An On-Line ICA-Mixture-Model-Based Self-Constructing Fuzzy Neural Network

Chin-Teng Lin, *Fellow, IEEE*, Wen-Chang Cheng, and Sheng-Fu Liang

***Abstract*—This paper proposes a new fuzzy neural network (FNN) capable of parameter self-adapting and structure self-constructing to acquire a small number of fuzzy rules for interpreting the embedded knowledge of a system from the given training data set. The proposed FNN is inherently a modified Takagi-Sugeno-Kang (TSK)-type fuzzy-rule-based model with neural network's learning ability. There are no rules initiated at the beginning and they are created and adapted through an on-line learning processing that performs simultaneous structure and parameter identification. In the structure identification of the precondition part, the input space is partitioned in a flexible way according to the newly proposed on-line independent component analysis (ICA) mixture model. The input space is thus represented by linear combinations of independent, non-Gaussian densities. The first input training pattern is assigned to the first rule initially by the on-line ICA mixture model. Afterwards, some additional significant terms (input variables) selected by the on-line ICA mixture model will be added to the consequent part (forming a liner equation of input variables) incrementally or create a new rule in the learning processing. The combined precondition and consequent structure identification scheme can make the network grow dynamically and efficiently. In the parameter identification, the consequent parameters are tuned by the backpropagation rule and the precondition parameters are turned by the on-line ICA mixture model. Both the structure and parameter identifications are done simultaneously to form a fast learning scheme. The derived on-line ICA mixture model also provide a natural linear transformation for each input variable to enhance the knowledge representation ability of the proposed FNN and reduce the required rules and achieve higher accuracy efficiently. In order to demonstrate the performance of the proposed FNN, several experiments covering the areas of system identification, classification, and image segmentation are carried out. Our experiments show that the proposed FNN can achieve significant improvements in the convergence speed and prediction accuracy with simpler network structure.**

***Index Terms*—Backpropagation rule, Gaussian mixture model, non-Gaussian mixture model, principal component analysis, Takagi-Sugeno-Kang (TSK) fuzzy rules.**

Manuscript received January 9, 2004; revised June 16, 2004. This work was supported in part by the Ministry of Education, Taiwan, R.O.C., under Grant EX-91-E-FAOE-4–4 and Ministry of Economic Affairs, Taiwan, R.O.C., under Grant 93-17-A-02-S1-032. This paper was recommended by Associate Editor Y. Nishio.

C.-T. Lin and S.-F. Liang are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C., and also with the Brain Research Center, University System of Taiwan, Taipei 112, Taiwan, R.O.C. (e-mail: ctlin@mail.nctu.edu.tw; sfliang@mail.nctu.edu.tw).

W.-C. Cheng is with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C. (e-mail: wccheng@mail.hit.edu.tw).

Digital Object Identifier 10.1109/TCSI.2004.840110

I. Introduction

IN RECENT years, the fuzzy neural network (FNN) has found widely in industrial, commercial, and image processing applications that require the analysis of uncertain and imprecise information due to its nice merge of the fuzzy inference system (FIS) and neural network (NN), which are complementary technologies in the design of adaptive intelligent systems. FIS is a popular computing framework based on the concept of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. With crisp inputs and outputs, FIS implements a nonlinear mapping from its input space to output space by a number of if-then rules. To build a FIS, we have to specify the fuzzy sets, fuzzy operators and the knowledge (rule) base. The selection of fuzzy if-then rules often relies on a substantial amount of heuristic observation to express proper strategy's knowledge. However, it is difficult for human experts to examine all the input-output data from a complex system to find a number of proper rules for the FIS.

Artificial neural network (ANN) learns from scratch by adjusting the interconnections between layers. A valuable property of ANN is that of generalization, whereby a trained network is able to provide a correct matching in the form of output data for a set of previously unseen input data. For constructing an ANN, the user needs to specify the architecture and learning algorithm. Learning mechanism of ANN does not rely on human expertise. Due to the homogenous structure of ANN, it is difficult to extract structured knowledge from either the weights or the configuration of the ANN. For many practical problems, *a priori* knowledge is usually obtained from human experts and it is more appropriate to express the knowledge as a set of fuzzy if-then rules. However, it is not easy to encode prior knowledge into an ANN.

To cope with the respective difficulties generated by ANN and FIS, integrating them into a functional system, i.e., FNN, has attracted the growing interest of researchers due to the growing need of adaptive intelligent systems to meet the real world requirements. The key advantage of the FNN approach over traditional ones lies on that the former doesn't require a mathematical description of the system while modeling. Moreover, in contrast to pure ANN or FIS methods, the FNN possesses both of their advantages; it brings the low-level learning and computational power of ANN into FIS and provides the high-level human-like thinking and reasoning of FIS into ANN [1]–[5]. The FNN solves the problems successfully which are encountered in many areas such as control, communications, pattern recognition, etc. [6]–[9].

One important task in the structure identification of a FNN is the partition of the input–output space, which influences the

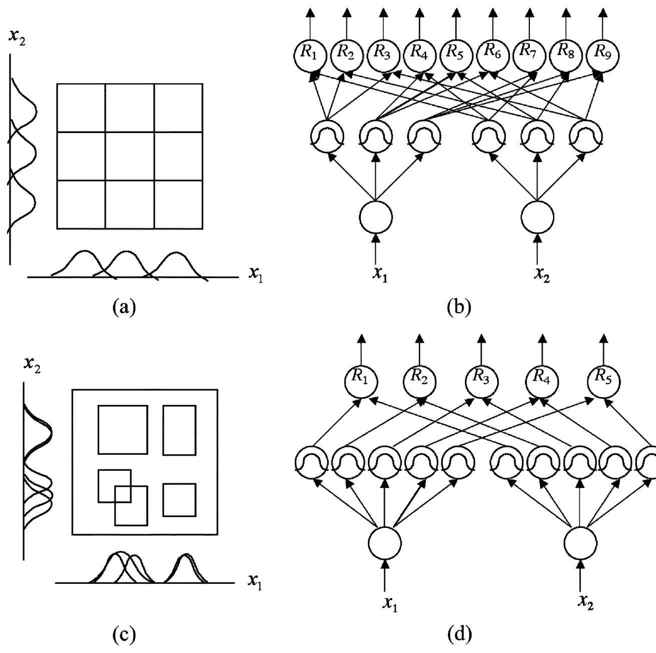

x_2
x_1
(a)
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9
x_1 x_2
(b)
x_2
x_1
(c)
R_1 R_2 R_3 R_4 R_5
x_1 x_2
(d)

Fig. 1. Fuzzy partitions of two-dimensional input space. (a) Grid-type partitioning. (b) If-then rules based on grid-type partitioning. (c) Clustering-type partitioning. (d) If-then rules based on clustering-type partitioning.

number of generated fuzzy rules. Efficient partition of input-output data may result in faster convergence and better performance for FNN. The most direct way is to partition the input space into grid types and each grid represents a fuzzy if-then rule [see Fig. 1(a)]. It is called grid-based partitioning. The major problem of such kind of partition is that the number of fuzzy rules increases exponentially if the number of input variables or that of partition increases. This is the so-called problem of curse of dimensionality. To cope with this problem, a clustering-based partition is employed which does reduce the number of generated rules [10]–[13]. The cluster-based algorithm provides a more flexible way for space partition to avoid drastic increase of fuzzy rules and thus generates the corresponding rule base with appropriate number of rules. For example, by observing the projected membership functions in Fig. 1(c), although the number of membership functions in Fig. 1(d) is more than that in Fig. 1(b), there are only five rules in Fig. 1(d); however, there are nine rules in Fig. 1(b). By observing the projected membership functions in Fig. 1(c), we find that some membership functions projected from different clusters have high similarity degrees. These highly similar membership functions should be combined to reduce the number of membership functions.

There are several methods for input space partitioning, which are to cluster the input training vectors in the input space, such as Kohonen learning rule, hyperbox method, product-space partitioning, fuzzy c-mean method, EM algorithm, etc. [15]–[18]. These methods are based on Gaussian membership functions. In general, the observed data can be categorized into several mutually exclusive classes [20], and the data in each class can be modeled as multivariate Gaussian, called the Gaussian mixture model (GMM). GMMs are widely used throughout the fields of machine learning and statistics. Despite their popularity, GMMs suffer from several serious drawbacks [14]. One major drawback is that if the dimension d of the problem space increases, the size of each covariance matrix, d^2, becomes prohibitively large. This problem has been solved by Tipping and Bishop [21] who replaced each Gaussian with a probabilistic principal component analysis (PCA) model. This allowed the dimensionality of each covariance to be effectively reduced while maintaining the richness of the model class. However, some recent research approaches try to reduce the information redundancy by capturing the statistical structure in observed data that is beyond second-order information. Independent component analysis (ICA) is a technique that exploits higher order statistical structure of the data. This method has recently gained attention due to its successful applications to signal processing problems including speech enhancement, discrete signal processing, image processing, etc. The goal of ICA is to linearly transform the data such that the transformed variables are as statistically independent from each other as possible [22]–[26]. This means that the value of any one of the components gives no information on the values of the other components. Basically, it finds directions in the input space which lead to independent components instead of just uncorrelated ones, as PCA does, so it reduces not only the number of rules but also the number of membership functions under a prespecified accuracy requirement dynamically.

Another drawback of GMMs is that it is based on Gaussian functions. In some situation, it could not be separated from each other. It is generalized by assuming the data in each class are generated by a linear combination of independent non-Gaussian sources [12], [14], [19], [27]. This model is called the ICA mixture model. This allows modeling of classes with non-Gaussian structure; e.g., platykurtic or leptokurtic probability density functions are used for learning and the gradient ascent method is used to maximize the log-likelihood function. In previous applications, this approach showed improved performance in data classification problems [28] and in learning efficient codes for representing different types of images [12], [19]. The advantage of this model is that it provides greater flexibility in modeling structure and in finding more features compared with GMMs or standard ICA algorithms. Although the ICA mixture model has many advantages in data clustering, the proper number of clusters should be given beforehand. Once the cluster number is determined, we have to stick to it until independent axes are obtained. In reality, the correct or proper number of clusters is usually unknown, and improper assignment of cluster number will affect the representation of learned independent axes a lot. Moreover, the existing ICA mixture model scheme is only suitable for off-line instead of on-line operation. Hence, to adopt this scheme, a large amount of representative data should be collected in advance, and the learning of ICA mixture model usually spends a lot of time through trial and errors.

To attack the aforementioned problems, in this paper we derive an on-line ICA mixture model to provide better and on-line partitioning of the input-output space for FNN, and propose a novel FNN model called ICA-mixture-model-based self-constructing FNN. This FNN can grow its structure and tune its parameters on the fly efficiently based on the derived on-line ICA mixture model. Several experiments covering the areas of system identification, classification, and image segmentation have been carried out based on the proposed FNN. These experiments show that the proposed FNN can achieve significant improvements in convergence speed and prediction accuracy.

II. Proposed On-Line ICA Mixture Model

The ICA mixture model is an unsupervised classification algorithm derived by modeling observed data as a mixture of several mutually exclusive classes that are described by linear combinations of independent, non-Gaussian densities [27]. It is used for learning a complete set of basis functions and these basis functions can be learned simultaneously.

Assume that the data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T\}$ are drawn independently and are to be clustered into the total number of classes, K, where K is assumed to be known in advance, T is the total number of data vectors, and each data vector $\mathbf{x}_t \in \mathbf{X}$ is n-dimensional. The component densities are non-Gaussian and the data within each class are presented by

$$\mathbf{x}_t = \mathbf{A}_k \mathbf{s}_k + \mathbf{m}_k, k = 1, 2, \ldots, K \tag{1}$$

where $\mathbf{A}_k$ is a $n \times m$ scalar matrix, $\mathbf{m}_k$ is the bias vector for class k, and $\mathbf{s}_k$ is called the source vector, i.e., the coefficients for each basis function, where n and m are the dimensions of the input vector $\mathbf{x}$ and the source vector $\mathbf{s}$, respectively. For simplicity, we consider the case where the number of sources is equal to the number of linear combinations. According to the values of $\mathbf{A}_k, \mathbf{m}_k$, and $\mathbf{s}_k$, there are K ways for presenting $\mathbf{x}_t$. However, we assume mutually exclusive classes and maximum-likelihood estimation results in one model that fits the data the best [16].

The likelihood of data is given by the joint density as

$$p(\mathbf{X}) = \prod_{t=1}^{T} p(\mathbf{x}_t) \tag{2}$$

where the mixture density is

$$p(\mathbf{x}_t) = \sum_{k=1}^{K} p(\mathbf{x}_t \mid C_k) p(C_k) \tag{3}$$

where C_k denotes the class k. The goal of ICA mixture model algorithm is to determine the parameters for each class, $(\mathbf{A}_k, \mathbf{m}_k)$, by using the maximum log-likelihood method. Therefore, the rule to update the basis function $\mathbf{A}_k$ for each class can be written as

$$\mathbf{A}_k(t+1) = \mathbf{A}_k(t) + p(C_k \mid \mathbf{x}_t) \frac{\partial}{\partial \mathbf{A}_k(t)} \log p(\mathbf{x}_t \mid C_k) \tag{4}$$

where the log-likelihood of the data for each class is

$$\log p(\mathbf{x}_t \mid C_k) = \log p(\mathbf{s}_k) - \log(\det |\mathbf{A}_k|) \tag{5}$$

and the probability for each class given the data vector $\mathbf{x}_t$ is

$$p(C_k \mid \mathbf{x}_t) = \frac{p(\mathbf{x}_t \mid C_k) p(C_k)}{p(\mathbf{x}_t)}. \tag{6}$$

The updating rule for the basis terms is

$$\mathbf{m}_k = \frac{\sum_{i=1}^{T} \mathbf{x}_i p(C_k \mid x_i)}{\sum_{i=1}^{T} p(C_k \mid x_i)} \tag{7}$$

where t is the data index $(t = 1, 2, \ldots, T)$.

Furthermore, for the automatic switching between super-Gaussian and sub-Gaussian models, a switching matrix $O_{k,l}$ shown in (9) can be used; i.e., source distributions are more peaked or less peaked than the Gaussian

$$\begin{aligned} &\text{super}-\text{Gaussian } (O_{k,l} = 1)\text{: } \log p(\mathbf{s}_k) \\ &\quad \propto -\sum_{l=1}^{n} |s_{k,l}|, \\ &\text{sub}-\text{Gaussian}(O_{k,l} = -1)\text{: } \log p(\mathbf{s}_k) \\ &\quad \propto \sum_{l=1}^{n} \left(\log(\cosh(s_{k,l})) - \frac{s_{k,l}^2}{2} \right) \end{aligned} \tag{8}$$

where n is dimensions of the source, $s_{k,l}$ is the lth dimension of the source in the kth class, and $O_{k,l}$ is an indicator which allows for automatic switching between super-Gaussian and sub-Gaussian models

$$O_{k,l} = \operatorname{sign}\left[E\{\operatorname{sec h}^2(s_{k,l})\} E\left\{s_{k,l}^2\right\} - E\{(\tanh(s_{k,l})) s_{k,l}\}\right]. \tag{9}$$

The above ICA mixture model is good for clustering, but it requires that a correct or proper cluster number should be given in advance for a set of training data, which is usually unknown in reality. To make the choice of proper cluster number automatic and to let the ICA mixture model useful for on-line clustering, an on-line ICA mixture model is first derived in this section. In this model, there is no cluster initially. When the first data vector is fed into it, the first cluster is generated. Then for the following incoming data vector (pattern), the on-line ICA mixture model will determine if this pattern belongs to the first (or existing) cluster or another new cluster should be generated to accommodate this new pattern. To make this decision, we let the log-likelihood value calculated in (5) to represent the degree to which the newly incoming pattern $\mathbf{x}_t$ belongs to the jth cluster, i.e., $F^j(\mathbf{x}_t) = \log p(\mathbf{x}_t | C_j)$. Then we define

$$F^{J_{\max}}(\mathbf{x}_t) = \max_{1 \le j \le J(t)} F^j(\mathbf{x}_t) \tag{10}$$

where the superscript $J_{\max}$ is the index for the maximum log-likelihood value among all log-likelihood values and $J(t)$ is the total number of clusters at time t. If $F^{J_{\max}}(\mathbf{x}_t) \ge F$, the corresponding new incoming pattern is added to the existed cluster with index $J_{\max}$ and the parameters of this cluster are updated properly, where F is a given threshold value. In this case, no new cluster is generated. If $F^{J_{\max}}(\mathbf{x}_t) < F$, a new cluster will be generated to accommodate this new pattern. The threshold value F is obtained empirically and it is a negative value.

In the rest of this section, we shall derive the details of the updating rules for the proposed on-line ICA mixture model. Assume the number of clusters at time t is $J(t)$. Then, the mixture probability at time t is

$$p(\mathbf{x}_t) = \sum_{j=1}^{J(t)} p(\mathbf{x}_t \mid C_j) p_t(C_j). \tag{11}$$

Therefore, the posterior probability is

$$p(C_j \mid \mathbf{x}_t) = \frac{p(\mathbf{x}_t \mid C_j) p_t(C_j)}{p(\mathbf{x}_t)} \tag{12}$$

where $p_{t-1}(C_j)$ is the prior probability at the preceding time step, which can be obtained by former calculation result of the jth cluster. Hence, $p_t(C_j)$ at this moment can be calculated by the following:

$$\begin{aligned} p_t(C_j) &= \frac{1}{t}\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i) \\ &= \frac{1}{t}\left[\sum_{i=1}^{t-1} p(C_j \mid \mathbf{x}_i) + p(C_j \mid \mathbf{x}_t)\right] \\ &= \frac{1}{t}[(t-1)p_{t-1}(C_j) + p(C_j \mid \mathbf{x}_t)]. \end{aligned} \tag{13}$$

Using the above results, we can obtain the following updating rules for the parameters of each cluster, including basis matrix ($\mathbf{B}_j = \mathbf{A}_j^{-1}$), mean ($\mathbf{m}_j$), and the criterion of data distribution ($O_{j,p}, j = 1, \ldots, J(t)$, and $p = 1, \ldots, n$) that determine if the distribution of data is super-Gaussian or sub-Gaussian with the previous calculation results. They are defined as follows:

$$\begin{aligned} \mathbf{m}_j(t) &\cong \frac{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)} \\ &= \frac{\sum_{i=1}^{t-1} p(C_j \mid \mathbf{x}_i)\mathbf{x}_i + p(C_j \mid \mathbf{x}_t)\mathbf{x}_t}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)}, \\ \mathbf{Cov}_j(t) &\cong \frac{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)(\mathbf{x}_i - \mathbf{m}_j(t))(\mathbf{x}_i - \mathbf{m}_j(t))^T}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)}. \end{aligned} \tag{14}$$

By substituting the term, $\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i) = tp_t(C_j)$, into (14), we can rewrite (14) as follows:

$$\begin{aligned} \mathbf{m}_j(t) &\cong \frac{1}{tp_t(C_j)}[(t-1)p_{t-1}(C_j)\mathbf{m}_j(t-1) + p(C_j \mid \mathbf{x}_t)\mathbf{x}_t] \\ \mathbf{Cov}_j(t) &\cong \frac{1}{tp_t(C_j)}\left[(t-1)p_{t-1}(C_j)\mathbf{Cov}_j(t-1)\right. \\ &\quad + (t-1)p_{t-1}(C_j)\mathbf{m}_j(t-1)\mathbf{m}_j(t-1)^T \\ &\quad \left. + p(C_j \mid \mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T\right] - \mathbf{m}_j(t)\mathbf{m}_j(t)^T. \end{aligned} \tag{15}$$

Let $O_{j,p}(t)$ be defined as the function of criterion which allows for automatic switching between super-Gaussian and sub-Gaussian models and then (9) can be further derived as

$$O_{j,p}(t) = \operatorname{sign}\left\{\frac{T_1(t)T_2(t)}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)} - \frac{T_3(t)}{\sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)}\right\} \tag{16}$$

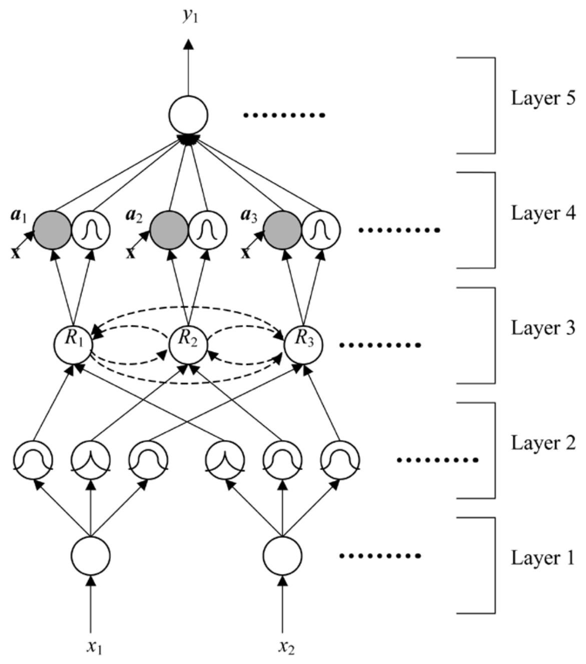


Fig. 2. Structure of the proposed on-line ICA-mixture-model-based FNN.

where

$$\begin{aligned} T_1(t) &= \sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)\sec\mathrm{h}^2(s_{j,p}(i)) \\ &= T_1(t-1) + p(C_j \mid \mathbf{x}_t)\sec\mathrm{h}^2(s_{j,p}(t)) \\ T_2(t) &= \sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)(s_{j,p}(i))^2 \\ &= T_2(t-1) + p(C_j \mid \mathbf{x}_t)s_{j,p}(t) \\ T_3(t) &= \sum_{i=1}^{t} p(C_j \mid \mathbf{x}_i)s_{j,p}(i)\tan\mathrm{h}^2(s_{j,p}(i)) \\ &= T_3(t-1) + p(C_j \mid \mathbf{x}_t)s_{j,p}(t)\tan\mathrm{h}^2(s_{j,p}(t)). \end{aligned} \tag{17}$$

Finally, the independent axes $\mathbf{B}_j(t)$ representing the axis of the jth cluster can be obtained by the following updating rule:

$$\mathbf{B}_j(t) = \mathbf{B}_j(t-1) + p(C_j \mid \mathbf{x}_t)\frac{\partial}{\partial \mathbf{B}_j(t-1)}\log p(\mathbf{x}_t \mid C_j). \tag{18}$$

III. Structure of the On-Line ICA Mixture-Model-Based FNN

In this section, a novel self-constructing FNN is developed based on the on-line ICA mixture model derived in the last section. The structure of the proposed FNN is shown in Fig. 2. This five-layered network realizes a FIS of the following form:

$$\begin{aligned} \text{Rule } i: \quad & \text{IF } x_1 \text{ is } A_1^i \text{ and } \ldots x_j \text{ is } A_j^i \ldots \text{and } x_n \text{ is } A_n^i, \\ & \text{THEN } y_i \text{ is } m_{0i} + a_{1i}x_1 + \cdots + a_{ji}x_j + \ldots \end{aligned} \tag{19}$$

where the current input data vector is $\mathbf{x}_t = [x_1, \ldots, x_n]^T$, n is the number of dimension, A_j^i is a fuzzy set, m_{0i} is the center of a symmetric membership function on y_i, and a_{ji} is a consequent parameter. It is noted that unlike the traditional TSK model where all the input variables are used in the output linear equation, only the significant ones are used in the proposed FNN; i.e., some a_{ji}'s in the above fuzzy rules are zero.

The FNN consists of nodes, each of which has some finite "fan-in" of connections represented by weight values from other nodes and "fan-out" of connections to other nodes. Associated with the fan-in of a node is an integration function f, which serves to combine information, activation, or evidence from other nodes. This function provides the net input for this node

$$\text{node-input} = f\left[u_1^{(k)}, u_2^{(k)}, \ldots, u_p^{(k)}; w_1^{(k)}, w_2^{(k)}, \ldots, w_p^{(k)}\right] \tag{20}$$

where $u_1^{(k)}, u_2^{(k)}, \ldots, u_p^{(k)}$ are inputs to this node and $w_1^{(k)}, w_2^{(k)}, \ldots, w_p^{(k)}$ are the associated link weights. The superscript (k) in (20) indicates the layer number. This notation will also be used in the following equations. A second action of each node is to output an activation value as a function of its node input

$$\begin{aligned}\text{node-output}^{(k)} &= o_i^{(k)} \\ &= a^{(k)}(\text{node-input}) \\ &= a^{(k)}(f)\end{aligned} \tag{21}$$

where $a(\cdot)$ denotes the activation function. We shall next describe the functions of the nodes in each of the five layers of the proposed FNN.

Layer 1: No computation is done in this layer. Each node in this layer, which corresponds to one input variable, only transmits input values to the next layer directly. That is,

$$\begin{aligned} f &= u_i^{(1)} \\ o_i^{(1)} &= a^{(1)}(f) = f.\end{aligned} \tag{22}$$

From the above equation, the link weight in layer one $[w_i^{(1)}]$ is unity.

Layer 2: Each node in this layer corresponds to one linguistic value ("small", "large", etc.) of one of the input variables in Layer 1. In other words, the membership value which specifies the degree to which an input value belongs a fuzzy set is calculated in Layer 2. In contrast to the types of membership functions used normally, such as triangular, trapezoidal, or Gaussian functions, the membership functions are determined by the on-line ICA mixture model in the proposed FNN.

In this layer, the output $\mathbf{x}$ from Layer 1 is projected into the independent axes obtained by the on-line ICA mixture model (as shown in Fig. 3) such that

$$\mathbf{s}_j = \mathbf{B}_j \mathbf{x}_j \tag{23}$$

where $\mathbf{x}_j = \mathbf{x} - \mathbf{m}_j, \mathbf{B}_j$ and $\mathbf{m}_j$ are the basis matrix and mean vector, respectively, determined by the on-line ICA mixture model, $j = 1, 2, \ldots, J(t)$, and $J(t)$ is the number of clusters at time t. That is, if the input data are classified into $J(t)$ clusters, the number of learned fuzzy rules will be $J(t)$.

With the choice of non-Gaussian membership function, the operation performed in this layer is

$$\begin{aligned}\mathbf{u}_i^{(2)} &= \mathbf{s}_i \\ f\left[u_{ij}^{(2)}\right] &= p\left(u_{ij}^{(2)}\right)\end{aligned} \tag{24}$$

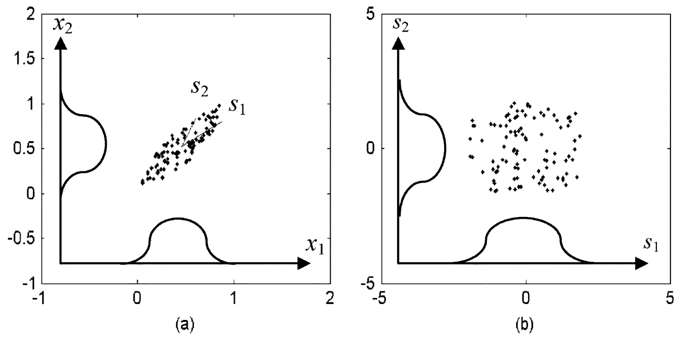


Fig. 3. Input space transformation by the on-line ICA mixture model in the structure learning of the proposed FNN. (a) The regions covered by the original axes. (b) The regions covered by the independent axes obtained by the on-line ICA mixture model.

where

$$p(u_{ij}) \propto \exp\left(-\log(\cosh(u_{ij})) - \frac{u_{ij}^2}{2}\right), \quad \text{for super-Gaussian}$$

$$p(u_{ij}) \propto \exp\left(\log(\cosh(u_{ij})) - \frac{u_{ij}^2}{2}\right), \quad \text{for sub-Gaussian}$$

$$o_i^{(2)} = a^{(2)}(f) = f \tag{25}$$

where u_{ij} is the transformed value of the jth term of the ith input variable $\mathbf{x}_i$. The transformation can be regarded as a linear combination of the original variables. With the transformation of input coordinates, the rule format in (19) should be modified as

$$\begin{aligned}\text{Rule } i\text{: IF } s_{i1} &= \sum_{k=1}^{n} b_{1k}^i (x_k - m_{ik}) + m_{ik} \text{ is } A_{i1} \text{ and} \ldots \\ s_{ij} &= \sum_{k=1}^{n} b_{jk}^i (x_k - m_{ik}) + m_{ik} \text{is } A_{ij}, \text{ and} \ldots \\ s_{\text{in}} &= \sum_{k=1}^{n} b_{nk}^i (x_k - m_{ik}) + m_{ik} \text{ is } A_{\text{in}} \\ &\text{THEN } y_i \text{is } m_{0i} + a_{1i}x_1 + \ldots + a_{ji}x_j + \ldots\end{aligned} \tag{26}$$

where b_{nk}^i the (n, k)th element of $\mathbf{B}_i, \mathbf{B}_i \in \Re^{n\times n}$ is the transformation matrix for rule i, and $\mathbf{s}_i \in \Re^n$ are the newly generated input variables and it is called the sources in ICA. The linguistic implication A_{ij} is now implicated by the new variable s_{ij}, which is a linear combination of the original variables. After transformation, the region that the membership functions cover is shown in Fig. 3(b). It is observed that the membership functions cover distribution of transformed data well, and thus a single fuzzy rule can associate this region with its proper output region (consequent).

Layer 3: A node in this layer represents one fuzzy rule and performs precondition matching of a rule. Here, we use the following AND operation for each Layer-2 node

$$\begin{aligned} f\left[u_i^{(3)}\right] &= \prod_i u_i^{(3)} \\ o_i^{(3)} &= a^{(3)}(f) = \frac{f}{\sum_{j=l}^{J(t)} f_j}.\end{aligned} \tag{27}$$

The link weight in Layer 3 $[w_i^{(3)}]$ is unity. The output of a Layer-3 node represents the firing strength of the corresponding fuzzy rule.

Layer 4: This layer is called the consequent layer. Two types of nodes are used in this layer and they are denoted as blank and shaded circles in Fig. 2, respectively. The node denoted by a blank circle (blank node) is the essential node representing a fuzzy set (described by a membership function) of the output variable. Different nodes in Layer 3 may be connected to the same blank node in Layer 4, meaning that the same consequent fuzzy set is specified for different rules. As to the shaded node, each node in Layer 3 has its own corresponding shaded node in Layer 4. One of the inputs to a shaded node is the output delivered from Layer 3 and the other inputs (terms) are the input variables from Layer 1. Combining these two types of nodes in Layer 4, we obtain the whole function performed by this layer as

$$f\left[u_i^{(4)}\right] = u_i^{(4)}$$
$$o_i^{(4)} = a^{(4)}(f) = \left(a_{0i} + \sum_j a_{ji}x_j\right) f \tag{28}$$

where $a_{0i} = m_{0i}$ is the center of output membership function and a_{ji} is the corresponding parameter.

Layer 5: Each node in this layer corresponds to one output variable. The node integrates all the actions recommended by Layer 4 and acts as a defuzzifier with

$$f\left[u_i^{(5)}\right] = \sum_i u_i^{(5)}$$
$$y_i = a^{(5)}(f) = f. \tag{29}$$

IV. Learning Rules of the On-Line ICA Mixture-Model-Based FNN

Two types of learning, structure and parameter learning, are used concurrently for constructing the proposed on-line ICA-mixture-model-based FNN. The structure learning includes both the precondition and consequent structure identification of a fuzzy if-then rule. Here precondition structure identification corresponds to the input-space partitioning and can be formulated as a combinational optimization problem with two objectives: to reduce the number of rules generated and to reduce the number of fuzzy sets on the universe of discourse of each input variable. As to the consequent structure identification, the main task is to decide when to generate a new membership function for the output variable and which significant terms (input variables) should be added to the consequent part (a linear equation) when necessary. In our system, we use the on-line ICA mixture model to realize the precondition and consequent structure identification of the proposed FNN.

For the parameter learning based on unsupervised and supervised learning algorithms, the parameters of the linear equations in the consequent parts are adjusted by the backpropagation rule to minimize a given cost function. The parameters in the precondition part are adjusted by the on-line ICA mixture model. The FNN can be used for normal operation at any time during the learning process without repeated training on the input-output patterns when on-line operation is required. There

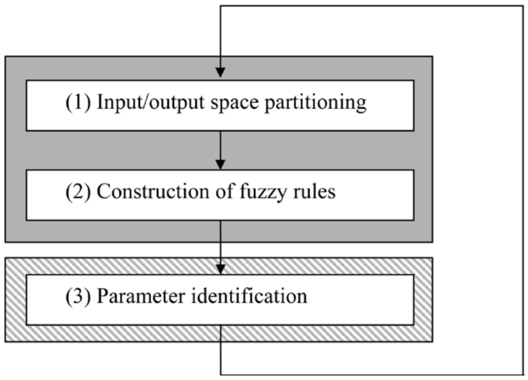

(1) Input/output space partitioning
(2) Construction of fuzzy rules
(3) Parameter identification

Fig. 4. Flowchart of the learning algorithm for the proposed FNN.

are no rules (i.e., no nodes in the network except the input-output nodes) in this network initially. They are created dynamically as learning proceeds upon receiving on-line incoming training data by performing the following learning processes simultaneously as shown in Fig. 4. In this figure, learning processes (1) and (2) belong to the structure learning phase and process (3) belongs to the parameter learning phase. In the rest of this section, the details of these learning processes are described in details.

A. Structure Learning by the On-Line ICA Mixture Model Algorithm

The way the input space is partitioned determines the number of rules extracted from training data as well as the number of fuzzy sets on the universal of discourse of each input variable. For each incoming pattern, the firing strength of a rule (i.e., the output of each layer2-node of the proposed FNN) can be interpreted as the degree that the incoming pattern belongs to the corresponding cluster. In other words, we can use the log-likelihood value calculated in (5) to represent the degree to which the newly incoming pattern $\mathbf{x}_t$ belongs to the jth cluster, i.e., $F^j(\mathbf{x}_t) = \log p(\mathbf{x}_t \mid C_j)$. Then, according to the on-line ICA mixture model [see (10)] derived in Section II, we can determine if a new cluster (i.e., new rule) should be generated (i.e., grow the network). This process is applied to both the input space and output space partitioning (clustering) simultaneously but individually. The detailed algorithms, called input space partitioning and output space partitioning, are given below.

Algorithm of Input Space Partitioning

```
IF $\mathbf{x}_t$ is the first incoming pattern THEN do
PART 1. {Generate a new rule with center $\mathbf{m}_1 = \mathbf{x}_t$, set the parameters
$\mathbf{Cov}_1 = [\mathbf{0}], p_1(\mathrm{C}_1) = 1, \mathbf{B}_1 = \mathbf{I}$, and $O_{1,p}$, where $p = 1, 2, \ldots, n$.
}
ELSE for each newly incoming pattern $\mathbf{x}_t$, do
PART 2. { Find $J_{\max} = \arg\max_{1\le j\le J(t)} F^j(\mathbf{x}_t)$,
IF $F^{J_{\max}}(\mathbf{x}_t) \ge F_{\mathrm{in}}$,
do the parameters updating steps of the on-line ICA mixture model derived in
Section II.
ELSE
{$J(t+1) = J(t) + 1$,
    generate a new fuzzy rule with $\mathbf{m}_{J(t+1)}(t+1) = \mathbf{x}_t$ and set the parameters of
    the new rule (cluster) as in PART 1.
    }
    }
```

Algorithm of Output Space Partitioning

```
    IF there is no output cluster
    do { PART 1 in Input Space Partitioning Algorithm, with $\mathbf{x}$ replaced by $\mathbf{d}$}
    ELSE
    do { Find $J_{\max} = \arg\max_{1 \le j \le J(t)} F^j(\mathbf{x}_t)$,
    IF $F^{J_{\max}}(\mathbf{x}_t) \ge F_{\text{out}}$,
    connect input cluster $J(t+1)$ to the existing output cluster $J_{\max}$,
    ELSE
    generate a new output cluster as the ELSE part of PART 2 of the Input Space
    Partitioning Algorithm, and connect input cluster $J(t+1)$ to this new output
    cluster.
    }
```

In the above algorithms, the threshold $F_{\text{in}}(F_{\text{out}})$ determines how many rules (clusters) will be generated in the input (output) space, where F_{in} and F_{out} should be negative since they are taken in natural log. For a larger value of F_{in}, more rules will be generated. The generation of a new input cluster corresponds to the generation of a new fuzzy rule, with its precondition part constructed by the input space partitioning algorithm in the above. At the same time, the above output space partitioning algorithm will decide the consequent part of the generated rule. The algorithm is based on the fact that different preconditions of different rules may be mapped to the same consequent fuzzy set. Since only the center of each output membership function is used for defuzzification, the consequent part of each rule may simply be regarded as a singleton. Compared to the general fuzzy rule-based models with singleton output where each rule has its own individual singleton value, fewer parameters are needed in the consequent part of the proposed FNN, especially for the case with a large number of rules.

B. Parameter Learning by the On-Line ICA Mixture Model and Backpropagation Algorithms

After the network structure is adjusted according to the current training pattern, the network then enters the parameter identification phase to adjust the parameters of the network optimally based on the same training pattern. Notice that the following parameter learning is performed on the whole network after structure learning, no matter whether the nodes (links) are newly added or are existent originally. The idea of backpropagation is used for this supervised learning. Considering the single-output case for clarity, our goal is to minimize the error function

$$E = \frac{1}{2}[y(t) - y^d(t)]^2 \tag{30}$$

where $y^d(t)$ is the desired output and $y(t)$ is the current output. For each training data set, starting at the input nodes, a forward pass is used to compute the activity levels of all the nodes in the network to obtain the current output $y(t)$. Then, starting at the output nodes, a backward pass is used to compute $(\partial E/\partial \mathbf{W})$ for all the hidden nodes. Assuming that $\mathbf{W}$ is the adjustable parameter in a node (e.g., a_{ji} in the FNN), then the general updating rule used is

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta\left(-\frac{\partial E}{\partial \mathbf{W}}\right) \tag{31}$$

where η is the learning rate and

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{W}} &= \frac{\partial E}{\partial(\text{activation function})} \times \frac{\partial(\text{activation function})}{\partial \mathbf{W}}\\ &= \frac{\partial E}{\partial a}\frac{\partial a}{\partial \mathbf{W}}.\end{aligned} \tag{32}$$

To show the updating rules, we shall show the computations of $(\partial E/\partial \mathbf{W})$ and update the parameter a_{ji}. First, we start the derivation from the output nodes. The error signal $[\delta_i^{(5)}]$, which needs to be computed and propagated, is derived by

$$\begin{aligned}\delta_i^{(5)} &= -\frac{\partial E}{\partial a^{(5)}}\\ &= y(t) - y^d(t).\end{aligned} \tag{33}$$

The updating rule for a_{ji} is

$$-\frac{\partial E}{\partial a_{ji}} = -\frac{\partial E}{\partial a^{(5)}}\frac{\partial a^{(5)}}{\partial a^{(4)}}\frac{\partial a^{(4)}}{\partial a_{ji}} \tag{34}$$

$$\begin{aligned}\frac{\partial a^{(5)}}{\partial a^{(4)}} &= 1\\ \frac{\partial a^{(4)}}{\partial a_{ji}} &= x_j u_i^{(4)}.\end{aligned} \tag{35}$$

Hence, the parameter is updated by

$$a_{ji}(t+1) = a_{ji}(t) + \eta(y^d(t) - y(t))x_j u_i^{(4)} \tag{36}$$

$$m_{0i}(t+1) = a_{0i}(t+1) = a_{0i}(t) + \eta(y^d(t) - y(t))u_i^{(4)}. \tag{37}$$

For the parameters $\mathbf{B}_j$ and $\mathbf{m}_j$ in Layer 2 of the proposed FNN, their updating rules can be determined by the proposed on-line ICA mixture model based on statistical independence under the constraint of minimizing the error function in (30). The on-line ICA mixture model with constraint is formulated as follows:

$$\begin{aligned}&\text{Maximize: } L(\mathbf{B}_j(t)) = \log(p(\mathbf{x}_t \mid C_j)),\\ &\text{Subject to: } E = 0.\end{aligned} \tag{38}$$

The problem of (38) is expressed as a constrained optimization problem which can be solved through the use of an augmented Lagrangian function. Hence, (18) can be rewritten as

$$\begin{aligned}\mathbf{B}_j(t+1) = \mathbf{B}_j(t) + \mu p(C_j \mid \mathbf{x}_t)\frac{\partial}{\partial \mathbf{B}_j(t)}\\ \times L(\mathbf{B}_j(t)) + \lambda p(C_j \mid \mathbf{x}_t)\frac{\partial E}{\partial \mathbf{B}_j(t)}\end{aligned} \tag{39}$$

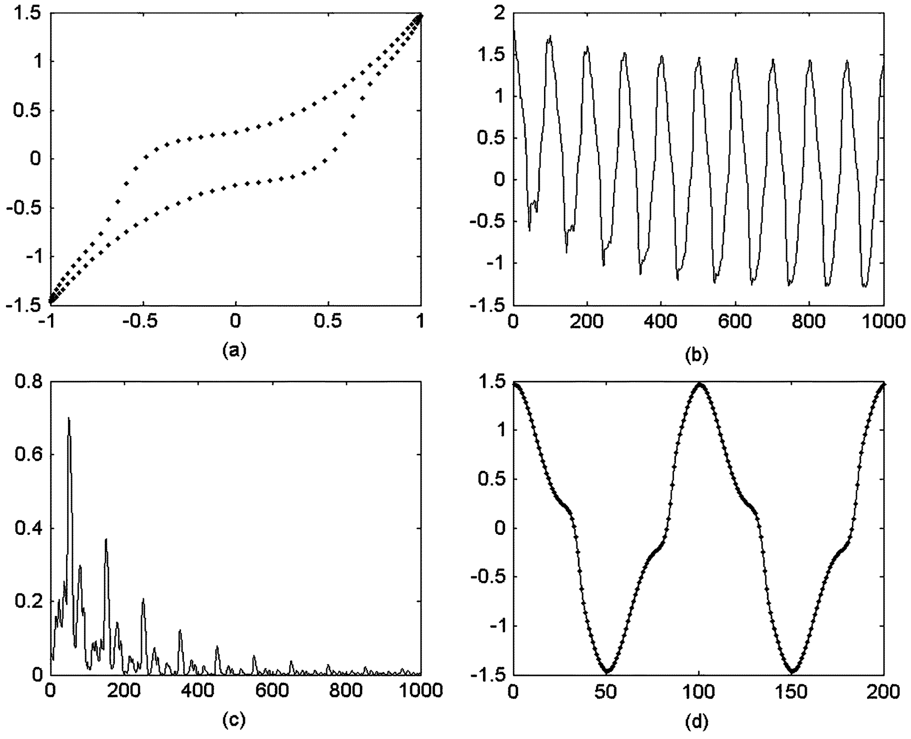


Fig. 5. Input training patterns of the target dynamic system. (b) The training procedure is performed for 1000 time steps. (c) Error function. (d) Simulation results of the FNN after 50 000 time steps. The dotted line denotes the output of the FNN and the solid line denotes the actual output.

where μ is the learning rate for maximizing the log likelihood, λ is the Lagrangian parameter, and

$$\frac{\partial E}{\partial \mathbf{B}_j} = \begin{bmatrix} \frac{\partial E}{\partial b_{j11}} & \frac{\partial E}{\partial b_{j12}} & \cdots & \frac{\partial E}{\partial b_{j1n}} \\ \vdots & \ddots & \ddots & \\ \vdots & \ddots & \frac{\partial E}{\partial b_{jkl}} & \ddots \\ \frac{\partial E}{\partial b_{jn1}} & \frac{\partial E}{\partial b_{jn2}} & \cdots & \frac{\partial E}{\partial b_{jnn}} \end{bmatrix}. \tag{40}$$

For each element of the above matrix, we compute

$$\begin{aligned} \frac{\partial E}{\partial b_{jkl}} &= \frac{\partial E}{\partial a^{(5)}} \frac{\partial a^{(5)}}{\partial b_{jkl}} \\ &= \frac{\partial E}{\partial a^{(5)}} \sum_i \frac{\partial a^{(5)}}{\partial a_i^{(4)}} \frac{\partial a_i^{(4)}}{\partial a_i^{(3)}} \frac{\partial a_i^{(3)}}{\partial a_j^{(2)}} \frac{\partial a_j^{(2)}}{\partial b_{jkl}} \end{aligned} \tag{41}$$

$$\begin{aligned} \frac{\partial a_i^{(4)}}{\partial a_i^{(3)}} &= m_{0i} + \sum_m a_{im} x_m \\ \frac{\partial a_i^{(3)}}{\partial a_j^{(2)}} &= \begin{cases} \frac{\sum_m a_m^{(2)} - a_i^{(2)}}{\sum_m a_m^{(2)}}, & \text{if } i = j \\ \frac{-a_j^{(2)}}{\sum_m a_m^{(2)}}, & \text{if } i \neq j \end{cases} \\ \frac{\partial a_j^{(2)}}{\partial b_{jkl}} &= \frac{\partial}{\partial b_{jkl}} \prod_{m=1}^{n} p_{jm}(s_{jm}) \\ &= \prod_{\substack{m=1,\\ m\neq k}}^{n} p_{jm}(s_{jm}) p'_{jk}(s_{jk}) x_{jl}. \end{aligned} \tag{42}$$

Substituting (42) into (41), we get the final updating rule for $\mathbf{B}_j$. Similar approach can derive the updating rule of $\mathbf{m}_j$.

V. Experiments

To verify the performance of the proposed FNN, several experiments are presented in this section. The experiments covering the areas of system identification, classification, and image segmentation are carried out and show that the proposed FNN can achieve significant improvements in the convergence speed and prediction accuracy.

A. Identification of Dynamic Systems

In this experiment, the proposed FNN is used to identify a dynamic system:

$$\begin{aligned} \hat{y}(k+1) = \hat{f}[&u(k), u(k-1), \ldots, u(k-p+1) \\ &y(k), y(k-1), \ldots, y(k-q+1)]. \end{aligned} \tag{43}$$

Since both the unknown plant and the FNN are driven by the same input, it adjusts itself with the goal of causing the output of the identification model to match that of the unknown plant. Upon convergence, their input-output relationship should match.

Example 1: The plant to be identified is guided by the difference equation

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k). \tag{44}$$

The output of the plant depends nonlinearly on both its past values and inputs, but the effects of the input and output values are additive. In applying the FNN to this identification problem, the used learning parameters are $\eta = 0.01, F_{\text{in}} = -5$, and $F_{\text{out}} = -5$, where F_{in} and F_{out} are the threshold parameters used in the input and output clustering processes, respectively. The training patterns are generated with $u(k) = \sin(2\pi k/100)$. Fig. 5(a) illustrates the distribution of the training patterns. The

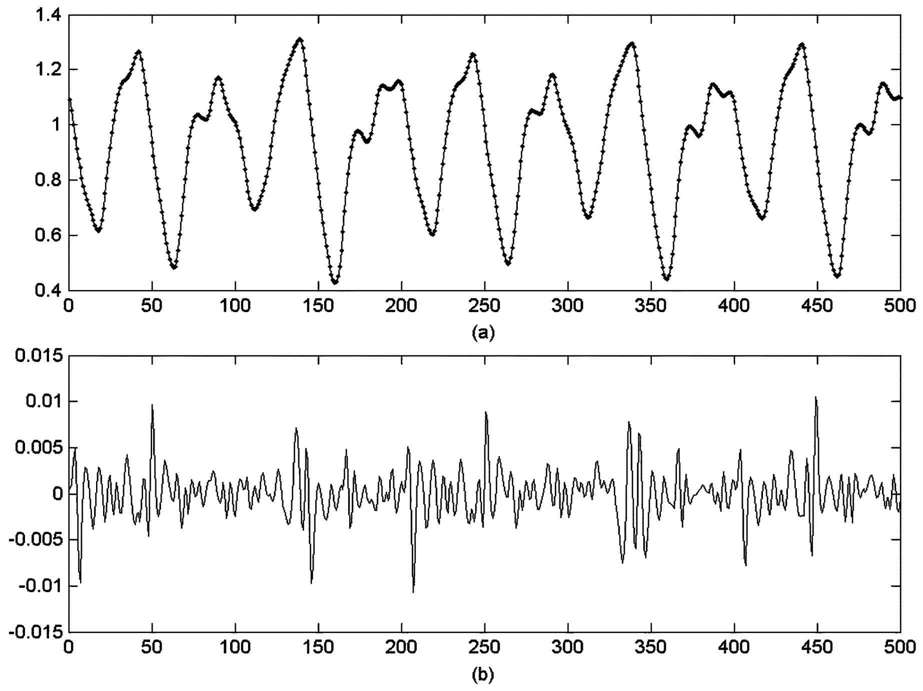


Fig. 6. (a) Prediction results of the FNN after training. The dotted line denotes the output of the FNN and the solid line denotes the actual output. (b) Prediction errors for testing.

TABLE I
INFLUENCE OF THE PARAMETERS F_{in} AND η ON THE PERFORMANCE OF THE PROPOSED FNN AND THE RESULTING (rms ERROR, NUMBER OF RULES)

η \ F_{in}	-5.5	-5	-4
0.01	(0.9263, 3)	(0.9486, 3)	(1.0499,4)
0.05	(1.0249, 2)	(0.8981, 3)	(1.0964,3)
0.1	(1.5824, 2)	(1.3120, 3)	(1.3167,3)
0.5	(4.8398, 4)	(4.7151, 4)	(5.2290,4)

training is performed for 1000 time steps [see Fig. 5(b)]. After training, three input and three output clusters are generated. Fig. 5(d) shows the outputs of the plant and the identification model after 50 000 time steps. In this figure, the outputs of the FNN are presented as the dotted curve while the plant outputs are presented as the solid curve. Since perfect identification result is achieved with our network, no additional terms need to be added to the consequent part.

In the above simulation, the parameters F_{in} and F_{out} need to be selected in advance. To give a clear understanding of the influence of these parameters on the structure and performance, different values of them are tested. For convenience, F_{in} and F_{out} are assigned to the same value. The generated network structure and corresponding root mean square (rms) errors and the number of rules are listed in Table I. From Table I, we can see that in certain ranges of the parameters, the rms error has no much change. According to our experiment, a higher value of F_{in} will increase the number of rules, but it is not necessarily reducing the rms error.

Example 2—Mackey–Glass Chaotic Time Series Prediction: We apply the proposed FNN to the Mackey–Glass time series prediction problem, which has been used in many studies in the FNN or NN communities. The Mackey–Glass time-delay differential equation is defined by

$$\frac{dx(t)}{dt} = \alpha x(t) + \frac{\beta x(t-\tau)}{1 + x^{10}(t-\tau)} \tag{45}$$

where $\alpha = -0.1$ and $\beta = 0.2$ in our experiment. When $x(0) = 1.2$ and $\tau = 17$, we have a nonperiodic and nonconvergent time series as shown in Fig. 6(a). Now we want to build an FNN that can predict $x(t+6)$ from the past values of this time series including, $x(t-18), x(t-12), x(t-6)$, and $x(t)$. Therefore, the input data format is $[x(t-18), x(t-12), x(t-6), x(t)]$ and the output is $x(t+6)$. From $t = 118$ to 1117, we collect 1000 data pairs. The first 500 data pairs are used for training while the others are used for testing. In applying the proposed FNN to this prediction problem, the used learning parameters are $\eta = 0.01, F_{\text{in}} = -15$, and $F_{\text{out}} = -15$. Fig. 6(a) shows the testing results of the FNN after training. The dotted line denotes the out of the FNN and the solid line denotes the actual output. The prediction errors by the proposed FNN are shown in Fig. 6(b). The average prediction error over 30 runs proposed FNN was 0.032, which was smaller than the prediction error (0.034) of the cooperative neural network ensembles presented in [29].

B. Experiments on Data Classification

In this section, four well-known benchmark data sets in classification—the iris data set, the Wisconsin breast cancer data set, the wine classification data set and Australian data set are used to evaluate the performance of the proposed FNN. These data sets are available from the University of California, Irvine, via an anonymous ftp site: ftp.ics.uci.edu/pub/machine-learning-databases [30].

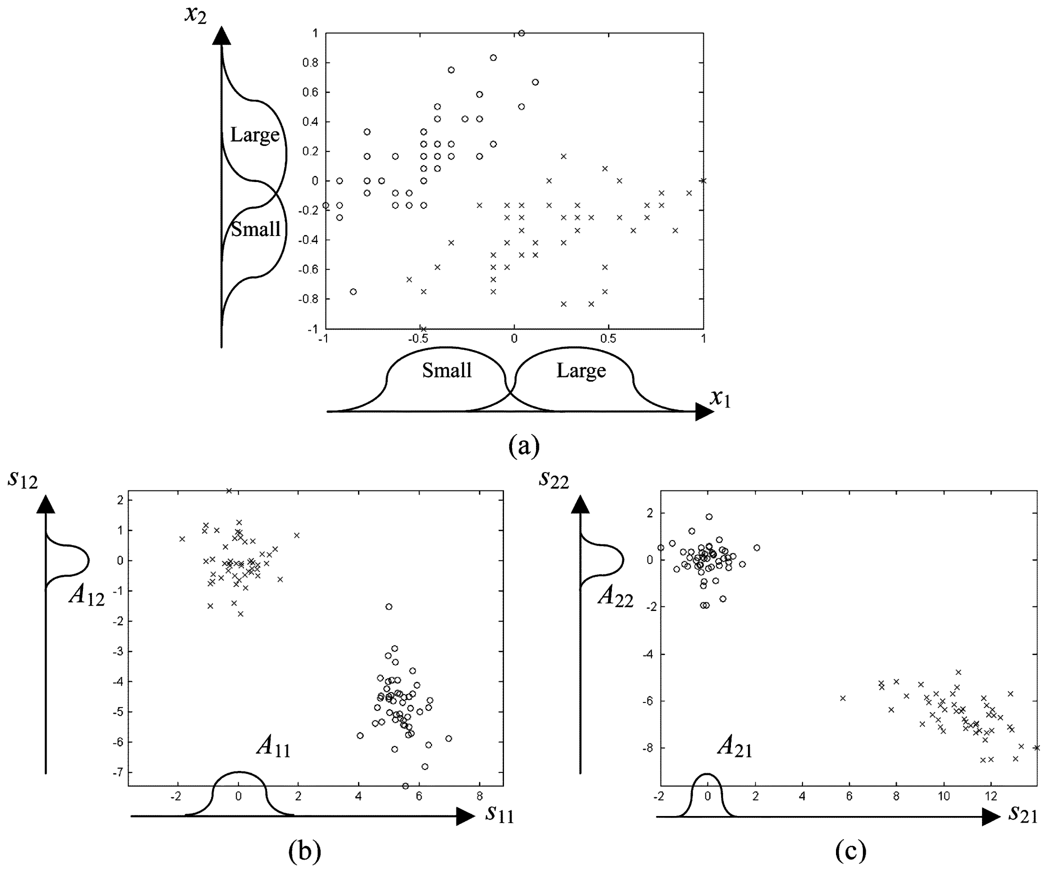


Fig. 7. The resultant membership functions of the proposed FNN with respect to Iris data. (a) The input vectors in $\mathbf{x}$ domain. (a) The independent vectors in $\mathbf{s}_1$ domain and the membership functions of rule 1. (c) The independent vectors in $\mathbf{s}_2$ domain and the membership functions of rule 2.

Example 1—Iris Data: The Fisher–Anderson iris data consist of four input measurements, sepal length (sl), sepal width (sw), petal length (pl), and petal width (pw), on 150 specimens of iris plant. Three species of iris are involved, Iris Sestosa, Iris Versiolor, and Iris Virginica, and each species contains 50 instances. To evaluate the effectiveness of the proposed FNN, 25 instances from each species were randomly selected as the training set and the remaining instances were used as the testing set. To perform classification, the output y of our system was used with the following classification rule:

$$\text{Iris} = \begin{cases} \text{Sestosa}, & \text{if } y < -0.5 \\ \text{Versiolor}, & \text{if } -0.5 \leq y < 0.5 \\ \text{Virginica}, & \text{if } 0.5 \leq y. \end{cases} \tag{46}$$

We set the threshold $F_{\text{in}} = F_{\text{out}} = -8$ and learning rate is $\eta = 0.01$ for the clustering algorithm. After learning, three clusters were revealed, so our structure consisted of three fuzzy rules and there are three fuzzy term sets for each input variable. Next, the parameter learning algorithm proceeds to fine tune the network to achieve a better performance. Fig. 7 shows the resultant memberships for illustration. We only plot two of the three classes Iris data and two of the four dimensions for simplicity. Fig. 7(a) shows the input vectors that defined in the $\mathbf{x}$ space. Figs. 7(b) and (c) present the independent space obtained by the on-line ICA-mixture-model. A_{11} and A_{12} denote the membership functions of Rule 1. A_{21} and A_{22} denote the membership functions of Rule 2. According to Fig. 7(b), we can find that the membership functions of Rule 1 can match Class "x" well after the transformation through ICA; in the meantime, Class "o" is pushed away from the membership functions of Rule 1, which makes the fire strength of Class "o" to Rule 1 be almost zero. Similarly, according to Fig. 7(c), the membership functions of Rule 2 can match Class "o" well after the transformation through ICA; in the meantime, Class "x" is pushed away from the membership functions of Rule 2. Consequently, we can find that the membership functions among these rules are not overlapped with each other.

We can also use this simplified example (two classes and two dimensions) as shown in Fig. 7 to discuss the semantics of the obtained fuzzy rules in the proposed model. These two resultant rules in the independent space can be represented as (47), shown at the bottom of the next page. After the transformation of $\mathbf{B}^{-1}$ (the inverse of transformation matrix), according to Fig. 7(a), the fuzzy rules shown in (47) can be modified and presented in the input space $\mathbf{x}$ with linguistic implications as (48), shown at the bottom of the next page. It is observed that the major difference between (47) and (48) is the space of the input vectors in precondition of fuzzy rules, but their consequent of the fuzzy rules are the same. The firing strength of fuzzy rules in (48) can be represented by the firing strength of fuzzy rules in (47), because the transformation is one-to-one mapping.

Table II shows the comparison of the proposed FNN with different iterations and Table III shows the contrast of the proposed FNN with different threshold values F_{in} for iris classification and the results are the average of ten different training

TABLE II
PERFORMANCE OF THE PROPOSED FNN WITH DIFFERENT ITERATIONS ON THE IRIS DATA CLASSIFICATION PROBLEM

Iteration number	Threshold F_{in}	Number of rules	Training error	Testing error	Average testing recognition rate
10	-8	3	0~2 (avg. 0.8)	0~3 (avg. 1.3)	98.27%
100	-8	3	0~2 (avg. 0.7)	0~3 (avg. 1.2)	98.4%

TABLE III
PERFORMANCE COMPARISONS OF THE PROPOSED FNN WITH DIFFERENT THRESHOLD VALUES $F_{\rm in}$ ON THE IRIS DATA CLASSIFICATION PROBLEM

Threshold F_{in}	Number of rules	Training error	Testing error	Average testing recognition rate
-6	5	0~3(avg. 1.5)	0~5(avg. 2.4)	96.79%
-8	3	0~2(avg. 0.7)	0~3(avg. 1.2)	98.4%
-10	1	0~3(avg. 1.1)	0~4(avg. 1.8)	97.07%

TABLE IV
PERFORMANCE COMPARISONS OF VARIOUS CLASSIFIERS ON THE IRIS DATA CLASSIFICATION PROBLEM

Classifiers	Number of rules	Testing error	Average testing recognition rate
FMMC [31]	—	2	97.2%
FUNLVQ+GFENCE [32]	—	0~6(avg. 2.75)	96.3%
FEBFC [33]	2	0~5(avg. 2.16~2.84)	96.7%~97.12%
Wu-and-Chen's [34]	3	—(avg. 2.84)	96.21%
SANFIS(I Ⅱ Ⅲ) [35]	3	0~4(avg. 1.9~2.1)	97.2%~97.47%
The propose FNN	3	0~2(avg.1.2~1.3)	98.27%~98.4%

TABLE V
PERFORMANCE OF THE PROPOSED FNN WITH DIFFERENT ITERATIONS ON THE WISCONSIN BREAST CANCER DATA CLASSIFICATION PROBLEM

Threshold F_{in}	Iteration number	Number of rules	Average testing recognition rate
-40	10	2	96.44%
-40	100	4	96.18%

TABLE VI
PERFORMANCE COMPARISONS OF THE PROPOSED FNN WITH DIFFERENT THRESHOLD VALUES $F_{\rm in}$ ON THE WISCONSIN BREAST CANCER DATA CLASSIFICATION PROBLEM

Threshold F_{in}	Iteration number	Number of rules	Average testing recognition rate
-60	100	1	95.92%
-50	100	2	96.76%
-40	100	4	96.18%

TABLE VII
PERFORMANCE COMPARISONS OF VARIOUS CLASSIFIERS ON THE WISCONSIN BREAST CANCER DATA CLASSIFICATION PROBLEM

Classifiers	Number of rules	Average testing recognition rate
FMSC [36]	—	94.9%
NEFCLASS [37]	—	92.7%
NNFS [38]	3	93.94%(~94.15%)
FEBFC [33]	6	94.67%(~95.14%)
SANFIS(I Ⅱ Ⅲ) [35]	2	96.07%~96.3%
The proposed FNN	2	96.44%~96.76%

and testing sets. In this example, the iteration is defined as 100. A higher value of $F_{\rm in}$ results in a larger rule number. It means the number of rules is increased or decreased depending on the parameter $F_{\rm in}$. Because the iris data consist of three-cluster patterns, the average testing classification rate of the FNN with three fuzzy rules is the highest. Table IV shows the comparison of the classification results of our FNN and other fuzzy classifiers on iris data.

Example 2—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. In this data set, 458 patterns are in the Benign class and the other 241 patterns are in the Malignant class. Since there are 16 patterns containing missed values, we used 683 patterns to evaluate the performance of the proposed FNN. To compare the performance with other classifiers, half of the 683 patterns were used as training set and the remaining patterns were used as the testing set. The data set was normalized to the range [0, 1]. We classified the output y of the structure using the following classification rule:

$$\text{Breast Cancer} = \begin{cases} \text{Bengin}, & \text{if } y < 0.5 \\ \text{Malignamt}, & \text{if } 1.5 \le y. \end{cases} \tag{49}$$

We set the threshold $F_{\rm in} = F_{\rm out} = -40$ and learning rate $\eta = 0.01$ for training. Two clusters were revealed in the final learning process. The learned structure consisted of 2 fuzzy rules and 2 fuzzy terms when the iteration is set for 10 for each input feature. In the same situation, when we set the iteration to be 100, the learned structure will be change into 4 fuzzy rules and 4 fuzzy terms for each input feature. We repeated the experiment on 10 different training sets (see Table V). In Table V, we can find that when the iteration number increases, the number of rules will increase. The situation may occur in updating the independent axes, $\mathbf{B}_j$. With the different transformation of input coordinates, the number of rules may be changed. Table VI shows the comparison of the proposed FNN with different thresholds $F_{\rm in}$ for the breast cancer data classification. Table VII shows the comparison between the learned structure models and other fuzzy, neural-network, and neuro-fuzzy classifiers on the same target problem. It shows that the recognition rate of our proposed FNN outperforms the listed classifiers.

Example 3—Wine Classification 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

$$\begin{cases} \text{Rule1(class“x")}: \text{IF } s_{11} \text{ is } A_{11} \text{ and } s_{12} \text{ is } A_{12} \text{ THEN } y \text{ is } m_0 + a_{11}x_1 + a_{21}x_2 \\ \text{Rule2(class“o")}: \text{IF } s_{21} \text{ is } A_{21} \text{ and } s_{22} \text{ is } A_{22} \text{ THEN } y \text{ is } m_0 + a_{12}x_1 + a_{22}x_2 \end{cases}. \tag{47}$$

$$\begin{cases} \text{Rule1(class“x")}: \text{IF } x_1 \text{ is Large and } x_2 \text{ is Small THEN } y \text{ is } m_0 + a_{11}x_1 + a_{21}x_2 \\ \text{Rule2 (class“o")}: \text{IF } x_1 \text{ is Small and } x_2 \text{ is Large THEN } y \text{ is } m_0 + a_{12}x_1 + a_{22}x_2 \end{cases}. \tag{48}$$

TABLE VIII
PERFORMANCE COMPARISONS OF VARIOUS CLASSIFIERS ON THE WINE DATA CLASSIFICATION PROBLEM

Classifiers	Number of rules	Average testing recognition rate
Cornran et al. [41]	60	100%(best)~98.31%(worst)
Ishibuchi et al. [40]	60	99.44%(best)~97.95%(worst)
Setnes et al. [39]	3	98.31%
SANFIS(I II III) [35]	3	99.44%
The proposed FNN	3	100%

intensity, hue, OD280/OD315 of diluted wines and proline. Corcoran *et al.* [41] applied a real-coded genetic-based method to learn 60 nonfuzzy if-then rules from 178 patterns and used a population of 1500 individuals for 300 generations with full replacement. Ishibuchi *et al.* [40] proposed an integer-coded GA and grid-partitioning to design a fuzzy classifier with 60 fuzzy rules from the 178 patterns. They used a population of 100 individuals and applied for 1000 generations with full replacement. Setnes *et al.* [39] applied a real-coded GA and c-means clustering algorithm on all the available 178 patterns to design a TSK model as a classifier. Nine features were selected during their proposed simplification and optimization process. In [35], the MCA clustering algorithm was proposed to solve this problem. With our scheme, three clusters were revealed in the final learning process. After applying the parameter learning process for five epochs, the classification error was reduced to zero. The comparison between our classifier and the above-mentioned fuzzy classifiers are shown in Table VIII.

Example 4—Australian Credit Approval Data: This dataset contains 690 patterns distributed into two output classes. Each pattern consists of 14 (6 Continuous and 8 Categorical) input features. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. This dataset is interesting because there is a good mix of attributes: continuous, nominal with small numbers of values, and nominal with larger numbers of values. There were originally a few missing values, but these have all been replaced by the overall median. We classified the output y of the structure using the following classification rule:

$$\text{Australian} = \begin{cases} \text{class } 0, & \text{if } y < 0.5 \\ \text{class } 1, & \text{if } 0.5 \le y < 1.5. \end{cases} \tag{50}$$

We set the threshold $F_{\text{in}} = F_{\text{out}} = -40$ and learning rate $\eta = 0.001$ for training. After structure learning, our structure consisted of two fuzzy rules. The results with the tenfold cross validation is showed in Table IX. According to the experimental results, the proposed FNN with only two fuzzy rules can reach higher accuracy than other methods.

Since the tenfold cross-validation testing model would produce more reliable results, the testing results of the proposed FNN on the Iris, Wine, Wisconsin and Australian datasets with the testing model of tenfold cross validation is presented in Table X. These experimental results show that, given a reasonable number of seed clusters, the proposed FNN is capable of automatically identifying the true cluster configuration. Hence, the proposed recursive on-line ICA mixture model can further reduce the number of required rules and achieve better system performance.

TABLE IX
PERFORMANCE COMPARISONS OF VARIOUS CLASSIFIERS ON THE AUSTRALIAN DATA CLASSIFICATION PROBLEM WITH THE TESTING MODEL OF 10-FOLD CROSS-VALIDATION

Classifiers	Average testing recognition rate
Bayes	84.9%
Radial	85.5%
BackProp	84.6%
LVQ	80.3%
The proposed FNN	86.7%

TABLE X
TESTING RESULTS OF THE PROPOSED FNN ON THE IRIS, WINE, WISCONSIN AND AUSTRALIAN DATASETS WITH THE TESTING MODEL OF TENFOLD CROSS VALIDATION

Datasets	Number of dimension	Number of classes	Average training recognition rate	Average testing recognition rate
Iris	4	3	97.9%	98%
Wine	13	3	99.6%	98.9%
Wisconsin	10	2	96.68%	96.92%
Australian	14	2	86.22%	86.65%

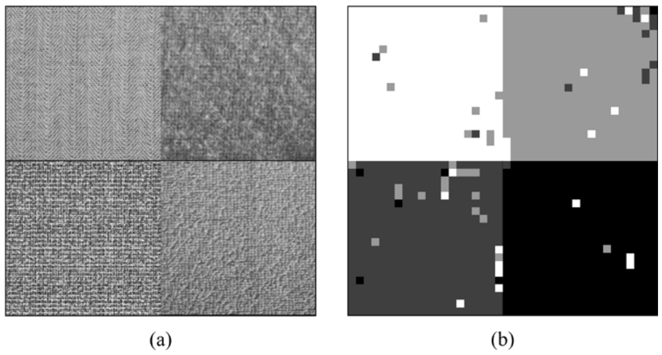


Fig. 8. Texture segmentation. (a) Texture of four different materials: (top-left) herringbone weave, (top-right) woolen cloth, (bottom-left) denim, (bottom-right) raffia. (b) The labels found by the proposed FNN are shown in different grey levels. The misclassified patches of size 10×10 pixels are shown from the square region of the texture.

C. Unsupervised Image Classification and Segmentation

In this section, we applied our FNN to learn multiple classes in a single image. The learned classes are mutually exclusive and the whole image is divided into small image patches for classification. Three experiments were performed to illustrate how the algorithm can identify textures in an image. In the first experiment, four texture images were taken and merged into one image. Fig. 8(a) shows the textures of four different materials: (top-left) herringbone weave, (top-right) woolen cloth, (bottom-left) denim, (bottom-right) raffia. Each of the texture image size is 200×200 pixels. Four classes of training patterns were adopted by randomly sampling 10×10 pixel patches from each texture image; i.e., no label information was taken into account. We classify the output y using the following classification rule:

$$\text{Class} = \begin{cases} \text{herringbone weave}, & \text{if } y < 1.5 \\ \text{woollen cloth}, & \text{if } 1.5 \le y < 2.5 \\ \text{denim}, & \text{if } 2.5 \le y < 3.5 \\ \text{raffia}, & \text{if } 3.5 \le y. \end{cases} \tag{51}$$

The automatic classification results of the image as shown in Fig. 8(b) was done by dividing the image into adjacent nonover-

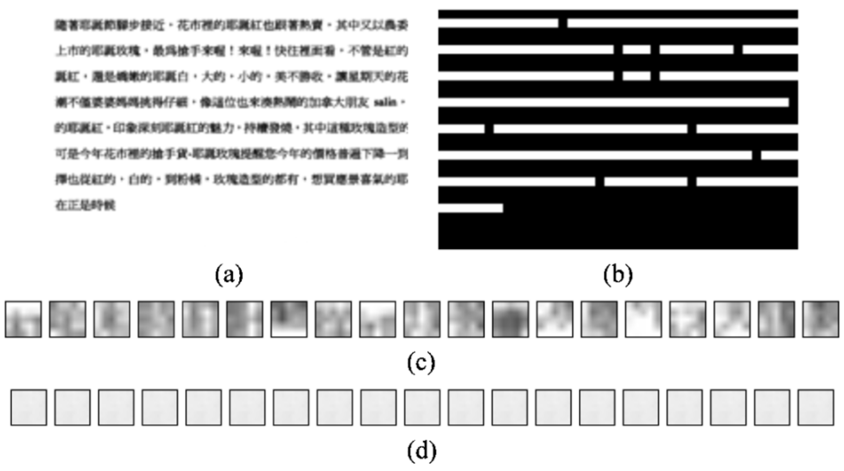


Fig. 9. Example of text extraction: The 5×5 pixel image patches were randomly sampled from the image and used as training patterns to the proposed FNN. (a) Original image. (b) Text image patches. (c) Picture image patches. (d) Background image patches.

lapping 10×10 pixel patches. The misclassified patches are shown with different grey levels from the square region of the texture.

In the second experiment, we used a text image of scanned newspaper articles. The training data set consisted of 5×5 pixel patches $(\mathrm{n} = 25)$ selected randomly from the images of two difference types. Each type is random selected with 50 patches. We classify the output y using the following classification rule:

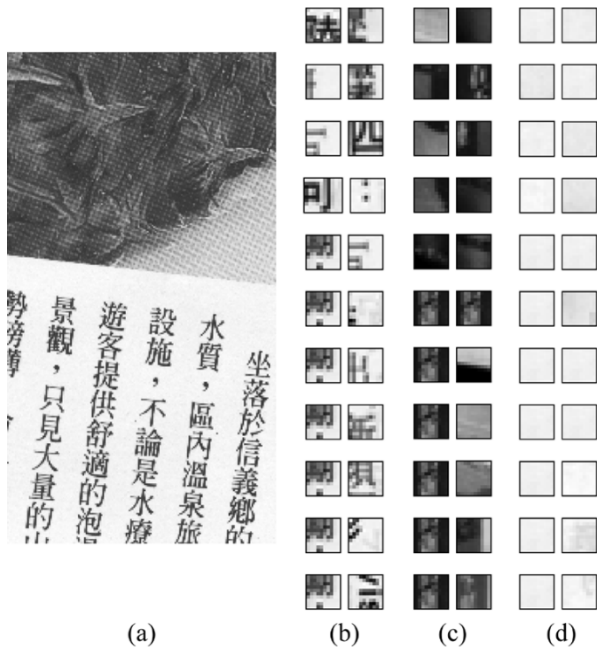
$$\text{Class} = \begin{cases} \text{Text}, & \text{if } y < 1.5 \\ \text{Background}, & \text{if } 1.5 \leq y. \end{cases} \tag{52}$$

Fig. 9(a) shows the original text image and Fig. 9(b) shows the classification result of the FNN. The 5×5 pixel patches are shown in Fig. 9(c) and (d), respectively. In Fig. 9(b), the text region is denoted by white color while the background region by black color. Obviously, the text region and the background region are successfully separated into two classes using the proposed on-line ICA-mixture-model-based FNN.

The final experiment shows the segmentation of a text/picture mixture image of scanned newspaper articles. This image contains text and a picture, and the goal is to separate text, picture, and background regions in the image apart. The training data set consists of 10×10 pixel patches selected randomly from the text, picture, and the background regions. In the training set, each class includes 50 image patches. The output y is classified by the following classification rule:

$$\text{Class} = \begin{cases} \text{Text}, & \text{if } y < 1.5 \\ \text{Picture}, & \text{if } 1.5 \leq y < 2.5 \\ \text{Background}, & \text{if } 2.5 \leq y. \end{cases} \tag{53}$$

Fig. 10(a) shows the scanned image. Fig. 10(b)–(d) illustrate examples of the image patches including text, picture, and background regions. Before learning, the prior values of the image patches were normalized to the range [0, 1]. Fig. 11 shows the classification result of the proposed FNN for this scanned image using image patches of 10×10 pixel size. When the iteration number is set to 100, the error is reduced to 0.0023. When the iteration number is increased to 500, the error rate almost reaches zero. Finally, the segmentation result of the whole scanned image is shown in Fig. 11(b).



坐落於信義鄉的
水質，區內溫泉旅
設施，不論是水療
遊客提供舒適的泡
景觀，只見大量的
勢磅礴

Fig. 10. Example of the scanned page. The 10×10 pixel images were randomly sampled from the images as training patterns to the proposed FNN. (a) Original image. (b) Text image patches. (c) Picture image patches. (d) Background image patches.

VI. Conclusion

In this paper, a novel FNN was proposed based on a newly derived on-line ICA mixture model. It is a general connectionist model of a fuzzy logic system, which can find its optimal structure and parameters automatically. Both the structure and parameter identifications are done simultaneously during on-line learning, so it can be used for normal operation at any time as learning proceeds without any assignment of fuzzy rules in advance. For structure learning, the proposed on-line ICA mixture model algorithm was able to identify the optimal number of clusters (i.e., rules) and simultaneously estimate the centers and variances of the clusters for constructing the FNN structure in a single pass without a priori knowledge of the distribution

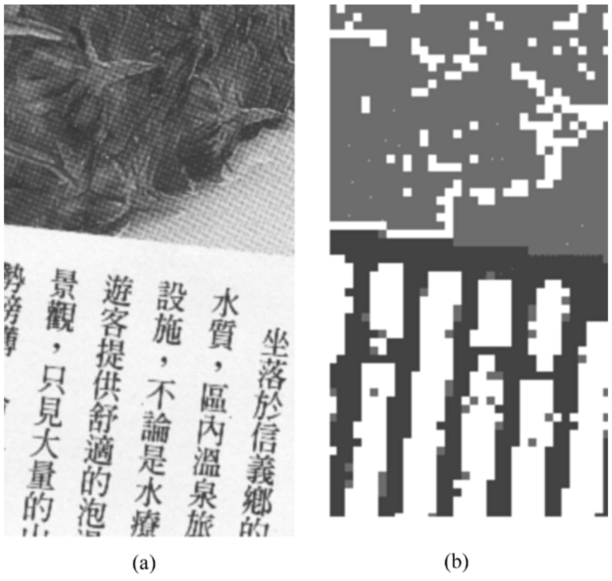


Fig. 11. Segmentation of an image scanned from a magazine: (a) Original image. (b) The segmentation result of the whole scanned image.

of the training data set. A novel network construction method for solving the dilemma between the number of rules and the number of consequent terms is developed. The number of generated rules and membership functions is small even for modeling a sophisticated system. As a summary, the proposed FNN can always find itself an economic network size, and the learning speed as well as the modeling ability is all appreciated. Several experiments covering the areas of system identification, classification, and image segmentation were carried out to demonstrate the performance of the proposed FNN. These experiments showed that the proposed FNN can achieve significant improvements in the convergence speed and prediction accuracy.

References

[1] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[2] C. T. Lin, *Neural Fuzzy Control Systems with Structure and Parameter Learning*. New York: World Scientific, 1994.

[3] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hell, 1996.

[4] R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[5] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro-Fuzzy Systems*. New York: Wiley, 1997.

[6] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm," *IEEE Trans. Neural Netw.*, vol. 3, pp. 801–806, Sep. 1992.

[7] K. Tanaka, M. Sano, and H. Watanabe, "Modeling and control of carbon monoxide concentration using a neuro-fuzzy technique," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 271–279, Aug. 1995.

[8] Y. Lin and G. A. Cunningham, "A new approach to fuzzy-neural system modeling," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 190–197, May 1995.

[9] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 7–31, Feb. 1993.

[10] L. Wang and R. Langari, "Building sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 454–458, Nov. 1995.

[11] E. H. Ruspini, "Recent development in fuzzy clustering," *Fuzzy Set and Possibility Theory*, pp. 113–147, 1982.

[12] T. W. Lee, M. S. Lewicki, and T. J. Sejnowski, "Unsupervised classification with nongaussian mixture models using ICA," *Adv. Neural Inf. Process. Syst.*, vol. 11, pp. 508–514, 1999.

[13] C. F. Juang and C. T. Lin, "An on-line self constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–32, Feb. 1998.

[14] T. W. Lee, M. Girolami, and T. J. Sejnowski, "Independent component analysis using an extended infomax algorithm for mixed sub-gaussian and super-gaussian sources," *Neural Comput.*, vol. 11, no. 2, pp. 417–441, 1999.

[15] J. C. Bezdek, J. Keller, R. Krisnapuram, and N. R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Boston, MA: Kluwer, 1999.

[16] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[17] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy Cluster Analysis*. New York: Wiley, 1999.

[18] G. J. McLachlan and T. Krishnan, *The EM Algorithms and Extensions*. New York: Wiley, 1997.

[19] T.-W. Lee and M. S. Lewicki, "Image processing methods using ICA mixture models," in *Independent Component Analysis: Principles and Practice*, S. Roberts and R. Everson, Eds. New York: Cambridge Univ. Press, 2001.

[20] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[21] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

[22] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural Computation*, vol. 12, no. 2, pp. 337–365, 2000.

[23] D. MacKay, Maximum Likelihood and Covariant Algorithms for Independent Component Analysis, Draft 3.7, 1996.

[24] U.-M. Bae and T.-W. Lee, "Blind signal separation in teleconferencing using the ICA mixture model," *Electron. Lett.*, vol. 36, no. 7, pp. 680–382, 2000.

[25] J. F. Cardoso, "High-order contrasts for independent component analysis," *Neural Comput.*, vol. 11, pp. 157–192, 1999.

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

[27] T. W. Lee, M. S. Lewicki, and T. J. Sejnowski, "ICA mixture models for unsupervised classification of non-Gaussian classes and automatic context switching in blind signal separation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, Oct. 2000.

[28] T. W. Lee, M. S. Lewicki, and T. J. Sejnowski, "ICA mixture models for unsupervised and automatic context switching," in *Proc. Int. Workshop ICA*, 1999, pp. 209–214.

[29] Md. M. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, Jul. 2003.

[30] C. L. Blake and C. J. Merz. (1998) *UCI Repository of Machine Learning Databases* [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[31] P. K. Simpson, "Fuzzy min-max neural networks—Part I: Classification," *IEEE Trans. Neural Netw.*, vol. 3, pp. 776–786, Sep. 1992.

[32] H. M. Lee, "A neural network classifier with disjunctive fuzzy information," *Neural Netw.*, vol. 11, no. 6, pp. 1113–1125, 1998.

[33] 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. Syst., Man, Cybern. B*, vol. 31, pp. 426–432, Jun. 2001.

[34] 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, Cybern. B, Cybern.*, vol. 29, pp. 25–40, Feb. 1999.

[35] J. S. Wang and C. S. George Lee, "Self-adaptive neuro-fuzzy inference systems for classification applications," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 6, Dec. 2002.

[36] B. C. Lovel and A. P. Bradley, "The multiscale classifier," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 2, pp. 124–137, Feb. 1996.

[37] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data," *Fuzzy Sets Syst.*, vol. 89, no. 3, pp. 277–288, 1997.

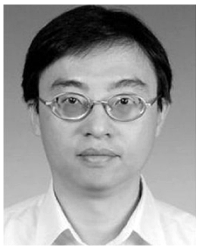
[38] R. Setiono and H. Liu, "Neural-network feature selector," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 654–662, Jun. 1997.

[39] M. Setnes and H. Roubos, "GA-fuzzy modeling and classification: Complexity and performance," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 509–522, Oct. 2000.

[40] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, pp. 601–618, Oct. 1999.

[41] A. L. Corcoran and S. Sen, "Using real-valued genetic algorithms to evolve rule sets for classification," in *Proc. 1st IEEE Conf. Evolutionary Computation*, Orlando, FL, Jun. 1994, pp. 120–124.

[42] P. Brazdil and J. Gama, LIACC, Univ. of Porto Rua Campo Alegre 823 4150 Porto, Portugal.



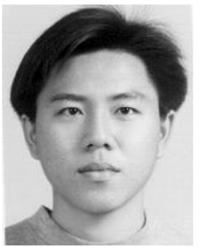
Chin-Teng Lin (S'88–M'91–SM'99–F'04) received the B.S. degree in control engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, R.O.C., in 1986, and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1992, respectively.

Since August 1992, he has been with the College of Electrical Engineering and Computer Science, National Chiao-Tung University, where he is currently the Associate Dean of the college and a professor of Electrical and Control Engineering Department. He has also served as the Director of Brain Research Center, NCTU Branch, University System of Taiwan since September 2003. He served as the Director of the Research and Development Office of the National Chiao-Tung University from 1998 to 2000, and the Chairman of the Electrical and Control Engineering Department from 2000 to 2003. His current research interests are neural networks, fuzzy systems, cellular neural networks (CNN), fuzzy neural networks (FNN), neural engineering, algorithms and VLSI design for pattern recognition, intelligent control, and multimedia (including image/video and speech/audio) signal processing, and intelligent transportation system (ITS). He is the book co-author of *Neural Fuzzy Systems—A Neuro-Fuzzy Synergism to Intelligent Systems* (Prentice Hall), and the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (New York: World Scientific, 1994). He has also published over 80 journal papers in the areas of neural networks, fuzzy systems, multimedia hardware/software, and soft computing, including 60 IEEE journal papers.

Dr. Lin is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi honorary societies. He is also a member of the IEEE Circuit and Systems Society (CASS), the IEEE Neural Network Society, the IEEE Computer Society, the IEEE Robotics and Automation Society, and the IEEE System, Man, and Cybernetics Society. Dr. Lin is the Distinguished Lecturer representing the NSATC of IEEE CASS from 2003 to 2005. He has been very active in the IEEE International Symposium on Circuits and Systems (ISCAS) by serving as the Organizing Committee member, as the International Liaison of ISCAS 2005 in Japan, and the Organizing Committee member as the Special Session Co-Chair of ISCAS 2006 in Greece. He has been the Executive Council member (Supervisor) of the Chinese Automation Association since 1998. He was the Executive Council member of the Chinese Fuzzy System Association Taiwan (CFSAT), from 1994 to 2001. Dr. Lin is the Society President of CFSAT since 2002. He has won the Outstanding Research Award granted by the National Science Council (NSC), Taiwan, since 1997 to present, the Outstanding Electrical Engineering Professor Award granted by the Chinese Institute of Electrical Engineering (CIEE) in 1997, the Outstanding Engineering Professor Award granted by the Chinese Institute of Engineering (CIE) in 2000, and the 2002 Taiwan Outstanding Information-Technology Expert Award. He was also elected to be one of the 38th Ten Outstanding Rising Stars in Taiwan, R.O.C., (2000). He currently serves as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: I—REGULAR PAPERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: II—EXPRESS BRIEFS, *International Journal of Speech Technology*, and the *Journal of Automatica*.

Wen-Chang Cheng received the B.S. degree in electronics engineering from National Cheng-Kung University, Tainan, Taiwan, R.O.C., the M.S. degree in electronics engineering from National Chung-Cheng University, Chiayi, Taiwan, R.O.C., in 1997 and 1999, respectively. He is currently working toward the Ph.D. degree in electrical and control engineering at National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

He is also a Lecturer in information management at Hsiuping Institute of Technology, Taichung, Taiwan, R.O.C. His current research interests include neuro-fuzzy systems, neural networks, image processing, machine learning, and artificial intelligence.



Sheng-Fu Liang was born in Tainan, Taiwan, R.O.C., in 1971. He received the B.S. and M.S. degrees in control engineering from the National Chiao-Tung University (NCTU), Taiwan, R.O.C., in 1994 and 1996, respectively. He received the Ph.D. degree in electrical and control engineering from NCTU in 2000.

Currently, he is a Research Assistant Professor in Electrical and Control Engineering, NCTU. Dr. Liang has also served as the Chief Executive of Brain Research Center, NCTU Branch, University System of Taiwan since September 2003. His current research interests are neural networks, fuzzy neural networks (FNN), brain-computer interface (BCI), and multimedia signal processing.