Expert Systems with Applications 38 (2011) 6202–6208

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A functional neural fuzzy network for classification applications

Chi-Feng Wu[a], Cheng-Jian Lin[b,*], Chi-Yung Lee[c]

[a] Department of Information Management and Communication, Wenzao Ursuline College of Languages, Kaohsiung City 807, Taiwan, ROC
[b] Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung County 411, Taiwan, ROC
[c] Department of Computer Science and Information Engineering, NanKai University of Technology, Nantou County 542, Taiwan, ROC

ARTICLE INFO

Keywords:
Entropy
Functional link neural networks (FLNNs)
Neural fuzzy networks (NFNs)
Classification
Online learning

ABSTRACT

This study presents a functional neural fuzzy network (FNFN) for classification applications. The proposed FNFN model adopts a functional neural network (FLNN) to the consequent part of the fuzzy rules. Orthogonal polynomials and linearly independent functions are used for a functional expansion of the FLNN. Thus, the consequent part of the proposed FNFN model is a nonlinear combination of input variables. The FNFN model can construct its structure and adapt its free parameters with online learning algorithms, which consist of structure learning algorithm and parameter learning algorithm. The structure learning algorithm is based on the entropy measure to determine the number of fuzzy rules. The parameter learning algorithm, based on the gradient descent method, can adjust the shapes of the membership functions and the corresponding weights of the FLNN. Finally, the FNFN model is applied to various simulations. The simulation results for the Iris, Wisconsin breast cancer, and wine classifications show that FNFN model has superior performance than other models for classification applications.

1. Introduction

Classification is one of the most frequent decision making tasks performed by humans. A classification problem for objects is the problem of assigning these objects into predefined groups or classes based on the number of observed attributes related to that object, and giving some criteria for determining whether a assigned object is in a particular group or not. Many decision problems in business, science, industry, and medicine can be treated as classification problems. Most traditional statistical classification approaches, such as discrimination analysis, are built on the Bayesian decision theory (Duda & Hart, 1973). These approaches generally have an explicit underlying probability model. The model is used to calculate the posteriority probability, and upon which a classification decision is made. One major drawback of statistical models is that they work well only when the underlying assumptions are correct. The effectiveness of these methods depends to a large extent on the various assumptions or conditions under which the models are developed. To make the model successfully applied, users must have a good knowledge of both data properties and model capabilities.

Neural networks (Setiono & Liu, 1997) have arisen as an important tool for pattern recognition, modeling, and prediction. Vast research activities in neural classification have shown that neural networks are promising alternatives to various conventional classification methods. However, the meaning of each neuron and the function of each weight are difficult to understand in the neural networks. A fuzzy entropy measure (Lee, Chen, Chen, & Jou, 2001) is employed to partition the input feature space into decision regions and to select relevant features with good separability for the classification task. According to the literature review mentioned before, neural fuzzy networks (NFNs) (Jang, 1993; Juang & Lin, 1998; Li & Lee, 2003; Lin & Lee, 1996; Lin & Lin, 1997; Lin, Tsai, & Liu, 2001; Mitra & Hayashi, 2000; Sun, Sun, Li, & Li, 2003; Takagi & Sugeno, 1985; Wang & Mendel, 1992) provide the advantages of both neural networks and fuzzy systems, unlike pure neural networks or fuzzy systems alone. NFNs 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.

Two typical types of NFNs are the Mamdani-type and the Takagi–Sugeno–Kang (TSK)-type. For Mamdani-type NFNs (Lin & Lin, 1997; Lin et al., 2001; Wang & Mendel, 1992), the minimum fuzzy implication is adopted in fuzzy reasoning. Meanwhile, for TSK-type NFNs (Jang, 1993; Juang & Lin, 1998; Li & Lee, 2003; Takagi & Sugeno, 1985), the consequence part of each rule is a linear combination of input variables. Many researches (Jang, 1993; Juang & Lin, 1998) have shown that TSK-type NFNs offer better network size and learning accuracy than Mamdani-type NFNs. In the typical TSK-type NFN, which is a linear polynomial of input variables, the model output is approximated locally by the rule hyper-planes. Nevertheless, the traditional TSK-type NFN does not take full advantage of the mapping capabilities that may be

* Corresponding author.
E-mail address: cjlin@ncut.edu.tw (C.-J. Lin).

offered by the consequent part. Introducing a nonlinear function, especially a neural structure, to the consequent part of the fuzzy rules has yielded the neural networks designed on approximate reasoning architecture (NARA) (Takagi, Suzuki, Koda, & Kojima, 1992) and the coactive neural fuzzy inference system (CANFIS) (Mizutani & Jang, 1995) models. These models (Mizutani & Jang, 1995; Takagi et al., 1992) apply multilayer neural networks to the consequent part of the fuzzy rules. Although the interpretability of the model is reduced, the representational capability of the model is markedly improved. The main disadvantages of multilayer neural network are slower convergence and greater computational complexity. Therefore, in our proposed model, called a functional neural fuzzy network (FNFN), uses the functional link neural network (FLNN) (Pao, 1989; Patra, Pal, Chatterji, & Panda, 1999) to the consequent part of the fuzzy rules. Thus, the consequent part of the proposed FNFN model is a nonlinear combination of input variables, which differs from the other existing models (Jang, 1993; Juang & Lin, 1998; Lin & Lin, 1997). The FLNN is a single-layer neural structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries with nonlinear functional expansion. The FLNN (Pao, Phillips, & Sobajic, 1992) was conveniently used for function approximation and pattern classification with faster convergence rate and less computational complexity than a multilayer neural network. Moreover, using the functional expansion can effectively increase the dimensionality of the input vector, so the hyperplanes generated by the FLNN will provide a good discrimination capability in input data space.

2. Structure of functional neural fuzzy network

This section describes the structure of FLNN model and the structure of the FNFN model. In FLNN, the input data usually incorporate high-order effects. Thus, the dimensionality of the input space is artificially increased using a functional expansion. Accordingly, the input representation is enhanced and linear separability is achieved in the extended space. The FNFN model adopted the FLNN, generating complex nonlinear combinations of input variables to the consequent part of the fuzzy rules. The rest of this section details these structures.

2.1. Functional link neural networks

The FLNN is a single-layer network in which the need for hidden layers is removed. Thus, the computational complexity is less and learning speed is faster. While the input variables generated by the linear links of neural networks are linearly weighted, the functional link acts on an element of input variables by generating a set of linearly independent functions (i.e., the use of suitable orthogonal polynomials for a functional expansion) and then evaluating these functions with the variables as the arguments. Therefore, the FLNN structure considers trigonometric functions. For example, for a two-dimensional input $X = [x_1, x_2]^T$, the enhanced input is obtained using trigonometric functions in $\Phi = [x_1, \sin(\pi x_1), \cos(\pi x_1), x_2, \sin(\pi x_2), \cos(\pi x_2), \ldots]^T$. Thus, the input variables can be separated in the enhanced space (Pao, 1989). In the FLNN structure with reference to Fig. 1, a set of basis functions Φ and a fixed number of weight parameters W represent $f_W(x)$. The theory behind the FLNN for multidimensional function approximation has been discussed elsewhere (Patra & Pal, 1995) and is analyzed later.

Consider a set of basis functions $B = \{\phi_k \in \Phi(A)\}_{k \in K}, K = \{1, 2, \ldots\}$, with the following properties: (1) $\phi_1 = 1$; (2) the subset $B_j = \{\phi_k \in B\}_{k=1}^{M}$ is a linearly independent set, meaning that if $\sum_{k=1}^{M} w_k \phi_k = 0$, then $w_k = 0$ for all $k = 1, 2, \ldots, j$; and (3) $\sup_j \left[\sum_{k=1}^{j} \|\phi_k\|_A^2\right]^{1/2} < \infty$.

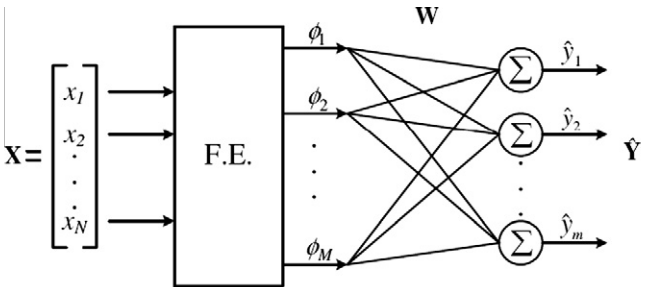

Fig. 1. Structure of a FLNN.

Let $B_M = \{\phi_k\}_{k=1}^{M}$ be a set of basis functions to be considered, as shown in Fig. 1. The FLNN comprises M basis functions $\{\phi_1, \phi_2, \ldots, \phi_M\} \in B_M$. The linear sum of the jth node is given by

$$\hat{y}_j = \sum_{k=1}^{M} w_{kj} \phi_k(\mathbf{X}) \tag{1}$$

where $\mathbf{X} \in A \subset \Re^n, \mathbf{X} = [x_1, x_2, \ldots, x_n]^T$ is the input vector and $W_j = [w_{1j}, w_{2j}, \ldots, w_{Mj}]^T$ is the weight vector associated with the jth output of the FLNN. $\hat{y}_j$ denotes the local output of the FLNN structure and the consequent part of the jth fuzzy rule in the FNFN model. Thus, (1) can be expressed in matrix form as $\hat{y}_j = W_j \Phi$, where $\Phi = [\phi_1(x), \phi_2(x), \ldots, \phi_M(x)]^T$ is the basis function vector, which is the output of the functional expansion block. The m-dimensional linear output may be given by $\hat{\mathbf{y}} = W\Phi$, where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]^T$, m denotes the number of functional link bases, which equals the number of fuzzy rules in the FNFN model, and W is an $(m \times M)$-dimensional weight matrix of the FLNN given by $W = [w_1, w_2, \ldots, w_m]^T$. The jth output of the FLNN is given by$\hat{y}'_j = \rho(\hat{y}_j)$, where the nonlinear function $\rho(\cdot) = \tanh(\cdot)$. Thus, the m-dimensional output vector is given by

$$\hat{Y} = \rho(\hat{y}) = f_{\mathbf{W}}(x) \tag{2}$$

where $\hat{Y}$ denotes the output of the FLNN. In the FNFN model, the corresponding weights of functional link bases do not exist in the initial state, and the amount of the corresponding weights of functional link bases generated by the online learning algorithm is consistent with the number of fuzzy rules. Section III details the online learning algorithm.

2.2. Structure of FNFN model

This subsection describes the FNFN model, which uses a nonlinear combination of input variables (FLNN). Each fuzzy rule corresponds to a sub-FLNN, comprising a functional link. Fig. 2 presents the structure of the proposed FNFN model. The FNFN model realizes a fuzzy IF-THEN rule in the following form.

Rule j:

IF $x_1$is A_{1j} and x_2 is $A_{2j} \ldots$ and x_i is $A_{ij} \ldots$ and x_N is A_{Nj}

then $\hat{y}_j = \sum_{k=1}^{M} w_{kj} \phi_k = w_{1j}\phi_1 + w_{2j}\phi_2 + \ldots + w_{Mj}\phi_M$

$$\hat{y}_j = \sum_{k=1}^{M} w_{kj} \phi_k(\mathbf{X}) \tag{3}$$

where x_i and $\hat{y}_j$ are the input and local output variables, respectively; A_{ij} is the linguistic term of the precondition part with Gaussian membership function, N is the number of input variables, w_{kj} is the link weight of the local output, ϕ_k is the basis trigonometric function of input variables, M is the number of basis function, and rule j is the jth fuzzy rule.

The operation functions of the nodes in each layer of the FNFN model are now described. In the following description, denotes the

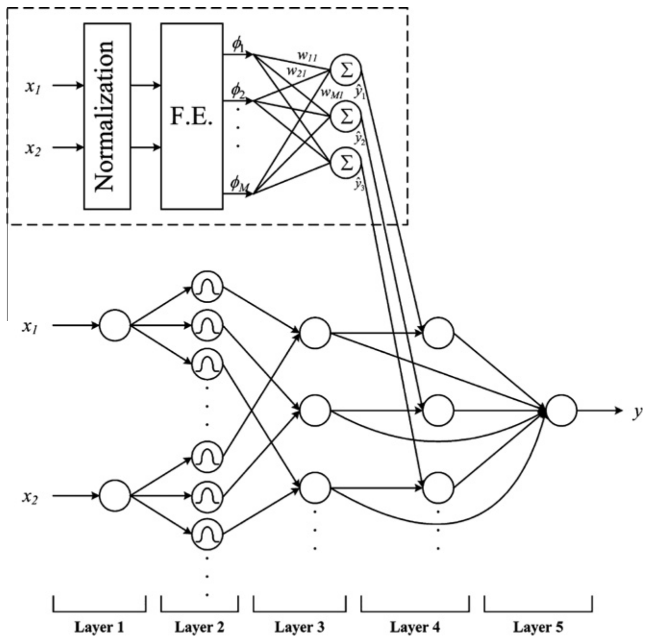

Fig. 2. Structure of proposed FNFN model.

output of a node in the lth layer. No computation is performed in layer 1. Each node in this layer only transmits input values to the next layer directly

$$u_i^{(1)} = x_i \tag{4}$$

Each fuzzy set A_{ij} is described here by a Gaussian membership function. Therefore, the calculated membership value in layer 2 is

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right) \tag{5}$$

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.

Nodes in layer 3 receive one-dimensional membership degrees of the associated rule from the nodes of a set in layer 2. Here, the product operator described earlier is adopted to perform the precondition part of the fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{6}$$

where the $\prod_i u_{ij}^{(2)}$ of a rule node represents the firing strength of its corresponding rule.

Nodes in layer 4 are called consequent nodes. The input to a node in layer 4 is the output from layer 3, and the other inputs are calculated from the FLNN that has not used the function $\tanh(\cdot)$, as shown in Fig. 2. For such a node

$$u_j^{(4)} = u_j^{(3)} \cdot \sum_{k=1}^{M} w_{kj}\phi_k \tag{7}$$

where w_{kj} is the corresponding link weight of the FLNN and ϕ_k is the functional expansion of input variables. The functional expansion uses a trigonometric polynomial basis function, given by $[x_1, \sin(\pi x_1), \cos(\pi x_1), x_2, \sin(\pi x_2), \cos(\pi x_2)]$ for two-dimensional input variables. Therefore, M is the number of basis functions, $M = 3 \times N$, where N is the number of input variables. Moreover, the output nodes of the FLNN depend on the number of fuzzy rules of the FNFN model.

The output node in layer 5 integrates all of the actions recommended by layers 3 and 4 and acts as a defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}\left(\sum_{k=1}^{M} w_{kj}\phi_k\right)}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}\hat{y}_j}{\sum_{j=1}^{R} u_j^{(3)}} \tag{8}$$

where R is the number of fuzzy rules and y is the output of the FNFN model.

As described earlier, the number of tuning parameters for the FNFN model is known to be $(2 + 3 \times P) \times N \times R$, where N, R, and P denote the number of inputs, existing rules, and outputs, respectively. The proposed FNFN model can be demonstrated to be a universal uniform approximation by the Stone–Weierstrass theorem (Rudin, 1976) for continuous functions over compact sets.

3. Learning algorithms of the FNFN model

This section presents an online learning algorithm for constructing the FNFN model. The proposed learning algorithm comprises a structure learning phase and a parameter learning phase. Fig. 3 presents flow diagram of the learning scheme for the FNFN model. Structure learning is based on the entropy measure used to determine whether a new rule should be added to satisfy the fuzzy partitioning of input variables. Parameter learning is based on supervised learning algorithms. The backpropagation algorithm minimizes a given cost function by adjusting the link weights in the consequent part and the parameters of the membership functions. Initially, there are no nodes in the network except the input–output nodes, i.e., there are no nodes in the FNFN model. The nodes are created automatically as learning proceeds, upon the reception of online incoming training data in the structure and parameter learning processes. The rest of this section details the structure learning phase and the parameter learning phase. Finally, in this section, the stability analysis of the FNFN model based on the Lyapunov approach is performed to ensure that the convergence property holds.

3.1. Structure learning phase

The first step in structure learning is to determine whether a new rule should be extracted from the training data and to determine the number of fuzzy sets in the universe of discourse of each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule, in which m_{ij} and σ_{ij} represent the mean and variance of that cluster, respectively. For each incoming pattern x_i, the rule firing strength can be regarded as the degree to which the incoming pattern belongs to the corresponding cluster. The entropy measure between each data point and each membership function is calculated based on a similarity measure. A data point of closed mean will have lower entropy. Therefore, the entropy values between data points and current membership functions are calculated to determine whether or not to add a new rule. For computational efficiency, the entropy measure can be calculated using the firing strength from $u_{ij}^{(2)}$ as

$$EM_j = -\sum_{i=1}^{N} D_{ij}\log_2 D_{ij} \tag{9}$$

where $D_{ij} = \exp\left(u_{ij}^{(2)^{-1}}\right)$ and $EM_j \in [0, 1]$. According to (9), the measure is used to generate a new fuzzy rule, and new functional link bases for new incoming data are described as follows. The maximum entropy measure

$$EM_{\max} = \max_{1 \leqslant j \leqslant R_{(t)}} EM_j \tag{10}$$

is determined, where $R_{(t)}$ is the number of existing rules at time t. If $EM_{\max} < \overline{EM}$, then a new rule is generated, where $\overline{EM} \in [0, 1]$ is a prespecified threshold that decays during the learning process.

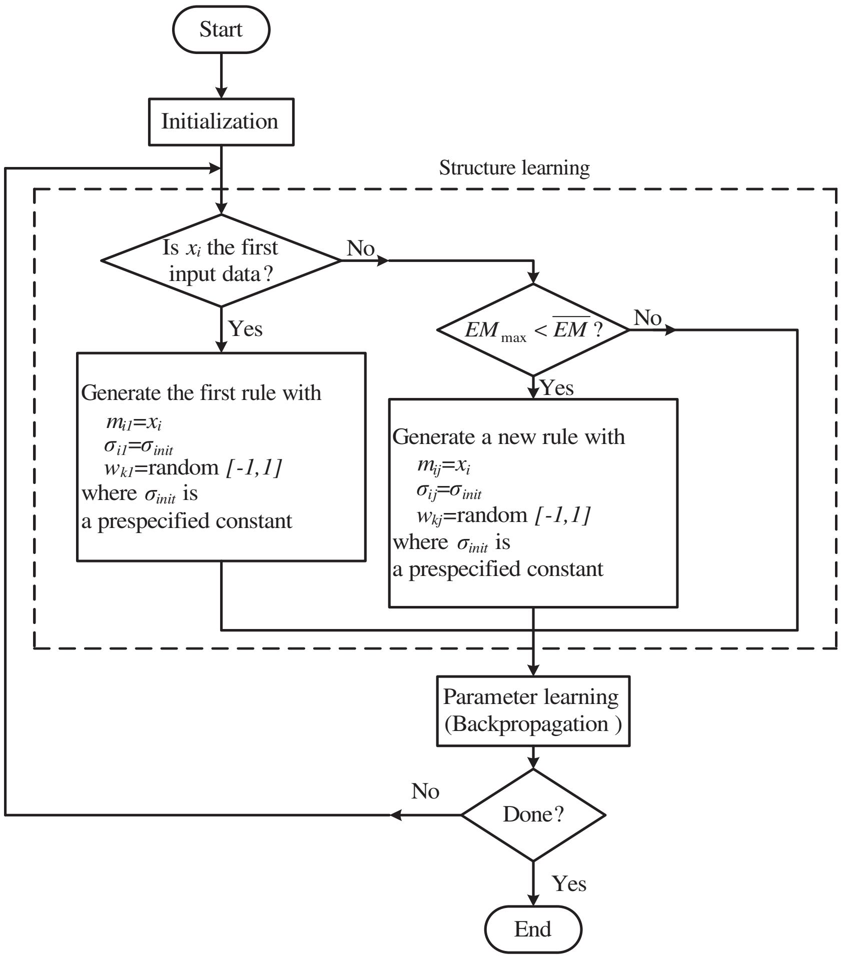
Start
Initialization
Structure learning
Is x_i the first input data?
No
Yes
Generate the first rule with $m_{i1}=x_i$ $\sigma_{i1}=\sigma_{init}$ w_{k1}=random [-1,1] where σ_{init} is a prespecified constant
$EM_{\max} < \overline{EM}$?
No
Yes
Generate a new rule with $m_{ij}=x_i$ $\sigma_{ij}=\sigma_{init}$ w_{kj}=random [-1,1] where σ_{init} is a prespecified constant
Parameter learning (Backpropagation)
Done?
No
Yes
End

Fig. 3. Flow diagram of the structure/parameter learning for the FNFN model.

In the structure learning phase, the threshold parameter $\overline{EM}$ is an important parameter. The threshold is set between zero and one. A low threshold leads to the learning of coarse clusters (i.e., fewer rules are generated), whereas a high threshold leads to the learning of fine clusters (i.e., more rules are generated). If the threshold value equals zero, then all the training data belong to the same cluster in the input space. Therefore, the selection of the threshold value $\overline{EM}$ will critically affect the simulation results. As a result of our extensive experiments and by carefully examining the threshold value $\overline{EM}$, which uses the range [0, 1], we concluded that there was a relationship between threshold value EM and the number of input variables (N). Accordingly, $\overline{EM} = \tau N$, where τ belongs to the range [0.26, 0.3]. Once a new rule has been generated, the next step is to assign the initial mean and variance to the new membership function and the corresponding link weight for the consequent part. Since the goal is to minimize an objective function, the mean, variance, and weight are all adjustable later in the parameter learning phase. Hence, the mean, variance, and weight for the new rule are set as

$$m_{ij}^{(R_{(t+1)})} = x_i \tag{11}$$

$$\sigma_{ij}^{(R_{(t+1)})} = \sigma_{init} \tag{12}$$

$$w_{kj}^{(R_{(t+1)})} = random[-1, 1] \tag{13}$$

where x_i is the new input and σ_{init} is a prespecified constant. The whole algorithm for the generation of new fuzzy rules and fuzzy sets in each input variable is as follows. No rule is assumed to exist initially.

```
• Step1: IF $x_i$ is the first incoming pattern THEN do
  {Generate a new rule
  with mean $m_{i1} = x_i$, variance $\sigma_{i1} = \sigma_{init}$,
  weight $w_{k1} = random[-1, 1]$
  where $\sigma_{init}$ is a prespecified constant
  }
  {Find $EM_{\max} = \max_{1 \leqslant j \leqslant R_{(T)}} EM_j$
  IF $EM_{\max} < \overline{EM}$
     do nothing
     ELSE
  {  $R(t+1) = R(t) + 1$
       generate a new rule
       with mean $m_{iR_{(t+1)}} = x_i$,
       variance $\sigma_{iR_{(t+1)}} = \sigma_{init}$ ,
       weight $w_{kR_{(t+1)}} = random[-1, 1]$
       where $\sigma_{init}$ is a prespecified constant.
     }
  }
```

3.2. Parameter learning phase

After the network structure has been adjusted according to the current training data, the network enters the parameter learning phase to adjust the parameters of the network optimally based on the same training data. The learning process involves determining the minimum of a given cost function. The gradient of the cost function is computed and the parameters are adjusted with the negative gradient. The backpropagation algorithm is adopted for this supervised learning method. When the single-output case is considered for clarity, the goal to minimize the cost function E is defined as

$$E(t) = \frac{1}{2}[y(t) - y^d(t)]^2 = \frac{1}{2}e^2(t) \tag{14}$$

where $y^d(t)$ is the desired output and $y(t)$ is the model output for each discrete time t. In each training cycle, starting at the input variables, a forward pass is adopted to calculate the activity of the model output $y(t)$.

When the backpropagation (BP) learning algorithm is adopted, the weighting vector of the FNFN model is adjusted such that the error defined in (14) is less than the desired threshold value after a given number of training cycles. The well-known backpropagation learning algorithm may be written briefly as

$$W(t+1) = W(t) + \Delta W(t) = W(t) + \left(-\eta \frac{\partial E(t)}{\partial W(t)}\right) \tag{15}$$

where, in this case, η and W represent the learning rate and the tuning parameters of the FNFN model, respectively.

Let $\mathbf{W} = [m, \sigma, w]^T$ T be the weighting vector of the FNFN model. Then, the gradient of error $E(\cdot)$ in (14) with respect to an arbitrary weighting vector W is

$$\frac{\partial E(t)}{\partial W} = e(t)\frac{\partial y(t)}{\partial W} \tag{16}$$

Recursive applications of the chain rule yield the error term for each layer. Then the parameters in the corresponding layers are adjusted. With the FNFN model and the cost function as defined in (14), the update rule for w_j can be derived as

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t) \tag{17}$$

where

$$\Delta w_{kj}(t) = -\eta_w \frac{\partial E}{\partial w_{kj}} = -\eta_w \cdot e \cdot \left(\frac{u_j^{(3)}\phi_k}{\sum_{j=1}^{R} u_j^{(3)}}\right)$$

Similarly, the update laws for m_{ij} and σ_{ij} are

$$m_{ij}(t+1) = m_{ij}(t) + \Delta m_{ij}(t) \tag{18}$$
$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \Delta\sigma_{ij}(t) \tag{19}$$

where

$$\Delta m_{ij}(t) = -\eta_m \frac{\partial E}{\partial m_{ij}} = -\eta_m \cdot e \cdot \left(\frac{u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}}\right) \cdot \left(\frac{2(u_i^{(1)} - m_{ij})}{\sigma_{ij}^2}\right)$$

$$\Delta \sigma_{ij}(t) = -\eta_\sigma \frac{\partial E}{\partial \sigma_{ij}} = -\eta_\sigma \cdot e \cdot \left(\frac{u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}}\right) \cdot \left(\frac{2(u_i^{(1)} - m_{ij})^2}{\sigma_{ij}^3}\right)$$

where η_w, η_m, and η_σ are the learning rate parameters of the weight, the mean, and the variance, respectively. In this study, both the link weights in the consequent part and the parameters of the membership functions in the precondition part are adjusted by using the backpropagation algorithm. Recently, many researchers (Juang & Lin, 1998; Wang & Mendel, 1993) tuned the consequent parameters using either LMS or recursive least squares (RLS) algorithms to obtain optimal parameters. However, they still used the backpropagation algorithm to adjust the precondition parameters.

4. Experimental results

4.1. Example 1: Iris data

The Iris data set (Fisher, 1936) contains 150 patterns that are distributed equally into three output species, Iris Setosa, Iris Versicolur and Iris Virginica. Each pattern consists of four input features: sepal length, sepal width, petal length, and petal width. We exploit these patterns to produce both the training data and testing data. In the experiment, 25 instances from each species were randomly selected as the training set (i.e., a total of 75 training patterns were used) and the remaining instances were used as the testing set. The data set was normalized to the range [0, 1], and the output y of the FNFN model was used with the following classification rules.

$$Iris = \begin{cases} \text{Setosa}, & \text{if } y < 1.5 \\ \text{Versicolour}, & \text{if } 1.5 \leqslant y < 2 \\ \text{Virginica}, & \text{if } y \geqslant 2.5 \end{cases} \tag{20}$$

The initial parameters $\eta_m = \eta_\sigma = \eta_w = 0.01$ and $\overline{EM} = [0.26, 0.3]$ were chosen. We repeated the experiment on 5 different training-test data sets that were obtained via a random process from the original Iris data. After learning, only 2–5 rules were generated in the FNFN model. Table 1 tabulated the results of the 5 different data sets in independent runs. The average testing accuracy rate of the FNFN model was 98.1%. Fig. 4 shows the input membership functions for Iris data classification.

The experiment calculated the training and testing accuracy rates by the FNFN and other existing models (the general MLP (Cho & Kim, 1995), the general NFN (Lin, Lin, & Shen, 2001), the self-organizing HCMAC (Lee, Chen, & Lu, 2003), the SANFIS (Wang & Lee, 2002), and a single CNFN classifier (Lin & Chen, 2003)). The comparison results are tabulated in Table 2. We can find that both the rates for FNFN model are higher than those of the other methods.

4.2. Example 2: Wisconsin Breast Cancer Diagnostic data

The Wisconsin Breast Cancer Diagnostic data set is available from the University of California, Irvine, via ftp://ftp.ics.uci.edu/pub/machine-learning-databases. Otherwise, for the Cleveland Heart Disease data, there are a few missing values in the data. In our experiments, these data sets were replaced by the average of the column (feature) regardless of the class labels. The data set contains 699 patterns distributed into two output classes, Benign (458 patterns) and Malignant (241 patterns). 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. Since there were 16 patterns containing missed values, only 683 patterns were used. In the experiment, half of the 683 patterns were used as the training data set and the remaining patterns were used as the test data set. We demarcated the output y of the FNFN model using the following rules.

$$class = \begin{cases} \text{Benign}, & \text{if } y < 1.5 \\ \text{Malignant}, & \text{if } 1.5 \leqslant y \end{cases} \tag{21}$$

Table 1
Experimental results of Iris data.

Experiment	1	2	3	4	5	Average
Accuracy (%)	97.3	98.7	98.7	97.3	98.7	98.1
Testing errors	2	1	1	2	1	1–2
Number of rules	4	3	5	2	3	2–5

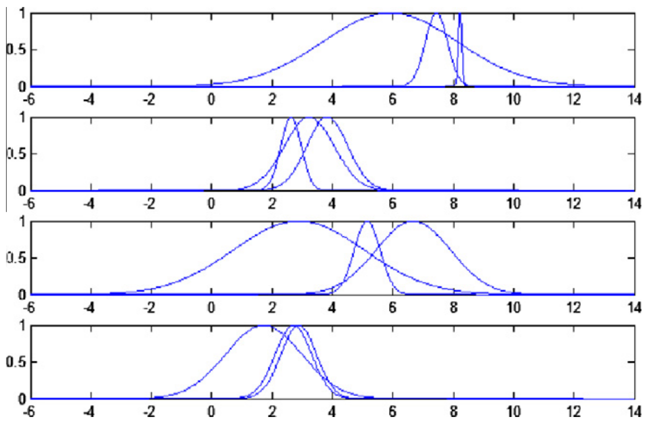

Fig. 4. Input membership functions for Iris data classification.

Table 2
Comparison of classification results on Iris data.

Models	Avg. training accuracy (%)	Avg. testing accuracy (%)
MLP (Cho and Kim, 1995)	98.5	94.7
NFN (Lin et al., 2001)	98	97
Self-organizing HCMAC (Lee et al., 2003)	98.6	97.3
SANFIS (Wang and Lee, 2002)	98.6	97.3
CNFN (Lin et al., 2003)	99.3	97.3
FNFN	99.5	98.1

We set parameters $\eta_m = \eta_\sigma = \eta_w = 0.01$ and $\overline{EM} = [0.26, 0.3]$ as initial values. The experiments were performed on 5 different training-test data sets that were obtained via a random process from the original data. After learning, there were 1–2 rules generated in the FNFN model. Table 3 shows the results of the 5 different data sets in independent runs. The average testing accuracy rate of the FNFN model was 98.3%.

We compare the performance of our model with that of other existing models (the FEBFC (Lee et al., 2001), the self-organizing HCMAC (Lee et al., 2003), the general MLP (Cho & Kim, 1995), the SANFIS (Wang & Lee, 2002), and the general NFN (Lin et al., 2001)), and the results are tabulated in Table 4. The averaged recognition rate of FNFN model is better than the other models.

Table 3
Experimental results of Wisconsin Breast Cancer Diagnostic data.

Experiment	1	2	3	4	5	Average
Accuracy (%)	97.7	98.2	97.7	98.2	99.7	98.3
Testing errors	8	6	8	6	1	1–8
Number of rules	1	2	1	2	2	1–2

Table 4
Comparison of classification results on Wisconsin Breast Cancer Diagnostic data.

Models	Avg. recognition rate (%)
FEBFC (Lee et al., 2001)	94.7
Self-organizing HCMAC (Lee et al., 2003)	95.7
MLP (Cho and Kim, 1995)	95.7
SANFIS (Wang and Lee, 2002)	96
NFN (Lin et al., 2001)	98
FNFN	98.3

Table 5
Experimental results on Wine data.

Experiment	1	2	3	4	5	Average
Accuracy (%)	98.9	98.9	99.4	99.4	98.9	99.1
Testing errors	2	2	1	1	2	1–2
Number of rules	1	1	1	2	1	1–2

Table 6
Comparison of classification results on Wine data.

Models	Average recognition rate (%)	Number of rules
Corcoran and Sen (1994)	99.5	60
Ishibuchi et al. (1999)	98.5	60
Setnes and Roubos (2000)	98.3	3
FNFN	99.1	1–2

4.3. Example 3: Wine data

The wine classification data set contains 178 wines that are brewed in the same region of Italy but derived from three different cultivars. Each pattern consists of 13 continuous features: alcohol, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavonoids, nonflavonoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines and proline. We demarcated the output y of the FNFN model using the following rules.

$$class = \begin{cases} \text{Type1}, & \text{if } y < 1.5 \\ \text{Type2}, & \text{if } 1.5 \leqslant y < 2.5 \\ \text{Type3}, & \text{if } y \leqslant 2.5 \end{cases} \tag{22}$$

For this problem, the initial parameters $\eta_m = \eta_\sigma = \eta_w = 0.01$ and entropy = [0.26, 0.3] were chosen. Five different training-test data sets were used in this experiment. These data sets were randomly selected from the original data. After learning, there were 1–2 rules generated in the FNFN model. Table 5 shows the results. The average testing accuracy rate of the FNFN model was 99.1%.

Corcoran and Sen (1994) applied all the 178 samples for learning 60 nonfuzzy IF-THEN rules in a real-coded genetic-based method learning approach. They used a population of 1500 individuals and applied 300 generations, with full replacement, to come up with the following result for ten independent trials: average classification rate 99.5%. Ishibuchi, Nakashima, and Murata (1999) proposed an integer-coded GA and grid-partitioning to design a fuzzy classifier with 60 fuzzy rules from the 178 patterns. Their population contained 100 individuals and they applied 1000 generations, with full replacement, to come up with the following result for ten independent trials: average classification rate 98.5%. Setnes and Roubos (2000) applied a real-coded GA and c-means clustering algorithm on all the available 178 patterns to design a TSK model as a classifier. Table 6 shows the comparison of performance between FNFN and other fuzzy, neural-network, and neural fuzzy classifiers. The recognition rates of FNFN outperform the listed classifiers except Cornran et al. Although the averaged recognition rate of FNFN model is a little worse than Cornran et al. FNFN model uses far fewer rules.

5. Conclusions

In this study of classification applications, we have proposed a functional neural fuzzy network (FNFN). The FNFN model uses the functional link neural network (FLNN) as the consequent part of the fuzzy rules. Therefore, the FNFN can form the consequent part of a nonlinear combination of the input variables to be approximated more effectively. In addition, the FNFN can automatically construct its structures and adjust free parameters by performing online learning schemes concurrently. The FNFN has the advantages of higher classification accuracy and less rule number. Finally, three examples have showed that the proposed FNFN has better performance than that of other methods.

References

Cho, S. B., & Kim, J. H. (1995). Combining multiple neural networks by fuzzy integral and robust classification. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 25*, 380–384.

Corcoran, A. L. & Sen, S. 1994. Using real-valued genetic algorithms to evolve rule sets for classification. In *Proceedings of 1st IEEE conference on evolutionary computation* (pp. 120–124). Orlando, FL.

Duda, P. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7*, 179–188.

Ishibuchi, H., Nakashima, T., & Murata, T. (1999). Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 29*(Oct.), 601–618.

Jang, J.-S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 23*(3), 665–685.

Juang, C. F., & Lin, C. T. (1998). An online self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems, 6*(1), 12–31.

Lee, H. M., Chen, C. M., Chen, J. M., & Jou, Y. L. (2001). An efficient fuzzy classifier with feature selection based on fuzzy entropy. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 31*(Jun), 426–432.

Lee, H. M., Chen, C. M., & Lu, Y. F. (2003). A self-organizing HCMAC neural-network classifier. *IEEE Transactions on Neural Networks, 14*, 12–15.

Li, C., & Lee, C. Y. (2003). Self-organizing neuro-fuzzy system for control of unknown plants. *IEEE Transactions on Fuzzy Systems, 11*(1), 135–150.

Lin, C. J., & Chen, C. H. (2003). Nonlinear system control using compensatory neurofuzzy networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E86-A*, 2309–2316.

Lin, C. T., & Lee, C. S. G. (1996). *Neural fuzzy systems: A neuro-fuzzy synergism to intelligent system. Englewood Cliffs*. NJ: Prentice-Hall.

Lin, C. J., & Lin, C. T. (1997). An ART-based fuzzy adaptive learning control network. *IEEE Transactions on Fuzzy Systems, 5*(4), 477–496.

Lin, F. J., Lin, C. H., & Shen, P. H. (2001). Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. *IEEE Transactions on Fuzzy Systems, 9*(Oct.), 751–759.

Lin, W. S., Tsai, C. H., & Liu, J. S. (2001). Robust neuro-fuzzy control of multivariable systems by tuning consequent membership functions. *Fuzzy Sets and Systems, 124*(2), 181–195.

Mitra, S., & Hayashi, Y. (2000). Neuro-fuzzy rule generation: Survey in soft computing framework. *IEEE Transactions on Neural Networks, 11*(3), 748–768.

Mizutani, E. & Jang, J.-S. R. (1995). Coactive neural fuzzy modeling. In *Proceedings of international conference on neural network* (Vol. 2, pp. 760–765). Perth, WA, Australia.

Pao, Y. H. (1989). *Adaptive pattern recognition and neural networks*. Reading, MA: Addison-Wesley.

Pao, Y. H., Phillips, S. M., & Sobajic, D. J. (1992). Neural-net computing and intelligent control systems. *International Journal of Control, 56*(2), 263–289.

Patra, J. C., & Pal, R. N. (1995). A functional link artificial neural network for adaptive channel equalization. *Signal Processing, 43*(May), 181–195.

Patra, J. C., Pal, R. N., Chatterji, B. N., & Panda, G. (1999). Identification of nonlinear dynamic systems using functional link artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 29*(2), 254–262.

Rudin, W. (1976). *Principles of mathematical analysis* (3rd ed.). New York: McGraw-Hill.

Setiono, R., & Liu, H. (1997). Neural-network feature selector. *IEEE Transactions on Neural Network*(May), 654–662.

Setnes, M., & Roubos, H. (2000). GA-fuzzy modeling and classification: Complexity and performance. *IEEE Transactions on Fuzzy Systems, 8*(Oct.), 509–522.

Sun, F., Sun, Z., Li, L., & Li, H. X. (2003). Neuro-fuzzy adaptive control based on dynamic inversion for robotic manipulators. *Fuzzy Sets and Systems, 134*(1), 117–133.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, SMC-15*(1), 116–132.

Takagi, H., Suzuki, N., Koda, T., & Kojima, Y. (1992). Neural networks designed on approximate reasoning architecture and their application. *IEEE Transactions on Neural Networks, 3*(5), 752–759.

Wang, J. S., & Lee, C. S. G. (2002). Self-adaptive neuro-fuzzy inference systems for classification applications. *IEEE Transactions on Fuzzy Systems, 10*, 790–801.

Wang, L. X., & Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 22*(6), 1414–1427.

Wang, L. X., & Mendel, J. M. (1993). Fuzzy adaptive filters, with application to nonlinear channel equalization. *IEEE Transactions on Fuzzy Systems, 1*(3), 161–170.