Support-Vector-Based Fuzzy Neural Network for Pattern Classification

Chin-Teng Lin, *Fellow, IEEE*, Chang-Mao Yeh, Sheng-Fu Liang, Jen-Feng Chung, and Nimit Kumar

***Abstract*—Fuzzy neural networks (FNNs) for pattern classification usually use the backpropagation or C-cluster type learning algorithms to learn the parameters of the fuzzy rules and membership functions from the training data. However, such kinds of learning algorithms usually cannot minimize the empirical risk (training error) and expected risk (testing error) simultaneously, and thus cannot reach a good classification performance in the testing phase. To tackle this drawback, a support-vector-based fuzzy neural network (SVFNN) is proposed for pattern classification in this paper. The SVFNN combines the superior classification power of support vector machine (SVM) in high dimensional data spaces and the efficient human-like reasoning of FNN in handling uncertainty information. A learning algorithm consisting of three learning phases is developed to construct the SVFNN and train its parameters. In the first phase, the fuzzy rules and membership functions are automatically determined by the clustering principle. In the second phase, the parameters of FNN are calculated by the SVM with the proposed adaptive fuzzy kernel function. In the third phase, the relevant fuzzy rules are selected by the proposed reducing fuzzy rule method. To investigate the effectiveness of the proposed SVFNN classification, it is applied to the Iris, Vehicle, Dna, Satimage, Ijcnn1 datasets from the UCI Repository, Statlog collection and IJCNN challenge 2001, respectively. Experimental results show that the proposed SVFNN for pattern classification can achieve good classification performance with drastically reduced number of fuzzy kernel functions.**

***Index Terms*—Fuzzy kernel function, fuzzy neural network (FNN), kernel method, mercer theorem, pattern classification, support vector machine (SVM).**

I. Introduction

As IS WIDELY known, both fuzzy logic and neural networks are aimed at exploiting human-like knowledge processing capability. The fuzzy logic system using linguistic information can model the qualitative aspects of human knowledge and reasoning processes without employing precise quantitative analyzes [1]. The neural network is a popular generation of information processing systems that demonstrate the ability to learn from training data [2]. Much research has been done on fuzzy neural networks (FNNs), which combine the capability of fuzzy reasoning in handling uncertain information and the capability of neural networks in learning from processes [3]–[5]. They have been successfully applied to classification, identification, control, pattern recognition, and image processing, etc. In particular, many learning algorithms of fuzzy (neural) classifiers have been presented and applied in pattern classification and decision-making systems [6], [7]. Moreover, several researchers have investigated the fuzzy-rule-based methods for pattern classification [8]–[11].

A fuzzy system consists of a bunch of fuzzy IF–THEN rules. Conventionally, the selection of fuzzy IF–THEN rules often relies on a substantial amount of heuristic observation to express proper strategy's knowledge. Obviously, it is difficult for human experts to examine all the input–output data to find a number of proper rules for the fuzzy system. Most preresearches used the backpropagation (BP) and/or C-cluster type learning algorithms to train parameters of fuzzy rules and membership functions from the training data [12], [13]. However, such learning only aims at minimizing the classification error in the training phase, and it cannot guarantee the lowest error rate in the testing phase. In statistical learning theory, the support vector machine (SVM) [14] has been developed for solving this bottleneck. The SVM performs structural risk minimization and creates a classifier with minimized VC dimension. As the VC dimension is low, the expected probability of error is low to ensure a good generalization. The SVM keeps the training error fixed while minimizing the confidence interval. So, the SVM has good generalization ability and can simultaneously minimize the empirical risk and the expected risk for pattern classification problems. More importantly, an SVM can work very well in a high dimensional feature space. However, the optimal solutions of SVM rely heavily on the property of selected kernel functions, whose parameters are always fixed and are chosen solely based on heuristics or trial-and-error nowadays. The FNN on the other hand is an approach that is based on adaptive local representations with iterative learning ability. With this motivation, a theoretical foundation for the FNN using the SVM method is developed in this paper. We exploit the knowledge representation power and learning ability of the FNN to determine the kernel functions of the SVM adaptively, and propose a novel adaptive fuzzy kernel function, which has been proven to be a Mercer kernel. There have been some researches on combining SVM with FNN [15]–[18]. In [15], a self-organizing map with fuzzy class memberships was used to reduce the training samples to speed up the SVM training. The objective of [16]–[18] was on

Manuscript received May 1, 2003; revised April 12, 2004 and April 6, 2005. This work was supported in part by the Ministry of Education, Taiwan, under Grant EX-91-E-FA06-4-4, and by the Ministry of Economic Affairs, Taiwan, under Grant 94-EC-17-A-02-S1-032.

C.-T. Lin is with the Department of Electrical and Control Engineering/Department of Computer Science, National Chiao-Tung University, Hsinchu 300, Taiwan, and also with the Brain Research Center, NCTU Branch, University System of Taiwan, Hsinchu 300, Taiwan (e-mail: ctlin@mail.nctu.edu.tw).

C.-M. Yeh and J.-F. Chung are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan. (e-mail: yeh120@dragon.ccut.edu.tw; jfchung.ece88g@nctu.edu.tw).

S.-F. Liang is with the Department of Biological Science and Technology, National Chiao-Tung University, Hsinchu 300, Taiwan, and also with the Brain Research Center, NCTU Branch, University System of Taiwan, Hsinchu 300, Taiwan (e-mail: sfliang@mail.nctu.edu.tw).

N. Kumar is with the IBM India Research Lab, Delhi 110016, India.

Digital Object Identifier 10.1109/TFUZZ.2005.861604

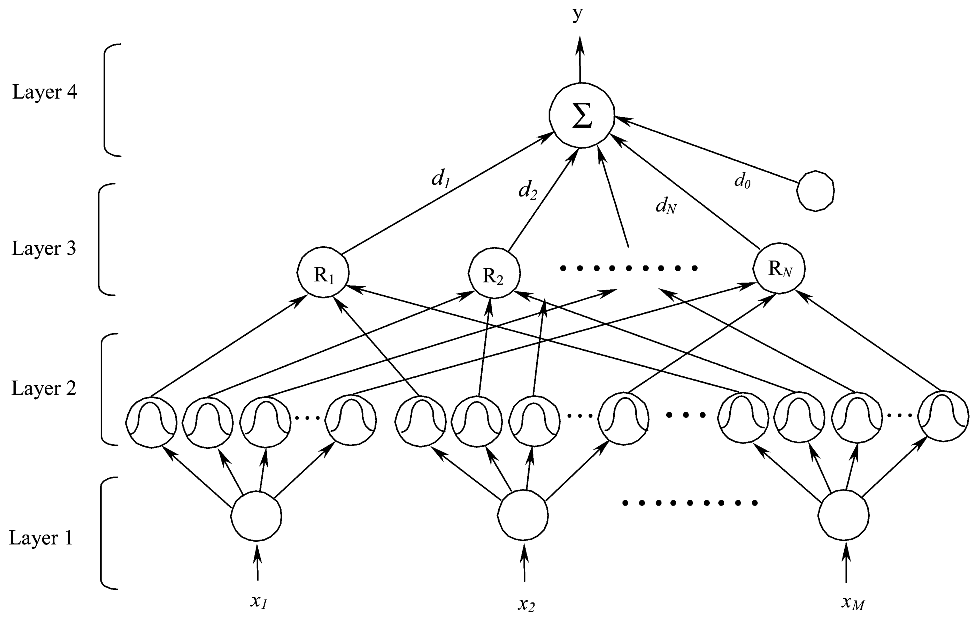

Fig. 1. Structure of the four-layered fuzzy neural network.

improving the accuracy of SVM on multiclass pattern recognition problems. The SVFNN developed in this paper can not only well maintain the classification accuracy, but also reduce the number of support vectors as compared with the regular SVM. The regular SVM suffers from the difficulty of long computational time in using nonlinear kernels on large datasets which come from many real applications. Therefore, one major contribution of this paper is to propose a systematical procedure to reduce the support vectors to deal with this problem.

In this paper, we develop a support-vector-based fuzzy neural network (SVFNN) for pattern classification, which is the realization of a new idea for the adaptive kernel functions used in the SVM. The use of the proposed fuzzy kernels provides the SVM with adaptive local representation power, and thus brings the advantages of FNN (such as adaptive learning and economic network structure) into the SVM directly. On the other hand, the SVM provides the advantage of global optimization to the FNN and also its ability to minimize the expected risk; while the FNN originally works on the principle of minimizing only the training error. The proposed learning algorithm of SVFNN consists of three phases. In the first phase, the initial fuzzy rule (cluster) and membership of network structure are automatically established based on the fuzzy clustering method. The input space partitioning determines the initial fuzzy rules, which is used to determine the fuzzy kernels. In the second phase, the means of membership functions and the connecting weights between layers 3 and 4 of SVFNN (see Fig. 1) are optimized by using the result of the SVM learning with the fuzzy kernels. In the third phase, unnecessary fuzzy rules are recognized and eliminated and the relevant fuzzy rules are determined. Experimental results on five datasets (Iris, Vehicle, Dna, Satimage, Ijcnn1) from the UCI Repository, Statlog collection and IJCNN challenge 2001 show that the proposed SVFNN classification method can automatically generate the fuzzy rules, improve the accuracy of classification, reduce the number of required kernel functions, and increase the speed of classification.

This paper is organized as follows. In Section II, the structure and initial construction of the FNN is presented. Section III describes the proposed adaptive fuzzy kernel with sound theoretic derivations. In Section IV, the learning algorithm of the SVFNN is developed. In Section V, the SVFNN is applied to solve several classification problems and performance comparisons with other classification methods are made. Finally, the conclusions are summarized in the Section VI.

II. Structure and Construction of Initial Fuzzy Neural Network

A. Structure of FNNs

A four-layered FNN is shown in Fig. 1, which is comprised of the input, membership function, rule, and output layers. Layer 1 accepts input variables, whose nodes represent input linguistic variables. Layer 2 is to calculate the membership values, whose nodes represent the terms of the respective linguistic variables. Nodes at Layer 3 represent fuzzy rules. The links before Layer 3 represent the preconditions of fuzzy rules, and the link after Layer 3 represent the consequences of fuzzy rules. Layer 4 is the output layer. This four-layered network realizes the following form of fuzzy rules:

$$\begin{aligned}&\text{Rule } \mathbf{R}_j: \text{ If } x_1 \text{ is } A_{1j} \text{ and } \ldots x_i \text{ is } A_{ij} \ldots \text{ and } x_M \text{ is } A_{Mj}\\&\text{Then } y \text{ is } d_j, \qquad j = 1, 2, \ldots, N\end{aligned} \tag{1}$$

where A_{ij} are the fuzzy sets of the input variables x_i, $i = 1, 2, \ldots, M$ and d_j are the consequent parameter of y. For the ease of analysis, a fuzzy rule 0 is added as

$$\begin{aligned}&\text{Rule } 0: \text{ If } x_1 \text{ is } A_{10} \text{ and} \ldots \text{ and } x_M \text{ is } A_{M0}\\&\text{Then } y \text{ is } d_0\end{aligned} \tag{2}$$

where A_{k0} is a universal fuzzy set, whose fuzzy degree is 1 for any input value x_i, $i = 1, 2, \ldots, M$ and d_0 is the consequent parameter of y in the fuzzy rule 0. Define $O^{(P)}$ and $a^{(P)}$ as the output and input variables of a node in layer P, respectively. The signal propagation and the basic functions in each layer are described as follows.

Layer 1- Input layer: 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

$$O^{(1)} = a_i^{(1)} = x_i \tag{3}$$

where $x_i, i = 1, 2, \ldots, M$ are the input variables of the FNN.

Layer 2—Membership function layer: Each node in this layer is a membership function that corresponds one linguistic label (e.g., fast, slow, 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 to a fuzzy set is calculated in Layer 2

$$O^{(2)} = u_i^{(j)}\left(a_i^{(2)}\right) \tag{4}$$

where $u_i^{(j)}(\cdot)$ is a membership function $u_i^{(j)}(\cdot) : R \to [0,1]$, $i = 1, 2, \ldots, M$, $j = 1, 2, \ldots, N$. With the use of Gaussian membership function, the operation performed in this layer is

$$O^{(2)} = e^{-\left(\left(a_i^{(2)} - m_{ij}\right)^2 / \sigma_{ij}^2\right)} \tag{5}$$

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

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

$$O^{(3)} = \prod_{i=1}^{M} a_i^{(3)} = e^{-[\mathbf{D}_j(\mathbf{x}-\mathbf{m}_j)]^T[\mathbf{D}_j(\mathbf{x}-\mathbf{m}_j)]} \tag{6}$$

where $\mathbf{D}_j = \text{diag}(1/\sigma_{1j}, \ldots, 1/\sigma_{Mj})$, $\mathbf{m}_j = [m_{1j}, m_{2j}, \ldots, m_{Mj}]^T$, $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_M]^T$ is the FNN input vector. The output of a Layer-3 node represents the firing strength of the corresponding fuzzy rule.

Layer 4—Output layer: The single node $O^{(4)}$ in this layer is labeled with Σ, which computes the overall output as the summation of all input signals

$$O^{(4)} = \sum_{j=1}^{N} d_j \times a_j^{(4)} + d_0 \tag{7}$$

where the connecting weight d_j is the output action strength of the Layer 4 output associated with the Layer 3 rule and the scalar d_0 is a bias. Thus the fuzzy neural network mapping can be rewritten in the following input–output form:

$$O^{(4)} = \sum_{j=1}^{N} d_j \times a_j^{(4)} + d_0 = \sum_{j=1}^{N} d_j \prod_{i=1}^{M} u_i^{(j)}(x_i) + d_0. \tag{8}$$

B. Construction of Fuzzy Rules

For constructing the initial fuzzy rules of the FNN, the fuzzy clustering method is used to partition a set of data into a number of overlapping clusters based on the distance in a metric space between the data points and the cluster prototypes. Each cluster in the product space of the input–output data represents a rule in the rule base. The goal is to establish the fuzzy preconditions in the rules. The membership functions in Layer 2 of FNN can be obtained by projections onto the various input variables x_i spanning the cluster space. In this work, we use an aligned clustering-based approach proposed in [19]. This method produces a partition result as shown in Fig. 2.

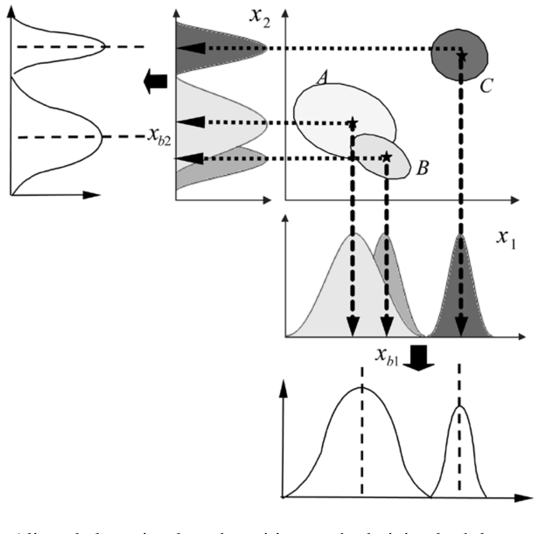

Fig. 2. Aligned clustering-based partition method giving both less number of clusters as well as less number of membership functions.

The input space partitioning is also the first step in constructing the fuzzy kernel function in the SVFNN. The purpose of partitioning has a two-fold objective.

- It should give us a minimum yet sufficient number of clusters or fuzzy rules.
- It must be in spirit with the SVM-based classification scheme.

To satisfy the aforementioned conditions, we use a clustering method which takes care of both the input and output values of a data set. That is, the clustering is done based on the fact that the points lying in a cluster also belong to the same class or have an identical value of the output variable. The class information of input data is only used in the training stage to generate the clustering-based fuzzy rules; however, in testing stage, the input data excite the fuzzy rules directly without using class information. In addition, we also allow existence of overlapping clusters, with no bound on the extent of overlap, if two clusters contain points belonging to the same class. We may have a clustering like the one shown in Fig. 3. Thus a point may be geometrically closer to the center of a cluster, but it can belong only to the nearest cluster, which has the points belonging to the same class as that point.

A rule corresponds to a cluster in the input space, with $\mathbf{m}_j$ and $\mathbf{D}_j$ representing the center and variance of that cluster. For each incoming pattern $\mathbf{x}$, the strength a rule is fired can be interpreted as the degree the incoming pattern belongs to the corresponding cluster. It is generally represented by calculating degree of membership of the incoming pattern in the cluster [20].

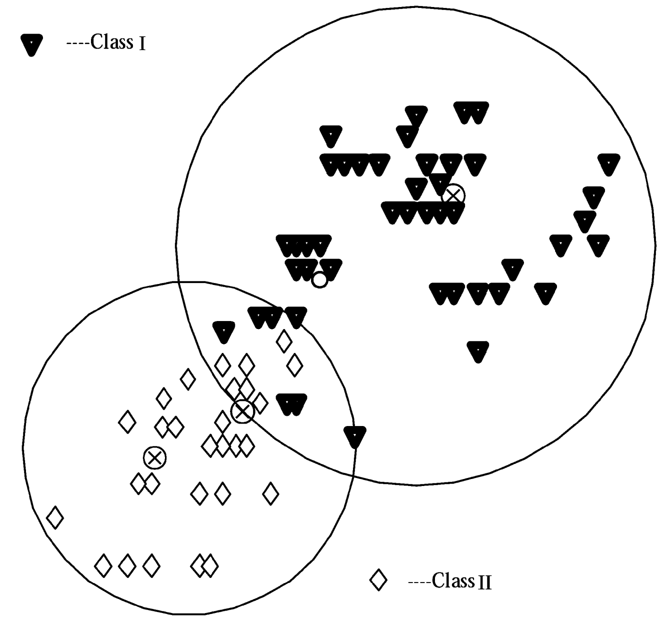

Fig. 3. Clustering arrangement allowing overlap and selecting the member points according to the labels (or classes) attached to them.

For computational efficiency, we can use the firing strength derived in (6) directly as this degree measure

$$F^j(\mathbf{x}) = \prod_{i=1}^{M} a_i^{(3)} = e^{-[\mathbf{D}_j(\mathbf{x}-\mathbf{m}_j)]^T[\mathbf{D}_j(\mathbf{x}-\mathbf{m}_j)]} \in [0,1] \quad (9)$$

where $F^j(\mathbf{x}) \in [0,1]$. In the above equation the term $[\mathbf{D}_j(\mathbf{x} - \mathbf{m}_j)]^T[\mathbf{D}_j(\mathbf{x} - \mathbf{m}_j)]$ is the distance between $\mathbf{x}$ and the center of cluster j. Using this measure, we can obtain the following criterion for the generation of a new fuzzy rule. Let $\mathbf{x}$ be the newly incoming pattern. Find

$$J = \arg \max_{1\leq j\leq c(t)} F^j(\mathbf{x}) \quad (10)$$

where $c(t)$ is the number of existing rules at time t. If $F^J \leq \overline{F}(t)$, then a new rule is generated, where $\overline{F}(t) \in (0,1)$ is a prespecified threshold that decays during the learning process. Once a new rule is generated, the next step is to assign initial centers and widths of the corresponding membership functions. Since our goal is to minimize an objective function and the centers and widths are all adjustable later in the following learning phases, it is of little sense to spend much time on the assignment of centers and widths for finding a perfect cluster. Hence, we can simply set

$$\mathbf{m}_{[c(t)+1]} = \mathbf{x} \quad (11)$$

$$\mathbf{D}_{[c(t)+1]} = \frac{-1}{\chi} \cdot \operatorname{diag}\left[\frac{1}{\operatorname{in}(F_j)} \cdots \frac{1}{\ln(F^J)}\right] \quad (12)$$

according to the first-nearest-neighbor heuristic [21], where $\chi \geq 0$ decides the overlap degree between two clusters. Similar methods are used in [22], [23] for the allocation of a new radial basis unit. However, in [22] the degree measure does not take the width $\mathbf{D}_j$ into consideration. In [23], the width of each unit is kept at a prespecified constant value, so the allocation result is, in fact, the same as that in [22]. In this paper, the width is taken into account in the degree measure, so for a cluster with larger width (meaning a larger region is covered), fewer rules will be generated in its vicinity than a cluster with smaller width. This is a more reasonable result. Another disadvantage of [22] is that another degree measure (the Euclidean distance) is required, which increases the computation load.

After a rule is generated, the next step is to decompose the multidimensional membership function formed in (11) and (12) to the corresponding 1-D membership function for each input variable. To reduce the number of fuzzy sets of each input variable and to avoid the existence of highly similar ones, we should check the similarities between the newly projected membership function and the existing ones in each input dimension. Before going to the details on how this overall process works, let us consider the similarity measure first. Since Gaussian membership functions are used in the SVFNN, we use the formula of the similarity measure of two fuzzy sets with Gaussian membership functions derived previously in [24]. Suppose the fuzzy sets to be measured are fuzzy sets A and B with membership function $\mu_A(x) = \exp\{-(x-c_1)^2/\sigma_1^2\}$ and $\mu_B(x) = \exp\{-(x-c_2)^2/\sigma_2^2\}$, respectively. The union of two fuzzy sets A and B is a fuzzy set $A \cup B$ such that $u_{A\cup B}(x) = \max[u_A(x), u_B(x)]$, for every $x \in U$. The intersection of two fuzzy sets A and B is a fuzzy set $A \cap B$ such that $u_{A\cap B}(x) = \min[u_A(x), u_B(x)]$, for every $x \in U$. The size or cardinality of fuzzy set A, $M(A)$, equals the sum of the support values of A: $M(A) = \sum_{x\in U} u_A(x)$. Since the area of the bell-shaped function, $\exp\{-x(x-m)^2/\sigma^2\}$, is $\sigma\sqrt{\pi}$ [25] and its height is always 1, it can be approximated by an isosceles triangle with unity height and the length of bottom edge $2\sigma\sqrt{\pi}$. We can then compute the fuzzy similarity measure of two fuzzy sets with such kind of membership functions. Assume $c_1 \geq c_2$ as in [24], we can compute $M|A \cap B|$ by

$$\begin{aligned} M\,|A\cap B| &= \sum_{x\in U}(\min[u_A(x), u_B(x)]) \\ &= \frac{1}{2}\frac{h^2\left[c_2 - c_1 + \sqrt{\pi}(\sigma_1+\sigma_2)\right]}{\sqrt{\pi}(\sigma_1+\sigma_2)} \\ &\quad + \frac{1}{2}\frac{h^2\left[c_2 - c_1 + \sqrt{\pi}(\sigma_1-\sigma_2)\right]}{\sqrt{\pi}(\sigma_2-\sigma_1)} \\ &\quad + \frac{1}{2}\frac{h^2\left[c_2 - c_1 - \sqrt{\pi}(\sigma_1+\sigma_2)\right]}{\sqrt{\pi}(\sigma_1-\sigma_2)} \end{aligned} \quad (13)$$

where $h(\cdot) = \max\{0, \cdot\}$. So the approximate similarity measure is

$$E(A,B) = \frac{M\,|A\cap B|}{M\,|A\cup B|} = \frac{M\,|A\cap B|}{\sigma_1\sqrt{\pi} + \sigma_2\sqrt{\pi} - M\,|A\cap B|} \quad (14)$$

where we use the fact that $M(A) + M(B) = M(A \cap B) + M(A \cap B)$ [24]. By using this similarity measure, we can check if two projected membership functions are close enough to be merged into one single membership function $\mu_c(x) = \exp\{-(x-c_3)^2/\sigma_3^2\}$. The mean and variance of the merged membership function can be calculated by

$$c_3 = \frac{c_1+c_2}{2} \quad (15)$$

$$\sigma_3 = \frac{\sigma_1+\sigma_2}{2}. \quad (16)$$

Fig. 2 illustrates this procedure, and the detailed algorithm is given in Section IV.

III. Adaptive Fuzzy Kernel

The proposed fuzzy kernel $K(\hat{\mathbf{x}}, \hat{\mathbf{z}})$ in this paper is defined as shown in (17) at the bottom of the page, where $\hat{\mathbf{x}} = [x_1, x_2, x_3, \ldots, x_M] \in R^M$ and $\hat{\mathbf{z}} = [z_1, z_2, z_3, \ldots, z_M] \in R^M$ are any two training samples, and $u_j(x_i)$ is the membership function of the jth cluster. Let the training set be $\mathbf{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_v, y_v)\}$ with explanatory variables $\mathbf{x}_i$ and the corresponding class labels y_i, for all $i = 1, 2, \ldots, v$ where v is the total number of training samples. Assume the training samples are partitioned into l clusters through fuzzy clustering in Section II. We can perform the following permutation of training samples:

$$\begin{aligned} \mathit{cluster}\ 1 &= \left\{\left(\mathbf{x}_1^1, y_1^1\right), \ldots, \left(\mathbf{x}_{k_1}^1, y_{k_1}^1\right)\right\} \\ \mathit{cluster}\ 2 &= \left\{\left(\mathbf{x}_1^2, y_1^2\right), \ldots, \left(\mathbf{x}_{k_2}^2, y_{k_2}^2\right)\right\} \\ &\ \vdots \\ \mathit{cluster}\ l &= \left\{\left(\mathbf{x}_1^l, y_1^l\right), \ldots, \left(\mathbf{x}_{k_l}^l, y_{k_l}^l\right)\right\} \end{aligned} \tag{18}$$

where k_g, $g = 1, 2, \ldots, l$ is the number of points belonging to the gth cluster, so that we have $\sum_{g=1}^{l} k_g = v$. Then the fuzzy kernel can be calculated by using the training set in (18), and the obtained kernel matrix $\mathbf{K}$ can be rewritten as the following form:

$$K = \begin{bmatrix} \mathbf{K}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{K}_l \end{bmatrix} \in R^{v \times v} \tag{19}$$

where $\mathbf{K}_g$, $g = 1, 2, \ldots, l$ is defined as shown in (20) at the bottom of the page. In order that the fuzzy kernel function defined by (17) is suitable for application in SVM, we must prove that the fuzzy kernel function is symmetric and positive–definite Gram Matrices [26]. To prove this, we first quote the following theorems.

Theorem 1 (Mercer Theorem [26]): Let X be a compact subset of R^n. Suppose K is a continuous symmetric function such that the integral operator $T_K : L_2(X) \to L_2(X)$

$$(T_K f)(\cdot) = \int_X K(\cdot, \mathbf{x}) f(\mathbf{x}) \quad d\mathbf{x} \geq 0 \tag{21}$$

is positive; that is

$$\int_{X \times X} K(\mathbf{x},\ \mathbf{z}) f(\mathbf{x}) f(\mathbf{z})\, d\mathbf{x} d\mathbf{z} \geq 0 \qquad \forall f \in L_2(X) \tag{22}$$

for all $f \in L_2(X)$. Then we can expand $K(\mathbf{x}, \mathbf{z})$ in a uniformly convergent series (on $X \times X$) in terms of T_K's eigen-functions $\phi_j \in L_2(X)$, normalized in such a way that $\|\phi_j\|_{L_2} = 1$, and positive associated eigenvalues $\lambda_j > 0$

$$K(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{z}). \tag{23}$$

The kernel is referred to as Mercer's kernel as it satisfies the above Mercer theorem.

Proposition 1 [27]: A function $K(\mathbf{x}, \mathbf{z})$ is a valid kernel iff for any finite set it produces symmetric and positive–definite Gram matrices.

Proposition 2 [28]: Let K_1 and K_2 be kernels over $X \times X$, $X \subseteq R^n$. Then the $K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z}) K_2(\mathbf{x}, \mathbf{z})$ function is also a kernel.

Definition 1 [29]: A function $f : R \to R$ is said to be a positive–definite function if the matrix $[f(x_i - x_j)] \in R^{n \times n}$ is positive semidefinite for all choices of points $\{x_1, \ldots, x_n\} \subset R$ and all $n = 1, 2, \ldots$.

Proposition 3 [29]: A block diagonal matrix with the positive–definite diagonal matrices is also a positive–definite matrix.

Theorem 2: For the fuzzy kernel defined by (17), if the membership functions $u(x_i) : R \to [0, 1]$, $i = 1, 2, \ldots, n$, are positive–definite functions, then the fuzzy kernel is a Mercer kernel.

Proof: First, we prove that the formed kernel matrix $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^{n}$ is a symmetric matrix. According to the definition of fuzzy kernel in (17), if x_i and z_i are in the jth cluster

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) &= \prod_{k=1}^{n} u_j(x_k) \cdot u_j(z_k) \\ &= \prod_{k=1}^{n} u_j(z_k) \cdot u_j(x_k) = K(\mathbf{z},\ \mathbf{x}) \end{aligned}$$

$$K(\hat{\mathbf{x}}, \hat{\mathbf{z}}) = \begin{cases} \prod_{i=1}^{M} u_j(x_i) \cdot u_j(z_i), & \text{if } \hat{\mathbf{x}} \text{ and } \hat{\mathbf{z}} \text{ are both in the } j\text{th cluster} \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

$$\mathbf{K}_g = \begin{bmatrix} K(x_1^g, x_1^g) & K(x_1^g, x_2^g) & \cdots & K\left(x_1^g, x_{k_g}^g\right) \\ K(x_2^g, x_1^g) & K(x_2^g, x_2^g) & \ddots & \vdots \\ \vdots & \ddots & \ddots & K\left(x_{k_g-1}^g, x_{k_g}^g\right) \\ K\left(x_{k_g}^g, x_1^g\right) & \cdots & K\left(x_{k_g}^g, x_{k_g-1}^g\right) & K\left(x_{k_g}^g, x_{k_g}^g\right) \end{bmatrix} \in R^{k_g \times k_g} \tag{20}$$

otherwise

$$K(\mathbf{x},\mathbf{z}) = K(\mathbf{z},\mathbf{x}) = 0.$$

So the kernel matrix is indeed symmetric. By the elementary properties of *Proposition 2*, the product of two positively defined functions is also a kernel function. And according to *Proposition 3*, a block diagonal matrix with the positive–definite diagonal matrices is also a positive–definite matrix. So the fuzzy kernel defined by (17) is a Mercer kernel. □

Since the proposed fuzzy kernel has been proven to be a Mercer kernel, we can apply the SVM technique to obtain the optimal parameters of SVFNN. It is noted that the proposed SVFNN is not a pure SVM, so it dose not minimize the empirical risk and expected risk exactly as SVMs do. However, it can achieve good classification performance with drastically reduced number of fuzzy kernel functions. The way to apply the SVM technique to obtain the optimal parameters of SVFNN is presented in Section IV in details.

IV. Learning Algorithm of SVFNN

The learning algorithm of the SVFNN consists of three phases. The details are given below:

Learning Phase 1—Establishing initial fuzzy rules

The first phase establishes the initial fuzzy rules, which were usually derived from human experts as linguistic knowledge. Because it is not always easy to derive fuzzy rules from human experts, the method of automatically generating fuzzy rules from numerical data is issued. The input space partitioning determines the number of fuzzy rules extracted from the training set and also the number of fuzzy sets. We use the centers and widths of the clusters to represent the rules. To determine the cluster to which a point belongs, we consider the value of the firing strength for the given cluster. The highest value of the firing strength determines the cluster to which the point belongs. The whole algorithm for the generation of new fuzzy rules as well as fuzzy sets in each input variable is as follows. Suppose no rules are existent initially.

```
IF $\mathbf{x}$ is the first incoming input pattern, THEN do
PART 1. { Generate a new rule with center $\mathbf{m}_1 = \mathbf{x}$ and width $\mathbf{D}_1 = \mathrm{diag}(1/\sigma_{\text{init}},\ldots,1/\sigma_{\text{init}})$,
 IF the output pattern $\mathbf{y}$ belongs to class 1 (namely, $\mathbf{y} = [1\ \ 0]$), $\boldsymbol{w}_{\text{Con}-1} = [1\ \ 0]$ for indicating output node 1 been excited.}
 ELSE { $\boldsymbol{w}_{\text{Con}-1} = [0\ \ 1]$, for indicating output node 2 been excited.}
ELSE for each newly incoming input $\mathbf{x}$, do
PART 2. { Find $J = \arg\max_{1\le j\le c(t)} F^j(\mathbf{X})$, as defined in (9).
 IF $\boldsymbol{w}_{\text{Con-J}} \neq \mathbf{y}$,
  { set $c(t+1) = c(t) + 1$ and generate a new fuzzy rule, with $\mathbf{m}_{c(t+1)} = \mathbf{x}$, $\mathbf{D}_{c(t+1)} = (-1/\chi)\mathrm{diag}\left(1/\ln\left(F^J\right),\ldots,1/\ln\left(F^J\right)\right)$ and $\boldsymbol{w}_{\text{Con}-c(t+1)} = \mathbf{y}$, where $\chi$ decides the overlap degree between two clusters. In addition, after decomposition, we have $m_{new-i} = x_i$, $\sigma_{new-i} = -\chi \times \ln(F^J)$, $i = 1,\ldots,M$. Do the following fuzzy measure for each input variable $i$:
Degree$(i,t) \equiv \max_{1\le j\le k_i} E\left[\mu(m_{\text{new}-\text{i}},\sigma_{\text{new}-\text{i}}),\mu(m_{ij},\sigma_{ij})\right]$
   where $E(\bullet)$ is defined in (14).
   IF Degree$(i,t) \le \rho(t)$,
   THEN adopt this new membership function, and set $k_i = k_i + 1$, where $k_i$ is the number of partitions of the $i$th input variable.
   ELSE merge the new membership function with closest one
      $m_{\text{new}-\text{i}} = m_{\text{closest}} = \frac{m_{\text{new}-\text{i}} + m_{\text{closest}}}{2}$
      $\sigma_{\text{new}-\text{i}} = \sigma_{\text{closest}} = \frac{\sigma_{\text{new}-\text{i}} + \sigma_{\text{closest}}}{2}$
  } } ELSE
 { If $F^J \le \overline{F}_{in}(t)$
  { generate a new fuzzy rule with $\mathbf{m}_{c(t+1)} = \mathbf{x}$, $\mathbf{D}_{c(t+1)} = (-1/\chi)\mathrm{diag}\left(1/\ln\left(F^J\right),\ldots,1/\ln\left(F^J\right)\right)$, and the respective consequent weight $\boldsymbol{w}_{\text{Con-a}(t+1)} = \mathbf{y}$. In addition, we also need to do the fuzzy measure for each input variable $i$.
} } }
```

In the previous algorithm, σ_{init} is a prespecified constant, $c(t)$ is the rule number at time t, χ decides the overlap degree between two clusters, and the threshold $\overline{F}_{in}$ determines the number of rules generated. For a higher value of $\overline{F}_{in}$, more rules are generated and, in general, a higher accuracy is achieved. The value $\rho(t)$ is a scalar similarity criterion, which is monotonically decreasing such that higher similarity between two fuzzy sets is allowed in the initial stage of learning. The prespecified values are given heuristically. In general, $\overline{F}(t) = 0.35$, $\beta = 0.05$, $\sigma_{\text{init}} = 0.5$, $\chi = 2$. In addition, after we determine the precondition part of fuzzy rule, we also need to properly assign the consequence part of fuzzy rule. Here we define two output nodes for doing two-cluster recognition. If output node 1 obtains higher exciting value, we know this input–output pattern belongs to class 1. Hence, initially, we should assign the proper weight $\boldsymbol{w}_{Con-1}$ for the consequence part of fuzzy rule. The above procedure gives us means $(\mathbf{m}_{ij})$ and variances (σ_{ij}^2) in (9). Another parameter in (7) that needs concern is the weight d_j associated with each $a_j^{(4)}$. We shall see later in **Learning Phase 2** how we can use the results from the SVM method to determine these weights.

Learning Phase 2—Calculating the parameters of SVFNN

Through learning phase (1), the initial structure of SVFNN is established and we can then use SVM [30] to find the optimal parameters of SVFNN based on the proposed fuzzy kernels. The dual quadratic optimization of SVM [31] is solved in order to obtain an optimal hyperplane for any linear or nonlinear space:

$$\text{maximize} \quad L\left(\overrightarrow{\alpha}\right) = \sum_{i=1}^{v}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{v} y_i y_j \alpha_i \alpha_j K\left(\mathbf{x}_i,\mathbf{x}_j\right)$$
$$\text{subject to} \quad 0 \le \alpha_i \le C,\ i = 1,2,\ldots,v$$

$$\text{and } \sum_{i=1}^{v} y_i \alpha_i = 0 \tag{24}$$

where $K(\mathbf{x}_i, \mathbf{x}_j)$ is the fuzzy kernel in (17) and C is a user-specified positive parameter to control the tradeoff between complexity of the SVM and the number of nonseparable points. This quadratic optimization problem can be solved and a solution $\overrightarrow{\alpha}_0 = \left(\alpha_1^0, \alpha_2^0, \ldots, \alpha_{nsv}^0\right)$ can be obtained, where α_i^0 are Lagrange coefficients, and nsv is the number of support vectors. The corresponding support vectors $\mathbf{sv} = [\mathbf{sx}_1, \mathbf{sx}_2, \ldots, \mathbf{sx}_i, \ldots \mathbf{sx}_{nsv}]$ can be obtained, and the constant (threshold) d_0 in (7) is

$$\begin{aligned} d_0 &= \frac{1}{2}\left[(w_0 \cdot x^*(1)) + (w_0 \cdot x^*(-1))\right] \text{ with} \\ w_0 &= \sum_{i=1}^{nsv} \alpha_i y_i x_i \end{aligned} \tag{25}$$

where nsv is the number of fuzzy rules (support vectors); the support vector $x^*(1)$ belongs to the first class and support vector $x^*(-1)$ belongs to the second class. Hence, the fuzzy rules of SVFNN are reconstructed by using the result of the SVM learning with fuzzy kernels. The means and variances of the membership functions can be calculated by the values of support vector $\mathbf{m}_j = \mathbf{sx}_j$, $j = 1, 2, \ldots, nsv$, in (5) and (6) and the variances of the multidimensional membership function of the cluster that the support vector belongs to, respectively. The coefficients d_j in (7) corresponding to $\mathbf{m}_j = \mathbf{sx}_j$ can be calculated by $d_j = y_j \alpha_j$. In this phase, the number of fuzzy rules can be increased or decreased. The adaptive fuzzy kernel is advantageous to both the SVM and the FNN. The use of variable-width fuzzy kernels makes the SVM more efficient in terms of the number of required support vectors, which are corresponding to the fuzzy rules in SVFNN.

Learning Phase 3—Removing irrelevant fuzzy rules

In this phase, we propose a method for reducing the number of fuzzy rules learning in Phases 1 and 2 by removing some irrelevant fuzzy rules and retuning the consequent parameters of the remaining fuzzy rules under the condition that the classification accuracy of SVFNN is kept almost the same. Several methods including orthogonal least squares (OLS) method and singular value decomposition QR (SVD-QR) had been proposed to select important fuzzy rules from a given rule base [32]–[34]. In [32] the SVD-QR algorithm select a set of independent fuzzy basis function that minimize the residual error in a least squares sense. In [33], an orthogonal least-squares method tries to minimize the fitting error according to the error reduction ratio rather than simplify the model structure [34]. The proposed method reduces the number of fuzzy rules by minimizing the distance measure between original fuzzy rules and reduced fuzzy rules without losing the generalization performance. To achieve this goal, we rewrite (8) as

$$\begin{aligned} O^{(4)} &= \sum_{j=1}^{N} d_j \times a_j^{(4)} + d_0 \\ &= \sum_{j=1}^{N} d_j \prod_{i=1}^{M} e^{-\left((x_i - m_{ij})^2/\sigma_{ij}^2\right)} + d_0 \end{aligned} \tag{26}$$

where N is the number of fuzzy rules after Learning phases 1 and 2. Now, we try to approximate it by the expansion of a reduced set

$$\begin{aligned} O^{\mathrm{Re}(4)} &= \sum_{q=1}^{R_z} \beta_q \times a_q^{\mathrm{Re}(4)} + d_0 \\ &= \sum_{q=1}^{R_z} \beta_q \prod_{i=1}^{M} e^{-\left((x_i - m_{iq}^{\mathrm{Re}})^2/\sigma_{iq}^{\mathrm{Re}^2}\right)} + d_0 \\ \text{and } a_q^{\mathrm{Re}(4)}(\mathbf{x}) &= \prod_{i=1}^{M} e^{-\left((x_i - m_{iq}^{\mathrm{Re}})^2/\sigma_{iq}^{\mathrm{Re}^2}\right)} \end{aligned} \tag{27}$$

where R_z is the number of reducing fuzzy rules with $N > R_z$, β_q is the consequent parameters of the remaining fuzzy rules, and m_{iq}^{Re} and $\sigma_{iq}^{\mathrm{Re}}$ are the mean and variance of reducing fuzzy rules. To this end, one can minimize [35]

$$\begin{aligned} \left\| O^{(4)} - O^{\mathrm{Re}(4)} \right\|^2 = &\sum_{j,q=1}^{N} d_j \times d_q \times a_j^{(4)}(\mathbf{m}_q) \\ &+ \sum_{j,q=1}^{R_z} \beta_j \times \beta_q \times a_j^{\mathrm{Re}(4)}\left(\mathbf{m}_q^{\mathrm{Re}}\right) \\ &- 2 \times \sum_{j=1}^{N} \sum_{q=1}^{R_z} d_j \times \beta_q \times a_j^{(4)}\left(\mathbf{m}_q^{\mathrm{Re}}\right) \end{aligned} \tag{28}$$

where $\mathbf{m}_q^{\mathrm{Re}} = \left[m_{1q}^{\mathrm{Re}}, m_{2q}^{\mathrm{Re}}, \ldots, m_{Mq}^{\mathrm{Re}}\right]^T$. Evidently, the problem of finding reduced fuzzy rules consists of two parts: one is to determine the reduced fuzzy rules and the other is to compute the expansion coefficients β_i. This problem can be solved by choosing the more important R_z fuzzy rules from the old N fuzzy rules. By adopting the sequential optimization approach in the reduced support vector method in [36], the approximation in (27) can be achieved by computing a whole sequence of reduced set approximations

$$O_r^{\mathrm{Re}(4)} = \sum_{q=1}^{r} \beta_q \times a_q^{\mathrm{Re}(4)} \tag{29}$$

for $r = 1, 2, \ldots, R_Z$. Then, the mean and variance parameters, $\mathbf{m}_q^{\mathrm{Re}}$ and σ_q^{Re}, in the expansion of the reduced fuzzy-rule set in (27) can be obtained by the following iterative optimization rule [36]:

$$\mathbf{m}_{q+1}^{\mathrm{Re}} = \frac{\sum_{j=1}^{N} d_j \times a_j^{(4)}\left(\mathbf{m}_q^{\mathrm{Re}}\right) \times \mathbf{m}_j}{\sum_{j=1}^{N} d_j \times a_j^{(4)}\left(\mathbf{m}_q^{\mathrm{Re}}\right)}. \tag{30}$$

According to (30), we can find the parameters, $\mathbf{m}_q^{\mathrm{Re}}$ and σ_q^{Re}, corresponding to the first most important fuzzy rule and then remove this rule from the original fuzzy rule set represented by $\mathbf{m}_j$, $j = 1, 2, \ldots, N$ and put (add) this rule into the reduced fuzzy rule set. Then the procedure for obtaining the reduced rules is repeated. The optimal coefficients β_q, $q = 1, 2, \ldots, R_z$, are then computed to approximate $O^{(4)} = \sum_{j=1}^{N} d_j \times a_j$ by $O^{\mathrm{Re}(4)} = \sum_{q=1}^{R_z} \beta_q \times a_q^{\mathrm{Re}}$ [36], and can be obtained as

$$\beta = [\beta_1, \beta_2, \ldots, \beta_{R_z}] = \mathbf{K}_{R_z \times R_z}^{-1} \times \mathbf{K}_{R_z \times N} \times \Theta \tag{31}$$

where (32) and (33), as shown at the bottom of the page, hold, and

$$\Theta = [d_1, d_2, \ldots, d_N]. \tag{34}$$

The whole learning scheme is iterated until the new rules are sufficiently sparse.

V. Experimental Results

The classification performance of the proposed SVFNN is evaluated on five well-known benchmark datasets. These five datasets can be obtained from the UCI repository of machine learning databases [37] and the Statlog collection [38] and IJCNN challenge 2001 [39], [40], respectively.

A. Data and Implementation

From the UCI Repository, we choose one dataset: Iris dataset. From Statlog collection we choose three datasets: Vehicle, Dna and Satimage datasets. The problem Ijcnn1 is from the first problem of IJCNN challenge 2001. These five datasets will be used to verify the effectiveness of the proposed SVFNN classifier. The first dataset (Iris dataset) is originally a collection of 150 samples equally distributed among three classes of the Iris plant namely Setosa, Verginica, and Versicolor. Each sample is represented by four features (septal length, septal width, petal length, and petal width) and the corresponding class label. The second dataset (Vehicle dataset) consists of 846 samples belonging to 4 classes. Each sample is represented by 18 input features. The third dataset (Dna dataset) consists of 3186 feature vectors in which 2000 samples are used for training and 1186 samples are used for testing. Each sample consists of 180 input attributes. The data are classified into three physical classes. All Dna examples are taken from Genbank 64.1. The four dataset (Satimage dataset) is generated from Landsat Multispectral Scanner image data. In this dataset, 4435 samples are used for training and 2000 samples are used for testing. The data are classified into six physical classes. Each sample consists of 36 input attributes. The five dataset (Ijcnn1 dataset) consists of 22 feature vectors in which 49 990 samples are used for training and 45 495 samples are used for testing. Each sample consists of 22 input attributes. The data are classified into two physical classes. The computational experiments were done on a Pentium III-1000 with 1024MB RAM using the Linux operation system.

For each problem, we estimate the generalized accuracy using different cost parameters $C = [2^{12}, 2^{11}, 2^{10}, \ldots, 2^{-2}]$ in (24). We apply 2-fold cross-validation for 10 times on the whole training data in Dna, Satimage and Ijcnn1, and then average all the results. We choose the cost parameter C that results in the best average cross-validation rate for SVM training to predict the test set. Because Iris and Vehicle datasets don't contain testing data explicitly, we divide the whole data in Iris and Vehicle datasets into two halves, for training and testing datasets, respectively. Similarly, we use the above method to experiment. Notice that we scale all training and testing data to be in $[-1, 1]$.

B. Experimental Results

Tables I–V present the classification accuracy rates and the number of used fuzzy rules (i.e., support vectors) in the SVFNN on Iris, Vehicle, Dna, Satimage and Ijcnn1 datasets, respectively. The criterion of determining the number of reduced fuzzy rules is the difference of the accuracy values before and after reducing one fuzzy rule. If the difference is larger than 0.5%, meaning that some important support vector has been removed, then we stop the rule reduction. In Table I, the SVFNN is verified by using Iris dataset, where the constant n in the symbol SVFNN-n means the number of the learned fuzzy rules. The SVFNN uses fourteen fuzzy rules and achieves an error rate of 2.6% on the training data and an error rate of 4% on the testing data. When the number of fuzzy rules is reduced to seven, its error rate increased to 5.3%. When the number of fuzzy rules is reduced to four, its error rate is increased to 13.3%. Continuously decreasing the number of fuzzy rules will keep the error rate increasing. From Tables II–V, we have the similar experimental results as those in Table I.

$$\mathbf{K}_{R_z \times R_z} = \begin{pmatrix} a_1^{\mathrm{Re}(4)}\left(\mathbf{m}_1^{\mathrm{Re}}\right) & a_1^{\mathrm{Re}(4)}\left(\mathbf{m}_2^{\mathrm{Re}}\right) & \cdots & a_1^{\mathrm{Re}(4)}\left(\mathbf{m}_{R_z}^{\mathrm{Re}}\right) \\ a_2^{\mathrm{Re}(4)}\left(\mathbf{m}_1^{\mathrm{Re}}\right) & a_2^{\mathrm{Re}(4)}\left(\mathbf{m}_2^{\mathrm{Re}}\right) & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{R_z-1}^{\mathrm{Re}(4)}\left(\mathbf{m}_{R_z}^{\mathrm{Re}}\right) \\ a_{R_z}^{\mathrm{Re}(4)}\left(\mathbf{m}_1^{\mathrm{Re}}\right) & \cdots & a_{R_z}^{\mathrm{Re}(4)}\left(\mathbf{m}_{R_z-1}^{\mathrm{Re}}\right) & a_{R_z}^{\mathrm{Re}(4)}\left(\mathbf{m}_{R_z}^{\mathrm{Re}}\right) \end{pmatrix} \tag{32}$$

$$\mathbf{K}_{R_z \times N} = \begin{pmatrix} a_1^{\mathrm{Re}(4)}(\mathbf{m}_1) & a_1^{\mathrm{Re}(4)}(\mathbf{m}_2) & \cdots & a_1^{\mathrm{Re}(4)}(\mathbf{m}_{R_x}) \\ a_2^{\mathrm{Re}(4)}(\mathbf{m}_1) & a_2^{\mathrm{Re}(4)}(\mathbf{m}_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{R_z-1}^{\mathrm{Re}(4)}(\mathbf{m}_N) \\ a_{R_z}^{\mathrm{Re}(4)}(\mathbf{m}_1) & \cdots & a_{R_z}^{\mathrm{Re}(4)}(\mathbf{m}_{N-1}) & a_{R_z}^{\mathrm{Re}(4)}(\mathbf{m}_N) \end{pmatrix} \tag{33}$$

TABLE I
EXPERIMENTAL RESULTS OF SVFNN CLASSIFICATION ON THE IRIS DATASET

SVFNN-*n* (SVFNN with *n* fuzzy rules)	Training process		Testing process	
	Error rate	C	Number of misclassification	Error rate
SVFNN-14	2.6%	2^{12}	3	4%
SVFNN -11	2.6%	2^{12}	3	4%
SVFNN -9	2.6%	2^{12}	3	4%
SVFNN -7	4%	2^{12}	4	5.3%
SVFNN -4	17.3%	2^{12}	10	13.3%
1. Input dimension is 4.				
2. The number of training data is 75.				
3. The number of testing data is 75.				

TABLE II
EXPERIMENTAL RESULTS OF SVFNN CLASSIFICATION ON THE VEHICLE DATASET

SVFNN-*n* (SVFNN with *n* fuzzy rules)	Training process		Testing porcess	
	Error rate	C	Number of misclassification	Error rate
SVFNN-321	13.1%	2^{11}	60	14.2%
SVFNN-221	13.1%	2^{11}	60	14.2%
SVFNN-171	13.1%	2^{11}	60	14.2%
SVFNN-125	14.9%	2^{11}	61	14.5%
SVFNN-115	29.6%	2^{11}	113	26.7%
1. Input dimension is 18.				
2. The number of training data is 423.				
3. The number of testing data is 423.				

TABLE III
EXPERIMENTAL RESULTS OF SVFNN CLASSIFICATION ON THE DNA DATASET

SVFNN-*n* (SVFNN with *n* fuzzy rules)	Training process		Testing process	
	Error Rate	C	Number of misclassification	Error rate
SVFNN-904	6.2%	2^4	64	5.4%
SVFNN-704	6.2%	2^4	64	5.4%
SVFNN-504	6.2%	2^4	64	5.4%
SVFNN-334	6.4%	2^4	69	5.8%
SVFNN-300	9.8%	2^4	139	11.7%
1. Input dimension is 180.				
2. The number of training data is 2000.				
3. The number of testing data is 1186.				

TABLE IV
EXPERIMENTAL RESULTS OF SVFNN CLASSIFICATION ON THE SATIMAGE DATASET

SVFNN-*n* (SVFNN with *n* fuzzy rules)	Training process		Testing process	
	Error Rate	C	Number of misclassification	Error Rate
SVFNN-1886	13.1%	2^6	176	8.8%
SVFNN-1586	13.1%	2^6	176	8.8%
SVFNN-1286	13.1%	2^6	176	8.8%
SVFNN-986	13.1%	2^6	176	8.8%
SVFNN-899	13.7%	2^6	184	9.2%
SVFNN-786	19.8%	2^6	316	15.8%
1. Input dimension is 36.				
2. The number of training data is 4435.				
3. The number of testing data is 2000.				

TABLE V
EXPERIMENTAL RESULTS OF SVFNN CLASSIFICATION ON THE IJNN1 DATASET

SVFNN-*n* (SVFNN with *n* fuzzy rules)	Training process		Testing porcess	
	Error rate	C	Number of misclassification	Error rate
SVFNN-1945	4.2%	2^{12}	1955	4.3%
SVFNN-1545	4.2%	2^{12}	1955	4.3%
SVFNN-1245	4.2%	2^{12}	1955	4.3%
SVFNN-1021	4.3%	2^{12}	2047	4.5%
SVFNN-977	14.5%	2^{12}	7416	16.3%
1. Input dimension is 22.				
2. The number of training data is 49990.				
3. The number of testing data is 45495.				

These experimental results show that the proposed SVFNN is good at reducing the number of fuzzy rules and maintaining the good generalization ability. Moreover, we also refer to some recent other classification performance include support vector machine and reduced support vectors methods [41]–[43]. The performance comparisons among the existing fuzzy neural network classifiers [44], [45], the RBF-kernel-based SVM (without support vector reduction) [41], reduced support vector machine (RSVM) [43] and the proposed SVFNN are made in Table VI. These results indicate that the SVFNN classifier produces lower testing error rates as compared to FNN classifiers [44], [45], and uses less support vectors as compared to the regular SVM using fixed-width RBF kernels [41]. As compared to RSVM [43], the proposed SVFNN can not only achieve high classification accuracy, but also reduce the number of support vectors quit well. It is noticed that although the SVFNN uses more support vectors in the Ijcnn1 dataset than the RSVM, it maintains much higher classification accuracy than the RSVM. In summary, the proposed SVFNN classifier exhibits better generalization ability on the testing data and use much smaller number of fuzzy rules.

VI. CONCLUSION

This paper proposed an SVFNN, which combines the superior classification power of SVMs in high-dimensional data spaces and the efficient human-like reasoning of FNN in handling uncertainty information. The SVFNN is the realization of a new idea for the adaptive kernel functions used in the SVM. The use of the proposed fuzzy kernels provides the SVM with adaptive local representation power, and thus brings the advantages of FNN (such as adaptive learning and economic network structure) into the SVM directly. The major advantages of the proposed SVFNN classification are as follows.

1) The proposed SVFNN can automatically generate fuzzy rules, and improve the accuracy and learning speed of classification.
2) It combined the optimal classification ability of SVM and the human-like reasoning of fuzzy systems. It improved the classification ability by giving SVM with adaptive

TABLE VI
CLASSIFICATION ERROR RATE COMPARISONS AMONG FNN, RBF-KERNEL-BASED SVM, RSVM AND SVFNN CLASSIFIERS, WHERE NA MEANS "NOT AVAILABLE"

Datasets	FNN [44, 45]		RBF-kernel-based SVM [41]		RSVM [43]		SVFNN	
	Number of fuzzy rules	Error rate	Number of support vectors	Error rate	Number of support vectors	Error rate	Number of Fuzzy rules	Error rate
Iris	NA	4.3%	16	3.3%	NA	NA	7	5.3%
Vehicle	NA	29.9%	343	13.4%	NA	NA	125	14.5%
Dna	NA	16.64%	1152	4.21%	372	7.7%	334	5.8%
Satimage	NA	8.9%	2170	8.3%	1826	10.1%	889	9.2%
Ijcnn1	NA	NA	4555	1.2%	200	8.4%	1021	4.5%

fuzzy kernels and increased the speed of classification by reduced fuzzy rules.

3) The fuzzy kernels using the variable-width fuzzy membership functions can make the SVM more efficient in terms of the number of required support vectors, and also make the learned FNN more understandable to human.
4) The ability of the structural risk minimization induction principle, which forms the basis for the SVM method to minimize the expected risk, gives better generalization ability to the FNN classification.

REFERENCES

[1] K. Tanaka and H. O. Wang, *Fuzzy Control Systems Design and Analysis*. New York: Wiley, 2001.
[2] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[3] M. Y. Chen and D. A. Linkens, "Rule-base self-generation and simplification for data-driven fuzzy models," *Fuzzy Sets Syst.*, vol. 142, pp. 243–265, Mar. 2004.
[4] J. S. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man. Cybern.*, vol. 23, no. 3, pp. 665–685, May 1993.
[5] K. Tanaka, M. Sano, and H. Wantanabe, "Modeling and control of carbon monoxide concentration using a neuro-fuzzy technique," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 4, pp. 271–279, Aug. 1995.
[6] L. Y. Cai and H. K. Kwan, "Fuzzy classifications using fuzzy inference networks," *IEEE Trans. Syst., Man, Cybern.*, pt. B, vol. 28, no. 4, pp. 334–347, Jun. 1998.
[7] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.
[8] J. C. Bezdek, S. K. Chuah, and D. Leep, "Generalized K-nearest neighbor rules," *Fuzzy Sets Syst.*, vol. 18, pp. 237–256, Apr. 1986.
[9] J.-S. Wang and C. S. G. Lee, "Self-Adaptive neuro-fuzzy inference systems for classification applications," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 6, pp. 790–802, Dec. 2002.
[10] L. I. Kuncheva, "How good are fuzzy IF-THEN classifiers?," *IEEE Trans. Syst., Man, Cybern.*, pt. B, vol. 30, pp. 501–509, Aug. 2000.
[11] H. Ishibuchi and T. Nakashima, "Effect of rule weights in fuzzy rule-based classification systems," *IEEE Trans. Fuzzy Syst.*, vol. 9, pp. 506–515, Aug. 2001.
[12] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Trans. Neural Networks*, vol. 11, no. 3, pp. 769–783, May 2000.
[13] K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive fuzzy rule-based classification system," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 4, pp. 238–250, Aug. 1996.
[14] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
[15] S. Sohn and C. H. Dagli, "Advantages of using fuzzy class memberships in self-organizing map and support vector machines," in *Proc. Int. Joint Conf. Neural Networks (IJCNN'01)*, vol. 3, Jul. 2001, pp. 1886–1890.
[16] C. F. Lin and S. D. Wang, "Fuzzy support vector machines," *IEEE Trans. Neural Networks*, vol. 13, pp. 464–471, Mar. 2002.
[17] T. Inoue and S. Abe, "Fuzzy support vector machines for pattern classification," in *Proc. Int. Joint Conf. Neural Networks (IJCNN'01)*, vol. 2, Jul. 2001, pp. 15–19.
[18] J. T. Jeng and T. T. Lee, "Support vector machines for the fuzzy neural networks," in *IEEE Int. Conf. Systems, Man, and Cybernetics (SMC'99)*, vol. 6, Oct. 1999, pp. 12–15.
[19] 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.
[20] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*. New York: Wiley, 1999.
[21] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, Dec. 1991.
[22] J. Platt, "A resource allocating network for function interpolation," *Neural Computat.*, vol. 3, pp. 213–225, 1991.
[23] J. Nie and D. A. Linkens, "Learning control using fuzzified self-organizing radial basis function network," *IEEE Trans. Fuzzy Syst.*, vol. 40, no. 6, pp. 280–287, Nov. 1993.
[24] C. T. Lin and C. S. G. Lee, "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 1, pp. 46–63, Feb. 1994.
[25] A. Papoulis, *Probability Random Variables and Stochastic Processes*. New York: McGraw-Hill, 1984.
[26] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philo. Trans. Royal Soc. London*, vol. A209, pp. 415–446, 1909.
[27] S. Saitoh, *Theory of Reproducing Kernels and Its Application*. White Plains, NY: Longman.
[28] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[29] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1985.
[30] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1990.
[31] B. Schölkopf, C. J. C. Burges, and A. J. Smola, *Advances in Kernel Methods—Support Vector Learning*. Cambridge, MA: MIT Press, 1999.
[32] J. Hohensoh and J. M. Mendel, "Two-pass orthogonal least-squares algorithm to train and reduce the complexity of fuzzy logic systems," *J. Intell. Fuzzy Syst.*, vol. 4, pp. 295–308, 1996.
[33] G. Mouzouris and J. M. Mendel, "A singular-value-QR decomposition based method for training fuzzy logic systems in uncertain environments," *J. Intell. Fuzzy Syst.*, vol. 5, pp. 367–374, 1997.
[34] J. Yen and L. Wang, "Simplifying fuzzy rule-based models using orthogonal transformation methods," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, no. 1, pp. 13–24, Feb. 1999.
[35] C. J. C. Burges, "Simplified support vector decision rules," in *Proc. 13th Int. Conf. Machine Learning*, L. Saitta, Ed., San Mateo, CA, 1996, pp. 71–77.

[36] B. Scholkopf *et al.*, “Input space versus feature space in kernel-based methods,” *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1000–1017, Sep. 1999.
[37] C. L. Blake and C. J. Merz. (1998) UCI repository of machine learning databases. Dept. Inform. Comput. Sci., Univ. California, , Irvine, CA. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[38] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. (1994) Machine learning, neural and statistical classification. [Online]. Available: ftp://ftp.stams.strath.ac.uk/pub/
[39] D. Prokhorov. IJCNN 2001 neural network competition. presented at Slide Presentation in IJCNN’01. [Online]. Available: http://www.geocities.com/ijcnn/nncijcnn01.pdf
[40] LIBSVM Datasets of IJCNN2001. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[41] C. W. Hsu and C. J. Lin, “A comparison of methods for multiclass support vector machines,” *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 415–525, Mar. 2002.
[42] Y. J. Lee and O. L. Mangasarian, “RSVM: Reduced support vector machines,” in *Proc. 1st SIAM Int. Conf. Data Mining*, 2001, pp. 1–17.
[43] K. M. Lin and C. J. Lin, “A study on reduced support vector machines,” *IEEE Trans. Neural Networks*, vol. 14, no. 6, pp. 1449–1459, Nov. 2003.
[44] 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, no. 3, pp. 426–432, Jun. 2001.
[45] M. R. Berthold and J. Diamond, “Constructive training of probabilistic neural networks,” *Neurocomput.*, vol. 19, pp. 167–183, 1998.

Chin-Teng Lin (F’05) received the B.S. degree from National Chiao-Tung University (NCTU), Taiwan, in 1986, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1992.

He is currently the Chair Professor of Electrical and Computer Engineering, Dean of Computer Science College, and Director of Brain Research Center at NCTU. He served as the Director of the Research and Development Office of NCTU from 1998 to 2000, the Chairman of Electrical and Control Engineering Department of NCTU from 2000 to 2003, and Associate Dean of the College of Electrical Engineering and Computer Science from 2003 to 2005. His current research interests are fuzzy neural networks, neural networks, fuzzy systems, cellular neural networks, 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 coauthor of the book *Neural Fuzzy Systems- A Neuro-Fuzzy Synergism to Intelligent Systems* (Prentice Hall, 1996), and the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (World Scientific, 1994). He has published over 90 journal papers in the areas of neural networks, fuzzy systems, multimedia hardware/software, and soft computing, including about 60 IEEE journal papers.

Dr. Lin served on the Board of Governors of the IEEE Circuits and Systems (CAS) Society in 2005 and the IEEE Systems, Man, Cybernetics (SMC) Society in 2003–2005. He was the Distinguished Lecturer of the IEEE CAS Society from 2003 to 2005. He is the International Liaison of the International Symposium of Circuits and Systems (ISCAS) 2005, in Japan, the Special Session Co-Chair of ISCAS 2006, in Greece, and the Program Co-Chair of IEEE International Conference on SMC 2006, in Taiwan. He has been the President of Asia Pacific Neural Network Assembly since 2004. He has received the Outstanding Research Award granted by National Science Council, Taiwan, since 1997 to the 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 (2000). Dr. Lin currently serves as an Associate Editors of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, PARTS I AND PART II, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, the IEEE TRANSACTIONS ON FUZZY SYSTEMS, and the *International Journal of Speech Technology*. He is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi honorary societies.

Chang-Mao Yeh received the B.S. degree in electronics engineering and the M.S. degree in computer science and information engineering from the National Yunlin University of Science and Technology, Yunlin, Taiwan, R.O.C., in 1994 and 1997, respectively. He is currently working toward the Ph.D. degree in electrical and control engineering at the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

He is a Lecturer in Information Networking Technology at Chungchou Institute of Technology, Taichung, Taiwan, R.O.C. His current research interests include neuro-fuzzy systems, support vector machine, machine learning, and pattern recognition.

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

From 2001 to 2005, he was a Research Assistant Professor in Electrical and Control Engineering, NCTU. In 2005, he joined the Department of Biological Science and Technology, NCTU, where he serves as an Assistant Professor. He has served as the Chief Executive of the Brain Research Center, NCTU Branch, University System of Taiwan, since September 2003. His current research interests are biomedical engineering, biomedical signal/image processing, machine learning, fuzzy neural networks (FNN), the development of brain-computer interface (BCI), and multimedia signal processing.

Jen-Feng Chung received the B.S. degree in computer science and information engineering from the Chung-Hua University, Hsinchu, Taiwan, and the M.S. degree in electrical engineering from the Chung-Hua University, Hsinchu, Taiwan, 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.

His current research interests are neural networks, VLSI signal processing, audio and image signal processing, and DSP architecture design.

Nimit Kumar recieved the B.Tech. degree from the Indian Institute of Technology, Kanpur, in 2004.

He is currently a Research Trainee at IBM India Research Lab., Delhi, India. His research focuses on statistical learning theory, kernel-based learning, neuro-fuzzy systems, web and text mining, visuo-motor control, and the natural basis of learning.

Mr. Kumar has served as a reviewer for the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS B and the IEEE TRANSACTIONS ON NEURAL NETWORKS.