International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 1, Number 1 (2011), pp. 31-54
© International Research Publications House
http://www. ripublication.com

Support-Vector-Based Fuzzy Neural Networks

Chin-Teng Lin[1], Chang-Mao Yeh[2], Jen-Feng Chung[3], Sheng-Fu Liang[4] and Her-Chang Pu[5]

[1]National Chiao-Tung University, Department of Computer Science, Department of Electrical and Control Engineering, Ta Hsueh Road, Hsinchu, Taiwan
[2,3,4]National Chiao-Tung University, Department of Electrical and Control Engineering, Ta Hsueh Road, Hsinchu, Taiwan
[5]National Chiao-Tung University, Department of Biological Science and Technology, Ta Hsueh Road, Hsinchu, Taiwan

Abstract

In this paper, novel fuzzy neural networks (FNNs) combining with support vector learning mechanism called support-vector-based fuzzy neural networks (SVFNNs) are proposed for pattern classification and function approximation. The SVFNNs combine the capability of minimizing the empirical risk (training error) and expected risk (testing error) of support vector learning 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 SVFNNs and train the 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 and SVR with the proposed adaptive fuzzy kernel function for pattern classification and function approximation, respectively. In the third phase, the relevant fuzzy rules are selected by the proposed fuzzy rule reduction method. To investigate the effectiveness of the proposed SVFNNs, they are applied to the Iris and Vehicle datasets for classification, and one- and two- variable functions for approximation, respectively. Experimental results show that the proposed SVFNNs can achieve good pattern classification and function approximation performance with drastically reduced number of fuzzy kernel functions (fuzzy rules).

Keywords: Fuzzy neural network, fuzzy kernel function, support vector machine, support vector regression, pattern classification, function approximation.

Introduction

It is an important key issue in many scientific and engineering fields to classify the acquired data or estimate an unknown function from a set of input-output data pairs. As is widely known, fuzzy neural networks (FNNs) have been proposed and successfully applied to solving these problems such as classification, identification, control, pattern recognition, and image processing, etc [1]-[4]. A fuzzy system consists of a bunch of fuzzy if-then rules. Conventionally, the fuzzy if-then rules were usually derived from human experts as linguistic knowledge. Obviously, it is not always easy to derive fuzzy rules from human experts or to examine all the input-output data to find a number of proper rules for the fuzzy system. So, most previous researches issue the method of automatically generating fuzzy rules from numerical data and use the backpropagation (BP) and/or C-cluster type learning algorithms to train parameters of fuzzy rules and membership functions from the training data [5], [6]. However, such learning algorithm only aims at minimizing the training error, and it cannot guarantee the lowest testing error rate in the testing phase. In addition, the local solutions and slow convergence often impose practical constraints in the function approximation problems [7].

In statistical learning theory, the support vector machine (SVM) [8] has been developed for solving these bottlenecks. SVM performs structural risk minimization and creates a classifier with minimized Vapnik Chervonenkis (VC) dimension [9]. As the VC dimension is low, the expected probability of error is low to ensure a good generalization. When SVM is employed to tackle the problems of function approximation and regression estimation, it is referred as the support vector regression (SVR) [10]. SVR can perform high accuracy and robust properties for function approximation with noise [11]. Some researches have been done on combining SVM with FNN [12]-[15]. In [12], the support vector learning mechanism provides the architecture to extract support vectors for generating fuzzy IF-THEN rules from the training data set. This method provides reliable performance in the cases of prediction. In [13], a self-organizing map with fuzzy class memberships is used to reduce the training samples to speed up the SVM training. The methods in [14]-[15] improve the accuracy of multi-class pattern recognition and regression estimation problems and reduce the influence of noises. However, the regular SVM and SVR suffer from the difficulty of long computational time in using nonlinear kernels on large datasets which come from many real applications.

In this paper, novel support-vector-based fuzzy neural networks (SVFNNs) which integrate the statistical support vector learning method into FNN and exploit the knowledge representation power and learning ability of the FNN to determine the kernel functions of the SVM/SVR adaptively are proposed. The SVFNNs combine the capability of minimizing the empirical risk (training error) and expected risk (testing error) of support vector learning in high dimensional data spaces and the efficient human-like reasoning of FNN in handling uncertainty information. In addition, we also propose a novel adaptive fuzzy kernel function, which has been proven to be a Mercer kernel, to bring the advantages of FNNs (such as adaptive learning and economic network structure) to SVM/SVR. The use of the proposed fuzzy kernels provides the SVM/SVR with adaptive local representation power such that the number of support vectors can be further reduced. The proposed learning algorithm consists of three learning phases to construct and train the SVFNNs. In the first phase, the fuzzy rules and membership functions are automatically determined based on the fuzzy clustering method. In the second phase, the parameters of FNN are calculated by the SVM and SVR with the proposed adaptive fuzzy kernel function for pattern classification and function approximation, respectively. In the third phase, the relevant fuzzy rules are selected by the proposed fuzzy rule reduction method. The proposed SVFNNs are applied to the Iris and Vehicle datasets for classification and one- and two-variable functions for approximation. Experimental results show that the proposed SVFNNs can automatically generate the fuzzy rules, improve the accuracy of pattern classification and function approximation, reduce the number of required kernel functions, and increase the speed in test phase.

The rest of this paper is organized as follows. Section II describes the structure and initial construction of SVFNNs. The learning algorithm of SVFNNs is developed in Section III. In Section IV, performance comparisons between SVFNNs and other classification and function approximation methods are made. Finally, the conclusions are summarized in Section V.

Structure and Construction of Initial Fuzzy Neural Network

Structure of Fuzzy Neural Network

The proposed SVFNN is a four-layered FNN that is comprised of the input, membership function, fuzzy rules, and output layers as shown in Fig.1. 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 links 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:

Rule R_j: IF x_1 is A_{1j} and …x_i is A_{ij}….. and x_M is A_{Mj},

THEN y is d_j , j=1, 2, ⋯, N, (1)

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

Rule 0: IF x_1 is A_{10} and … and x_M is A_{M0},

THEN y is d_0, (2)

where A_{k0} is a universal fuzzy set, whose fuzzy degree is 1 for

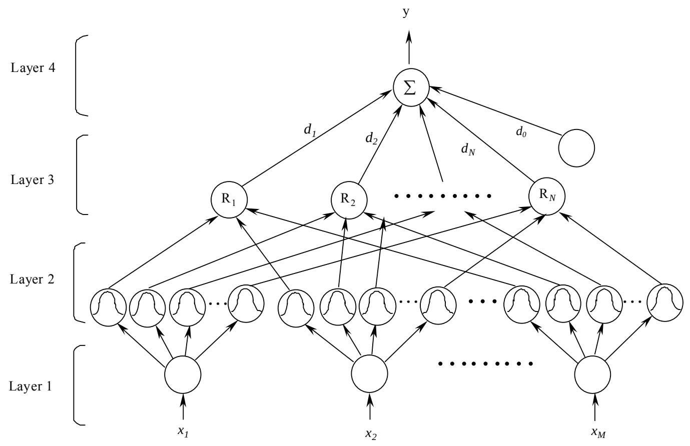


Figure 1: The structure of the four-layered fuzzy neural network.

any input value x_i, i =1, 2, ⋯, 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, ⋯, 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:

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

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

$$O^{(2)} = e^{-\frac{(a_i^{(2)} - m_{ij})^2}{\sigma_{ij}^2}} , \tag{5}$$

where m_{ij} and σ_{ij} are, respectively, the center (or mean) and the width (or variance) of the Gaussian membership function of the j-th term of the i-th 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

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

where $\mathrm{D}_j = diag[1/\sigma_{1j}, \cdots, 1/\sigma_{Mj}]$, $\mathbf{m}_j=[m_{1j}, m_{2j}, \ldots, m_{Mj}]^T$, $\mathbf{x}=[x_1, x_2, x_3, \cdots, x_M]^T$ is the FNN input vector of FNN. 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 and can be computed as:

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

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:

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

Construction of Fuzzy Rules

In order to construct the initial fuzzy rules of 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. In this study, the aligned clustering-based approach proposed in [16] is used to partition a set of data to establish the fuzzy preconditions in the rules.

In the classification problem, the training set is $\mathbf{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_v, y_v)\}$ with explanatory variable $\mathbf{x}_i$ and the corresponding binary class labels $y_i \in \{-1,+1\}$, for all i=1, ···, v, where v is the number of data. We use a clustering method which takes care of both the input and output values of a data set to satisfy the aforementioned conditions. 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 the 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. 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. In the function approximation problem, the Cartesian product-space of the input and output is applied to the clustering algorithm [17]. The training samples are partitioned into characteristic regions where the system behaviors are approximated. The input data set is formed by combining the input vector $\mathbf{x}=[x_1, x_2, x_3, \ldots, x_M]^T$ and the corresponding output value y_i.

Based on the clustering-based approach to construct initial fuzzy rules of FNN, first the input datasets are partitioned. For each incoming pattern **b**,

$$\mathbf{b}=\begin{cases}\mathbf{x} & \text{for classification dataset,}\\ [\mathbf{x}; y]^T & \text{for regression dataset,}\end{cases} \tag{9}$$

the strength a rule is fired can be interpreted as the degree the incoming pattern belongs to the corresponding cluster. 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. We can use the firing strength derived in (6) directly as this degree measure

$$F^j(\mathbf{b})=\prod_{i=1}^{M} a_i^{(3)} = e^{-[\mathbf{D}_j(\mathbf{b}-\mathbf{m}_j)]^T[\mathbf{D}_j(\mathbf{b}-\mathbf{m}_j)]} \in [0,1], \tag{10}$$

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

$$J = \arg \max_{1\le j\le c(t)} F^j(\mathbf{b}), \tag{11}$$

where $c(t)$ is the number of existing rules at time t. If $F^J \le F(t)$, then a new rule is generated, where $F(t)\in(0, 1)$ 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

$$\mathbf{m}_{[c(t)+1]}=\mathbf{b}, \tag{12}$$

$$\mathbf{D}_{[c(t)+1]}=\frac{-1}{\chi}\cdot diag\left[\frac{1}{\ln(F^J)} \;\cdots\; \frac{1}{\ln(F^J)}\right], \tag{13}$$

according to the first-nearest-neighbor heuristic [18], where $\chi \ge 0$ decides the overlap degree between two clusters. Similar methods are used in [19], [20] for the allocation of a new radial basis unit. However, in [19] the degree measure doesn't take the width $\mathbf{D}_j$ into consideration. In [20], the width of each unit is kept at a prespecified constant value, so the allocation result is, in fact, the same as that in [20]. 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 [20] 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 (12) and (13) 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 SVFNN, we use the formula of the similarity measure of two fuzzy sets with Gaussian membership functions derived previously in [19]. Suppose that 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 $\mu_{A\cup B}(x)=\max[u_A(x), u_B(x)]$, for every $x\in U$. The intersection of two fuzzy sets *A* and *B* is a fuzzy set $A\cap B$ such that $\mu_{A\cap B}(x)=\min[u_A(x), u_B(x)]$, for every $x\in U$. The size or cardinality of fuzzy set *A*, $M(A)$, equals the sum of the support values of *A*: $M(A)=\Sigma u_A(x)$, for $x\in U$. Since the area of the bell-shaped function, $\exp\{-(x-m)^2/\sigma^2\}$, is $\sigma\sqrt{\pi}$ 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 [21], we can compute $M|A\cap B|$ by

$$M\left|A\cap B\right|=\sum_{x\in U}(\min[u_A(x),u_B(x)])=\frac{1}{2}\frac{h^2\left[c_2-c_1+\sqrt{\pi}(\sigma_1+\sigma_2)\right]}{\sqrt{\pi}(\sigma_1+\sigma_2)}+ \frac{1}{2}\frac{h^2\left[c_2-c_1+\sqrt{\pi}(\sigma_1-\sigma_2)\right]}{\sqrt{\pi}(\sigma_2-\sigma_1)}+\frac{1}{2}\frac{h^2\left[c_2-c_1-\sqrt{\pi}(\sigma_1+\sigma_2)\right]}{\sqrt{\pi}(\sigma_1-\sigma_2)} \tag{14}$$

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

$$E(A,B)=\frac{M\left|A\cap B\right|}{M\left|A\cup B\right|}=\frac{M\left|A\cap B\right|}{\sigma_1\sqrt{\pi}+\sigma_2\sqrt{\pi}-M\left|A\cap B\right|}, \tag{15}$$

where we use the fact that $M(A)+M(B)=M(A\cap B)+M(A\cup B)$ [21]. 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

$$c_3=\frac{c_1+c_2}{2}, \tag{16}$$

$$\sigma_3=\frac{\sigma_1+\sigma_2}{2}. \tag{17}$$

The detailed learning algorithm is given in next section.

Learning Algorithm

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 layer 3 and layer 4 of SVFNN (see Fig. 1) are optimized by using the result of the support vector learning method with the fuzzy kernels for pattern classification and function approximation, respectively. In the third phase, unnecessary fuzzy rules are recognized and eliminated and the relevant fuzzy rules are determined.

Learning Phase 1 – Establishing initial fuzzy rules
The first phase establishes the initial fuzzy rules. 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 of SVFNN 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.

In the above 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 F_{in} determines the number of rules generated. For a higher value of 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 pre-specified values are given heuristically. In general, the threshold F_{in} = 0.35, prespecified constant σ_{init} = 0.5, the overlap degree χ = 2. In addition, after we determine the precondition part of fuzzy rule, we also need to properly assign the consequence part of fuzzy rule. For pattern classification, 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 $\mathbf{w}_{Con-1}$ for the consequence part of fuzzy rule. Another parameter in (7) that needs concern is the weight d_j associated with each $\alpha_j^{(4)}$. It is presented in **Learning Phase 2** to show how we can use the results to determine these weights.

```
IF b is the first incoming input pattern THEN do
PART 1. {Generate a new rule with center $\mathbf{m}_l=\mathbf{b}$ and
 width $\mathbf{D}_1=diag[1/\sigma_{init}, \cdots, 1/\sigma_{init}]$,
 IF b is the classification dataset
 { IF the output pattern y belongs to class 1
 (namely, $\mathbf{y}$=[1 0]),
 {$\mathbf{w}_{Con-1}$=[1 0] for indicating output node 1
 been excited, }
 ELSE { $\mathbf{w}_{Con-1}$=[0 1] for indicating output
 node 2 been excited.}}
 }
ELSE for each newly incoming input $\mathbf{b}$, do
PART 2. {Find $J$=arg(max $F^j(\mathbf{b})$, for $1 \le j \le c(t)$)
as defined in (10).
 IF ($\mathbf{w}_{Con-J} \neq \mathbf{y}$ for classification) or
 ($F^J \le F_{in}(t)$ for regression)
{ set $c(t+1)=c(t)+1$ and generate a new fuzzy
rule,
 with $\mathbf{m}_{c(t+1)}=\mathbf{b}$ and
$\mathbf{D}_{c(t+1)}=(-1/\chi)\text{diag}(1/\ln(F^J), \cdots, 1/\ln(F^J))$,
 where $\chi$ decides the overlap degree between
 two clusters. The $\mathbf{w}_{Con-c(t+1)}=\mathbf{y}$ for classification.
In addition, after decomposition, we have $m_{new-i}=b_i$,
$\sigma_{new-i}=-\chi\times\ln(F^J)$, $i=1, \cdots, M$. Do the following
fuzzy measure for each input variable $i$:
 {$Degree(i, t) \equiv \max_{1\le j\le k_i}(E[\mu(m_{new-i}, \sigma_{new-i}), \mu(m_{ij}, \sigma_{ij})])$,
where $E(\cdot)$ is defined in (15).
 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
$m_{new-i} = m_{closest} = \frac{m_{new-i} + m_{closest}}{2}$,
$\sigma_{new-i} = \sigma_{closest} = \frac{\sigma_{new-i} + \sigma_{closest}}{2}$.
 } }
 ELSE
{IF $F^J \le F_{in}(t)$ for classification
{generate a new fuzzy rule with $\mathbf{m}_{c(t+1)}=\mathbf{b}$,
$\mathbf{D}_{c(t+1)}=(-1/\chi)\text{diag}(1/\ln(F^J), \cdots, 1/\ln(F^J))$, and the
 respective consequent weight $\mathbf{w}_{Con-c(t+1)}=\mathbf{y}$.
In addition, we also need to do the fuzzy measure
for each input variable $i$.
} } }
```

Learning Phase 2 - Calculating the parameters of SVFNN
Through above method, the initial structure of SVFNN is established. If SVFNN is applied to pattern classification, we can then use the SVM method [22] to find the

optimal parameters of SVFNN that can solve classification problem. The dual quadratic optimization of SVM [23] is solved in order to obtain an optimal hyperplane for any linear or nonlinear space:

$$\text{maximize } L(\vec{\alpha}) = \sum_{i=1}^{v} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{v} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } 0 \le \alpha_i \le C,\ i=1, 2, \cdots, v, \text{ and } \sum_{i=1}^{v} y_i \alpha_i = 0, \tag{18}$$

where $K(\mathbf{x}_i,\mathbf{x}_j)$ is the fuzzy kernel that is defined as [24]

$$K(\hat{\mathbf{x}},\ \hat{\mathbf{z}}) = \begin{cases} \prod_{i=1}^{M} u_j(x_i)\cdot u_j(z_i), & \text{if } \hat{\mathbf{x}} \text{ and } \hat{\mathbf{z}} \text{ are both in the } j\text{-th cluster} \\ 0, & \text{otherwise,} \end{cases} \tag{19}$$

where $\hat{\mathbf{x}} = [x_1, x_2, x_3, \cdots, x_M] \in R^M$ and $\hat{\mathbf{z}} = [z_1, z_2, z_3, \cdots, z_M] \in R^M$ are any two training samples, and $u_j(x_i)$ is the membership function of the j-th cluster, and C is a user-specified positive parameter to control the tradeoff between complexity of SVM and the number of nonseparable points. This quadratic optimization problem can be solved and a solution $\vec{\alpha}_0 = (\alpha_1^0, \alpha_2^0, \cdots, \alpha_{nsv}^0)$ can be obtained, where α_i^0 are Lagrange coefficients, and nsv is the number of support vectors. The corresponding support vectors $\mathbf{sv} = [\mathbf{sx}_1, \mathbf{sx}_2, \cdots, \mathbf{sx}_i, \cdots, \mathbf{sx}_{nsv}]$ can be obtained, and the constant (threshold) d_0 in (7) is

$$d_0 = \frac{1}{2}\left[\left(w_0 \cdot x^*(1)\right) + \left(w_0 \cdot x^*(-1)\right)\right] \text{ with } w_0 = \sum_{i=1}^{nsv} \alpha_i y_i x_i, \tag{20}$$

where the support vector $x^*(1)$ belongs to the first class and support vector $x^*(-1)$ belongs to the second class.

If SVFNN is applied to function approximation, the optimal parameters of SVFNN are trained by using the ε -insensitivity loss function SVR [22] based on the fuzzy kernels [24]. The dual quadratic optimization of SVR [23] is solved in order to obtain an optimal hyperplane for any linear or nonlinear space:

$$\text{maximize } L(\alpha,\alpha^*) = -\varepsilon\sum_{i=1}^{v}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{v}(\alpha_i^* - \alpha_i) y_i$$

$$-\frac{1}{2}\sum_{i,j=1}^{v}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to constraints

$$\sum_{i=1}^{v}\alpha_i^* = \sum_{i=1}^{v}\alpha_i,\ 0 \le \alpha_i^* \le C,\ 0 \le \alpha_i \le C,\ \text{i}=1, 2, \cdots, v, \tag{21}$$

where $K(\mathbf{x}_i,\mathbf{x}_j)$ is the fuzzy kernel that is defined as [24], ε is a previously chosen nonnegative number ε for ε-insensitive loss function, and *C* is a user-specified positive parameter to control the tradeoff between complexity of SVR and the number of nonseparable points. This quadratic optimization problem can be solved and a solution $\vec{\alpha}=(\alpha_1,\ \alpha_2,\ \ldots\ldots,\ \alpha_{nsv})$ and $\vec{\alpha}^*=(\alpha_1^*,\ \alpha_2^*,\ \ldots\ldots,\ \alpha_{nsv}^*)$ can be obtained, where α_i and α_i^* are Lagrange coefficients. The corresponding support vectors **sv**=[**sx**$_1$, **sx**$_2$, ⋯, **sx**$_i$, ⋯, **sx**$_{nsv}$] can be obtained, and the constant (threshold) d_0 in (7) is

$$d_0=\frac{1}{v}\left(\sum_{i=1}^{v}(y_i-\mathbf{x}_i^T\mathbf{w}_0)\right) \text{ with } \mathbf{w}_0=\sum_{i=1}^{nsv}(\alpha_i^*-\alpha_i)\mathbf{x}_i, \tag{22}$$

Hence, the fuzzy rules of SVFNN are reconstructed by using the results of the SVM and SVR learning with fuzzy kernels, for pattern classification and function approximation, respectively. 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, ⋯, *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 $d_j=y_j(\alpha_j^*-\alpha_j)$. In this phase, the use of variable-width fuzzy kernels makes SVM and SVR 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, the number of fuzzy rules learning in Phases 1 and 2 is reduced by removing some irrelevant fuzzy rules and the consequent parameters of the remaining fuzzy rules is retuned under the condition that the accuracy of SVFNN for pattern classification or function approximation is kept almost the same. The 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

$$O^{(4)}=\sum_{j=1}^{N}d_j\times a_j^{(4)}+d_0=\sum_{j=1}^{N}d_j\prod_{i=1}^{M}e^{-\frac{(x_i-m_{ij})^2}{\sigma_{ij}^2}}+d_0\ , \tag{23}$$

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:

$$O^{\mathrm{Re}(4)}=\sum_{q=1}^{R_z}\beta_q\times a_q^{\mathrm{Re}(4)}+d_0=\sum_{q=1}^{R_z}\beta_q\prod_{i=1}^{M}e^{-\frac{(x_i-m_{iq}^{\mathrm{Re}})^2}{\sigma_{iq}^{\mathrm{Re}^2}}}+d_0$$

and

$$a_q^{\mathrm{Re}(4)}(\mathbf{x})=\prod_{i=1}^{M}e^{-\frac{(x_i-m_{iq}^{\mathrm{Re}})^2}{\sigma_{iq}^{\mathrm{Re}^2}}}\ , \tag{24}$$

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 and $\mathbf{m}_q^{\mathrm{Re}}=[m_{1q}^{\mathrm{Re}}, m_{2q}^{\mathrm{Re}}, \cdots, m_{Mq}^{\mathrm{Re}},]^T$. For choosing the more important R_z fuzzy rules from the old N fuzzy rules, the approximation in (24) can be achieved by computing a whole sequence of reduced set approximations [25]

$$O_r^{\mathrm{Re}(4)}=\sum_{q=1}^{r}\beta_q\times a_q^{\mathrm{Re}(4)}, \tag{25}$$

for r=1, 2, $\cdots$, 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 (24) can be obtained by the following iterative optimization rule [25]:

$$\mathbf{m}_{q+1}^{\mathrm{Re}}=\frac{\sum_{j=1}^{N} d_j\times a_j^{(4)}(\mathbf{m}_q^{\mathrm{Re}})\times\mathbf{m}_j}{\sum_{j=1}^{N} d_j\times a_j^{(4)}(\mathbf{m}_q^{\mathrm{Re}})}. \tag{26}$$

According to (26), 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, $\cdots$, 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, $\cdots$, R_z, are then computed to approximate $O^{(4)}=\Sigma(d_j\times a_j)$, for j=1, $\cdots$, N, by $O^{\mathrm{Re}(4)}=\Sigma(\beta_q\times a_q^{Re})$ [25], for q=1, $\cdots$, R_z, and can be obtained as

$$\beta=[\beta_1,\beta_2........,\beta_{R_z}]=\mathbf{K}_{R_z\times R_z}{}^{-1}\times\mathbf{K}_{R_z\times N}\times\Theta, \tag{27}$$

where

$$\mathbf{K}_{R_z\times R_z}=\begin{pmatrix} a_1^{\mathrm{Re}(4)}(\mathbf{m}_1^{\mathrm{Re}}) & a_1^{\mathrm{Re}(4)}(\mathbf{m}_2^{\mathrm{Re}}) & \cdots & a_1^{\mathrm{Re}(4)}(\mathbf{m}_{R_z}^{\mathrm{Re}}) \\ a_2^{\mathrm{Re}(4)}(\mathbf{m}_1^{\mathrm{Re}}) & a_2^{\mathrm{Re}(4)}(\mathbf{m}_2^{\mathrm{Re}}) & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{R_z-1}^{\mathrm{Re}(4)}(\mathbf{m}_{R_z}^{\mathrm{Re}}) \\ a_{R_z}^{\mathrm{Re}(4)}(\mathbf{m}_1^{\mathrm{Re}}) & \cdots & a_{R_z}^{\mathrm{Re}(4)}(\mathbf{m}_{R_z-1}^{\mathrm{Re}}) & a_{R_z}^{\mathrm{Re}(4)}(\mathbf{m}_{R_z}^{\mathrm{Re}}) \end{pmatrix} \tag{28}$$

and

$$\mathbf{K}_{R_z\times N}=\begin{pmatrix} a_1^{\mathrm{Re}(4)}(\mathbf{m}_1) & a_1^{\mathrm{Re}(4)}(\mathbf{m}_2) & \cdots & a_1^{\mathrm{Re}(4)}(\mathbf{m}_{R_x}) \\ a_2^{\mathrm{Re}(4)}(\mathbf{m}_1) & a_2^{\mathrm{Re}(4)}(\mathbf{m}_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{R_z-1}^{\mathrm{Re}(4)}(\mathbf{m}_N) \\ a_{R_z}^{\mathrm{Re}(4)}(\mathbf{m}_1) & \cdots & a_{R_z}^{\mathrm{Re}(4)}(\mathbf{m}_{N-1}) & a_{R_z}^{\mathrm{Re}(4)}(\mathbf{m}_N) \end{pmatrix} \tag{29}$$

and

$$\Theta=[d_1,d_2,\cdots,d_N]. \tag{30}$$

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

Experimental Results and Discussions

In order to evaluate the performance of the proposed SVFNNs, we apply SVFNNs to the Iris dataset from UCI repository [26] and the Vehicle dataset from Statlog collection [27] for classification, and the one- and two- variable functions for approximation, respectively.

Pattern Classification

The Iris dataset is originally a collection of 150 samples belonging to three classes. The Vehicle dataset consists of 846 samples belonging four classes. Because the Iris and Vehicle datasets do not contain testing data explicitly, we divide the whole data in Iris and Vehicle datasets into two halves as the training and testing datasets, respectively.

Tables 1 and 2 show the classification accuracy rates and the number of used fuzzy rules (i.e., support vectors) in the SVFNN applied to the Iris and Vehicle datasets, respectively. The criterion of determining the number of reduced fuzzy rules is the difference of 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. The generalized accuracy is estimated by using different cost parameters C=[2^{12}, 2^{11}, 2^{10}, …, 2^{-2}] in (18). We apply 2-fold cross-validation for 100 times on the whole training data in the Iris and Vehicle datasets and then average all the results. The cost parameter C that results in the best average cross-validation rate for SVM training is chosen to predict the test set. In Table 1, the proposed SVFNN is verified by using the Iris dataset, where the constant *n* in the symbol SVFNN-*n* means the number of the learned fuzzy rules. It 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 is 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. In Table 2, the proposed SVFNN is verified by using the Vehicle dataset and we have the similar experimental results as those in Table 1.

Table 1: 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 2: Experimental results of SVFNN classification on the Vehicle dataset.

SVFNN-*n* (SVFNN with *n* fuzzy rules)	Training process		Testing process	
	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 3: Classification error rate comparisons among FNNs, RBF-kernel-based SVM, RSVM and SVFNN classifiers, where NA means "not available".

Datasets	FNN [28, 29]		RBF-kernel-based SVM [30]		RSVM [31]		SVFNN	
	FUZZY RULES	Error rate	support vectors	Error rate	support vectors	Error rate	Fuzzy rules	Error rate
Iris	NA	4.3%	16	3.3%	14	4%	7	5.3%
Vehicle	NA	29.9%	343	13.4%	198	14%	125	14.5%

The performance comparisons among the existing fuzzy neural network classifiers [28], [29], the RBF-kernel-based SVM (without support vector reduction) [30], reduced support vector machine (RSVM) [31] and the proposed SVFNNs are made in Table 3.

These results indicate that the testing error rates of SVFNN are lower than FNN classifiers [28], [29], and the SVFNN uses less support vectors as compared to the regular SVM using fixed-width RBF kernels [30]. As compared to RSVM [31], the proposed SVFNN can not only achieve high classification accuracy, but also reduce the number of support vectors quit well. For FNN and SVM, the computational cost depends on the number of the rules (FNN) and the number of the support vectors (SVM), respectively. According to Table 3, it can be found that the computational cost of the proposed SVFNN is less than the regular SVM and RSVM in testing stage.

We also perform a receiver operating characteristic (ROC) analysis [32]-[34] to evaluate the assessing performance of the proposed SVFNNs for pattern classification as shown in Fig. 2. The ROC curve is a plot of the classification sensitivity (i.e. true positive rate) as the ordinate versus the specificity (i.e. false positive rate) as the abscissa. It is obtained by continuously varying the threshold associated with its decision function. In an ideal condition, if the area under the ROC curve is equal to 1, it shows the best performance. In Fig. 2, the area under the ROC curve is 0.92. This result shows that the SVFNNs can perform good classification performance.

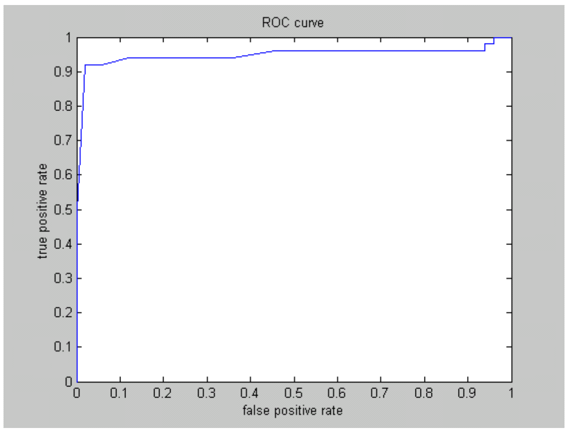


Figure 2: The ROC analysis of the SVFNN for the classification problem.

Function Approximation

The SVFNN is verified by the one- and two-variable functions for approximation problems. These two functions have been widely used in the literature [35]-[36]. The first function is a one-variable function defined as

$$f^{(1)}(x) = x^{2/3} \text{ with } x \in [-2, \ 2]. \tag{31}$$

The second function is a two-variable Gaussian function defined as

$$f^{(2)}(x, y) = \exp\{-2(x^2 + y^2)\}$$

with

$$x \in [-1, \ 1], \quad y \in [-1, \ 1]. \tag{32}$$

Plots of these two functions are shown in Figs. 3(a) and 4 (a).

There are two sets of training data for each function. One is noiseless and the other is noisy. In the first function, the noiseless training set has 50 points that are generated by randomly selecting, where $x \in$[-2, 2]. The testing set has 200 points that are randomly generated by the same function in the same range. In the second function, the 150 training examples are generated by randomly selecting, where $x \in$[-1, 12], $y \in$[-1, 1]. The testing set has 600 points that are randomly generated by the same function in the same range. The noisy training sets are generated by adding independent and identically distributed (i.i.d.) Gaussian noise, with zero mean and 0.25 standard deviation, to the original training sets. It is noted that the signal to noise ratio (SNR) is roughly equal to 4 (1/0.25=4).

The root-mean-square-error (RMSE) is used to quantify the performance of methods and it is defined as

$$\text{RMSE} = \sqrt{\sum_{i=1}^{v}(y_i - \hat{y}_i)/v}, \tag{33}$$

where y_i is the desired output, $\hat{y}_i$ is the system output, and v is the number of the used training or testing data. The ε -insensitivity parameter and cost parameter C in (21) are selected from the range of ε =[0.1, 0.01, 0.001, 0.0001] and C=[10^{-1}, 10^{0}, 10^{1}, …, 10^{5}], respectively. For the SVFNN training, we choose the ε -insensitivity parameter and cost parameter C that results in the best RMSE average to calculate the testing RMSE. Similarly, the parameters of SVR [37] for comparison are also selected by using the same method, too.

Tables 4 and 5 show the training and testing RMSEs and the number of used fuzzy rules (i.e., support vectors) in the SVFNN on the two functions (Eqs. (31) and (32)), respectively. The training and testing RMSEs can reach a nice level by selecting a proper parameter set for {ε, C }. 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 0.2%, meaning that some important support vectors has been removed, the we stop the rule reduction. In Table 4 (a), the SVFNN is verified by the one-variable function defined as (31), where the constant n in the symbol SVFNN-n means the number of the learned fuzzy rules. It uses nineteen fuzzy rules and achieves a root mean square error (RMSE) value of 0.0009 on the training data and an RMSE value of 0.0056 on the testing data. When the number of fuzzy rules is reduced to twelve, its testing error rate is increased to 0.0060. When the number of fuzzy rules is reduced to eleven, its testing error rate is increased to 0.0092. Continuously decreasing the number of fuzzy rules will keep the error rate increasing. In Table 4 (b), the independent and identically distributed (i.i.d.) Gaussian noise, with zero mean and 0.25 standard deviation is added to the function in Table 4(a). It uses twenty-five fuzzy rules and achieves a root mean square error (RMSE) value of 0.001 on the training data and an RMSE value of 0.078 on the testing data. When the number of fuzzy rules is reduced to fifteen, its testing error rate is increased to 0.081. When the number of fuzzy rules is reduced to fourteen, its testing error rate is increased to 0.139. In Table 5, the SVFNN is verified by the two-variable functions defined as (32) and we have the similar experimental results as those in Table 4. These experimental results show that the proposed SVFNN can properly reduce the number of required fuzzy rules and maintain the good generalization ability as shown in Figs. 3(b) and 4(b). The cost parameter C controls the trade-off between the training error and the VC dimension of the model. Since the decision boundaries for the classification problems are usually sharper than the approximation curves for the prediction problems, the cost parameter is higher for classification problems and lower for prediction problems as comparing Tables 1, 2, 4 and 5. This phenomenon can also be observed in many related studies [11], [30].

The performance comparisons among the adaptive-network-based fuzzy inference system (ANFIS) [38], the RBF-kernel-based SVR (without support vector reduction)

[37], and the proposed SVFNN are made in Tables 6 and 7. These results indicate that SVFNN maintains the function approximation accuracy and use less support vectors as compared to the regular SVR using fixed-width RBF kernels [37]. The computational cost of the proposed SVFNN is also less than the regular SVR in the testing stage. In addition, according to Table 7 the testing results of SVFNN trained by the noisy dataset are close to results trained by the dataset without noise. It demonstrates that the proposed SVFNN has better robustness compared to conventional neuro-fuzzy inference systems, although the SVFNN uses little more rules compare with the ANFIS.

Conclusions

This paper proposes novel support-vector-based fuzzy neural networks (SVFNNs), which combine the capability of statistical optimization and global generalization of support vector learning in high dimensional data spaces and the efficient human-like reasoning of FNN in handling uncertainty information. A novel adaptive fuzzy kernel function is also proposed to bring the advantages of FNNs to SVM/SVR directly and the use of the proposed fuzzy kernels provides the SVM/SVR with adaptive local representation power. The major advantages of the proposed SVFNNs are: (1) The proposed SVFNNs can automatically generate fuzzy rules and improve the accuracy of classification and approximation function. (2) They combine the optimal learning ability of SVM/SVR and the human-like reasoning of fuzzy systems. The pattern classification and function approximation ability of SVM and SVR can be improved by using the adaptive fuzzy kernels, respectively, and the operation speed can be increased by reduced fuzzy rules. (3) The ability of the structural risk minimization induction principle that forms the basis for the SVM/SVR to minimize the expected risk, gives better generalization ability to the FNN models. In the future work, we will try to develop a mechanism to automatically select the optimal initial values of the parameters used in the first phase training and the penalty parameter in the second phase training. We will also apply the proposed method to deal with huge data sets and other real problems.

Table 4: (a).Experimental results of SVFNN on the first function using training data without noise.

SVFNN-*n*	Training process		Testing process
(SVFNN with *n* fuzzy rules)	*C*	*RMSE*	*RMSE*
SVFNN-19	100	0.0009	0.0056
SVFNN-16	100	0.0009	0.0056
SVFNN-12	100	0.0009	0.0060
SVFNN-11	100	0.0015	0.0092
1. The first function is $f^{(1)}(x)=x^{2/3}$ with $x\in[-2, 2]$. 2. The number of training data is 50. 3. The number of testing data is 200.			

(b). Experimental results of SVFNN on the first function using training data with noise.

SVFNN-*n* (SVFNN with *n* fuzzy rules)	Training process		Testing process
	C	*RMSE*	*RMSE*
SVFNN-25	100	0.001	0.078
SVFNN-20	100	0.001	0.078
SVFNN-15	100	0.001	0.081
SVFNN-14	100	0.0057	0.139
1. The first function is $f^{(1)}(x)=x^{2/3}$ with $x\in[-2, 2]$. 2. The number of training data is 50. 3. The number of testing data is 200.			

Table 5: (a). Experimental results of SVFNN on the second function using training data without noise.

SVFNN-*n* (SVFNN with *n* fuzzy rules)	Training process		Testing process
	C	*RMSE*	*RMSE*
SVFNN-33	100	0.0018	0.0037
SVFNN-24	100	0.0018	0.0037
SVFNN-17	100	0.0018	0.0040
SVFNN-16	100	0.002	0.0089
1. The second function is $f^{(2)}(x, y)=\exp\{-2(x^2+y^2)\}$ with $x\in[-1, 1], y\in[-1, 1]$. 2. The number of training data is 150. 3. The number of testing data is 600.			

(b). Experimental results of SVFNN on the second function using training data with noise.

SVFNN-*n* (SVFNN with *n* fuzzy rules)	Training process		Testing process
	C	*RMSE*	*RMSE*
SVFNN-33	100	0.018	0.051
SVFNN-24	100	0.018	0.051
SVFNN-17	100	0.018	0.054
SVFNN-16	100	0.045	0.121
1. The second function is $f^{(2)}(x, y)=\exp\{-2(x^2+y^2)\}$ with $x\in[-1, 1], y\in[-1, 1]$. 2. The number of training data is 150. 3. The number of testing data is 600.			

Acknowledgment

This work was supported in part by the Ministry of Education, Taiwan, R.O.C., under

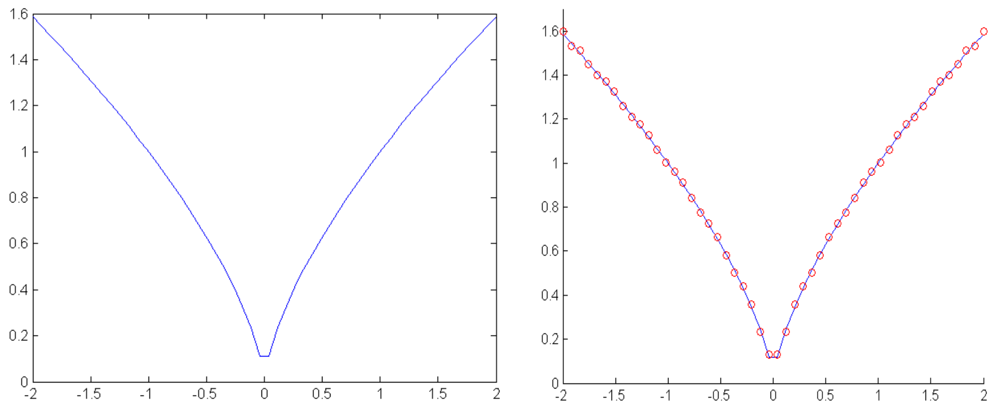
Grant EX-91-E-FA06-4-4 and the Ministry of Economic Affairs, Taiwan, R.O.C., under Grant 94-EC-17-A-02-S1-032.

Table 6: Comparisons RMSE using the training data without noise.

FUNCTION	ANFIS [38]		RBF-kernel-based SVR [37]		SVFNN	
	Number of fuzzy rules	RMSE	Number of support vectors	RMSE	Number of Fuzzy rules	RMSE
$f^{(1)}(x)$	11	0.0067	50	0.0054	12	0.006
$f^{(2)}(x,y)$	9	0.0039	122	0.0018	17	0.004

Table 7: Comparisons RMSE using the training data with noise.

FUNCTION	ANFIS [38]		RBF-kernel-based SVR [37]		SVFNN	
	Number of fuzzy rules	RMSE	Number of support vectors	RMSE	Number of Fuzzy rules	RMSE
$f^{(1)}(x)$	12	0.5	49	0.07	15	0.081
$f^{(2)}(x,y)$	9	0.305	139	0.04	17	0.054



(a) (b)

Figure 3: (a) The desired output of the function shown in (31). (b) The resulting approximation by SVFNN.

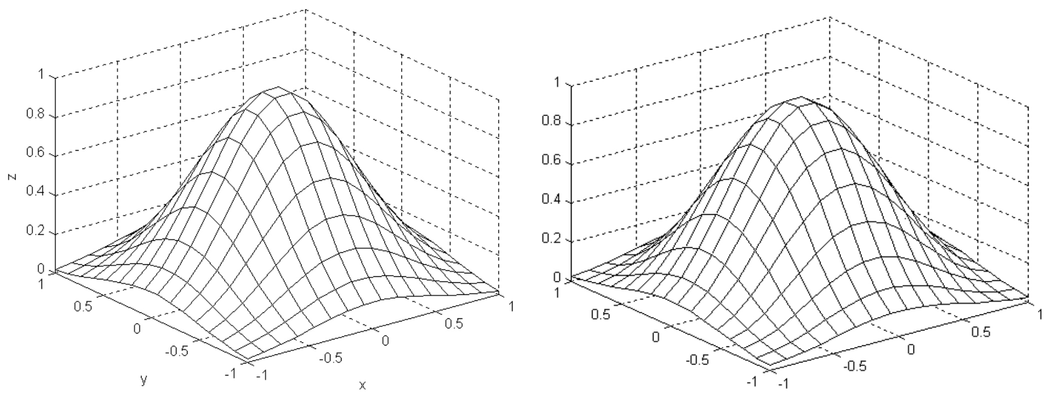


Figure 4: (a) The desired output of the function shown in (32). (b) The resulting approximation by SVFNN.

References

[1] W. Y. Wang, T. T. Lee, C. L. Liu, and C. H. Wang, "Function approximation using fuzzy neural networks with robust learning algorithm," IEEE Trans. Syst., Man, Cybern. Pt B, Vol. 27, pp. 740-747, Aug. 1997.

[2] C. C. Chuang, S. F. Su, and S. S. Chen, "Robust TSK fuzzy modeling for function approximation with outliers," IEEE Trans. Fuzzy Syst., Vol. 9, pp. 810-821, Dec. 2001.

[3] H. Pomares, I. Rojas, J. Ortega, J. Gonzalez, and A. Prieto, "Systematic approach to a self-generating fuzzy rule-table for function approximation," IEEE Trans. Syst., Man, Cybern. Pt B, Vol. 30, pp. 431-447, June 2000.

[4] S. Wu, M. J. Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," IEEE Trans. Fuzzy Syst., Vol. 9, pp. 578-594, Aug. 2001.

[5] B. Gabrys and A. Bargiela "General fuzzy min-max neural network for clustering and classification," IEEE Trans. Neural Networks, Vol. 11, pp. 769-783, May 2000.

[6] K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive fuzzy rule-based classification system," IEEE Trans. Fuzzy Syst., Vol. 4, pp. 238-250, Aug. 1996.

[7] M. Figueiredo and F. Gomide, "Design of fuzzy systems using neurofuzzy networks," IEEE Trans. Neural Networks, Vol. 10, pp. 815-827, July 1999.

[8] V. Vapnik, Statistical Learning Theory, New York: Wiley, 1998.

[9] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000.

[10] V. Vapnik, S. Golowich, and A. J. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in Neural Information Processing Systems. Cambridge, MA: MIT Press, Vol. 9, 1997.
[11] C. C. Chuang, S. F. Su, J. T. Jeng, and C. C. Hsiao, "Robust support vector regression network for function approximation with outliers," IEEE Trans. Neural Networks, Vol. 13, pp. 1322-1330, Nov. 2002.
[12] J. H. Chiang, and P. Y. Hao, "Support vector learning mechanism for fuzzy rule-based modeling: a new approach," IEEE Trans. Fuzzy Syst., Vol. 12, pp. 1-12, Feb. 2004.
[13] S. Sohn and C. H. Dagli, "Advantages of using fuzzy class memberships in self-organizing map and support vector machines," Proc. International Joint Conference on Neural Networks (IJCNN'01), Vol. 3, pp. 1886-1890, July 2001.
[14] Z. Sun and Y. Sun, "Fuzzy support vector machine for regression estimation," IEEE International Conference on Systems, Man, and Cybernetics (SMC'03), Vol. 4, pp. 3336-3341, Oct. 2003.
[15] C. F. Lin and S. D. Wang, "Fuzzy support vector machines," IEEE Trans. Neural Networks, Vol. 13, pp. 464-471, March 2002.
[16] 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, pp. 12-32, Feb. 1998.
[17] M. Setnes, R. Babuska, and H. B. Verbruggen, "Rule-based modeling: precision and transparency," IEEE Trans. Syst., Man, Cybern. Part C, Vol. 28, pp. 165-169, Feb. 1998.
[18] C. T. Lin and C. S. G. Lee "Neural-network-based fuzzy logic control and decision system," IEEE Trans. Comput., Vol. 40, pp. 1320-1336, Dec. 1991.
[19] J. Platt, "A resource allocating network for function interpolation," Neural Computat., Vol. 3, pp. 213-225, 1991.
[20] J. Nie and D. A. Linkens, " Learning control using fuzzified self-organizing radial basis function network," IEEE Trans. Fuzzy Syst., Vol. 40, pp. 280-287, Nov. 1993.
[21] 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, pp. 46-63, Feb. 1994.
[22] V. N. Vapnik, The Nature of Statistical Learning Theory, New York: Springer-Verlag, 1990.
[23] B. Schölkopf, C. J. C. Burges, and A. J. Smola, Advances in Kernel Methods—Support Vector Learning, Cambridge, MA: MIT Press, 1999.
[24] C. T. Lin, C. M. Yeh, S. F. Liang, J. F. Chung, and N. Kumar, "Support vector based fuzzy neural network for pattern classification," accept IEEE Trans. Fuzzy Syst.
[25] B. Scholkopf, S. Mika, C. Burges, etc "Input space versus feature space in kernel-based methods," IEEE Trans. Neural Networks, Vol. 10, pp.1000-1017, Sep. 1999.

[26] C. L. Blake and C. J. Merz, (1998) UCI repository of Machine Learning Databases, Univ. California, Dept. Inform. Comput. Sci., Irvine, CA. [Online]. Available at: http://www.ics.uci.edu/~mlearn/MLRepository.html.

[27] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, (1994) Machine Learning, Neural and Statistical Classification [Online]. Available at: ftp://ftp.stams.strath.ac.uk/pub/.

[28] 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. Pt B, Vol. 31, pp. 426-432, June 2001.

[29] M. R. Berthold and J. Diamond, "Constructive training of probabilistic neural networks," Neurocomputing, Vol. 19, pp. 167-183, 1998.

[30] C. W. Hsu and C. J. Lin, "A comparison of methods for multiclass support vector machines," IEEE Trans. Neural Networks, Vol. 13, pp. 415-525, March 2002.

[31] K. M. Lin and C. J. Lin, "A study on reduced support vector machines," IEEE Trans. Neural Networks, Vol. 14, pp.1449-1459, Nov. 2003.

[32] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic ROC curve," Radiology, Vol. 143, pp. 29-36, 1982.

[33] C. E. Meta, "Some practical issues of experimental design and data analysis in radiological ROC studies," Investigat. Radiol., Vol. 24, pp. 234-245, 1989.

[34] K. Woods and K. W. Bowyer, "Generating ROC curves for artificial neural networks," IEEE Trans. on Medical Imaging, Vol. 16, pp.329-337, June 1997.

[35] K. Liano, "Robust error measure for supervised neural network learning with outliers," IEEE Trans. Neural Networks, Vol. 7, pp.246-250, Jan. 1996.

[36] A. Suarez and J. F. Lutsko, "Globally optimal fuzzy decision trees for classification and regression," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 21, pp. 1297-1311, Dec. 1999.

[37] S. R. Gunn. (1999) Support vector regression-Matlab toolbox. Univ. Southampton, Southampton, U. K.. [Online]. Available at: http://kernel-machines.org.

[38] J. S. R. Jang, C. T. Sun, and E. Mizutani, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, Upper Saddle River, NJ, 1997.

Authors Biography

Dr. Chin-Teng (CT) Lin received B.S. degree from National Chiao-Tung University (NCTU), Taiwan in 1986 and Ph.D. degrees in electrical engineering from Purdue University, U.S.A., in 1992. He is currently the Chair Professor of Electrical and Control Engineering, Associate Dean of the College of Electrical Engineering and Computer Science, 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, and the Chairman of Electrical and Control Engineering Department of NCTU from 2000 to 2003. 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). Dr. Lin 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* (World Scientific). Dr. Lin has 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 an IEEE Fellow for his contributions to biologically inspired information systems. He serves on Board of Governors at IEEE Circuits and Systems (CAS) Society in 2005 and IEEE Systems, Man, Cybernetics (SMC) Society in 2003-2005. He is the Distinguished Lecturer of IEEE CAS Society from 2003 to 2005. Dr. Lin is the International Liaison of 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. Dr. Lin has received the Outstanding Research Award granted by National Science Council, 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. Dr. Lin was also elected to be one of the 38th Ten Outstanding Rising Stars in Taiwan (2000). Dr. Lin currently serves as Associate editors of IEEE Transactions on Circuits and Systems, Part I & Part II, IEEE Transactions on Systems, Man, Cybernetics, IEEE Transactions on Fuzzy Systems, and International Journal of Speech Technology. Dr. Lin 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 National Chiao-Tung University, Hsinchu, Taiwan, R.O.C. He is also 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, pattern recognition.

Jen-Feng Chung was born in Miaoli, Taiwan, in 1971. He 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 (NCTU), Hsinchu, Taiwan. His current research interests are cellular neural networks (CNNs), VLSI signal processing, audio and image signal processing, and CPU/DSP architecture design.

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. He received 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. 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 biomedical engineering, biomedical signal/image processing, machine learning, fuzzy neural networks (FNN), the development of brain-computer interface (BCI), and multimedia signal processing.

Her-Chang Pu was born in Taipei, Taiwan, in 1972. He received the B.S. and M.S. degree in automatic control engineering from Feng-Chia University, Taichung, Taiwan, in 1998. He received the Ph.D. degree in Electrical and Control Engineering from National Chiao-Tung University (NCTU) in 2003, Hsinchu, Taiwan. Currently, he is a Research Assistant Professor in Electrical and Control Engineering, NCTU. His current research interests are in the areas of artificial neural networks, fuzzy systems, pattern recognition, machine vision and intelligent transportation system (ITS).