Support vector fuzzy adaptive network in regression analysis

Judong Shen\textsuperscript{a}, Yu-Ru Syau\textsuperscript{b}, E.S. Lee\textsuperscript{a,}\textsuperscript{*}

\textsuperscript{a}Department of Industrial and Manufacturing Systems Engineering, Kansas State University, Manhattan, KS 66506, United States
\textsuperscript{b}Department of Information Management, National Formosa University, Yunlin 632, Taiwan

Received 7 February 2007; accepted 16 March 2007

Abstract

Neural-fuzzy systems have been proved to be very useful and have been applied to modeling many humanistic problems. But these systems also have problems such as those of generalization, dimensionality, and convergence. Support vector machines, which are based on statistical learning theory and kernel transformation, are powerful modeling tools. However, they do not have the ability to represent and to aggregate vague and ill-defined information. In this paper, these two systems are combined. The resulting support vector fuzzy adaptive network (SVFAN) overcomes some of the difficulties of the neural-fuzzy system. To illustrate the proposed approach, a simple nonlinear function is estimated by first generating the training and testing data needed. The results show that the proposed network is a useful modeling tool.

\copyright\ 2007 Elsevier Ltd. All rights reserved.

Keywords: Support vector machines; Fuzzy adaptive network; Neural network; Fuzzy logic; Statistical learning theory

1. Introduction

Fuzzy adaptive networks or neural-fuzzy systems [1,2], which are formed by the combination of neural network and fuzzy logic, are useful approaches for the modeling of systems whose behavior is strongly influenced by human judgments, perceptions, or emotions. However, the fuzzy adaptive network (FAN) has several problems such as those of generalization, dimensionality, and convergence. The support vector machine (SVM) [3–5], which is based on statistical learning theory and the use of kernel transformation, overcomes some of the problems. Thus, to combine the SVM and FAN, thus taking the advantages of both approaches, appears to be desirable.

Various researchers have investigated the use of the SVM in fuzzy-neural systems. Jeng et al. [6–8] did some basic research on applying support vector regression in determining the initial structure of fuzzy neural networks. Chan et al. [9–12] proposed methods to select the structure of neural networks and neural-fuzzy systems based on support vector regression and applied them to the modeling of nonlinear dynamic systems. The \(\epsilon\)-insensitive learning algorithm from the SVR was integrated with the traditional fuzzy modeling by Leski [13,14] to build the connection between the fuzzy modeling and the statistical learning theories. Kim et al. [15,16] tried to construct new fuzzy

\* Partially supported by the National Science Council of the Republic of China under contract NSC 93-2213-E-150-032.

\* Corresponding author. Tel.: +1 785 532 5606; fax: +1 785 532 3738.

E-mail addresses: jdshen@ksu.edu (J. Shen), yrsyau@nfu.edu.tw (Y.-R. Syau), eslee@ksu.edu (E.S. Lee).

In this paper, the SVM learning algorithms are applied to the various FAN networks and a support vector fuzzy adaptive network (SVFAN) is formulated for regression. Furthermore, the proposed SVFAN retains most of the advantages of the original SVM and FAN systems.

The FAN network is briefly summarized in Section 2. Sections 3 and 4 discuss some earlier works in the areas of fuzzy systems and support vector networks. The algorithm for the SVFAN is presented in Section 5. Section 6 illustrates the numerical approach by actually modeling a nonlinear equation. Finally, in Section 7, some discussions are presented.

2. Fuzzy adaptive network

The FAN is briefly summarized in the following. For a detailed discussion, the reader can consult the literature [1,2].

A fuzzy inference system basically consists of three conceptual components: a rule base, which contains a set of fuzzy if-then rules; a database, which defines the membership functions used in the fuzzy rules; and a reasoning mechanism, which performs the inference procedure based on the given rules to derive the desired output.

Various fuzzy inference systems can be constructed depending on the fuzzy if-then rules and the aggregation procedure. The fuzzy adaptive network (FAN) [1] is a particular inference system that uses the Takagi and Sugeno (TS) fuzzy inference models [19]. The network provides a comprehensive visualization and adaptability system by retaining both the representation ability of fuzzy systems and the learning ability of neural networks.

The FAN is a five-layered feed forward network (see Fig. 1) in which each node performs a particular node function on the incoming signals, which is characterized by a set of parameters. In order to reflect different adaptive capabilities, the nodes are represented by circles and squares. Circle nodes represent fixed nodes without parameters, while square nodes are adaptive nodes with parameters to be adjusted.

Nodes in layer 1 are adaptive nodes. The output of node \(k \) is defined as:

\[
 f_{1,k} = \mu_{F_{j,h_j}}(x_j), \quad \text{for } h_j = 1, \ldots, n_j, \text{ and } j = 1, \ldots, p 
\]  

where, \( x_j \) is the input to this node; \( F_{j,h_j} \) is the \( h \)th fuzzy set associated with the input \( x_j \); \( \mu_{F_{j,h_j}} \) is the membership function defined for \( F_{j,h_j} \), and \( n_j \) is the number of fuzzy sets for each particular \( x_j \). The membership function adopted
here is the Gaussian function, parameterized by the parameter set \((v_{j,h_j}, \sigma_{j,h_j})\), with the form:

\[
\mu_{F_{j,h_j}}(x) = \exp \left[ - \left( \frac{x_j - v_{j,h_j}}{\sigma_{j,h_j}} \right)^2 \right].
\] (2)

Nodes in layer 2 perform the fuzzy aggregation for the premise part of fuzzy if-then rules using the fuzzy AND operation and output the product, \(w^l\). The notation \(A_l\) represents a fuzzy AND operator. The number of nodes is equal to the number of combinations in choosing one node from each subgroup in layer 1, which is equal to the number of fuzzy if-then rules. The output from this layer is:

\[
f_{2,l} = w^l = \prod_{j=1}^{p} a_{F_{j,h_j}}(x_j), \quad \text{for all } h_j, \quad l = 1, \ldots, m.
\] (3)

Nodes in layer 3 are fixed nodes, which normalize the outputs of layer 2.

Nodes in the 4th layer are adaptive nodes. Each node, denoted as \(Y^l_f\), performs the following function for the consequence part of the fuzzy if-then rule.

\[
f_{4,l} = w^l Y^l = f^l_0 + c^l_1 x_1 + \cdots + f^l_p x_p, \quad l = 1, \ldots, m,
\] (4)

where consequence parameters \(c^l_j\) are symmetric triangular fuzzy numbers.

The single node in layer 5 is a fixed node, which performs the function of overall aggregation of all the fuzzy if-then rules from layer 4.

2.1. Performance measure

The performance of the FAN is usually measured by the difference between the desired output and actual output, called error measure. The error measure is defined as the least squares difference between the desired output \(Y_i\) and the estimated output from the network \(\hat{Y}_i\), where both \(Y_i\) and \(\hat{Y}_i\) are assumed for computational simplicity to be symmetric triangular fuzzy numbers. Therefore, \(Y_i = (y_i, e_i)\) and \(\hat{Y}_i = (\hat{y}_i, \hat{e}_i)\) are the modes of fuzzy numbers \(Y_i\) and \(\hat{Y}_i\) respectively, and correspondingly, \(e_i\) and \(\hat{e}_i\) are their spreads. The consequence parameters \(c^l_j\) are assumed as the symmetric triangular fuzzy numbers: \(c^l_j = (a^l_j, b^l_j)\), \(j = 0, \ldots, p, l = 1, \ldots, m\). The overall least squares error is:

\[
E = \frac{1}{N} \left\{ \sum_{i=1}^{N} \left[ (y_i - \hat{y}_i)^2 + (e_i - \hat{e}_i)^2 \right] \right\}
\] (5)

where \(N\) is the number of observations of the training data.

2.2. Learning algorithms

The FAN uses two sets of adaptive parameters: the premise parameters and the consequence parameters. Consequently, the learning process includes the learning of both sets of parameters.

The premise parameters set, \([\{v_{1,1}, \sigma_{1,1}\}, \{v_{1,2}, \sigma_{1,2}\}, \ldots, \{v_{1,n_1}, \sigma_{1,n_1}\}], \ldots, \{v_{p,1}, \sigma_{p,1}\}, \{v_{p,2}, \sigma_{p,2}\}, \ldots, \{v_{p,n_p}, \sigma_{p,n_p}\}]\), represents the fuzzy partitions used in the fuzzy rules. The learning of these parameters is achieved by the use of back propagation. The error obtained in layer 5 is back propagated to layer 1. A gradient decent method is applied to update the premise parameters based on the error function given as follows:

\[
E = \frac{1}{N} \sum_{i=1}^{N} e^2 = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.
\] (6)

The back propagation approach is essentially a decent gradient method. The equations can be obtained by differentiation.

The consequence parameters set, \([c^0_0, c^1_1, \ldots, c^0_p, c^1_1, \ldots, c^0_p, c^1_1, \ldots, c^0_p]\), represents the coefficients of the linear functions in the fuzzy if-then rules. Since the resulting equations are similar to the fuzzy
regression equations obtained by Tanaka [20]. Tanaka’s approach, namely learning, is obtained by solving a linear programming. Since this linear programming problem was formulated in fuzzy regression, a user’s confidence variable, \( \alpha \in (0, 1) \) or the \( \alpha \)-cut, appears in the set of linear programming equations.

Thus, the FAN learning algorithm can be summarized as follows:

**Learning Algorithm of the FAN**

begin
  epoch = 0;
  set \( \alpha \) value;
  initialize subjectively premise parameter set \( \{v_k, \sigma_k\}_{k=1,\ldots,n} \);
  while termination condition not satisfied do
    begin
      identify the consequence parameter set \( \{c_j\} \) by solving LP problem;
      evaluate the error measure;
      calculate back propagated errors;
      update premise parameter set \( \{v_k, \sigma_k\} \) with the back propagated errors;
      epoch = epoch + 1;
    end
  end.

3. Support vector machines and neuro-fuzzy systems

In order to combine the SVM with the neuro-fuzzy systems, it is useful to examine the similarities and differences between these systems. Here, neuro-fuzzy systems are represented by fuzzy sets, neural networks, and the FAN, which is a combination of fuzzy and neural network. This comparison, which is well known in the literature, is summarized in Table 1. From this table, it can be seen that FANs and SVMs are almost complementary to each other. Thus, a combination of SVMs with FANs appears to be a desirable approach.

Another system, namely the radial basis function network, has been shown, under some minor restrictions, to be equivalent to fuzzy inference systems [21]. Shen [22] has shown that some types of fuzzy system are equivalent to SVMs.

4. Support vector networks

Various investigators [23–25] have proposed network representations for support vector machines. The function-mapping equation for support vector classification can be represented as:

\[
f(x) = \text{sgn} \left( \sum_{i=1}^{#SV} \alpha_i^* y_i K(x_i, x) + b^* \right)\]  

and for regression as:

\[
f(x) = \sum_{i=1}^{#SV} (\alpha_i - \alpha_i^*) K(x_i, x) + b^*\]  

where \( x \) is the input vector, \( \alpha \) represents the Lagrange multiplier, \( b \) is the bias and \( K(\ , \) \) represents the kernel function. Any symbol with the superscript * denotes optimal value. The summation is over the number of support vectors and the symbol \#SV represents the number of support vectors.

The corresponding support vector network (SVN) is illustrated in Fig. 2 [23–25]. Superficially, this SVN consists of six layers. The first layer is the input vector. The second layer inputs the support vectors, which are obtained by solving the quadratic programming problem. The third layer maps the inputs from layer 1 and layer 2 to a high-dimensional feature space via the function \( \phi \), where dot products are computed. The mapped vectors are not really computed in this
Table 1
Comparison of different systems

<table>
<thead>
<tr>
<th>Property of system</th>
<th>Fuzzy system</th>
<th>NN</th>
<th>FAN</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge representation</td>
<td>✗</td>
<td>○</td>
<td>✗</td>
<td>○</td>
</tr>
<tr>
<td>Imprecision tolerance</td>
<td>✗</td>
<td>✗</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Uncertainty tolerance</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Interpretation capability</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Learning capability</td>
<td>○</td>
<td>■</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Adaptability</td>
<td>○</td>
<td>✔</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Suitable for high-dimension data</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Suffering from “curse of dimensionality”</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Sparsity property of solutions</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Optimality solution property</td>
<td>○ (Local)</td>
<td>○ (Local)</td>
<td>● (Global)</td>
<td></td>
</tr>
<tr>
<td>Amount of training data needed</td>
<td>No</td>
<td>○ (Large)</td>
<td>○ (Large)</td>
<td>● (Small)</td>
</tr>
</tbody>
</table>

The symbols used above are denoted as follows: ○ — bad; ● — good.

layer. The fourth layer computes the dot products of the combinations of the input vector and any support vector. The fifth layer computes the weights of the kernel functions, which are obtained from solving a quadratic programming problem with \( w_i = y_i\alpha_i \) for classification and \( w_i = \alpha_i^+ - \alpha_i^- \) for regression. The sixth layer linearly combines the weighted kernels as the output of the classification or regression results. Since the layer 3 (represented by dashed squares) does not physically exist in the SVM algorithms, the SVN only has five layers. The dot products of kernels are directly computed from the input vector and the support vectors using the kernel function.

The support vector network can be simplified as a three-layer structure. The first layer is the input layer, which is made up of the input vector and all the support vectors. The hidden layer is the kernel layer, which applies a nonlinear transformation from the input space to the feature space, where the inner products are computed. The number of the support vectors, which are identified by the SVM [26], determines the size of the hidden layer. Usually the number of support vectors is very small compared to the training patterns; therefore, the size of the hidden layer is moderate. The last layer is the output layer, which computes the weighted sum of all the kernels and their corresponding hyperplane classifier for classification or their corresponding regressor for regression.

5. Support vector fuzzy adaptive network

Based on the above discussions and following the earlier works on SVN, various SVFANs can be formulated. In this paper, only a SVFAN for regression will be established. SVFANs for other purposes were also established [22].
Given $I$ training data pairs $(\{x_1, y_1\}, \ldots, \{x_I, y_I\}) \subset \mathbb{R}^n \times \mathbb{R}$ with unknown joint distribution $P(x, y)$, also given a first-order TS fuzzy model \cite{19} with the following $m$ fuzzy rules

$$R_i: \text{IF } x_i \text{ is } A_{i1} \text{ and } x_2 \text{ is } A_{i2} \text{ and } \ldots \text{ and } x_n \text{ is } A_{in},$$

THEN $y = b_0^i + b_1^i x_1 + b_2^i x_2 + \cdots + b_n^i x_n$;

\begin{equation}
(9)
\end{equation}

where $i = 1, \ldots, m$. The goal of the general SVFAN for regression problem is to construct a good neuro-fuzzy system or FAN that has a good structure, good knowledge interpretability, high prediction accuracy, and thus has good generalization capability with a small number of fuzzy rules or hidden units.

Using the sum-product inference mechanism with Gaussian kernels or Gaussian membership functions, the fuzzy antecedent firing strength of the SVFAN network can be represented by:

$$\mu_{A^i(x)} = \prod_{j=1}^n \exp \left[ -\frac{(x_j - c_j^i)^2}{2(\sigma_j^i)^2} \right] = \exp \left[ -\frac{1}{2} \sum_{j=1}^n \left( \frac{x_j - c_j^i}{\sigma_j^i} \right)^2 \right]$$

\begin{equation}
(10)
\end{equation}

where $c_j^i$ and $\sigma_j^i$, $i = 1, \ldots, m$; $j = 1, \ldots, n$ are the centers and dispersions of the membership functions or the parameters of the Gaussian functions for the $i$th rule and the $j$th input variable. The initial values of the parameters are usually set in such a way that the membership functions along each axis satisfy $\varepsilon$-completeness, normality, and convexity.

Given an input $x$, the output of this fuzzy model can be represented as a weighted averaging aggregation of outputs of the individual rules as follows:

$$y = f(x, b') = \sum_{i=1}^m (b_0^i + b_1^i x_1 + b_2^i x_2 + \cdots + b_n^i x_n) \prod_{j=1}^n \exp \left[ -\frac{(x_j - c_j^i)^2}{2(\sigma_j^i)^2} \right]$$

\begin{equation}
(11)
\end{equation}

Let $p^{(i)}(x)$ represent the normalized firing strength of the $i$th rule for input $x$,

$$p^{(i)}(x) = \frac{\prod_{j=1}^n \exp \left[ -\frac{(x_j - c_j^i)^2}{2(\sigma_j^i)^2} \right]}{\sum_{i=1}^m \prod_{j=1}^n \exp \left[ -\frac{(x_j - c_j^i)^2}{2(\sigma_j^i)^2} \right]}$$

then \eqref{eq:11} can be rewritten as:

$$y = f(x, b') = \sum_{i=1}^m p^{(i)}(x)(b_0^i + b_1^i x_1 + b_2^i x_2 + \cdots + b_n^i x_n) = \sum_{i=1}^m p^{(i)}(x) b_i^T x'$$

\begin{equation}
(12)
\end{equation}

where $b_i^T = [b_0^i \ b_1^i \cdots b_n^i] \in \mathbb{R}^{n+1}$ is the vector of consequent parameters of the $i$th rule, in which $b_0^i$ denotes the bias of the $i$th local model; $x' = [x]$ is the augmented input vector.

Suppose $\varphi^i(x) = p^{(i)}(x) x$, $\varphi^i(x') = p^{(i)}(x) x'$, $\varphi(x) = [\varphi^1(x) \varphi^2(x) \cdots \varphi^m(x)] \in \mathbb{R}^{mn}$, and $\varphi(x') = [\varphi^1(x') \varphi^2(x') \cdots \varphi^m(x')] \in \mathbb{R}^{m(m+1)}$, then \eqref{eq:10} can be further rewritten as:

$$y = f(x, b') = \sum_{i=1}^m \varphi^i(x') b_i^T = \varphi(x') B'$$

\begin{equation}
(13)
\end{equation}

where $B' = [b_1^T b_2^T \cdots b_m^T]^T \in \mathbb{R}^{m(m+1)}$ denotes the consequent parameter vector of all $m$ fuzzy rules (each rule has $n + 1$ weights). Eq. \eqref{eq:13} is a linear regression model.
Combining the $\varepsilon$-insensitive loss function and the SVR learning algorithm with the linear regression model, Eq. (13), we obtain the following cost function, which should be minimized:

$$
\min_{B \in \mathbb{R}^m, b \in \mathbb{R}} f(B, b) = \frac{1}{2} B^T B + C \sum_{i=1}^l (\xi_i + \xi_i^*),
$$

(14)

where $B = [b_1^T, \ldots, b_m^T]^T \in \mathbb{R}^{mn}, b = p^1(x)b_0^1 + p^2(x)b_0^2 + \cdots + p^m(x)b_0^m$ denotes the overall bias because of $\hat{y}_i = \varphi(x_i)B' = \varphi(x_i)B + b$. $C$ is a positive and predefined constant which influences the trade-off between the complexity of the regression model and the approximation error, $\xi_i, \xi_i^* \geq 0$ are slack variables since, according to the $\varepsilon$-insensitive loss function, some points will fall outside the $\varepsilon$-insensitive zone, which means some points may not satisfy the inequalities $y_i - \varphi(x_i)B - b \leq \varepsilon$ and $\varphi(x_i)B + b - y_i \leq \varepsilon$. $\varepsilon$ is also a user-defined precision parameter that determines the size of an $\varepsilon$ tube. Thus, the constraints for the model using $\varepsilon$-insensitive learning are:

$$
y_i - \varphi(x_i)B - b \leq \varepsilon + \xi_i
$$

(15)

$$
\varphi(x_i)B + b - y_i \leq \varepsilon + \xi_i^*.
$$

Combining Eqs. (14) and (15), we obtain the following quadratic programming problem:

$$
\min_{B \in \mathbb{R}^m, b \in \mathbb{R}} f(B, b) = \frac{1}{2} B^T B + C \sum_{i=1}^l (\xi_i + \xi_i^*)
$$

s.t. \begin{align*}
    y_i - \varphi(x_i)B - b & \leq \varepsilon + \xi_i \\
    \varphi(x_i)B + b - y_i & \leq \varepsilon + \xi_i^* \\
    \xi_i, \xi_i^* & \geq 0, \quad i = 1, \ldots, l.
\end{align*}

(16)

By solving the above quadratic programming problem, the parameters for the linear regression model, the overall bias $b$, and the support vectors $n_{sv}$, can be obtained. Each support vector should form one fuzzy rule. Thus, we have the equivalences:

- The number of fuzzy rules, $m =$ the number of support vectors, $n_{sv}$
- $c_i^j = x_{svij}$; where $i = 1, 2, \ldots$, and $j = 1, 2, \ldots, n$. 

5.1. Architecture of the SVFAN

The architecture of the SVFAN is shown in Fig. 3, which consists of nodes interconnected by directional links. SVFAN is a five-layer network. The network follows closely the existing networks proposed in the literature (see Figs. 1 and 2). Each layer is associated with a particular step in the fuzzy inference process. Each node performs a particular node function on the incoming signal, which is characterized by a set of parameters. Some of the nodes are adaptive nodes, whose parameters can be tuned by learning procedure. In order to reflect different adaptive capabilities, two different nodes, namely, circles and squares, are used in the figure. Circle nodes represent fixed nodes without parameters, while square nodes are adaptive nodes with parameters to be adjusted. The functions of each layer are described below. The output from node $k$ in layer $r$ is denoted as $f_r^k$.

Layer 1: (Input layer) — Nodes in layer 1 are fixed nodes. Nodes in this layer represent the inputs of the systems, which include all the $n_{sv}$ support vectors and the currently being tested input vector. The output of node $i$ is the $i$th support vector and is expressed as:

$$
f_i^1 = x_{svi}, \quad i = 1, \ldots, n_{sv}.
$$

Each node of the $n_{sv}$ support vectors corresponds to a single fuzzy rule.

Virtual Layer: A virtual layer is added to show the mapping function $\varphi(x)$, which maps the data from input space to a higher dimensional feature space. This layer is only for clarity and is not needed in actual computation.

Layer 2: (Kernel layer) — Nodes in layer 2 are adaptive nodes. Each node represents a kernel function, which calculates the dot product of the currently being tested vector and the support vectors: $\phi(x) \cdot \phi(x_{svi}), i = 1, \ldots, n_{sv}$. 


The kernels or dot products are equal to the multidimensional Gaussian membership functions in the first-order TS fuzzy model. The output of node $k$ is defined as:

$$f^i_2 = K(x_{svi}, x) = \exp \left[ -\frac{(x - x_{svi})^2}{2\sigma_i^2} \right] \quad i = 1, \ldots, n_{sv}.$$  

**Layer 3**: (normalization layer) — Nodes in layer 3 are fixed nodes, which normalize the outputs of layer 2. The output from this layer is:

$$f^i_3 = \bar{w}_i = \frac{w_i}{\sum_{t=1}^{n_{sv}} w_t} = \frac{K(x, x_{svi})}{\sum_{t=1}^{n_{sv}} K(x, x_{svt})}, \quad i = 1, \ldots, n_{sv}.$$  

**Layer 4**: (consequence layer) — Nodes in this layer are adaptive nodes. Each node, denoted as $y_i$, performs the following function for the consequence part of the fuzzy IF-THEN rule in the first-order TS fuzzy model.

$$f^i_4 = y_i = b^i_0 + b^i_1 x_1 + b^i_2 x_2 + \cdots + b^n_{x_n}, \quad i = 1, \ldots, n_{sv},$$

where consequence parameters $b^i_j$ are real values.

**Layer 5**: (defuzzification layer) — The single node in this layer is a fixed node, which performs the function of overall aggregation of all the fuzzy IF-THEN rules from layer 4. The output signal of this node is:

$$f^i_5 = \hat{y} = \sum_{l=1}^{n_{sv}} \bar{w}_l y_l.$$  

5.2. Performance measure

The performance of the SVFAN is measured by the difference between the desired output and predicted output. For each individual input–output pair $(x_i, y_i)$, let $\hat{y}_i$ be the estimated network output for the input vector $x_i$, then the overall mean square error is

$$E = \frac{1}{l} \sum_{i=1}^{l} (y_i - \hat{y}_i)^2$$  

where $\hat{y}_i = \sum_{k=1}^{n_{sv}} \frac{K(x_i, x_{svk})}{\sum_{t=1}^{n_{sv}} K(x_i, x_{svt})} \left( b^i_0 + b^T x_i \right), \quad i = 1, \ldots, l$, and $l$ is the number of the training data.
5.3. Learning of the SVFAN

Both the network structure and the parameters are learned. The number of fuzzy rules depends on the number of support vectors, which must be obtained during learning. Thus, the learning algorithms include both the network structure identification algorithm, which comes from the SVR algorithm, and the parameter estimation algorithms, which include the least-squares algorithm for adjusting consequence parameters and the gradient descent algorithm for adjusting the Gaussian premise parameter $\sigma$. In addition, we also have the parameters $C$ and $\epsilon$. The overall learning algorithm for the SVFAN is summarized in Fig. 4.

**Learning Algorithm of the SVFAN**

input: Initial parameters of Gaussian kernels, error $\epsilon$ and regularization parameter $C$; training data; learning rate.
begin
epoch = 0;
set value of learning rate $\eta$;
initialize subjectively error $\epsilon$ and regularization parameter $C$ and premise parameter set $\sigma$;
while termination condition not satisfied do
begin
use SVR learning algorithm to determine the number of fuzzy rules and the SVs are output as the centers of the Gaussian MFs or kernels;
adjust the consequence parameter set $\{b^j_i\}$ by solving the least-square algorithm;
evaluate the error measure;
calculate back propagated errors;
update premise parameter set $\sigma$ with the back propagation method;
set the updated premise parameter set $\sigma$ as the new kernel width parameter $\sigma$;
epoch = epoch + 1;
end
end.

6. Numerical example

To illustrate the approach, the following artificial function, which was adopted from Cheng and Lee [1], will be estimated by using the proposed SVFAN regression.

$$f(x_1, x_2) = 24.234r^2(0.75 - r^2) + 5.$$ (18)

The function $f(x_1, x_2)$ is plotted in Fig. 5. Based on this function, one hundred input–output pairs $(x_i, y_i)$, $x_i = (x_{1i}, x_{2i})^T$ and outputs $y_i$ are generated in the following way.

The input $(x_{1i}, x_{2i})$, $i = 1, 2, \ldots, 100$, are generated randomly with uniform distribution on $[0, 10]^2$. The output $y_i$ is generated by

$$y_i = f(x_{1i}, x_{2i}), \quad \text{for } i = 1, 2, \ldots, 100.$$

Among these one hundred generated input–output pairs, the first 70 input–output pairs were used as the training data and the remaining 30 input–output pairs as the testing data.

6.1. Parameter selection

The selections of the initial values of the parameters play an important role in the performance of the SVFAN network. The parameters needing to be considered are the upper bound $C$, the error band $\epsilon$, and the kernel parameters $\sigma$. Parameter $C$ determines the trade-off between the model complexity and the degree to which deviations larger than $\epsilon$ are tolerated in optimization formulation. Parameter $\epsilon$ controls the width of the $\epsilon$-insensitive zone. It also decides the
Fig. 4. Three learning methods of the SVFAN: one structure learning (adaptive SVR), and two parameters learning (LS and GD).

Fig. 5. Plot of the function $f(x_1, x_2)$.

number of support vectors used to construct the number of fuzzy rules or the complexity of the network. The larger the value $\varepsilon$, the fewer support vectors are selected. Improper selection of these parameters may result in overfitting or underfitting problems and thus result in an improper model, which with the information available is either too simple or too complex. Choosing these parameters is difficult and time consuming. There exists little guidance in the literature. Therefore, in this study, we varied the parameters to select the optimal values that give the best estimation performance.

6.2. Performance criterion

In order to compare the predicted performance of the SVFAN with the results for FAN due to Jiao [27], the following three indices are used:

- Number of iterations needed to obtain convergence.
- Mean squared error ($MSE$).
- Average percentage error ($APE$).

The first two indices are already defined previously. The last one, $APE$, is defined as:

$$APE = \frac{1}{l} \sum_{i=1}^{l} \left( \frac{|y_i - \hat{y}_i|}{|y_i|} \right) \cdot 100\%,$$

(19)
Table 2
Performance of the SVFAN

<table>
<thead>
<tr>
<th>Learning algorithm</th>
<th>SVFAN</th>
<th>SVFAN</th>
<th>SVFAN</th>
<th>SVFAN</th>
</tr>
</thead>
<tbody>
<tr>
<td># Training samples</td>
<td>70</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td># Testing samples</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Parameters (C, ε);</td>
<td>(100, 1);</td>
<td>(1000, 0.8);</td>
<td>(1000, 0.6);</td>
<td>(1, 1);</td>
</tr>
<tr>
<td>RBF, σ (initial)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td># Fuzzy rules (m)</td>
<td>22</td>
<td>25</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>MSE</td>
<td>0.0017</td>
<td>0.0007</td>
<td>0.0000605</td>
<td>0.0144</td>
</tr>
<tr>
<td>APE for training (%)</td>
<td>0.0164</td>
<td>0.00657</td>
<td>0.000313</td>
<td>0.00007</td>
</tr>
<tr>
<td>APE for testing</td>
<td>0.0513</td>
<td>0.20</td>
<td>2.06</td>
<td>3.59</td>
</tr>
</tbody>
</table>

Table 3
Relationship between the number of fuzzy rules and parameters (RBF Kernel: σ = 1)

<table>
<thead>
<tr>
<th>C</th>
<th>ε</th>
<th>#Fuzzy rules (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>57</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>38</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td>0.5</td>
<td>36</td>
</tr>
<tr>
<td>10</td>
<td>0.6</td>
<td>29</td>
</tr>
<tr>
<td>10</td>
<td>0.8</td>
<td>25</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>100</td>
<td>0.1</td>
<td>53</td>
</tr>
<tr>
<td>100</td>
<td>0.5</td>
<td>36</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>1000</td>
<td>0.1</td>
<td>53</td>
</tr>
<tr>
<td>1000</td>
<td>0.2</td>
<td>42</td>
</tr>
<tr>
<td>1000</td>
<td>0.3</td>
<td>41</td>
</tr>
<tr>
<td>1000</td>
<td>0.4</td>
<td>37</td>
</tr>
<tr>
<td>1000</td>
<td>0.5</td>
<td>36</td>
</tr>
<tr>
<td>1000</td>
<td>0.6</td>
<td>29</td>
</tr>
<tr>
<td>1000</td>
<td>0.8</td>
<td>25</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
<td>22</td>
</tr>
</tbody>
</table>

where l is the number of data. The smaller the values of MSE and APE, the better the generalization performance of the model is or the higher the predicted accuracy is.

6.3. Experimental results

The SVFAN was learned with a MATLAB-based computer program using the obtained dataset. The results obtained by the SVFAN were compared with the results obtained using FAN II and FAN III [27]. Using different parameters for the SVFAN (see Table 2), a different number of fuzzy rules and different generalization performances based on MSE and APE was obtained. Table 2 used four different sets of parameters. For comparison purposes, the estimated results were also plotted against the actual data by using both two and three-dimensional figures. The plotted figures cannot separate the estimated from the desired values.

The objectives for both the SVFAN and the FAN are to find the optimal number of fuzzy rules and to obtain a balance between the prediction accuracy and the model complexity. The number of fuzzy rules used can represent the model complexity of the fuzzy systems. For the FAN approaches, there exists no effort to optimize the number of fuzzy rules. The number of fuzzy rules for FAN I and FAN II is determined based on the granule size and the number of variables. In FAN III, this number is obtained using a clustering algorithm, which may be far from the optimal one. The advantage of the SVFAN is that it is easy to control the model complexity in the SVFAN because the number of fuzzy rules is the same as the number of support vectors, which is determined by the quadratic programming algorithm.
Table 4
Comparison of performance results

<table>
<thead>
<tr>
<th>Learning algorithm</th>
<th>SVFAN</th>
<th>FAN II</th>
<th>FAN III</th>
</tr>
</thead>
<tbody>
<tr>
<td># Fuzzy rules (m)</td>
<td>25</td>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>Iteration to converge</td>
<td>95</td>
<td>511</td>
<td>139</td>
</tr>
<tr>
<td>$MSE$</td>
<td>0.0007</td>
<td>0.010</td>
<td>0.004</td>
</tr>
<tr>
<td>$APE$ for training (%)</td>
<td>0.00657</td>
<td>0.647</td>
<td>0.574</td>
</tr>
<tr>
<td>$APE$ for testing (%)</td>
<td>0.20</td>
<td>1.411</td>
<td>5.577</td>
</tr>
</tbody>
</table>

Fig. 6. Convergences behavior of SVFAN (a), FAN II (b) and FAN III (c).

Table 3 shows the influence of the parameters used on the number of fuzzy rules. Several conclusions can be obtained from Table 3. For example, it appears that the wider the error band $\epsilon$ is, the smaller the number of fuzzy rules is.

The performances of the SVFAN are compared with the FAN approaches in Table 4 by considering four different criteria. From the table, it is obvious that the SVFAN results in better prediction accuracy and a more compact model over FANs.

The convergence behaviors during training for the SVFAN, FAN II, and FAN III with the same training data are plotted in Fig. 6. Compared with FAN II and FAN III, the SVFAN converges much faster than the FAN. It may be due to the fact that the SVFAN always yield a better initial network in terms of number of rules and parameters than the FAN.
7. Discussions

In this paper, the SVM and FAN are combined by considering two important aspects, the use of support vectors to reduce the structure or the complexity of the network and hence the model, and the use of the $\epsilon$-insensitive and the regularization parameter $C$ to balance between model complexity and the available data. The combination is accomplished owing to the fact that all the semi-positive definite fuzzy systems based on membership functions, which can be used as Mercer kernels, are functional equivalents to support vector machines [22]. The two systems are combined in such a way that the resulting SVFAN system possesses some advantages, such as fuzzy linguistic representation ability, learning ability, and generalization ability of the original approaches.

Although only a very simple example was used to illustrate the approach, the results show the advantages of the proposed system. The main problem in using the proposed system is the determination of the numerical values a priori of the parameters such as error, the band $\epsilon$, the upper bond $C$, and the kernel parameter $\sigma$. Further researches are needed to devise better approaches to select more appropriate parameter values.

References

