| \hline
| Class 1 | $d_{1k}$ | 0:10 | 0:40 | 0:10 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 |
| Class 2 | $d_{2k}$ | 0:10 | 0:15 | 0:70 | 0:90 | 0:70 | 0:15 | 0:10 | 0:00 | 0:00 |
| Class 3 | $d_{3k}$ | 0:00 | 0:08 | 0:15 | 0:30 | 0:40 | 0:80 | 1:00 | 0:80 | 0:40 |
| Class 4 | $d_{4k}$ | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:10 | 0:40 | 0:70 | 0:40 |

Table 1. Training Data Set 1 (10 training samples).

<table>
<thead>
<tr>
<th>$k$</th>
<th>$1$</th>
<th>$2$</th>
<th>$3$</th>
<th>$4$</th>
<th>$5$</th>
<th>$6$</th>
<th>$7$</th>
<th>$8$</th>
<th>$9$</th>
<th>$10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{1k}$</td>
<td>0:05</td>
<td>0:15</td>
<td>0:15</td>
<td>0:20</td>
<td>0:25</td>
<td>0:30</td>
<td>0:35</td>
<td>0:40</td>
<td>0:45</td>
<td>0:50</td>
</tr>
</tbody>
</table>
| \hline
| Class 1 | $d_{1k}$ | 0:05 | 0:10 | 0:30 | 0:40 | 0:30 | 0:10 | 0:05 | 0:00 | 0:00 |
| Class 2 | $d_{2k}$ | 0:00 | 0:10 | 0:12 | 0:15 | 0:28 | 0:70 | 0:82 | 0:90 | 0:82 |
| Class 3 | $d_{3k}$ | 0:00 | 0:00 | 0:00 | 0:08 | 0:10 | 0:15 | 0:20 | 0:30 | 0:34 |
| Class 4 | $d_{4k}$ | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 |

<table>
<thead>
<tr>
<th>$k$</th>
<th>$11$</th>
<th>$12$</th>
<th>$13$</th>
<th>$14$</th>
<th>$15$</th>
<th>$16$</th>
<th>$17$</th>
<th>$18$</th>
<th>$19$</th>
<th>$20$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{1k}$</td>
<td>0:55</td>
<td>0:60</td>
<td>0:65</td>
<td>0:70</td>
<td>0:75</td>
<td>0:80</td>
<td>0:85</td>
<td>0:90</td>
<td>0:95</td>
<td>1:00</td>
</tr>
</tbody>
</table>
| \hline
| Class 1 | $d_{1k}$ | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 |
| Class 2 | $d_{2k}$ | 0:28 | 0:15 | 0:12 | 0:10 | 0:00 | 0:00 | 0:00 | 0:00 | 0:00 |
| Class 3 | $d_{3k}$ | 0:55 | 0:80 | 0:93 | 1:00 | 0:93 | 0:80 | 0:55 | 0:40 | 0:34 |
| Class 4 | $d_{4k}$ | 0:00 | 0:00 | 0:05 | 0:10 | 0:22 | 0:40 | 0:62 | 0:70 | 0:62 |

Table 2. Training Data Set 2 (20 training samples).
Figure 2. Estimated membership function when trained by Data Set 1: (a) $\alpha=0.4$, $E_t=0.1$, $\delta_t=0.02$, $N=1$, $M=9$, $P=4$; (b) $\alpha=0.6$, $E_t=0.05$, $\delta_t=0.01$, $N=1$, $M=10$, $P=4$.

5. Conclusions

A novel fuzzy neural network is proposed by combining the structure and learning ability of ANNs, and the fuzzy classification ability of fuzzy algorithms. The resultant FNN is capable of adaptively organizing its hidden layer, and also capable of learning the membership function of each fuzzy class directly from training samples. The learning and recalling of the FNN are fast. Simulation results on the learning and estimating of the membership functions of a classification problem have demonstrated the uses of the proposed FNN. The minimum and maximum operations of the second and third layers of the proposed FNN can be

Figure 3. Estimated membership function when trained by Data Set 2: (a) $\alpha=0.3$, $E_t=0.1$, $\delta_t=0.02$, $N=1$, $M=17$, $P=4$; (b) $\alpha=0.4$, $E_t=0.05$, $\delta_t=0.01$, $N=1$, $M=19$, $P=4$. 

Copyright © 2001. All Rights Reserved.
implemented efficiently using the digital method (with a minor hardware addition to allow the actual minimum or maximum value to be output) described in a recent paper (Kwan 1992). The proposed FNN can be also used in applications such as a fuzzy controller or other fuzzy knowledge-base systems where the FNN is required to learn fuzzy inference rules, define the termset or modify the termset shape of fuzzy inference rules from: (a) observed operator responses; (b) from experimental input-output data; or (c) from input-output data gathered from specialists.

ACKNOWLEDGMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and in part by the Micronet.

REFERENCES