Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 483535, 21 pages
doi:10.1155/2012/483535

*Research Article*

# An Adaptive Fuzzy Min-Max Neural Network Classifier Based on Principle Component Analysis and Adaptive Genetic Algorithm

**Jinhai Liu, Zhibo Yu, and Dazhong Ma**

*School of Information Science and Engineering, Northeastern University, Shenyang 110004, China*

Correspondence should be addressed to Jinhai Liu, jh_lau@126.com

Received 31 August 2012; Accepted 25 October 2012

Academic Editor: Bin Jiang



A novel adaptive fuzzy min-max neural network classifier called AFMN is proposed in this paper. Combined with principle component analysis and adaptive genetic algorithm, this integrated system can serve as a supervised and real-time classification technique. Considering the loophole in the expansion-contraction process of FMNN and GFMN and the overcomplex network architecture of FMCN, AFMN maintains the simple architecture of FMNN for fast learning and testing while rewriting the membership function, the expansion and contraction rules for hyperbox generation to solve the confusion problems in the hyperbox overlap region. Meanwhile, principle component analysis is adopted to finish dataset dimensionality reduction for increasing learning efficiency. After training, the confidence coefficient of each hyperbox is calculated based on the distribution of samples. During classifying procedure, utilizing adaptive genetic algorithm to complete parameter optimization for AFMN can also fasten the entire procedure than traversal method. For conditions where training samples are insufficient, data core weight updating is indispensible to enhance the robustness of classifier and the modified membership function can adjust itself according to the input varieties. The paper demonstrates the performance of AFMN through substantial examples in terms of classification accuracy and operating speed by comparing it with FMNN, GFMN, and FMCN.

## 1. Introduction

The merge of fuzzy set theory [1–6] stimulates its development on pattern recognition and classification. The capacity for fuzzy logic to divide the complex class boundaries has generated a lot of achievements in neuro-fuzzy pattern recognition systems [7–23]. The fuzzy min-max neural network (FMNN) which is proposed in [24] puts a solid foundation for further research in this field. The FMNN utilizes hyperbox fuzzy sets to represent a region of the n-dimensional pattern space; input samples which fall in a hyperbox have full

memberships. An n-dimensional hyperbox can be defined by stating its min and max vertices. This algorithm is to find suitable hyperboxes for each input patterns with a three-step process: expansion, overlap, and contraction. But the contraction of hyperboxes of different classes may lead to classification error which is demonstrated in [25], and its performance highly depends on the initialization of the sequence of the training data and the expansion coefficient which controls the size of hyperbox.

The proposed GFMN [26] is also an online classifier based on hyperbox fuzzy set concept. Its improvement lies in proposing a new membership function which monotonically decreases with a growing distance from a cluster prototype, thus eliminating the likely confusion between cases of equally likely and unknown inputs [26]. But the contraction process problem remains. This situation is the same with the proposal of a modified fuzzy min-max neural network with a genetic-algorithm-based rule extractor for pattern classification even though it is creative to use genetic algorithm to minimize the numbers of features of input dataset [27]. In FMCN [25], a new learning algorithm called fuzzy min-max neural network classifier with compensatory neuron architecture (FMCN) architecture has been reported. This method introduces compensatory neurons to handle the confusion in overlap regions and disposal of the contraction process. However, the FMCN does not allow hyperboxes of different class to be overlapped which results in the increasing number of neurons in the middle layer of network, thus consuming much more time during training and testing. And the algorithm distinguishes the simple overlap and containment. In fact, even though FMCN performs better than FMNN and GFMN in most cases, its structural complexity increases and consumes more time during training and testing. Meanwhile, it omits a kind of overlap [28] which results in classification error. Another improved network based on data core is called data-core-based fuzzy min-max neural network (DCFMN). DCFMN [28] can adjust the membership function according to samples distribution in a hyperbox to get a higher classification accuracy, and its structural is simpler than FMCN. However, all these four networks cannot perform well with relatively insufficient training samples. A weighted fuzzy min-max neural network (WFMM) is proposed in [29]. The membership function of WFMM is designed to take the frequency of input patterns into consideration.

The proposed AFMN owns its advantages in several aspects. First, the proposed AFMN maintains the simple architecture of FMNN and adds preprocessing for input patterns, and its technique is principle component analysis (PCA) [30]. This kind of data dimensionality reduction technique can reduce the number of features of input patterns and extract the useful information. It is known that without preprocessing of training dataset, it is hard to practically implement the classifier due to the high dimensionality, the redundancy, and even noise inherent in input patterns.

Second, considering that there are nodes of more than one class in a hyperbox, it is not reasonable to allocate full membership for any input pattern that falls in the hyperbox. So the confidence coefficient for each hyperbox is introduced for resolving this confusion to achieve a higher classification accuracy.

Third, membership function is modified according to the inspiration of data core from DCFMN. The concept of data core which can update itself during testing aids to adjust the membership based on the training samples distribution. And loopholes that existed in FMNN, GFMN, and FMCN overlap test cases are found out and resolved by rewriting the rules. Meanwhile, adaptive genetic algorithm (AGA) [31–33] is utilized in classifying algorithm for parameters optimization instead of traversal method to improve the speed and accuracy of the entire neural network classifier.

Finally, the proposal of this new classifier is not only a original attempt of theory, but also an important initial step for its application on the running pipeline for working condition recognition which is a typical nonlinear control system [34–39].

The rest of the paper is organized as follows. Section 2 analyzes the traditional fuzzy neural network classifier. Section 3 introduces the AFMN classifier system in detail. Section 4 provides abundant examples to demonstrate the performance of AFMN. Section 5 concludes with summary.

## 2. Analysis of Precedent Fuzzy Min-Max Neural Network Classifier

FMNN learning algorithm consists of three procedures: (1) expansion, (2) overlap test, and (3) contraction. Its rule is to find a suitable hyperbox for each input pattern. If the appropriate hyperbox exists (even after expansion), its size cannot exceed the minimum and maximum limits. After expansion, all hyperboxes that belong to different classes have to be checked by overlap test to determine if any overlap exists. So a dimension by dimension comparison between hyperboxes of different class is performed. FMNN designs four test cases, at least one of the four cases is satisfied, then overlap exists between the two hyperboxes. Otherwise, a new hyperbox needs to be added to the network. If no overlaps occur, the hyperboxes are isolated and no contraction is required. Otherwise, a contraction process is needed to eliminate the confusion in overlapped areas.

GFMN focuses on the disadvantages of the membership function proposed in FMNN and proposes an improved membership function that the membership value can decrease steadily when input patterns get far away from the hyperbox.

FMCN distinguishes the simple overlap and containment and introduces overlapped compensation neurons (OCNs) and containment compensation neurons (CCNs) to solve the confusion in the overlap region.

However, there exists two cases in the overlap area that FMNN, GFMN, and FMCN cannot operate properly on the hyperbox adjustment. Figure 1 depicts the two hyperboxes overlap cases. The positions of minimum and maximum points are described below:

$$\begin{aligned} v_{k1} = v_{j1} < w_{j1} < w_{k1}, \\ v_{k1} = v_{j1} < w_{j1} = w_{k1}. \end{aligned} \tag{2.1}$$

When input data that satisfies this condition is trained according to the overlap test rules designed in FMNN, GFMN, and FMCN, overlap cannot be checked because they do not satisfy any one of the four cases in overlap test. However, obviously the two hyperboxes are partly overlapped in Figure 1(a), and the other two hyperboxes are fully overlapped in Figure 1(b). This case shows that the loophole exists in the overlap test case of the three algorithms. Especially in the case depicted in Figure 1(b), the network cannot cancel one of the two identical hyperboxes, which means creating the same hyperbox twice. Meanwhile, the number of nodes will increase if overlap occurs between two hyperboxes of the same class and increase the computation complexity. Figure 2 emphasizes this situation again, there should be four hyperboxes after training, but the overlap test regards the number of hyperboxes as five. Just as precedent discussion shows, the cases in the overlap are not complete and need revising.

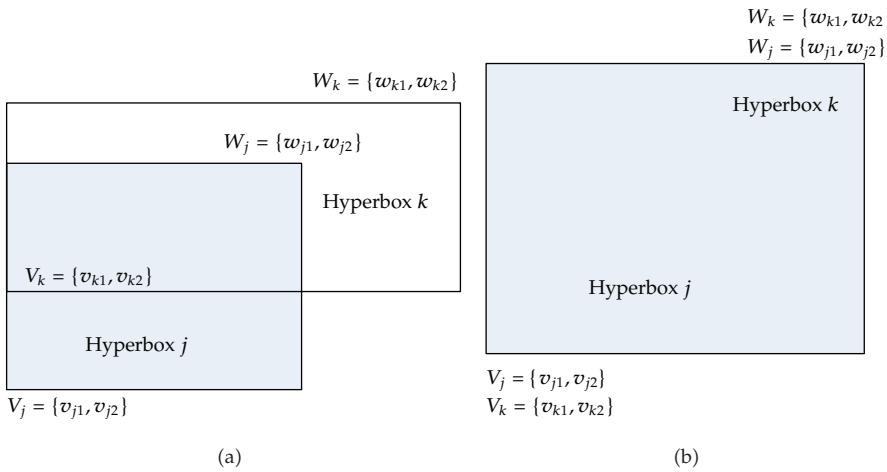



**Figure 1:** Two overlap cases.

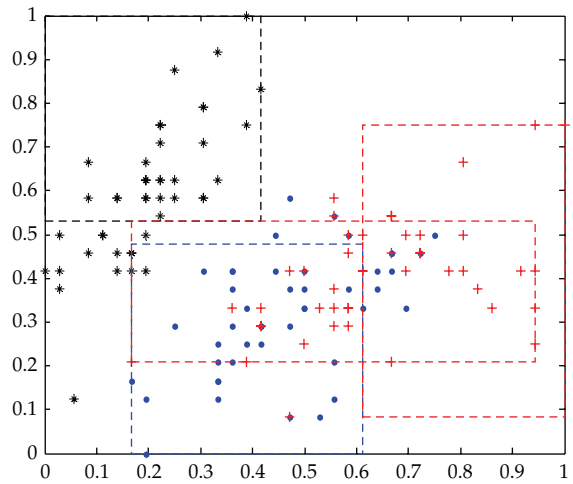
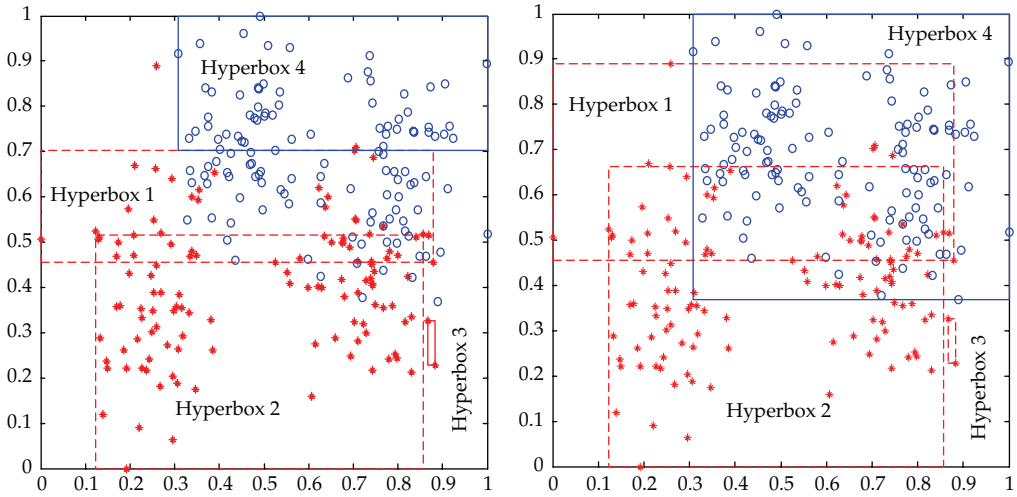



**Figure 2:** A confusing hyperbox generation.

Another disadvantage of the traditional classifier and the same with DCFMN is that they do not take verifying the efficiency of a hyperbox into consideration.

The idea of testing the efficiency of a hyperbox is inspired by the situation that in a hyperbox there are input patterns of more than one class. For the convenience of explanation here we name input patterns of the certain class that its hyperbox belongs to as primary patterns (PPs) and those of any other class as subordinate patterns (SPs). Figure 3 shows the hyperboxes generated according to the learning algorithm of FMNN and DCFMN. Among them, hyperboxes 1–3 belong to class 1 and hyperbox 4 belongs to class 2. We can notice that in hyperbox1 of class 1, there are more SPs than PPs which shows that the creation of hyperbox is not appropriate and may insert a negative impact in classification.

Meanwhile, in other traditional fuzzy min-max neural network classifiers, input data is not preprocessed before training. The redundancy and noise of data can undermine the performance of classification and consume more time during training and testing.




(a) Hyperboxes generated by FMNN (b) Hyperboxes generated by DCFMN

**Figure 3:** Two inappropriate hyperbox creation.

In AFMN, the problem is solved by using principle component analysis (PCA) to reduce the dimensionality of input data and adopting genetic algorithm to fast select the optimal parameters combination during test procedure instead of traversal method.

## 3. AFMN Architecture

### 3.1. Basic Definitions

#### 3.1.1. Confidence Coefficient

The hyperboxes generated during training are in different sizes and the input patterns included a hyperbox may belong to different classes which means the hyperbox cannot guarantee that an input pattern that falls within it fully belongs to its class. Figure 4 shows a hyperbox creation result in which there are input patterns of three classes A, B, and C. Obviously it is not rational to regard the membership of all input patterns that fall in the hyperbox B as 1 because there are PPs and OPs at the same time in the same hyperbox. This problem can be removed by accounting for the proportion of PPs patterns to total patterns in the same hyperbox. By calculating the proportion of the PPs of total patterns in the same box, the confidence coefficient of each hyperbox can be gotten. Let $H = \{\eta_1, \eta_2, \ldots\}$, $\eta_k$ be the confidence coefficient of $k$th hyperbox.

Two possibilities have to be considered when designing $H$.

(1) Just like the discussion before, we name input patterns of the certain class that its hyperbox belongs to as primary patterns (PPs) and those of any other class as subordinate patterns (SPs), we should should consider the portion of PPs to total patterns and that of PP and SP patterns.

(2) If the amount of training data of different classes is different, for eliminating the training error caused by this imbalance inherent in the samples, normalizing

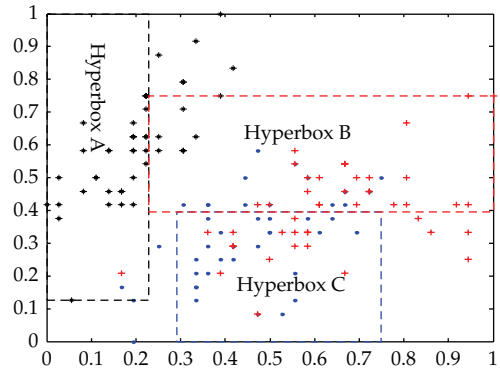



**Figure 4:** The Meaning of Confidence Coefficient.

initially the patterns by introducing parameter $\xi$ is necessary which means distributing a weight value for each class.

And the resolution consists of two steps.

*Step 1.* Compute Weight Value $\xi$ for Each Class.

For there are $p$ classes, the number of input patterns for each class is relatively $\varphi_1$, $\varphi_2, \ldots, \varphi_p$; the function that decides weight value $\xi_k$ for each class is given by

$$\xi_k = \frac{\varphi_k}{\max(\varphi_1, \varphi_2, \ldots, \varphi_p)}, \quad k = 1, 2 \ldots p. \tag{3.1}$$

*Step 2.* Compute Confidence Coefficient for Each Hyperbox.

For the $j$th hyperbox $b_j$ and $b_j \in c_k$, the corresponding $\eta_j$ is given by.

$$\eta_j = \frac{\phi_{jk}/\xi_k}{\left(\phi_{j1}/\xi_1 + \phi_{j2}/\xi_2 + \cdots + \phi_{jk}/\xi_k + \cdots + \phi_{jp}/\xi_p\right)}, \tag{3.2}$$

where $\phi_{jk}(k = 1, 2, \ldots, p)$ represents the number of input patterns of class $k$ in hyperbox $b_j$, $j = 1, 2, \ldots m$; $m$ is the number of hyperboxes. And the value of $\eta_j$ is decided by

$$\eta_j = \begin{cases} \eta_j & \eta_j > \beta, \\ 0 & \eta_j \leq \beta, \end{cases} \tag{3.3}$$

where $\beta$ ranges from 0.1 to 1.

### 3.1.2. AFMN Architecture Overview

The architecture of AFMN is shown in Figure 5. The connections between input and middle layer are stored in matrices $V$ and $W$. The connections between middle and the output layer

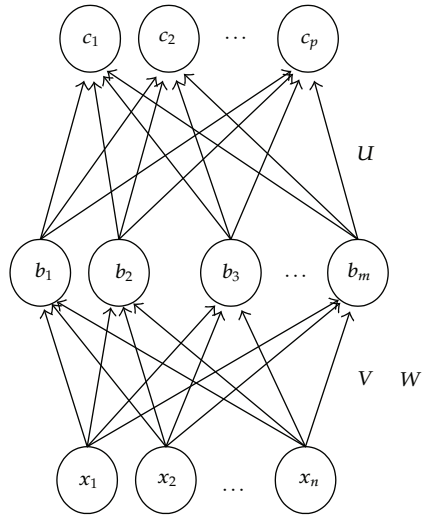



**Figure 5:** AFMN architecture overview.

are binary valued and stored in $U$. The equation for assigning the values from $b_j$ to the output layer node $c_i$ is as follows:

$$u_{ji} = \begin{cases} 1, & \text{if } b_o \in c_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

### 3.2. *Fuzzy Hyperbox Membership Function*

The membership function $b_j(X_h)$ for an input $X_h$ is given by

$$\begin{aligned} b_j(X_h) &= 1 - \max_{i=1\cdots n}\left(\max\left(f\left(\alpha_{1i}, \beta_i\right), f\left(\alpha_{2i}, \beta_i\right)\right)\right), \\ \alpha_{1i} &= x_{hi} - w_{ji}, \\ \alpha_{2i} &= v_{ji} - x_{hi}, \\ \beta_i &= \gamma\left(1 - d_{ji}\right)\lambda_j, \end{aligned} \tag{3.5}$$

where $d_{ji} = c_{ji} - ((v_{ji} + w_{ji})/2)$; $c_{ji}$ is the geometrical core that is known as data core. It is given by

$$c_{ji} = \frac{1}{j_N} \sum_{q=1}^{j_N} x_{hi}^q \quad v_{ji} \le c_{ji} \le w_{ji}, \tag{3.6}$$

where $j_N$ is the number of patterns belonging to its hyperbox's class. $x_{hi}^q$ is the patterns belonging to its hyperbox's class.

$\lambda$ is given by

$$\lambda_j = \frac{\max \phi}{\phi_j}, \quad j = 1, \ldots, m, \tag{3.7}$$

where $\phi_j$ indicates the number of PPs in the hyperbox $j$. $f$ is a two-parameter ramp threshold function as follows

$$f(\alpha_{oi}, \beta_i) = \begin{cases} 1, & \alpha_{oi}\beta_i > 1, \\ \alpha_{oi}\beta_i, & 1 \geq \alpha_{oi}\beta_i \geq 0, \quad \text{o} = 1, 2, \\ 0, & \alpha_{oi}\beta_i < 0. \end{cases} \tag{3.8}$$

## 3.3. Learning Algorithm

### 3.3.1. Data Preprocessing by Principle Component Analysis (PCA)

Principle analysis is chosen as a data dimensionality reduction technique that removes redundant features from the input data. The input data after dimensionality reduction can accelerate the training and testing procedure meanwhile improving the network performance because PCA picks up primary features from original dataset to avoid affecting by the redundancy and noise within it. In this paper, the number of features we choose depends on the how many dimensions can include 80% of the total information.

### 3.3.2. Hyperbox Expansion

This procedure decides the number and min-max points of hyperboxes, its rule is as follows.

If the following criterion is satisfied

$$n\theta \geq \sum_{i=1}^{n} \left(\max\left(w_{ji}, x_{hi}\right) - \min\left(v_{ji}, x_{hi}\right)\right), \tag{3.9}$$

where $\theta$ controls the size of a hyperbox $0 < \theta \leq 1$.

If the expansion criterion has been met, the minimum and maximum points of the hyper box are adjusted using the following equation

$$\begin{aligned} v_{ji}^{\text{new}} &= \min\left(v_{ji}^{\text{old}}, x_{hi}\right), \quad i = 1, 2, \ldots, n, \\ w_{ji}^{\text{new}} &= \max\left(w_{ji}^{\text{old}}, x_{hi}\right), \quad i = 1, 2, \ldots, n. \end{aligned} \tag{3.10}$$

Otherwise, create a new hyperbox and its min and max points are adjusted as below:

$$\begin{aligned} v_{ji}^{\text{new}} &= x_{hi}, \quad i = 1, 2, \ldots, n, \\ w_{ji}^{\text{new}} &= x_{hi}, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{3.11}$$

Repeat the procedure until all the input patterns finish training.

### 3.4. Hyperbox Overlap Test

As previously stated, new cases have to resolve the problem existed in FMNN; so first for testing if two hyperboxes are fully overlapped, we design the case as bellow:

$$v_{ji} = v_{ki} < w_{ji} = w_{ki}. \tag{3.12}$$

If the case can be satisfied, that means two hyperboxes of the same class fully overlap, then one of them will be removed from the network.

Here assuming $\alpha^{\text{old}} = 1$, $\Delta = 1$ initially, for hyperbox $j$ and hyperbox $k$, the four overlap cases and the corresponding overlap value for the $i$th dimension are given as follows.

*Case 1* ($v_{ji} \le v_{ki} < w_{ji} \le w_{ki}$). One has

$$\alpha^{\text{new}} = \min\left(w_{ji} - v_{ki}, \alpha^{\text{old}}\right). \tag{3.13}$$

*Case 2* ($v_{ki} \le v_{ji} < w_{ki} \le w_{ji}$). One has

$$\alpha^{\text{new}} = \min\left(w_{ki} - v_{ji}, \alpha^{\text{old}}\right). \tag{3.14}$$

*Case 3* ($v_{ji} < v_{ki} < w_{ki} < w_{ji}$). One has

$$\alpha^{\text{new}} = \min\left(\min\left(w_{ki} - v_{ji}, w_{ji} - v_{ki}\right), \alpha^{\text{old}}\right). \tag{3.15}$$

*Case 4* ($v_{ki} < v_{ji} < w_{ji} < w_{ki}$). One has

$$\alpha^{\text{new}} = \min\left(\min\left(w_{ji} - v_{ki}, w_{ki} - v_{ji}\right), \alpha^{\text{old}}\right). \tag{3.16}$$

If $\alpha^{\text{old}} - \alpha^{\text{new}} > 0$, then $\Delta = i_{\circ}$. If any dimension cannot satisfy any of the four cases, then $\Delta = 0$. Otherwise if $\Delta \neq 0$, then there is overlap between hyperbox $j$ and hyperbox $k$.

### 3.5. Hyperbox Contraction

If overlap exists between hyperboxes of different classes, the network will allocate 1 for the overlap region, thus generating the classification confusion. And only one of the $n$ dimension needs to be adjusted to keep the hyperbox as large as possible. For $\Delta = i$, then $\Delta$th dimension is that we should select. The adjustment should be made as follows.

*Case 1* ($v_{j\Delta} \le v_{k\Delta} < w_{j\Delta} \le w_{k\Delta}$). One has

$$v_{k\Delta}^{\text{new}} = w_{j\Delta}^{\text{new}} = \frac{v_{k\Delta}^{\text{old}} + w_{j\Delta}^{\text{old}}}{2}. \tag{3.17}$$

*Case 2* ($v_{k\Delta} \le v_{j\Delta} < w_{k\Delta} \le w_{j\Delta}$). One has

$$v_{j\Delta}^{\text{new}} = w_{k\Delta}^{\text{new}} = \frac{v_{j\Delta}^{\text{old}} + w_{k\Delta}^{\text{old}}}{2}. \tag{3.18}$$

*Case 3* ($v_{j\Delta} \le v_{k\Delta} < w_{k\Delta} \le w_{j\Delta}$). One has

$$\begin{aligned} \text{if} \quad & w_{k\Delta} - v_{j\Delta} < w_{j\Delta} - v_{k\Delta} \\ \text{then} \quad & v_{j\Delta}^{\text{new}} = w_{k\Delta}^{\text{old}} \\ \text{otherwise} \quad & w_{j\Delta}^{\text{new}} = v_{k\Delta}^{\text{old}}. \end{aligned} \tag{3.19}$$

*Case 4* ($v_{k\Delta} \le v_{j\,\triangle} < w_{j\Delta} \le w_{k\Delta}$). One has

$$\begin{aligned} \text{If} \quad & w_{k\Delta} - v_{j\Delta} < w_{j\Delta} - v_{k\Delta} \\ \text{then} \quad & w_{k\Delta}^{\text{new}} = v_{j\Delta}^{\text{old}} \\ \text{otherwise} \quad & v_{k\Delta}^{\text{new}} = w_{j\Delta}^{\text{old}}. \end{aligned} \tag{3.20}$$

Through all the precedent procedures, parameters $V$ and $W$ are determined. The entire learning procedure can be summarized in Figure 6.

## 3.6. Classifying Algorithm

### 3.6.1. Genetic Algorithm in Network Classifying Procedure

GA is bestowed the task of finding the best parameter combination instead of the traversal method. Compared with the traditional traversal method to search for appropriate parameters for the network to achieve its best performance, genetic algorithm has two advantages.

(a) For traversal method, choosing an appropriate step is an obstacle. Setting too small step size can achieve a better classification performance at the cost of more time consuming. Otherwise, testing procedure will be fast at the cost of a relatively low accuracy.

(b) For high classification accuracy that means setting the step short. Genetic algorithm completes this task faster than traversal.

The GA fitness function used is defined as

$$f = \frac{1}{N_{\text{misc}}}. \tag{3.21}$$

The genetic operation implemented consists of the following six steps.

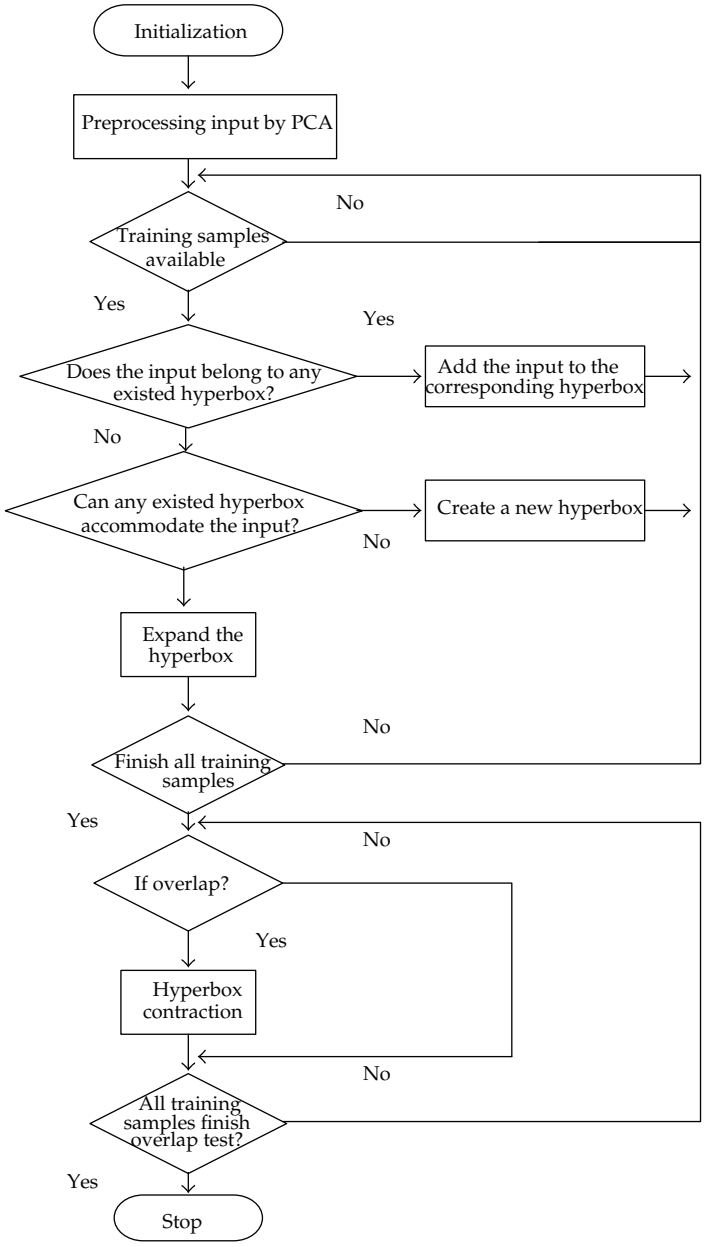



**Figure 6:** AFMN learning algorithm.

*Step 1* (initialization). Set the range for each parameter and initialize the population string in each generation. Here $\theta$ ranges from 0 to 1, $\beta$ ranges from 0.1 to 1, and $\gamma$ ranges from 1 to 10.

*Step 2* (selection). Select the certain numbers of pairs of strings from the current population according to the rule known as roulette wheel selection.

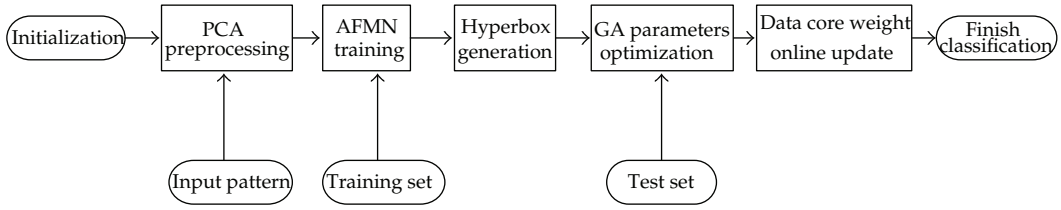



**Figure 7:** AFMN classifying algorithm.

*Step 3* (crossover). For each selected pair, choose the bit position for crossover. The rule is specified as bellow:

$$P_c = \begin{cases} 0.8 - \dfrac{f' - f_{\text{avg}}}{f_{\max} - f'}, & f' > f_{\text{avg}}, \\ 0.8, & f' < f_{\text{avg}}, \end{cases} \tag{3.22}$$

where $f'$ indicates the lager fitness value in the pair, $f_{\max}$ is the maximum fitness value, and $f_{\text{avg}}$ is the average fitness value of the current population.

*Step 4* (mutation). For each bit value of the strings, apply the following mutation operation according to the possibility defined as below:

$$P_m = \begin{cases} 0.1 * \dfrac{f_{\max} - f}{f_{\max} - f_{\text{avg}}}, & f > f_{\text{avg}}, \\ 0.1 + 0.2 * \dfrac{f_{\max} - f}{f_{\max} - f_{\text{wst}}}, & f < f_{\text{avg}}, \end{cases} \tag{3.23}$$

where $f$ is the fitness value of the mutation individual.

*Step 5* (elitist strategy). Select a string with maximum fitness and pass it to the next generation directly.

*Step 6* (termination test). Here we use the number of generations as a condition for genetic algorithm termination.

### *3.6.2. The Entire Classifier System*

The learning and classification algorithm can be summarized in the flowchart in Figure 7.

## 4. Results

### *4.1. Examples to Demonstrate the Effectiveness of Overlap Test and Contraction Cases*

Just as the previous discussion about the cases represented in Figures 1(a) and 1(b). When overlap occurs in such case, the overlap and contraction algorithm of FMNN, GFMN,

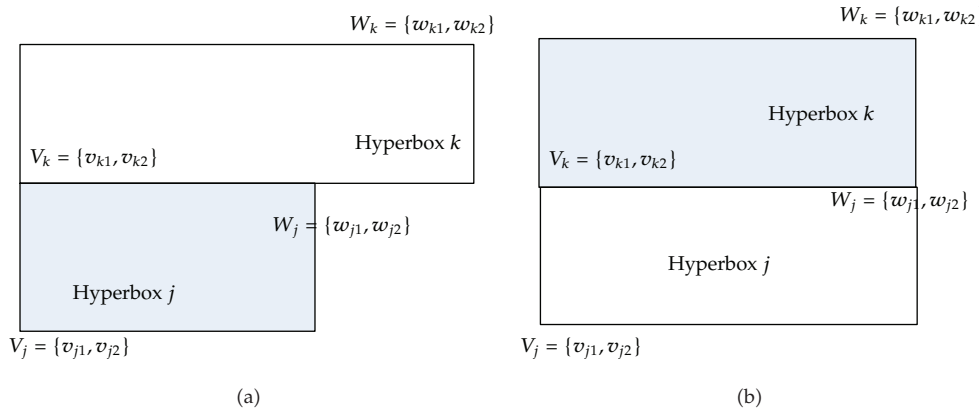



**Figure 8:** The correct hyperbox division by AFMN.

**Table 1:** The effectiveness of PCA process.

| Training dataset (%) | 20% | 40% | 60% | 80% | 100% |
|---|---|---|---|---|---|
| Error numbers without PCA/time consumed (s) | 35/20.45 | 25/29.16 | 20/39.78 | 21/60.78 | 9/80.49 |
| Error numbers with PCA/time consumed (s) | 33/18.15 | 26/24.78 | 20/30.46 | 18/50.01 | 11/69.48 |

and FMCN will create misclassification error. This problem is solved by the revised overlap test cases. The hyperbox generation result is shown in Figures 8(a) and 8(b).

## *4.2. The Working of PCA and Genetic Algorithm*

### *4.2.1. Principle Component Analysis*

To understand the effect of PCA in improving classification efficiency by implementing data dimensionality reduction, we use AFMN to classify five groups of complete GLASS dataset [40], and one is preprocessed by PCA to get a simplified input pattern and guarantee the remaining formation is not less than 80%. The number of classification error is shown in Table 1. The training dataset ranges from 20% to 100%. The training dataset is selected randomly each time, and the entire glass dataset is used for testing. The experiment is conducted 100 times. The result is represented in Table 1.

From the results in the table, it is demonstrated that principle analysis can complete the task of dimensionality reduction, and it is important to notice that adding PCA is not bound to increase the classification accuracy which is verified in 40% and 100% training set. But thanks to its ability of reducing the dimensionality of the raw dataset, the consuming time has been shorten rapidly.

### *4.2.2. Genetic Algorithm for Parameter Optimization*

The task of genetic algorithm is to find the appropriate combination of three parameters for best classification performance faster. And the result with genetic algorithm should be no worse than without it. Here Iris dataset is chosen for demonstration, 10% of the given dataset is for training and the rest for testing. The experiment is repeated 100 times to get

**Table 2:** Parameter optimization using GA.

| | With GA | Without GA |
|---|---|---|
| Minimum error number | 5 | 6 |
| Average consuming time (s) | 0.9832 | 1.4976 |

**Table 3:** Iris data ($\theta = 0.2$).

| Data (%) | AFMN | | FMCN | | GFMN | | FMNN | |
|---|---|---|---|---|---|---|---|---|
| | LE | TE | LE | TE | LE | TE | LE | TE |
| 35 | 1.92 | 5.10 | 3.84 | 7.14 | 7.69 | 9.18 | 11.53 | 14.29 |
| 45 | 0 | 3.61 | 2.98 | 6.02 | 8.96 | 8.43 | 10.45 | 12.05 |
| 50 | 0 | 2.67 | 0 | 4.0 | 1.33 | 5.33 | 6.67 | 12 |
| 60 | 0 | 1.67 | 0 | 3.33 | 3.33 | 5 | 4.44 | 10 |
| 70 | 0 | 0 | 0 | 2.22 | 1.9 | 4.44 | 4.76 | 8.89 |

LE: learning error; TE: testing error.

the minimum misclassification numbers and the average consuming time. The result is shown in Table 2.

Table 2 demonstrates that GA can find better combination of parameters and its speed is faster. Its ability is important for application in real world.

## 4.3. Performance on the General Dataset

### 4.3.1. Various Dataset for Training and Testing with Complete Dataset

Here for the given iris dataset, 35%, 45%, 50%, 60%, and 70% of the dataset were selected randomly for training purpose and the complete dataset for testing. The performance of learning and testing is shown in Table 3. It is obvious that AFMN has a better performance with fewer misclassifications.

### 4.3.2. Different Dataset for Training and Testing

In this section, datasets such as wine, thyroid, and ionosphere are selected for comparing the abilities of several network classifier (Table 4). 50% of each dataset is randomly selected for training and the entire dataset for testing. We conduct 100 times experiment for each dataset. Results show, in terms of classification accuracy, that the FMCN and AFMN have the very approximate performance, but from the consuming time and stability of these two classifiers, obviously AFMN is better than FMCN which demonstrates its advantage (Table 5).

## 4.4. Test the Robustness of AFMN with Noise-Contaminated Dataset

The robustness of a network classifier is important especially in application. 50% of Iris data was randomly chosen for training, and the entire Iris dataset was used for testing. For the purpose of checking robustness of AFMN, FNCN, FMNN, and GFMN, We added the random noise to the Iris data set. The amplitude of noise added in the Iris data set is 1%, 5%, and 10%. The expansion coefficient varies from 0.01 to 0.4 in step of 0.02. One hundred

**Table 4:** AFMN performance on different datasets.

| Dataset | AFMN | | | | FMCN | | | | GFMN | | | | FMNN | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | misclassification (%) | | | | misclassification (%) | | | | misclassification (%) | | | | misclassification (%) | | | |
| | Min | Max | Ave | Std | Min | Max | Ave | Std | Min | Max | Ave | Std | Min | Max | Ave | Std |
| Thyroid | 0.47 | 4.19 | 2.28 | 0.35 | 0.47 | 4.19 | 2.32 | 0.39 | 0.47 | 4.19 | 2.27 | 0.60 | 0.93 | 4.19 | 2.41 | 0.47 |
| Wine | 0 | 2.93 | 1.67 | 0.41 | 0 | 2.81 | 1.65 | 0.49 | 0 | 3.37 | 1.52 | 0.89 | 0.56 | 5.06 | 2.05 | 0.80 |
| Ionosphere | 9.97 | 12.54 | 12.49 | 1.15 | 9.97 | 13.68 | 13.26 | 1.24 | 9.69 | 13.96 | 12.08 | 1.88 | 6.27 | 8.26 | 7.39 | 1.44 |

**Table 5:** Classification consuming time of FMCN and AFMN.

| Consuming time/per trial | FMCN/s | AFMN/s |
|---|---|---|
| Thyroid | 25.312 | 4.145 |
| Wine | 18.145 | 7.461 |
| Ionosphere | 265.28 | 43.19 |

experiments were performed for getting accuracy result. The result is shown in Table 6, when the amplitude of noise is 1%, the maximum and minimum misclassification of four methods is the same with the numbers of experiment with precise data. It proves that all the methods have robustness. But as the amplitude of noise increases, the performance of four methods becomes worse. Although the performance becomes worse, from Table 6, the average misclassification in AFMN increases more slowly than others, and the AFMN has better robustness.

### *4.5. Fixed Training Dataset Size (60% of Iris Dataset)*

In this simulation, the effect of expansion coefficient is studied on the performance of AFMN, FMCN, GFMN, and FMNN. 60% of iris data is chosen for training and the entire iris data for testing. The expansion coefficient varies form 1.0 to 1 in step of 0.1. The results of training and testing are shown in Figures 9 and 10, respectively.

From the result we can conclude, that FMCN is vulnerable to the fluctuation of expansion coefficient, and GFMN and FMNN have a relatively higher classification error. Compared with them, AFMN performs better.

### *4.6. Test on Synthetic Image*

The dataset consists of 950 samples belonging to two nested classes which make the classification more difficult. Figure 11 shows the synthetic image.

Figure 12 shows the performance of AFMN, FMCN, GFMN, and FMNN on this specified data set. 60% of dataset is randomly selected for training. Expansion coefficient varies from 0 to 0.2 in the step of 0.02. Obviously, AFMN works better than any other algorithm both in training and testing.

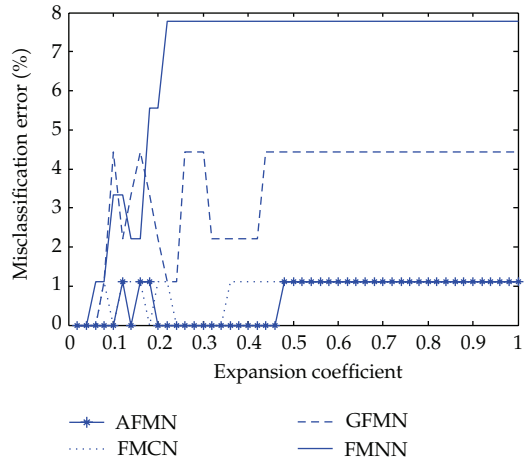



**Figure 9:** Recognition for 60% randomly selected training data.

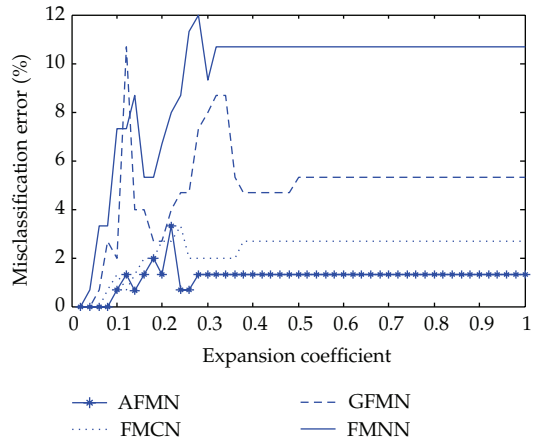



**Figure 10:** Recognition for entire Iris data.

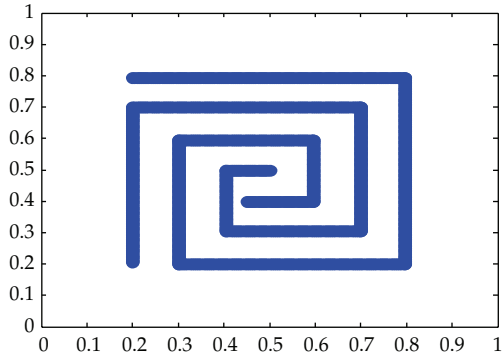


**Figure 11:** Synthetic test image.

**Table 6:** Robustness test on different amplitudes of noise.

| Amplitude of noise (%) | AFMN misclassification (%) | | | FMCN misclassification (%) | | | GFMN misclassification (%) | | | FMNN misclassification (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Min | Max | Ave | Min | Max | Ave | Min | Max | Ave | Min | Max | Ave |
| 1 | 0 | 1.33 | 1.12 | 0 | 3.33 | 1.42 | 0 | 3.33 | 1.55 | 0 | 4 | 2.09 |
| 5 | 0 | 2.67 | 1.30 | 0 | 3.33 | 1.85 | 0 | 3.33 | 1.91 | 0 | 4.67 | 2.24 |
| 10 | 0.67 | 2.67 | 1.70 | 0.67 | 4 | 2.09 | 0.67 | 4.67 | 2.38 | 1.33 | 5.33 | 2.68 |

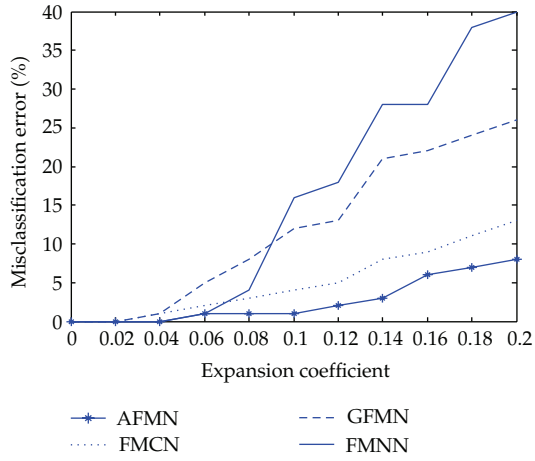



**Figure 12:** Performance on synthetic data.

### *4.7. Comparison with Other Traditional Classifier*

In this section we can compare the performance of AFMN, FMCN, GFMN, and FMNN on the iris dataset as Table 7 shows. The results show AFMN has no misclassification.

### *4.8. Comparison with Nodes Number (Hyperbox Number)*

The complexity of the created network after training affects the speed and efficiency of classification. 100% iris dataset is selected for training to see how many nodes created in the middle layer after training by each classifier. The results are shown in Figure 13.

As the expansion coefficient increases, the number of nodes decreases. AFMN, GFMN, and FMNN can generate a relatively simple structure network. In contrast, the architecture of FMCN is much more complex.

## 5. Conclusion

This paper proposes a complete classification system based on a new neural algorithm called AFMN, principle analysis algorithm, and genetic algorithm. The development of this classifier derives from the modification and completion of the fuzzy min-max neural network proposed by Simpson. Unlike the following neural algorithm for clustering and classification

**Table 7:** Comparison with other traditional classifiers.

| Technique | Misclassification |
|---|---|
| Bayes classifier[1] | 2 |
| k-nearest neighborhood[1] | 4 |
| Fuzzy k-nn[2] | 4 |
| Fisher ratios[1] | 3 |
| Ho-kashyap[1] | 2 |
| Perceptron[3] | 3 |
| Fuzzy perceptron[3] | 2 |
| FMNN[1] | 2 |
| GFMN[1] | 1/0 |
| GFMN[3] | 0 |
| FMCN[1] | 0 |
| FMCN[3] | 0 |
| AFMN[1] | 0 |
| AFMN[3] | 0 |

[1]Training set is of 75 data points (25 from each class) and test set consists of remaining data points.
[2]Training data is of 36 data points (12 from each class) and test set consists of 36 data points; results are then scaled up for 150 points.
[3]Training and testing data are the same.

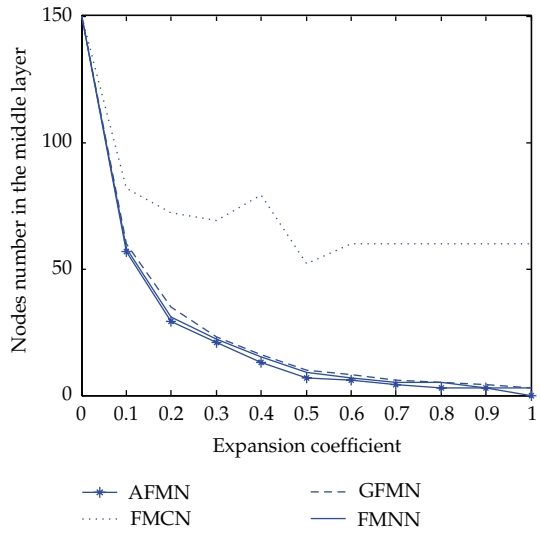



**Figure 13:** Node numbers generated by algorithm.

such as GFMN and FMCN, our classifier system is more complete and practical. The advantage of AFMN can be summarized as follows.

(1) AFMN adds preprocessing for input patterns and its technique is principle component analysis (PCA). This kind of data dimensionality reduction technique can reduce the number of features of input patterns and extract the useful information. This means saving the training and testing consuming time, meanwhile making the algorithm more suitable for application on real data for pattern classification.

(2) The introduction of confidence coefficient is overlooked by precedent neural algorithm for clustering and classification. Considering that there are nodes of more than one class in a hyperbox, the confidence of hyperboxes must be different, thus the operation that allocate 1 for any input pattern that falls in the hyperbox is not reasonable. So in the AFMN we calculate the confidence coefficient of each hyperbox for more precise classification.

(3) Adaptive genetic algorithm (AGA) is utilized in testing for parameters optimization while disposing of the step-setting obstacle in traversal method for parameters optimzation. GA can find the proper parameters combination more precisely and faster.

(4) Modification to the membership function ensures the self-adjustment according to the samples distribution and maintains the data core concept proposed in DCFMN. The data core can update itself online during classifying procedure, which is an indispensable ability to improve the classifier performance when training samples are insufficient.

(5) AFMN solves the problem existing in the overlap test of FMNN, GFMN, and FMCN; thus it can generate hyperboxes properly and remove redundant ones. By rewriting the contraction rules, AFMN maintains the simple architecture of FMNN, and abundant simulations demonstrate its high recognition rate.

In conclusion, integrated with principle component analysis for dimensionality reduction and genetic algorithm for parameters optimization, AFMN is a fast fuzzy min-max neural network classifier with high recognition rate and robustness. The use of AFMN network will be explored out of the laboratory.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants nos. 61104021, 61034005, and 61203086), the National High Technology Research and Development Program of China (2012AA040104), and the Fundamental Research Funds for the Central Universities of China (N100304007).

## References

[1] J. C. Bezdek and S. K. Pal, *Fuzzy Models for Pattern Recognition*, IEEE Press, Piscataway, NJ, USA, 1992.
[2] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.
[3] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Communications of the ACM*, vol. 37, no. 3, pp. 77–84, 1994.
[4] S. Mitra and S. K. Pal, "Fuzzy sets in pattern recognition and machine intelligence," *Fuzzy Sets and Systems*, vol. 156, no. 3, pp. 381–386, 2005.
[5] S. Abe and M. S. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 1, pp. 18–28, 1995.
[6] J. C. Bezdek and S. K. Pal, *Fuzzy Models for Pattern Recognition*, IEEE Press, Piscataway, NJ, USA, 1992.
[7] G. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: an adaptive resonance algorithm for rapid, stable classification of analog patterns," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '91)*, vol. 2, pp. 411–416, Seattle, Wash, USA, 1991.
[8] R. J. Wai and J. D. Lee, "Adaptive fuzzy-neural-network control for maglev transportation system," *IEEE Transactions on Neural Networks*, vol. 19, no. 1, pp. 54–70, 2008.

[9] S. Yilmaz and Y. Oysal, "Fuzzy wavelet neural network models for prediction and identification of dynamical systems," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1599–1609, 2010.

[10] E. Kolman and M. Margaliot, "Are artificial neural networks white boxes?" *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 844–852, 2005.

[11] C. F. Juang, S. H. Chiu, and S. J. Shiu, "Fuzzy system learned through fuzzy clustering and support vector machine for human skin color segmentation," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 37, no. 6, pp. 1077–1087, 2007.

[12] T. Hasegawa, S. Horikawa, and T. Furuhashi, "A study on fuzzy modeling of BOF using a fuzzy neural network," in *Proceedings of the 2nd International Conference on Fuzzy Systems, Neural Networks and Genetic Algorithms (IIZUKA)*, pp. 1061–1064, 1992.

[13] P. G. Campos, E. M. J. Oliveira, T. B. Ludermir, and A. F. R. Araújo, "MLP networks for classification and prediction with rule extraction mechanism," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 1387–1392, July 2004.

[14] A. Rizzi, M. Panella, F. M. F. Mascioli, and G. Martinelli, "A recursive algorithm for fuzzy Min-Max networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '00)*, vol. 6, pp. 541–546, July 2000.

[15] A. Rizzi, M. Panella, and F. M. F. Mascioli, "Adaptive resolution min-max classifiers," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 402–414, 2002.

[16] M. Meneganti, F. S. Saviello, and R. Tagliaferri, "Fuzzy neural networks for classification and detection of anomalies," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 848–861, 1998.

[17] A. Rizzi, M. Panella, and F. M. F. Mascioli, "Adaptive resolution min-max classifiers," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 402–414, 2002.

[18] S. Abe and M. S. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 1, pp. 18–28, 1995.

[19] R. Tagliaferri, A. Eleuteri, M. Menegatti, and F. Barone, "Fuzzy min-max neural networks: from classification to regression," *Soft Computing*, vol. 5, no. 16, pp. 69–76, 2001.

[20] M. Meneganti, F. S. Saviello, and R. Tagliaferri, "Fuzzy neural networks for classification and detection of anomalies," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 848–861, 1998.

[21] A. Rizzi, M. Panella, F. M. F. Mascioli, and G. Martinelli, "A recursive algorithm for fuzzy Min-Max networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN'00)*, pp. 541–546, July 2000.

[22] P. Liu and H. Li, "Efficient learning algorithms for three-layer regular feedforward fuzzy neural networks," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 545–558, 2004.

[23] W. Pedrycz, "Heterogeneous fuzzy logic networks: fundamentals and development studies," *IEEE Transactions on Neural Networks*, vol. 15, no. 6, pp. 1466–1481, 2004.

[24] P. K. Simpson, "Fuzzy min-max neural networks-I: classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 776–786, 1992.

[25] A. V. Nandedkar and P. K. Biswas, "A fuzzy min-max neural network classifier with compensatory neuron architecture," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 42–54, 2007.

[26] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 769–783, 2000.

[27] A. Quteishat, C. P. Lim, and K. S. Tan, "A modified fuzzy min-max neural network with a genetic-algorithm-based rule extractor for pattern classification," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 40, no. 3, pp. 641–650, 2010.

[28] H. Zhang, J. Liu, D. Ma, and Z. Wang, "Data-core-based fuzzy min-max neural network for pattern classification," *IEEE Transactions on Neural Networks*, vol. 22, no. 3, pp. 2339–2352, 2011.

[29] H. J. Kim and H. S. Yang, "A weighted fuzzy min-max neural network and its application to feature analysis," in *Proceedings of the 1st International Conference on Natural Computation (ICNC '05)*, Lecture Notes on Computer Science, pp. 1178–1181, August 2005.

[30] M. Kallas, C. Francis, L. Kanaan, D. Merheb, P. Honeine, and H. Amoud, "Multi-class SVM classification combined with kernel PCA feature extraction of ECG signals," in *Proceedings of the 19th International Conference on Telecommunications (ICT '12)*, pp. 1–5, April 2012.

[31] J. D. Schaffer and A. Morishma, "An adaptive crossover mechanism for genetic algorithms," in *Proceedings of the 2nd International Conference on Genetic Algorithms*, p. 3640, 1987.

[32] N. P. Jawarkar, R. S. Holambe, and T. K. Basu, "Use of fuzzy min-max neural network for speaker identification," in *Proceedings of the International Conference on Recent Trends in Information Technology (ICRTIT '11)*, pp. 178–182, June 2011.

[33] A. Quteishat and C. P. Lim, "A modified fuzzy min-max neural network with rule extraction and its application to fault detection and classification," *Applied Soft Computing Journal*, vol. 8, no. 2, pp. 985–995, 2008.

[34] Z. Huaguang and Q. Yongbing, "Modeling, identification, and control of a class of nonlinear systems," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 2, pp. 349–354, 2001.

[35] H. Zhang, Q. Gong, and Y. Wang, "Delay-dependent robust H∞ control for uncertain fuzzy hyperbolic systems with multiple delays," *Progress in Natural Science*, vol. 18, no. 1, pp. 97–104, 2008.

[36] H. G. Zhang, Y. C. Wang, and D. R. Liu, "Delay-dependent guaranteed cost control for uncertain stochastic fuzzy systems with multiple time delays," *Progress in Natural Science*, vol. 17, no. 1, pp. 95–102, 2007.

[37] H. G. Zhang, M. Li, J. Yang, and D. D. Yang, "Fuzzy model-based robust networked control for a class of nonlinear systems," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 39, no. 2, pp. 437–447, 2009.

[38] X. R. Liu, H. G. Zhang, and J. Dai, "Delay-dependent robust and reliable H∞ fuzzy hyperbolic decentralized control for uncertain nonlinear interconnected systems," *Fuzzy Sets and Systems*, vol. 161, no. 6, pp. 872–892, 2010.

[39] H. G. Zhang, Y. H. Luo, and D. Liu, "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1490–1503, 2009.

[40] C. Blake, E. Keogh, and C. J. Merz, "UCI repository of machine learning databases," University of California, Irvine, 1998, http://www.ics.uci.edu/~mlearn/MLRepository.htm.