International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:8, 2010

Fuzzy Rules Generation and Extraction from Support Vector Machine Based on Kernel Function Firing Signals

Prasan Pitiranggon, Nunthika Benjathepanun, Somsri Banditvilai, and Veera Boonjing

Abstract—Our study proposes an alternative method in building Fuzzy Rule-Based System (FRB) from Support Vector Machine (SVM). The first set of fuzzy IF-THEN rules is obtained through an equivalence of the SVM decision network and the zero-ordered Sugeno FRB type of the Adaptive Network Fuzzy Inference System (ANFIS). The second set of rules is generated by combining the first set based on strength of firing signals of support vectors using Gaussian kernel. The final set of rules is then obtained from the second set through input scatter partitioning. A distinctive advantage of our method is the guarantee that the number of final fuzzy IF-THEN rules is not more than the number of support vectors in the trained SVM. The final FRB system obtained is capable of performing classification with results comparable to its SVM counterpart, but it has an advantage over the black-boxed SVM in that it may reveal human comprehensible patterns.

Keywords—Fuzzy Rule Base, Rule Extraction, Rule Generation, Support Vector Machine.

I. Introduction

ARTIFICIAL Neural Networks (ANN) and Support Vector Machine (SVM) are great tools to approximate functions, recognize patterns, or predict outcomes [12], [23]; However, SVM are well accepted to be superior in performance over ANN in many applications, especially in Optical Character Recognition [18]. Despite their great performance, they both suffer from their black-box characteristics [3], [8], [17]. As a remedy to the problem, rule extraction is needed to make them white-box.

A limited number of studies of rule extraction from SVM have been conducted to obtain more understandable rules in order to explain how a decision was made or why a certain result was achieved. Much of the motivation for the field of rule extraction from support vector machines carries over from the more established area of rule extraction from artificial neural networks [3], [17].

Techniques used in extracting rules from SVM are both from the ones created specifically for SVM and the ones used successfully in other systems but can be applied to SVM. Only the techniques designed specifically for SVM will be related to because they are more relevant to our study. Other techniques can be found elsewhere [14].

P. Pitiranggon, N. Benjathepanun, and V. Boonjing are with the School of Computer Science, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, Chalongkrung Rd., Ladkrabang, Bangkok 10520, Thailand (phone: +668-1303-6484; email: prasan.pitiranggon@hotmail.com, kbnunthi@kmitl.ac.th, and kbveera@kmitl.ac.th).

S. Banditvilai is with the Department of Applied Statistics, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, Chalongkrung Rd., Ladkrabang, Bangkok 10520, Thailand (email: kbsomsri@kmitl.ac.th).

Techniques specifically intended as SVM rule extraction techniques are based on translucency, which can be either pedagogical or decompositional, and scope, which can be either classification or regression. Pedagogical techniques are those that try to relate inputs with outputs without making use of system structure, but decompositional techniques do make use of structure of the system. Classification techniques are the ones trying to differentiate input patterns, but regression techniques are trying to approximate function values. Previous studies on rule extraction techniques for SVM using decompositional techniques are SVM + Prototype by Nunez [20], Tree related method by Barakat [4], and Cubes and separating hyperplane related by Fung [10], while the ones using pedagogical techniques are Iter by Huysmans [13] and Minerva also by Huysmans [14].

The technique used in our study is considered decompositional in translucency and classification in scope. Unlike others, our technique makes use of strength of firing signal of support vectors partly similar to the way SVM makes decision. Moreover, our technique guarantees that the number of final rules is no more than the number of support vectors obtained by SVM. We also prove the equivalence of SVM and a type of fuzzy system to legitimize the fuzzy IF-THEN rules obtained.

II. Background

In decompositional techniques, Nunez et al. proposed a method called SVM + Prototypes [20]. This decompositional algorithm extracts interval classification rules from a trained SVM using hypercubes. The SVM + Prototypes algorithm is an iterative process that starts by training an SVM to obtain support vectors. It then uses a clustering algorithm to find new subsets and calculate the prototype (centroid) of each cluster (in low dimensional space). For each centroid, it finds the support vector located farthest from the prototype and uses the prototype as center and the support vector as vertex to create a hypercube in the input space. Then a partition test on each of the hypercubes is performed. This partition test is performed to minimize the level of overlapping between cubes for which the predicted class is different. If all subsets are processed, convert all of the current hypercubes into rules. Ellipsoids can also be used in place of hypercubes. For another decompositional technigue, decision tree is used by Barakat et al. [4]. This method makes use of the information provided by the support vectors and the parameters associated with them. The approach handles the rule-extraction by first, in a

learning stage, using labeled patterns to train an SVM and get an SVM model (classifier) with acceptable accuracy, precision, and recall. In the second stage of rule generation, the objective of this stage is to express the concepts learned by the model in a comprehensible form. The steps are firstly select the patterns that become support vectors, but discard their class label, then use the SVM model to predict the class label of those patterns, hence a special synthetic data set is generated. Finally the synthetic data set is used to train a machine learning technique with explanation capability, hence symbolic rules that represent the concepts learned by the SVM model are generated. As the last decompositional technique mentioned, Fung et al. proposed a method which only works when the input data set is linearly separable [10]. All input data are transformed into square observations in the interval 0 to 1. Then the method searches for a cube with one vertex on the separating hyperplane and the other located in the region below the separating hyperplane. Optimal cubes can be found from these cubes in two ways - volume maximization and point coverage maximization. The optimal cube divides the region below the separating hyperplane into two new regions - region above and on the right hand side of the cube. For an N-dimensional input space, one rule will create N new regions. Then a new optimal cube is found recursively for each new region. The algorithm stops after a predefined maximum number of iterations.

Iter [13] is the first pedagogical method for SVM rule extraction mentioned. The main idea of the algorithm is to iteratively expand a number of hypercubes until they cover the entire input space. The algorithm starts with the creation of a user defined number of random starting cubes. These cubes correspond to points in the input space. In each iteration, the following steps are executed: firstly, for each hypercube and for each input dimension, calculate how far the cube can be expanded to both extremes of the dimension before it intersects with another cube, call these distances LowerLimit and UpperLimit. Secondly, for each hypercube and for each input dimension, calculate the size of the update. The update equals a user-specified constant, unless this size would result in overlapping cubes. If this is the case then the update is smaller such that the two blocks become adjacent. Thirdly, for each hypercube and for each input dimension, create two temporary cubes adjacent to the original cube along the opposite sides of each input dimension with a width of update value from the second step. For each of both cubes, create a number of random points lying within the cube and calculate the mean prediction for these points according to the trained continuous regression model. Call the difference between each of both means and the mean prediction for the original cube respectively LowerDiff and UpperDiff. Lastly find the global minimum over all cubes of these differences and combine the temporary cube for which the difference was minimal with its original cube. Update the mean prediction for this cube and remove all other temporary cubes. Each of these cubes can then be converted into a rule of the following form:

IF Var 1 ∈ [Value1Low ,Value1High] AND Var 2 ∈ [Value2Low,Value2High] ... AND Var M ∈ [ValueMLow,ValueMHigh] THEN predict some Constant

with M the dimension of the input space. Minerva [14] is another pedagogical method for SVM rule extraction. Minerva is similar to sequential covering algorithm. The covering algorithm extracts a rule set by learning one rule first, removing the input data covered by that rule, and iterating on the remainder of the data. Starting from an empty rule set, the sequential covering algorithm first looks for a rule that is highly accurate for predicting a certain class. If the accuracy of this rule is above a user-defined threshold, the rule is added to the set of already found rules, and the algorithm is repeated over the rest of the inputs that were not correctly classified by this rule. If the accuracy of the rule is below this threshold, the algorithm ends. Because the rules in the rule set can be overlapping, the rules are first sorted according to their accuracy on the training data before they are returned to the user. In Minerva, there are differences compared to the sequential covering algorithms above; the most important one is that the rules are required to be non-overlapping. Another difference is that other sequential covering algorithms stop if the performance of the rule is below a certain threshold.

For our technique, we look for unbounded support vectors which are the data points used as base locations to define separating hyperplane (Fig. 1) and use the support vectors obtained in the previous step to build trained SVM decision network to classify testing data (Fig. 2). We show that SVM and a type of FRB, called Adaptive Network Fuzzy Inference System (ANFIS), are equivalent which means fuzzy IF-THEN rules can represent SVM decisions without loss of functionality. The first set of IF-THEN rules has equal sign in antecedent, e.g., IF $x = c$ THEN $y = d$, and this form will be called equality conditional IF-THEN rules. Then many equality conditional IF-THEN rules can be combined around support vectors based on firing strength, as part of rule generation, to form our FRB rules with range signs, e.g., IF $x > c1$ AND $x < c2$ THEN $y = d$, this form will be called range conditional IF-THEN rules. This second set of rules are modified further by combining completely coincided ranges among the IF-THEN conditionals (a process called input scatter partitioning), and this set of rules is our final set needed.

III. Method

A. Finding Unbounded Support Vectors

We use non-linear SVM classifier [2], [6], [7], [19], [23] in our study because it can handle both linear and non-linear data sets. We are given an input of Q data points $\{(X_i, d_i)\}, i = 1, \ldots, Q$ with input data $X_i \in \Re^n$ and binary class labels $d_i \in \{-1, +1\}$ the SVM classifier satisfies the following conditions:

$$W.\Phi(X) + w_0 \geq +1 - \xi_i, \qquad d_i = +1 \tag{1}$$

$$W.\Phi(X) + w_0 \leq -1 + \xi_i, \qquad d_i = -1 \tag{2}$$

where W is weight vector, and the w_0 is a bias constant value; the two values are obtained from training the SVM. The function $\Phi(.)$ is a non-linear function which maps the low dimensional input space into high dimensional space. The $d_i = +1$ means the output is the class we want to identify,

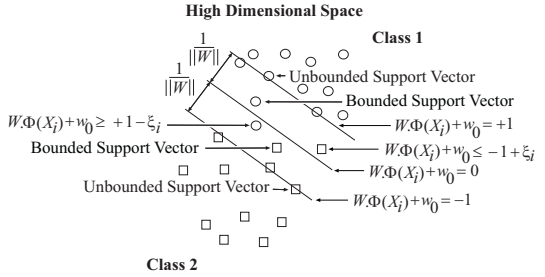

Fig. 1. Bounded and unbounded support vectors, misclassification vectors, and separating hyperplane

and the $d_i = -1$ means the output is the other class. The ξ_i is a slack variable to allow misclassification. The separating hyperplane, which is the dividing line between the two classes, is represented by an equation:

$$W.\Phi(X) + w_0 = 0 \tag{3}$$

The margin between the two classes (Fig. 1) can be maximized by minimizing:

$$\frac{1}{2} \parallel W \parallel^2 + C \sum_{i=1}^{Q} \xi_i \tag{4}$$

subject to

$$d_i[W.\Phi(X) + w_0] - 1 + \xi_i \geq 0, \qquad i = 1, \ldots, Q \tag{5}$$

$$\xi_i \geq 0 \tag{6}$$

Separating hyperplane ($W.\Phi(X)+w_0 = 0$) which is used in main classification decision and formed from two boundaries - class 1 boundary ($W.\Phi(X) + w_0 = +1$) and class 2 boundary ($W.\Phi(X) + w_0 = -1$) is shown in Fig. 1. Class 1 boundary is formed from unbounded support vectors of class 1 (circles), and class 2 boundary is formed from unbounded support vectors of class 2 (squares). Margin is a distance between the two boundaries which is equal to $\frac{2}{\|W\|}$. Bounded support vectors are support vectors which are not on the class boundary but are closer to the separating hyperplane. Misclassification vector of class 1 ($W.\Phi(X_i)+w_0 \geq +1-\xi_i$) is a vector which is considered to be class 1 even though it is located at a distance $1-\xi_i$ (or less) beyond separating plane into class 2 hyperspace. Misclassification vector of class 2 ($W.\Phi(X_i) + w_0 \leq -1+\xi_i$) is a vector which is considered to be class 2 even though it is located at a distance $1 - \xi_i$ (or less) beyond separating plane into class 1 hyperspace.

The part involving $\parallel W \parallel^2$ in the function maximizes the margin between the two classes in the feature space while the part involving C and ξ_i minimizes the misclassification error. The positive real constant C is a penalty parameter for misclassification. The Lagrangian with primal variables to the constraint optimization problem is given by

$$L_P(W, w_0, \Lambda, \xi, \Gamma) = \frac{1}{2} \parallel W \parallel^2 + C \sum_{i=1}^{Q} \xi_i + \sum_{i=1}^{Q} \lambda_i [d_i[W.\Phi(X_i) + w_0] - 1 + \xi_i] - \sum_{i=1}^{Q} \gamma_i \xi_i \tag{7}$$

where $\Lambda = (\lambda_1, \ldots, \lambda_Q)^T, \lambda_i \geq 0, \Gamma = (\gamma_1, \ldots, \gamma_Q)^T, \gamma_i \geq 0$ are the Lagrange multiplier vectors. The solution to the optimization problem is given by the saddle point of the Lagrangian where all partial derivatives with respect to W, w_0, and ξ_i go to zero. The Karush-Kuhn-Tucker complementary conditions,

$$\lambda_i[d_i(W.\Phi(X_i) + w_0) - 1 + \xi_i] = 0, \quad i = 1, \ldots, Q \tag{8}$$

must also be satisfied.

This gives dual form of (7):

$$L_D(\Lambda) = \sum_{i=1}^{Q} \lambda_i - \frac{1}{2} \sum_{i=1}^{Q} \sum_{j=1}^{Q} \lambda_i \lambda_j d_i d_j (\Phi(X_i).\Phi(X_j)) \tag{9}$$

where $(\Phi(X_i).\Phi(X_j)) = K(X_i, X_j)$ is called kernel function. The kernel function must satisfy Mercers Condition which is an existence of a mapping $\Phi(X)$ and an expansion of a symmetric kernel function,

$$K(X_i, X_j) = \sum_{k} (\Phi(X_i).\Phi(X_j)) \tag{10}$$

iff

$$\iint K(X_i, X_j) g(X_i) g(X_j) dX_i dX_j \geq 0 \tag{11}$$

For all $g(X)$ such that

$$\int g^2(X) dx < \infty \tag{12}$$

There are a few kernel functions which satisfy Mercers Condition. In this study, we use Gaussian kernel because it will make the creation of equivalent fuzzy rule-based system possible, and it has been proved to satisfy Mercers Condition.

To get support vectors, we need to maximize (9) subject to:

$$\sum_{i=1}^{Q} \lambda_i d_i = 0 \tag{13}$$

$$0 \leq \lambda_i \leq C, \qquad i = 1, \ldots, Q \tag{14}$$

Any input vector with non-zero Lagrange multiplier is a support vector.

There are two kinds of support vectors bounded and unbounded (Fig. 1). The unbounded support vectors are the ones used for defining the separating hyperplane. These unbounded support vectors guarantee maximal margin between the two classes; in terms of calculations, they have Lagrange multipliers greater than zero but less than the penalty parameter $C(\lambda_i > 0, \lambda < C)$. The bounded support vectors are the ones closer to the separating hyperplane than the unbounded support vectors, so these vectors geometrically bound the two classes; they have Lagrange multipliers equal to the penalty parameter In this study, we only need the unbounded support vectors because they are the ones defining the separating hyperplane, and the unbounded support vectors from one side of the separating hyperplane for the class being identified will be used for generating fuzzy IF-THEN rules.

One last parameter we need before reaching our final classifier equation is:

$$w_0 = \frac{1}{m} \sum_{k=1}^{m} (\frac{1}{d_k} - W.X_k) \tag{15}$$

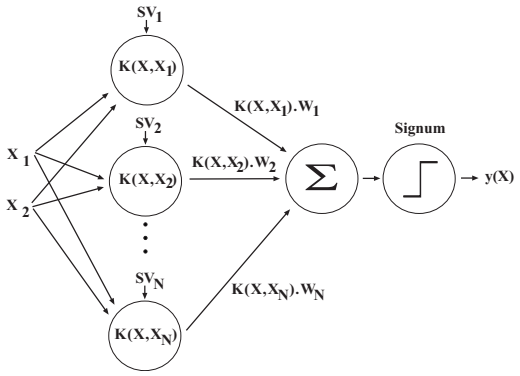

Fig. 2. An example of SVM decision network structure

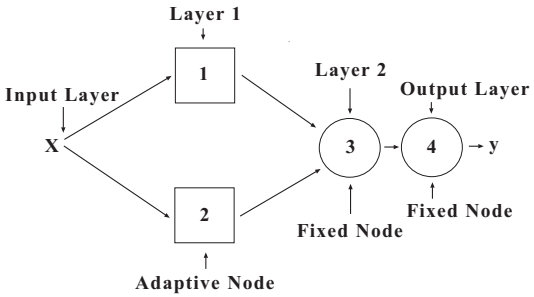

Fig. 3. An example of adaptive network with layers of functional nodes interconnected. Squares represent dynamic functional nodes, and circles represent fixed functional nodes.

where m = number of unbounded support vectors and

$$W = \sum_{k=1}^{N} \lambda_k d_k X_k \tag{16}$$

where N = number of all support vectors

We can now get the final classifier:

$$y(X) = sign(\sum_{i=1}^{N} d_i \lambda_i K(X_s, X_i) + w_0) \tag{17}$$

where X_s = unbounded support vector

This final equation will be used in the next step.

B. Constructing Trained SVM Decision Network

Equation (17) can be used to construct a trained SVM decision network. We will create FRB in the next step to be functionally equivalent to this SVM decision network to make it legitimate that IF-THEN rules can represent SVM decisions. Graphical representation of trained SVM decision network, which can be used to classify input data, is shown in Fig. 2.

C. Building Functionally Equivalent FRB from Trained SVM

The purpose of this part is to transform SVM decision network into fuzzy IF-THEN rules, but before we do that, we will make it legitimate that SVM decision network can be transformed into IF-THEN rules without loss in functionality by a formal proof of the functional equivalence between SVM decision network and a simplified class of fuzzy inference systems [15], [16].

1) Fuzzy IF-THEN Rules and Fuzzy Inference System Functions: Fuzzy Rule-Based System (FRB) is a system which contains IF-THEN rules [16], [22]. FRB is a fuzzy system which is characterized by having language component easily related to by human. The IF part contains antecedent (condition), and the THEN part contains consequent (outcome). The rule-based form uses linguistic variables as its antecedents and consequents. The antecedents express an inference or the inequality, which should be satisfied. The consequents are those, which we can infer, and is the output if the antecedent is satisfied. The fuzzy rule-based system uses IF-THEN rule-based system, given by, IF antecedent, THEN consequent. Each IF-THEN statement is a rule. An FRB usually contains many rules.

There are two well-known types of fuzzy inference method. Mamdanis fuzzy inference method is the most commonly seen inference method; it was developed by Mamdani in 1975 [18]. Another inference method is Takagi-Sugeno method of fuzzy inference process (TS method); this method was introduced by Sugeno in1985 [16]. The main difference between the two methods is in the consequent. Mamdani fuzzy systems use fuzzy sets as rule consequent while TS fuzzy systems use linear functions of input variables as rule consequent. Sugeno type system can be further divided into zero-ordered and first-ordered type. The zero-ordered type contains only constant in its consequent, but the first-ordered type contains linear equation with variables from the antecedent part.

There is a layered network system called Adaptive Network-Based Fuzzy Inference System (ANFIS) [16] which can be made functionally equivalent to ANN. ANFIS is based on Adaptive Network (Fig. 3) which contains layers of functional nodes with connectors; square nodes are dynamic nodes which depend on node parameters, and circle nodes are fixed nodes which have empty set of parameters. We can obtain fuzzy IF-THEN rules from the ANFIS which is derived from ANN. In this study, instead of making ANFIS equivalent to ANN, we make the ANFIS equivalent to SVM.

Fuzzy inference system is composed of a set of fuzzy IF-THEN rules, a database containing membership functions of linguistic labels, and an inference mechanism called fuzzy reasoning. Only TS FRB will be shown to be equivalent to SVM using the following model as an illustration.

Suppose we have a rule base consisting of two fuzzy IF-THEN rules of TS type:

Rule 1: If x_1 is A_1 and x_2 is B_1 then $f_1 = a_1x_1 + b_1x_2 + c_1$

Rule 1: If x_1 is A_2 and x_2 is B_2 then $f_1 = a_2x_1 + b_2x_2 + c_2$

Then the fuzzy reasoning mechanism can be illustrated in Fig. 4 where the firing strength of ith rule is obtained as the T-norm (usually minimum or multiplication operator) of the membership values on the premise part. In our case, we only use multiplication operator in T-norm step. Strength after T-

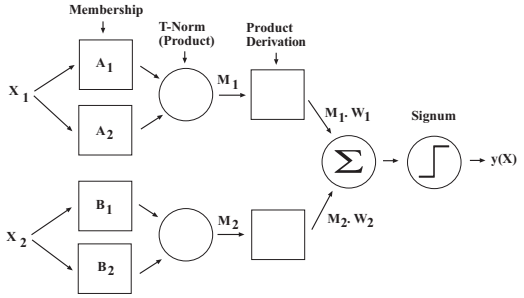

Fig. 4. An example of ANFIS equivalent to SVM, which uses Gaussian kernel function with two input vectors

norm with multiplication operator is:

$$M_i = \mu_{A_i}(X_1)\mu_{B_i}(X_2) \tag{18}$$

Note that overall output can be chosen as the weighted sum of each rules output:

$$f(X) = \sum_{i=1}^{R} M_i.w_i \tag{19}$$

where R is the number of fuzzy IF-THEN rules. And the decision equation is:

$$y(X) = sign(f(X) + b) \tag{20}$$

In summary, layer 1 calculates membership values, layer 2 performs T-norm operation, layer 3 derives the product of each rule's output and corresponding normalized weight, layer 4 sums its inputs to form the overall output, and layer 5 makes a decision by using signum function. The layer structure of this FRB is the form of ANFIS mentioned earlier.

2) Required Conditions for Functional Equivalence: The functional equivalence between a trained SVM decision network (Fig. 2) and ANFIS (Fig. 4) can be established if the following are true:

- The number of input patterns is equal to the number of fuzzy IF-THEN rules.
- The output of each fuzzy IF-THEN rule is composed of a constant.
- The membership functions within each rule are chosen as Gaussian functions with the same variance.
- The T-norm operator used to compute each rules firing strength is multiplication.
- Both the SVM decision network and the fuzzy inference system under consideration use the weighted sum method to derive their overall outputs.

3) Proof of functional equivalence between SVM decision network and ANFIS: Firstly functions in SVM decision network are stated:
Let P be a set of functions: $P = \{f_1, f_2, f_3, f_4\}$
Let X be a set of input vectors:
$X = \{X_1, X_2, X_3, \ldots, X_n\}$ where n is the total number of input vectors
Let S be a set of unbounded support vectors:
$S = \{S_1, S_2, S_3, \ldots, S_N\}$ where N is the total number of unbounded support vectors
Let $f_1(x, y)$ be Gaussian kernel function; we obtain

$$f_1(X, S_i) = exp\left[\frac{-(\parallel X - S_i \parallel)^2)}{\sigma^2}\right], \quad i = 1, \ldots, N \tag{21}$$

Let $f_2(x, y)$ be multiplication function; we obtain

$$f_2(f_1(X, S_i), w_i) = w_i.f_1(X, S_i) \tag{22}$$

Let $f_3(x) = \sum_{i=1}^{N} x_i$, we obtain

$$f_3(X) = \sum_{i=1}^{N} f_{2_i}(X) \tag{23}$$

Let $f_4(x) = sign(x + a)$, we obtain

$$f_4 = sign(f_3 + a) \tag{24}$$

Let y be the output; we obtain

$$y = f_4(f_3(f_2(f_1(X, S_i)))) = f_4 \circ f_3 \circ f_2 \circ f_1(X, S_i) \tag{25}$$

Secondly functions in ANFIS are stated.
Let Q be a set of functions in ANFIS: $Q = \{g_1, g_2, g_3, g_4\}$
Let X be a set of input vectors:
$X = \{X_1, X_2, X_3, \ldots, X_n\}$ where n is the total number of input vectors
Let S be a set of unbounded support vectors:
$S = \{S_1, S_2, S_3, \ldots, S_N\}$ where N is the total number of unbounded support vectors
Let $g_1(X, S_i)$ be Gaussian membership function with T-norm operation

$$\mu_{A_1}(x_1) = exp\left[\frac{-(x_1 - c_{A_1})^2}{\sigma_1^2}\right] \tag{26}$$

where $\mu_{A_1}(x_1)$ is membership function

$$M_i = \mu_{A_i}(x_1)\mu_{B_i}(x_2) \tag{27}$$

where M_i is result of T-norm operation at i
We obtain

$$g_1(X, S_i) = exp\left[\frac{-(\parallel X - S_i \parallel)^2}{\sigma^2}\right], \quad i = 1, \ldots, N \tag{28}$$

Let $g_2(x, y)$ be multiplication function; we obtain

$$g_2(g_1(X, S_i), w_i) = w_i.g_1(X, S_i) \tag{29}$$

Let $g_3(x) = \sum_{i=1}^{N} x_i$, we obtain

$$g_3(X) = \sum_{i=1}^{N} g_{2_i}(X) \tag{30}$$

Let $g_4(x) = sign(x + a)$, we obtain

$$g_4 = sign(g_3 + a) \tag{31}$$

Let z be the output; we obtain

$$z = g_4(g_3(g_2(g_1(X, S_i)))) = g_4 \circ g_3 \circ g_2 \circ g_1(X, S_i) \tag{32}$$

Proof: Since $f1 = g1, f2 = g2, f3 = g3$, and $f4 = g4$, then $y = z$, and X and Si are the same input in both systems; therefore, the two systems are functionally equivalent. ■

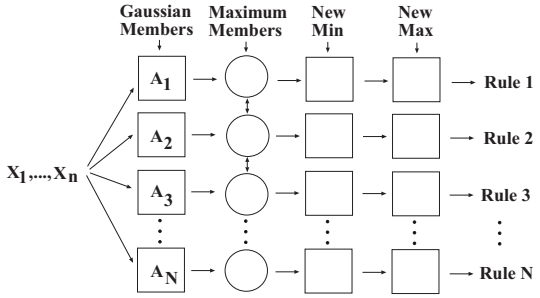

Fig. 5. Rules generation algorithm based on kernel firing strength

4) Equality Conditional IF-THEN Rules: An example of the equality conditional IF-THEN rules (Fig. 4) which is equivalent to SVM is in the form:
IF $x_1 = a_1$ AND $x_2 = b_1$ THEN $y = d_1$
IF $x_1 = a_2$ AND $x_2 = b_2$ THEN $y = d_2$

More generally, the equality conditional IF-THEN statements are in the following form:
If $x_{11} = a_{11}$ AND $x_{12} = a_{12}$ AND $\ldots x_{1n} = a_{1n}$ then $y = v_1$
If $x_{21} = a_{21}$ AND $x_{22} = a_{22}$ AND $\ldots x_{2n} = a_{2n}$ then $y = v_2$
If $x_{31} = a_{31}$ AND $x_{32} = a_{32}$ AND $\ldots x_{3n} = a_{3n}$ then $y = v_3$
.
.
.
If $x_{Q1} = a_{Q1}$ AND $x_{Q2} = a_{Q2}$ AND $\ldots x_{Qn} = a_{Qn}$ then $y = v_Q$
where a_{ij} are constants in $\Re$, v_i are binary values $[-1, 1]$ signifying the two classes equivalent to SVM classification, x_{ij} are n dimensional input data, and Q is total number of input data patterns in the dataset.

D. Rules Generation Based on Firing Strength

The purpose of this step is to generate preliminary rules based on the strongest firing signals associated with unbounded support vectors in high dimensional space.

In Fig. 5, all input patterns are entered into system one at a time. Gaussian kernel function as a membership function is calculated between current input and each of the support vectors, and the highest value is considered the strongest signal which will be the only one fired, and the rest will be ignored. The fired row then stores cumulative min and max value which will be replaced by new min or new max if it occurs. After all input patterns have been entered, min and max values in each row will be used as a range in conditional of each IF-THEN rule.

Schematic diagram of ANFIS implementing rule generation from unbounded support vectors found from previous step is shown in Fig. 5; X_i is input vector, and A_i is Gaussian kernel function of unbounded support vector and input vector.

The final min and max values of each row are used as a range of the newly generated IF-THEN rules. The range conditional IF-THEN statements are in the form:
Rule 1: If $(x_{11} > a_{11-}$ AND $x_{11} < a_{11+})$ AND $(x_{12} > a_{12-}\ AND\ x_{12} < a_{12+})$ AND $\ldots (x_{1n} > a_{1n-}\ AND\ x_{1n} < a_{1n+})$ THEN $y = v_1$
Rule 2: If $(x_{21} > a_{21-}$ AND $x_{21} < a_{21+})$ AND $(x_{22} > a_{22-}\ AND\ x_{22} < a_{22+})$ AND $\ldots (x_{2n} > a_{2n-}\ AND\ x_{2n} < a_{2n+})$ THEN $y = v_2$
Rule 3: If $(x_{31} > a_{31-}$ AND $x_{31} < a_{31+})$ AND $(x_{32} > a_{32-}\ AND\ x_{32} < a_{32+})$ AND $\ldots (x_{3n} > a_{3n-}\ AND\ x_{3n} < a_{3n+})$ THEN $y = v_3$
.
.
.
Rule N: If $(x_{N1} > a_{N1-}$ AND $x_{N1} < a_{N1+})$ AND $(x_{N2} > a_{N2-}\ AND\ x_{N2} < a_{N2+})$ AND $\ldots (x_{Nn} > a_{Nn-}\ AND\ x_{Nn} < a_{Nn+})$ THEN $y = v_N$
where a_{ij-} are lower range values (cumulative min) and a_{ij+} are upper range values (cumulative max) in $\Re$. N is the total number of unbounded support vectors, and n is the dimension of input vectors.

E. Input Scatter Partitioning

The purpose of this step is to reduce generated rules and refine rule extraction in low dimensional space. We can combine many range conditional IF-THEN statements from previous step together as long as it does not cause misclassification. Algorithms pseudo code for input scatter partitioning is:
[Pre-loop condition: Total number of IF-THEN rules equal to total number of support vectors]
FOR i = 1 TO N
[N = total number of generated rules]
IF rule i was eliminated THEN NEXT i
DO WHILE (no class overlap from another class) or (maximum value or minimum value of the input data set reached)
Expand ranges of IF-THEN conditional at i by a small value (less than 10% of min value of an attribute)
IF there is class overlap GOTO END WHILE
END IF
END WHILE
END IF
NEXT i
FOR i = 1 TO N
DO WHILE (there are still rules to merge for this i)
IF two ranges coincide then merge the two rules by retaining the larger ranges
END IF
END WHILE
NEXT i
[Post-loop condition: Number of IF-THEN rules are the same or less than rules in pre-condition]

We can use set membership symbol in place of greater than and less than signs as our final form of rules.
IF $x_{11} \in [a_{11-}, a_{11+}]$ AND $x_{12} \in [a_{12-}, a_{12+}]$ AND $\ldots x_{1n} \in [a_{1n-}, a_{1n+}]$ THEN $y = v_1$

IV. Experimental Results

We perform classification using both SVM and our fuzzy IF-THEN rules on five benchmark data sets which can

TABLE I
COMPARISON OF ERRORS FROM SVM AND RULES

Data Characteristics					SVM		Rules	
Data	N[a]	C[b]	Type	Att[c]	USV	%Er[d]	Rules	%Er
Iris	150	3	R[e]	4	37.07	2.89	7.27	6.22
Wine	178	3	R	13	157.57	14.04	52.53	8.99
Wisc	699	2	I[f]	10	300.6	5.27	67.5	4.83
Haber	306	2	I	4	105.7	29.08	63.7	46.41
Ionos	351	2	I, R	34	278.0	37.04	165.4	6.84

[a] Number of instances
[b] Number of classes
[c] Number of attributes
[d] Percent errors
[e] Real
[f] Integer

be downloaded from UCI Machine Learning Repository at http://www.ics.uci.edu; the five data sets are Iris [9], Wine [1], Wisconsin Breast Cancer [24], Habermans Survival Data [11], and Ionosphere [21]. These data sets are chosen to represent different number of instances, number of classes, input data type, and number of attributes. Ten-fold cross validation technique [5] is used in each data set. Data Characteristics are data set name (Data), number of instances (N), number of classes (C), data type (Type) which can be integer (I) or real (R), and number of attributes (Att). Results from each data set are average number of unsigned support vectors (USV), average percent error from SVM, average number of rules from our method, and average percent error from our method, and the classification results are shown in Table I. SVM performs classification with less error in Iris and Haberman (Haber) data sets than our method, but our method performs better in Wine, Wisconsin Breast Cancer (Wisc), and Ionosphere (Ionos) data sets. Average number of rules in each data set is lower than number of unsigned support vectors as claimed by our method.

The main reason why percent errors in SVM are different from our method is because of the noise (misclassification vectors) in the input data. SVM makes use of USV from both class 1 and class 2 (see Fig. 1) to make classification decisions, but our method uses only USV from class 1. If there is more misclassification vectors in class 1 region in hyperspace, our method performs worse than SVM. But if there is more misclassification vectors in class 2 region in hyperspace, SVM performs worse than our method.

V. CONCLUSION

The proposed method is shown to be a good alternative method for rule extraction from SVM and has an advantage over the decision method of SVM by revealing reasons behind the decision. And this makes it more attractive to be used in classification or prediction whenever we want to have insight into the way classification decision is made. The results of our experiments have shown that our method can outperform SVM decisions in some data sets, but most percent errors of the two methods are not far apart. It can be stated that the results of the errors from the two methods are comparable. Another advantage of our method compared to others is the guarantee that the number of rules in the final set will not exceed the number of support vectors. One of the suggestions for future study would be to use clustering algorithm in high noise data sets. In data sets with high noise, performance of IF-THEN rules by our algorithm may not perform well in classification. Also if there are large number of input data, scalability will suffer. K means clustering may be used in these cases to handle noisy data and also help scalability. After k means clustering is run, our algorithm can be implemented to obtain IF-THEN rules from SVM. Another suggestion for future study would be a modification of our method to handle data sets with a categorical data type. Logical AND operator can be used in the antecedents of IF-THEN statements to handle this kind of data.

REFERENCES

[1] S. Aeberhard, D. Coomans, and O. de Vel, "The Classification Performance of RDA," *Tech. Rep.,* no. 92-01, 1992.
[2] S. Ali and K. A. Smith-Miles, "Improved Support Vector Machine Generalization Using Normalized Input Space," In: A. Sattar and B.H. Kang (Eds.), *Artificial Intelligence*, LNAI 4304, Springer-Verlag, Berlin, Heidelberg, pp. 362-371, 2006.
[3] R. Andrews, J. Diederich, and A. B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowledge-Based Systems,* vol. 8, no. 6, pp. 373-389, 1995.
[4] N. Barakat and J. Diederich, "Eclectic Rule-Extraction from Support Vector Machines," *International Journal of Computational Intelligence,* vol. 2, no. 1, pp. 59-62, 2005.
[5] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995, pp. 372-375.
[6] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery,* vol. 2, pp. 121-167, 1998.
[7] C. Cortes and V. N. Vapnik, "Support Vector Networks," *Machine Learning,* vol. 20, pp. 273-297, 1995.
[8] J. Diederich (Ed.), "Rule Extraction from Support Vector Machines," *Studies in Computational Intelligence,* vol. 80, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 3-30.
[9] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annual Eugenics,* vol. 7, Part II, pp. 179-188, 1936.
[10] G. Fung, S. Sandilya, and R. B. Rao, "Rule extraction from linear support vector machines," in *Proceedings of the 11th ACM SIGKDD international Conference on Knowledge Discovery in Data Mining,* 2005, pp. 32-40.
[11] S. J. Haberman, "Generalized Residuals for Log-Linear Models," in *Proceedings of the 9th International Biometrics Conference,* Boston, 1976, pp. 104-122.
[12] S. Haykin, *Neural Networks, A Comprehensive Foundation,* Macmillan, New York, NY, 1994.
[13] J. Huysmans, B. Baesens, and J. Vanthienen, "ITER: an algorithm for predictive regression rule extraction," in *8th International Conference on Data Warehousing and Knowledge Discovery,* Springer Verlag, lncs 4081, 2006, pp. 270-279.
[14] J. Huysmans, R. Setiono, B. Baesens, and J. Vanthienen, "Minerva: Sequential Covering for Rule Extraction," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics,* vol. 38, no. 2, pp. 299-309, 2008.
[15] J. -S. R. Jang and C. -T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. Neural Networks,* vol. 4, pp. 156-158, 1992.
[16] J. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing.* Prentice Hall International, 1997, pp. 333-342.
[17] E. Kolman and M. Margaliot, "Are artificial neural networks white boxes?" *IEEE Trans. Neural Networks,* vol. 16, no. 4, pp. 844-852, 2005.
[18] S. Kumar,*Neural Networks: A Classroom Approach.* McGraw-Hill, International Edition, 2005, pp. 273-304.
[19] K. R. Muller, A. J. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen and V. N. Vapnik, "Predicting time series with support vector machines," in *ICANN,* 1997, pp. 999-1004.
[20] H. Nunez, C. Angulo, and A. Catala, "Rule Extraction Based on Support and Prototype Vectors," in *Studies in Computational Intelligence,* vol. 80, Springer-Verlag, Berlin, Heidelberg, 2008 pp. 109-134.

[21] V. J. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Technical Digest,* vol. 10, pp. 262-266, 1989.
[22] S. Sivanandam, S. Sumathi, and S. Deepa, *Introduction to Fuzzy Logic using MATLAB.* Springer Verlag, Berlin, Heidelberg, 2007.
[23] V. N. Vapnik, *Statistical Learning Theory.* John Wiley & Sons, 1998, pp. 375-520.
[24] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proceedings of the National Academy of Sciences,* vol. 87, pp. 9193-9196, 1990.