Efficient Self-Adaptive Learning Algorithm for TSK-Type Compensatory Neural Fuzzy Networks

Cheng-Hung Chen

Abstract

In this paper, a TSK-type compensatory neural fuzzy network (TCNFN) for classification applications is proposed. The TCNFN model is a five-layer structure, which combines the traditional Takagi-Sugeno-Kang (TSK). Layer 3 of the TCNFN model contains adaptive compensatory fuzzy operations, which make fuzzy logic systems more adaptive and effective. Furthermore, a self-adaptive learning algorithm, which consists of the structure learning and the parameter learning, is also proposed. The structure learning is based on the degree measure to determine the number of fuzzy rules and the parameter learning is based on the gradient descent algorithm to adjust the parameters of the TCNFN. The advantages of the proposed method are that it converges quickly and that the fuzzy rules that are obtained are more precise. The performance of TCNFN compares excellently with other various existing methods.

Keywords: TSK-type neural fuzzy network, compensatory fuzzy operation, degree measure, backpropagation, classification.

1. Introduction

Traditional statistical classification procedures such as discriminant analysis are built on the Bayesian decision theory [1]. In these procedures, an underlying probability model must be assumed in order to calculate a posterior probability upon which the classification decision is made. One major limitation of the statistical models is that they work well when the assumptions are satisfied. The effectiveness of these methods depends to a large extent on the various assumptions or conditions under which the models are developed. Users must have good knowledge of both data properties and model capabilities before the models can be successfully applied.

Corresponding Author: Cheng-Hung Chen is with Department of Electrical Engineering, National Formosa University, No. 64, Wunhua Road, Huwei Township, Yunlin County 632, Taiwan.
E-mail: chchen.ee@nfu.edu.tw
Manuscript received 5 Oct. 2011; revised 28 Nov. 2011; accepted 10 Dec. 2012.

Neural networks [2] have emerged as an important tool for classification tasks. The recent and vast research activities in neural classification have established that neural networks are promising alternatives to various conventional classification methods. However, they usually encounter problems of slow convergence, and low understandability of the associated numerical weights. A fuzzy entropy measure [3] is employed to partition input feature space into decision regions and to select relevant features with good separability for the classification task. This approach lacks a definite method to determine the number of fuzzy rules required and the membership functions associated with each rule. Moreover, it also has a lack of an effective learning algorithm to refine the membership functions to minimize output errors. When the views above are summarized, it can be said that, in contrast to pure neural or fuzzy methods, neural fuzzy networks [4]-[14] possess the advantages of both neural networks and fuzzy systems. Neural fuzzy networks bring the low-level learning and computational power of neural networks into fuzzy systems and give the high-level human-like thinking and reasoning of fuzzy systems to neural networks.

In this study, a TSK-type compensatory neural fuzzy network (TCNFN) is proposed. The conventional neural fuzzy network can only adjust fuzzy membership functions by using fixed fuzzy operations, such as *Min* and *Max*. The compensatory operation-based neural fuzzy network with adaptive fuzzy reasoning is more effective and adaptive than the conventional neural fuzzy networks with non-adaptive fuzzy reasoning. Therefore, an effective neural fuzzy network should be able not only to adaptively adjust fuzzy membership functions but also to dynamically adjust fuzzy operators. Many researchers [15]-[17] have used the compensatory operation in fuzzy systems successfully.

A self-adaptive learning algorithm is proposed which can automatically construct the TCNFN. It consists of structure learning and parameter learning. The structure learning algorithm determines whether or not to add a new node that satisfies the fuzzy partition of input variables. Initially, the TCNFN model has no rules. The rules are automatically generated from training data by degree measure. The parameter learning algorithm is based on backpropagation to tune the free parameters in the TCNFN model simultaneously to minimize the cost

function. The advantages of the proposed method are summarized as follows: 1) it does not require assistance from a human expert, and its structure is obtained from the input data; 2) it can make the obtained fuzzy rules more precise using an adaptive compensatory fuzzy operation. 3) it converges quickly and it can obtain a higher correct classification rate. Therefore, the TCNFN model uses the compensatory fuzzy inference method that can make the fuzzy logic system more adaptive and effective.

This paper is organized as follows. Section 2 describes the basic structure and functions of the TCNFN. The self-adaptive learning algorithm including structure and parameter learning algorithms for the TCNFN model is presented in Section 3. In Section 4, the results of TCNFN simulation on two classification problems are discussed. Finally, conclusions are given in the last section.

2. Structure of TSK-Type Compensatory Neural Fuzzy Network

The section describes TSK-type compensatory neural fuzzy network (TCNFN). Compensatory operators are used to optimize fuzzy logic reasoning and to select optimal fuzzy operators. Therefore, an effective neural fuzzy network should be able not only to adaptively adjust fuzzy membership functions but also to dynamically optimize adaptive fuzzy operators. Figure 1 shows the structure of the TCNFN, which is systematized into N input variables, R-term nodes for each input variable, M output nodes and $N\times R$ membership function nodes. The TCNFN consists of five layers, $R\times(N\times2+2+(N+1)\times M)$ parameters and $N+(N\times R)+2\times R+M$ nodes, where R denotes the number of existing rules. Nodes in layer 1 are input nodes, which represent input variables. Nodes in layer 2 are called membership functions nodes to express the input fuzzy linguistic variables. Nodes in this layer are used to calculate Gaussian membership values. Each node in layer 3 is called a compensatory rule node and nodes in layer 4 are called consequent nodes. The number of nodes in layer 3 equals the number of compensatory fuzzy sets that correspond to each external linguistic input variable. Each compensatory rule node has a corresponding consequent node, which calculates a weighted linear combination of input variables. Nodes in layer 5 are called output nodes, each of which node is an individual output of the system.

The TCNFN realizes a fuzzy model in the following form.

$$\text{Rule}-j:$$

$$\text{IF}\left[x_1 \text{ is } A_{1j} \text{ and } x_2 \text{ is } A_{2j} \ldots \text{and } x_i \text{ is } A_{ij} \ldots \text{and } x_N \text{ is } A_{Nj}\right]^{1-\gamma_j+\frac{\gamma_j}{N}} \quad (1)$$

$$\text{THEN } y' \text{ is } w_{0j}+w_{1j}x_1+w_{2j}x_2+\ldots+w_{ij}x_i+\ldots w_{Nj}x_N$$

where x_i is the input variable; y' is the output variable; A_{ij} is the linguistic term of the precondition part; $\gamma_j \in [0,1]$ is the compensatory degree, and w_{0j} and w_{ij} are the corresponding parameters of consequent part.

Next, the operation functions of the nodes in each layer of the TCNFN are described. In the following, $u^{(l)}$ denotes the output of a node in the lth layer.

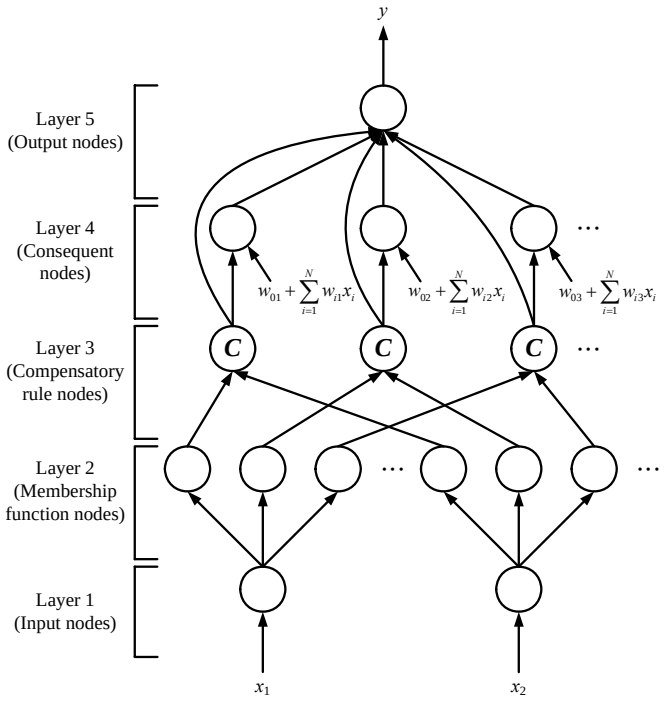

Figure 1. Structure of the proposed TCNFN.

Layer 1 (*Input Node*): No computation is performed in this layer. Each node in this layer is an input node, which corresponds to a single input variable, and only transmits directly input values to the next layer.

$$u_i^{(1)} = x_i. \quad (2)$$

Layer 2 (*Membership function Node*): Nodes in this layer correspond to a single linguistic label of the input variables in layer1. Therefore, the calculated membership value that specifies the degree to which an input value belongs to a fuzzy set in layer 2. The performed Gaussian membership function in layer 2 is

$$u_{ij}^{(2)} = \exp\left(-\frac{\left(u_i^{(1)}-m_{ij}\right)^2}{\sigma_{ij}^2}\right) \quad (3)$$

where m_{ij} and σ_{ij} are the mean and variance of the Gaussian membership function, respectively, of the jth term of the ith input variable x_i.

Layer 3 (*Compensatory Rule Node*): Nodes in this layer represent the precondition part of a fuzzy logic rule. They receive the one-dimensional membership degrees

of the associated rule from the nodes of a set in layer 2. The compensatory fuzzy operator described elsewhere is adopted to perform the IF-condition matching of fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \left(\prod_i u_{ij}^{(2)} \right)^{1-\gamma_j + \frac{\gamma_j}{N}} \tag{4}$$

where $\gamma_j = c_j^2 / \left(c_j^2 + d_j^2 \right) \in [0,1]$ is called the compensatory degree and $c_j, d_j \in [-1,1]$. The purpose of tuning c_j and d_j is to increase the adaptability of the fuzzy operator.

Layer 4 (*Consequent Node*): Nodes in this layer are called consequent nodes. The input to a node of layer 4 is the output from layer 3, and the other inputs are the input variables from layer 1, as shown in Fig. 1. For such a node,

$$u_j^{(4)} = u_j^{(3)} \left(w_{0j} + \sum_{i=1}^{N} w_{ij} x_i \right) \tag{5}$$

where the summation is over all the inputs and w_{ij} are the corresponding parameters of consequent part.

Layer 5 (*Output Node*): Each node in this layer corresponds to one output variable. The node integrates all of the actions recommended by layers 3 and 4 and acts as a defuzzifier with,

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)} \left(w_{0j} + \sum_{i=1}^{N} w_{ij} x_i \right)}{\sum_{j=1}^{R} u_j^{(3)}} \tag{6}$$

where R is the number of fuzzy rules and N is the number of input variables.

3. The Self-Adaptive Learning Algorithm

In this section, we present a self-adaptive learning algorithm for constructing the TCNFN. The proposed learning algorithm consists of a structure learning phase and a parameter learning phase. Structure learning is based on the degree used to determine the number of fuzzy rules. Parameter learning is based upon supervised learning algorithms. The backpropagation algorithm that minimizes a given cost function adjusts the weights in the consequent part, the parameters of the membership functions, the weights of the feedback, and the compensatory degree.

Initially, there are no nodes in the network except the input-output nodes, i.e., there are no any rule nodes and membership. The rule nodes are created dynamically and automatically as learning proceeds upon receiving incoming training data by performing the structure and parameter learning processes. The details of the structure learning phase and the parameter learning phase are described in the rest of this section.

A. *The Structure Learning Phase*

The first step in structure learning is to determine whether to extract a new rule from the training data as well as to determine the number of fuzzy sets in the universal of discourse for each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule, with m_{ij} and σ_{ij} representing the mean and variance of that cluster, respectively. For each incoming pattern x_i, the strength a rule is fired can be interpreted as the degree to which the incoming pattern belongs to the corresponding cluster.

For computational efficiency, we can use the compensatory operation of the firing strength obtained from $[\Pi u_{ij}^{(2)}]^{1-\gamma+\gamma/N}$ directly as the degree

$$F_j = \left(\prod_i u_{ij}^{(2)} \right)^{1-\gamma_j + \gamma_j / N} \tag{7}$$

where $F_j \in [0,1]$. Using this degree, we can obtain the following criterion for the generation of a new fuzzy rule for new, incoming data. It is described as follows. Find the maximum degree F_{max}

$$F_{\max} = \max_{1 \le j \le R_{(t)}} F_j \tag{8}$$

where $R(t)$ is the number of existing rules at time t. If $F_{\max} \le \overline{F}$, then a new rule is generated where $\overline{F} \in (0,1)$ is a prespecified threshold that decays during the learning process. Once a new rule is generated, the next step is to assign initial mean, variance, and weight of feedback for the new membership function. Since our goal is to minimize an objective function and the mean, variance, and weight of feedback are all adjustable later in the parameter learning phase. Hence, the mean, variance, and weight of feedback for the new membership function are set as follow:

$$m_{ij}^{(R_{(t+1)})} = x_i \tag{9}$$

$$\sigma_{ij}^{(R_{(t+1)})} = \sigma_{init} \tag{10}$$

where x_i is the new input data and σ_{init} is a pre-specified constant. Since the generation of a membership function corresponds to the generation of a new fuzzy rule, the compensatory degree $c_j^{(R_{(t+1)})}$, $d_j^{(R_{(t+1)})}$, i.e., $\gamma_j^{(R_{(t+1)})} = \left(c_j^{(R_{(t+1)})} \right)^2 / \left(\left(c_j^{(R_{(t+1)})} \right)^2 + \left(d_j^{(R_{(t+1)})} \right)^2 \right)$, and the weight, $w_j^{(R_{(t+1)})}$, associated with a new fuzzy rule has to be decided. Generally, the compensatory degree $c_j^{(R_{(t+1)})}$, $d_j^{(R_{(t+1)})}$, and the weight $w_j^{(R_{(t+1)})}$ are selected with random values in [-1,1].

B. *The Parameter Learning Phase*

After the network structure is adjusted according to the current training pattern, the network then enters the parameter learning phase to adjust the parameters of the network optimally based on the same training pattern. The learning process involves the determination of minimize a given cost function. The gradient of the cost function is computed and adjusted along the negative gradient. The idea of backpropagation algorithm is used for this supervised learning method. Considering the single output case for clarity, our goal is to minimize the cost function E is defined as

$$E(t)=\frac{1}{2}\left(y(t)-y^{d}(t)\right)^{2} \tag{11}$$

where $y^{d}(t)$ is the desired output and $y(t)$ is the current output. The parameter learning algorithm based on backpropagation is described below:

Layer 4: The error term to be propagated is calculated as

$$\delta^{(4)}=-\frac{\partial E(t)}{\partial y(t)}=y^{d}(t)-y(t). \tag{12}$$

The parameter of consequent part is updated by the amount

$$\Delta w_{0j}(t)=-\frac{\partial E(t)}{\partial w_{0j}(t)}=\left(-\frac{\partial E}{\partial u^{(5)}}\right)\left(\frac{\partial u^{(5)}}{\partial u_j^{(4)}}\right)\left(\frac{\partial u_j^{(4)}}{\partial w_{0j}}\right)=\frac{\delta^{(4)}u_j^{(3)}}{\sum_{j=1}^{R}u_j^{(3)}} \tag{13}$$

and

$$\Delta w_{ij}(t)=-\frac{\partial E(t)}{\partial w_{ij}(t)}=\left(-\frac{\partial E}{\partial u^{(5)}}\right)\left(\frac{\partial u^{(5)}}{\partial u_j^{(4)}}\right)\left(\frac{\partial u_j^{(4)}}{\partial w_{ij}}\right)=\frac{\delta^{(4)}u_j^{(3)}x_i}{\sum_{j=1}^{R}u_j^{(3)}}. \tag{14}$$

The parameter of consequent part in the output layer is updated according to the following equation:

$$w_{0j}(t+1)=w_{0j}(t)+\eta_w\Delta w_{0j}(t) \tag{15}$$

$$w_{ij}(t+1)=w_{ij}(t)+\eta_w\Delta w_{ij}(t) \tag{16}$$

where factor η_w is the learning rate parameter of the parameter and t denotes the jth iteration number.

Layer 3: In this layer only the error term needs to be computed and propagated

$$\begin{aligned}\delta^{(3)}&=-\frac{\partial E(t)}{\partial u_j^{(3)}}=\left(-\frac{\partial E}{\partial y}\right)\left(\frac{\partial y}{\partial u_j^{(3)}}\right).\\&=\delta^{(4)}w_j\end{aligned} \tag{17}$$

To eliminate the constraint $\gamma_j\in[0,1]$, we redefine $\gamma_j(t)$ as follows

$$\gamma_j(t)=\frac{c_j^{\,2}(t)}{c_j^{\,2}(t)+d_j^{\,2}(t)} \tag{18}$$

$$\begin{aligned}\Delta\gamma_j(t)&=-\frac{\partial E(t)}{\partial\gamma_j(t)}=\left(-\frac{\partial E}{\partial u_j^{(3)}}\right)\left(\frac{\partial u_j^{(3)}}{\partial\gamma_j}\right).\\&=\delta^{(3)}\left(\frac{1}{n}-1\right)\ln\left(\prod_i u_{ij}^{(2)}\right)u_j^{(3)}\end{aligned} \tag{19}$$

We have

$$c_j(t+1)=c_j(t)+\eta_c\left\{\frac{2c_j(t)d_j^{\,2}(t)}{[c_j^{\,2}(t)+d_j^{\,2}(t)]^2}\right\}\Delta\gamma_j(t) \tag{20}$$

$$d_j(t+1)=d_j(t)-\eta_d\left\{\frac{2c_j^{\,2}(t)d_j(t)}{[c_j^{\,2}(t)+d_j^{\,2}(t)]^2}\right\}\Delta\gamma_j(t) \tag{21}$$

$$\gamma_j(t+1)=\frac{c_j^{\,2}(t+1)}{c_j^{\,2}(t+1)+d_j^{\,2}(t+1)}. \tag{22}$$

In all the above formulas, η_c and η_d are the learning rate of the parameter $c_j(t)$ and the parameter $d_j(t)$, respectively.

Layer 2: The error term is calculated as follows:

$$\begin{aligned}\delta^{(2)}&=-\frac{\partial E(t)}{\partial u_{ij}^{(2)}}=\left(-\frac{\partial E}{\partial u_j^{(3)}}\right)\left(\frac{\partial u_j^{(3)}}{\partial u_{ij}^{(2)}}\right)\\&=\delta^{(3)}\left(1-\gamma_j+\frac{\gamma_j}{N}\right)\left(\prod_i u_{ij}^{(2)}\right)^{-\gamma_j+\frac{\gamma_j}{N}}\left(\prod_{l\neq i}u_{lj}^{(2)}\right)\end{aligned} \tag{23}$$

where l is the lth dimension. The update mean is

$$\begin{aligned}\Delta m_{ij}(t)&=-\frac{\partial E(t)}{\partial m_{ij}(t)}=\left(-\frac{\partial E}{\partial u_{ij}^{(2)}}\right)\left(\frac{\partial u_{ij}^{(2)}}{\partial m_{ij}}\right).\\&=\delta^{(2)}u_{ij}^{(2)}\left(\frac{2(u_i^{(1)}-m_{ij})}{\sigma_{ij}^2}\right)\end{aligned} \tag{24}$$

The updated variance is

$$\begin{aligned}\Delta\sigma_{ij}(t)&=-\frac{\partial E(t)}{\partial\sigma_{ij}(t)}=\left(-\frac{\partial E}{\partial u_{ij}^{(2)}}\right)\left(\frac{\partial u_{ij}^{(2)}}{\partial\sigma_{ij}}\right).\\&=\delta^{(2)}u_{ij}^{(2)}\left(\frac{2(u_i^{(1)}-m_{ij})^2}{\sigma_{ij}^3}\right)\end{aligned} \tag{25}$$

The mean and variance of the membership functions in this layer are updated as follows:

$$m_{ij}(t+1)=m_{ij}(t)+\eta_m\Delta m_{ij}(t) \tag{26}$$

$$\sigma_{ij}(t+1)=\sigma_{ij}(t)+\eta_\sigma\Delta\sigma_{ij}(t) \tag{27}$$

where η_m and η_σ are the learning rate parameters of the mean and the variance for the Gaussian function, respectively.

4. Illustrative Examples

In this section, we evaluate the performance of the proposed TCNFN model using two better-known

benchmark data sets for classification problems. The first example uses the Iris data and the second example uses the Wisconsin breast cancer data. The two benchmark data sets are available from the University of California, Irvine, via an anonymous ftp address ftp://ftp.ics.uci.edu/pub/machine-learning-databases. In the following simulations, the parameters and number of training epochs were based on the desired accuracy. In short, the trained TCNFN model was stopped once its high learning efficiency was demonstrated.

A. *Iris Data Classification*

The Fisher-Anderson iris data consists of four input measurements, sepal length (*sl*), sepal width (*sw*), petal length (*pl*), and petal width (*pw*), on 150 specimens of the iris plant. Three species of iris were involved, *Iris Sestosa*, *Iris Versiolor* and *Iris Virginica*, and each species contains 50 instances.

In the Iris data experiment, 25 instances with four features from each species were randomly selected as the training set (i.e., a total of 75 training patterns were used as the training data set) and the remaining instances were used as the testing set. The 75 training patterns were obtained via a random selection process from the original Iris dataset of 150 patterns. The learning rate $\eta_w = \eta_c = \eta_d = \eta_m = \eta_\sigma = 0.01$, and the prespecified threshold $\overline{F} = 10^{-4}$ are used. The training process is continued for 100 epochs. After the structure and parameter learning, there are three fuzzy rules generated.

Figure 2 (a)-(f) show the distribution of the training pattern and the final assignment of the fuzzy rules (i.e., distribution of input membership functions). Since the region covered by a Gaussian membership function is unbounded, in Figure 2 (a)-(f), the boundary of each ellipse represent a rule with a firing strength of 0.5. The TCNFN model has zero re-substitution error when the three fuzzy rules were generated. We compared the testing accuracy of our model with that of other methods -- the traditional multilayer neural network, the standard radial basis function network (RBFN), the SONFIN model [11], and the TCNFN without compensatory operation. Five experiments were used. These experiments calculated the classification accuracy and the values of the average produced on the testing set using the traditional multilayer neural network, the radial basis function network (RBFN), the SONFIN model, the TCNFN model without compensatory operation, and the proposed TCNFN model.

During the learning phase, the learning curves from the proposed TCNFN model, the TCNFN without compensatory operation, the SONFIN model, and the RBFN model are shown in Figure 3. Table 1 shows that the experiments with the TCNFN model result in high accuracy, with an accuracy percentage ranging from 96% to 98.67%. The means of re-substitution accuracy was 97.33%. The average classification accuracy of the TCNFN model was better than that of other methods. Table 2 shows the comparison of the classification results of the TCNFN model with other methods [3], [7], [18]-[20] on the iris data. The results show that the proposed TCNFN model is able to keep similar average substitution accuracy.

Table 1. Classification accuracy using various methods for the Iris data.

Model / Experiment #	Neural Network	RBFN	SONFIN [11]	Non-compensatory TCNFN	TCNFN
1	92	93.33	98.67	97.33	97.33
2	93.33	97.33	97.33	96	98.67
3	93.33	96	98.67	98.67	97.33
4	92	96	96	93.33	97.33
5	93.33	93.33	96	96	96
Average (%)	92.8	95.2	97.33	96.27	97.33

Table 2. Average re-substitution accuracy comparison of various models for the Iris data classification problem.

Methods	Average re-substitution accuracy (%)
FEBFC [3]	96.91
SANFIN [7]	97.33
FMMC [18]	97.3
FUNLVQ+GFENCE [19]	96.3
Wu-and-Chen's [20]	96.21
TCNFN	97.33

B. *Wisconsin Breast Cancer Diagnostic Data*

The Wisconsin breast cancer diagnostic data set contains 699 patterns distributed into two output classes, "benign" and "malignant." Each pattern consists of nine input features: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. 458 patterns are in the benign class and the other 241 patterns are in the malignant class. Since there were 16 patterns containing missing values, we used 683 patterns to evaluate the performance of the proposed TCNFN model. To compare the performance with other models, we used half of the 683 patterns as the training set and the remaining patterns as the testing set.

Experimental conditions were the same as the previous experiment. We also used half of the original data patterns as the training data (randomly selected) and the remaining patterns as the testing data. For training, the training patterns were randomly chosen, and the remaining patterns were used for testing. The learning rate $\eta_w = \eta_c = \eta_d = \eta_m = \eta_\sigma = 0.01$, and the prespecified threshold are $\overline{F} = 10^{-10}$ used. The training process is continued for 100 epochs. After the structure and parameter learning, there are two fuzzy rules obtained.

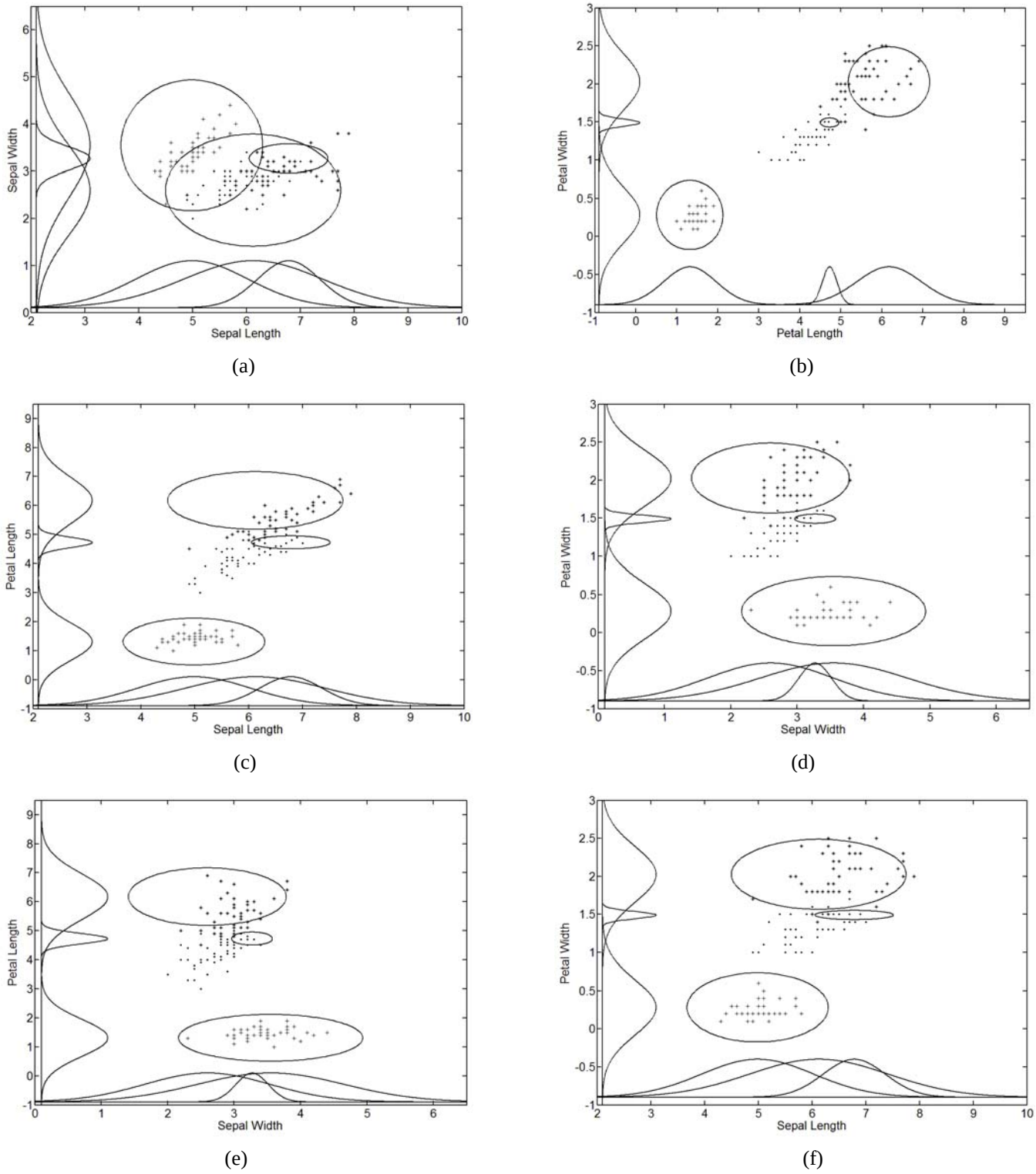

Figure 2. The distribution of input training patterns and final assignment of three rules. (a) For the *Sepal Length* and *Sepal Width* dimensions. (b) For the *Petal Length* and *Petal Width* dimensions. (c) For the *Sepal Length* and *Petal Length* dimensions. (d) For the *Sepal Width* and *Petal Width* dimensions. (e) For the *Sepal Width* and *Petal Length* dimensions. (f) For the *Sepal Length* and *Petal Width* dimensions.

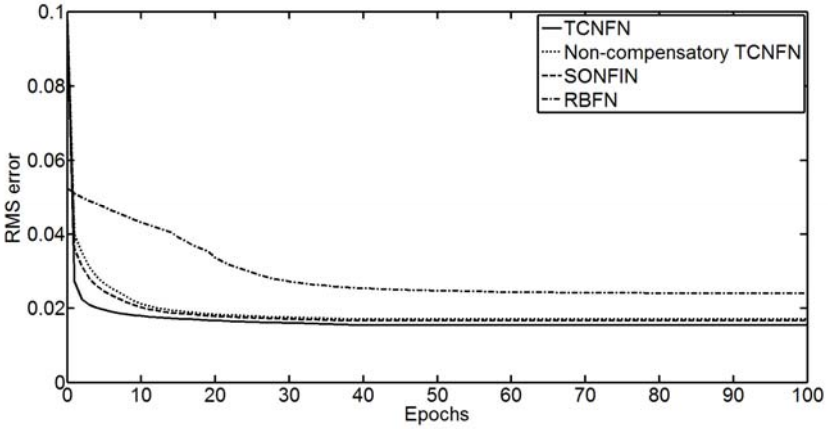

Figure 3. Learning curves of the TCNFN model, the non-compensatory TCNFN model, the SONFIN model and the RBFN model.

Five experiments also were used. These experiments calculated the classification accuracy and the values of the average produced on the testing set by the neural network, the RBFN model, the SONFIN model, the TCNFN model without compensatory operation, and the proposed TCNFN model. During the supervised learning phase, 100 epochs of training were performed. Figure 4 shows the membership functions for each input feature. The learning curves from the proposed TCNFN model, the TCNFN without compensatory operation, the SONFIN model, and the RBFN model are shown in Figure 5. The performance of the TCNFN model is better than the performance of all other models.

Table 3. Classification accuracy for the Wisconsin Breast Cancer Diagnostic data.

Model / Experiment #	Neural Network	RBFN	SONFIN [11]	Non-Compensatory TCNFN	TCNFN
1	91.23	92.98	95.31	96.49	96.78
2	90.06	93.27	96.49	96.2	95.91
3	89.77	92.69	95.75	94.74	97.37
4	89.18	93.86	96.63	95.03	97.07
5	90.35	93.57	97.22	95.61	97.66
Average (%)	90.12	93.27	96.28	95.61	96.96

Table 3 shows that the experiments with the TCNFN model result in high accuracy, with an accuracy percentage ranging from 95.91% to 97.66%. The means of re-substitution accuracy was 96.96%. The average classification accuracy of the TCNFN model was better than that of other methods. We compared the testing accuracy of our model with that of other methods [2]-[4], [7], [21]. Table 4 shows the comparison between the learned TCNFN models and other fuzzy, neural networks, and neural fuzzy networks. The average classification accuracy of the TCNFN model is better than that of other methods.

Table 4. Average accuracy comparison of various models for Wisconsin Breast Cancer Diagnostic data.

Models	Average accuracy (%)
NNFS [2]	94.15
FEBFC [3]	95.14
NEFCLASS [4]	92.7
SANFIS [7]	96.3
MSC [21]	94.9
TCNFN	96.96

5. Conclusion

A TSK-type compensatory neural fuzzy network (TCNFN) was proposed in this paper. Compensatory operators were used to select fuzzy logic reasoning and fuzzy operators. Therefore, an effective neural fuzzy network should be able not only to adaptively adjust fuzzy membership functions but also to dynamically adapt fuzzy operators. A self-adaptive learning algorithm was also proposed for performing structure learning and parameter learning. Two examples showed that the proposed TCNFN improves the system performance in terms of a fast learning convergence, a high correct classification rate, and parsimonious system structures.

Acknowledgment

This research was sponsored by the National Science Council, Taiwan, R.O.C., under Grant NSC 101-2221-E-150-085.

References

[1] P. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, New York: Wiley, 1973.

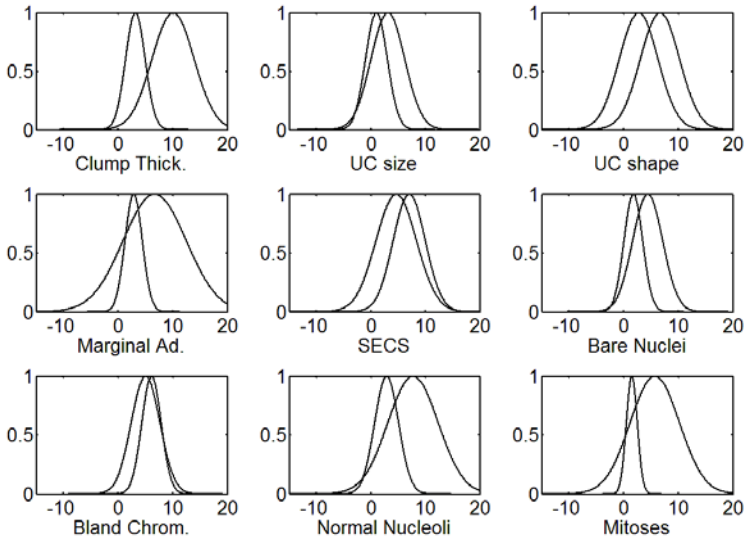

Figure 4. Input membership functions for breast cancer classification.

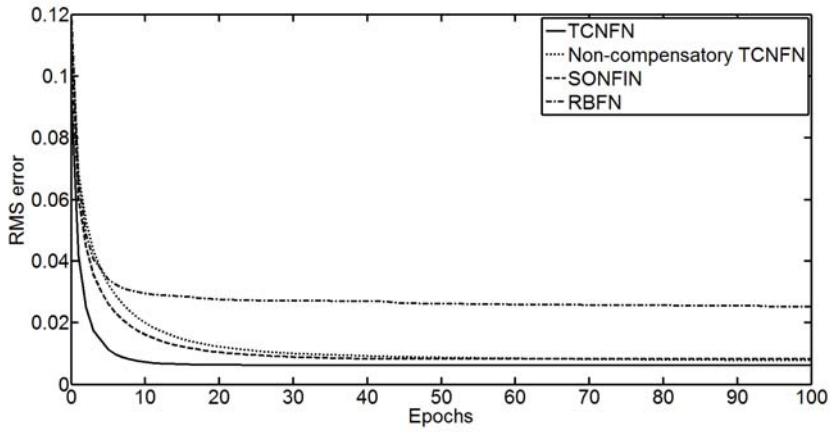

Figure 5. Learning curves of the TCNFN model, the non-compensatory TCNFN model, the SONFIN model and the RBFN model.

[2] R. Setiono and H. Liu, "Neural-network feature selector," *IEEE Trans. on Neural Network*, vol. 8, no. 3, pp. 654-662, 1997.

[3] H. M. Lee, C. M. Chen, J. M. Chen, and Y. L. Jou, "An efficient fuzzy classifier with feature selection based on fuzzy entropy," *IEEE Trans. on Syst., Man, Cybern. B*, vol. 31, pp. 426-432, 2001.

[4] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data," *Fuzzy Sets and Syst.*, vol. 89, pp. 277-288, 1997.

[5] M. Russo, "FuGeNeSys-a fuzzy genetic neural system for fuzzy modeling," *IEEE Trans. on Fuzzy Syst.*, vol. 6, no. 3, pp. 373-388, 1998.

[6] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system (SuPFuNIS)," *IEEE Trans. on Neural Networks*, vol. 13, no. 3, pp. 578-599, 2002.

[7] J. S. Wang and C. S. George Lee, "Self-adaptive neuro-fuzzy inference systems for classification applications," *IEEE Trans. on Fuzzy Systems*, vol. 10, no. 6, pp. 790-802, 2002.

[8] C. F. Juang, S. H. Chiu, and S. W. Chang, "A self-organizing TS-type fuzzy network with support vector learning and its application to classification problems," *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 5, pp. 998-1008, 2007.

[9] W. Y. Wang, Y. H. Chien, and I. H. Li, "An on-line robust and adaptive T-S fuzzy-neural controller for more general unknown systems," *International Journal of Fuzzy Systems*, vol. 10, no. 1, pp. 33-43, 2008.

[10] C. C. Chuang, J. T. Jeng, and C. W. Tao, "Two-stages support vector regression for fuzzy neural networks with outliers," *International Journal of Fuzzy Systems*, vol. 11, no. 1, pp. 20-28, 2009.

[11] C. F. Juang and C. T. Lin, "An online self-constructing neural fuzzy inference network

and its applications," *IEEE Trans. on Fuzzy Systems*, vol. 6, no. 1, pp. 12-31, 1998.
[12] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: survey in soft computing framework," *IEEE Trans. on Neural Networks*, vol. 11, no. 3, pp. 748-768, 2000.
[13] S. Osowski and T. H. Linh, "ECG beat recognition using fuzzy hybrid neural network," *IEEE Trans. on Biomedical Engineering*, vol. 48, no. 11, pp. 1265-1271, 2001.
[14] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge based learning," *IEEE Trans. on Syst. Man. and Cybern., part B*, vol. 31, no. 6, pp. 902-918, 2001.
[15] Y. Q. Zhang and A. Kandel, "Compensatory neurofuzzy systems with fast learning algorithms," *IEEE Trans. on Neural Networks*, vol. 9, no. 1, pp. 83-105, 1998.
[16] H. Seker, D. E. Evans, N. Aydin, and E. Yazgan, "Compensatory fuzzy neural networks-based intelligent detection of abnormal neonatal cerebral doppler ultrasound waveforms," *IEEE Trans. on Information Technology in Biomedicine*, vol. 5, no. 3, pp. 187-194, 2001.
[17] A. V. Nandedkar and P. K. Biswas, "A fuzzy min-max neural network classifier with compensatory neuron architecture," *IEEE Trans. on Neural Networks*, vol. 18, no. 1, pp. 42-54, 2007.
[18] P. K. Simpson, "Fuzzy min-max neural networks-part I: classification," *IEEE Trans. on Neural Networks*, vol. 3, pp. 776-786, 1992.
[19] H. M. Lee, "A neural network classifier with disjunctive fuzzy information," *Neural Networks*, vol. 11, no. 6, pp. 1113-1125, 1998.
[20] T. P. Wu and S. M. Chen, "A new method for constructing membership functions and fuzzy rules from training examples," *IEEE Trans. on Syst. Man. and Cybern., part B*, vol. 29, pp. 25-40, 1999.
[21] B. C. Lovel and A. P. Bradley, "The multiscale classifier," *IEEE Trans. on Pattern Anal. Machine Intell*, vol. 18, pp. 124-137, 1996.

Cheng-Hung Chen received the B.S. and M.S. degrees in computer science and information engineering from the Chaoyang University of Technology, Taiwan, R.O.C., in 2002 and 2004, respectively, and the Ph.D. degree in electrical and control engineering from the National Chiao Tung University, Taiwan, R.O.C., in 2008. Currently, he is an Assistant Professor of Electrical Engineering Department, National Formosa University, Yunlin County, Taiwan, R.O.C. His current research interests are fuzzy systems, neural networks, evolutionary algorithms, intelligent control, pattern recognition, and image processing. He has authored or coauthored more than 40 papers published in the referred journals and conference proceedings. He is also a member of the Chinese Fuzzy Systems Association (CFSA), the Taiwanese Association for Artificial Intelligence (TAAI), and the IEEE Computational Intelligence Society.