Fuzzy Neural Network Classification Design using Support Vector Machine

Chin-Teng Lin, Chang-Moun Yeh, and Chun-Fei Hsu
Department of Electrical and Control Engineering
National Chiao-Tung University, Hsinchu, Taiwan

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 classification (SVFNNC) is proposed. The SVFNNC 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 two learning phases is developed to construct the SVFNNC and train its parameters. In the first phase, the fuzzy rules and membership functions are automatically determined by clustering principle. In the second phase, the parameters of FNN are calculated by the SVM with the proposed adaptive fuzzy kernel function. To investigate the effectiveness of the proposed SVFNNC, it is applied to the Iris, Vehicle and Dna datasets. Experimental results show that the proposed SVFNNC can achieve good classification performance with drastically reduced number of fuzzy kernel functions.

Index Terms – Fuzzy neural network, fuzzy kernel function, support vector machine.

I. INTRODUCTION

Much research has been done on fuzzy neural networks (FNNs), which combine the capability of fuzzy reasoning in handling uncertain information [1] and the capability of neural networks in learning from processes [2]. They have been successfully applied to classification, identification, control, pattern recognition, and image processing, etc [3]. In particular, many learning algorithms of fuzzy (neural) classifiers have been presented and applied in pattern classification and decision-making systems [4]. 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 pre-researches used 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 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) [7] 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. However, the optimal solutions of SVM rely heavily on the property of selected [8] kernel functions, whose parameters are always fixed and are chosen solely based on heuristics or trial-and-error nowadays.

In this paper, we develop a support-vector-based fuzzy neural network classification (SVFNNC), which combines the superior classification power of SVM in high dimensional data spaces and the high efficient human-like reasoning power of FNN in handling uncertainty information. The SVFNNC 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. Finally, the experimental results on four datasets (Iris, Vehicle, Dna) from the UCI Repository and Statlog collection show that the proposed SVFNNC 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.

II. CONSTRUCTION OF INITIAL FUZZY NEURAL NETWORK

A four-layered fuzzy neural network (FNN) is shown in Fig 1, which is comprised of the input, membership function, rule, and output layers. This four-layered network realizes the following form of fuzzy rules:

\[
\text{Rule } R_j : \quad \text{If } x_i = A_{ij} \text{ and } \ldots \text{ and } x_m = A_{jm}, \quad \text{Then } y = d_j, \quad j = 1, 2, \ldots, n, \quad (1)
\]

where \( A_{ij} \) are the fuzzy sets of the input variables \( x_i \) and \( d_j \) are the consequent parameter of \( y \). For the ease of implementation, we use \( x_i \) and \( d_j \) as numerical values instead of fuzzy membership functions.
of analysis, a fuzzy rule 0 is added as:

Rule 0 : If \( x_1 \) is \( A_{01} \) and ... and \( x_m \) is \( A_{0m} \), Then \( y \) is \( d_0 \),

(2)

where \( A_{ki} = 1 \) for \( k = 1, 2, \cdots, m \) are the fuzzy sets of the input variables, and \( d_0 \) is the consequent parameter of \( y \) in the fuzzy rule 0. Thus, the overall output as the summation of all input signals:

\[
O^{(4)} = \sum_{j=1}^{n} d_j x_j^{(4)} + d_0,
\]

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 FNN mapping can be rewritten in the following input-output form:

\[
O^{(4)} = \sum_{j=1}^{n} d_j x_j^{(4)} + d_0 = \sum_{j=1}^{n} d_j \prod_{i} u_i^{(x)} + d_0.
\]

Moreover, 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. 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. In this work, we use an aligned clustering-based approach proposed in [9]. This method produces a partition result as shown in Fig. 2.

A rule corresponds to a cluster in the input space, with \( m_i \) and \( D_i \) representing the center and variance of that cluster. For each incoming pattern \( x \), the strength a rule is fired can be interpreted as the degree the incoming pattern belongs to the corresponding cluster. For computational efficiency, we can use the firing strength derived in (3) directly as this degree measure

\[
F'(x) = \prod_{i} d_i^{(3)} = e^{\sum_{i} d_i \prod_{j} u_j^{(x)}} \in [0, 1]
\]

where \( x = [x_1, x_2, x_3, \cdots, x_m]^T \) is the FNN input vector, and \( F'(x) \in [0, 1] \). In the above equation the term

\[
[D_i (x - m_i)]^T[D_i (x - m_i)]
\]

is the distance between \( x \) and the center of cluster \( i \). Using this measure, we can obtain the following criterion for the generation of a new fuzzy rule. Let \( x \) be the newly incoming pattern. Find

\[
J = \arg\max_{1 \leq j \leq c(t)} F'(x).
\]

where \( c(t) \) is the number of existing rules at time \( t \). If \( F' \leq \overline{F} (t) \), 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

\[
m_{i(t)+1} = x_i,
\]

(7)

\[
d_{i(t)+1} = \frac{-1}{\beta} \text{diag} \left[ \frac{1}{\text{diag}(F_i)} \right]
\]

(8)

according to the first-nearest-neighbor heuristic, where \( \beta \geq 0 \) decides the overlap degree between two clusters.

III. FUZZY KERNEL

The proposed fuzzy kernel \( K(\hat{x}, \hat{z}) \) in this paper is defined as

\[
k(\hat{x}, \hat{z}) = \prod_{i=1}^{k} u_i(\hat{x}_i, \hat{z}_i), \quad \text{if } \hat{x}_i \text{ and } \hat{z}_i \text{ are both in the } j \text{-th cluster}
\]

(11)

\[
0, \quad \text{otherwise}
\]

where \( \hat{x} = [x_1, x_2, x_3, \cdots, x_n] \in R^n \) and \( \hat{z} = [z_1, z_2, z_3, \cdots, z_n] \in R^n \) are any two training samples, and \( u_i(\hat{x}_i) \) is the membership function of the \( j \)-th cluster. Assume the training samples \( \mathbf{s} = (x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m) \in \mathbf{X} \) are partitioned into \( m \) classes of clusters through fuzzy clustering in Section II and \( n \) is the total number of training samples. We can perform the following permutation of training samples

\[
\begin{align*}
\text{cluster } 1 &= \{ (x_{i_1}^1, y_{i_1}^1), \cdots, (x_{i_k}^1, y_{i_k}^1) \} \\
\text{cluster } 2 &= \{ (x_{i_1}^2, y_{i_1}^2), \cdots, (x_{i_k}^2, y_{i_k}^2) \} \\
&\vdots \\
\text{cluster } m &= \{ (x_{i_1}^m, y_{i_1}^m), \cdots, (x_{i_k}^m, y_{i_k}^m) \}
\end{align*}
\]

(12)

where \( k \) is the number of points belonging to the \( i \)-th cluster, so that we have \( \sum_{i=1}^{m} k_i = n \). Then the fuzzy kernel can be calculated by using the training set in (12), and the obtained kernel matrix \( K \) can be rewritten as the following form

\[
K = \begin{bmatrix}
K_{11} & K_{12} & \cdots & K_{1m} \\
K_{21} & K_{22} & \cdots & K_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
K_{m1} & K_{m2} & \cdots & K_{mm}
\end{bmatrix} \in R^{m \times m}
\]

(13)

where \( K_{ij} \) is defined as

\[
K_{ij} = \begin{bmatrix}
K(\hat{x}_{i1}, \hat{z}_{j1}) & K(\hat{x}_{i1}, \hat{z}_{j2}) & \cdots & K(\hat{x}_{i1}, \hat{z}_{jm}) \\
K(\hat{x}_{i2}, \hat{z}_{j1}) & K(\hat{x}_{i2}, \hat{z}_{j2}) & \cdots & K(\hat{x}_{i2}, \hat{z}_{jm}) \\
\vdots & \vdots & \ddots & \vdots \\
K(\hat{x}_{im}, \hat{z}_{j1}) & K(\hat{x}_{im}, \hat{z}_{j2}) & \cdots & K(\hat{x}_{im}, \hat{z}_{jm})
\end{bmatrix} \in R^{k \times k}
\]

(14)

In order that the fuzzy kernel function defined by (11) is suitable for application in SVM, we must prove that the fuzzy kernel function is symmetric and positive-definite Gram Matrices.

Theorem 1 : For the fuzzy kernel defined by (11), if the membership functions \( u(x) : R \rightarrow [0, 1], \quad i = 1, 2, \cdots, n \), are positive-definite functions, then the fuzzy kernel is a Mercer kernel.

IV. LEARNING ALGORITHM OF SVFNNC

The learning algorithm of the SVFNNC consists of two phases. The first phase establishes initial fuzzy rules. In the second phase, the optimal parameters of SVFNNC are calculated by SVM technique. The details are given below:
Learning Phase 1 – Establishing initial fuzzy rules

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 \( x \) is the first incoming input pattern THEN do

\[ \text{PART 1.} \]

Generate a new rule with center \( m_i = x \) and width

\[ D_i = \text{diag} \left( \frac{1}{\sigma_{i_1}}, \ldots, \frac{1}{\sigma_{i_n}} \right), \]

where \( \sigma_{\text{init}} \) is a prespecified constant. After decomposition, we have one-dimensional membership functions, with \( m_{i_j} = x_j \) and \( \sigma_{i_j} = \sigma_{\text{init}}, \quad i = 1, \ldots, n. \) In addition, after we determine the preconceived 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 bigger excitation value, we know this input-output pattern belongs to class 1. Hence, initially, we should assign the proper weight \( w_{\text{Con-1}} \) for the consequence part of fuzzy rule as follows:

\[ \text{IF the output pattern } y \text{ belongs to class 1 (namely, } y = [1 \ 0], \text{ then } w_{\text{Con-1}} = [1 \ 0] \text{ for indicating output node 1 been excited; ELSE } w_{\text{Con-1}} = [0 \ 1] \text{ for indicating output node 2 been excited.} \]

ELSE for each newly incoming input \( X \), do

\[ \text{PART 2.} \]

Find \( J = \arg \max_{i \in \mathcal{I}(t)} F^{\beta}(x) \), as defined in (5). We should check if the newly incoming output pattern \( y \) is different from the maximal excitation rule:

\[ \text{IF } w_{\text{Con-1}} \neq y, \text{ set } c(t + 1) = c(t) + 1 \text{ and generate a new fuzzy rule, with } m_{i(t+1)} = x, \]

\[ D_{i(t+1)} = \frac{1}{\beta} \text{diag} \left( \frac{1}{\ln(F^{\beta})}, \ldots, \frac{1}{\ln(F^{\beta})} \right), \text{ and } w_{\text{Con-1(t+1)}} = y, \]

where \( \beta \) decides the overlap degree between two clusters. In addition, after decomposition, we have \( m_{nsv-1} = x_j \), \( \sigma_{nsv-1} = -\beta \times \ln(F^{\beta}), \text{ i.e. } i = 1, \ldots, n. \)

Do the following fuzzy measure for each input variable \( i \):

\[ \text{Degree}(i, t) = \max_{k_j \leq F^{\beta}(i)} E \left[ \mu(m_{nsv-1}, \sigma_{nsv-1}), \mu(m_j, \sigma_j) \right] \]

where \( k_j \) is the number of partitions of the \( j \)th input variable, and \( E(\cdot) \) is defined in (10). IF \( \text{Degree}(i, t) \leq \rho(t) \) THEN adopt this new membership function, and set \( k = k_j + 1. \) ELSE set the projected membership function as the closest one.)} \)

ELSE

\[ \text{Continue to check if } F^{\beta} \geq F_{\text{in}}(t). \text{ If answer is YES, we do nothing. Otherwise, we also generate a new fuzzy rule with } m_{i(t+1)} = x, \]

\[ D_{i(t+1)} = \frac{1}{\beta} \text{diag} \left( \frac{1}{\ln(F^{\beta})}, \ldots, \frac{1}{\ln(F^{\beta})} \right), \text{ and the respective consequent weight } w_{\text{Con-1(t+1)}} = y. \]

In addition, we also need to do the fuzzy measure for each input variable \( i \).} \]

Learning Phase 2 - Calculating the parameters of SVFNNC

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

maximize \( L[\bar{c}] = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \)

subject to \( 0 \leq \alpha_i \leq C, i = 1, 2, \ldots, l \), and \( \sum_{i=1}^{n} y_i \alpha_i = 0 \).

where \( K(x_i, x_j) \) is the fuzzy kernel in (11), \( x_i \)'s are training data, 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 \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \) can be obtained, where \( \alpha_0 \) are Lagrange coefficients, and \( nsv \) is the number of support vectors. The corresponding support vectors \( sv = [s_{x_1}, s_{x_2}, \ldots, s_{x_n}] \) can be obtained, and the constant (threshold) \( d_0 \) in (3) is

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

where \( nsv \) is the number of fuzzy rules (support vectors); the support vector \( x_i^* \) (1) belongs to the first class and support vector \( x_i^*-1 \) belongs to the second class. Hence, the parameters in the fuzzy rules of SVFNNC can be calculated by \( d_j = y_j \alpha_j \) in (3) and \( m_j = s_{x_j}, j=1, 2, \ldots, nsv \), where \( d_j \) is the coefficient corresponding to \( m_j = s_{x_j}. \)

V. EXPERIMENTAL RESULTS

The classification performance of the proposed SVFNNC is evaluated on four well-known benchmark datasets. From the UCI Repository, we choose one dataset: Iris dataset. From Statlog collection we choose two datasets: Vehicle, and Dna datasets. These three datasets will be used to verify the effectiveness of the proposed SVFNNC. We scale all training data to be in \([-1, 1]\) and also accordingly adjust testing data to be in \([-1, 1]\). It is noted that in the original Iris and Vehicle datasets, testing data are not available, so we repeat the 2-fold cross-validation 100 times using different partitions of the datasets and report the best cross-validation rate here.

These experimental results show that the proposed SVFNNC is good at maintaining the good generalization ability. Moreover, performance
comparisons among the existing fuzzy neural network classifiers [6], the RBF-kernel-based SVM [8], and the proposed SVFNNC are made in Table I. This table shows that the SVFNNC produces lower testing error rates as compared to FNN classifiers [10], [11], and uses less support vectors as compared to the normal SVM using fixed-width RBF kernels. In summary, the proposed SVFNNC exhibits better generalization ability on the testing data and use much smaller number of fuzzy rules.

VI. CONCLUSIONS

This paper proposed a support-vector-based fuzzy neural network classifier (SVFNNC), which combines the superior classification power of support vector machine (SVM) in high dimensional data spaces and the efficient human-like reasoning of fuzzy neural network (FNN) in handling uncertainty information. The SVFNNC 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 into the SVM directly. The major advantages of the proposed SVFNNC are as follows:

1. The proposed SVFNNC 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 fuzzy kernels.
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


TABLE I. Classification error rate comparisons among FNN classifier, SVM classifier, and SVFNNC, where NA means “not available”.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of rules</td>
<td>Error rate</td>
<td>Number of support vectors</td>
</tr>
<tr>
<td>Iris</td>
<td>NA</td>
<td>4.3%</td>
<td>16</td>
</tr>
<tr>
<td>Vehicle</td>
<td>NA</td>
<td>29.9%</td>
<td>343</td>
</tr>
<tr>
<td>Dna</td>
<td>NA</td>
<td>16.6%</td>
<td>1152</td>
</tr>
</tbody>
</table>

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

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