Extraction of Fuzzy Rules by Using Support Vector Machines

Shuwei Chen, Jie Wang, Dongshu Wang
School of Electrical Engineering
Zhengzhou University
Zhengzhou 450001, Henan, China
{swchen, wj, wangdongshu}@zzu.edu.cn

Abstract

This paper proposes an architecture to extract fuzzy rules based on support vector machines (SVMs). Firstly, support vectors are obtained from the training data set to generate fuzzy IF-THEN rules with membership functions described in terms of kernel functions via support vector machine learning procedure. Then, a combined fuzzy rule base is created based on both the generated rules and linguistic rules of human experts. Thus, it has the inherent advantages that the rule base is optimized automatically during the SVM learning procedure, and, takes both “subjective” experts’ prior knowledge and “objective” training data into account. An example is given to show the effectiveness of the proposed method.

1. Introduction

The last 40 years have seen the great successful applications of fuzzy set theory to a variety of areas including control and system identification, signal and image processing, decision and prediction[8, 13]. In recent years, the learning (modeling) of fuzzy systems has become an active area, for the case that it is the foundation of successful application of fuzzy systems. The key problems for building a fuzzy system are acquirement of inference rules and membership functions[5].

To find fuzzy inference rules, there are usually two kinds of information available: linguistic information obtained from human experts, and, numerical information obtained from sensor measurements. The first kind of information has been used extensively for the modeling of fuzzy systems by translating experts’ prior knowledge into fuzzy rules. However, its objectiveness has often been questioned, because it greatly relies on expert knowledge, which is subjective and varies from individual to individual. The second kind of information has been used recently for finding fuzzy rules[1, 9, 10, 11, 12, 22] by using neural networks, evolutionary algorithms and many other data mining techniques. But the theoretical foundation of these methods are still weak now, especially the generalization ability of the models built via these methods are unknown.

Support Vector Machine (SVM)[2, 7, 12, 20] proposed by Vapnik is a new machine learning method based on the Statistical Learning Theory. Statistical Learning Theory [18, 19] is mainly developed to resolve the overfitting problem experienced by most learning machines. SVMs have been successfully applied to pattern recognition and function estimation problems [2, 7, 15, 16, 17] because of its high generalization ability. In this method the original input space is mapped into a higher dimensional input space called feature space via a nonlinear mapping φ, and an optimal separating hyperplane is constructed in the feature space to maximize the generalization ability. The nonlinear mapping φ is usually unknown and complicated. Fortunately, the computation related to φ could be induced to the computation of a kernel function in the feature space. It can be seen as an approximate implementation of what Vapnik has defined as structure risk minimization, an inductive principle that aims at minimizing an upper bound on the generalization error of a model.

Recently, SVMs have been used in finding fuzzy rules from numerical data. For example, Kim et al. [14] presented a support vector fuzzy inference system for modeling T-S fuzzy model, and used gradient descent method to reduce the number of fuzzy rules. Chen and Wang [4] revealed that additive fuzzy systems and kernel machines are fundamentally related, and proposed kernel-based positive definite fuzzy classifier (PDFC). Chiang and Hao [6] presented a support vector learning architecture to extract support vectors for generating fuzzy rules from training data set, and described the fuzzy system in terms of kernel functions. Castro et al. [3] showed the relationship between SVMs and T-S-K fuzzy systems, and gave methods to extract rules from SVMs. Yu et al. [23] proposed a approach to construct fuzzy classifiers based on SVMs by investigating the connection between fuzzy classifiers and support
vector classifiers, and the link between fuzzy rules and kernel functions.

The above SVM-based learning mechanisms provide some fundamental ideas for extracting fuzzy rules from numerical data. But the other kind of information, linguistic information from experts, is ignored. We think that each of the numerical and linguistic information alone is usually incomplete. So, we develop a general approach that uses both kinds of information, simultaneously and cooperatively, to model fuzzy systems. The key ideas of our new approach are to generate fuzzy rules from numerical data by using SVMs, and, collect these fuzzy rules and the linguistic fuzzy rules into a common fuzzy rule base.

After a brief description of the theory of support vector machines in Section 2, Section 3 introduces a three step procedure for extracting fuzzy rules from numerical data and experts’ prior knowledge via support vector machines. In Section 4, the effectiveness of our method is verified via examples. Conclusions are given in Section 5.

2. Support Vector Machine

Let \textit{l}-dimensional training data \(x_i\) \((i=1, 2, \cdots, n)\) belong to class I or class II and the associated label be \(y_i = 1\) for class I and \(-1\) for class II. If these data are linearly separable in the feature space, the decision function can be determined as:

\[
f(x) = w^T \phi(x) + b, \tag{1}
\]

where \(\phi(x)\) is a mapping function that maps \(x\) into the feature space, \(w\) is an \(l\)-dimensional vector and \(b\) is a scalar. To separate data linearly, the decision function satisfies the following conditions:

\[
y_i(w^T \phi(x_i) + b) \geq 1, \quad i = 1, 2, \cdots, n. \tag{2}
\]

If the problem is linearly separable in the feature space, there are an infinite number of decision functions that satisfy (2). Among them the hyperplane is required to have the largest margin between two classes. Here the margin is the minimum distance from the separating hyperplane to the input data and this is given by \(f(x)/\|w\|\). And the separating hyperplane with the maximum margin is called the optimal separating hyperplane.

Then in order to obtain the optimal separating hyperplane with the maximum margin, we must find \(w\) with the minimum \(\|w\|\). This leads to solving the following optimization problem:

\[
\min \frac{1}{2} w^T w \quad \text{s. t.} \quad y_i(w^T \phi(x_i) + b) \geq 1, \quad \forall i. \tag{3}
\]

When training data are not linearly separable in the feature space, slack variables is introduced into (2) as follows:

\[
y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i. \tag{4}
\]

The optimal separating hyperplane is determined so that the maximization of the margin and the minimization of the training error are achieved.

\[
\min \frac{1}{2} w^T w + \frac{C}{2} \sum_{i=1}^{n} \xi_i \quad \text{s. t.} \quad y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i, \tag{5}
\]

where \(C\) is a parameter that determines the tradeoff between the maximum margin and the minimum classification error. By forming the Lagrangian, the primal optimization problem can be translated into a dual quadratic programming problem:

\[
\max W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad \text{s. t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \text{and} \quad C \geq \alpha_i \geq 0, \quad \forall i. \tag{6}
\]

where \(K(x_i, x_j)\) is a kernel function defined as

\[
K(x_i, x_j) = \phi(x_i)^T \phi(x_j), \tag{7}
\]

By this, it is not necessary to explicitly know \(\phi(\cdot)\), and only the kernel function is enough for training SVMs. An \(x_i\) with nonzero \(\alpha_i\) is called a support vector. Let \(SV\) be the index set of support vectors.

Solving (6) for \(\alpha\) gives a decision function of the form

\[
f(x) = \text{sgn} \left( \sum_{i \in SV} y_i \alpha_i K(x_i, x) + b \right) \tag{8}
\]

whose decision boundary is a hyperplane in the feature space, and translates to nonlinear boundary in the original space.

There some widely used kernel functions such as polynomial kernels,

\[
K(x_i, x_j) = (x_i^T x_j + 1)^d,
\]

Gaussian Kernels, \textit{i.e.} RBF kernels,

\[
K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2).
\]

We can also get new kernel functions from simpler kernels by some computation such as addition and multiplication.

3. SVM Approach to Build Fuzzy Systems

3.1. FBF-based fuzzy system

In this paper, we consider a general fuzzy model with \(m\) fuzzy rules of the form

\[
\text{Rule } j: \quad \text{IF } x^1 \text{ is } A^j_1 \text{ and } x^2 \text{ is } A^j_2 \text{ and } \cdots \text{ and } x^l \text{ is } A^j_l \quad \text{THEN } y \text{ is } B_j, \quad j = 1, 2, \cdots, m \tag{9}
\]
where $x^i$ are the input variables; $y$ is the output variable of the fuzzy system; and $A^i_j$ and $B_j$ are linguistic terms characterized by fuzzy membership functions $\mu_{A^i_j}(x^i)$ and $\mu_{B_j}(y)$, respectively. If we choose product as the fuzzy conjunction operator, addition for fuzzy rule aggregation, and height defuzzification, then the overall fuzzy inference function is

$$f(x) = \frac{\sum_{j=1}^{m} b_j \prod_{i=1}^{l} \mu_{A^i_j}(x^i)}{\sum_{j=1}^{m} \prod_{i=1}^{l} \mu_{A^i_j}(x^i)}, \quad (10)$$

where $b_j$ is the point in the output space at which $\mu_{B_j}(y)$ achieves its maximum value. Furthermore, if we choose Gaussian membership function, then the fuzzy system is called fuzzy basis function (FBF)-based fuzzy inference system [21].

Let the fuzzy basis function be denoted as

$$p_j(x) = \frac{\prod_{i=1}^{l} \mu_{A^i_j}(x^i)}{\sum_{j=1}^{m} \prod_{i=1}^{l} \mu_{A^i_j}(x^i)}, \quad (11)$$

then $f(x)$ can be viewed as a linear combination of the fuzzy basis functions. In other word, the fuzzy inference system in (10) is equivalent to the FBF expansion

$$f(x) = \sum_{j=1}^{m} p_j(x)b_j, \quad (12)$$

For SVM learning, we modify the fuzzy basis function as [6]

$$p_j(x) = \prod_{k=1}^{l} \exp \left( -\frac{1}{2} \left( \frac{x^k - \bar{x}^k_j}{\sigma_k} \right)^2 \right), \quad (13)$$

where $\bar{x}^k$ and $\sigma_k$ are real-valued parameters, and the value $\sigma_k$ can be determined as suggested in [21]. The modified fuzzy basis function can be an eligible candidate of kernel function of the SVM since it satisfies Mercer’s Theorem [18]. In fact, it is a kernel produced from RBF kernel.

### 3.2. Learning framework

With the specialized fuzzy basis function (13) as the kernel function for SVM, we describe the SVM learning procedure as follows:

**Step 1–Assign parameters**

For given data set $(x_i, y_i), i = 1, 2, \cdots, n$, assign trade-off constant $C$ in (6) and kernel parameters $\sigma_k$ in (13).

**Step 2–Generate fuzzy rules from SVMs**

Using a SVM algorithm solving (6) to find support vectors $x^*_j$s whose components are centers of corresponding Gaussian membership functions, and the number of support vectors is the number of fuzzy rules. The generated rules take the form:

$$\text{IF } x^1 \text{ is } sv^1_j \text{ and } \cdots \text{ and } x^l \text{ is } sv^l_j, \text{ THEN } y_j = B_j,$$

where $sv^l_j$ is the fuzzy set of the $i$-th component of support vector $x^*_j$ by using Gaussian membership function.

**Step 3–Create a combined fuzzy rule base**

Because there are usually few support vectors generated during the SVM learning procedure, the number of fuzzy rules is small consequently. Thus, there maybe some “blind” zones for the rule base generated through the above two steps, i.e., the rule base may not be complete. To overcome this drawback, we put the linguistic rules given by the human expert and the generated rules from numerical data together to create a combined rule base. If there are conflicting rules, i.e., rules that have the same IF part but a different THEN part, we ask other experts to check which rule is more appropriate, and delete the other conflicting rules.

The three step procedure creates a combined fuzzy rule base whose rules are from either those generated from numerical data or linguistic rules. It determines a fuzzy system takes the form of (9), and a mapping of the form (10).

### 4. Example

The example is based on a nonlinearly separable situation where there are two circular shell classes, which is taken from [6]. In this case, class 1 contains a set of 16 “red” points ($y_i = 1$), whereas class 2 contains a set of 16 “blue” points ($y_i = -1$), as shown in Fig. 1.

![Figure 1. Scatter plot](image.png)
where \( p_j(x) \) is fuzzy basis function with \( \sigma_1 = \sigma_2 = 0.8 \) in quadratic programming-classification problem (QP,C).

After solving the (QP,C), those patterns with \( \alpha_i > 0 \) are termed as support vectors. It can be shown that 6 support vectors are obtained at positions of \( A = [0.13, 1.27]^T, B = [0.84, 0.50]^T, \ldots \), and \( F = [-1.03, -1.27]^T \), as shown in Fig. 2, with \( \alpha_1 \) equals to 4.2349, 4.1210, . . . , and 4.2349, respectively.

![Figure 2. Final decision boundary and six support vectors, marked with circles, obtained from the SVM learning][1]

Clearly, the 6 support vectors are assigned as fuzzy basis functions (i.e., kernel functions of the SVM modeling). The decision surface is then represented as series expansion of fuzzy basis functions, and this can naturally related to a set of fuzzy IF-THEN rules, as shown in Table 1. The phrase “\( x^i \) (or \( y \)) is near \( \alpha_i(\text{or} b) \)” is represented by the membership function \( \exp(-[(x^i - \alpha_i)^2 / 2\sigma^2]) \) with different \( \sigma \).

<table>
<thead>
<tr>
<th>Table 1. Rules obtained by SVM learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: IF ( x^1 ) is near 0.13 and ( x^2 ) is near 1.27, THEN ( y ) is near -1</td>
</tr>
<tr>
<td>R2: IF ( x^1 ) is near 0.84 and ( x^2 ) is near 0.59, THEN ( y ) is near -1</td>
</tr>
<tr>
<td>R3: IF ( x^1 ) is near -0.15 and ( x^2 ) is near -0.45, THEN ( y ) is near -1</td>
</tr>
<tr>
<td>R4: IF ( x^1 ) is near 0.15 and ( x^2 ) is near 0.45, THEN ( y ) is near 1</td>
</tr>
<tr>
<td>R5: IF ( x^1 ) is near -0.84 and ( x^2 ) is near -0.59, THEN ( y ) is near 1</td>
</tr>
<tr>
<td>R6: IF ( x^1 ) is near -0.13 and ( x^2 ) is near -1.27, THEN ( y ) is near 1</td>
</tr>
</tbody>
</table>

It can be seen from Table 1 that there are some “blind” zones, such as, we do not know from the table what \( y \) will be when \( x^1 \) lies between 0.15 and 0.84, or \( x^2 \) lies between 0.59 and 1.27, which are all somewhat big gaps. After consulting experts, we add two rules, shown in Table 2, to form the overall combined fuzzy rule base for decision, which can cover almost all possible conditions of this example.

<table>
<thead>
<tr>
<th>Table 2. Rules obtained from experts</th>
</tr>
</thead>
<tbody>
<tr>
<td>R7: IF ( x^1 ) is near 0.54 or ( x^2 ) is near 0.93, THEN ( y ) is near -1</td>
</tr>
<tr>
<td>R8: IF ( x^1 ) is near -0.54 or ( x^2 ) is near -0.93, THEN ( y ) is near 1</td>
</tr>
</tbody>
</table>

5. Conclusions

An architecture to extract fuzzy rules based on support vector machines is proposed, which takes both “subjective” experts’ prior knowledge and “objective” training data into account, and has satisfactory generalization ability and overfitting prevention capability. It can be seen from the paper that fuzzy system (12) with the fuzzy basis function (13) is similar with the decision function (8) of support vector machine without bias, i.e. \( b = 0 \). So, our future work will focus on revealing the essential relationship between fuzzy systems and support vector machines.

Acknowledgement

This work was supported partly by the Fund for Outstanding Researchers from Henan Province of China under Grant No. 074200510013, and the Natural Science Foundation from Education Department of Henan Province of China under Grant No. 2008A510015.

References


