การวิเคราะห์หากฎเกณฑ์การตัดสินใจจากซัพพอร์ตเวกเตอร์แมชชีนโดยพิจารณาตามความแข็งแกร่งของฟังก์ชันเคอร์เนล

DECISION RULE EXTRACTION FROM SUPPORT VECTOR MACHINE BASED ON KERNEL FUNCTION FIRING STRENGTH

ประสาน พิติรางค์กร

PRASAN PITIRANGGON

วิทยานิพนธ์นี้สำหรับการศึกษาตามหลักสูตรปริญญาปรัชญาดุษฎีบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2554

KMITL – 2011 – SC – D – 002 - 012

DECISION RULE EXTRACTION FROM SUPPORT VECTOR MACHINE BASED ON KERNEL FUNCTION FIRING STRENGTH

PRASAN PITIRANGGON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2011

KMITL-2011-SC-D-002-012

หัวข้อวิทยานิพนธ์	การวิเคราะห์หากฎเกณฑ์การตัดสินใจจากซัพพอร์ตเวกเตอร์แมชชีนโดยพิจารณาตามความแข็งแกร่งของฟังก์ชันเคอร์เนล
นักศึกษา	นาย ประสาน พิติรางค์กร
รหัสประจำตัว	49062953
ปริญญา	ปรัชญาดุษฎีบัณฑิต
สาขาวิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2554
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ดร. นันทิกา เบญจเทพานันท์

บทคัดย่อ

วิทยานิพนธ์นี้ได้ศึกษา วิธีการวิเคราะห์หากฎเกณฑ์การตัดสินใจจากซัพพอร์ตเวกเตอร์แมชชีน และได้แสดงให้เห็นว่าการตัดสินใจของซัพพอร์ตเวกเตอร์แมชชีน ซึ่งถือกันว่าเป็นเสมือนกล่องดำสามารถถูกแทนที่ได้โดยกฎเกณฑ์ฟัซซี่ในการตรวจสอบเงื่อนไข (Fuzzy IF-THEN rules) โดยพิสูจน์ว่ากฎเกณฑ์การตัดสินใจของซัพพอร์ตเวกเตอร์แมชชีน กับกฎเกณฑ์ฟัซซี่แบบซูเกโน่ลำดับศูนย์ (zero-ordered Sugano) มีความสมมูลกัน ซึ่งเป็นการบ่งชี้ว่าเราสามารถใช้กฎเกณฑ์ฟัซซี่ตรวจสอบเงื่อนไขแทนวิธีการจากซัพพอร์ตเวกเตอร์แมชชีนได้จริง

วิทยานิพนธ์นี้ยังได้เสนอ วิธีการ วิเคราะห์กฎเกณฑ์การตัดสินใจจาก ซัพพอร์ตเวกเตอร์แมชชีน โดยพิจารณาตามความแข็งแกร่งของฟังก์ชันเคอร์เนล กับการขยายสเปซของซัพพอร์ตเวกเตอร์ที่ไม่มีขอบเขต (unbounded support vector) ซึ่งมั่นใจได้ว่าจำนวนของกฎการตรวจสอบเงื่อนไข (IF-THEN rules) จะน้อยกว่าหรือเท่ากับจำนวนซัพพอร์ตเวกเตอร์ และสามารถดึงกฎเกณฑ์การตัดสินใจที่ง่ายต่อความเข้าใจของมนุษย์ออกมาได้ จากนั้นทำการทดสอบวิธีการดังกล่าว กับข้อมูล 2 ชุด ได้แก่ ชุดข้อมูลมาตรฐาน (benchmark data sets) ที่นิยมในปัญหาการจำแนกประเภท และ ชุดข้อมูลที่ใช้ในการพยากรณ์วิกฤตค่าเงิน ซึ่งให้กฎเกณฑ์ฟัซซี่ตรวจสอบเงื่อนไข ที่สามารถทำนายผลลัพธ์ได้ถูกต้องใกล้เคียงกับวิธีมาตรฐานของซัพพอร์ตเวกเตอร์แมชชีน

Thesis Title	Decision Rule Extraction from Support Vector Machine Based on Kernel Function Firing Strength
Student	Mr. Prasan Pitiranggon
Student ID	49062953
Degree	Doctor of Philosophy
Program	Computer Science
Year	2011
Thesis Advisor	Asst. Prof. Dr. Nunthika Benjathepanun

ABSTRACT

This thesis studies rule extraction from Support Vector Machine (SVM). The thesis shows that decisions made by SVM, which is regarded as a black-boxed system, can be represented by fuzzy IF-THEN rules, which is regarded as a white-boxed system, by giving a proof that SVM decision network and the zero-ordered Sugeno Fuzzy Rule Base (FRB) type of the Adaptive Neuro-Fuzzy Inference System (ANFIS) are functionally equivalent indicating that SVM's decision can actually be represented by fuzzy IF-THEN rules. The thesis then proposes a rule extraction method based on kernel function firing strength and unbounded support vector space expansion. An advantage of the method is the guarantee that the number of final fuzzy IF-THEN rules is equal or less than the number of unbounded support vectors in standard SVM, and it may reveal human comprehensible patterns. The method is compared against standard SVM using popular benchmark data sets, and the results are comparable. An application of the method using non-benchmark data set is also conducted by applying the method to a data set with variables related to predicting currency crises. The result is a set of rules representing SVM's decisions on the currency crisis prediction with comparable accuracy.

ACKNOWLEDGEMENTS

A number of people have made contributions to this thesis and my Ph.D. study in one way or another, and I would like to express my sincere appreciation to all of them.

More than anything else, I am always thankful to family and friends for their continual and tireless supports throughout my life and especially during my Ph.D. study.

I am grateful to Associate Professor Dr. Veera Boonjing for giving invaluable comments and suggestions on a published paper and this thesis.

I am indebted to Assistant Professor Dr. Somsri Banditvilai for her insights into all my research studies right from the beginning of my Ph.D. study. She also had to give up her late hours to listen to my research progress countless times.

Special thanks go to Ms. Supapan Thearpiriyakij for providing knowledge and data on currency crises.

I would also like to thank Assistant Professor Dr. Nualsawat Hiranskolwong and Associate Professor Dr. Damras Wongsawang for their useful comments and supports for this thesis.

Most importantly, this thesis and my Ph.D. study would not be possible without my advisor, assistant professor Nunthika Benjathepanun, who was always there when help was needed. I deeply thank her with gratitude for everything she has done for me.

Prasan Pitiranggon

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figures Page

CHAPTER 1

Introduction

1.1 Introduction

Artificial Neural Networks (ANN) and Support Vector Machine (SVM) are great tools to approximate functions, recognize patterns, or predict outcomes [7, 8, 14, 24, 30]. However, SVM are well accepted to be superior in performance over ANN in many applications, especially in Optical Character Recognition [22]. Despite their great performance, they both suffer from their black-boxed characteristics [3, 9, 21]. We do not see explicitly how SVM makes decisions. This absence of a capability to explain in human comprehensible form the process by which an SVM arrives at a specific result hinders the possibility of more widespread acceptance of them, and makes them less suitable for safety-critical applications. In safety-critical or medical applications, an explanation capability is an absolute requirement.

A very different form of knowledge representation is provided by fuzzy rule-based systems. The theory of fuzzy logic allows us to handle and manipulate linguistic information based on perceptions rather than equations [28]. Fuzzy rule bases (FRBs) include a collection of IF-THEN rules, stated in natural linguistic terms, describing how the input affects the output. Thus, the knowledge is expressed in a form that humans can easily understand, verify, and refine. FRB system can be considered a white-boxed system.

A limited number of studies of rule extraction from SVM have been conducted to obtain more understandable rules. 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 ANN [3] [21]. It has been shown that ANN including radial basis function networks are equivalent to fuzzy systems [18], and this proves that ANN's decisions can be represented by fuzzy IF-THEN rules. But none of the previous studies on rule extraction from SVM has proved that SVM's decisions can be represented by fuzzy IF-THEN rules. They simply extracted IF-THEN rules from SVM and claimed that the rules obtained explained the decisions of SVM. There should be a proof that SVM's decisions are equivalent to fuzzy IF-THEN rules in order to make sure that the extracted rules are actually useful in reflecting SVM's decisions.

There was a proof based on study of the equivalence of Radial Basis Function Networks and ANFIS [18] which indicates that classification decisions of RBFN can be represented by fuzzy IF-THEN rules. Our study will provide a proof that the hidden decisions inside SVM are equivalent to the zero-ordered Takagi-Sugeno Fuzzy Rule-Based System (FRB) type of the Adaptive Network Fuzzy Inference System (ANFIS) provided that some special conditions are met.

1.2 Research Objective

The objective is to propose an alternative method in extracting fuzzy IF-THEN rules from SVM using Gaussian kernel function which is one of the kernel functions used in standard SVM. The final IF-THEN rules obtained should be capable of performing classification with results comparable to its SVM counterpart, but it has an advantage over the SVM in that it may reveal human comprehensible patterns which cannot be seen from SVM decision network. We also want to show that classification decisions of SVM can actually be represented by a type of fuzzy IF-THEN rules by providing a proof that the hidden decisions inside SVM can be revealed through the zero-ordered Takagi-Sugeno Fuzzy Rule-Based System (FRB) type of the Adaptive Neuro-Fuzzy Inference System (ANFIS).

1.3 Scope of Thesis

This thesis studies theories of SVM decision making and rule extraction algorithms for SVM. Fuzzy IF-THEN rules can be shown to represent SVM decision through the proof of functional equivalence of SVM decision network and ANFIS. An algorithm to extract fuzzy rules from SVM is then proposed. The algorithm obtained is used with benchmark data sets to compare with standard SVM in classification. A non-benchmark data set is also used to compare classification results between the two methods in order to show the applicability of our method.

1.4 Organization of Thesis

The following are the organization of the remainder of this thesis. In Chapter 2, necessary background components to support our work are described, namely, the SVM, ANFIS,

and previous rule extraction techniques for SVM, Chapter 3 presents proof of functional equivalence between SVM and ANFIS and also the rule extraction method for SVM using kernel function firing strength and input space expansion, In Chapter 4, results from the studies with benchmark and non-benchmark data are presented together with discussion of the results, and finally, conclusion and recommendation are given in Chapter 5.

CHAPTER 2

Literature Review

Support Vector Machine (SVM), Fuzzy system, and Adaptive Neuro-Fuzzy Inference System (ANFIS) are introduced in this chapter to provide background for our studies. Previous techniques for rule extraction of SVM are described to show how they are done so that they can be compared to our technique in the next chapter. Then previous studies on functional equivalence of Radial Basis Function Networks (RBFN) and a type of fuzzy system is presented as the basis for our proof of equivalence of SVM and a fuzzy system in the next chapter.

2.1 Support Vector Machine

SVM evolved from the sound theory to the implementation and experiments, while the Artificial Neural Networks (ANN) followed more heuristic path, from applications and extensive experimentation to the theory. The very strong theoretical background of SVM did not make it widely appreciated at the beginning. The publication of the first papers by Vapnik, Chervonenkis and co-workers in 1964 and 1965 went largely unnoticed till 1992 [20]. This was due to a widespread belief in the statistical and machine learning community that, despite being theoretically appealing, SVM was not suitable for practical applications. It was taken seriously only when excellent results on practical learning benchmarks were achieved in digit recognition, computer vision, and text categorization. Today, SVM shows equivalent or better results than ANN and other statistical models, on the most popular benchmark problems [7].

2.1.1 The Basics of Learning from Data

The learning problem of SVM is that there is some unknown and nonlinear function $y = f(X)$ between input vector X and scalar output y (or the vector output Y as in the case of multiclass SVM). There is no information about the underlying joint probability functions. Thus, one must perform learning from data with unknown probability distribution [30]. The only information available is a training data set $D = \{(X_i, y_i) \in X \times Y\}$, $i = 1,...,l$ where l stands for the number of the training data pairs and is therefore equal to the size of the training data

set D. Often, y_i is denoted as d_i, where d stands for a desired value. Since SVM needs to have training data sets, it belongs to the supervised learning techniques.

The problem stated above is similar to the classic statistical inference. However, there are several very important differences between the approaches and assumptions in training SVM and the ones in classic statistics and ANN modeling. Classic statistical inference is based on the following fundamental assumptions:

- Data can be modeled by a set of functions with linear parameters; this is a foundation of a parametric paradigm in learning from experimental data.
- In most of real-life problems, a stochastic component of data is the normal probability distribution law, that is, the underlying joint probability distribution is a Gaussian distribution.
- Because of the second assumption, the induction paradigm for parameter estimation is the maximum likelihood method, which is reduced to the minimization of the sum-of-errors-squares cost function in most engineering applications.

All three assumptions on which the classic statistical paradigm relied turned out to be inappropriate for many contemporary real-life problems [20] because of the following facts:

- Data in modern problems are high-dimensional, and if the underlying mapping is not very smooth the linear paradigm needs an exponentially increasing number of terms with an increasing dimensionality of the input space X. This is known as "the curse of dimensionality."
- The underlying real-life data generation laws may typically be very far from the normal distribution and a model-builder must consider this difference in order to construct an effective learning algorithm.
- From the first two points it follows that the maximum likelihood estimator, and consequently the sum-of-error-squares cost function, should be replaced by a new induction paradigm that is uniformly better, in order to model non-Gaussian distributions.
- Modern data sets are typically sparse meaning the data sets contain small number of the training data pairs.

ANN and SVM can overcome contemporary real-life data problems. ANN and SVM are the so-called nonparametric models. The term, nonparametric, does not mean that the ANN and SVM models do not have parameters at all. Unlike in classic statistical inference, the parameters

are not predefined and their number depends on the training data used. This set of parameters W is the very subject of learning and generally these parameters are called "weights". These parameters may have different geometrical or physical meanings. Depending upon the hypothesis space of functions H we are working with, the parameters W are usually the following:

- the hidden and the output layer weights in multilayer perceptrons
- the rules and the parameters, i.e., the positions and shapes of fuzzy subsets
- the coefficients of a polynomial or Fourier series
- the centers and variances of Gaussian basis functions as well as the output layer weights of RBFN
- the support vector weights in SVM

Two basic constructive approaches are widely accepted in designing a good learning model [30].

Firstly, a model that chooses an appropriate structure of the model (order of polynomials, number of hidden layer neurons, number of rules in the fuzzy logic model), keeps the estimation error (variance of the model) fixed, and minimizes the training error (empirical risk). This approach is called empirical risk minimization (ERM).

Secondly, a model that keeps the value of the training error (approximation error or empirical risk) fixed (equal to zero or equal to some acceptable level), and minimizes the variance. This approach is called structural risk minimization (SRM).

ANN uses the first approach while SVM uses the second. In both approaches, the resulting model should resolve the tradeoff between underfitting and overfitting the training data; this is known as a good generalization property of the model which is a desired property to make the model performs well on unseen data.

2.1.2 Empirical Risk Minimization

Suppose we have a machine whose task is to learn the mapping $X_i \rightarrow y_i$ where X_i is an input vector and y_i is a scalar output. The machine is actually defined by a set of possible mapping $X \rightarrow f(X,\alpha)$, where the functions $f(X,\alpha)$ are labeled by the adjustable parameters α. The machine is assumed to be deterministic which means for a given input X and a choice of α, it will always produce the same output $f(X,\alpha)$. A particular choice of α generates the so-

called trained machine. Thus, for example, a neural network with fixed architecture, with α corresponding to the weights and biases, is a learning machine in this sense [20].

The expectation of the test error for a trained machine is:

$$R(\alpha) = \int \frac{1}{2}|y - f(X,\alpha)|dP(X,y) \tag{2.1}$$

Generally probability distribution $P(X,y)$ is not known in practice. The quantity $R(\alpha)$ is called the expected risk or actual risk which is an average actual error on population data set. The empirical risk $R_{emp}(\alpha)$ is defined to be the mean error on the training data set for a fixed, finite number of observations l:

$$R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(x_i,\alpha)| \tag{2.2}$$

Note that no probability distribution appears here. $R_{emp}(\alpha)$ is a fixed number for a particular choice of α and for a particular training set $\{X_i, y_i\}$.

The quantity $\frac{1}{2}|y_i - f(X_i,\alpha)|$ is called the loss. ANN learns from data by reducing the empirical risk. One can never reduce the actual risk by exact calculations because the probability distribution $P(X,y)$ is very difficult to find in practice. The empirical risk will get closer to the actual risk when l gets larger. But reducing just the empirical risk alone can cause a problem of overfitting to the input data.

2.1.3 The Overfitting Problem

The mathematical term well-posed problem [30] is defined as mathematical models of physical phenomena with properties that:

1. A solution exists.

2. The solution is unique.

3. The solution depends continuously on the data, in some reasonable topology.

The learning-from-data problem is ill-posed [20]. The basic source of the ill-posedness of the problem is due to the infinite number of possible solutions to the learning problem. For

illustration, it is useful to remember that all functions that interpolate data points will result in a zero value for training error (empirical risk) as shown, in the case of regression, in Fig. 2.1. The figure shows a simple example of three-out-of-infinitely-many different interpolating functions of training data pairs sampled from a noiseless function $y = \sin(x)$.

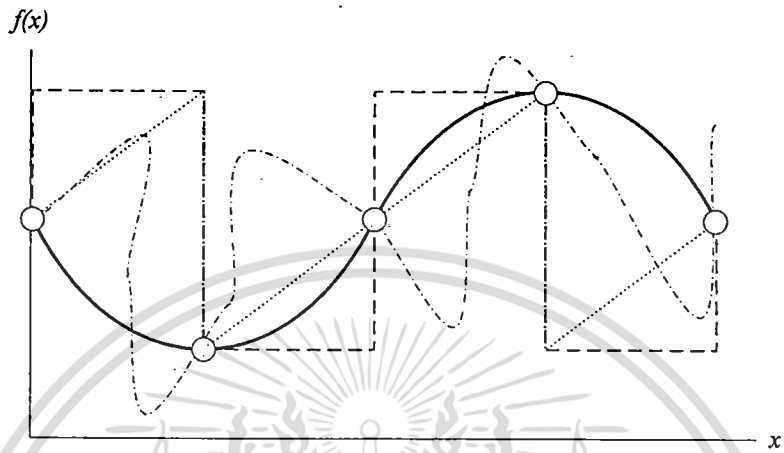


Figure 2.1 The overfitting problem in regression. Three-out-of-infinitely-many interpolating functions resulting in a training error equal to 0. The training is done with 5 data points (circles). However, dashed, dotted, and dot-dashed lines are bad models of a true function $y = \sin(x)$ (solid line).

In Fig. 2.1, each interpolant results in a training error equal to zero, but at the same time, each one is a very bad model of the true underlying dependency between x and y because all three functions perform very poorly outside the training inputs. In other words, none of these three particular interpolants can generalize well. However, not only interpolating functions can mislead; there are many other approximating functions (learning machines) that will minimize the empirical risk (approximation or training error) but not necessarily the generalization error (true, expected or guaranteed risk). This follows from the fact that a learning machine is trained by using some particular sample of the true underlying function, and consequently it always produces biased approximating functions. These approximants depend necessarily on the specific training data pairs used.

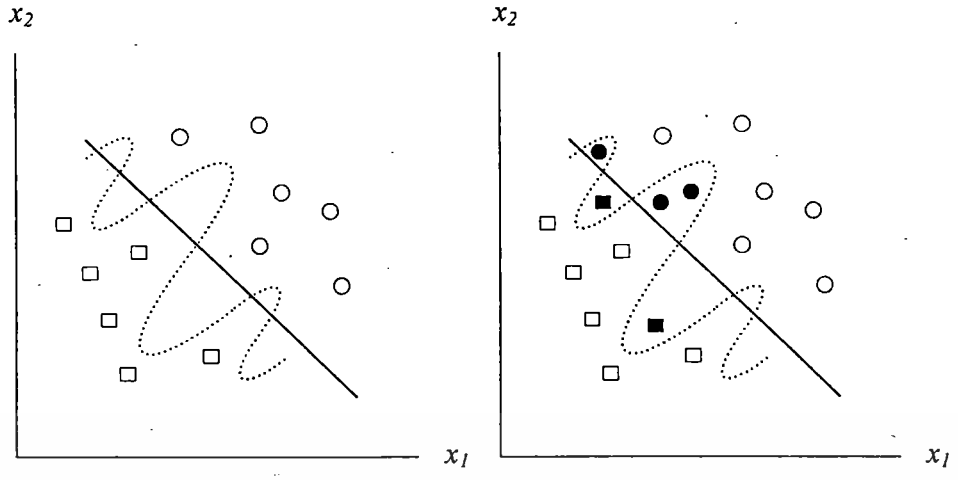


Figure 2.2 Overfitting in the case of linearly separable classification problem. Left: the perfect classification of the training data (empty circles and squares) by both low order linear model (solid line) and high order nonlinear one (dotted curve). Right: Wrong classification of some of the test data are shown (filled circles and squares) by a high capacity model, but all test data are correctly classified by the simple linear separation boundary.

Figure 2.2 shows a simple classification example where the classes are linearly separable. However, in addition to a linear separation, the learning was also performed by using a model of a high capacity (e.g., the one with Gaussian basis functions, or the one created by a high order polynomial, over the 2-dimensional input space) that produced a perfect separation boundary (empirical risk equals zero) too. However, such a model is overfitting the data and it will definitely perform very badly on unseen, test samples. Filled circles and squares in the right hand graph are all wrongly classified by the nonlinear model. Actually a simple linear separation boundary correctly classifies both the training and the test data.

A solution to this overfitting problem proposed in the framework of the statistical learning theory (SLT) is to reduce value of a parameter related to the capacity of learning machine called Vapnik-Chervonenkis Dimension (VC-dimension) through an induction principle of the SRM and its algorithmic realization through the SVM. Reducing overfitting means the learning machine will perform classification or regression better on unseen data.

2.1.4 Vapnik-Chervonenkis Dimension

The VC dimension is a property of a set of functions $\{f(\alpha)\}$ where α is a generic set of

parameters whose choice specifies a particular function [7, 30]. And the VC dimension can be defined for various classes of function f. As an illustration on VC dimension, we will only consider functions that correspond to the two-class pattern recognition case, so that $f(X,\alpha) \in \{-1,+1\}\ \forall X,\alpha$ where the positive class is $+1$, and the negative class is -1. Now if a given set of l points can be labeled in all possible 2^l ways, and for each labeling, a member of the set $\{f(\alpha)\}$ can be found which correctly assigns those labels, we say that set of points is shattered by that set of functions. The VC dimension for the set of functions $\{f(\alpha)\}$ is defined as the maximum number of training points that can be shattered by $\{f(\alpha)\}$.

Suppose that in the space in which the data live is in $\Re^2$, and the set $\{f(\alpha)\}$ consists of oriented straight lines, so that for a given line, all points on one side are assigned the class $+1$, and all points on the other side, the class -1. The orientation is shown in Figure 2.3 by an arrow, specifying on which side of the line points are to be assigned the label $+1$. While it is possible to find three points that can be shattered by this set of functions, it is not possible to find four. Thus the VC dimension of the set of oriented lines in $\Re^2$ is three.

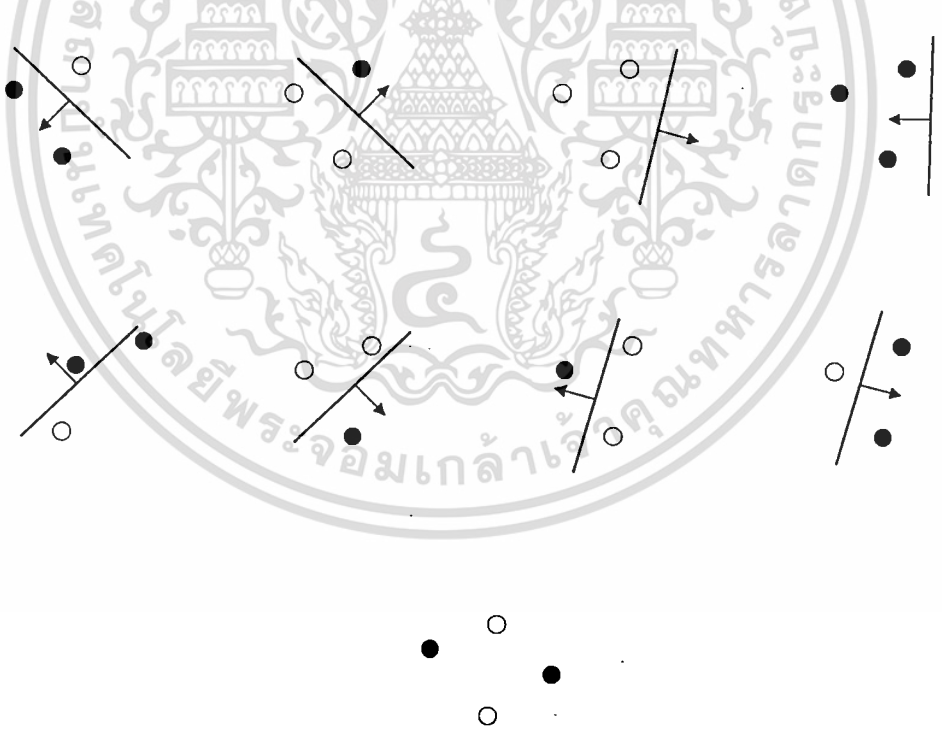


Figure 2.3 Illustration of VC dimension. All three points in $\Re^2$ can be shattered by straight lines, but four points at the bottom of the figure cannot be shattered by a straight line.

2.1.5 Structural Risk Minimization

The Structural Risk Minimization principle [20, 30] tries to minimize an expected risk (actual risk or the cost function) R comprising two terms given for the SVM as

$$R = \Omega(l,h) + R_{emp} \tag{2.3}$$

and it is based on the fact that for the classification learning problem with a probability of at least $1-\eta$ the following bound holds:

$$R(w_n) \le \Omega\left(\frac{h}{l}, \frac{\ln(\eta)}{l}\right) + R_{emp}(w_n) \tag{2.4}$$

The first term on the right hand side is named a VC confidence (confidence term or confidence interval) defined as

$$\Omega\left(\frac{h}{l}, \frac{\ln(\eta)}{l}\right) = \sqrt{\frac{h\left[\ln\left(\frac{2l}{h}\right)+1\right] - \ln\left(\frac{\eta}{4}\right)}{l}} \tag{2.5}$$

The parameter h is the VC (Vapnik-Chervonenkis) dimension of a set of functions. For binary classification, h is the maximal number of points which can be separated (shattered) into two classes in all possible 2^h ways by using the functions of the learning machine.

The notation for risks given above by using $R(w_n)$ denotes that an expected risk is calculated over a set of functions $f(X,W_n)$ of increasing complexity. Different bounds can also be formulated in terms of other concepts such as growth function or annealed VC entropy [22]. Bounds also differ for regression tasks. However, the general characteristics of the dependence of the confidence interval on the number of training data l and on the VC dimension h is similar and given in Fig. 2.4 [20].

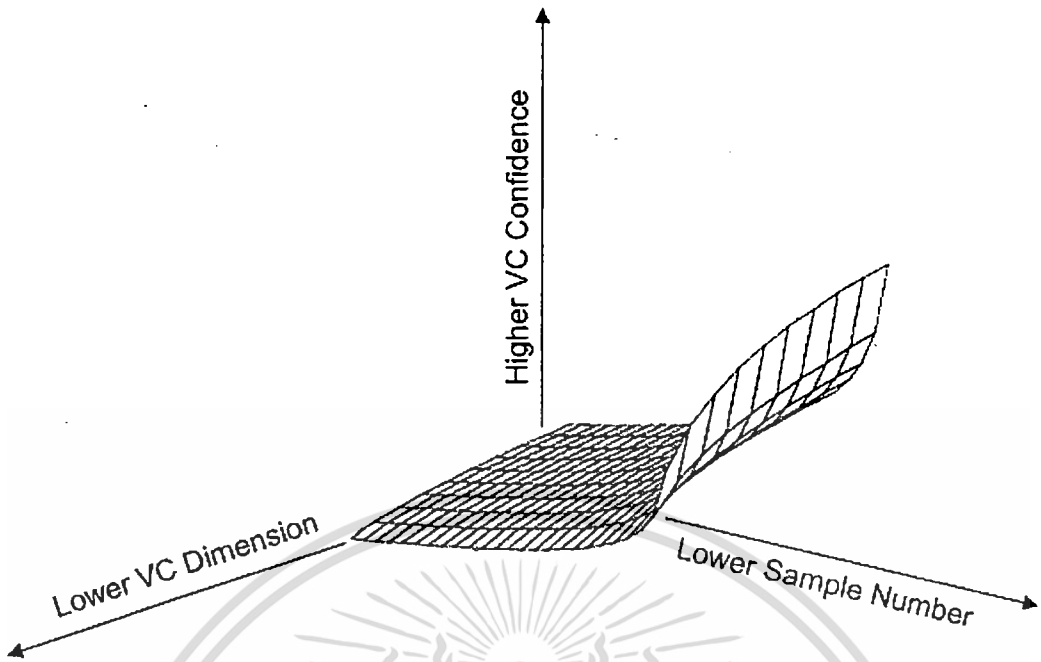


Figure 2.4 The dependency of VC confidence $\Omega(h,l,\eta)$ on the number of training data l and the VC dimension $h(h<l)$ for a fixed confidence level.

Equations (2.4) and (2.5) show that when the number of training data increases, i.e., for $l \to \infty$ (with other parameters fixed), an expected (true) risk $R(W_n)$ is very close to empirical risk $R_{emp}(W_n)$ because $\Omega \to 0$. On the other hand, when the probability approaches 1, the generalization bound grows large, because in the case when $\eta \to 0$, the value of $\Omega \to \infty$. This means that any learning machine obtained from a finite number of training data cannot have an arbitrarily high confidence level. There is always a trade-off between the accuracy provided by bounds and the degree of confidence in these bounds. Figure 2.4 also shows that the VC confidence interval increases with an increase in a VC dimension h for a fixed number of the training data pairs l.

The SRM is an inductive principle for learning from finite training data sets. It proved to be very useful when dealing with small samples. The basic idea of the SRM is to choose a model of the right capacity to describe the given training data pairs. This can be done by restricting the hypothesis space H of approximating functions and simultaneously controlling their flexibility (complexity). Thus, learning machines will be those models that, by increasing the number of parameters (typically called weights W_i here), form a nested structure in the following sense

$$H_1 \subset H_2 \subset H_3 \ldots H_{n-1} \subset H_n \ldots H \tag{2.6}$$

In such a nested set of functions, every function always contains a previous, less complex, function. Typically, H_n may be a set of polynomials in one variable of degree n, fuzzy logic model having n rules, multilayer perceptrons having n hidden layer neurons, or SVM structured over n support vectors. The goal of learning is one of a subset selection that matches training data complexity with approximating model capacity. In other words, a learning algorithm chooses an optimal polynomial degree or, an optimal number of hidden layer neurons or, an optimal number of fuzzy logic model rules, for a polynomial model or NN or fuzzy logic model respectively. For learning machines linear in parameters, this complexity (expressed by the VC dimension) is given by the number of weights. For approximating models nonlinear in parameters, the calculation of the VC dimension is often not an easy task. Nevertheless, even for these networks, by using simulation experiments, one can find a model of appropriate complexity. Next we will use all these theories in the construction of SVM.

2.1.6 Support Vector Machines in Classification

A support vector learning machine can be thought of as

- a set of functions implemented in an SVM or
- an induction principle or
- an algorithmic procedure for implementing the induction principle on the given set of functions [20].

SVM implements the SRM induction principle on the given set of functions. It implements the strategy by keeping the training error fixed and minimizes the confidence interval. Below, for an easy to understand sequence, we first consider a simple example of linear decision rules (i.e., the separating functions will be hyperplanes) for binary classification of linearly separable data. In such a problem, we are able to perfectly classify data pairs, meaning that an empirical risk can be set to zero. It is the easiest classification problem and yet an excellent introduction of all relevant and important ideas underlying the statistical learning theory, SRM, and SVM.

The subsequent problems will gradually increase in complexity. It will begin with a linear maximal margin classifier for linearly separable data where there is no sample overlapping. Afterwards, we will allow some degree of overlapping of training data pairs. However, we will still try to separate classes by using linear hyperplanes. This will lead to the linear soft margin classifier for overlapping classes. Finally in problems where linear decision hyperplanes are no longer feasible, the mapping of an input space into the so-called feature space (hyperdimensional space) will take place resulting in the nonlinear classifier. There will be no dedicated section on construction of SVM for regression as we only deal with classification in this study.

2.1.7 Linear Maximal Margin Classifier for Linearly Separable Data

Consider the problem of binary classification or dichotomization. Training data are given as

$$(X_1, y_1), (X_2, y_2), \ldots, (X_l, y_l), X \in \Re^n, y \in \{+1, -1\} \tag{2.7}$$

For reasons of visualization, we will consider the case of a two-dimensional input space, i.e., $X \in \Re^2$. Data are linearly separable and there are many different hyperplanes that can perform separation (Fig. 2.5). Actually, for $X \in \Re^2$, the separation is performed by planes $w_1 x_1 + w_2 x_2 + b = 0$. In other words, the decision boundary, i.e., the separation line in input space is defined by the equation $w_1 x_1 + w_2 x_2 + b = 0$. We need to find the best separation plane.

The difficult part is that all we have at our disposal are sparse training data. Thus, we want to find the optimal separating function without knowing the underlying probability distribution $P(X, y)$. There are many functions that can solve given pattern recognition tasks. In such a problem setting, the statistical learning theory shows that it is crucial to restrict the class of functions implemented by a learning machine to one with a complexity that is suitable for the amount of available training data.

In the case of a classification of linearly separable data, this idea is transformed into the following approach – among all the hyperplanes that minimize the training error (i.e., empirical risk) find the one with the largest margin. By looking at Fig. 2.5 we will find that the dashed separation line shown in the right graph seems to promise good classification while facing

previously unseen data, the so-called generalization phase. At least, it seems to probably be better in generalization than the dashed decision boundary having smaller margin shown in the left graph. This can also be expressed as a classifier with smaller margin having higher expected risk. When the largest margin is found, the SRM principle has actually been used, and this will be described in section 2.1.9.

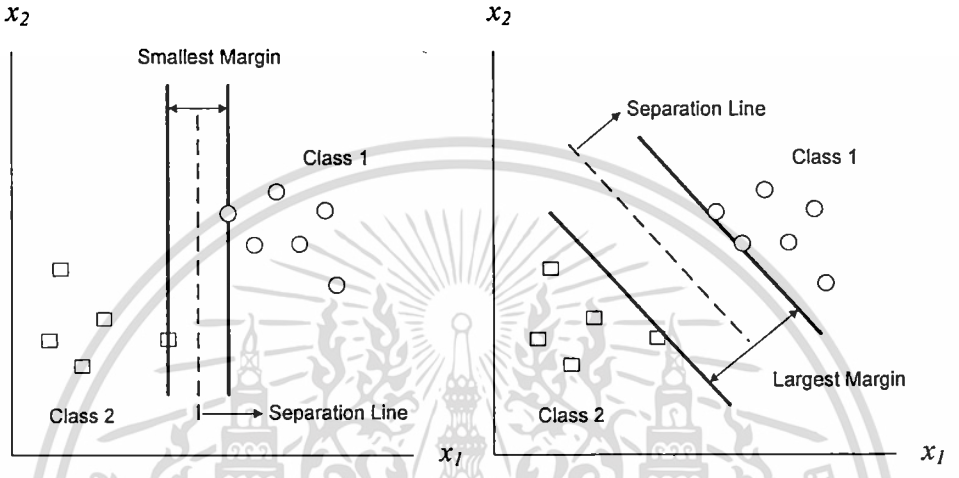


Figure 2.5 Largest and smallest margin between two classes. Two-out-of-many separating lines: a good one with a large margin (right) and a less acceptable separating line with a small margin (left)

2.1.8 Construction of SVM for Linearly Separable Data

The first step is to form a hyperplane W normal to all input vectors X [7], we obtain

$$W \cdot X = 0 \tag{2.8}$$

where $W \in \Re^n$, $X = \{X_1, X_2, X_3, \ldots, X_l\}$, $X \in \Re^n$, l is the total number of input vectors, and n is the dimension of input vectors. W is a hyperplane with direction normal to all input vectors X and passes through origin. Then we move W to some point between class 1 and class 2, we obtain a discriminant function

$$d(X, W, b) = W \cdot X + b = 0 = \sum_{i=1}^{n} w_i x_i + b \tag{2.9}$$

where the scalar b is called a bias and $X, W \in \Re^n$ [20]. This equation is depicted in Fig. 2.6.

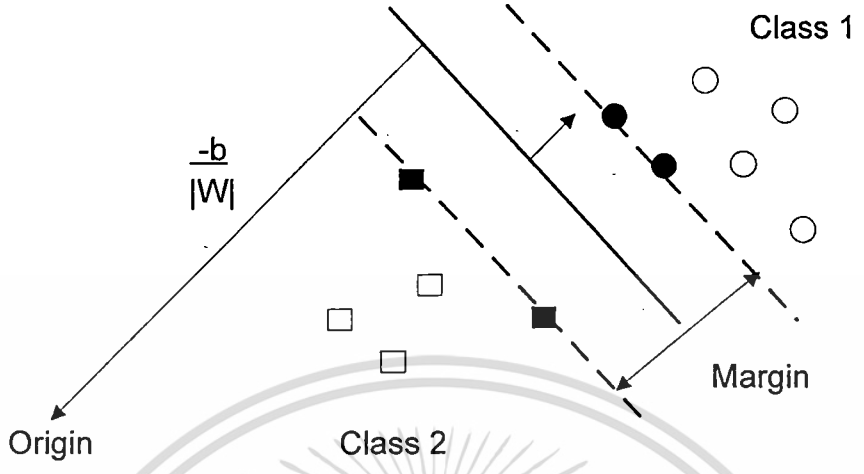


Figure 2.6 Linear separating hyperplane between two classes.

The discriminant rules are:

if $d(X_p, W, b) > 0$, the unseen pattern X_p belongs to class 1, i.e., $output = +1$,

and

if, $d(X_p, W, b) < 0$, the unseen pattern X_p belongs to class 2, i.e., $output = -1$.

We can combine the two discriminant rules into one output function (indicator function):

$$y = sign(d(X_p, W, b)) \tag{2.10}$$

For $X \in \Re^2$, the output function y given by (2.10) is a step-wise function, and the discriminant function $d(X, W, b)$ is a hyperplane. There is one more mathematical object in classification problems called a separation boundary that lives in the same n-dimensional space of input vectors X. Separation boundary separates vectors X into two classes. Here, in cases of linearly separable data, the boundary is also a separating hyperplane. The separation boundary is an intersection of a discriminant function $d(X, W, b)$ and a space of input features. It is given by

$$d(X, W, b) = 0 \tag{2.11}$$

All these functions and relationships can be followed, for two-dimensional inputs X, in Fig. 2.7. In this particular case, the decision boundary i.e., separating hyperplane is actually a separating line in an $x_1 - x_2$ plane and a discriminant function $d(X,W,b)$ is a plane over the 2-dimensional space of features, i.e., over an $x_1 - x_2$ plane.

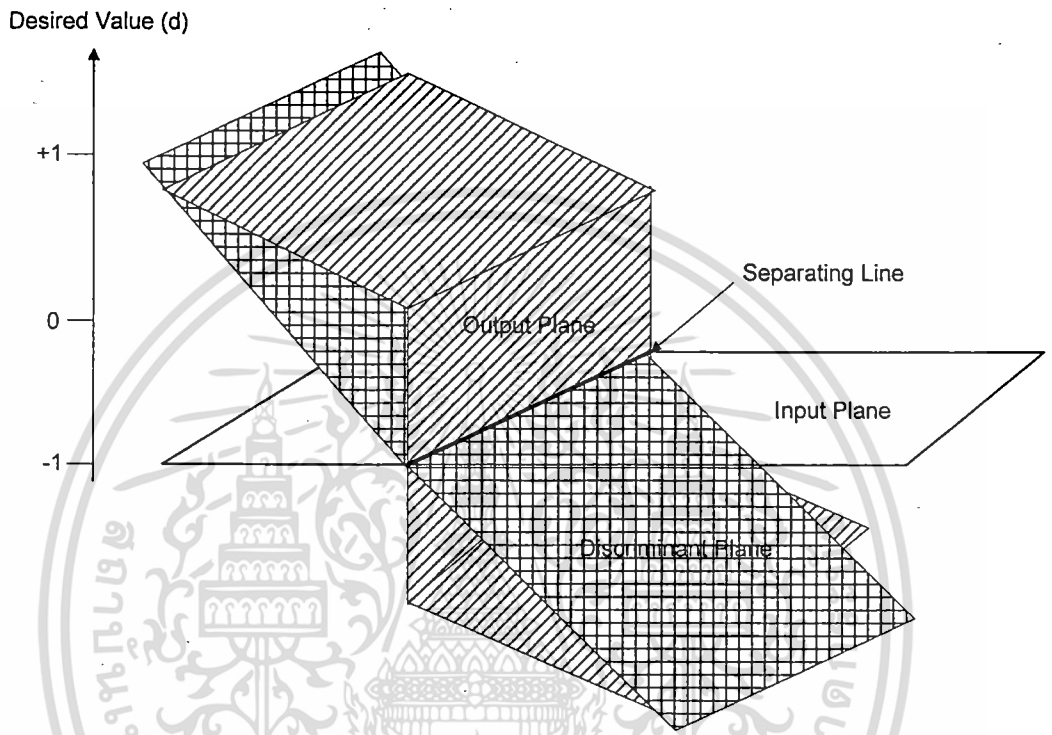


Figure 2.7 Two dimensional input, discriminant, and output plane. The definition of a discriminant function or hyperplane $d(X,W,b)$, a separating boundary $d(X,W,b) = 0$ and an output function $y = sign(d(X,W,b))$ whose value represents a learning, or SVM's output.

In the case of one-dimensional training patterns x, discriminant function $d(X,W,b)$ is a straight line in an $x - y$ plane. An intersection of this line with an x-axis defines a point that is a separation boundary between two classes. This can be followed in Fig. 2.8. Before attempting to find an optimal separating hyperplane having the largest margin, we introduce the concept of the canonical hyperplane. We depict this concept with the help of the one-dimensional example shown in Fig. 2.8.

The discriminant plane $d(X,W,b)$ shown in Fig. 2.8 is also a canonical plane. Namely, the values of d and of y are the same and both are equal to $|1|$ for the support vectors depicted by a dark square and circle. At the same time, for all other training patterns, we have $|d| > |y|$. In

order to present a notion of this new concept of the canonical plane, first note that there are many hyperplanes that can correctly separate data. In Fig. 2.8, three different discriminant functions $d(X,W,b)$ are shown. There are infinitely many more. In fact, given $d(X,W,b)$, all functions $d(X,kW,kb)$, where k is a positive scalar, are correct discriminant functions too. Because parameters (W,b) describe the same separation hyperplane as parameters (kW,kb) there is a need to introduce the notion of a canonical hyperplane [30].

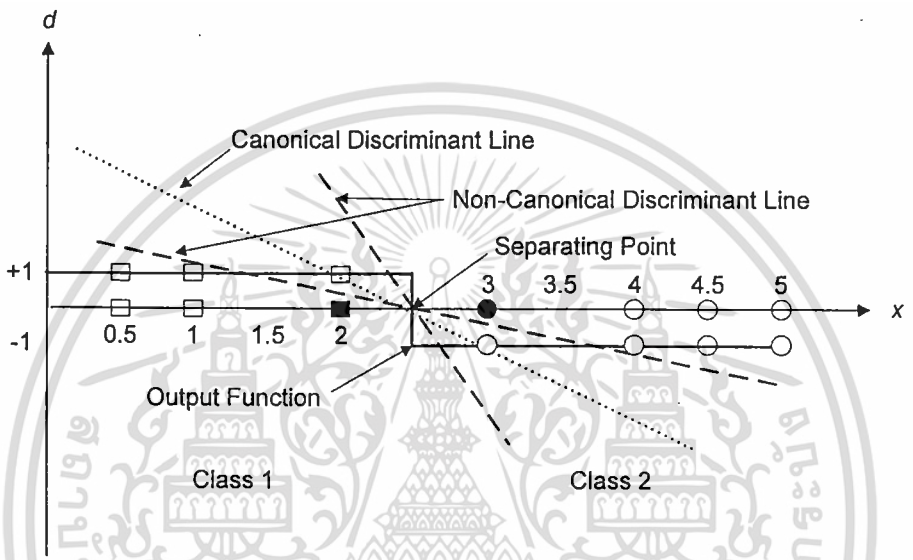


Figure 2.8 One dimensional input, discriminant, and output plane. SV classification for one-dimensional inputs by the linear discriminant function, and graphical presentation of a canonical hyperplane is shown. For one-dimensional inputs, it is actually a canonical straight line (depicted as a straight dotted line) that passes through points $(+2,+1)$ and $(+3,-1)$ defined as the support vectors (a dark square and circle). The two dashed lines are the two other discriminant hyperplanes (i.e., straight lines). The training input patterns $\{x_1 = 0.5, x_2 = 1, x_3 = 2\} \in$ Class 1 have a desired value $y = +1$. The inputs $\{x_4 = 3, x_5 = 4, x_6 = 4.5, x_7 = 5\} \in$ Class 2 have the desired value $y = -1$.

A hyperplane is in the canonical form with respect to training data $X \in \Re^n$ if

$$\min|W \cdot X_i + b| = 1 \tag{2.12}$$

The dotted line $d(X,W,b) = -2x + 5$ in Fig. 2.8 fulfills (2.12) because its minimal absolute value for the given seven training patterns belonging to two classes is 1. It achieves this value for two patterns, chosen as support vectors, namely for $x_3 = 2$, and $x_4 = 3$. For all other patterns, $|d| > 1$. There are many different hyperplanes (planes and straight lines for 2-D and 1-D problems in Figs. 2.7 and 2.8 respectively) that have the same separation boundary (solid line and a dot in Figs. 2.7 and 2.8 respectively). At the same time there are far fewer hyperplanes that can be defined as canonical ones fulfilling (2.12). In Fig. 2.8, i.e., for a 1-dimensional input vector X, the canonical hyperplane is unique. This is not the case for training patterns of higher dimension. Depending upon the configuration of class elements, various canonical hyperplanes are possible. Therefore, there is a need to define an optimal canonical hyperplane (OCSH) as a canonical hyperplane having a maximal margin. This search for a separating, maximal margin, canonical hyperplane is the ultimate learning goal in statistical learning theory underlying SVM. The hyperplane obtained from a limited training data must have a maximal margin because it will probably better classify new data. It must be in canonical form because this will ease the quest for significant patterns, here called support vectors. The canonical form of the hyperplane will also simplify the calculations. Finally, the resulting hyperplane must ultimately separate training patterns.

To maximize the separating margin, we need to find an expression for the margin first. We derive an expression for margin M between the closest members from two classes by first considering the data points on the positive side of the separating hyperplane characterized by W, b [22]. For convenience, we refer to this separating hyperplane as Π. Note that the direction of W is normal to Π. X_+ is the data point from class 1 that is closest to Π. Let X_Π be the unique point on Π that is closest to X_+ (see Fig. 2.9).

We are interested in maximizing $\|X_+ - X_\Pi\|$. From the defining equation of Π and Eq. 2.12:

$$W \cdot X_+ + b = 1 \tag{2.13}$$

$$W \cdot X_\Pi + b = 0 \tag{2.14}$$

From this pair of equations we may write:

$$W \cdot (X_+ - X_\Pi) = 1 \tag{2.15}$$

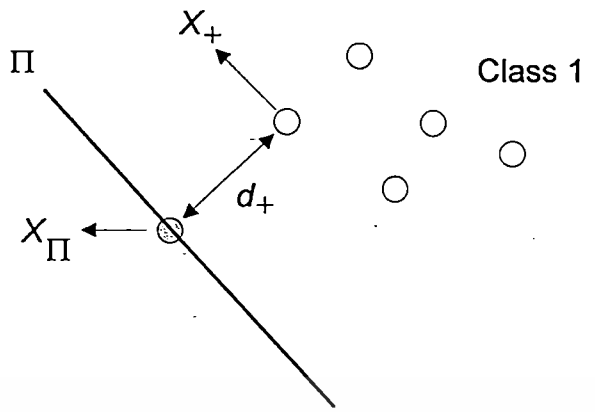


Figure 2.9 Closest data point to the hyperplane. X_+ is the data point from class 1 closest to hyperplane Π and X_Π is the unique point on Π closest to X_+.

However, since $X_+ - X_\Pi$ is also normal to Π, it has the same direction as W, we can expand the left hand side of Eq. 2.15 as:

$$W \cdot (X_+ - X_\Pi) = W \cdot \left(\|X_+ - X_\Pi\| \frac{W}{\|W\|} \right) \tag{2.16}$$

$$W \cdot (X_+ - X_\Pi) = \left(\|X_+ - X_\Pi\| \frac{\|W\|^2}{\|W\|} \right) \tag{2.17}$$

$$W \cdot (X_+ - X_\Pi) = \|X_+ - X_\Pi\| \|W\| \tag{2.18}$$

From Eq. 2.15 this implies that

$$\|X_+ - X_\Pi\| \|W\| = 1 \tag{2.19}$$

Or the distance d_+ is

$$d_+ = \|X_+ - X_\Pi\| = \frac{1}{\|W\|} \tag{2.20}$$

$$M = d_+ + d_- = \frac{2}{\|W\|} \tag{2.21}$$

This important result will have a great consequence for the constructive (i.e., learning) algorithm in a design of a maximal margin classifier. It will lead to solving a quadratic programming (QP) problem which will be shown shortly. Hence, the gradient learning in ANN will be replaced by solution of the QP problem here. This is the next important difference between the ANN and SVM and follows from the implementation of SRM in designing SVM, instead of a minimization of the sum of error squares, which is a standard cost function for ANN.

Equation 2.21 shows that minimization of a norm of a hyperplane normal weight vector $\|W\| = \sqrt{W^T \cdot W} = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$ leads to a maximization of margin M. Because a minimization of $\sqrt{f}$ is equivalent to the minimization of f, the minimization of a norm $\|W\|$ equals a minimization of $\|W\|^2 = W^T \cdot W = \sum_{i=1}^{n} w_i^2 = w_1^2 + w_2^2 + \ldots + w_n^2$, and this leads to a maximization of a margin M. Hence, the learning problem is

$$\text{Minimize } \frac{1}{2}\|W\|^2 \tag{2.22}$$

subject to constraints introduced and given in (2.23) below. A multiplication of $\|W\|^2$ by 0.5 is for numerical convenience only, and it does not change the solution. Note that in the case of linearly separable classes empirical error equals zero ($R_{emp} = 0$) and minimization of $\|W\|^2$ corresponds to a minimization of a confidence term Ω. The optimal canonical separating hyperplane (OCSH) with the largest margin defined by $M = 2/\|W\|$, specifies support vectors, i.e., training data points closest to it, which satisfy $y_j(W.X_j + b) = 1, j = 1, \ldots, N_{SV}$ where N_{SV} is the total number of support vectors. For all the other (non-support vector data points) the OCSH satisfies inequalities $y_j(W.X_j + b) > 1$. In other words, for all the data, OCSH should satisfy the following constraints

$$y_i(W \cdot X_i + b) \geq 1 \qquad i = 1, \ldots, l \tag{2.23}$$

where l denotes a number of training data points. The last equation can be easily checked visually in Figs. 2.7 and 2.8 for two-dimensional and one-dimensional input vectors X respectively. Thus, in order to find the optimal separating hyperplane having a maximal margin, a learning machine should minimize $\|W\|^2$ subject to the inequality constraints (2.23). This is a

classic quadratic optimization problem with inequality constraints. Such an optimization problem is solved by the saddle point of the Lagrange functional (Lagrangian)

$$L(W,b,\Lambda)=\frac{1}{2}\|W\|^2-\sum_{i=1}^{l}\lambda_i\left[y_i\left(W\cdot X_i+b\right)-1\right] \tag{2.24}$$

where the λ_i are Lagrange multipliers. The search for an optimal saddle point (W_0,b_0,Λ_0) is necessary because Lagrangian L must be minimized with respect to W and b, and has to be maximized with respect to nonnegative λ_i (i.e., $\lambda_i \geq 0$ should be found). In forming the Lagrangian, for constraints of the form $f_i > 0$, the inequality constraints equations are multiplied by nonnegative Lagrange multipliers (i.e., $\lambda_i \geq 0$) and subtracted from the objective function.

This problem can be solved either in a primal space (which is the space of parameters W and b) or in a dual space (which is the space of Lagrange multipliers λ_i). The second approach gives insightful results and we will consider the solution in a dual space below. In order to do that, we use Karush-Kuhn-Tucker (KKT) conditions for the optimum of a constrained function. In our case, both the objective function (2.24) and constraints (2.23) are convex and KKT conditions are necessary and sufficient conditions for a maximum of (2.24). These conditions are, at the saddle point (W_0,b_0,Λ_0), derivatives of Lagrangian L with respect to primal variables should vanish which leads to

$$\frac{\partial L}{\partial W_0}=0, \qquad \text{i.e.,} \qquad W_0=\sum_{i=1}^{l}\lambda_i y_i X_i \tag{2.25}$$

$$\frac{\partial L}{\partial b_0}=0, \qquad \text{i.e.,} \qquad \sum_{i=1}^{l}\lambda_i y_i=0 \tag{2.26}$$

and the KKT complementarity conditions below (stating that at the solution point the products between dual variables and constraints equals zero) must also be satisfied,

$$\lambda_i\left[y_i\left(W\cdot X_i+b\right)-1\right]=0, \qquad i=1,\ldots,l \tag{2.27}$$

Substituting (2.25) and (2.26) into a primal variables Lagrangian $L(W,b,\Lambda)$ (2.24), we change to the dual variables Lagrangian $L_d(\lambda)$

$$L_d(\lambda) = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i}^{l} \sum_{j}^{l} y_i y_j \lambda_i \lambda_j (X_i \cdot X_j) \tag{2.28}$$

In order to find the optimal hyperplane, a dual Lagrangian $L_d(\lambda)$ has to be maximized with respect to nonnegative λ_i (i.e., λ_i must be in the nonnegative quadrant) and with respect to the equality constraint as follows

$$\lambda_i \geq 0 \qquad i = 1, \ldots, l \tag{2.29}$$

$$\sum_{i=1}^{l} \lambda_i y_i = 0 \tag{2.30}$$

Note that the dual Lagrangian $L_d(\lambda)$ is expressed in terms of training data and depends only on the scalar products of input patterns $(X_i \cdot X_j)$. The dependency of $L_d(\lambda)$ on a scalar product of inputs will be very handy later when analyzing nonlinear decision boundaries and for general nonlinear regression. Note also that the number of unknown variables equals the number of training data l. After learning, the number of free parameters is equal to the number of support vectors but it does not depend on the dimensionality of input space. Such a standard quadratic optimization problem can be expressed in a matrix notation as follows:

$$\text{Maximize } L_d(\Lambda) = \Lambda \cdot \mathbf{1} - \frac{1}{2} \Lambda^T \mathbf{H} \Lambda \tag{2.31}$$

subject to

$$Y \cdot \Lambda = 0 \tag{2.32}$$

$$\lambda_i \geq 0 \qquad i = 1, \ldots, l \tag{2.33}$$

where $\Lambda = [\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_l]$, $\mathbf{H}$ denotes the Hessian matrix $H_{ij} = y_i y_j (X_i \cdot X_j)$ of this problem, and $\mathbf{1} = [11 \ldots 1]$. (Note that maximization of (2.31) equals a minimization of $L_d(\Lambda) = -\Lambda \cdot \mathbf{1} + \frac{1}{2} \Lambda^T \mathbf{H} \Lambda$, subject to the same constraints). Solutions λ_{0i} of the dual optimization problem above determine the parameters W_o and b_0 of the optimal hyperplane according to (2.25) and (2.27) as follows

$$W_0 = \sum_{i=1}^{l} \lambda_{0i} y_i X_i \tag{2.34}$$

$$b_0 = \frac{1}{N_{SV}}\left[\sum_{s=1}^{N_{SV}}\left(\frac{1}{y_s} - X_s \cdot W_0\right)\right] \tag{2.35}$$

$$b_0 = \frac{1}{N_{SV}}\left[\sum_{s=1}^{N_{SV}}\left(y_s - X_s \cdot W_0\right)\right] \tag{2.36}$$

In deriving (2.36) we used the fact that y can be either $+1$ or -1, and $1/y = y$. N_{SV} denotes the number of support vectors. There are two important observations about the calculation of W_0. First, an optimal weight vector W_0, is obtained in (2.34) as a linear combination of the training data points and second, W_0 (same as the bias term b_0) is calculated by using only the selected data points called support vectors. The fact that the summations in (2.34) goes over all training data patterns (i.e., from 1 to l) is irrelevant because the Lagrange multipliers for all non-support vectors equal zero ($\lambda_{0i} = 0$, $i = N_{SV} + 1, \ldots, l$). Finally, having calculated W_0 and b_0, we obtain a discriminant hyperplane $d(X)$ and an output function $y = sign(d(X))$ as given below

$$d(X) = \sum_{i=1}^{l} w_{0i} \cdot x_i + b_0 = \sum_{i=1}^{l} y_i \lambda_i (X_i \cdot X) + b_0, \tag{2.37}$$

$$y = sign(d(X)) \tag{2.38}$$

Training data patterns having non-zero Lagrange multipliers are called support vectors. For linearly separable training data, all support vectors lie on the margin and they are generally just a small portion of all training data (typically, $N_{SV} << l$). Figure 2.10 shows the geometry of standard results for non-overlapping classes.

Before presenting applications of OCSH for both overlapping classes and classes having nonlinear decision boundaries, we will comment only on whether and how support vector based linear classifiers actually implement the SRM principle.

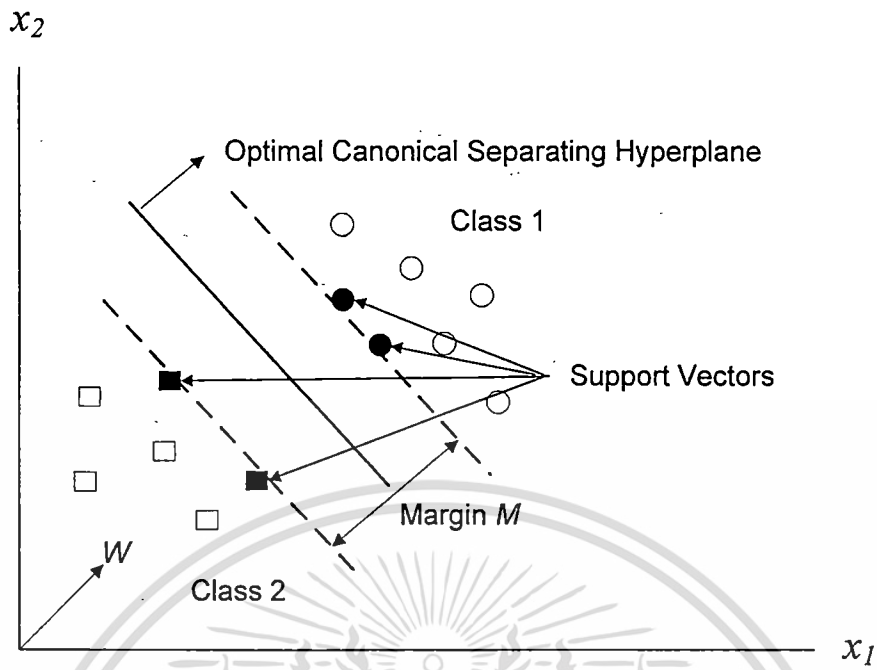


Figure 2.10 The optimal canonical separating hyperplane with the largest margin intersects halfway between the two classes. The points closest to it (satisfying $y_j|W \cdot X_j + b| = 1$, $j = 1,\ldots,N_{SV}$) are support vectors and the OCSH satisfies $y_i(W \cdot X_i + b) \geq 1$, $i = 1,\ldots,l$ (where l denotes the number of training data and N_{SV} stands for the number of support vectors). Four support vectors are the filled circles and squares.

2.1.9 Structural Risk Minimization in Support Vector Machines

It can be shown that an increase in margin reduces the number of points that can be shattered i.e., the increase in margin reduces the VC dimension, and this leads to the decrease of the SVM capacity [7, 22]. In other words, by minimizing $\|W\|$ (i.e., maximizing the margin) the SV machine training actually minimizes the VC dimension and consequently a generalization error (expected risk) at the same time. This is achieved by imposing a structure on the set of canonical hyperplanes and then, during the training, by choosing the one with a minimal VC dimension. A structure on the set of canonical hyperplanes is introduced by considering various hyperplanes having different $\|W\|$. In other words, we analyze sets S_A such that $\|W\| \leq A$. Then, if $A_1 \leq A_2 \leq A_3 \leq \ldots \leq A_n$, we introduced a nested set $S_{A1} \subset S_{A2} \subset S_{A3} \subset \ldots \subset S_{An}$. Thus, if we impose the constraint $\|W\| \leq A$, then the canonical hyperplane cannot be closer than $1/A$

to any of the training points X_i. Vapnik [30] states that the VC dimension h of a set of canonical hyperplanes in $\Re^n$ such that $\|W\| \leq A$ is

$$h \leq \min\{R^2 A^2, n\} + 1 \tag{2.39}$$

where all the training data points (vectors) are enclosed by a hypersphere of the smallest radius R (Fig. 2.11).

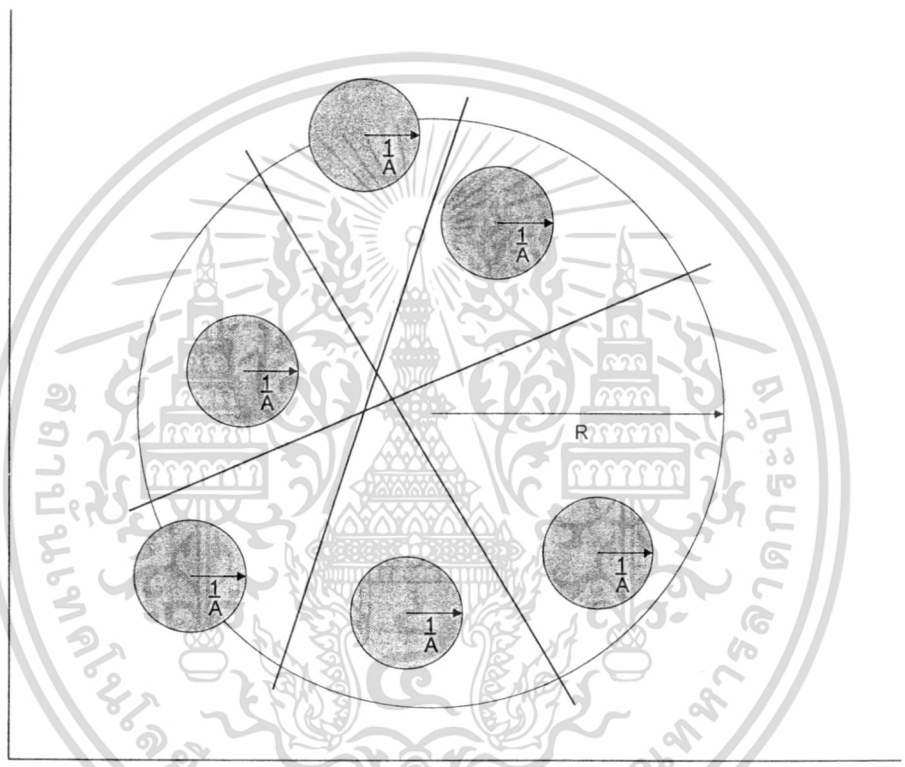


Figure 2.11 Reducing VC dimension by increasing size of hyperspheres around data points. Hyperspheres with radius 1/A within a hypersphere with radius R are increased in size to reduce VC dimension.

Therefore, a small $\|W\|$ results in a small h, and minimization of $\|W\|$ is an implementation of the SRM principle. In other words, a minimization of the canonical hyperplane weight norm $\|W\|$ minimizes the VC dimension according to (2.39). Once the support vectors have been found, we can calculate the bound on the expected probability of committing an error on a test example as follows:

$$E_l[P(error)] \leq \frac{E[N_{SV}]}{l} \tag{2.40}$$

where E_l denotes expectation over all training data sets of size l. Note how easy it is to estimate this bound that is independent of the dimensionality of the input space. Therefore, an SVM having a small number of support vectors will have good generalization ability even in a very high-dimensional space.

2.1.10 Linear Soft Margin Classifier for Overlapping Classes

The learning procedure presented above is valid for linearly separable data, meaning for training data sets without overlapping. Such problems are rare in practice. At the same time, there are many instances when linear separating hyperplanes can be good solutions even when data are overlapped [20, 30]. However, quadratic programming solutions as given above cannot be used in the case of overlapping because the constraints $y_i(W \cdot X_i + b) \geq 1, \quad i = 1,\ldots,l$ cannot be satisfied. In the case of an overlapping (see Fig. 2.12), the overlapped data points cannot be correctly classified and for any misclassified training data point X_i, the corresponding λ_i will tend to infinity. This particular data point (by increasing the corresponding λ_i value) attempts to exert a stronger influence on the decision boundary in order to be classified correctly (see Fig. 2.12). When the λ_i value reaches the maximal bound, it can no longer increase its effect, and the corresponding point will stay misclassified. In such a situation, the algorithm introduced above chooses almost all training data points as support vectors. To find a classifier with a maximal margin, the algorithm for linear separable data presented earlier must be changed allowing some data to be unclassified. We must leave some data on the wrong side of a decision boundary. In practice, we allow a soft margin, and all data inside this margin (whether on the correct side of the separating line or on the wrong one) are neglected. The width of a soft margin can be controlled by a corresponding penalty parameter C (introduced below) that determines the trade-off between the training error and VC dimension of the model.

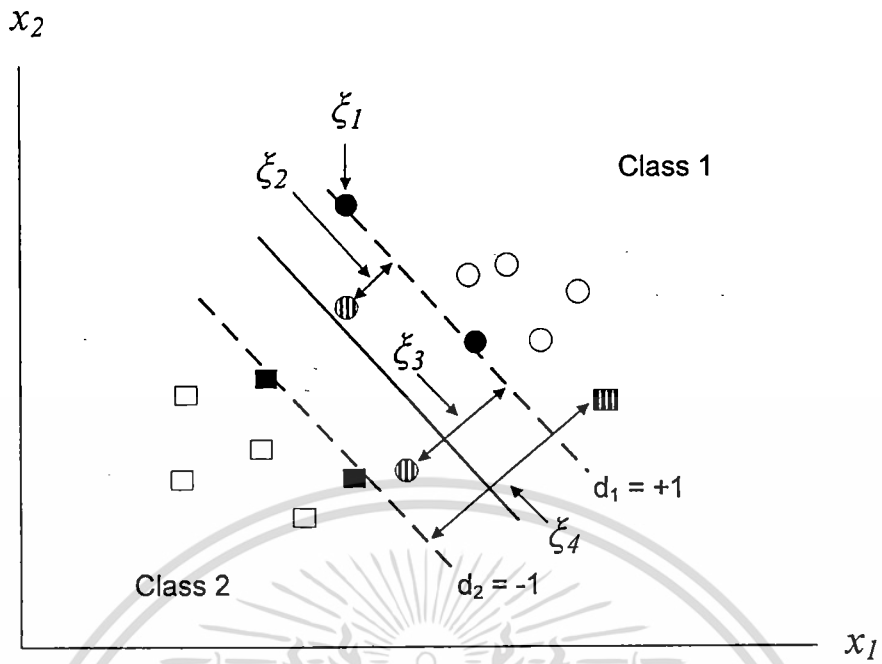


Figure 2.12 The soft decision boundary for a dichotomization problem with data overlapping. Separation line (solid), margins (dashed), unbounded support vectors (black training data points with ξ_1), and bounded support vectors (striped circle with ξ_2). 1 misclassifications for positive class (striped circle with ξ_3) and 1 misclassification for negative class (striped square with ξ_4). $\xi_1 = 0,\ 0 \le \xi_2 \le 1,\ \xi_3 = 1 - d_2 = 2 > 1,\ \xi_4 = 1 + d_1 = 2 > 1$.

The question now is how to measure the degree of misclassification and how to incorporate such a measure into the hard margin learning algorithm given earlier. The simplest method would be to form the following learning problem

$$\text{Minimize } \frac{1}{2}\|W\|^2 + C \text{ (number of misclassified data)} \tag{2.41}$$

where C is a penalty parameter, trading off the margin size (defined by $\|W\|$, i.e., by $W^T \cdot W$) for the number of misclassified data points. Large C leads to small number of misclassifications, bigger $W^T \cdot W$ and consequently to the smaller margin and vice versa. Obviously taking $C = \infty$ requires that the number of misclassified data is zero and, in the case of an overlapping this is not possible. Hence, the problem may be feasible only for some value $C < \infty$. However, the serious problem with (2.41) is that the error's counting cannot be accommodated within the reliable, well understood, and well developed quadratic programming approach. Also, the

counting only cannot distinguish between huge errors and close misses. The possible solution is to measure the distances ξ_i of the points crossing the margin from the corresponding margin and trade their sum for the margin size as given below:

$$\text{Minimize } \frac{1}{2}\|W\|^2 + C \text{ (sum of distances of the wrong side points)} \tag{2.42}$$

By generalizing the optimal hard margin algorithm, a nonnegative slack variables ξ_i $(i = 1,\ldots,l)$ in the statement of the optimization problem for the overlapped data points is introduced. The separating hyperplane must now satisfy

$$\text{Minimize } \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{l}\xi_i \tag{2.43}$$

subject to

$$y_i(W \cdot X_i + b) \geq 1 - \xi_i, i = 1,\ldots,l,\ \xi_i \geq 0 \tag{2.44}$$

i.e., subject to

$$(W \cdot X_i + b) \geq 1 - \xi_i \text{ for } y_i = +1, \xi_i \geq 0 \tag{2.45}$$

$$(W \cdot X_i + b) \geq -1 + \xi_i, \text{ for } y_i = -1, \xi_i \geq 0 \tag{2.46}$$

Hence, for such a generalized optimal separating hyperplane, the functional to be minimized comprises an extra term accounting the cost of overlapping errors. In fact the cost function (2.43) can be even more general as given below

$$\text{Minimize } \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{l}\xi_i^k \tag{2.47}$$

subject to same constraints. This is a convex programming problem that is usually solved only for $k = 1$ or $k = 2$, and such soft margin SVM are dubbed L1 and L2 SVM respectively. By choosing exponent $k = 1$, neither slack variables ξ_i nor their Lagrange multipliers γ_i appear in a dual Lagrangian L_d. Same as for a linearly separable problem presented previously, for L1 SVM ($k = 1$) here, the solution to a quadratic programming problem is given by the saddle point of the primal Lagrangian $L_p(W, b, \xi, \lambda, \gamma)$ shown below

$$L_p(W,b,\xi,\lambda,\gamma) = \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}\lambda_i\{y_i(W\cdot X_i + b) - 1 + \xi_i\}$$
$$- \sum_{i=1}^{l}\gamma_i\xi_i \text{, for L1 SVM} \qquad (2.48)$$

where λ_i and γ_i are the Lagrange multipliers. Again, we should find an optimal saddle point $(W_0, b_0, \xi_0, \lambda_0, \gamma_0)$ because the Lagrangian L_p has to be minimized with respect to W, b and ξ, and maximized with respect to nonnegative λ_i and γ_i. As before, this problem can be solved in either a primal space or dual space (which is the space of Lagrange multipliers λ_i and γ_i). Again, we consider a solution in a dual space as given below by using standard conditions for an optimum of a constrained function

$$\frac{\partial L}{\partial W_0} = 0, \quad \text{i.e.,} \quad W_0 = \sum_{i=1}^{l}\lambda_i y_i X_i \qquad (2.49)$$

$$\frac{\partial L}{\partial b_0} = 0, \quad \text{i.e.,} \quad \sum_{i=1}^{l}\lambda_i y_i = 0 \qquad (2.50)$$

$$\frac{\partial L}{\partial \xi_{i0}} = 0, \quad \text{i.e.,} \quad \lambda_i + \gamma_i = C \qquad (2.51)$$

and the KKT complementarity conditions below

$$\lambda_i\left[y_i(W\cdot X_i + b) - 1 + \xi_i\right] = 0 \qquad i = 1,\ldots,l \qquad (2.52)$$

$$\gamma_i\xi_i = (C - \lambda_i)\xi_i = 0 \qquad i = 1,\ldots,l \qquad (2.53)$$

At the optimal solution, due to the KKT conditions, the last two terms in the primal Lagrangian L_p given by (2.48) vanish and the dual variables Lagrangian $L_d(\lambda)$, for L1 SVM, is not a function of γ_i. In fact, it is same as the hard margin classifier's L_d given before and repeated here for the soft margin one,

$$L_d(\Lambda) = \sum_{i=1}^{l}\lambda_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}y_i y_j \lambda_i \lambda_j (X_i \cdot X_j) \qquad (2.54)$$

In order to find the optimal hyperplane, a dual Lagrangian $L_d(\lambda)$ has to be maximized with respect to nonnegative and smaller than or equal to C, λ_i. In other words with

$$C \geq \lambda_i \geq 0 \qquad i = 1,\ldots,l \tag{2.55}$$

and under the constraint (2.50), i.e., under

$$\sum_{i=1}^{l} \lambda_i y_i = 0 \tag{2.56}$$

Thus, the final quadratic optimization problem is practically the same as that for the separable case, with the only difference being in the modified bounds of the Lagrange multipliers λ_i. The penalty parameter C, which is now the upper bound on λ_i, is determined by the user. The selection of a good C value is always done experimentally. We can also readily change to the matrix notation of the problem above as in (2.31). Most important of all is that the learning problem is expressed only in terms of unknown Lagrange multipliers λ_i, and known inputs and outputs. Furthermore, optimization does not solely depend upon inputs X_i which can be of a very high (inclusive of an infinite) dimension, but it depends upon a scalar product of input vectors X_i. It is this property we will use in the next section where we design SVM that can create nonlinear separation boundaries. Finally, expressions for both a discriminant function $d(X)$ and an output function $y = sign(d(X))$ for a soft margin classifier are same as for linearly separable classes.

From Eq. 2.52 follows that there are only three possible solutions for λ_i (see Fig. 2.12):

1. $\lambda_i = 0$, $\xi_i = 0$, data point X_i is correctly classified,
2. $C > \lambda_i > 0$ then, the two complementarity conditions must result in $y_i(W \cdot X_i + b) - 1 + \xi_i = 0$, and $\xi_i = 0$. Thus, $y_i(W \cdot X_i + b) = 1$ and X_i is a support vector. The support vectors with $C \geq \lambda_i \geq 0$ are called unbounded or free support vectors. They lie on the two margins,
3. $\lambda_i = C$, then, $y_i(W \cdot X_i + b) - 1 + \xi_i = 0$, and $\xi_i \geq 0$, and X_i is a support vector. The support vectors with $\lambda_i = C$ are called bounded support vectors. They lie on the wrong side of the margin. For $1 > \xi_i \geq 0$, X_i is still correctly classified, and if $\xi_i \geq 1$, X_i is misclassified.

For L2 SVM the second term in the cost function (2.47) is quadratic, i.e., $C\sum_{i=1}^{l}\xi_i^2$, and this leads to changes in a dual optimization problem,

$$L_d(\lambda) = \sum_{i=1}^{l}\lambda_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \lambda_i \lambda_j \left(X_i \cdot X_j + \frac{\delta_{ij}}{C} \right) \tag{2.57}$$

subject to

$$\lambda_i \geq 0,\ i = 1,\ldots,l, \text{ and } \sum_{i=1}^{l}\lambda_i y_i = 0 \tag{2.58}$$

where, $\delta_{ij} = 1$ for $i = j$, and it is zero otherwise. Note the change in Hessian matrix elements given by second terms in (2.57), as well as that there is no longer an upper bound on λ_i. The $1/C$ is added to the diagonal entries of $\mathbf{H}$ ensuring its positive definiteness and stabilizing the solution. We use the most popular L1 SVM here, because they usually produce more sparse solutions, i.e., they create a decision function by using fewer support vectors than the L2 SVM.

2.1.11 The Nonlinear Classifier

The linear classifiers presented previously are very limited. Mostly, real-life classes are not only overlapped but the genuine separation functions are nonlinear hypersurfaces [20, 30]. But a nice characteristic of the approach presented above is that it can be easily extended to create nonlinear decision boundaries. The motivation for such an extension is that an SVM that can create a nonlinear decision hypersurface will be able to classify nonlinearly separable data. This will be achieved by considering a linear classifier in the so-called feature space that will be introduced shortly. A very simple example of a need for designing nonlinear models is given in Fig. 2.13 where the true separation boundary is quadratic. It is obvious that no errorless linear separating hyperplane can be found now. The best linear separation function shown as a solid straight line would make 5 misclassifications (striped data points; 3 in the negative class and 2 in the positive one). Yet, if we use the nonlinear separation boundary we are able to separate two classes without any error. Generally, for n-dimensional input patterns, instead of a nonlinear curve, an SVM will create a nonlinear separating hypersurface.

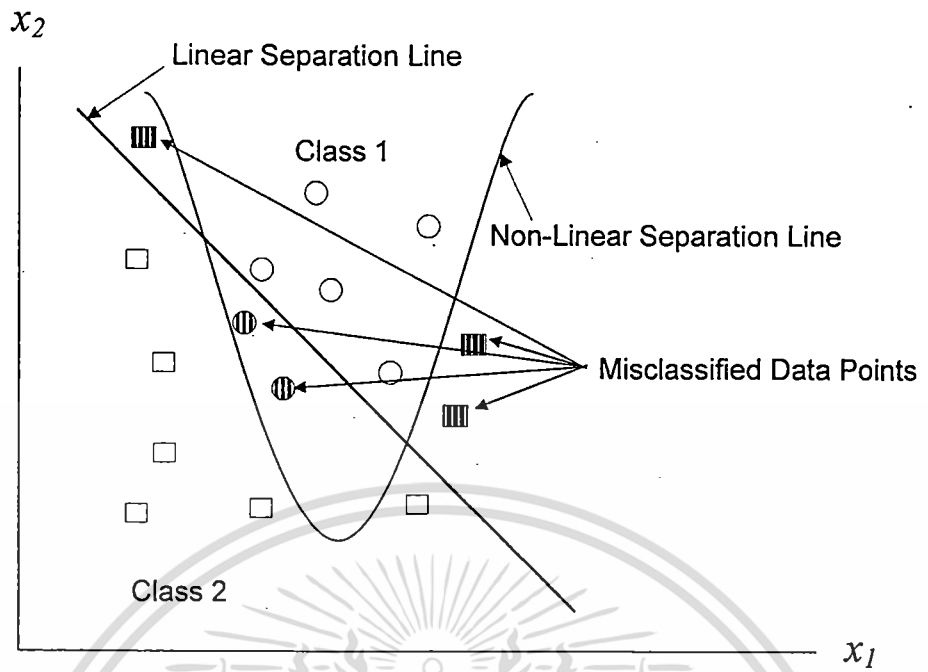


Figure 2.13 A nonlinear SVM without data overlapping. A true separation is a quadratic curve. The nonlinear separation line (curve), the linear one (straight) and data points misclassified by the linear separation line (the striped training data points) are shown. There are 3 misclassified negative data and 2 misclassified positive ones.

The basic idea in designing nonlinear SVM is to map input vectors $X \in \Re^n$ into vectors $\Phi(X)$ of a higher dimensional feature space F (where Φ represents mapping: $\Re^n \to \Re^f$), and to solve a linear classification problem in this feature space

$$X \in \Re^n \to \Phi(X) = [\phi_1(X), \phi_2(X), \ldots, \phi_n(X)] \in \Re^f \tag{2.59}$$

A mapping $\Phi(X)$ is chosen in advance. i.e., it is a fixed function. Note that an input space (X space) is spanned by components X_i of an input vector X and a feature space F (Φ-space) is spanned by components $\phi_i(X)$ of a vector $\Phi(X)$. By performing such a mapping, we hope that in a Φ-space, our learning algorithm will be able to linearly separate images of X by applying the linear SVM formulation presented above. (In fact, it can be shown that for a whole class of mappings the linear separation in a feature space is always possible. Such mappings will correspond to the positive definite kernels that will be shown shortly). We also expect this approach to again lead to solving a quadratic optimization problem with similar constraints in a

Φ-space. The solution for an output function $y = sign(W \cdot \Phi(X) + b) = sign\left(\sum_{i=1}^{l} y_i \lambda_i (\Phi(X_i) \cdot \Phi(X)) + b\right)$, which is a linear classifier in a feature space, will create a nonlinear separating hypersurface in the original input space given by (2.60) below. The equation for an $y(X)$ just given above can be rewritten in a form as follows:

$$y(X) = sign\left(\sum_{i=1}^{l} y_i \lambda_i (\Phi(X_i) \cdot \Phi(X)) + b\right)$$
$$= sign\left(\sum_{i=1}^{l} y_i \lambda_i K(X_i, X) + b\right) \quad = sign\left(\sum_{i=1}^{l} v_i K(X_i, X) + b\right) \qquad (2.60)$$

where v_i corresponds to the output layer weights of the SVM's network and $K(X_i, X)$ denotes the value of the kernel function that will be introduced shortly. (v_i equals $y_i \lambda_i$ in the classification case presented above). Note the difference between the weight vector W whose norm should be minimized and which is the vector of the same dimension as the feature space vector $\Phi(X)$ and the weightings $v_i = \lambda_i y_i$ that are scalar values composing the weight vector V whose dimension equals the number of training data points l. The $(l - N_{SV})$ of v_i components are equal to zero, and only N_{SV} entries of V are nonzero elements.

There is a basic problem when mapping an input X-space into higher order F-space which is that the calculation of the scalar product $\Phi^T(X)\Phi(X)$ can be computationally very discouraging if the number of features f (i.e., dimensionality f of a feature space) is very large. This problem is connected with a phenomenon called the "curse of dimensionality." For example, to construct a decision surface corresponding to a polynomial of degree two in an n-dimensional input space, a dimensionality of a feature space $f = n(n+3)/2$. In other words, a feature space is spanned by f coordinates of the form $z_1 = x_1, \ldots, z_n = x_n$ (n coordinates), $z_{n+1} = (x_1)^2, \ldots, z_{2n} = (x_n)^2$ (next n coordinates), $z_{2n+1} = (x_1 x_2), \ldots, z_f = (x_n x_{n-1})$ ($n(n-1)/2$ coordinates), and the separating hyperplane created in this space, is a second-degree polynomial in the input space [30]. Thus, constructing a polynomial of degree two only, in a 256-dimensional input space, leads to a dimensionality of a feature space f = 33, 152. Performing a scalar product operation with vectors of such, or higher, dimensions, is not a cheap computational task. The problems become serious (and fortunately only seemingly unsolvable) if we want to construct a polynomial of degree 4 or 5 in the same 256-dimensional space leading to the construction of a decision hyperplane in a billion-dimensional feature space.

This explosion in dimensionality can be avoided by noticing that in the quadratic optimization problem given by (2.54), as well as in the final expression for a classifier, training data only appear in the form of scalar products $X_i \cdot X_j$. These products will be replaced by scalar products $\Phi(X) \cdot \Phi(X)_i = [\phi_1(X), \phi_2(X), \ldots, \phi_n(X)] \cdot [\phi_1(X_i), \phi_2(X_i), \ldots, \phi_n(X_i)]$ in a feature space F, and the latter can be and will be expressed by using the kernel function $K(X_i, X_j) = \Phi(X_i) \cdot \Phi(X_j)$.

Note that a kernel function $K(X_i, X_j)$ is a function in input space. Thus, the basic advantage in using kernel function $K(X_i, X_j)$ is in avoiding performing a mapping $\Phi(X)$ at all. Instead, the required scalar products in a feature space $\Phi(X_i) \cdot \Phi(X_j)$, are calculated directly by computing kernels $K(X_i, X_j)$ for given training data vectors in an input space. In this way, we bypass a possibly extremely high dimensionality of a feature space F. Thus, by using the chosen kernel $K(X_i, X_j)$, we can construct an SVM that operates in an infinite dimensional space (such a kernel function is a Gaussian kernel function given in Table 2.1). In addition, as will be shown below, by applying kernels we do not even have to know what the actual mapping $\Phi(X)$ is. A kernel is a function K such that

$$K(X_i, X_j) = \Phi(X_i) \cdot \Phi(X_j) \tag{2.61}$$

There are many possible kernels, and the most popular ones are given in Table 2.1. All of them should fulfill the so-called Mercer's conditions [30]. Mercer's Condition is an existence of a mapping Φ(X) and an expansion of a symmetric kernel function

$$K(X_i, X_j) = \sum_k \Phi_k(X_i) \cdot \Phi_k(X_j) \tag{2.62}$$

iff

$$\iint K(X_i, X_j) g(X_i) g(X_j) dX_i dX_j \geq 0 \tag{2.63}$$

For all g(X) such that

$$\int g^2(X) dX < \infty \tag{2.64}$$

Table 2.1 Popular Admissible Kernel Functions

Kernel Functions	Type of Classifier
$K(X,X_i)=(X^T\cdot X_i)$	Linear dot product
$K(X,X_i)=((X^T\cdot X_i)+1)^d$	Polynomial degree d
$K(X,X_i)=\exp(\|X-X_i\|^2/\sigma^2)$	Gaussian
$K(X,X_i)=\tanh((X^T\cdot X_i)+b)$	Multilayer perceptron
$K(X,X_i)=1/\sqrt{\|X-X_i\|^2+\beta}$	Inverse multiquadric

Finally, we arrive at the point of presenting the learning in nonlinear classifiers in which we are ultimately interested here. The learning algorithm for a nonlinear SVM classifier follows from the design of an optimal separating hyperplane in a feature space. This is the same procedure as the construction of a hard and soft margin classifiers in an X -space previously. In a $\Phi(X)$ -space, the dual Lagrangian, given previously by (2.54) is now

$$L_d(\Lambda)=\sum_{i=1}^{l}\lambda_i-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}y_iy_j\lambda_i\lambda_j\left(\Phi(X_i)\cdot\Phi(X_j)\right) \tag{2.65}$$

and, according to (2.65), by using chosen kernels, we should maximize the following dual Lagrangian

$$L_d(\Lambda)=\sum_{i=1}^{l}\lambda_i-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}y_iy_j\lambda_i\lambda_jK(X_i,X_j) \tag{2.66}$$

subject to

$$\lambda_i\geq 0\,,\ i=1,\ldots,l \text{ and } \sum_{i=1}^{l}\lambda_iy_i=0 \tag{2.67}$$

In a more general case, because of a noise or due to generic class features, there will be an overlapping of training data points. Nothing but constraints for λ_i change. Thus, the nonlinear soft margin classifier will be the solution of the quadratic optimization problem given by (2.66) subject to constraints

$$C\geq\lambda_i\geq 0\,,\ i=1,\ldots,l \text{ and } \sum_{i=1}^{l}\lambda_iy_i=0 \tag{2.68}$$

Again, the only difference to the separable nonlinear classifier is the upper bound C on the Lagrange multipliers λ_i. In this way, we limit the influence of training data points that will remain on the wrong side of a separating nonlinear hypersurface. After the dual variables are calculated, the discriminant hypersurface $d(X)$ is determined by

$$d(X) = \sum_{i=1}^{l} y_i \lambda_i K(X, X_i) + b = \sum_{i=1}^{l} v_i K(X, X_i) + b \tag{2.69}$$

and the output function is

$$y(X) = sign[d(X)] = sign\left[\sum_{i=1}^{l} v_i K(X, X_i) + b\right] \tag{2.70}$$

Note that the summation is not actually performed over all training data but rather over the support vectors, because only for them do the Lagrange multipliers differ from zero. The existence and calculation of a bias b is now not a direct procedure as it is for a linear hyperplane. Depending upon the applied kernel, the bias b can be implicitly part of the kernel function. Same as for the linear SVM, (2.66) can be written in a matrix notation as maximize

$$L_d(\Lambda) = \Lambda \cdot \mathbf{1} - \frac{1}{2} \Lambda^T \mathbf{H} \Lambda \tag{2.71}$$

subject to

$$Y \cdot \Lambda = 0 \tag{2.72}$$

$$C \geq \lambda_i \geq 0 \qquad i = 1, \ldots, l \tag{2.73}$$

where $\Lambda = [\lambda_1, \lambda_2, \ldots, \lambda_l]$, $\mathbf{H}$ denotes the Hessian matrix $\left(H_{ij} = y_i y_j K(X_i, X_j)\right)$ of this problem and $\mathbf{1} = [1\ 1 \ldots 1]$. Note that if $K(X_i, X_j)$ is the positive definite matrix, so is the matrix $y_i y_j K(X_i, X_j)$. Equation (2.70) is the output function for nonlinear SVM classifier which can be used in real-life data sets.

2.2 Existing Rule Extraction Techniques for SVM

2.2.1 Decompositional Techniques

The first decompositional technique proposed was SVM + Prototypes [25]. This decompositional algorithm extracts classification rules from a trained SVM using ellipsoids or rectangles. The 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 calculates 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 an ellipsoid in the input space. The problem to be approached is how to build these ellipsoids for a class from the information provided by a trained SVM. There are two aspects that can be taken into account to perform this task:

1. Ellipsoids fitting: The set of ellipsoids must fit the form of the discriminant function defined by the SVM, exhibiting as low overlapping between classes as possible to avoid problems of multiple instances in the set of rules.
2. Ellipsoids coverage: Ellipsoids should be built to cover as much data as possible with the purpose of producing a compact set of rules.

It is possible to define an ellipsoid associated to an input data set by determining the covariance matrix of the set and finding their eigenvectors and eigenvalues. In this form, a set of axes for an ellipsoid following the directions of greater variance of the data is obtained, the vertices being determined from their own values. The idea underlying in the method is that the ellipsoids should adjust to the form of the limit of decision generated by the SVM. In order to obtain an ellipsoid with these characteristics, its orientation is defined by the support vectors, which are explicitly used for determining the associated set of axes of the ellipsoid and vertices by means of geometric methods.

The algorithm to build the ellipsoid starts by determining the first axis of the ellipsoid as well as the first vertex in this axis. The second vertex along with the first axis is defined. Next, the algorithm enters into a loop in which, both, the axis and its first vertex are determined in each iteration by determining next axis vertex, as well as the second vertex. Finally, all axes and vertices are located, and a single ellipsoid covering all class 1 input data can be formed. The

single ellipsoid will then undergo class overlapping test. If there is an overlap, partitioning of the ellipsoid will be performed resulting in multiple smaller ellipsoids. The partitioning will go on until there is no more class overlapping.

Once the final set of ellipsoids is obtained, the rule equations can be derived from the centre of prototypes generating an equation-type rule with a form:

$$\text{IF } Ax_1^2 + Bx_2^2 + Cx_1x_2 + Dx_1 + Ex_2 + F \leq G \text{ THEN CLASS.}$$

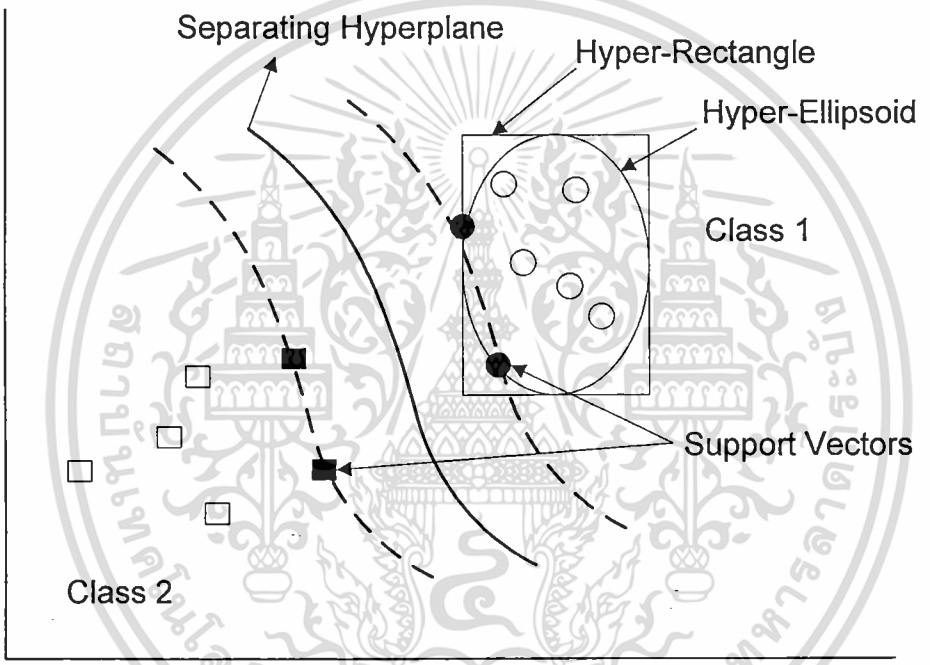


Figure 2.14 Rectangle formed from ellipsoid of positive class in prototype method.

Another form of IF-THEN rules can be obtained by forming a rectangle from the single ellipsoid (see Fig. 2.14). Then a partition test on each of the rectangles is performed (Fig. 2.15). This partition test is performed to minimize the level of overlapping between rectangles for which the predicted class is different. If all subsets are processed, it converts all of the current rectangles into rules. The interval rules have the form:

$$\text{IF } x_1 \in [v_{11}, v_{12}] \wedge x_2 \in [v_{21}, v_{22}] \wedge \ldots \wedge x_m \in [v_{m1}, v_{m2}] \text{ THEN Class.}$$

The interval form is the final form used by this SVM + Prototypes method.

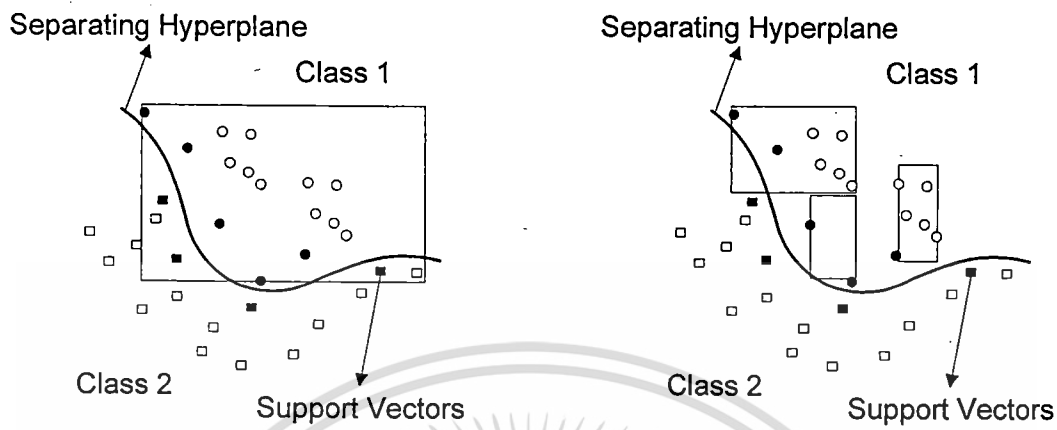


Figure 2.15 Partitioning on a rectangle in prototype method is performed to obtain rectangles without overlapping from negative class.

Another decompositional technique, which involves using cubes beneath separating hyperplane, only works when the input data set is linearly separable [12]. For this technique, 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 (Fig. 2.16). 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. Each final cube generates one rule.

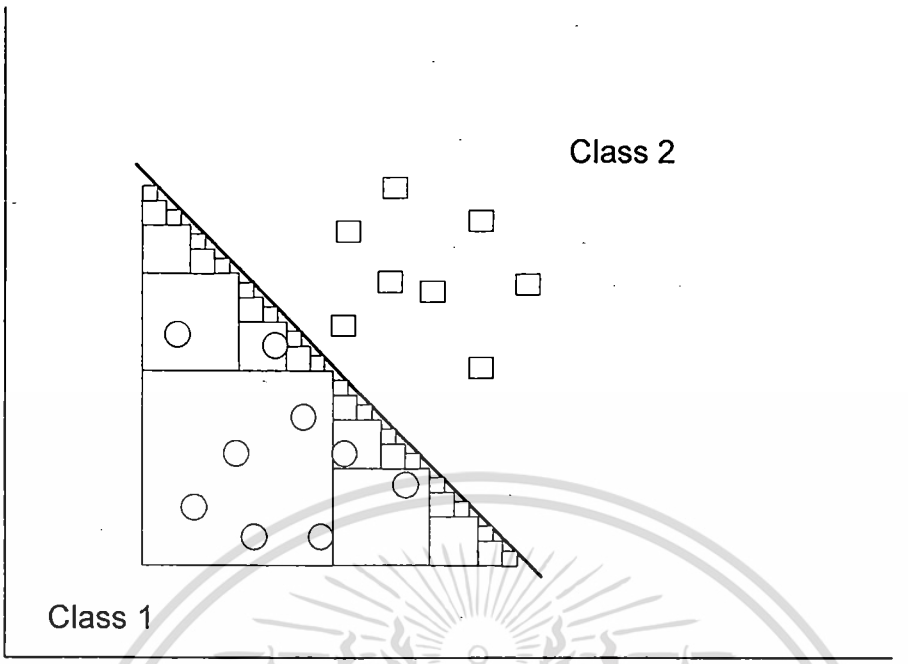


Figure 2.16 Cubes and hyperplane rule extraction method for linear input data is shown.

2.2.2 Pedagogical Techniques

ITER [16] is the first pedagogical method for SVM rule extraction. 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, it calculates how far the cube can be expanded to both extremes of the dimension before it intersects with another cube. These distances are called 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. The difference between each of both means and the mean prediction for the original cube are called LowerDiff and UpperDiff

respectively. Lastly it finds 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.

Minerva [17] 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.

2.2.3 Hybrid between Decompsitional and Pedagogical Techniques

In a hybrid decompositional-pedagogical technigue [5], decision tree is used. 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 classifier with acceptable accuracy and precision. In the second stage of rule generation which is a decompositional part, the objective 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 decision tree which is a learning machine with explanation capability; hence symbolic rules that represent the concepts learned by the SVM model are generated. The training and resulting of the decision tree is considered a pedagogical part.

2.3 Functional Equivalence between Neural Network and Fuzzy System

2.3.1 Fuzzy Logic System

The Fuzzy Logic tool was introduced in 1965 by Lotfi Zadeh [28], and is a mathematical tool for dealing with uncertainty. It offers to a soft computing partnership the important concept of computing with words. It provides a technique to deal with imprecision and information granularity. The fuzzy theory provides a mechanism for representing linguistic constructs such as "many," "low," "medium," "often," "few". In general, the fuzzy logic provides an inference structure that enables appropriate human reasoning capabilities. On the contrary, the traditional binary set theory describes crisp events, events that either do or do not occur. It uses probability theory to explain if an event will occur, measuring the chance with which a given event is expected to occur. The theory of fuzzy logic is based upon the notion of relative graded membership and so are the functions of mentation and cognitive processes. The utility of fuzzy sets lies in their ability to model uncertain or ambiguous data, Fig. 1.17, so often encountered in real life.

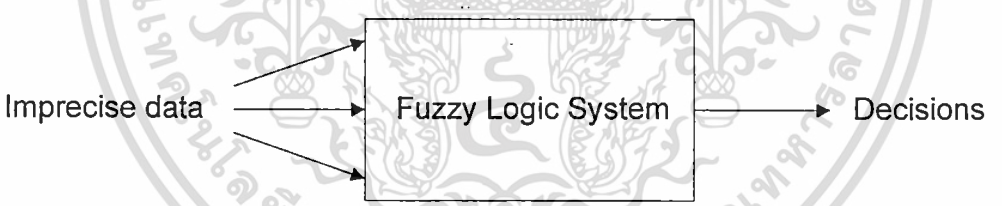


Figure 2.17 A fuzzy logic system which accepts imprecise data such as low, medium, high and provides decisions

Fuzzy rule base (FRB) or fuzzy inference system (FIS) is a system which contains IF-THEN rules [28]. Fuzzy sets form the building blocks for fuzzy IF–THEN rules which have the general form "IF X is A THEN Y is B," where A and B are fuzzy sets. The term "fuzzy systems" refers mostly to systems that are governed by fuzzy IF–THEN rules. The IF part of an implication is called the antecedent whereas the second, THEN part is a consequent. The connectors present in the rule statement are "OR" or "AND" to make the necessary decision rules. A fuzzy system is a set of fuzzy rules that converts inputs to outputs. The basic configuration of a pure fuzzy system

is shown in Fig. 2.18. The fuzzy inference engine (algorithm) combines fuzzy IF–THEN rules into a mapping from fuzzy sets in the input space X to fuzzy sets in the output space Y based on fuzzy logic principles. From a knowledge representation viewpoint, a fuzzy IF–THEN rule is a scheme for capturing knowledge that involves imprecision. The main feature of reasoning using these rules is its partial matching capability, which enables an inference to be made from a fuzzy rule even when the rule's condition is only partially satisfied. The most important part of a fuzzy system is its rules; smart rules give smart systems.

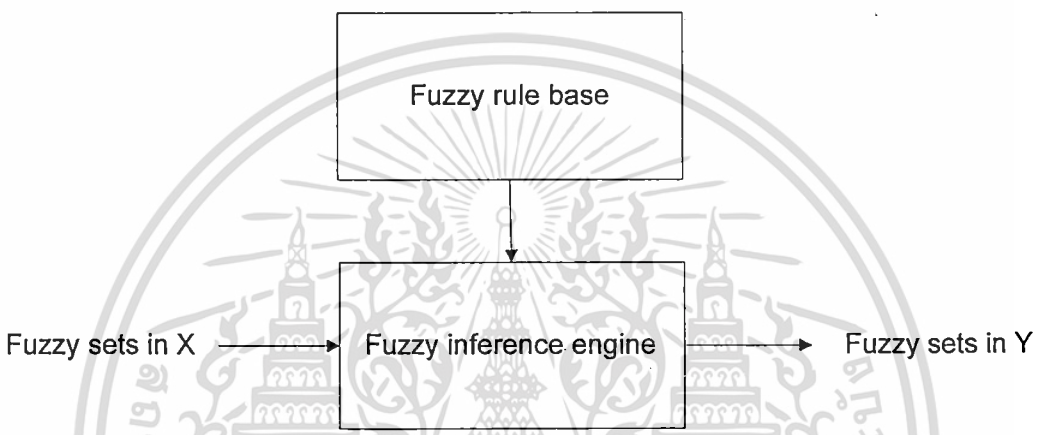


Figure 2.18 Configuration of a pure fuzzy system.

2.3.1.1 Fuzzy Inference Methods

There are two well-known types of fuzzy inference method [19]. Mamdani's fuzzy inference method is the most commonly seen inference method; it was developed by Mamdani in 1975. Another inference method is Takagi–Sugeno method of fuzzy inference process (TS method); this method was introduced by Sugeno in 1985. 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.

2.3.1.2 Mamdani's Fuzzy Inference Method

Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. Mamdani's method was among the first control systems built using fuzzy set theory. It was proposed by Mamdani as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani's effort was based on Zadeh's paper on fuzzy algorithms for complex systems and decision processes. Fuzzy rules are always written in the following form:

> if (input 1 is membership function 1) and/or (input 2 is membership function 2) and/or. . . then (output n is output membership function n)

For example:

> if temperature is high and humidity is high then room is hot.
> There would have to be membership functions that define high temperature (input 1), high humidity (input 2), and a hot room (output 1)

2.3.1.3 Takagi–Sugeno Fuzzy Method

The Takagi-Sugeno (TS) fuzzy model was proposed by Takagi, Sugeno, and Kang in an effort to formalize a system approach to generating fuzzy rules from an input–output data set. Sugeno fuzzy model is also known as TS model. A typical fuzzy rule in a Sugeno fuzzy model has the format

> IF x is A and y is B THEN $z = f(x, y)$

where A and B are fuzzy sets in the antecedent; $z = f(x, y)$ is a crisp function in the consequent. Usually $f(x, y)$ is a polynomial in the input variables x and y , but it can be any other functions that can appropriately describe the output of the output of the system within the fuzzy region specified by the antecedent of the rule. When $f(x, y)$ is a first-order polynomial, we have the first-order Sugeno fuzzy model. When f is a constant, we then have the zero-order Sugeno fuzzy

model, which can be viewed either as a special case of the Mamdani FIS where each rule's consequent is specified by a fuzzy singleton, or a special case of Tsukamoto's fuzzy model where each rule's consequent is specified by a membership function of a step function centered at the constant.

2.3.2 Adaptive Neuro-Fuzzy Inference Systems

Adaptive Neuro-Fuzzy Inference System or Adaptive Network-Based Fuzzy Inference System (ANFIS) [19] is based on Adaptive Network (Fig. 2.19) 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. Functionally, there are almost no constraints on the node functions of the adaptive networks. Adaptive networks can be used to model a wide variety of applications of modeling, decision making, signal processing, and control. An adaptive network which is equivalent to fuzzy rule-based system is ANFIS (Fig. 2.20).

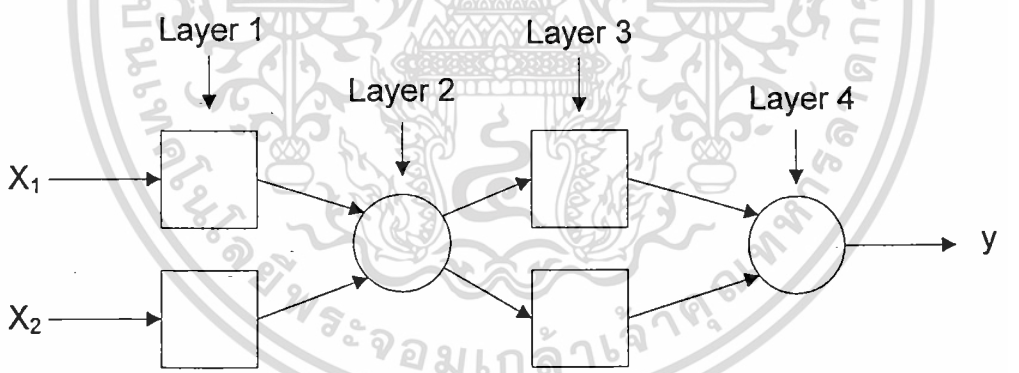


Figure 2.19 Example of Adaptive Network. Square nodes are dynamic nodes with parameterized functions, and circle nodes are fixed function node without parameters.

2.3.2.1 ANFIS Architecture

For simplicity, we assume that the fuzzy inference system under consideration has two inputs x and y and one output z. For a first-order Sugeno fuzzy model, a common rule set with two fuzzy IF-THEN rules is the following:

Rule 1: If x is A_1 and y is B_1, then $f_1 = p_1x + q_1y + r_1$

Rule 2: If x is A_2 and y is B_2, then $f_2 = p_2x + q_2y + r_2$

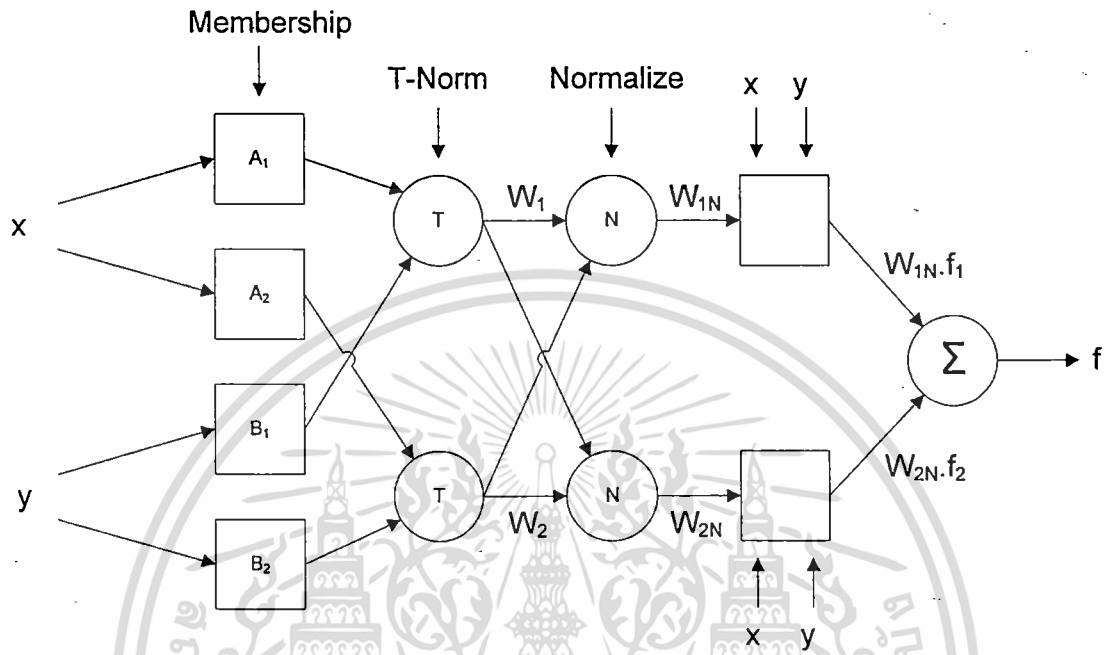


Figure 2.20 Example of ANFIS architecture. ANFIS architecture is as shown where nodes of the same layer have similar functions. Here we denote the output of the i^{th} node in layer l as $O_{l,i}$.

Figure 2.20 is the implementation of the two rules in ANFIS architecture:

Layer 1: Every node i in this layer is an adaptive node with a node function

$$O_{1,i} = \mu A_i(x), \text{ for } i = 1,2, \text{ or}$$
$$O_{1,i} = \mu B_{i-2}(y), \text{ for } i = 3,4 \qquad (2.75)$$

where x (or y) is the input to node i and A_i (or B_{i-2}) is a linguistic label (such as "small" or "large") associated with this node. In other words, O_{li} is the membership grade of a fuzzy set A (A_1, A_2, B_1 or B_2) and it specifies the degree to which the given input x (or y) satisfies the quantifier A. Here the membership function for A can be any appropriate parameterized membership function, such as the Gaussian function:

$$\mu_a(x) = \frac{1}{1+\left|\frac{x-c_i}{a_i}\right|^{b_i}} \tag{2.76}$$

where $\{a_i, b_i, c_i\}$ is the parameter set. As the values of these parameters change, the bell-shaped function varies accordingly, thus exhibiting various forms of membership functions for fuzzy set A. Parameters in this layer are referred to as premise parameters.

Layer 2: Every node in this layer is a fixed node labeled Π, whose output is the product of all the incoming signals:

$$O_{2,i} = w_i = \mu A_i(x)\mu B_i(y) \qquad i = 1,2. \tag{2.77}$$

Each node output represents the firing strength of a rule. In general, any other T-norm operators (fuzzy set intersection operators) that perform fuzzy AND can be used as the node function in this layer.

Layer 3: Every node in this layer is a fixed node labeled N. The i^{th} node calculates the ratio of the i^{th} rule's firing strength to the sum of all rules' firing strengths:

$$O_{3,i} = w_{iN} = \frac{w_i}{w_1 + w_2} \qquad i = 1,2. \tag{2.78}$$

Layer 4: Every node i in this layer is an adaptive node with a node function

$$O_{4,i} = w_{iN} f_i = w_{iN}(p_i x + q_i y + r_i) \qquad i = 1,2. \tag{2.79}$$

where w_{iN} is a normalized firing strength from layer 3 and $\{p_i, q_i, r_i\}$ is the parameter set of this node. Parameters in this layer are referred to as consequent parameters.

Layer 5: The single node in this layer is a fixed node labeled Σ, which computes the overall output as the summation of all incoming signals:

$$\text{Overall output} = O_{5,1} = \sum_{i=1}^{2} w_{iN} f_i = \frac{\sum_{i=1}^{2} w_i f_i}{\sum_{i=1}^{2} w_i} \tag{2.80}$$

Thus we have constructed an adaptive network that is functionally equivalent to a Sugeno fuzzy model. Note that the structure of this adaptive network is not unique; we can combine layers 3 and 4 to obtain an equivalent network with only four layers. Similarly, we can also perform the weight normalization at the last layer. In the extreme case, we can even shrink the whole network into a single adaptive node with the same parameter set. The assignment of node functions and the network configuration are arbitraty, as long as each node and each layer perform meaningful and modular functionalities.

2.3.3 Functional Equivalence between RBFN and ANFIS

Functional Equivalence between RBFN and ANFIS can be proven [18]. RBFN employs local receptive fields to perform function mappings (Fig. 2.21). Typically, receptive field function is chosen as a Gaussian function. Thus the radial basis function computed by a hidden unit is maximal when the input vector is near the center of that unit. The output of an RBFN can be computed in two ways, weighted sum of the function value associated with each receptive field and weighted average of the strengths.

Functional equivalence between an RBFN and a fuzzy inference system can be established if:

1. The number of receptive field units is equal to the number of fuzzy IF-THEN rules.
2. The output of each fuzzy IF-THEN rule is composed of a constant.
3. The membership functions within each rule are chosen as Gaussian functions with the same variance.
4. The T-norm operator used to compute each rule's firing strength is multiplication.
5. Both the RBFN and the fuzzy inference system under consideration use the same method to derive their overall outputs.

We can use Fig. 2.20 to illustrate the equivalence. Layer 1 calculates membership values, layer 2 performs T-norm operation using multiplication operator, layer 3 normalizes layer 2's output, layer 4 updates the weights with $x = y = 0$ to obtain $O_{4,i} = w_{Ni}\left(p_i x + q_i y + r_i\right) = w_i r_i$, and layer 5 sums its inputs to form the overall output. We can conclude that TS FRB is functionally equivalent to RBFN.

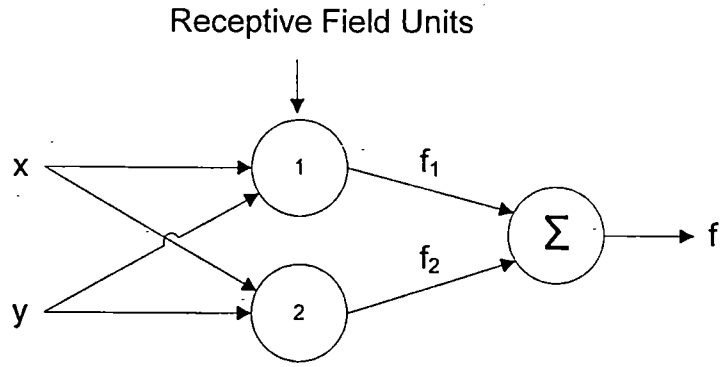


Figure 2.21 An example of Radial Basis Function Network. This RBFN is equivalent to the ANFIS in Fig. 2.20.

Because of the functional equivalence, we can apply what is known about one model to the other, and vice versa. In other words, we can apply the learning rules of RBFN to fuzzy inference systems, and the learning rules of fuzzy inference systems can also be utilized to find the structure and parameters of RBFN's.

CHAPTER 3

Methods

We describe methods in standard SVM and rule extraction in this chapter. Proof of functional equivalence of SVM and ANFIS is also provided. We implement procedures in steps 3.1.1 through 3.1.7 using MATLAB program which is shown together with program flowchart and description in appendix C.

3.1 Obtaining Unbounded Support Vectors through Training

3.1.1 Input Data Normalization

We first define the total number of samples N as the total rows of the raw matrix X_r for each data set. We then define the dimension of the sample pattern n as the total colomns of the matrix X_r. Each raw data point has the form x_{rij}, where i is the row number where $i = 1,2,3,\ldots,N$,and j is the column number where $j = 1,2,3,\ldots,n$. We obtain matrix of the raw input pattern data X_r as

$$\begin{matrix} x_{r11} & x_{r12} & x_{r13} & \cdots & x_{r1n} \\ x_{r21} & x_{r22} & x_{r23} & \cdots & x_{r2n} \\ x_{r31} & x_{r32} & x_{r33} & \cdots & x_{r3n} \\ & & \vdots & & \\ x_{rN1} & x_{rN2} & x_{rN3} & \cdots & x_{rNn} \end{matrix}$$

Each input data column has to be normalized to a range between -1 and $+1$ according to the formula [22]:

$$x_{ij} = \frac{x_{rij} - \min(X_{rj})}{\max(X_{rj}) - \min(X_{rj})} \times (U - L) + L \tag{3.1}$$

where x_{ij} is the normalized scalar value at row i where $i = 1,2,3,\ldots,N$ and column j where $j = 1,2,3,\ldots,n$, x_{rij} is a raw data point at row i and column j, X_{rj} is a raw column vector and U and L are the scalar upper and lower normalization bound. This type of normalization method is used to normalize the data matrix into a desired bound. We change the bound values to between $+1$ and -1, so in our case $U = +1$ and $L = -1$. After normalization, the raw matrix X_r becomes normalized matrix X.

3.1.2 Class Label Data Assignment

For class label data, we assign positive class as $d_i = +1$ and assign negative class as $d_i = -1$, where i is the row number where $i = 1,2,3,\ldots,N$. For multiclassifier, we use one-against-the-rest technique by assigning the positive class as $+1$ and assigning the rest of the classes as -1. For example, there are 3 classes: class 1, class 2, and class 3; we first assign class 1 first as $+1$ and assign both class 2 and class 3 as -1 and perform classification. We then assign class 2 as $+1$ and assign both class 3 and class 1 as -1 and perform classification. Finally we do the same for class 3.

3.1.3 Ten-Fold Cross Validation Data Preparation

We first randomize our normalized input pattern data rows. We then split data into 10 equal subsets to perform tenfold cross validation [6]. If it is not possible to make the tenth subset equal to each of the rest, we make the tenth subset number closest to the number of each of the rest of the subsets. The ten subsets are rotated ten times. In each rotation, one subset is used as an out-of-sample data for testing, and the remaining 9 subsets are used as training data.

3.1.4 Hessian Matrix Preparation

The Hessian matrix $\mathbf{H}$ is formed by the Gaussian kernel of data points multiplied into their respective classes. We prepare the Hessian matrix before doing optimization in later step.

3.1.5 Penalty Value Selection

Select the penalty parameter C to balance the empirical risk and structural risk to get the lowest errors in optimization step. We assign C value as 0.1, 1, 10, 100, 1,000, and 10,000 for

each training and testing of each data set. The result with the least errors from a particular C value is selected as results.

3.1.6 Class Margin Optimization

We do nonlinear class margin optimization using standard quadratic program optimization. The objective function is $\Lambda \cdot \mathbf{1} + (1/2)\Lambda^T \mathbf{H}\Lambda$ with constraints $\Lambda \cdot D = 0$ and $0 < \Lambda < C$. Quadratic optimization with constraints using Lagrange multipliers and KKT conditions is performed. The optimization yields the optimized Lagrange multipliers vector lambda (Λ).

3.1.7 Unbounded Support Vector Retention

We retain the rows with optimized Langrange multipliers Λ values greater than zero but less than C because these are USV that form optimal canonical separating hyperplane. We pick out these rows by using row index of input vector matrix X.

3.2 Construction of Trained SVM Decision Network

Equation (2.70) can be used to construct a trained SVM decision network [22]. We will create FRB in the next step and show that it is functionally equivalent to this SVM decision network in order to make it legitimate that IF-THEN rules can represent SVM decisions. Graphical representation of trained SVM decision network, which can be used to test input data, is shown in Fig. 3.1.

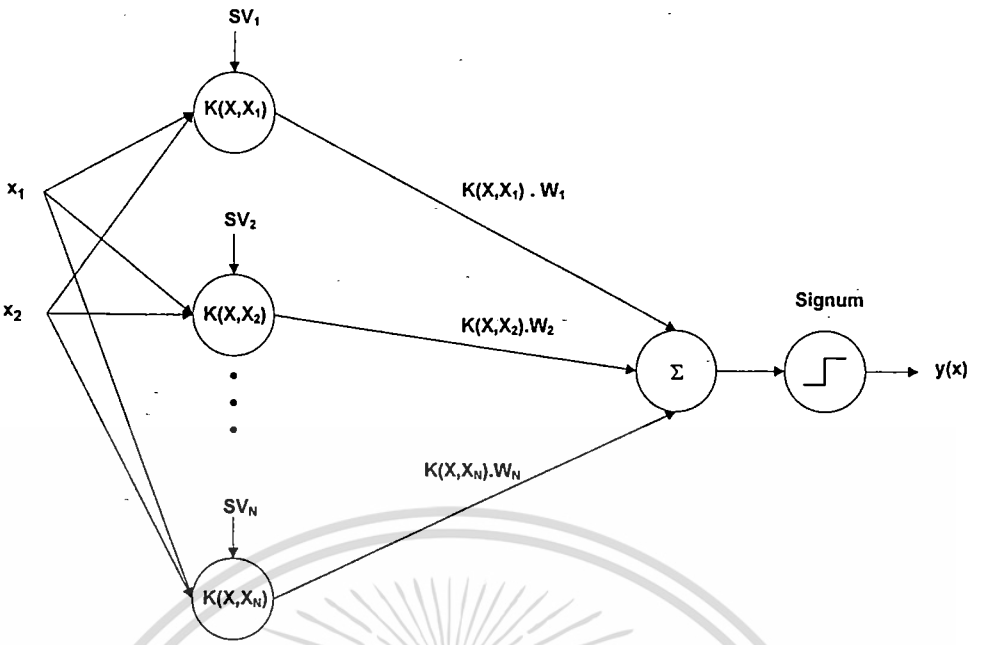


Figure 3.1 An example of SVM decision network structure created from equation (2.70) is illustrated.

3.3 Proof of functional equivalence between SVM decision network and ANFIS

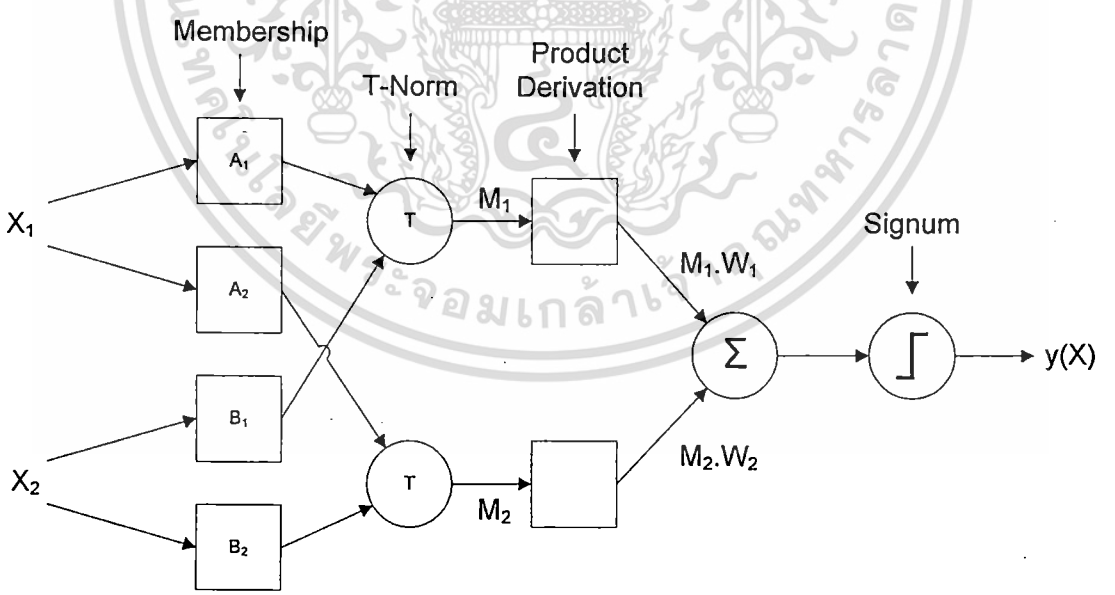


Figure 3.2 ANFIS equivalent to SVM is shown.

Zero-order Sugano 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 Sugano type:

Rule 1: If x_1 is A_1 and x_2 is B_1 then $f_1 = a_1 x_1 + b_1 x_2 + c_1$
Rule 2: If x_1 is A_2 and x_2 is B_2 then $f_2 = a_2 x_1 + b_2 x_2 + c_2$

then the fuzzy reasoning mechanism can be illustrated in Fig. 3.2 where the firing strength of i^{th} rule is obtained as the T-norm (multiplication) of the membership values on the premise part. Strength after T-norm with multiplication operator is:

$$M_i = \mu_{Ai}(x_1)\mu_{Bi}(x_2) \tag{3.2}$$

where $\mu_{Ai}(x_1)$ is membership of x_1 in fuzzy set A at rule i, and $\mu_{Bi}(x_2)$ is membership of x_2 in fuzzy set B at rule i.

Note that overall output can be chosen as the weighted sum of each rule's output:

$$f(X) = \sum_{i=1}^{R} M_i \cdot w_i \tag{3.3}$$

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

$$y(X) = sign[f(X) + b] \tag{3.4}$$

In summary, layer 1 calculates membership values, layer 2 performs T-norm operator, 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. Our mathematical proof of the functional equivalence of SVM decision network and ANFIS is presented next.

Definition 3.1 Functional equivalence of two systems: Two systems, with one system containing a set of functions $F = \{f_i\}$ and the other system containing a set of functions $G = \{g_i\}$ where $i = 1,2,3,\ldots,n$ and $n =$ total number of functions in each system, are said to be functionally equivalent if $domain(f_i) = domain(g_i)$ and for all $x \in domain(f_i)$, $f_i(x) = g_i(x)$.

We want to show that SVM is functional equivalent to ANFIS by assigning functions and parameters to layers of an ANFIS such that

1. The number of unbounded support vectors is equal to the number of fuzzy IF-THEN rules,
2. The output of each fuzzy IF-THEN rule is composed of a constant,
3. The membership functions within each rule are chosen as Gaussian functions with the same variance,
4. The T-norm operator used to compute each rule's firing strength is multiplication,
5. Both the SVM and the fuzzy inference system under consideration use the same method to derive their overall outputs,

and by showing that when inputs to the two systems are the same, the outputs will be the same, as defined in definition 3.1.

Proof.

Functions in SVM decision network

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

$$f_1(X,S_i) = e^{\left[\frac{-(\|X-S_i\|)^2}{\sigma^2}\right]} \qquad i = 1,2,3,\ldots,N \qquad (3.5)$$

Let $f_2(X,w)$ be multiplication function; we obtain

$$f_{2i}(f_1(X,S_i),w_i) = w_i f_i(X,S_i) \qquad i = 1,2,3,\ldots,N \qquad (3.6)$$

Let $f_3(X)$ be summation function of f_2 we obtain

$$f_3(X) = \sum_{i=1}^{N} f_{2i}(X) \qquad (3.7)$$

Let f_4 be signum function of f_3 plus bias, we obtain

$$f_4 = signum(f_3 + b) \qquad (3.8)$$

Let y be the output, we obtain

$$y = f_4(f_3(f_2(f_1(X,S_i)))) = f_4 \circ f_3 \circ f_2 \circ f_1(X,S_i) \qquad (3.9)$$

Functions in ANFIS

Let Q be a set of functions: $Q = \{g_1, g_2, g_3, g_4\}$.

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 m be dimension of each input vector $X_i = \{x_1, x_2, x_3, \ldots, x_m\}$

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 $g_1(x_i, s_{ij})$ be Gaussian membership function with T-norm operation

$$\mu_{ij}(x_i) = e^{\left[\frac{-(\|x_i-s_{ij}\|)^2}{\sigma_i^2}\right]} \qquad i = 1,2,3,\ldots,m \text{ and } j = 1,2,3,\ldots,N \qquad (3.10)$$

where $\mu_{ij}(x_i)$ is membership function

$$M_j = \mu_{1j}(x_1)\mu_{2j}(x_2)\mu_{3j}(x_3)\ldots\mu_{mj}(x_m) \qquad i = 1,2,3,\ldots,m \text{ and } j = 1,2,3,\ldots,N \tag{3.11}$$

where M_j is result of T-norm operation at j, we obtain

$$g_1(X_i) = e^{\left[\frac{-(\|X_i - S_j\|)^2}{\sigma^2}\right]} \qquad i = 1,2,3,\ldots,n \text{ and } j = 1,2,3,\ldots,N \tag{3.12}$$

Let $g_2(g_1, w)$ be multiplication function, we obtain

$$g_{2j}(g_1(X, S_j), w_j) = w_j g_1(X, S_j) \qquad j = 1,2,3,\ldots,N \tag{3.13}$$

Let $g_3 = \sum_{j=1}^{N} g_{2j}$, we obtain

$$g_3 = \sum_{j=1}^{N} g_{2j} \tag{3.14}$$

Let g_4 be signum function of g_3 plus bias, we obtain

$$g_4 = signum(g_3 + b) \tag{3.15}$$

Let z be the output, we obtain

$$z = g_4(g_3(g_2(g_1(X, S_i)))) = g_4 \circ g_3 \circ g_2 \circ g_1(X, S_i) \tag{3.16}$$

Since f_1 is equal to g_1, f_2 is equal to g_2, f_3 is equal to g_3, and f_4 is equal to g_4, the two systems are functionally equivalent□

3.4 Rule Extraction Based on Kernel Function Firing Strength

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

Let X be a set of input vectors of class 1: $X = \{X_1, X_2, X_3, \ldots, X_n\}$ where n is total number of class 1 input vectors. Let S be a set of unbounded support vectors of class 1: $S = \{S_1, S_2, S_3, \ldots, S_N\}$ where N is total number of unbounded support vectors of class 1. In Fig. 3.3, 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.

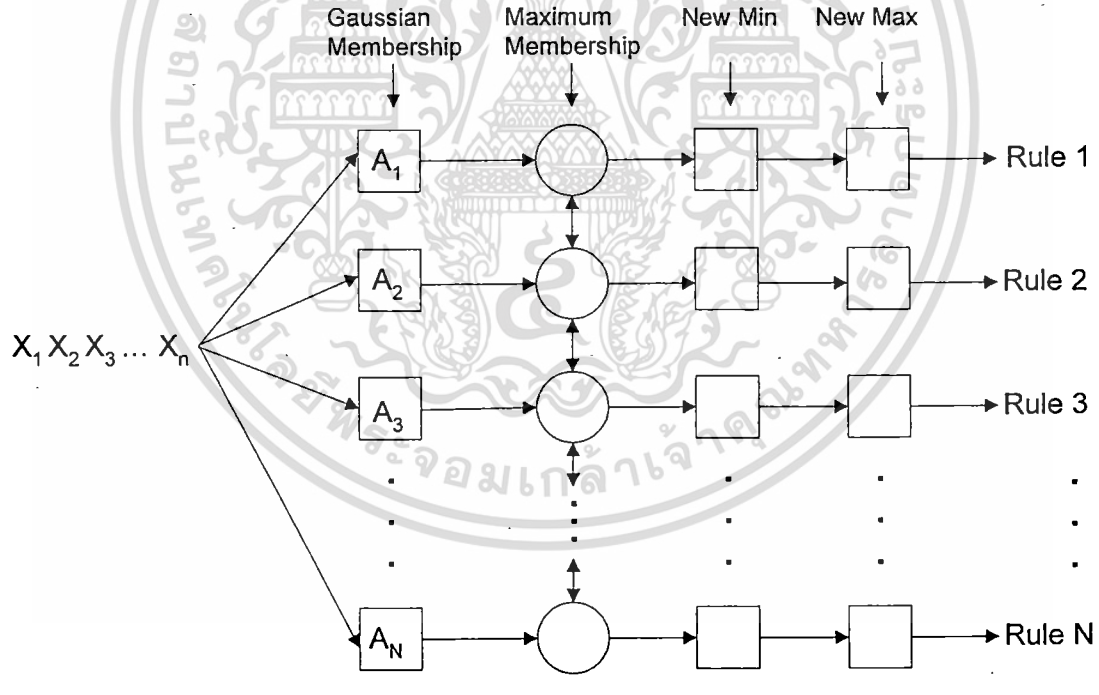


Figure 3.3 Schematic diagram showing ANFIS implementing rule generation from unbounded support vectors found from previous step. X_j is input vector, and A_i is Gaussian kernel function of unbounded support vector and input vector.

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 ...
$(x_{1n} > a_{1n\_}$ 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 ...
$(x_{2n} > a_{2n\_}$ AND $x_{2n} < a_{2n_+})$ THEN $y = v_2$
⋮

Rule i: If $(x_{i1} > a_{i1\_}$ AND $x_{i1} < a_{i1_+})$ AND $(x_{i2} > a_{i2\_}$ AND $x_{i2} < a_{i2_+})$ AND ...
$(x_{in} > a_{in\_}$ AND $x_{in} < a_{in_+})$ THEN $y = v_i$
⋮
Rule N: If $(x_{N1} > a_{N1\_}$ AND $x_{N1} < a_{N1_+})$ AND $(x_{N2} > a_{N2\_}$ AND $x_{N2} < a_{N2_+})$ AND ...
$(x_{Nn} > a_{Nn\_}$ 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.

We can use set membership symbol in place of greater than and less than signs in the form:

$$\text{If } x_{i1} \in [a_{i1\_}, a_{i1_+}] \text{ AND } x_{i2} \in [a_{i2\_}, a_{i2_+}] \text{ AND } \ldots\ x_{1n} \in [a_{in\_}, a_{in_+}] \text{ THEN } y = v_i$$

This format will be used in our final IF-THEN rule results because it is simple and compact.

3.5 Rule Refinement by Input Space Expansion

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. Algorithm's pseudo code for input space expansion is:

[Pre-loop condition: 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 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 WHILE

NEXT i

[Post-loop condition: Number of IF-THEN rules are the same or less than rules in pre-condition]

Flowchart of the rule refinement algorithm is shown is Fig. 3.4.

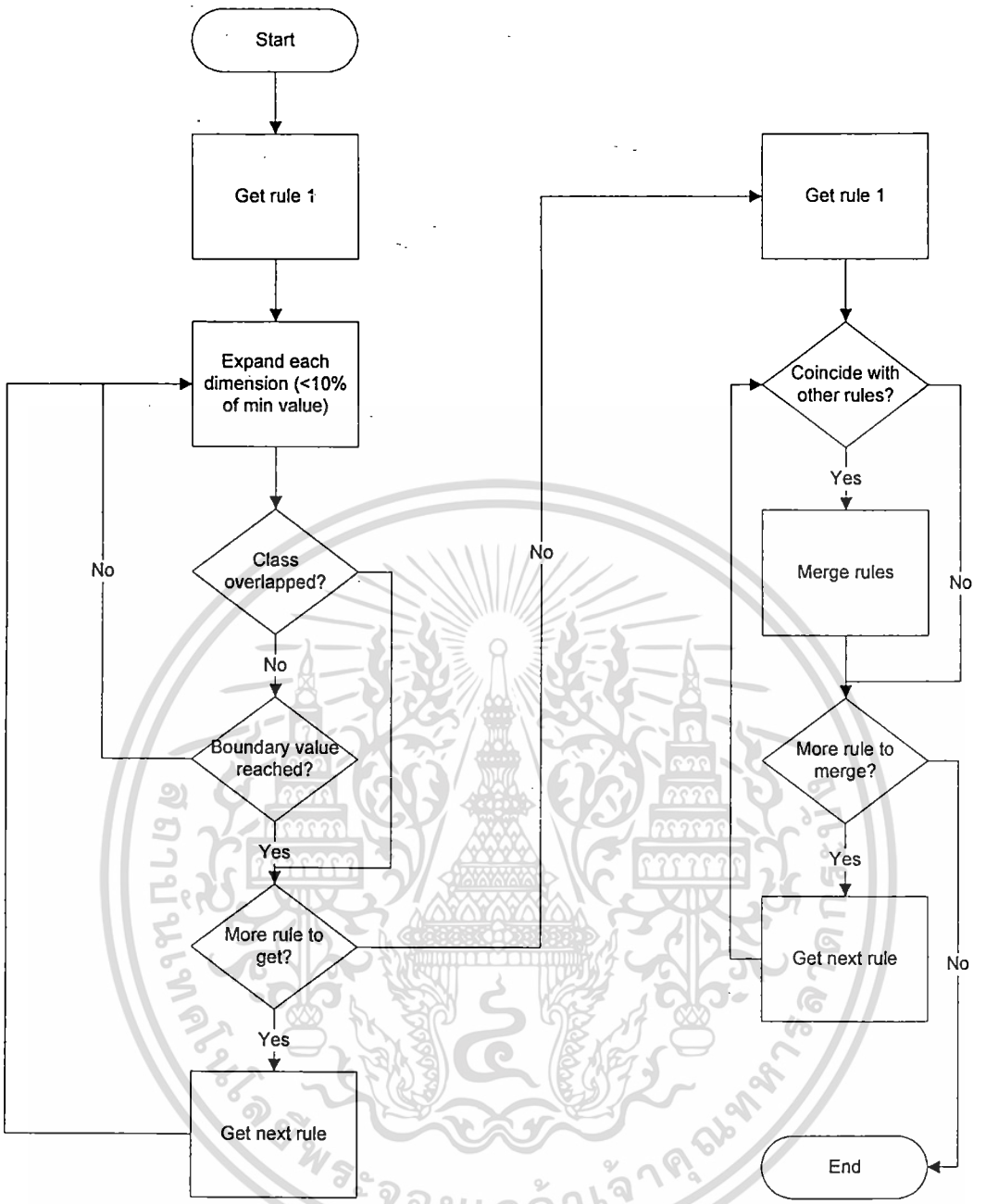


Figure 3.4 Flowchart of rule refinement method.

CHAPTER 4

Experimental Evaluation

4.1 Benchmark Data Sets

We perform classification using both SVM and our fuzzy IF-THEN rules on five well-known and popular benchmark data sets – Iris [11], Wine [1], Wisconsin Breast Cancer [31], Haberman's Survival Data [13], and Ionosphere [27]. 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 [6] is used in each data set. The results we get from each fold of standard SVM testings are number of unbounded support vectors (USV) and percent classification error (%Error). We sum the USV and %Error from all ten folds and average the values by dividing by ten. Detailed information about these benchmark data sets can be found at UCI Machine Learning Repository web site (http://archive.ics.uci.edu/ml/datasets.html).

The results we get from each fold of our fuzzy IF-THEN rule testing are number of rules and percent classification error. We sum the number of rules and %Error from all ten folds and average the values by dividing by ten the same as what we did with the standard SVM.

4.2 Experimental Results for Benchmark Datasets

Results obtained from each data set are average number of unbounded support vectors (USV), average percent error from SVM, average number of rules from our method, and average percent error from our method, and the results are shown in Table 4.1.

SVM performs classification with less error in Iris and Haberman data sets than our method, but our method performs better in Wine, Wisconsin Breast Cancer, and Ionosphere data sets. Average number of rules in each data set is lower than number of unsigned support vectors as claimed by our method.

Table 4.1 Comparison of errors from SVM and Rules from benchmark data sets.

Data Characteristics					SVM		Rules	
Data Set	**Instances**	**Classes**	**Type**	**Attributes**	**USV**	**%Err**	**Rules**	**%Err**
Iris	150	3	Real	4	37.07	2.89	7.27	6.22
Wine	178	3	Real	13	157.57	14.04	52.53	8.99
Wisconsin	699	2	Int	10	300.6	5.27	67.5	4.83
Haberman	306	2	Int	4	105.7	29.08	63.7	46.41
Ionosphere	351	2	Int, Real	34	278	37.04	165.4	6.84

4.3 Non-benchmark Data Sets

There have been studies on models of currency crises since the 1970's. Three models have been widely accepted to explain different currency crises occurred around the world [10, 23, 26, 29]. The first model is based on the study of Krugman in 1979; it is used to explain currency crises in some countries in Latin America in 1970's to 1980's. The second model is based on the study of Obstfeld in 1994; it is used to explain currency crises in some countries in Europe in 1992 and Mexico in 1994. The third model is used to explain currency crises in some countries in Asia in 1997 to 1998. Many indicators and indices are believed to be statistically linked to these models. Through these variables, researchers have used many techniques to come up with the most accurate detection systems. These techniques involve traditional statistical methods [29] or more recently Artificial Neural Network (ANN) [23]. Statistical methods commonly used in financial forecasting are logit and probit. ANN commonly used in many applications are Feed Forward Network with Back Propagation (BP) and more recently Support Vector Machine (SVM) [4, 15].

Either ANN or SVM does not reveal clearly to human comprehension how it makes decision which is known as black-boxed characteristic. There is a need to have fuzzy if-then rules for the decision in order to make human expert understand the decision, so we will extract how SVM makes each decision whether it is a currency crisis or a non crisis in this study.

This study uses SVM classifier with data from countries in Latin America and Asia [29] to detect currency crises during 1980 to 2002. The data used are 11 explanatory variables which

have been shown in the past to be significantly associated with the occurrences of currency crises.

4.3.1 Explanatory Variables for Currency Crises

The 11 explanatory variables are the following:

1. Overvaluation of real exchange rate (OVERRER)
2. Liquid liabilities of monetary authority and commercial banks to foreign reserves (M2_FR)
3. Dummy for capital liberalization (LIBDUM_FORI)
4. Dummy for the ratio of short-term debts to foreign reserves (DUMST_FR)
5. Ratio of external debt to Gross National Income (EXD_GNI)
6. Current account as a percentage of Gross Domestic Product (CA_GDP)
7. Domestic credit growth (CREGROW)
8. Growth rate of Gross Domestic Product (GDPGROW)
9. Lending boom as a percentage change of ratio of financial system claims on private sector relative to Gross Domestic Product over four-year period (LB)
10. Capital inflows reversal (KAREVS)
11. Dummy for regional contagion effect (CONT)

4.3.2 Data Sources for Currency Crises

Data Sources for Currency Crises are obtained from the following:

1. International Financial Statistics CD-ROM (IFS), IMF
2. Direction of Trade Statistics Year Books and CD-ROM (DOTS), IMF
3. Balance of Payment Statistics Year Books (BOP), IMF
4. World Development Indicators CD-ROM (WDI), The World Bank
5. Global Development Finance CD-ROM (GDF), The World Bank
6. Exchange Arrangements and Exchange Restrictions Annual Report (EAER), IMF
7. Consolidated Banking Statistics (CBS), Bank for International Settlements (BIS)
8. Joint BIS-IMF-OECD-World Bank Statistics on External Debt (Joint-SED)

9. Die Falligkeitsverteilung der Internationalen Bankausleihung (FIB), BIS
10. Key indicators of developing Asian and Pacific countries, Asian Development Bank (ADB)

4.3.3 Selected Countries

Countries selected for currency crises study are the following: Argentina (Arg), Bolivia (Bol), Chile (Chi), Ecuador (Ecu), Mexico (Mex), Paraguay (Par), Peru (Per), Uruguay (Uru), Venezuela (Ven), India (India), Indonesia (Indo), Korea (Kor), Malaysia (Mal), Pakistan (Pak), Philippines (Phi), Singapore(Sin), Sri Lanka (Slk), and Thailand (Thai)

4.3.4 Currency Data Preparation

This study follows the empirical implementation proposed by Esquivel and Larrain in 1998 [10]. The crises are required to be at least five months apart, i.e., a country could have a maximum of two crises per year. Most of the explanatory variables entered in lagged form due to the purpose of the study to interpret the empirical results as the one-period-ahead probability of a currency crisis. The explanatory variables can be classified into two types. Firstly, the stock variables, whose units are measured at one point in time and are observed every month. The variables in this first group are OVERRER, M2_FR, LIBDUM*FORI, DUMST_FR, and EXD_GNI. Secondly, the flow variables, whose units are measured per unit of time, are observed, in this case, on yearly basis. The variables in this group are CA_GDP, CREGROW, GDPGROW, LB, KAREVS, and CONT. All variables are entered in lagged form for one period except CONT which is contemporaneous. When a crisis occurs late in year t and one tries to use the explanatory variables of year t-1 to explain this crisis, sometimes many of explanatory variables change abruptly in the months before the collapse, and there was real evidence from changes in some of the explanatory variables, e.g. RER and foreign reserves, only a few months before the collapse. With respect to the stock variables if the crisis occurs late in year t (period "b" of year t), we should assume that the crisis occurred early in year t+1, and instead of taking the year-end value as usual, we take the mid-year value of year t to explain the crisis occurring in this particular

period. This adjustment is made only for the stock explanatory variables. Esquivel and Larrain suggest that we should consider the characteristics of the flow variable, which indicates the change of a variable during one period in time, e.g., changing of the CA_GDP in year t. For these flow variables the year end value of year t-1 is to explain the crisis occurs in year t, even if it takes place in period "b" of year t. This adjustment should provide more accuracy for the study, which attempts to estimate the impact of the explanatory variables on the one-period-ahead probability of crisis on yearly basis.

4.4 Experimental Results for Non-Benchmark Datasets

We perform classification using both SVM and our fuzzy if-then rules on Latin America compared to Asia data sets. Ten-fold cross validation technique is used in each data set to evaluate effectiveness of the classification.

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 4.2. SVM performs classification with less error in Asia data sets than our method, but our method performs better in Latin America data sets.

Table 4.2 Comparison of errors from SVM and Rules for currency data set

Data Set	SVM		SVSE Rules	
	Avg USV	%Error	Avg Rules	%Error
Latin	33.90	29.47	33.90	28.02
Asia	14.70	10.63	14.70	15.46

The final IF-THEN rules obtained by our rule extraction method for Latin America entire data set, and the final IF-THEN rules for Asia entire data set are shown in appendix B.

4.5 Discussion

For the first benchmark data set, Iris data set, the average number of USV is 37.07 and the average number of rules is 7.27 which is much lower. The percent error of standard SVM is 2.89 and the percent error of fuzzy IF-THEN rules is 6.22 which is slightly higher. This data set is a low noise data set with noise mostly locating on the positive class since there are more errors by IF-THEN rules which uses only positive class vectors in its decisions.

In Wine data set, the average number of USV is 157.57 and the average number of rules is 52.53 which is much lower. The percent error of standard SVM is 14.04 and the percent error of fuzzy IF-THEN rules is 8.99 which is slightly lower. This data set is a low noise data set with noise mostly locating on the negative class since there are more errors by standard SVM which uses both positive and negative class vectors in its decisions.

In Wisconsin data set, the average number of USV is 300.6 and the average number of rules is 67.5 which is much lower. The percent error of standard SVM is 5.27 and the percent error of fuzzy IF-THEN rules is 4.83 which is slightly lower. This data set is a low noise data set with noise locating slightly more on the negative class since there are more errors by standard SVM.

In Haberman data set, the average number of USV is 105.7 and the average number of rules is 63.7 which is much lower. The percent error of standard SVM is 29.08 and the percent error of fuzzy IF-THEN rules is 46.41 which is slightly higher. This data set is a high noise data set with noise mostly locating on the positive class since there are more errors by IF-THEN rules.

For the last benchmark data set, Ionosphere data set, the average number of USV is 278 and the average number of rules is 165.4 which is much lower. The percent error of standard SVM is 37.04 and the percent error of fuzzy IF-THEN rules is 6.84 which is much lower. This data set is a high noise data set with noise mostly locating on the negative class since there are more errors by standard SVM.

For the first non-benchmark data set, Latin data set, the average number of USV is 33.90 and the average number of rules is 33.90 which is exactly the same. This means the data patterns are so complex that there are no coincided rules to merge. The percent error of standard SVM is 29.47 and the percent error of fuzzy IF-THEN rules is 28.02 which is slightly lower. This data set is a high noise data set with noise locating on both the positive and negative class since the errors by standard SVM and IF-THEN rules are very close.

For the last non-benchmark data set, Asia data set, the average number of USV is 14.70 and the average number of rules is 14.70 which is exactly the same. This means the data patterns are also very complex. The percent error of standard SVM is 10.63 and the percent error of fuzzy IF-THEN rules is 15.46 which is slightly higher. This data set is a low noise data set with noise mostly locating on the positive class since there are more errors by IF-THEN rules which uses only positive class vectors in its decisions.

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 positive class and negative class to make classification decisions, but our method uses only USV from positive class. If there is more misclassification vectors in positive class region in hyperspace, our method performs worse than SVM. But if there are more misclassification vectors in negative class region in hyperspace, SVM performs worse than our method.

Another reason the errors are different between SVM and our method is because our method generates rules with min and max up to the limit of input data ranges from the training set. When we test a vector from a testing set, if the vector happens to fall outside these min and max limits, then the vector will be classified as a negative class. While in the same condition, SVM can handle the out-of-range vector (unseen data which are out-of-range of training data) and may potentially classified the vector correctly because SVM calculates sum of the kernel firing strength and uses signum function to make final decision.

Despite the fact that there are differences in errors between the two methods, SVM and our method seem to perform classifications with comparable percent errors across data sets except for Ionosphere, where our method seems to perform much more accurately than SVM. Although the consistency in classification errors from these results shows that our method can extract rules from SVM decisions effectively for the data sets studied, we need to keep in mind that this consistency depends on the characteristics of input data.

CHAPTER 5

Conclusion and Recommendation

5.1 Conclusion

Proof has been provided by us to show that fuzzy IF-THEN rules can actually represent the hidden decision making of SVM. Fuzzy IF-THEN rules are considered white-boxed because human experts can see and understand the IF-THEN rules better than the SVM decision algorithm.

The proposed rule extraction method by us is shown to be a good alternative method for classification application similar to SVM but has an advantage of 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. 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. Moreover, our method makes use of kernel function firing strength similar to SVM, and this makes it potentially closer to SVM's decision method than previous studies.

The results of our benchmark and non-benchmark 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.

5.2 Recommendation

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.

Another suggestion for future study is to use data sets from other applications in order to obtain useful human comprehensible rules extracted from the pattern classification by SVM in different expert domains.

An interesting study to do is to use both positive class and negative class in rule extraction. Input vectors and unbounded support vectors from positive class can be used to extract positive class rules exactly like in this thesis. Input vectors and unbounded support vectors from negative class can be included to extract negative class rules. This may give insights to human experts in both positive and negative IF-THEN rules.

REFERENCES

[1] S. Aeberhard, D. Coomans, and O. de Vel, The Classification Performance of RDA, **Tech. Rep.** no. 92-01, 1992.

[2] S. Ali, K. Smith-Miles, Improved Support Vector Machine Generalization Using Normalized Input Space, In: A. Sattar and B.H. Kang (Eds.), **AI** 2006, 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] I. Arciniegas, SVM Sensitivity Analysis: An Application to Currency Crises Aftermaths, **IEEE Transactions on Systems, Man, and Cybernetics** – Part A: Systems and Humans, Vol. 34, No. 3, pp. 387-398, 2004.

[5] 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.

[6] C. Bishop, **Neural Networks for Pattern Recognition**, Oxford University Press, New York, 1995, pp 372 – 375.

[7] C. Burges, A tutorial on support vector machines for pattern recognition, **Data Mining and Knowledge Discovery**, Vol. 2, pp. 121-167, 1998.

[8] C. Cortes and V. Vapnik, Support Vector Networks, **Machine Learning**, Vol. 20, pp. 273-297, 1995.

[9] J. Diederich (Ed.), Rule Extraction from Support Vector Machines Studies in **Computational Intelligence**, Vol. 80, Springer-Verlag, Berlin, Heidelberg, 2008.

[10] G.Esquivel and F.Larrain, Explaining Currency Crisis, **http://www.cid.harvard.edu/hiid/646.pdf**, pp. 1-42, 1998.

[11] R. Fisher, The use of multiple measurements in taxonomic problem, **Annual Eugenics**, 7, Part II, 179-188 (1936); also in **Contributions to Mathematical Statistics**, John Wiley, NY, 1950.

[12] G. Fung, S. Sandilya, and R. Rao, Rule extraction from linear support vector machines, In **Proceedings of the 11th ACM SIGKDD international Conference on Knowledge Discovery in Data Mining**, pp. 32–40, 2005.

[13] S. Haberman, Generalized Residuals for Log-Linear Models, **Proceedings of the 9th International Biometrics Conference**, Boston, pp. 104-122, 1976.

[14] S. Haykin, **Neural Networks, A Comprehensive Foundation**, Macmillan, New York, NY, 1994.

[15] X. Hui and J. Sun. An Application of Support Vector Machine to Companies' Financial Distress Prediction, V. Torra et al. (Eds.): **MDAI** 2006, LNAI 3885, pp. 274 – 282, 2006.

[16] 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, pp. 270-279, 2006.

[17] 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.

[18] J. Jang and C. Sun, Functional equivalence between radial basis function networks and fuzzy inference systems, **IEEE Trans. Neural Networks**, Vol. 4, pp. 156–158, 1992.

[19] J. Jang, C. Sun, and E. Mizutani, **Neuro-Fuzzy and Soft Computing**, Prentice Hall International, 1997.

[20] V. Kecman, Support Vector Machines – An Introduction, in **Support Vector Machines: Theory and Applications**, Lipo Wang (Ed.), Springer Verlag, Berlin, Heidelberg, 2005.

[21] E. Kolman and M. Margaliot, Are artificial neural networks white boxes? **IEEE Trans. Neural Networks**, Vol. 16, No. 4, pp. 844–852, 2005.

[22] S. Kumar, **Neural Networks: A Classroom Approach**, McGraw-Hill, International Edition, 2005.

[23] C. Lin, H. Khan, Y. Wang, and R. Chang, A New Approach to Modeling Early Warning Systems for Currency Crises: can a machine-learning fuzzy expert system predict the currency crises effectively? **CIRJE**-F-411, 2006.

[24] K. Muller, A. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen and V. Vapnik, Predicting time series with support vector machines, In: **ICANN** 1997, pp. 999-1004.

[25] H. Nunez et al., Rule Extraction Based on Support and Prototype Vectors, In: **Studies in Computational Intelligence**, Vol. 80, Springer Verlag, Berlin, Heidelberg, pp. 109-134, 2008.

[26] T. Peltonen, Are Emerging Market Currency Crises Predictable? A Test. **European Central Bank Working Paper,** No. 571, 2006.

[27] V. Sigillito, S. Wing, L. Hutton, and K. Baker, Classification of radar returns from the ionosphere using neural networks. **Johns Hopkins APL Technical Digest**, Vol. 10, pp. 262-266, 1989.

[28] S. Sivanandam, S. Sumathi and S. Deepa, **Introduction to Fuzzy Logic using MATLAB**, Springer Verlag, Berlin, Heidelberg, 2007.

[29] S. Thearpiriyakij, J. Breitung, An Empirical Approach to Currency Crises: Latin America and Asia. **Diplomabeit, Rheinisch-Friedrich-Wilhelms-Universitat**, Bonn, Jun 2005.

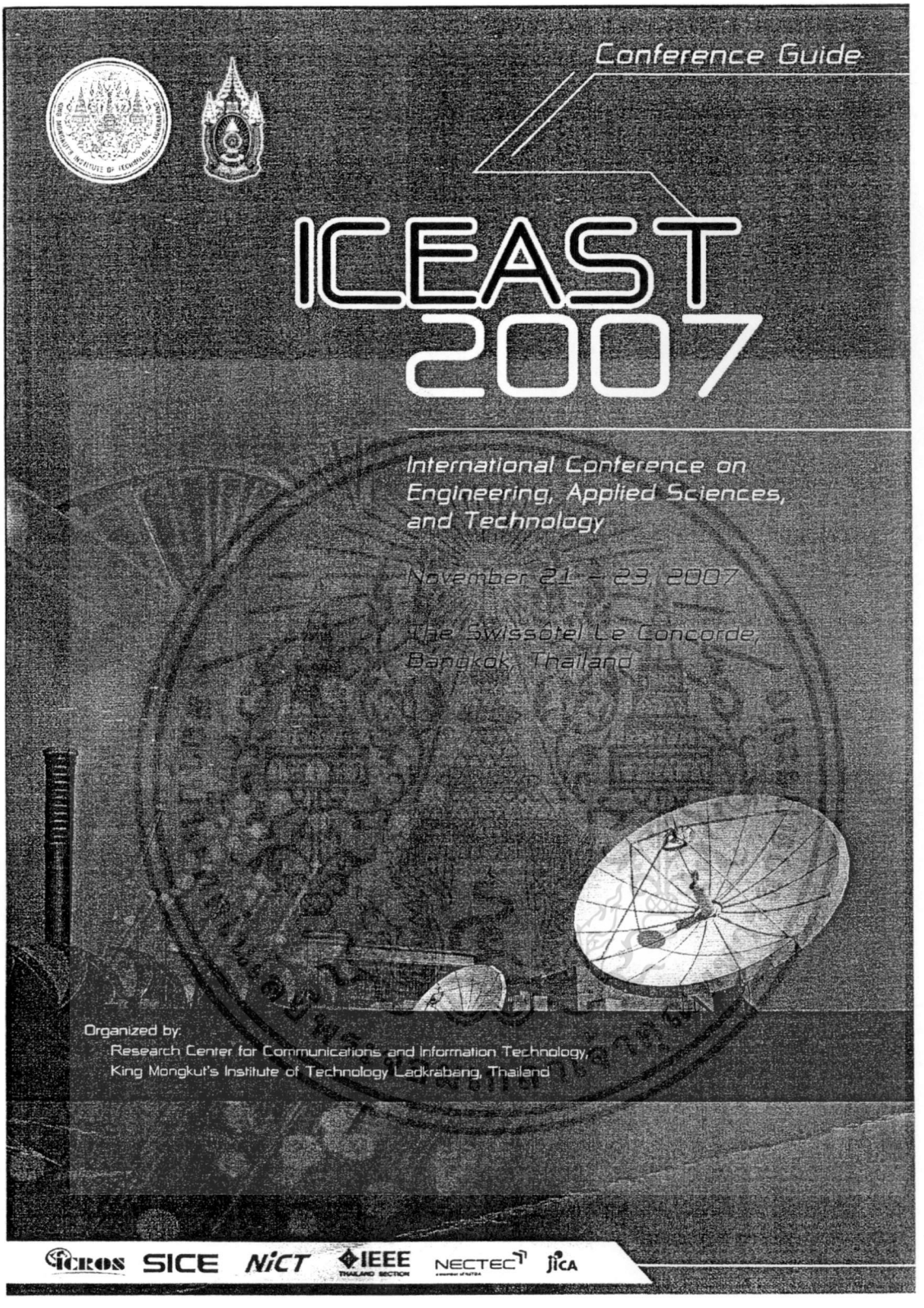
[30] V. Vapnik, **Statistical Learning Theory**, John Wiley&Sons, 1998.

[31] W. Wolberg and O. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, **Proceedings of the National Academy of Sciences**, U.S.A., Volume 87, pp. 9193-9196, December 1990.

APPENDIX A

Publications

1. N. Benjathepanun and P. Pitiranggon, Early Warning System for Currency Crises using Support Vector Machine: Empirical Study on Latin America and Asia, In **Proc. of the International Conference on Engineering, Applied Sciences, and Technology** (ICEAST 2007), pp. 877-880, 2007.
2. P. Pitiranggon, S. Banditvilai, and N. Benjathepanun, Detection of Currency Crises by a Novel Rule Extraction Method from Support Vector Machine, **International Journal of Mathematical Models and Methods in Applied Sciences**, Vol. 4, No. 3, pp. 141-149, 2010.
3. P. Pitiranggon, S. Banditvilai, N. Benjathepanun, and V. Boonjing, Rule Extraction for Support Vector Machine Using Input Space Expansion, In **Proc. of the 3rd Asian Conference on Intelligent Information and Database Systems** (ACIIDS 2011), LNAI 6592, pp. 100-109, 2011, Springer Verlag Berlin Heidelberg 2011.



Early Warning System for Currency Crises using Support Vector Machine: Empirical Study on Latin America and Asia

N. Benjathepanun[1], P. Pitiranggon[2]
[1] Faculty of Science, King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand
[2] Ph.D. Student of Computer Science, Faculty of Science, KMITL, Thailand

Abstract—**Support Vector Machine (SVM) is used with explanatory variables known to be associated with currency crises to predict occurrences of currency crises in 9 countries in Latin America and 9 countries in Asia from 1991 to 2002 based on data from 1980 to 2002. The overall results of out-of-sample prediction are considered satisfactory, and the prediction system may be used as an early warning system for currency crises for countries in this study. SVM, applied to currency crises, is shown to be a good alternative method to statistical methods, e.g., logit and probit, or artificial neural networks (ANN) with back propagation (BP) in financial forecast application.**

Keywords—**Support Vector Machine, currency crises, Latin America, Asia**

I. INTRODUCTION

Currency crises have devastating effects on the country economy. It would be beneficial to have a warning system before the real crisis strikes. In order to predict currency crises, nature of occurrences of crises must be understood. There have been studies in this field since the 1970's. Three models have been widely accepted to explain different currency crises occurred around the world [1]. The first model is based on the study of Krugman in 1979; it is used to explain currency crises in some countries in Latin America in 1970's to 1980's. The second model is based on the study of Obstfeld in 1994; it is used to explain currency crises in some countries in Europe in 1992 and Mexico in 1994. The third model is used to explain currency crises in some countries in Asia in 1997 to 1998. Many indicators and indices are believed to be statistically linked to these models. Through these variables, researchers have used many techniques to come up with the most accurate prediction systems. These techniques involve traditional statistical methods [1] or more recently Artificial Neural Network (ANN)[2],[3]. Statistical methods commonly used in financial forecasting are logit and probit. ANN commonly used in many applications are Feed Forward Network with Back Propagation (BP) and more recently Support Vector Machine (SVM)[4]-[6]. BP has been shown to outperform logit and probit in in-sample and out-of-sample prediction in financial applications. SVM has been shown to outperform BP in out-of-sample classification in many applications.

SVM is a state-of-the-art non-linear function classifier and regression estimator originated by Vapnik in 1995 [7]. SVM is similar to Feed Forward ANN, but generalize better. SVM looks for a separating hyperplane which guarantees a maximized margin between two classes of data in high dimensional space in classification application which is the method used in this study. The SVM is based on Statistical Learning Theory and involves a balance of minimization between Empirical Risk and Vapnik-Chervonenkis (VC) Confidence. SVM looks for data points close to the separating hyperplane and designate them support vectors. These support vectors will be used to shape the hyperplane which is the plane separating the two classes, and in our case, the two classes are "currency crisis" and "tranquility." Fundamentals of Statistical Learning Theory leading to SVM can be found in many literatures [7]-[10].

A previous study on currency crises prediction [2] using ANN shows a good result for both in-sample and out-of-sample predictions. In contrast, another study [3] using ANN with a design to really test whether currency crises are predictable or not shows that out-of-sample prediction is weak.

Our study, inspired by previous study using probit and logit with the same dataset by Thearpiriyakij [1], uses SVM classifier with training and testing data from countries in Latin America and Asia from 1979 to 2001 to predict currency crises during 1980 to 2002. The data used are 11 explanatory variables which have been shown in the past to be significantly associated with the occurrences of currency crises. SVM is implemented using MATLAB version 6.1. Kernel method is third order polynomial function. Significance of each explanatory variable contributed in the pattern recognition is studied by leave-one-out method.

-The 11 explanatory variables are:

1. OVERRER (V1): Overvaluation of real exchange rate = ln (avg RER_{t-60}) – ln (RER_t)
   RER is real exchange rate
   avg RER_{t-60} is average RER over 60 months prior to t
   RER_t is RER at time t
2. M2_FR (V2): M2 (liquid liabilities of monetary authority and commercial banks) to foreign reserves
3. LIBDUM*FORI (V3): Dummy for capital liberalization
   =0 , if no substantial control;
   =1 , if substantial control on capital transactions
4. DUMST_FR (V4): Dummy for the ratio of short-term debts to foreign reserves
   = 1 ,if the ratio exceeds unity; = 0 , otherwise
5. EXD_GNI (V5): Ratio of external debt to GNI (Gross National Income)

978-974-8308-56-2 ©2007 KMITL

6. CA_GDP (V6): Current account as a percentage of GDP (Gross Domestic Product)
7. CREGROW (V7): Domestic credit growth
8. GDPGROW (V8): Growth rate of GDP, negative annual growth of GDP is used
9. LB (V9): Lending boom is percentage change of ratio of financial system claims on private sector relative to GDP over four-year period
10. KAREV (V10): Capital inflows reversal
    $= KA_{t-1} - KA_t$
    KA_{t-1} and KA_t are capital account balance from official reserves of period t-1 and t
11. CONT (V11): Dummy for regional contagion effect
    = 1 , if there is a crisis in at least on country in the same geographical region contemporaneously;
    = 0 , otherwise

II. METHODOLOGY

-Data Source

1. International Financial Statistics CD-ROM (IFS), IMF
2. Direction of Trade Statistics Year Books and CD-ROM (DOTS), IMF
3. Balance of Payment Statistics Year Books (BOP), IMF
4. World Development Indicators CD-ROM (WDI), The World Bank
5. Global Development Finance CD-ROM (GDF), The World Bank
6. Exchange Arrangements and Exchange Restrictions Annual Report (EAER), IMF
7. Consolidated Banking Statistics (CBS), Bank for International Settlements (BIS)
8. Joint BIS-IMF-OECD-World Bank Statistics on External Debt (Joint-SED)
9. Die Falligkeitsverteilung der Internationalen Bankausleihung (FIB), BIS
10. Key indicators of developing Asian and Pacific countries, Asian Development Bank (ADB)

-Countries selected for this study are:

Argentina (Arg), Bolivia (Bol), Chile (Chi), Ecuador (Ecu), Mexico (Mex), Paraguay (Par), Peru (Per), Uruguay (Uru), Venezuela (Ven), India (India), Indonesia (Indo), Korea (Kor), Malaysia (Mal), Pakistan (Pak), Philippines (Phi), Singapore(Sin), Sri Lanka (Slk), and Thailand (Thai)

-Empirical Implementation

This study follows the empirical implementation proposed by Esquivel and Larrain in 1998 [12]. The crises are required to be at least five months apart, i.e., a country could have a maximum of two crises per year. Most of the explanatory variables entered in lagged form due to the purpose of the study to interpret the empirical results as the one-period-ahead probability of a currency crisis. The explanatory variables can be classified into two types. Firstly, the stock variables, whose units are measured at one point in time and are observed every month. The variables in this first group are OVERRER, M2_FR, LIBDUM*FORI, DUMST_FR, and EXD_GNI. Secondly, the flow variables, whose units are measured per unit of time, are observed, in this case, on yearly basis. The variables in this group are CA_GDP, CREGROW, GDPGROW, LB, KAREVS, and CONT. All variables are entered in lagged form for one period except CONT which is contemporaneous. In application there are problems which show that the modeling is not adequate. When a crisis occurs late in year t and one tries to use the explanatory variables of year t-1 to explain this crisis, sometimes many of explanatory variables change abruptly in the months before the collapse, and there was real evidence from changes in some of the explanatory variables, e.g. RER and foreign reserves, only a few months before the collapse. With respect to the stock variables if the crisis occurs late in year t (period "b" of year t), we should assume that the crisis occurred early in year t+1, and instead of taking the year-end value as usual, we take the mid-year value of year t to explain the crisis occurring in this particular period. This adjustment is made only for the stock explanatory variables. Esquivel and Larrain suggest that we should consider the characteristics of the flow variable, which indicates the change of a variable during one period in time, e.g., changing of the CA_GDP in year t. For these flow variables the year end value of year t-1 is to explain the crisis occurs in year t, even if it takes place in period "b" of year t. Table 1 presents this procedure after adjustment. This adjustment should provide more accuracy for the study, which attempts to estimate the impact of the explanatory variables on the one-period-ahead probability of crisis on yearly basis.

TABLE 1
TIME ADJUSTMENT OF EXPLANATORY VARIABLES

Occurrence of crisis		Explanatory variables		
Actual time	After adjustment	Stock variables	Flow variables	Contem-poraneous variable
In period "a" of year t	Year t	Year-end value of year t-1 (December value)	Year-end value of year t-1 (December value)	In period "a" of year t
In period "b" of year t	Year t+1	Mid-year value of year t (June value)	Year-end value of year t-1 (December value)	In period "b" of year t

-Determination of currency crises [12]

CPI-based RER is used to determine currency crises, and a currency crisis occurs according to the following formula:

$CRISIS_{it}^{DEP} = 1$,if $(\Delta^3 q_{it}^{cpi} > 0.+15)$ or $(\Delta^1 q_{it}^{cpi} > 2.54 \sigma_i^{\Delta q}$ and $\Delta^1 q_{it}^{cpi} > 0.04)$

$CRISIS_{it}^{DEP} = 0$, otherwise

$CRISIS_{it}^{DEP}$ means currency crisis identified by large and abrupt depreciation of the CPI-based RER.

$(\Delta^3 q_{it}^{cpi} > 0.15)$ means accumulated three-month change of CPI-based RER is 15 percent or more.

($\Delta^{1} q_{it}^{cpi} > 2.54\ \sigma_i^{\Delta q}$ and $\Delta^{1} q_{it}^{cpi} > 0.04$) means one month change in CPI-based RER is more than 2.54 times the country-specific standard deviation of the CPI-based RER monthly growth rate and the one month change in CPI-based RER exceeds 4 percent.

-Data in in-sample study:

-Training data: 11 explanatory variables and actual crises from 1980 to 2002

-Test data: 11 explanatory variables from1980 to 2002

-Data in out-of-sample study:

-Training data: 11 explanatory variables and actual crises starting from 1980 to 1990 to predict year 1991 then use the data 1980 to year 1991 to predict year 1992, etc.

-Test data: 11 explanatory variables of each year from 1991 to 2002.

-Data to test significance of a variable: Leave one variable out of the variable set which means using just 10 variables out of 11 variables and rotate the one left out until each variable is tested. The years of the dataset are the same as in in-sample study.

-Method

-SVM algorithm:

1. Data Normalization [11] to the range -1 to +1 using formula: $-1 + ((X_i - \text{Min})/(\text{Max} - \text{Min}))^* 2$
2. SVM implementation is shown in Table 2.
3. Criteria for correct prediction: If the prediction for year t from year t-1 is crisis (tranquil) and the fact in our data is also crisis (tranquil), then the prediction is considered correct.

TABLE 2
OPERATION SUMMARY OF THE SUPPORT VECTOR MACHINE LEARNING ALGORITHM [10]

Given	A training set T Comprising vectors $X_k \in R^n$ and desired output vectors $d_k \in \{-1,+1\}$
Initialize	• Choose a kernel function $K(\cdot,\cdot)$ • Set up the Hessian matrix : $H_{ij} = d_i d_j K(X_i, X_j)$ • Set C
Maximize	• $\Lambda^T I - \frac{1}{2} \Lambda^T H \Lambda$ Subject to the constraints : • $\Lambda \cdot D = 0$ • $\Lambda \geq 0$ • $\Lambda \leq CI$ using any quadratic program optimizer Obtain optimized Lagrange multipliers.
Predict Class	Given any test input X, set the class : $y(X)=\text{sign}\left(\sum_{i=1}^{n_s} d_i \hat{\lambda}_i K(X, X_i) + \hat{\omega}_o\right)$

III. RESULTS

-In-sample results

	Actual			
	C = value	Tranquil	Crisis	Total
Predicted	Tranquil	Correct non-crisis prediction (%)	Type I error Missing signal (%)	
	Crisis	Type II error False alarm (%)	Correct crisis prediction (%)	
	Total			Grand Total

	Actual			
	C = 0.1	Tranquil	Crisis	Total
Predicted	Tranquil	336 (98.82%)	37 (50.00%)	373
	Crisis	4 (1.18%)	37 (50.00%)	41
	Total	340	74	414

	Actual			
	C = 1	Tranquil	Crisis	Total
Predicted	Tranquil	337 (99.12%)	11 (14.86%)	348
	Crisis	3 (0.88%)	63 (85.14%)	66
	Total	340	74	414

	Actual			
	C = 10	Tranquil	Crisis	Total
Predicted	Tranquil	340 (100%)	2 (2.70%)	342
	Crisis	0 (0%)	72 (97.30%)	72
	Total	340	74	414

	Actual			
	C = 50	Tranquil	Crisis	Total
Predicted	Tranquil	340 (100%)	1 (1.35%)	341
	Crisis	0 (0%)	73 (98.65%)	73
	Total	340	74	414

	Actual			
	C = 100	Tranquil	Crisis	Total
Predicted	Tranquil	340 (100%)	0 (0%)	340
	Crisis	0 (0%)	74 (100%)	74
	Total	340	74	414

-Out-of-Sample prediction results from 1991 to 2002

	Actual			
	C = 0.1	Tranquil	Crisis	Total
Predicted	Tranquil	165 (87.30%)	20 (74.07%)	185
	Crisis	24 (12.70%)	7 (25.93%)	31
	Total	189	27	216

	Actual			
	C = 1	Tranquil	Crisis	Total
Predicted	Tranquil	156 (82.54%)	18 (66.67%)	174
	Crisis	33 (17.46%)	9 (33.33%)	42
	Total	189	27	216

	Actual			
	C = 10	Tranquil	Crisis	Total
Predicted	Tranquil	150 (79.37%)	19 (70.37%)	169
	Crisis	39 (20.63%)	8 (29.63%)	47
	Total	189	27	216

	Actual			
	C = 50	Tranquil	Crisis	Total
Predicted	Tranquil	146 (77.25%)	19 (70.37%)	165
	Crisis	43 (22.75%)	8 (29.63%)	51
	Total	189	27	216

	Actual			
	C = 100	Tranquil	Crisis	Total
Predicted	Tranquil	146 (77.25%)	19 (70.37%)	165
	Crisis	43 (22.75%)	8 (29.63%)	51
	Total	189	27	216

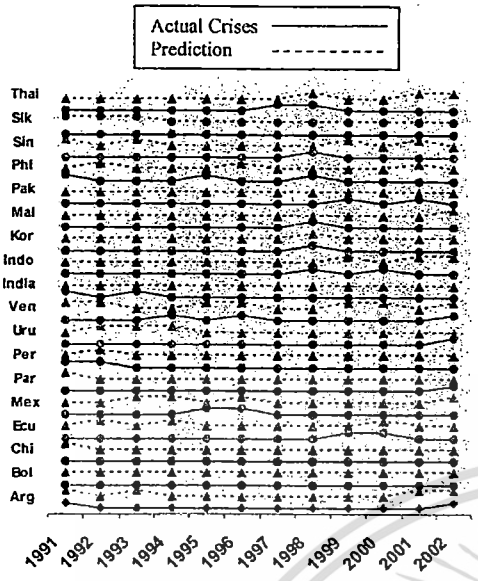


Fig. 1. Out-of-sample prediction plot with best results at C = 1

-Significance of Variable Results: In-sample testing, C= 100

V	1	2	3	4	5	6	7	8	9	10	11
Type I error	0	2	2	2	1	2	1	2	1	0	0
Type II error	0	0	0	0	0	1	0	0	0	0	0
Significance	N	Y	Y	Y	Y	Y	Y	Y	Y	N	N

N = No, Y = Yes

IV. DISCUSSION

-In-sample prediction

At C = 100, the result is perfect; there are no errors at all. But this will cause overfitting the effect of which appears in the out-of-sample testing.

-Out-of-sample prediction

The best results are from C = 1. Missing signal error is 66.67% which means the correct prediction of 33.33%. False alarm error is very low at 17.46 %, and overall error is 23.61%. These accuracies are considered good when we compare them with other studies [2],[3].

If we give an argument that sometimes there can be a delay of the onset of the currency crises, then we may extend our forecast to within two years instead of one year as criteria for correct prediction. Since our aim is to build an early warning system, two years early prediction[2] may also be useful for such a purpose. The result of this change of correct prediction criteria makes the prediction for out-of-sample much better. The best result is missing signal error at 51.85% with C = 1 which means 48.15% correct prediction of the crises (this result is not shown in the Results section above).

-Significance of variables

V1, V10, and V11 do not change the outcome of in-sample prediction at C = 100 (which has no error), so that means they have no effect in prediction for this set of data, and they can be considered less significant in our study.

V. CONCLUSION

Our best performance in out-of-sample prediction of currency crises for the one year prediction criteria is at C = 1 the result of which is 33% correctly predict crises and 82% correctly predict non-crises. The overall correct predictions (combined crises and non-crises) are 76%. If we translate these results into non-technical language, it would be: When our system predicts that there will not be a currency crisis, eight in ten times, there will not be a crisis in one year from now. When our system predicts that there will be a currency crisis, one third of the times, there will be a currency crisis in one year from now. And finally, whatever prediction made by our system, seven in ten times, it will be correct. Such a statement should be considered satisfactory if one wants to utilize an early warning system for currency crises. SVM, well known for its best generalization, is shown once again to be a viable method in forecast application.

REFERENCES

[1] S. Thearpiriyakij, and J. Breitung, "An Empirical Approach to Currency Crises: Latin America and Asia," *Diplomabeit, Rheinisch-Friedrich*-Wilhelms-Universitat, Bonn, Jun. 2005.

[2] C. Lin, H. Khan, Y. Wang, and R. Chang, "A New Approach to Modeling Early Warning Systems for Currency Crises: can a machine-learning fuzzy expert system predict the currency crises effectively?," *CIRJE-F-411*, Apr. 2006.

[3] T. Peltonen, "Are Emerging Market Currency Crises Predictable? A Test," *European Central Bank Working Paper No. 571*, 2006.

[4] I. Arciniegas, "SVM Sensitivity Analysis: An Application to Currency Crises Aftermaths," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, Vol. 34, No. 3, pp. 387-398, May 2004.

[5] X. Hui and J. Sun, "An Application of Support Vector Machine to Companies' Financial Distress Prediction," V. Torra et al. (Eds.): *MDAI 2006, LNAI 3885*, pp. 274 – 282, 2006.

[6] K. Muller, A. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen and V. Vapnik, "Predicting time series with support vector machines," *ICANN*, pp. 999-1004, 1997.

[7] V. Vapnik, *Statistical Learning Theory*, John Wiley&Sons , 1998.

[8] C. Cortes and V. Vapnik, "Support Vector Networks," *Machine Learning*, 20, pp. 273-297, 1995.

[9] S. Haykin, *Neural Networks. A Comprehensive Foundation* Macmillan, New York, NY, 1994.

[10] S. Kumar, *Neural Networks: A Classroom Approach*, McGraw-Hill, International Edition 2005.

[11] S. Ali and K. Smith-Miles, "Improved Support Vector Machine Generalization Using Normalized Input Space," A. Sattar and B.H. Kang (Eds.): *AI 2006, LNAI 4304*, pp. 362 – 371, 2006.

[12] G. Esquivel and F. Larrain. "Explaining Currency Crisis." [illegible], pp. 1-42, 1998

INTERNATIONAL JOURNAL of MATHEMATICAL MODELS AND METHODS IN APPLIED SCIENCES

ISSN: 1998-0140

FORMAT: Format (.doc) or Format (LaTeX)

Year 2010

All papers of the journal were peer reviewed by two independent reviewers. Acceptance was granted when both reviewers' recommendations were positive.

INTERNATIONAL JOURNAL OF MATHEMATICAL MODELS AND METHODS IN APPLIED SCIENCES

Detection of Currency Crises by a Novel Rule Extraction Method from Support Vector Machine

Prasan Pitiranggon, Somsri Banditvilai , and Nunthika Benjathepanun

Abstract—This study attempts to obtain a set of human comprehensible fuzzy if-then rules for the detection of currency crises from Support Vector Machine (SVM). SVM is used with explanatory variables known to be associated with currency crises to detect occurrences of currency crises. Fuzzy if-then rules are then obtained from the SVM through our novel rule extraction method which is called Support Vector Space Expansion (SVSE) method in order to unveil human comprehensible patterns behind SVM black-boxed system decision. The overall results of detection of currency crises of the fuzzy if-then rules are comparable to those from the SVM, and the if-then rules obtained may be used by financial experts to try to explain patterns of related financial statuses when currency crises occur, plus the if-then rules can also be easily incorporated into a software program using any popular computer language.

Keywords—Currency Crises, Fuzzy Rule Base, Rule Extraction, Support Vector Machine.

I. Introduction

Rule extraction methods are used to obtain fuzzy if-then rules from artificial neural networks (ANN) and SVM [1], [5]. SVM [4], [12], has been shown to outperform ANN in classification in many applications [18], so the if-then rules obtained from SVM should be superior to rules obtained from ANN in many applications. Methods for SVM rule extraction can be either pedagogical or decompositional. 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.

We have proposed a decompositional rule extraction technique from SVM in our recently published paper [16] which we will now call it Support Vector Space Expansion (SVSE) rule extraction method. Unlike other rule extraction methods from SVM, our technique makes use of strength of firing signals of support vectors partly similar to the way the original SVM makes decision, then each support vector expands its space to cover non-support vectors with the strongest Gaussian kernel function values. SVSE also guarantees that the number of final rules is equal or less than the number of support vectors obtained by SVM. We have validated our method against SVM using 5 benchmark data sets and found that the classification power of the SVSE is comparable to SVM.

In this study, we are trying to apply SVSE to a non-benchmark data; specifically, we want to use SVSE with financial data to predict currency crises [6].

There have been studies in the field of currency crises since the 1970's. Three models have been widely accepted to explain different currency crises occurred around the world [13], [15], [17]. The first model is based on the study of Krugman in 1979; it is used to explain currency crises in some countries in Latin America in 1970's to 1980's. The second model is based on the study of Obstfeld in 1994; it is used to explain currency crises in some countries in Europe in 1992 and Mexico in 1994. The third model is used to explain currency crises in some countries in Asia in 1997 to 1998. Many indicators and indices are believed to be statistically linked to these models. Through these variables, researchers have used many techniques to come up with the most accurate detection systems. These techniques involve traditional statistical methods [17], ANN [15], and more recently SVM [2] [8].

Either ANN or SVM does not reveal clearly to human comprehension how it makes decision; this is known as black-boxed characteristic [1], [11]. There is a need to have fuzzy if-then rules for the decision in order to make human expert understand the decision, so we will extract how SVM makes each decision in the form of if-then rules, through which we will predict whether there is a currency crisis or no crisis in a particular year.

II. Background

This study uses SVM classifier with data from countries in Latin America and Asia [17] to detect currency crises during 1980 to 2002. The data used are 11 explanatory variables which have been shown in the past to be significantly associated with the occurrences of currency crises.

The 11 explanatory variables are:

1) Overvaluation of real exchange rate (OVERRER)

2) Liquid liabilities of monetary authority and commercial banks to foreign reserves (M2_FR)

3) Dummy for capital liberalization (LIBDUM_FORI)

4. Dummy for the ratio of short-term debts to foreign reserves (DUMST_FR)

5. Ratio of external debt to Gross National Income (EXD_GNI)

6. Current account as a percentage of Gross Domestic Product (CA_GDP)

P. Pitiranggon is with King Mongkut's Institute of Technology Ladkrabang, Chalongkrung Rd., Ladkrabang, Bangkok 10520, Thailand (phone: +668-1303-6484; e-mail: prasan.pitiranggon@hotmail.com).

S. Banditvilai is with King Mongkut's Institute of Technology Ladkrabang (e-mail: kbsomsri@kmitl.ac.th).

N. Benjathepanun is with King Mongkut's Institute of Technology Ladkrabang (e-mail: kbnunthi@kmitl.ac.th).

7. Domestic credit growth (CREGROW)
8. Growth rate of Gross Domestic Product (GDPGROW)
9. Lending boom as a percentage change of ratio of financial system claims on private sector relative to Gross Domestic Product over four-year period (LB)
10. Capital inflows reversal (KAREVS)
11. Dummy for regional contagion effect (CONT)

Even though SVM can detect currency crisis patterns, the way it makes decision is not obvious to human comprehension. So our novel decompositional rule extraction method, SVSE, is used to reveal human comprehensible if-then rules in this study. It uses each support vector to pair with each non-support vector to select the strongest firing signal among the support vectors and expand the coverage in input space until no more non-support vectors left to pair with. Other studies on rule extraction techniques for SVM using decompositional techniques are SVM + Prototype [14], Tree related method [3] , and Cubes and separating hyperplane method [7], while the ones using pedagogical techniques are Iter [9] and Minerva [10].

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 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, the algorithm converts all of the current hypercubes into rules. Ellipsoids can also be used in place of hypercubes. For another decompositional technigue, decision tree is used. This tree related method makes use of the information provided by the support vectors and the parameters associated with them. In the first stage which is a learning stage, the approach handles the rule-extraction by using labeled patterns to train an SVM and get an SVM model as a classifier with acceptable accuracy. In the second stage which is a rule generation stage, the objective 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. Cubes and Separating Hyperplane method is the last decompositional technique mentioned. In this method, 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 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, the algorithm calculates how far the cube can be expanded to both extremes of the dimension before it intersects with another cube; these distances are called LowerLimit and UpperLimit. Secondly, for each hypercube and for each input dimension, the algorithm calculates 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, the algorithm creates 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, the algorithm creates a number of random points lying within the cube and calculates the mean prediction for these points according to the trained continuous regression model. The difference between each of both means and the mean prediction for the original cube respectively are called 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. The mean prediction for this cube is updated, and all other temporary cubes are removed. Each of these cubes can then be converted into a rule of the following form:

if Var 1 $\in$ [Value1Low ,Value1High] and Var 2. $\in$ [Value2Low,Value2High]

. . . and Var M $\in$ [ValueMLow,ValueMHigh] then predict some Constant

where M is the dimension of the input space. Minerva is the other pedagogical method for SVM rule extraction mentioned. 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). Then fuzzy if-then rules can be generated around support vectors based on firing strength to form our Fuzzy Rule Base (FRB) rules, e.g., IF x > c1 AND x < c2 THEN y = d. The rules are modified further by combining completely coincided ranges among the if-then conditionals using a fuzzy logic process called input scatter partitioning, and this set of rules is our final set needed.

III. Method

Data Source for Currency Crisis Study

1) International Financial Statistics CD-ROM (IFS), IMF

2) Direction of Trade Statistics Year Books and CD-ROM (DOTS), IMF

3) Balance of Payment Statistics Year Books (BOP), IMF

4) World Development Indicators CD-ROM (WDI), The World Bank

5) Global Development Finance CD-ROM (GDF), The World Bank

6) Exchange Arrangements and Exchange Restrictions Annual Report (EAER), IMF

7) Consolidated Banking Statistics (CBS), Bank for International Settlements (BIS)

8) Joint BIS-IMF-OECD-World Bank Statistics on External Debt (Joint-SED)

9) Die Falligkeitsverteilung der Internationalen Bankausleihung (FIB), BIS

10) Key indicators of developing Asian and Pacific countries, Asian Development Bank (ADB)

Countries selected for this study:

Argentina (Arg), Bolivia (Bol), Chile (Chi), Ecuador (Ecu), Mexico (Mex), Paraguay (Par), Peru (Per), Uruguay (Uru), Venezuela (Ven), India (India), Indonesia (Indo), Korea (Kor), Malaysia (Mal), Pakistan (Pak), Philippines (Phi), Singapore(Sin), Sri Lanka (Slk), and Thailand (Thai)

Data Preparation

This study follows the empirical implementation proposed by Esquivel and Larrain in 1998 [6]. The crises are required to be at least five months apart, i.e., a country could have a maximum of two crises per year. Most of the explanatory variables entered in lagged form due to the purpose of the study to interpret the empirical results as the one-period-ahead probability of a currency crisis. The explanatory variables can be classified into two types. Firstly, the stock variables, whose units are measured at one point in time and are observed every month. The variables in this first group are OVERRER, M2_FR, LIBDUM*FORI, DUMST_FR, and EXD_GNI. Secondly, the flow variables, whose units are measured per unit of time, are observed, in this case, on yearly basis. The variables in this group are CA_GDP, CREGROW, GDPGROW, LB, KAREVS, and CONT. All variables are entered in lagged form for one period except CONT which is contemporaneous. When a crisis occurs late in year t and one tries to use the explanatory variables of year t-1 to explain this crisis, sometimes many of explanatory variables change abruptly in the months before the collapse, and there was real evidence from changes in some of the explanatory variables, e.g. RER and foreign reserves, only a few months before the collapse. With respect to the stock variables if the crisis occurs late in year t (period "b" of year t), we should assume that the crisis occurred early in year t+1, and instead of taking the year-end value as usual, we take the mid-year value of year t to explain the crisis occurring in this particular period. This adjustment is made only for the stock explanatory variables. Esquivel and Larrain suggest that we should consider the characteristics of the flow variable, which indicates the change of a variable during one period in time, e.g., changing of the CA_GDP in year t. For these flow variables the year end value of year t-1 is to explain the crisis occurs in year t, even if it takes place in period "b" of year t. This adjustment should provide more accuracy for the study, which attempts to estimate the impact of the explanatory variables on the one-period-ahead probability of crisis on yearly basis.

SVM Classification

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:

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

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

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 $\Phi(\bullet)$ 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, 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:

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

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

$$\frac{1}{2}\|W\|^2 + C\sum_{i=1}^{Q}\xi_i \quad (4)$$

subject to

$$d_i\left[W.\Phi(X_i)+w_0\right]-1+\xi_i \geq 0, \quad i=1,\ldots,Q \quad (5)$$

$$\xi_i \geq 0 \quad (6)$$

Separating hyperplane $(W.\Phi(X_i)+w_0=0)$, which is used in main classification decision and formed from two boundaries - class 1 boundary $(W.\Phi(X_i)+w_0=+1)$ and class 2 boundary $(W.\Phi(X_i)+w_0=-1)$ is shown in Fig. 1. Class 1 boundary is formed from unbounded support vectors of class 1 (represented as circles in Fig. 1), and class 2 boundary is formed from unbounded support vectors of class 2 (represented as squares in Fig. 1). 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.

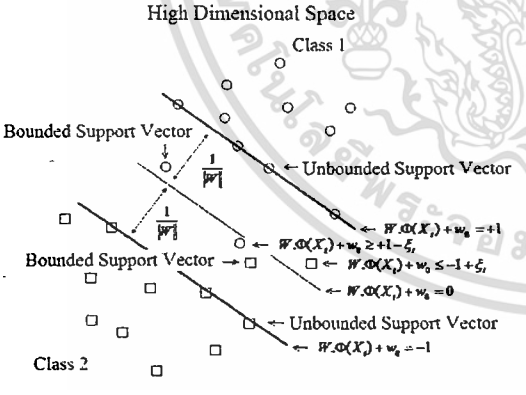


Fig. 1. Bounded and unbounded support vectors, misclassification vectors, and separating hyperplane

The part involving $\|W\|^2$ 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

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

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,

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

must also be satisfied.

This gives dual form of (7):

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

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

$$K(X_i,X_j)=\sum_k(\Phi_k(X_i).\Phi_k(X_j)) \quad (10)$$

iff

$$\int\int K(X_i,X_j)g(X_i)g(X_j)dX_i dX_j \geq 0 \quad (11)$$

For all g(X) such that

$$\int g^2(X)dX < \infty \quad (12)$$

There are a few kernel functions which satisfy Mercer's 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 Mercer's Condition.

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

$$\sum_{i=1}^{Q}\lambda_i d_i = 0 \quad (13)$$

$$0 \leq \lambda_i \leq C; i=1,\ldots,Q \quad (14)$$

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 $C(\lambda_i > 0; \lambda_i < C)$. 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 $C(\lambda_i > 0; \lambda_i = C)$.

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

$$w_0 = \frac{1}{m}\left[\sum_{k=1}^{m}\left(\frac{1}{d_k} - W.X_k\right)\right] \quad (15)$$

where m = number of unbounded support vectors and

$$W = \sum_{k=1}^{N} \lambda_k d_k X_k \quad (16)$$

where N = number of all support vectors

We can now get the final classifier:

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

where X_s = unbounded support vector

This final equation is used in the SVM detection of currency crises. The unbounded support vectors obtained earlier are used in the rule generation step.

Rule Extraction

A. 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. 2, 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 for implementing rule generation from unbounded support vectors found from previous step is shown in Fig. 2; X_i is input vector, and A_i is Gaussian kernel function of unbounded support vector and input vector.

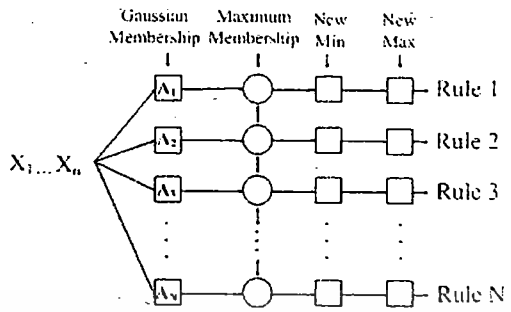


Fig. 2. Rules generation algorithm based on kernel firing strength

The final min and max values of each row are used as a range of the newly generated if-then rules. The 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 $x_{12} < a_{11+}$) AND ... ($x_{1n} > a_{1n-}$ 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 $x_{22} < a_{21+}$) AND ... ($x_{2n} > a_{2n-}$ 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 $x_{32} < a_{31+}$) AND ... ($x_{3n} > a_{3n-}$ 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 ... ($x_{Nn} > a_{Nn-}$ 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.

B. 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 if-then statements from previous step together as long as it does not cause misclassification. Algorithm's pseudo code for input scatter partitioning is:

[Pre-loop condition: 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 ... $x_{1n} \in [a_{1n-}, a_{1n+}]$ THEN $y = v_1$

IV. Results

We perform classification using both SVM and our fuzzy if-then rules on Latin America compared to Asia data sets. Ten-fold cross validation technique is used in each data set to evaluate effectiveness of the classification.

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 Asia data set than our method, but our method performs better in Latin America data set.

TABLE I
COMPARISON OF ERRORS FROM SVM AND SVSE RULES

Data	SVM		SVSE Rules	
	Avg USV	%Error	Avg Rules	%Error
Latin	33.90	29.47	33.90	28.02
Asia	14.70	10.63	14.70	15.46

The main reason why percent errors in SVM are different from our method is because of the misclassification vectors in the input data. SVM makes use of USV from both class 1 and class 2 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.

The final if-then rules obtained by our rule extraction method for Latin America entire data set, and the final if-then rules for Asia entire data set are shown in appendix.

V. Conclusion

The results of our empirical study have shown that the SVSE method can outperform SVM decisions in the Latin America data set, but under perform in the Asia data set. Percent errors of the two methods are not significantly different. It can be concluded that the results of the errors from the two methods are comparable, just like the results from 5 benchmark data sets in our previously published paper [16].

The SVSE method is shown to be a good method for rule extraction from SVM in currency crisis application 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 if-then rules obtained can be easily incorporated into a computer program using any popular computer language.

A suggestion for future study is to use data sets from other applications in order to obtain useful human comprehensible rules extracted from the pattern classification by SVM in different expert domains.

Appendix

The final form of fuzzy if-then rules is in the form:

IF $x_1 \in [a_{1-}, a_{1+}]$ AND $x_2 \in [a_{2-}, a_{2+}]$ AND ... $x_{11} \in [a_{11-}, a_{11+}]$ THEN y = crisis occurs

Where x_i is one of the 11 parameters from our data set; a_{i-} is the lower range of each parameter, and a_{i+} is the upper range of each parameter.

In order to save space, we present the rules in the format:

Rule no.	$[a_{1-}, a_{1+}]$	$[a_{2-}, a_{2+}]$	$[a_{3-}, a_{3+}]$
	$[a_{4-}, a_{4+}]$	$[a_{5-}, a_{5+}]$	$[a_{6-}, a_{6+}]$
	$[a_{7-}, a_{7+}]$	$[a_{8-}, a_{8+}]$	$[a_{9-}, a_{9+}]$
	$[a_{10-}, a_{10+}]$	$[a_{11-}, a_{11+}]$	

Final fuzzy rules for currency crises for Latin America data set:

1.	[0.06, 1.06],	[4.16, 10.66],	[0.00, 12.60],
	[1, 1],	[0.01, 0.41],	[-4.08, -0.18],
	[41, 171],	[-1.85, 11.15],	[-110, 226],
	[443, 3823],	[1, 1]	
2.	[-0.61, 1.09],	[2.88, 11.38],	[0.00, 14.40],
	[1, 1],	[-0.06, 0.44],	[-5.18, -0.08],
	[106, 276],	[-12.69, 3.31],	[-89, 247],
	[-1130, 3290],	[1, 1]	
3.	[-0.61, 0.49],	[1.87, 8.87],	[0.00, 12.60],
	[1, 1],	[-0.06, 0.44],	[-4.89, -0.69],
	[150, 290],	[-10.96, 2.04],	[-85, 251],
	[1767, 5147],	[1, 1]	
4.	[-0.60, 0.58],	[4.43, 10.33],	[0.00, 9.00],
	[1, 1],	[-0.07, 0.50],	[-3.33, 3.89],
	[10, 631],	[-1.79, 8.61],	[-84, 207],
	[-4396, -1348],	[1, 1]	
5.	[-0.68, 1.02],	[5.37, 15.37],	[0.00, 14.40],
	[0, 1],	[-0.05, 0.45],	[-6.89, -0.89],
	[114, 294],	[-6.09, 11.91],	[-94, 242],

	[-2030, 2650],	[1, 1]			[0, 1],	[-0.02, 0.48],	[-14.59, 2.21],
6.	[-0.60, 1.10],	[0.86, 56.36],	[0.00, 14.40],		[-27, 343],	[-12.23, 11.77],	[-89, 247],
	[0, 1],	[-0.06, 0.44],	[-14.60, 6.70],		[-15093, -7813],	[0, 1]	
	[379, 7229],	[-12.50, 11.50],	[-126, 210],	21.	[-0.68, 1.02],	[3.26, 56.26],	[0.00, 14.40],
	[-15208, 26392],	[0, 1]			[0, 1],	[-0.01, 0.49],	[-14.50, 6.80],
7.	[-0.60, 1.10],	[1.09, 18.59],	[0.00, 14.40],		[-25, 685],	[-12.63, 11.37],	[-88, 248],
	[0, 1],	[-0.06, 0.44],	[-6.08, 6.82],		[9118, 26278],	[0, 1]	
	[378, 988],	[-12.40, 11.60],	[-120, 216],	22.	[-0.68, 1.02],	[0.69, 24.69],	[0.00, 14.40],
	[-9999, 5861],	[0, 1]			[0, 1],	[0.02, 0.42],	[-14.55, 3.45],
8.	[-0.65, 1.05],	[1.01, 28.51],	[0.00, 14.40],		[-27, 383],	[-12.58, 11.42],	[-100, 236],
	[0, 1],	[0.00, 0.50],	[-14.68, 6.62],		[9133, 26293],	[0, 1]	
	[-31, 459],	[-12.41, 11.59],	[-101, 235],	23.	[-0.63, 1.07],	[1.02, 41.02],	[0.00, 14.40],
	[9126, 26286],	[0, 1]			[0, 1],	[0.02, 0.42],	[-14.65, 6.65],
9.	[-0.65, 1.05],	[7.02, 17.02],	[0.00, 14.40],		[-33, 687],	[-12.17, 11.83],	[-114, 222],
	[1, 1],	[-0.06, 0.44],	[-10.36, -4.96],		[9114, 26274],	[0, 1]	
	[-33, 117],	[-8.72, 9.28],	[-83, 253],	24.	[-0.45, 0.75],	[0.77, 4.27],	[0.00, 10.80],
	[-2751, 1929],	[1, 1]			[0, 0],	[-0.06, 0.44],	[-7.53, -4.53],
10.	[-0.30, 0.64],	[1.28, 13.46],	[0.00, 7.20],		[-24, 66],	[6.36, 11.36],	[-82, 254],
	[1, 1],	[-0.03, 0.45],	[-3.99, -0.04],		[-1385, 1216],	[0, 0]	
	[131, 398],	[-7.04, 1.44],	[-97, 198],	25.	[-0.35, 0.65],	[0.76, 4.76],	[-0.31, 15.89],
	[-1049, 1545],	[1, 1]			[0, 0],	[0.03, 0.43],	[-5.32, -2.32],
11.	[-0.65, 1.05],	[0.83, 33.33],	[0.00, 14.40],		[-30, 70],	[-7.02, 0.98],	[-90, 198],
	[0, 1],	[-0.01, 0.49],	[-14.68, 6.62],		[-989, 1091],	[1, 1]	
	[378, 1628],	[-12.20, 11.80],	[-81, 255],	26.	[-0.28, 0.10],	[0.91, 3.67],	[0.00, 3.37],
	[-15143, 15797],	[0, 1]			[0, 0],	[-0.06, 0.15],	[-7.05, -3.18],
12.	[-0.61, 1.09],	[0.80, 56.30],	[0.00, 14.40],		[6, 44],	[-1.00, 7.35],	[-28, 47],
	[0, 1],	[-0.02, 0.48],	[-14.59, 6.71],		[-238, 555],	[1, 1]	
	[377, 5007],	[-12.68, 12.32],	[-107, 229],	27.	[-0.69, 1.11],	[3.86, 21.36],	[0.00, 14.40],
	[-15285, 26315],	[0, 1]			[0, 1],	[-0.03, 0.47],	[-10.50, 0.00],
13.	[-0.65, 1.05],	[1.01, 9.51],	[0.00, 14.40],		[367, 717],	[-11.70, 9.30],	[-106, 230],
	[0, 1],	[-0.05, 0.45],	[-14.50, -10.90],		[-4467, 4633],	[0, 1]	
	[-31, 149],	[-7.26, 11.74],	[-128, 208],	28.	[-0.69, 1.01],	[0.77, 56.27],	[0.00, 14.40],
	[-4647, 1593],	[0, 1]			[0, 1],	[-0.04, 0.46],	[-14.65, 6.65],
14.	[-0.61, 1.09],	[0.97, 8.97],	[0.00, 14.40],		[383, 3393],	[-11.70, 12.30],	[-97, 239],
	[0, 1],	[0.00, 0.40],	[-14.28, -7.68],		[-15320, 26280],	[0, 1]	
	[-30, 160],	[-3.03, 11.97],	[-116, 220],	29.	[-0.63, 1.07],	[0.95, 55.95],	[0.00, 14.40],
	[-5744, -544],	[0, 1]			[0, 1],	[-0.04, 0.46],	[-14.80, 6.80],
15.	[-0.13, 0.37],	[3.87, 10.56],	[3.88, 15.73],		[375, 7235],	[-12.14, 11.86],	[-99, 237],
	[1, 1],	[0.15, 0.49],	[-8.84, 1.08],		[-15310, 26290],	[0, 1]	
	[-5, 50],	[-1.57, 5.36],	[-44, 132],	30.	[-0.64, 1.06],	[21.90, 55.90],	[-0.30, 14.10],
	[-545, 783],	[1, 1]			[0, 1],	[-0.01, 0.49],	[-14.53, 6.77],
16.	[-0.69, 0.51],	[1.12, 7.62],	[0.00, 14.40],		[-31, 809],	[-11.76, 12.24],	[-125, 211],
	[1, 1],	[-0.03, 0.47],	[-13.38, -9.18],		[-15254, 17506],	[0, 1]	
	[-24, 56],	[-9.15, 4.85],	[-124, 212],	31.	[-0.32, 0.28],	[1.74, 4.74],	[-1.76, 7.24],
	[-2644, 996],	[1, 1]			[0, 0],	[-0.06, 0.44],	[-3.53, -1.73],
17.	[-0.66, 0.14],	[2.44, 6.44],	[1.54, 14.14],		[-22, 38],	[-5.39, -1.39],	[0, 192],
	[1, 1],	[0.01, 0.41],	[-7.69, -5.59],		[-247, 793],	[1, 1]	
	[-28, 32],	[5.37, 11.37],	[-114, 126],	32.	[-0.11, 0.45],	[1.50, 7.73],	[7.39, 11.26],
	[-357, 1463],	[1, 1]			[1, 1],	[0.02, 0.46],	[1.28, 6.55],
18.	[-0.66, 0.94],	[0.70, 7.20],	[-0.11, 14.29],		[-3, 83],	[-4.76, 2.10],	[-83, 63],
	[1, 1],	[0.02, 0.42],	[-11.12, -6.92],		[-48, 2448],	[1, 1]	
	[-32, 88],	[-4.89, 9.11],	[-82, 254],	33.	[-0.45, 0.13],	[2.15, 5.01],	[0.00, 3.60],
	[-3282, 358],	[0, 0]			[1, 1],	[0.06, 0.46],	[-5.47, -3.25],
19.	[-0.67, 0.83],	[0.72, 7.22],	[-0.95, 15.25],		[-24, 42],	[3.07, 7.59],	[-70, 66],
	[0, 1],	[0.01, 0.41],	[2.51, 6.71],		[-1736, 338],	[1, 1]	
	[-31, 89],	[-12.30, 3.70],	[-93, 243],	34.	[-0.09, 0.57],	[1.48, 5.53],	[0.00, 3.60],
	[451, 5131],	[0, 0]			[1, 1],	[-0.03, 0.50],	[-10.22, -4.24],
20.	[-0.66, 1.04],	[5.25, 34.25],	[-0.46, 15.74],		[30, 166],	[4.84, 9.00],	[-97, 55],

	[120, 2400],	[1, 1]	
35.	[-0.13, 0.47],	[0.85, 3.85],	[0.00, 5.40],
	[0, 0],	[-0.05, 0.45],	[-4.22, -2.42],
	[-3, 57],	[-2.75, 3.25],	[-118, 122],
	[210, 1510],	[0, 0]	
36.	[0.02, 0.60],	[1.16, 2.35],	[0.00, 3.79],
	[0, 0],	[-0.05, 0.21],	[-4.65, 2.90],
	[17, 168],	[1.17, 4.97],	[-105, 63],
	[-3313, 20],	[1, 1]	

Final fuzzy rules for currency crises for Asia data set:

1.	[-0.30, 0.30],	[51.91, 88.41],	[0.00, 9.00],
	[0, 1],	[-0.09, 3.81],	[-8.47, 15.83],
	[-16, 53],	[-8.19, 8.81],	[-102, 137],
	[-3775, 20924],	[0, 1]	
2.	[-0.31, 0.29],	[0.56, 9.56],	[-1.00, 8.00],
	[1, 1],	[-0.05, 0.95],	[-8.34, -2.94],
	[-13, 46],	[-7.90, 9.10],	[-69, 122],
	[-3761, -1681],	[1, 1]	
3.	[-0.22, 0.28],	[0.92, 7.92],	[-1.66, 9.14],
	[1, 1],	[-0.02, 0.78],	[-6.63, -2.43],
	[-12, 47],	[-1.04, 8.96],	[-82, 157],
	[-3955 ,-835],	[1, 1]	
4.	[-0.27, 0.23],	[0.71, 13.21],	[0.00, 9.00],
	[0, 1],	[0.00, 1.50],	[-6.17, 1.63],
	[-11, 48],	[-8.30, 8.70],	[-74, 117],
	[8070, 14830],	[0, 1]	
5.	[-0.32, 0.14],	[2.95, 22.78],	[0.00, 1.80],
	[0, 0],	[0.01, 0.19],	[-2.40, 4.43],
	[10, 29],	[0.79, 6.50],	[-60, 81],
	[-3697, 995],	[1, 1]	
6.	[-0.29, 0.21],	[2.40, 11.40],	[0.00, 9.00],
	[1, 1],	[-0.06, 1.04],	[-8.57, -3.77],
	[-14, 55],	[-1.94, 9.06],	[-90, 149],
	[-4004, -1144],	[0, 0]	
7.	[-0.26, 0.24],	[0.71, 51.71],	[0.00, 9.00],
	[0, 1],	[0.00, 3.80],	[-8.48, 15.82],
	[-16, 53],	[-7.99, 9.01],	[-75, 116],
	[9719, 33119],	[0, 1]	
8.	[0.01, 0.23],	[3.69, 27.27],	[0.00, 1.80],
	[0, 0],	[0.01, 0.29],	[-5.93, -2.64],
	[8, 39],	[2.55, 7.32],	[5, 48],
	[131, 7279],	[1, 1]	
9.	[-0.25, 0.25],	[9.07, 20.07],	[0.00, 9.00],
	[0, 1],	[-0.04, 1.16],	[-8.34, -5.04],
	[-18, 51],	[-8.13, 8.87],	[-92, 147],
	[636, 6096],	[0, 1]	
10.	[-0.27, 0.23],	[0.93, 10.93],	[0.00, 9.00],
	[1, 1],	[0.00, 1.10],	[-2.82, 2.88],
	[-13, 46],	[-7.31, 2.69],	[-89, 150],
	[-1886, 3053],	[1, 1]	
11.	[-0.30, 0.29],	[8.92, 16.44],	[0.00, 7.20],
	[1, 1],	[-0.05, 0.48],	[-6.98, -3.84],
	[-12, 50],	[0.04, 8.75],	[-65, 150],
	[-1483, 439],	[1, 1]	
12.	[-0.21, 0.29],	[3.43, 7.43],	[0.00, 7.20],
	[0, 0],	[-0.07, 0.43],	[-5.80, -3.40],
	[-18, 51],	[0.39, 8.39],	[-82, 157],
	[-2633, -1073],	[0, 0]	
13.	[-0.29, 0.21],	[0.95, 9.95],	[0.00, 9.00],
	[0, 1],	[-0.04, 1.06],	[-8.28, -2.28],
	[-10, 49],	[-4.81, 9.19],	[-77, 114],
	[2439, 7379],	[1, 1]	
14.	[-0.27, 0.30],	[1.02, 5.09],	[-1.66, 9.72],
	[1, 1],	[2.27, 3.74],	[-2.12, 15.63],
	[-8, 49],	[-7.44, 8.51],	[-75, 118],
	[-938, 5272],	[1, 1]	
15.	[-0.32, 0.28],	[10.77, 20.77],	[0.00, 9.00],
	[1, 1],	[-0.08, 1.02],	[-8.62, -2.92],
	[-14, 55],	[-6.70, 9.30],	[-105, 134],
	[-2667, 2272],	[0, 0]	
16.	[-0.27, 0.23],	[0.86, 7.86],	[0.00, 9.00],
	[1, 1],	[-0.03, 1.07],	[-8.37, -5.97],
	[-16, 54],	[-1.10, 8.90],	[-105, 134],
	[602, 4242],	[0, 0]	

REFERENCES

[1] 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.

[2] I. Arciniegas. SVM Sensitivity Analysis: An Application to Currency Crises Aftermaths. IEEE Transactions on Systems, Man, and Cybernetics – Part A: *Systems and Humans*, Vol. 34, No. 3, pp. 387-398, 2004.

[3] 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.

[4] C. Cortes and V. N. Vapnik, "Support Vector Networks, *Machine Learning*," vol. 20, pp. 273-297, 1995.

[5] J. Diederich (Ed.), "Rule Extraction from Support Vector Machines," *Studies in Computational Intelligence*, vol. 80, Springer-Verlag, Berlin, Heidelberg, pp. 3-30, 2008.

[6] G. Esquivel and F. Larrain, Explaining Currency Crisis, pp. 1-42, 1998. Available: http://www.cid.harvard.edu/hnid/646.pdf

[7] 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*, pp. 32–40, 2005.

[8] X. Hui and J. Sun. An Application of Support Vector Machine to Companies' Financial Distress Prediction, V. Torra et al. (Eds.): *MDAI 2006*, LNAI 3885, pp. 274 – 282, 2006.

[9] 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, pp. 270-279, 2006.

[10] 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.

[11] E. Kolman and M. Margaliot, "Are artificial neural networks white boxes?" *IEEE Trans. Neural Networks*, vol. 16, no. 4, pp. 844-852, 2005.

[12] S. Kumar, *Neural Networks: A Classroom Approach*. McGraw-Hill, International Edition, 2005, pp. 273-304.

[13] C. Lin, H. Khan, Y. Wang, and R. Chang. A New Approach to Modeling Early Warning Systems for Currency Crises: can a machine-learning fuzzy expert system predict the currency crises effectively? *CIRJE-F-411*, 2006.

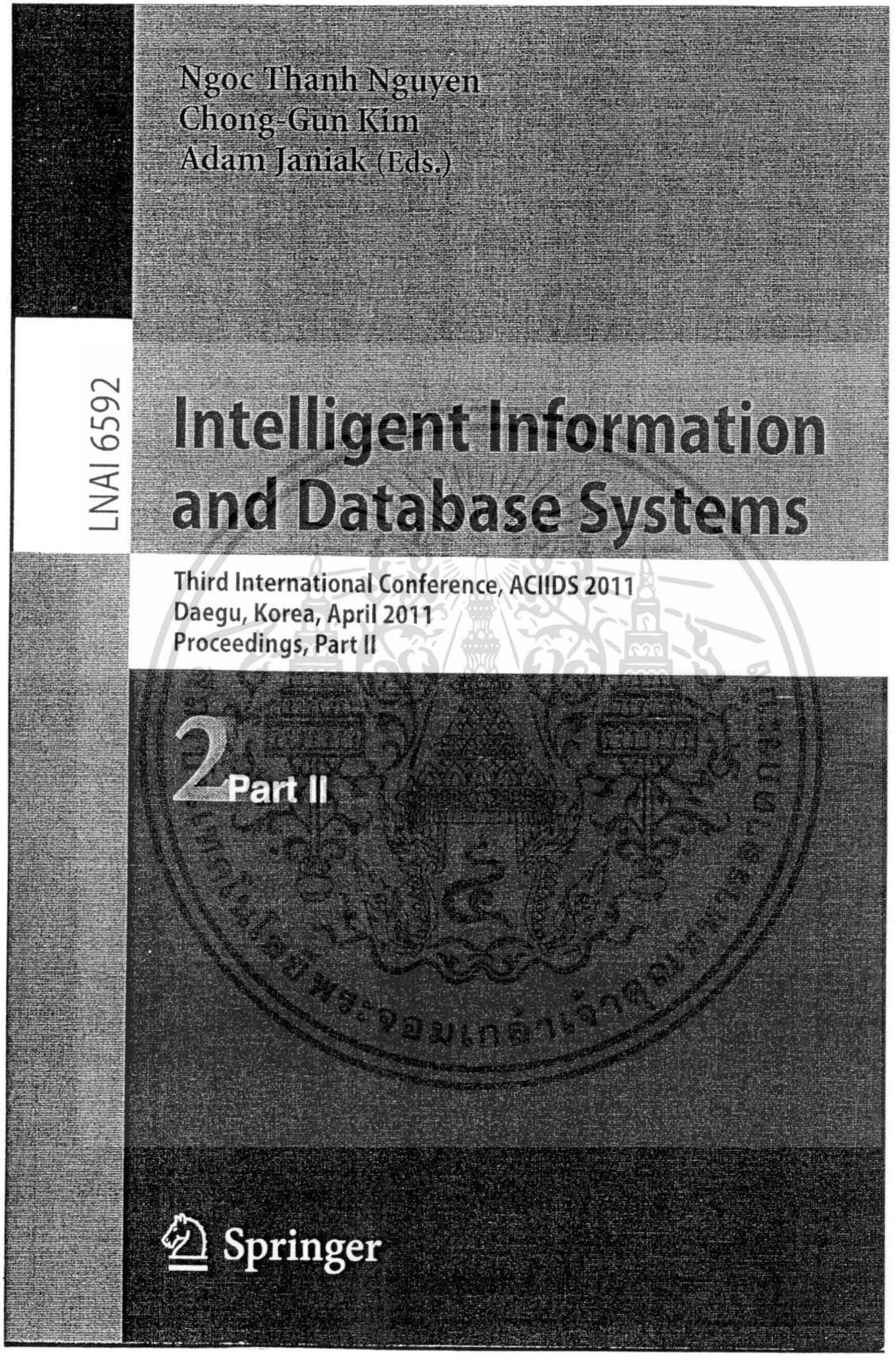
[14] H. Nunez, C. Angulo, and A. Catala, "Rule Extraction Based on Support and Prototype Vectors." in: *Studies in Computational Intelligence*, vol. 80, Springer-Verlag, Berlin, Heidelberg, pp. 109-134, 2008.

[15] T. Peltonen. Are Emerging Market Currency Crises Predictable? A Test. *European Central Bank Working Paper*, No. 571, 2006.

[16] P. Pitiranggon, N. Benjathepanun, S. Banditvilai, and V. Boonjing, "Fuzzy Rules Generation and Extraction from Support Vector Machine Based on Kernel Function Firing Signals," *International Journal of Engineering and Applied Science*, vol. 6, no. 4, pp. 244-251, 2010.

[17] S. Thearpiriyakij, J. Breitung. An Empirical Approach to Currency Crises: Latin America and Asia. *Diplomabeit, Rheinisch-Friedrich-Wilhelms-Universitat, Bonn*, Jun 2005.

[18] V. N. Vapnik, *Statistical Learning Theory*. John Wiley&Sons. 1998, pp. 375-520.



Rule Extraction for Support Vector Machine Using Input Space Expansion

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

Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, Chalongkrung Rd., Ladkrabang, Bangkok 10520, Thailand
prasan.pitiranggon@hotmail.com, kbnunthi@kmitl.ac.th, kbsomsri@kmitl.ac.th, kbveera@kmitl.ac.th
http://www.kmitl.ac.th

Abstract. Fuzzy Rule-Based System (FRB) in the form of human comprehensible IF-THEN rules can be extracted from Support Vector Machine (SVM) which is regarded as a black-boxed system. We first prove that SVM decision network and the zero-ordered Sugeno FRB type of the Adaptive Network Fuzzy Inference System (ANFIS) are equivalent indicating that SVM's decision can actually be represented by fuzzy IF-THEN rules. We then propose a rule extraction method based on kernel function firing strength and unbounded support vector space expansion. An advantage of our method is the guarantee that the number of final fuzzy IF-THEN rules is equal or less than the number of support vectors in SVM, and it may reveal human comprehensible patterns. We compare our method against SVM using popular benchmark data sets, and the results are comparable.

Keywords: Rule extraction, fuzzy IF-THEN rules, Support Vector Machine, pattern classification.

1 Introduction

Support Vector Machine (SVM) is a great tool to approximate functions, recognize patterns, or predict outcomes [13], [17]. Despite its great performance, it suffers from its black-boxed characteristics [2], [4]. To make it white-boxed, rule extraction is needed [12]. 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. Techniques specifically intended as SVM rule extraction techniques are based on translucency and scope. Translucency can be either pedagogical or decompositional, and scope 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 [15] which uses input clustering to obtain

N.T. Nguyen, C.-G. Kim, and A. Janiak (Eds.): ACIIDS 2011, LNAI 6592, pp. 100–109, 2011.

prototype vectors and geometrical formulas to obtain ranges for IF-THEN rules, tree related method [3] which uses a tree technique on support vectors to obtain IF-THEN rules, and cubes and separating hyperplane related [6] which uses cubes extending from separating hyperplane and can be used only in linear SVM, while the ones using pedagogical techniques are Iter [8] which uses randomly generated vectors in input space to cover entire input vectors and Minerva [9] which is similar to Iter but uses Sequential Covering method additionally.

None of the previous studies uses kernel function strength in rule extraction. Our study makes use of the kernel function strength similar to the way SVM makes decisions to extract rules; therefore, the decision rules obtained are closer to SVM decisions. Moreover, our technique guarantees that the number of final rules is less than the number of support vectors obtained by SVM. The technique used in our study is considered decompositional in translucency and classification in scope. Our technique looks for unbounded support vectors, which are the data points used as base locations to define separating hyperplane, to build trained SVM decision network to classify testing data. We can prove 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. We also compare our method with SVM classifier using popular benchmark data sets as a validation.

2 Method

2.1 Finding Unbounded Support Vectors

We go through standard procedure of SVM using input data and Gaussian kernel to get unbounded support vectors [13].

2.2 Constructing Trained SVM Decision Network

We can use the unbounded support vectors obtained in the previous step to construct a trained SVM decision network. Graphical representation of trained SVM decision network, which can be used to classify input data, is shown in Fig. 1.

2.3 Proof of Functional Equivalence of SVM and FRB

The purpose of this part is to formally prove that SVM decision network is equivalent to fuzzy IF-THEN rules which means SVM decision can be represented by fuzzy IF-THEN rules without loss in functionality.

Fuzzy IF-THEN Rules and Fuzzy Inference System Functions. There are two well-known types of fuzzy inference method. Mamdani's fuzzy inference method and Takagi-Sugeno (TS) method [11]. TS method can be further divided into zero-ordered and first-ordered type. The zero-ordered type contains only

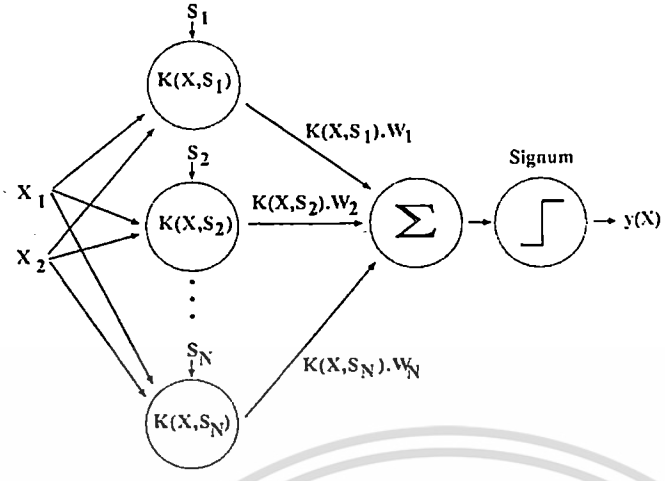


Fig. 1. A graphical representation of an SVM decision network is shown, where the leftmost X_i is input vector, S_i is unbounded support vector, $K(X, S_i)$ is Gaussian kernel function where X is each input vector, W_i is weight, and $y(X)$ is output decision

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) [11] which can be made functionally equivalent to Artificial Neural Networks (ANN). ANFIS is based on Adaptive Network 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 (Fig. 2). We can obtain fuzzy IF-THEN rules from the ANFIS which is derived from ANN [10]. 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 zero-ordered TS FRB will be shown to be equivalent to SVM using the following example 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 2: 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. 2, where the firing strength of i^{th} 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-norm with multiplication operator is:

$$M_i = \mu_{A_i}(X_1)\mu_{B_i}(X_2) \ . \tag{1}$$

Note that overall output can be chosen as the weighted sum of each rule's output:

$$f(X) = \sum_{i=1}^{R} M_i.w_i \tag{2}$$

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

$$y(X) = sign(f(X) + b) \ . \tag{3}$$

Required Conditions for Functional Equivalence. The functional equivalence between a trained SVM decision network (Fig. 1) and ANFIS (Fig. 2) 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 rule's 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.

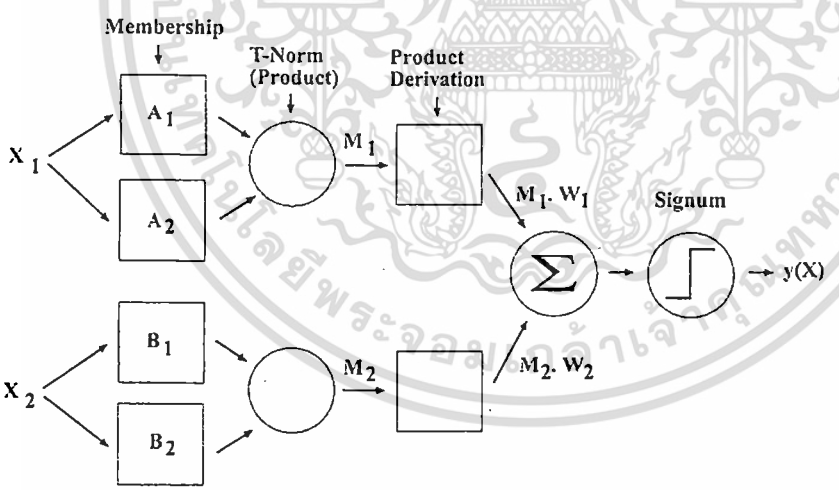


Fig. 2. An example of ANFIS equivalent to SVM, which uses Gaussian kernel function with two input vectors

Functions in SVM decision network. 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

$$f_1(X, S_i) = exp\left[\frac{-(\| X - S_i \|)^2)}{\sigma^2}\right], \quad i = 1, \ldots, N \,. \tag{4}$$

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

$$f_2(f_1(X, S_i), w_i) = w_i . f_1(X, S_i) \,. \tag{5}$$

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

$$f_3(X) = \sum_{i=1}^{N} f_{2_i}(X) \,. \tag{6}$$

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

$$f_4 = sign(f_3 + a) \,. \tag{7}$$

Let y be the output; we obtain

$$y = f_4(f_3(f_2(f_1(X, S_i)))) = f_4 \circ f_3 \circ f_2 \circ f_1(X, S_i) \,. \tag{8}$$

Functions in ANFIS. 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

$$\mu_{A_1}(x_1) = exp\left[\frac{-(x_1 - c_{A_1})^2}{\sigma_1^2}\right] \tag{9}$$

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

$$M_i = \mu_{A_i}(x_1)\mu_{B_i}(x_2) \tag{10}$$

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

$$g_1(X, S_i) = exp\left[\frac{-(\| X - S_i \|)^2}{\sigma^2}\right], \quad i = 1, \ldots, N . \tag{11}$$

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

$$g_2(g_1(X, S_i), w_i) = w_i . g_1(X, S_i) . \tag{12}$$

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

$$g_3(X) = \sum_{i=1}^{N} g_{2_i}(X) . \tag{13}$$

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

$$g_4 = sign(g_3 + a) . \tag{14}$$

Let z be the output; we obtain

$$z = g_4(g_3(g_2(g_1(X, S_i)))) = g_4 \circ g_3 \circ g_2 \circ g_1(X, S_i) . \tag{15}$$

The Proof of the Equivalence of SVM and ANFIS

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.

2.4 Rules Extraction Based on Firing Strength

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

In Fig. 3, 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 with the class label we want to identify ignoring the support vectors of the other class, 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. 3; X_i is input vector, and A_i is Gaussian kernel function of unbounded support vector and input vector.

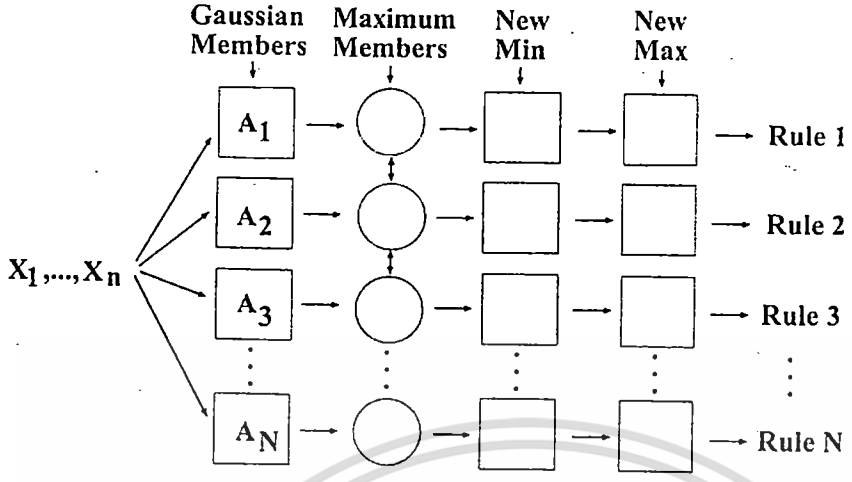


Fig. 3. Rule extraction algorithm based on kernel firing strength

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.

2.5 Refining Rules by Unbounded Support Vector Space Expansion

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 space expansion 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 (e.g., 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 $y = v_1$

3 Experimental Results

We perform classification using both SVM and our fuzzy IF-THEN rules on six benchmark data sets which can be downloaded from UCI Machine Learning Repository at http://www.ics.uci.edu. The six data sets are Iris [5], Wine [1], Wisconsin Breast Cancer [18], Habermans Survival Data [7], Ionosphere [16], and Spect Heart [14]. These data sets are chosen to represent different number of instances, number of classes, input data types, and number of attributes. Data Characteristics are data set name (Data Set), number of instances (N), number of classes (Class), data type (Type) which can be real (R), integer (I), or binary (B), and number of attributes (Attribute). Each data set is randomly split into ten parts. In rotation, nine parts are the training data and the remaining part is the testing data in a ten-fold cross validation technique except the Spect Heart which contains its own training and testing data separately. Procedures in the method section are then applied to these data.

Results from each data set are average number of unsigned support vectors (Avg USV), average percent error (Avg %Error) from SVM, average number of rules (Avg Rules) from our method, and average percent error from our method, and the classification results are shown in Table 1. SVM performs classification

Table 1. Comparison of Errors from SVM and Rules

Data Characteristics					SVM		Rules	
Data Set	N[a]	Class	Type	Attribute	Avg USV	Avg %Error	Avg Rules	Avg %Error
Iris	150	3	R	4	37.07	2.89	7.27	6.22
Wine	178	3	R	13	157.57	14.04	52.53	8.99
Wisconsin	699	2	I	10	300.6	5.27	67.5	4.83
Haberman	306	2	I	4	105.7	29.08	63.7	46.41
Ionosphere	351	2	I, R	34	278.0	37.04	165.4	6.84
Spect Heart	187	2	B	22	73.0	10.16	20.0	36.36

[a] Number of instances.

with less error in Iris, Haberman, and Spect Heart data sets than our method, but our method performs better in Wine, Wisconsin Breast Cancer, and Ionosphere data sets. Average number of rules in each data set is lower than number of unsigned support vectors as claimed by our method.

4 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 plus the fuzzy IF-THEN rules obtained can be easily incorporated into computer program.

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.

References

1. Aeberhard S., Coomans D., de Vel, O.: The Classification Performance of RDA. Tech. Rep., no. 92-01 (1992)
2. Andrews, R., Diederich, J., Tickle, A.B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-Based Systems 8(6), 373–389 (1995)

3. Barakat, N., Diederich, J.: Eclectic Rule-Extraction from Support Vector Machines. International Journal of Computational Intelligence 2(1), 59–62 (2005)
4. Diederich, J. (ed.): Rule Extraction from Support Vector Machines. SCI, vol. 80, pp. 3–30. Springer, Heidelberg (2008)
5. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annual Eugenics 7(Part II), 179–188 (1936)
6. Fung, G., Sandilya, S., Rao, R.B.: Rule extraction from linear support vector machines. In: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 32–40 (2005)
7. Haberman, S.J.: Generalized Residuals for Log-Linear Models. In: Proceedings of the 9th International Biometrics Conference, Boston, pp. 104–122 (1976)
8. Huysmans, J., Baesens, B., Vanthienen, J.: ITER: An algorithm for predictive regression rule extraction. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 270–279. Springer, Heidelberg (2006)
9. Huysmans, J., Setiono, R., Baesens, B., Vanthienen, J.: Minerva: Sequential Covering for Rule Extraction. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics 38(2), 299–309 (2008)
10. Jang, J.S.R., Sun, C.T.: Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE Trans. Neural Networks 4, 156–158 (1992)
11. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-Fuzzy and Soft Computing, pp. 333–342. Prentice Hall International, Englewood Cliffs (1997)
12. Kolman, E., Margaliot, M.: Are artificial neural networks white boxes? IEEE Trans. Neural Networks 16(4), 844–852 (2005)
13. Kumar, S.: Neural Networks: A Classroom Approach, International Edition, pp. 273–304. McGraw-Hill, New York (2005)
14. Kurgan, L.A., Cios, K.J., Tadeusiewicz, R., Ogiela, M., Goodenday, L.S.: Knowledge Discovery Approach to Automated Cardiac SPECT Diagnosis. Artificial Intelligence in Medicine 23(2), 149–169 (2001)
15. Nunez, H., Angulo, C., Catala, A.: Rule Extraction Based on Support and Prototype Vectors. SCI, vol. 80, pp. 109–134. Springer, Heidelberg (2008)
16. Sigillito, V.J., Wing, S.P., Hutton, L.V., Baker, K.B.: Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Technical Digest 10, 262–266 (1989)
17. Vapnik, V.N.: Statistical Learning Theory, pp. 375–520. John Wiley & Sons, Chichester (1998)
18. Wolberg, W.H., Mangasarian, O.L.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Proceedings of the National Academy of Sciences 87, 9193–9196 (1990)

APPENDIX B

Currency Crises Fuzzy IF-THEN Rules

The final form of fuzzy IF-THEN rules is in the form:

IF $x_1 \in [a_{1-}, a_{1+}]$ AND $x_2 \in [a_{2-}, a_{2+}]$ AND ... $x_{11} \in [a_{11-}, a_{11+}]$ THEN y = crisis occurs

where x_i is one of the 11 parameters from our data set; a_{i-} is the lower range of each parameter, and a_{i+} is the upper range of each parameter.

In order to save space, we present the rules in the format:

Rule no.	$[a_{1-}, a_{1+}]$	$[a_{2-}, a_{2+}]$	$[a_{3-}, a_{3+}]$
	$[a_{4-}, a_{4+}]$	$[a_{5-}, a_{5+}]$	$[a_{6-}, a_{6+}]$
	$[a_{7-}, a_{7+}]$	$[a_{8-}, a_{8+}]$	$[a_{9-}, a_{9+}]$
	$[a_{10-}, a_{10+}]$	$[a_{11-}, a_{11+}]$	

Final fuzzy rules for currency crises for Latin America:

1.	[0.06, 1.06],	[4.16, 10.66],	[0.00, 12.60],
	[1, 1],	[0.01, 0.41],	[-4.08, -0.18],
	[41, 171],	[-1.85, 11.15],	[-110, 226],
	[443, 3823],	[1, 1]	
2.	[-0.61, 1.09],	[2.88, 11.38],	[0.00, 14.40],
	[1, 1],	[-0.06, 0.44],	[-5.18, -0.08],
	[106, 276],	[-12.69, 3.31],	[-89, 247],
	[-1130, 3290],	[1, 1]	
3.	[-0.61, 0.49],	[1.87, 8.87],	[0.00, 12.60],
	[1, 1],	[-0.06, 0.44],	[-4.89, -0.69],
	[150, 290],	[-10.96, 2.04],	[-85, 251],
	[1767, 5147],	[1, 1]	

4.	[-0.60, 0.58],	[4.43, 10.33],	[0.00, 9.00],
	[1, 1],	[-0.07, 0.50],	[-3.33, 3.89],
	[10, 631],	[-1.79, 8.61],	[-84, 207],
	[-4396, -1348],	[1, 1]	
5.	[-0.68, 1.02],	[5.37, 15.37],	[0.00, 14.40],
	[0, 1],	[-0.05, 0.45],	[-6.89, -0.89],
	[114, 294],	[-6.09, 11.91],	[-94, 242],
	[-2030, 2650],	[1, 1]	
6.	[-0.60, 1.10],	[0.86, 56.36],	[0.00, 14.40],
	[0, 1],	[-0.06, 0.44],	[-14.60, 6.70],
	[379, 7229],	[-12.50, 11.50],	[-126, 210],
	[-15208, 26392],	[0, 1]	
7.	[-0.60, 1.10],	[1.09, 18.59],	[0.00, 14.40],
	[0, 1],	[-0.06, 0.44],	[-6.08, 6.82],
	[378, 988],	[-12.40, 11.60],	[-120, 216],
	[-9999, 5861],	[0, 1]	
8.	[-0.65, 1.05],	[1.01, 28.51],	[0.00, 14.40],
	[0, 1],	[0.00, 0.50],	[-14.68, 6.62],
	[-31, 459],	[-12.41, 11.59],	[-101, 235],
	[9126, 26286],	[0, 1]	
9.	[-0.65, 1.05],	[7.02, 17.02],	[0.00, 14.40],
	[1, 1],	[-0.06, 0.44],	[-10.36, -4.96],
	[-33, 117],	[-8.72, 9.28],	[-83, 253],
	[-2751, 1929],	[1, 1]	
10.	[-0.30, 0.64],	[1.28, 13.46],	[0.00, 7.20],
	[1, 1],	[-0.03, 0.45],	[-3.99, -0.04],
	[131, 398],	[-7.04, 1.44],	[-97, 198],
	[-1049, 1545],	[1, 1]	
11.	[-0.65, 1.05],	[0.83, 33.33],	[0.00, 14.40],
	[0, 1],	[-0.01, 0.49],	[-14.68, 6.62],
	[378, 1628],	[-12.20, 11.80],	[-81, 255],
	[-15143, 15797],	[0, 1]	

12.	[-0.61, 1.09],	[0.80, 56.30],	[0.00, 14.40],
	[0, 1],	[-0.02, 0.48],	[-14.59, 6.71],
	[377, 5007],	[-12.68, 12.32],	[-107, 229],
	[-15285, 26315],	[0, 1]	
13.	[-0.65, 1.05],	[1.01, 9.51],	[0.00, 14.40],
	[0, 1],	[-0.05, 0.45],	[-14.50, -10.90],
	[-31, 149],	[-7.26, 11.74],	[-128, 208],
	[-4647, 1593],	[0, 1]	
14.	[-0.61, 1.09],	[0.97, 8.97],	[0.00, 14.40],
	[0, 1],	[0.00, 0.40],	[-14.28, -7.68],
	[-30, 160],	[-3.03, 11.97],	[-116, 220],
	[-5744, -544],	[0, 1]	
15.	[-0.13, 0.37],	[3.87, 10.56],	[3.88, 15.73],
	[1, 1],	[0.15, 0.49],	[-8.84, 1.08],
	[-5, 50],	[-1.57, 5.36],	[-44, 132],
	[-545, 783],	[1, 1]	
16.	[-0.69, 0.51],	[1.12, 7.62],	[0.00, 14.40],
	[1, 1],	[-0.03, 0.47],	[-13.38, -9.18],
	[-24, 56],	[-9.15, 4.85],	[-124, 212],
	[-2644, 996],	[1, 1]	
17.	[-0.66, 0.14],	[2.44, 6.44],	[1.54, 14.14],
	[1, 1],	[0.01, 0.41],	[-7.69, -5.59],
	[-28, 32],	[5.37, 11.37],	[-114, 126],
	[-357, 1463],	[1, 1]	
18.	[-0.66, 0.94],	[0.70, 7.20],	[-0.11, 14.29],
	[1, 1],	[0.02, 0.42],	[-11.12, -6.92],
	[-32, 88],	[-4.89, 9.11],	[-82, 254],
	[-3282, 358],	[0, 0]	
19.	[-0.67, 0.83],	[0.72, 7.22],	[-0.95, 15.25],
	[0, 1],	[0.01, 0.41],	[2.51, 6.71],
	[-31, 89],	[-12.30, 3.70],	[-93, 243],
	[451, 5131],	[0, 0]	

20.	[-0.66, 1.04],	[5.25, 34.25],	[-0.46, 15.74],
	[0, 1],	[-0.02, 0.48],	[-14.59, 2.21],
	[-27, 343],	[-12.23, 11.77],	[-89, 247],
	[-15093, -7813],	[0, 1]	
21.	[-0.68, 1.02],	[3.26, 56.26],	[0.00, 14.40],
	[0, 1],	[-0.01, 0.49],	[-14.50, 6.80],
	[-25, 685],	[-12.63, 11.37],	[-88, 248],
	[9118, 26278],	[0, 1]	
22.	[-0.68, 1.02],	[0.69, 24.69],	[0.00, 14.40],
	[0, 1],	[0.02, 0.42],	[-14.55, 3.45],
	[-27, 383],	[-12.58, 11.42],	[-100, 236],
	[9133, 26293],	[0, 1]	
23.	[-0.63, 1.07],	[1.02, 41.02],	[0.00, 14.40],
	[0, 1],	[0.02, 0.42],	[-14.65, 6.65],
	[-33, 687],	[-12.17, 11.83],	[-114, 222],
	[9114, 26274],	[0, 1]	
24.	[-0.45, 0.75],	[0.77, 4.27],	[0.00, 10.80],
	[0, 0],	[-0.06, 0.44],	[-7.53, -4.53],
	[-24, 66],	[6.36, 11.36],	[-82, 254],
	[-1385, 1216],	[0, 0]	
25.	[-0.35, 0.65],	[0.76, 4.76],	[-0.31, 15.89],
	[0, 0],	[0.03, 0.43],	[-5.32, -2.32],
	[-30, 70],	[-7.02, 0.98],	[-90, 198],
	[-989, 1091],	[1, 1]	
26.	[-0.28, 0.10],	[0.91, 3.67],	[0.00, 3.37],
	[0, 0],	[-0.06, 0.15],	[-7.05, -3.18],
	[6, 44],	[-1.00, 7.35],	[-28, 47],
	[-238, 555],	[1, 1]	
27.	[-0.69, 1.11],	[3.86, 21.36],	[0.00, 14.40],
	[0, 1],	[-0.03, 0.47],	[-10.50, 0.00],
	[367, 717],	[-11.70, 9.30],	[-106, 230],
	[-4467, 4633],	[0, 1]	

28.	[-0.69, 1.01],	[0.77, 56.27],	[0.00, 14.40],
	[0, 1],	[-0.04, 0.46],	[-14.65, 6.65],
	[383, 3393],	[-11.70, 12.30],	[-97, 239],
	[-15320, 26280],	[0, 1]	
29.	[-0.63, 1.07],	[0.95, 55.95],	[0.00, 14.40],
	[0, 1],	[-0.04, 0.46],	[-14.80, 6.80],
	[375, 7235],	[-12.14, 11.86],	[-99, 237],
	[-15310, 26290],	[0, 1]	
30.	[-0.64, 1.06],	[21.90, 55.90],	[-0.30, 14.10],
	[0, 1],	[-0.01, 0.49],	[-14.53, 6.77],
	[-31, 809],	[-11.76, 12.24],	[-125, 211],
	[-15254, 17506],	[0, 1]	
31.	[-0.32, 0.28],	[1.74, 4.74],	[-1.76, 7.24],
	[0, 0],	[-0.06, 0.44],	[-3.53, -1.73],
	[-22, 38],	[-5.39, -1.39],	[0, 192],
	[-247, 793],	[1, 1]	
32.	[-0.11, 0.45],	[1.50, 7.73],	[7.39, 11.26],
	[1, 1],	[0.02, 0.46],	[1.28, 6.55],
	[-3, 83],	[-4.76, 2.10],	[-83, 63],
	[-48, 2448],	[1, 1]	
33.	[-0.45, 0.13],	[2.15, 5.01],	[0.00, 3.60],
	[1, 1],	[0.06, 0.46],	[-5.47, -3.25],
	[-24, 42],	[3.07, 7.59],	[-70, 66],
	[-1736, 338],	[1, 1]	
34.	[-0.09, 0.57],	[1.48, 5.53],	[0.00, 3.60],
	[1, 1],	[-0.03, 0.50],	[-10.22, -4.24],
	[30, 166],	[4.84, 9.00],	[-97, 55],
	[120, 2400],	[1, 1]	
35.	[-0.13, 0.47],	[0.85, 3.85],	[0.00, 5.40],
	[0, 0],	[-0.05, 0.45],	[-4.22, -2.42],
	[-3, 57],	[-2.75, 3.25],	[-118, 122],
	[210, 1510],	[0, 0]	

36.	[0.02, 0.60],	[1.16, 2.35],	[0.00, 3.79],
	[0, 0],	[-0.05, 0.21],	[-4.65, 2.90],
	[17, 168],	[1.17, 4.97],	[-105, 63],
	[-3313, 20],	[1, 1]	

Final fuzzy rules for currency crises for Asia:

1.	[-0.30, 0.30],	[51.91, 88.41],	[0.00, 9.00],
	[0, 1],	[-0.09, 3.81],	[-8.47, 15.83],
	[-16, 53],	[-8.19, 8.81],	[-102, 137],
	[-3775, 20924] ,	[0, 1]	
2.	[-0.31, 0.29],	[0.56, 9.56],	[-1.00, 8.00],
	[1, 1],	[-0.05, 0.95],	[-8.34, -2.94],
	[-13, 46],	[-7.90, 9.10],	[-69, 122],
	[-3761, -1681],	[1, 1]	
3.	[-0.22, 0.28],	[0.92, 7.92],	[-1.66, 9.14],
	[1, 1],	[-0.02, 0.78],	[-6.63, -2.43],
	[-12, 47],	[-1.04, 8.96],	[-82, 157],
	[-3955 ,-835],	[1, 1]	
4.	[-0.27, 0.23],	[0.71, 13.21],	[0.00, 9.00],
	[0, 1],	[0.00, 1.50],	[-6.17, 1.63],
	[-11, 48],	[-8.30, 8.70],	[-74, 117],
	[8070, 14830],	[0, 1]	
5.	[-0.32, 0.14],	[2.95, 22.78],	[0.00, 1.80],
	[0, 0],	[0.01, 0.19],	[-2.40, 4.43],
	[10, 29],	[0.79, 6.50],	[-60, 81],
	[-3697, 995],	[1, 1]	
6.	[-0.29, 0.21],	[2.40, 11.40],	[0.00, 9.00],
	[1, 1],	[-0.06, 1.04],	[-8.57, -3.77],
	[-14, 55],	[-1.94, 9.06],	[-90, 149],
	[-4004, -1144],	[0, 0]	

7.	[-0.26, 0.24],	[0.71, 51.71],	[0.00, 9.00],
	[0, 1],	[0.00, 3.80],	[-8.48, 15.82],
	[-16, 53],	[-7.99, 9.01],	[-75, 116],
	[9719, 33119],	[0, 1]	
8.	[0.01, 0.23],	[3.69, 27.27],	[0.00, 1.80],
	[0, 0],	[0.01, 0.29],	[-5.93, -2.64],
	[8, 39],	[2.55, 7.32],	[5, 48],
	[131, 7279],	[1, 1]	
9.	[-0.25, 0.25],	[9.07, 20.07],	[0.00, 9.00],
	[0, 1],	[-0.04, 1.16],	[-8.34, -5.04],
	[-18, 51],	[-8.13, 8.87],	[-92, 147],
	[636, 6096],	[0, 1]	
10.	[-0.27, 0.23],	[0.93, 10.93],	[0.00, 9.00],
	[1, 1],	[0.00, 1.10],	[-2.82, 2.88],
	[-13, 46],	[-7.31, 2.69],	[-89, 150],
	[-1886, 3053],	[1, 1]	
11.	[-0.30, 0.29],	[8.92, 16.44],	[0.00, 7.20],
	[1, 1],	[-0.05, 0.48],	[-6.98, -3.84],
	[-12, 50],	[0.04, 8.75],	[-65, 150],
	[-1483, 439],	[1, 1]	
12.	[-0.21, 0.29],	[3.43, 7.43],	[0.00, 7.20],
	[0, 0],	[-0.07, 0.43],	[-5.80, -3.40],
	[-18, 51],	[0.39, 8.39],	[-82, 157],
	[-2633, -1073],	[0, 0]	
13.	[-0.29, 0.21],	[0.95, 9.95],	[0.00, 9.00],
	[0, 1],	[-0.04, 1.06],	[-8.28, -2.28],
	[-10, 49],	[-4.81, 9.19],	[-77, 114],
	[2439, 7379],	[1, 1]	
14.	[-0.27, 0.30],	[1.02, 5.09],	[-1.66, 9.72],
	[1, 1],	[2.27, 3.74],	[-2.12, 15.63],
	[-8, 49],	[-7.44, 8.51],	[-75, 118],
	[-938, 5272],	[1, 1]	

15.	[-0.32, 0.28],	[10.77, 20.77],	[0.00, 9.00],
	[1, 1],	[-0.08, 1.02],	[-8.62, -2.92],
	[-14, 55],	[-6.70, 9.30],	[-105, 134],
	[-2667, 2272],	[0, 0]	
16.	[-0.27, 0.23],	[0.86, 7.86],	[0.00, 9.00],
	[1, 1],	[-0.03, 1.07],	[-8.37, -5.97],
	[-16, 54],	[-1.10, 8.90],	[-105, 134],
	[602, 4242],	[0, 0]	

Appendix C

MATLAB Code for Nonlinear SVM

C.1 MATLAB Program Description

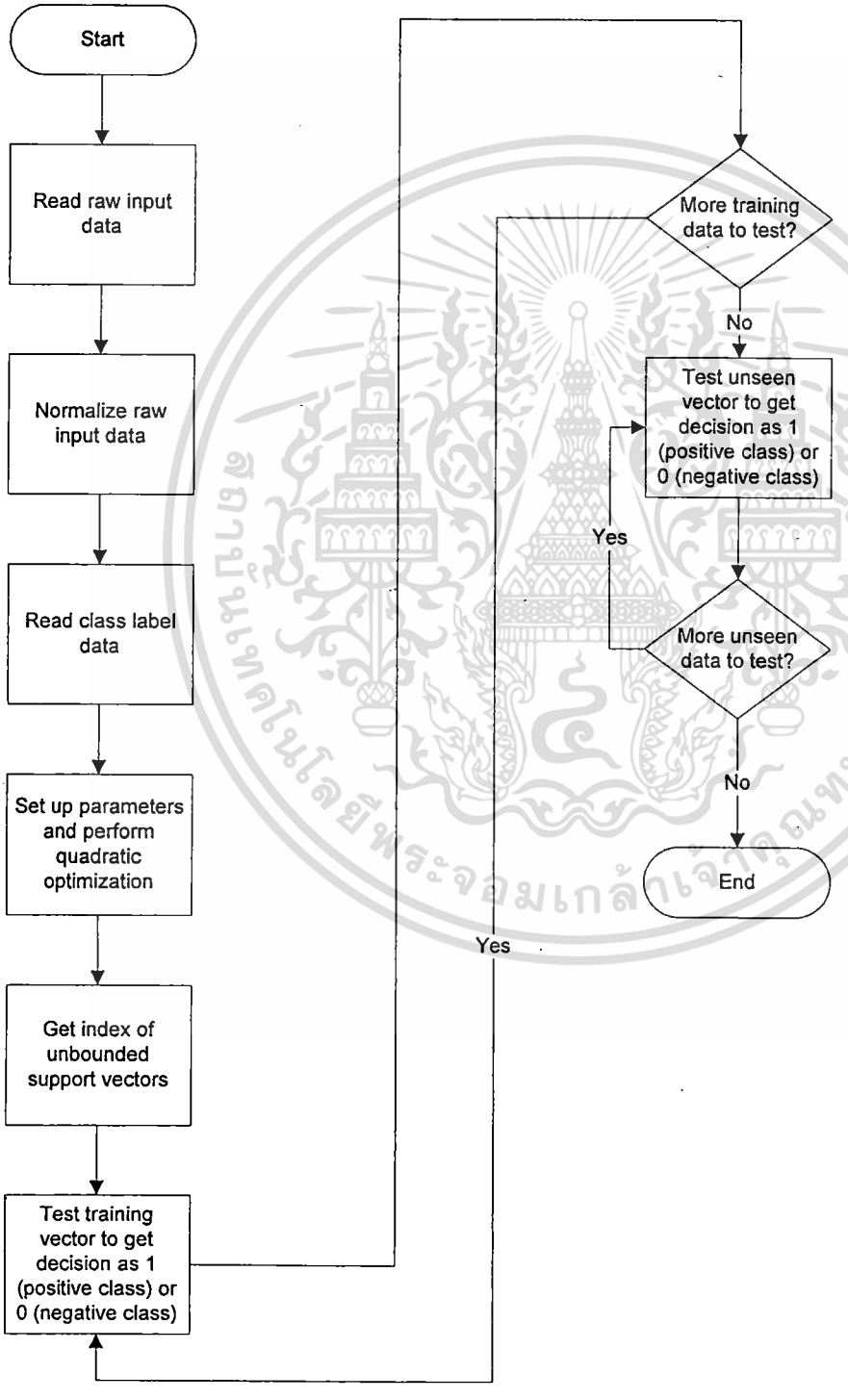
We describe how we implement procedures from section 3.1.1 through 3.1.7 with MATLAB in this section; the flowchart, list of variables, and code will follow in later sections.

Initially, we define input dimension of the data set being used and store it in inputDimension. We then read in N rows of raw samples of the matrix X_r and store it in rawInput. We transpose rawInput and store raw input in inputTransTemp. We normalize raw input and store it in normAll. We calculate the rows to be used as training subset and store it in normTrain. We calculate the rows to be used as testing subset and store it in normTest. Corresponding to each data point, the desired class is indicated in the vector D; our class label data are already in the form $+1$ and -1. We read in class label data and store it in DAll. We use the same subset of training data as in normTrain and store it in DTrain. We use the same subset of testing data as in normTest and store it in DTest. The size of the data set is stored in q, and a small threshold epsilon is defined for the detection of support vectors. The penalty parameter C to balance between empirical risk and structural risk has to be predefined to get the lowest errors. The Hessian matrix $\mathbf{H}$ is formed by the Gaussian kernel of datapoints multiplied into their respective classes. Since the standard MATLAB quadratic optimizer minimizes an objective function, we negate the dual Lagrangian.

In accordance with the details above, we set f = -ones(q,1); the number of equality constraints numeqconstraints = 1; the matrix $A = D^T$ with only one row since there is only one constraint; and b = 0. Before optimizing, we need to set the upper and lower bounds for each of the Lagrange multipliers. This is specified in q x 1 column vector, vub, set to infinity and column vector, vlb, set to zero. An initial point x_0 is specified to 0. Performing the optimization yields the optimized Lagrange multipliers vector lambda, the value of the function at the minimum, and the optimality status. Finally, since the decision region calculations require lambda, H and the optimal bias, what we now need is the optimal bias. This is done by averaging over all the support vectors. The svindex stores all indices of lambda greater than zero (bounded support vectors);

usvindex stores unbounded support vectors; ns is the number of support vectors; and w_0 is the optimal bias. After training, we perform testing on the testing data subset. We obtain the decision results and calculate SVM percent errors.

C.2 Program Flowchart



C.3 List of Variables in MATLAB Program

The variables used in the program are shown in Table C.1.

Table C.1 List of variables used in MATLAB program.

No.	Variable Name	Description
1	inputDimension	Dimension of input data
2	fid	File pointer
3	rawInput	Raw input data
4	inputTransTemp	Temporary input data to be normalized
5	column(:,i)	Temporary column to be normalized
6	my_min(i)	Min of a column to be normalized
7	my_max(i)	Max of a column to be normalized
8	columnNorm(:,i)	Normalized column
9	normAll	All normalized input data
10	normTrain	Normalized training data
11	normTest	Normalized test data
12	fidD	Class label data file pointer
13	DTemp	Temporary raw class label data
14	DAll	All class label data
15	DTrain	Training class label data
16	DTest	Test class label data
17	q	Total number of training data rows
18	C	Penalty parameter*
19	epsilon	Tolerance for Support Vector detection**
20	H	Hessian matrix
21	f	Column matrix of -1
22	vlb	Lower bound of lambdas***
23	vub	Upper bound for lambdas****
24	x0	Initial column input vector*****
25	numeqconstraints	Number of equality constraints
26	A	Setup column for equality constraints
27	b	Bias value
28	alpha	Index of input data
29	lambda	Values of Lagrange multipliers
30	svindex	Index of support vectors
31	usvindex	Index of unbounded support vectors
32	numUSV	Number of unbounded support vectors
33	zTrain	Discriminant value of training input vector
34	decisionTrain	Output decisions of all training input data
35	TrainDecision	Transpose of training output decisions
36	zTest	Discriminant value of each test input vector
37	decisionTest	Output decisions of all test input data
38	TestDecision	Transpose of test output decisions

* Set to 0.1, 1, 10, 100, 1,000, and 10,000 to find optimal result;

** Set to e^{-6}; *** Set to 0; **** Set to C; ***** Set to 0.

C.4 MATLAB Code

```
1.  % ==========================================================
2.  %   MATLAB code that implements SVM for non-linearly separable data
3.  %   using Gaussian kernel and 10-fold Cross Validation
4.  % ==========================================================
5.  clear;
6.  clc;
7.  inputDimension = 11;
8.  fid = fopen('RawInputLatin.txt','r');          %Read in raw input data
9.  rawInput = fscanf(fid, '%f %f', [207 inf]);
10. fclose(fid);
11. inputTransTemp = rawInput;

12. % Split into 10 subsets: 1-20, 21-40, 41-60, 61-81, 82-102, 103-123, 124-144, 145-165,
13. %166-186, 187-207

14. for i = 1:inputDimension                       %Data normalization
15.     column(:,i) = inputTransTemp(:,i);
16.     my_min(i) = min(column(:,i));
17.     my_max(i) = max(column(:,i));
18.     columnNorm(:,i) = ((column(:,i) - my_min(1,i))/(my_max(1,i) - my_min(1,i))*2) - 1;
19. end %for
20. normAll = columnNorm;
21. normTrain = normAll(21:207,:);                 %Training data
22. normTest = normAll(1:20,:);                    %Test data

23. fidD = fopen('DNormLatin.txt','r');            %Read in class label data
24. DTemp = fscanf(fidD, '%f', [1 inf]);
25. DAll = DTemp';
26. DTrain = DAll(21:207,:);                       %Training label data
```

```
27.  DTest = DAll(1:20,:);                %Test label data

28.  q = size(normTrain,1);               %Number of training samples
29.  C = 100;                             %Choose penalty parameter value

30.  epsilon = C*1e-6;                    %tolerance for Support Vector Detection

31.  H = zeros(q,q);                      %Initialize Hessian matrix
32.  for i=1:q                            %Set up the Hessian
33.      for j=1:q
34.          H(i,j) =  DTrain(i)*DTrain(j)*(exp(-((sqrt((normTrain(i,:)-normTrain(j,:))*
             (normTrain(i,:)- normTrain(j,:))'))^2)/((0.5)^2)));
35.      end %for
36.  end %for

37.  f = -ones(q,1);

38.  H = H+1e-10*eye(size(H));            %Add small amount of zero order regularisation to
39.                                       %avoid problems when Hessian is badly conditioned.

40.  %Parameters for the Optimization problem
41.  vlb = zeros(q,1);                    %Lower bound of lambdas = 0
42.  vub = C*ones(q,1);                   %Upper bound C
43.  x0 = zeros(q,1);                     %Initial point is 0
44.  numeqconstraints = 1;                %Number of equality constraints
45.  A = DTrain';                         %Set up the equality constraint
46.  b = 0;

47.  %Solve the Optimization Problem
48.  [alpha lambda how] = qp(H, f, A, b, vlb, vub, x0, numeqconstraints);

49.  svindex = find( alpha > epsilon);
```

```
% Find unbounded support vectors
usvindex = find( alpha > epsilon & alpha < (C - epsilon));
numUSV = length(usvindex)
if length(usvindex) > 0
    b0 =  (1/length(usvindex))*sum(DTrain(usvindex) - H(usvindex,svindex)*
        alpha(svindex).*DTrain(usvindex));
else
    fprintf('No support vectors on margin - cannot compute bias.\n');
end %if

for j = 1 : length(DTrain)          %Start in-sample testing
    zTrain = b0;
    inputTrain = normTrain(j,:);
    m = 0;
    for i = 1 : length(DTrain)
        if (abs(alpha(i)) > epsilon)
            m = m +1;
            zTrain = zTrain + DTrain(i)*alpha(i)*(exp(-((sqrt((inputTrain-
                normTrain(i,:))*(inputTrain-normTrain(i,:))'))^2)/((0.5)^2)));
        end % if
    end  % for i
    decisionTrain(j) = sign(zTrain);
end % for j
TrainDecision = decisionTrain'    %End in-sample testing

for j = 1 : length(DTest)           %Start out-of-sample testing
    zTest = b0;
    inputTest = normTest(j,:);
    m = 0;
    for i = 1 : length(DTrain)
        if (abs(alpha(i)) > epsilon)
            m = m +1;
```

```
78.             zTest = zTest + DTrain(i)*alpha(i)*(exp(-((sqrt((inputTest-normTrain(i,:))*
                        (inputTest-normTrain(i,:))'))^2)/((0.5)^2)));
79.             end %if
80.         end %for i
81.         decisionTest(j) = sign(zTest);
82.     end % for j
83.     TestDecision = decisionTest'
84.     end  %End out-of-sample testing
```

Appendix D

Glossary

ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
ERM	Empirical Rick Minimization
FIS	Fuzzy Inference System
KKT	Karush-Kuhn-Tucker
NN	Neural Network
OCHP	Optimal Canonical Hyperplane
QP	Quadratic Programming
RBFN	Radial Basis Function Network
SLT	Statistical Learning Theory
SRM	Structural Risk Minimization
SV	Support Vector
SVM	Support Vector Machine
TS FRB	Takagi-Sugeno Fuzzy Rule Base
USV	Unbounded Support Vector
VC Dimension	Vapnik-Chervonenkis Dimension

BIOGRAPHY

Prasan Pitiranggon received a bachelor and a master degree in electrical engineering from the City College of New York in 1991 and 1993. He then came back to work with several private companies in Thailand before deciding to go back to study a doctorate degree in computer science at King Monkut's Institute of Technology Ladkrabang in 2006. He has presented his research work at the International Conference in Engineering, Applied Sciences, and Technology in 2007. And he has joined the 3rd Asian Conference on Intelligent Information and Database Systems in Korea in April 2011 to do a presentation of another one of his research work.