STUDY ON MULTI-SVM SYSTEMS AND THEIR APPLICATIONS TO PATTERN RECOGNITION

LI, Boyang

Graduate School of Information, Production and Systems
Waseda University

July, 2010

To my parents and my wife,
for all those times you stood by me;
for all the truth that you made me see;
for all the joy you brought to my life;
for all the wrong that you made right;
for every dream you made come true;
for all the love I found in you.

I'll be forever thankful for you.

Abstract

Kernel-based techniques represent a major development in machine learning algorithms. As one of the most widely used kernel-based technique, Support Vector Machine (SVM) has attracted a lot of interest from researchers in recent years. SVMs are a group of supervised learning methods that can be applied to pattern recognitions (classification or regression), and numerous books are available for an in-depth overview of the theoretical basis of these algorithms.

In comparison with other methods in most current literatures, the SVM-based methods show advantages in terms of generalization performance and the fitness accuracy. However, there still exist some challenges when applying SVMs to the real-world pattern recognition problems. Four main challenges are the large size of the training data, the high dimension of the input, the presence of noises and interaction, and the imbalance of data.

In order to handle the aforementioned challenges inherent in the real-world classification and regression problems, in this thesis, three improved approaches based on SVM are mainly developed in three important stages (preprocessing/dimensionality reduction/recognition) of the typical pattern recognition system respectively: *Fast SVM Training Based on Separation Boundary Detection*, *Feature Selection Using Correlation-based SVM Filter* and *Classification Using Fuzzy Decision-making SVM*.

Fast SVM training method based on the separation boundary detection technique is proposed for solving the challenge of large training data in the preprocessing stage. Two main advantages of this method is scaling down the training data obviously and reserving Support Vectors (SVs) maximally. In this approach, the separation boundary detection technique is first introduced to extract samples near the separation boundary. Second, to avoid the overfitting and preserve the structure properties, the clustering is implemented on the whole training data. The reconstructed training data consists of samples selected by the separation boundary detector and centroids of clusters. It is much smaller than the original training data, and makes the SVM training process much faster. Because the samples extracted by the separation boundary detection are likely to be SVs, the proposed approach will not degrade the classification accuracies.

Feature selection approach based on correlation-based feature clustering and SVM-based feature ranking is proposed for solving the challenge of high dimensional input. This method focuses on finding a good feature set consisting of features that are highly correlated with the output, yet uncorrelated with each other. For evaluating the influence quantities of features on the output, this work defines a feature sensitivity in SVM as the criterion. Features are ranked based on their sensitivities, and the ones with low sensitivities are eliminated. Then, a correlation-based clustering is adapted based on Affinity Propagation (AP) and implemented to divide the features into some clusters. Then, the features that are ranked first in the clusters are selected as the significant features. In the correlation-based clustering, correlation coefficient and information gain function are used as the criterions for evaluating the feature correlation.

Classification model using a fuzzy decision-making separation boundary is proposed for solving the challenges of noises, interaction and imbalance of data. Considering the vagueness between the real-world data sets caused by noises and interactions, a fuzzy decision-making function is first designed to take the place of the classical sign function in the prediction part of SVM. This flexible design of decision-making separation boundary is more similar to the real-world environment than conventional SVMs. Then an offset parameter is introduced into the fuzzy decision-making function to modify the excursion of separation boundary caused by the imbalance of data. The offset is calculated as the Weighted Harmonic Mean (WHM) of the decision values of SVs. This method can reduce the influence of noises on the calculation of offset.

Since it has been shown by several researchers that multiple pattern recognition systems can result in effective solutions to difficult real-world tasks, this work proposes two multiple SVM pattern recognition systems based on the improved single SVMs: *Multiple Support Vector Classifier System* and *Multiple Support Vector Regression Network.*

In the proposed Multiple Support Vector Classifier System, the task of classifier modeling is formulated as the piecewise approximation of the separation boundary. The proposed multiple SVM classifier system is constructed based on piecewise partition and interpolation. First, the separation boundary detection method is used to identify the separation boundary. Second, the separation boundary is partitioned into some subsets based on the rough set theory. Based on the partition of separation boundary, training data sets are constructed to describe the subtasks for identifying segments of the separation boundary. Then, local SVMs are subsequently trained to solve the respective subtasks. Finally, the decisions of the local SVMs are appropriately combined based on a probabilistic interpretation to obtain the final classification decision.

Modular SVR Network is proposed for time series prediction. As same as Multiple Support Vector Classifier System, it is also constructed based on the principle of ”divide-and-conquer”. The proposed system is constructed based on the price domain partition. The price domain of time series is first partitioned into several sub price regions. Then, the properties of these sub price regions are described by their respective SVRs. These SVRs are used as ”local experts”. Finally, outputs of the SVRs are combined by a combinator. Because the partition of training data is implemented in the output space (price domain) which has a much smaller dimensionality than the input space, the model is easier to be constructed.

Preface

The common theme of this thesis is developing some improved SVM approaches and multi-SVM systems for real-world pattern recognition problems. The material is organized in seven chapters. Most of the material has been published or considered to publish in journal papers and conference papers.

The material in Chapter 2 can be found in

- Boyang Li, Qiangwei Wang and Jinglu Hu, "A Fast SVM Training Method for Very Large Datasets", in *Proc. of IEEE International Joint Conference on Neural Networks 2009 (IJCNN 2009)*, pp. 1784-1789, Atlanta, Georgia, USA, 2009.

which has been extended into a journal paper

- Boyang Li, Qiangwei Wang and Jinglu Hu, "Fast SVM Training Using Edge Detection on Very Large Datasets", submitted to *IEEJ Transactions on Electrical and Electronic Engineering (TEEE)*, 2009.

The material in Chapter 3 can be found in

- Boyang Li, Qiangwei Wang and Jinglu Hu, "Feature Subset Selection: A Correlation-based SVM Filter Approach", in *IEEJ Transactions on Electrical and Electronic Engineering (TEEE), Vol.6, No.2*, 2010.

The materials in Chapter 4 can be found in

- Boyang Li, Jinglu Hu and Kotaro Hirasawa, "Support Vector Machine Classifier with WHM Offset for Unbalanced Data", in *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol.12, No.1*, pp. 94-101, 2008.

- Boyang Li, Jinglu Hu and Kotaro Hirasawa, "An Improved Support Vector Machine with Soft Decision-Making Boundary", in *Proc. of IASTED International Conference on Artificial Intelligence and Applications (AIA2008)*, pp. 40-45, Innsbruck, Austria, 2008.

- Boyang Li, Liangpeng Ma, Jinglu Hu and Kotaro Hirasawa, "Gene classification using an improved SVM classifier with soft decision boundary", in *Proc. of SICE Annual Conference 2008*, pp. 2476-2480, Tokyo, Japan, 2008.

- Boyang Li, Jinglu Hu and Kotaro Hirasawa, “Fuzzy Decision-making SVM with An Offset for Real-world Lopsided Data Classification”, in *Proc. of SICE-ICASE International Joint Conference 2006*, pp. 143-148, Pusan, Korea, 2006. (Young Author Award)
- Boyang Li, Jinglu Hu, Kotaro Hirasawa, Pu Sun and Kenneth Marko, “Support Vector Machine with Fuzzy Decision-Making for Real-world Data Classification”, in *Proc. of IEEE World Congress on Computational Intelligence 2006 (WCCI 2006) – International Joint Conference on Neural Networks 2006 (IJCNN 2006)*, pp. 1314-1319, Vancouver, Canada, 2006.

The material in Chapter 5 has been extended into a journal paper

- Boyang Li, Qiangwei Wang and Jinglu Hu, “Multi-SVM Classifier System with Piecewise Interpolation”, submitted to *IEEJ Transactions on Electrical and Electronic Engineering (TEEE)*, 2010.

The material in Chapter 6 has been presented in

- Boyang Li, Jinglu Hu and Kotaro Hirasawa, “Financial Time Series Prediction Using A Support Vector Regression Network”, in *Proc. of IEEE World Congress on Computational Intelligence 2008 (WCCI 2008) – International Joint Conference on Neural Networks 2008 (IJCNN 2008)*, pp. 621-627, Hong Kong, China, 2008.

which is extended into a journal paper

- Boyang Li, Qiangwei Wang and Jinglu Hu, “A Modular SVR Network for Forex Prediction”, to be submitted to *IEEJ Transactions on Electrical and Electronic Engineering (TEEE)*, 2010.

Acknowledgements

After four years, at Waseda university, as a doctoral student, I have learned one thing – I could never have done any of this, particularly the research and writing that went into this dissertation, without the support and encouragement of a lot of people.

First, I would like to thank my supervisor, Professor Jinglu Hu. You've been my friend, my mentor, my confidant, my colleague, and a never-ending fount of moral support. You have given so much of yourself to help me succeed. If I do take the academic path, I only hope that I can be half the supervisor that you have been to me. Whatever path I do take, I will be prepared because of you.

I am also very grateful to Professor Kotaro Hirasawa and Professor Sei-ichiro Kamata who proofread all the thesis. Many thanks are for their valuable comments.

I am also thankful for Professor Kotaro Hirasawa and Professor Fuliang Yin for their guidance, invaluable advices and comments on both my research and my future research career plans.

I am indebted to Dr. Shingo Mabu, Dr. Kaoru Shimada, Dr. Jing Zhou, and Ms. Lu Yu for their constructive and fruitful discussions in the start of my research.

I've also been fortunate to have a great group of friends at Waseda University, Dalian University of Technology (DLUT) and Kitakyushu Science and Research Park (KSRP). This includes my classmates in Neural Computing Lab at Waseda Univ. (Jun Zhang, Yang Chen, Benhui Chen, Wenxiang Dou, Qiangwei Wang, Yu Cheng, Lan Wang, Bo Zhou et al.), my best friends in Digital Signal Processing Lab at DLUT (Lei Yang, Hui Ma, Bin Jiao, Yanfei Cheng, et al.), the hardcore Happy Hour crew in KSRP (Zhan Gao, Ting Sun, Miao Yu, Yiding Zhang, Yupeng Zhang, Ting Ma, et al.), and many other students and faculty, too numerous to name. Not only are you the people I can discuss my research with and goof off with, but also you are confidants who I can discuss my troubles with and who stand by me through thick and thin. This, I believe, is the key to getting through a Ph.D. program – having good friends to have fun with and complain to.

I would also like to express my thanks to my research group (Qiangwei Wang, Zhifeng Zhang and Liangpeng Ma). I have learned so much from all of you, from figuring out what research is, to choosing a research agenda, to learning how to present my work. Your constructive criticism and collaboration have been tremendous assets throughout my Ph.D.

Finally, I would like to dedicate this work to my family: Songsen Li, Jing Chen and Yu Lian. Without your unending support and love, I never would have made it through this process or any of the tough times in my life. Thank you.

Kitakyushu, Japan Boyang Li
March 29, 2010

Table of Contents

List of Tables

List of Figures

Glossary

Some notations may have different meaning locally.

Notations

x^T	transpose
b	bias
α_k	Lagrange multipliers
$\#SV$	number of support vectors
$\mathcal{D}$	input space
Ω	output space
ε	threshold in feature selection
p	preferences parameter in Affinity Propagation
δ	offset parameter
β_i	Blackman weights in the calculation of WHM offset
σ^2	common width of RBF kernel
d	exponential quantity of the polynomial kernel
ψ_i	output of mean filter
ϕ_i	output of median filter
R	mathematical equivalence relation in rough set theory
μ_i	centers of training data sets corresponding to subtasks
P_i	probabilities
f	statistics in 5×2 cv F test

Operators and Functions

sign(x)	an odd mathematical function that extracts the sign of x
$\mathcal{L}(\cdot)$	Lagrangian function in construction of optimization problems
$\varphi(\cdot)$	mapping function from the input space to the feature space, $\varphi(\cdot) : \mathcal{D} \to \Omega$
$K(x_i, x_k)$	kernel function in SVM, $K(x_i, x_k) = \varphi(x_i) \cdot \varphi(x_k)$
$\sum$	sum
$y(x)$	classifier function
$f(x)$	decision value function
$g(x)$	final output of multiple SVM classifier system

$f_i(t)$	target outputs of the transformation layer in modular SVR network
$\hat{f}_i(t)$	actual outputs of the transformation layer in modular SVR network
$\eta(x\|f(x))$	sensitivity function of function $f(x)$ to variable x
$s(i,k)$	similarity function in Affinity Propagation
$r(i,k)$	responsibility function in Affinity Propagation
$a(i,k)$	availability function in Affinity Propagation
$C(X,Y)$	correlation coefficient of X and Y
$H(X)$	entropy of a variable X
$IG(X\|Y)$	information gain function

Abbreviations

ANN	Artificial Neural Networks
AP	Affinity propagation
ARIMA	Auto-Regressive Integrated Moving Average
BD-FSVM	Separation Boundary Detection Based Fast Support Vector Machine
CD	Correct Down Trend
CP	Correct Up Trend
CSF	Correlation-based Support Vector Machine Filter
Forex	Foreign Exchange
GA	Genetic Algorithms
GFS	Gradient-based Feature Selection
IG	Information Gain
kNN	k Nearest Neighbor
LDA	Linear Discriminant Analysis
LSVM	Lagrangian Support Vector Machine
MA	Moving Average
MAE	Mean Absolute Error
PCA	Principal Component Analysis
PMSVC	Piecewise Interpolation Based Multi-SVM Classifier System
PSO-SVM	Particle Swarm Optimization - Support Vector Machine
PTA	Plus and Take Away
QDA	Quadratic Discriminant Analysis
QP	Quadratic Programming
RBF	Radial Basis Function
RIS	Resource Information System
RSVM	Reduced Support Vector Machine
RTS-SVM	Reduced Training Set - Support Vector Machine
RVM	Relevance Vector Machines
SFS	Sequential Forward Search
SSE	Sum of Square Error
SV	Support Vector
SVM	Support Vector Machine
SVR	Support Vector Regression
WSVM	Weighted Support Vector Machine
WHM	Weighted Harmonic Mean

Chapter 1

Introduction and Motivation

1.1 Pattern Recognition

Pattern recognition has become more and more popular and important to us since 1960s and it induces attractive attention coming from a wider areas. Pattern recognition is not unfamiliar with everyone, it is a subject researching object description and classification method. Pattern recognition is also a collection of mathematical, statistical, heuristic and inductive techniques of fundamental role in executing the tasks like human being on computers. In a sense, pattern recognition is figuring out actual problems via mathematical methods.

1.1.1 Pattern recognition methods

Pattern recognition includes a lot of methods which impelling the development of numerous applications in different fields. The practicability of these methods is intelligent emulation.

- **Statistical pattern recognition:** Statistical decision and estimation theories is a classical method of pattern recognition which was found out during a long developing process. It is based on the feature vector distributing which getting from probability and statistical model [16]. In detail, in statistical pattern recognition, we put the features in some optional order, and then we can regard the set of features as a feature vector. Also statistical pattern recognition deals with features only without consider the relations between features.

- **Data clustering:** Its aim is to find out a few similar clusters in a mass of data which need not any information of the known clusters. It is an unsupervised method. In general, the method of data clustering can be partitioned two classes, one is hierarchical clustering, and the other is partition clustering.

- **Fuzzy sets:** The recognition process is often fuzzy and uncertain. And in reality, we can't always give complete answers or classification, so theory of fuzzy sets come into being. Fuzzy sets can describe the extension and intension of a concept effectively. Two principles proposed by Marr (1982) [17] and Keller (1995) [18] which can be think as the general role of fuzzy sets in pattern recognition. The pattern recognition system based on fuzzy sets theory can imitate thinking process of human being widely and deeply.

- **Neural networks:** Neural networks are developing very fast since the first neural networks model MP was proposed since 1943, especially the Hopfield neural networks and famous BP arithmetic came into being after. This approach applies biological concepts to machines to recognize patterns. The outcome of this effort is the invention of artificial neural networks which is set up by the elicitation of the physiology knowledge of human brain. Neural networks is composed of a series of different, associate unit. Neur-Pattern-Recognition is a very attractive since it requires minimum a priori knowledge, and with enough layers and neurons, an ANN can create any complex decision region.

- **Structural pattern recognition:** The concept of structural pattern recognition is not based on a firm theory which relies on segmentation and features extraction. Structural pattern recognition emphases on the description of the structure, namely explain how some simple sub-patterns compose one pattern. There are two main methods in structural pattern recognition, syntax analysis and structure matching. The basis of syntax analysis is the theory of formal language, the basis of structure matching is some special technique of mathematics based on sub-patterns. Structural pattern recognition always associates with statistic classification or neural networks through which we can deal with more complex problem of pattern recognition, such as recognition of multidimensional objects.

- **Syntactic pattern recognition:** This method major emphasizes on the rules of composition. And the attractive aspect of syntactic methods is its suitability for dealing with recursion. When finish customizing a series of rules which can describe the relation among the parts of the object, syntactic pattern recognition which is a special kind of structural pattern recognition can be used.

- **Approximate reasoning approach:** This method which uses two concepts: fuzzy applications and compositional rule of inference can cope with the problem for rule based pattern recognition (Kumar S.Ray ,J.Ghoshal ,1996) [19].

- **Logical combinatorial approach:** This method is presented, and works mainly in Spanish and Russian, which works with the descriptions of the objects. This approach can apply for both supervised pattern recognition and unsupervised pattern recognition.

- **Support Vector Machine (SVM):** SVM is a relative new thing with simple structure; it has been researched widely since it was proposed. The method of SVM is proposed base on the statistical theory. It is an effective tool that can solve the problems of pattern recognition and function estimation, especially can solve classification and regression problem, has been applied to a wide range for pattern recognition such as face detection, verification and recognition, object detection and recognition, speech recognition, etc.

In this thesis, we also focus on the applications of SVM for pattern recognition. This work tries to develop some advantage approaches based on SVM and improves the performances of pattern recognition systems.

1.1.2 Pattern recognition systems

A pattern recognition system based on any pattern recognition method mainly includes four mutual-associate and differentiated processes: (1) data acquisition, (2) preprocessing and enhancement, (3) dimensionality reduction and (4) recognition.

- At the first step, depending on the environment within which the objects are to be classified, data are acquired using a set of sensors or measuring devices. In addition, usually some data that have already been classified or described are assumed to be available for learning (the so-called training set).

- At the second step, some preprocessing and enhancement can be applied, such as noise reduction, artefact filtering, encoding and enhancement for extracting input pattern vectors.

- At the third step, selected features are extracted, aiming at a manageable amount of information without discarding valuable information, or to make it computationally feasible.

- At the fourth step, the recognizer is constructed, or in other words, a transformation relationship is established between features and classes. This can be, for instance, a linear or nonlinear discriminant function, nearest neighbor rule, a nearest mean (prototype) recognition rule or a Bayesian rule for computing a posteriori class probability, an artificial neural network or a kernel-based method.

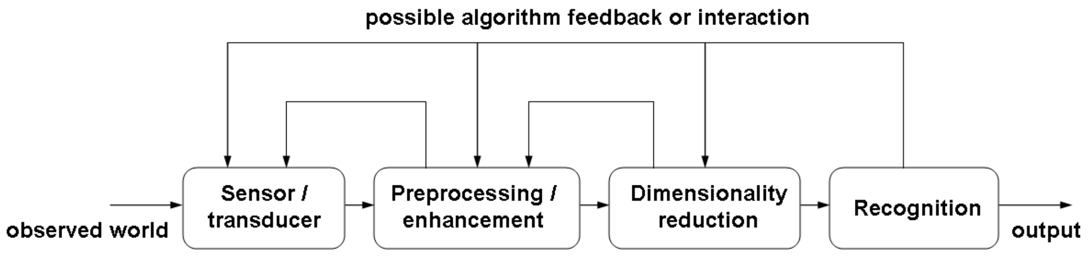


Figure 1.1: Typical pattern recognition system.

In typical pattern recognition systems, the step of recognition is the kernel of the system. It mainly includes classification and regression. Classification is a pattern recognition problem of assigning an object to a class. The output of the pattern recognition system is an integer label, such as classifying a product as “+1” or “-1” in a quality control test. Regression is a generalization of a classification task, and the output of the pattern recognition system is a real-valued number, such as predicting the share value of a firm based on past performance and stock market indicators. Figure 1.1 depicts a typical pattern recognition system [2].

In this thesis, we mainly exert ourselves to develop some improved algorithms and models based on SVM in the preprocessing and enhancement step, the dimensionality reduction step and the recognition step.

1.2 Support Vector Machine

As an intelligent machine-learning algorithm with outstanding performances for pattern recognition, Support Vector Machine (SVM) is used as the basis in our work.

SVM algorithm was first developed in 1963 by Vapnik and Lerner [20] and Vapnik and Chervonenkis [21] as an extension of the Generalized Portrait algorithm. This algorithm is firmly grounded in the framework of statistical learning theory – Vapnik Chervonenkis (VC) theory, which improves the generalization ability of learning machines to unseen data [22]. SVM stirred up attention only in 1995 with the appearance of Vapnik’s book – “The Nature of Statistical Learning Theory” [1]. In the last few years SVM have shown excellent performance in many real-world applications such as optical character recognition, text categorization, time series prediction or biological sequence analysis.

SVM is a nonlinear pattern recognition algorithm based on kernel methods. In contrast to linear methods, the kernel methods map the original parameter vectors into a higher (possibly infinite) dimensional feature space through a nonlinear kernel function. Without the need to compute the nonlinear mapping explicitly, dot-products can be computed efficiently in higher dimensional space. The dominant feature which makes SVM very attractive is that classes which are nonlinearly separable in the original space can be linearly separated in the higher dimensional feature space. For addressing the problems of nonseparable databases, the slack variables are introduced into SVM to relax the margin constraints. Thus SVM is capable to solve complex nonlinear nonseparable pattern recognition problems.

Important characteristics of SVM are its ability to solve pattern recognition problems by means of convex quadratic programming (QP), and also the sparseness resulting from this QP problem.

1.3 Challenges

Although SVM-based methods show advantages in terms of generalization performance and the recognition accuracy, the pattern recognition of real-world data using SVM still faces several challenges:

- *Huge training data*

  To make the training of classifier executable, most learning algorithms require a suitable amount of training data which scales with the number of inputs. SVM is one of the kernel methods and formulated as quadratic programming (QP) problems. Denote the number of inputs by n, then the training time complexity of QP is $O(n^3)$ and its space complexity is at least $O(n^2)$. In other words, the training time and space complexities have exponential relationships with the size of training data. Hence, a major stumbling block in SVM is the high training time and space complexities for large datasets, which is commonly encountered in real-world pattern recognition applications.

- *High dimensional input space*

  Depending on the acquisition resolution, many real-world databased consist of hundreds to thousands of measurements. While the higher dimension of this input potentially makes classifier a unique and powerful technique for a certain application, on the other hand it complicates the computation and the design of an appropriate method to handle it.

- *Noise and interaction*

  Since the acquisitions (training data and test data) are usually obtained from real-world, thus these databases are usually affected by interaction and many kinds of noises between classes. In most actual designs of classifiers, noises and interaction make the boundary between classes not clear. Many noise reduction approaches have proposed. However, it is expected that the classifier should be robust against these imperfections.

- *Imbalance in database*
  In most actual classification applications, databases are usually unbalanced. That is, the size of one class is commonly much larger than the others. This phenomenon widely exists in the real-world and it is the main reason for causing the excursion of separation boundaries in SVM classifiers. Thus, it is required to construct a classifier which can modify the separation boundaries and overcome the excursion.

1.4 Goals of the Thesis

In this thesis, advanced pattern recognition models based on Support Vector Machines (SVM) are developed and applied to real-world problems. The work presented here aims to assess the performances of the improved SVM-based models in comparison with classical methods within these actual applications.

This thesis also shows how the improved techniques handle the aforementioned challenges inherent in the classification and regression problems. Although the methods and results presented here are based on SVM models, they may give insight for other approaches.

1.5 Thesis Outlines and Main Contributions

This thesis presents our work that has been done over the last four years. It consists of seven chapters. Chapter 1 gives a background and an outline for the whole thesis. Chapter 2 introduces a fast SVM training algorithm based on the separation boundary detection technique. Chapter 3 is devoted to a feature selection method based on the correlation-based clustering and SVM sensitivity-based feature ranking. Chapter 4 improves classification model using a fuzzy decision-making SVM. Chapter 5 and Chapter 6 propose two different multiple SVM systems for classification and regression problems respectively. Finally, Chapter 7 gives a summary for the whole thesis. The flow of this thesis is depicted in Figure 1.2.

This thesis summarizes the research on SVM models, especially multi-SVM systems and their applications to real-world pattern recognition problems. Many attempts have been made to take

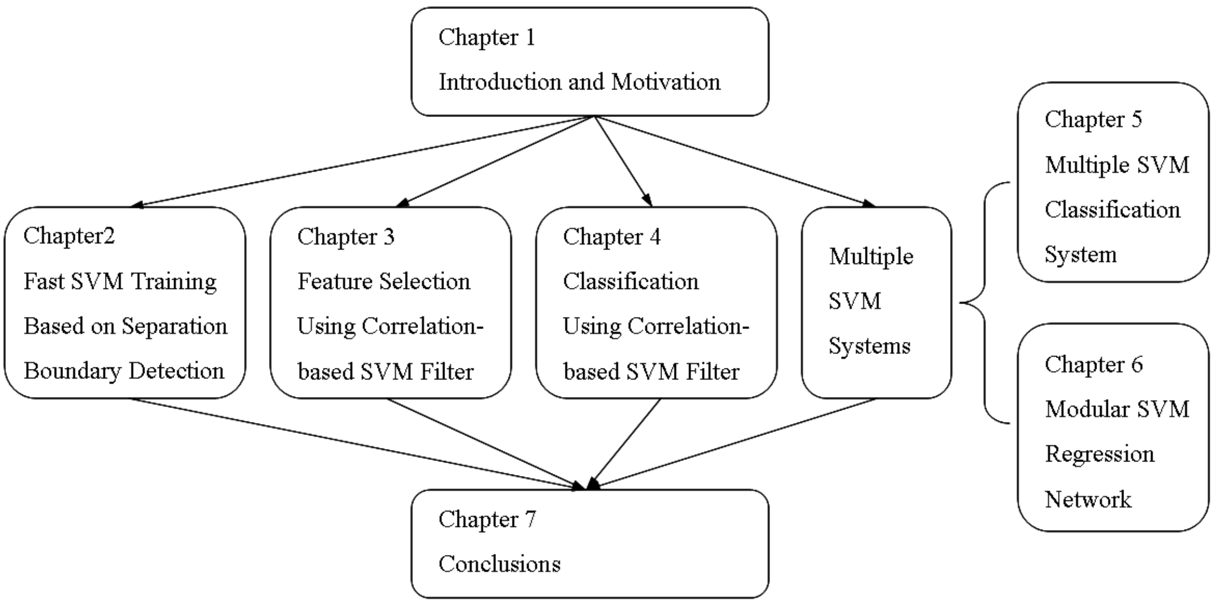


Figure 1.2: Flow diagram of this thesis.

benefit of advanced pattern recognition techniques when applied to real-world databases. At the same time these databases serve as the benchmarks of the developed techniques.

Chapter 2 introduces a fast SVM training method based on the separation boundary detection technique for solving the challenge of large training data. This approach is used in the preprocessing stage for scaling down the training data. First, the separation boundary detection technique is introduced to extract samples near the separation boundary. These samples are likely to be Support Vectors (SVs) in SVM. Then, to avoid the overfitting and preserve the structure properties, clustering is also implemented in the whole training data. The reconstructed training data consists of samples selected by the separation boundary detector and centroids of clusters. It is much smaller than the original training data, and makes the SVM training process much faster, but without degrading the classification accuracies.

The main contributions related to this fast SVM training method are that:

- The proposed fast training method is a combination of machine learning and the knowledge of image processing. In this method, the separation boundary detection is introduced into SVM to extract useful information.

- Support Vectors (SVs) are maximally reserved in the reconstructed training data, since samples near the separation boundary are likely to be SVs.
- The local properties around the separation boundary and the general properties of the entire training data are all preserved because of using separation boundary detection and clustering.

Chapter 3 reports a feature selection approach based on correlation-based feature clustering and SVM-based feature ranking for solving the challenge of high dimensional input. This method focuses on finding a good feature set consisting of features which are highly correlated with the output, yet uncorrelated with each other. First, the feature sensitivity in SVM is defined to evaluate the influence quantity of feature on the output. The features with low influence quantities on the output are eliminated. Then, a correlation-based clustering is implemented to divide the features into some clusters and selects the features that are ranked first in their clusters as the selected features. In the correlation-based clustering, correlation coefficient and information gain are used as the criterions for evaluating the feature correlation.

The proposed feature selection approach is distinctive to other feature selection algorithms in the following issues:

- The proposed model consists of supervised and unsupervised algorithms. The model is easy to be implemented. And the selected features also fit the recognizer.
- Feature sensitivity based on SVM is defined in this approach. The outstanding performances and structure of SVM can give some help on the feature ranking.
- The feature clustering method based on correlation analysis is proposed. The correlation between features are taken into account in the construction of model.
- Affinity propagation (AP) is used to implement clustering. Two criterions of correlation (correlation coefficient and information gain) are defined in AP clustering.

Chapter 4 explores a classification model using a fuzzy decision-making separation boundary for solving the challenges of noises, interaction and imbalance of data. Considering the vagueness between the real-world data sets caused by noises and interactions, a fuzzy decision-making function is first designed to take the place of the classical sign function in the prediction part of SVM. This flexible design of decision-making separation boundary is more similar to the real-world environment than conventional SVMs. Also, an offset parameter is introduced into the fuzzy decision-making function to modify the excursion of separation boundary caused by the imbalance of data. The offset is calculated as the Weighted Harmonic Mean (WHM) of the decision values of SVs.

The contributions related to this classification model are that:

- The classical hard separation boundary is replaced by a fuzzy decision-making boundary. The fuzzy decision-making separation boundary describes the gray zone between classes more exactly than the conventional method.
- The fuzzy decision-making boundary is constructed based on statistical methods. Its design is flexible and changeable to fit the distribution of training data.
- An offset parameter is proposed to calculate the excursion of separation boundary. Adding the offset into the fuzzy decision-making function, the separation boundary can be modified for solving the problem of data imbalance.
- The application of WHM algorithm can reduce the influence of noises on the calculation of the offset.

Chapter 5 proposes a multiple SVM classifier system for data classification. It has been shown by several researchers that multiple pattern recognition systems can result in effective solutions to difficult real-world tasks. The proposed multiple SVM classifier system is constructed based on piecewise partition and interpolation. First, the separation boundary detection method is used to identify the separation boundary. Then, the separation boundary is partitioned into some subsets based on the rough set theory. Based on the partition of separation boundary, training data sets are constructed to describe the subtasks for identifying segments of the separation boundary. Local SVMs are subsequently trained to solve the respective subtasks. Finally, the decisions of the local SVMs are appropriately combined based on a probabilistic interpretation to obtain the final classification decision.

The main differences between the proposed multiple SVM classifier system with other multiple classifier systems and the contributions related to this model are shown as follows:

- The main contribution of the proposed multiple SVM classifier system is its particular partition method. It tries to extract and partition the area around the separation boundary, but not the whole data space. This partition method ensures that the subtasks are the identifications of the segments of the separation boundary.
- Separation boundary detection technique is also used in this system to extract the area around the separation boundary.

- The centers of the sub training data sets are restricted to near the separation boundary, because of using rough set theory. As an extra benefit, it also makes the sub training data sets balanced.
- As same as other multiple classifier systems, local SVMs can be trained in parallel, and the training time cost can be reduced by dividing the complex task into some subtasks.

Chapter 6 proposes a multiple SVM regression system for time series prediction. Based on the algorithm design paradigm "Divide-and-conquer", the proposed modular SVR predictor system is constructed by the price domain partition. In the transformation layer, the price domain of time series is partitioned into several sub-regions to train local SVRs. The properties of these sub-regions are described by their respective SVRs. In the prediction layer, the outputs of SVRs are combined by a combinator.

The contributions related to the modular SVR network are that:

- In the proposed model, the partition is implemented in the price domain. Because the price is also the output of time series prediction and its dimensionality is one, the partition is much easier than other multiple recognizers systems.
- The influences of inputs on the prediction are different in each price level. In the proposed model, this problem is solved by constructing several SVRs for different price levels as "local experts".
- Some priori knowledge could also be added into the price domain partition such as Fibonacci support, Fibonacci resistance, and so on.

Chapter 7 concludes this work, summarizes the thesis and gives suggestions for further research.

Chapter 2

Fast SVM Training Based on Separation Boundary Detection

2.1 Introduction

As a kernel-based method, Support Vector Machine (SVM) attracted a lot of interest on various real-world applications in recent years. As introduced in the previous chapter, SVM is formulated as a quadratic programming (QP) problem. Denote the number of training samples by n, then the training time complexity of QP problem is $O(n^3)$ and its space complexity is at least $O(n^2)$. Hence, a major stumbling block in SVM training is the high training time and space complexities for large datasets, which is commonly encountered in real-world pattern recognition applications.

In order to reduce the time and space complexities, some different kinds of techniques have been developed in the preprocessing and enhancement stage of the pattern recognition systems.

- The first kind of the popular techniques is to obtain low-rank approximations on the kernel matrix, by using the Nyström method [30], greedy approximation [31], sampling [32] or matrix decompositions [33]. However, the resulting rank of the kernel matrix is still too high to be handled efficiently.

- The second kind of approaches tries to break down the large QP problem into a series of smaller QP problems, such as chunking and some decomposition methods [34, 35, 36]. Although these methods only suggest memory requirements linear in the number of training examples, they still need a long training time because the time complexity is closely related to the number of iterations.

- The third kind of approaches is to avoid the QP problems, including the core vector machine algorithm, scale-up methods and so on [37, 38, 39]. Tsang et al. also proposed variations of the Lagrangian SVM (LSVM) to obtain the solution with a fast iterative scheme [40, 41, 42]. However, for nonlinear kernels, these methods still require a large matrix.

- The fourth kind of approaches is scaling down the training data before the SVM training process. Shin et al. proposed a fast training algorithm based on quick identification of support vectors [43]. However, their algorithm appears to have some difficulties in dealing with some data sets, because of ignoring the global properties of the training data set. In addition, Pavlov et al. [44] and Collobert et al. [45] used boosting and a neural-network-based “gater” to combine several small SVMs, each of them is trained on a small data subsample. Lee and Mangasarian proposed the reduced SVM (RSVM), which uses a random rectangular subset of the kernel matrix [46].

Scaling down the training data before the training process is a direct and radical kind of approaches. The process of scaling down the training data can be easily implemented without any learning algorithms. Hence, our research also focuses on developing an effective approach to scaling down the training data.

Most of the methods mentioned here can reduce the size of training dataset, but there are still many non-relevant samples are used as the training samples. The basic problem of scaling down the training data is how the non-relevant samples in the training dataset can be identified.

First, we should define a criterion for selecting samples from the original training data set. In SVM, Support Vectors (SVs) are the outcome of the training process. The separation boundary is constructed based on SVs. Thus, to maintain the same accuracy, it is very important to keep SVs [47]. Because the process of scaling down the training data is implemented before SVM training, the criterion for selecting samples is defined as that ***the selected samples should possibly become SVs***.

Based on this criterion, a separation boundary detection method is proposed in this thesis, for scaling down the training dataset. In digital image processing, the edge detection is a technique reducing the amount of data and filtering out the useless information, while preserving the important structural properties in an image. This is also stated in the process of scaling down the training data. Therefore, a separation boundary detection technique is introduced to SVM training to preserve local properties around the separation boundary.

In addition, to avoid overfitting, the clustering technique is applied to preserve the distribution properties of the entire data. Any clustering methods can play this role, such as K-means and so on.

The reconstructed training dataset consists of samples detected by the separation boundary detection and centroids of clusters. The size of the reconstructed training dataset is much smaller than the original training dataset, thus the memory consumption of QP solver also becomes smaller than conventional SVM training.

Two main characteristics of this fast SVM training method are that,

- The proposed method focuses on the samples around the separation boundary between classes, so SVs are reserved maximally.
- Clustering method is used on the original training dataset, so the structure properties of the entire dataset are also preserved.

This chapter is organized as follows: Section 2.2 gives some issues that should be considered when developing the fast SVM training approach. In Section 2.3, we discuss why SVs are important in SVMs and why SVs should be reserved before SVM training. Section 2.4 introduces the separation boundary detection based fast SVM training method and shows the process of scaling down the training data in detail. Section 2.5 gives the proofs on validity of the separation boundary detection in the proposed approach. Simulation results and discussions are presented in Section 2.6. Concluding remarks are given in Section 2.7.

2.2 Issues to be Considered

Since SVMs are formulated as QP problems, the training time and space complexities are $O(n^3)$ and $O(n^2)$ respectively, in which n is the number of training samples. Hence, there can be advantages in reducing the size of training data before trying to select SVs, in terms of speed and space requirements.

The training data reduction should not affect the classification results. If it can improve the classification results when compared with using the whole training dataset, it would be an added benefit. On the other hand, if the accuracy is decreased, it must be in an acceptable range. In SVM, in order to maintain the same accuracy, it is important to keep the samples that could possibly become SVs. Therefore, the criterion for selecting samples is defined as that *the selected samples should have a higher likelihood of being SVs*.

Suppose that the whole dataset can be expressed as an image and each class has a certain color, then decision boundaries can be considered as the edge in an image. According to SVM theory, samples close to the separation boundaries have a higher likelihood of being SVs. Therefore, samples close to the decision boundaries can be hereby detected as points near the edge.

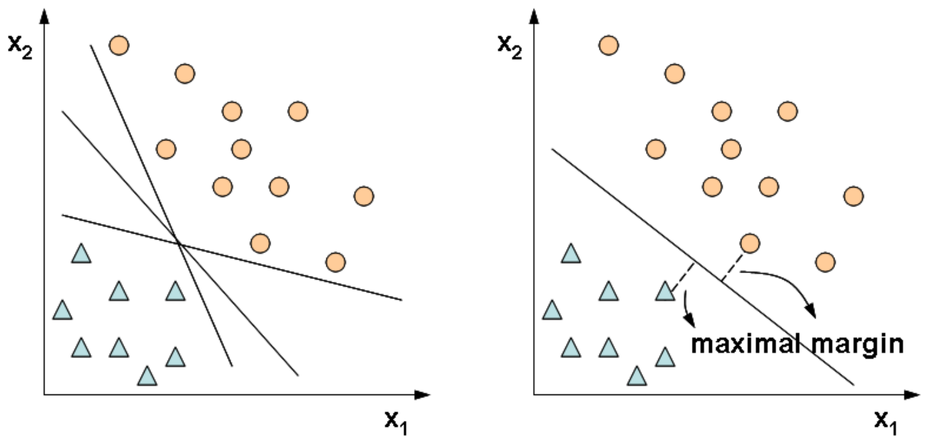


Figure 2.1: Linear classification: (left) – example of classification problem where the separating hyperplane is not unique, (right) – example of a unique hyperplane which corresponds to a maximal margin of the closest points to the separating hyperplane.

2.3 Support Vectors in SVM

Why are support vectors so important in SVM?

Consider the class of hyperplanes $w^T x + b = 0$, $w \in \mathbb{R}^n$, $b \in \mathbb{R}$, corresponding to a decision function

$$f(x) = \text{sign}(w^T x + b) \tag{2.3.1}$$

First, we consider the case of linearly separable data. A hyperplane can separate two classes of data in many possible ways (see Figure 2.1 left). There is no unique separating hyperplane, unless we add a criterion to decide which is the best or the *optimal separating hyperplane*.

Basically the idea of learning from examples is to recognize the pattern of a class by examining the training points corresponding to that class. To calculate the optimal separating hyperplane, Support Vector Machines (SVM) is proposed. Optimal separating hyperplane is defined as the maximum-margin hyperplane in the higher dimensional feature space.

Vapnik and Chervonenkis introduced the *Generalized Portrait*, a learning algorithm for separable problems, by constructing a hyperplane which maximally separates the classes (*maximum margin*):

$$\max_{w,b} \min\{\|x - x_k\| : x \in \mathbb{R}^n, w^T x + b = 0, k = 1, \dots, N\} \tag{2.3.2}$$

The use of the maximum-margin hyperplane is motivated by statistical learning theory, which provides a probabilistic test error bound which is minimized when the margin is maximized.

The Lagrangian for this problem is

$$\mathcal{L}(w,b,e;\alpha) = \frac{1}{2}w^T w - \sum_{k=1}^{N} \alpha_k \{y_k[w^T x_k + b] - 1\} \tag{2.3.3}$$

with Lagrange multipliers $\alpha_k \geq 0$ for $k = 1, \ldots, N$. The solution is characterized by the saddle point of the Lagrangian

$$\max_{\alpha} \min_{w,b} \mathcal{L}(w,b;\alpha) \tag{2.3.4}$$

The parameters of the maximum-margin hyperplane are derived by solving a quadratic programming (QP) optimization problem as the dual problem in the Lagrange multipliers α_k

$$\begin{aligned} \max_{\alpha} \mathcal{J}_D(\alpha) &= -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l x_k^T x_l \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \\ \text{such that} & \quad \sum_{k=1}^{N} \alpha_k y_k = 0 \end{aligned} \tag{2.3.5}$$

The original SVM was a linear classifier. However, in the kernel trick $K(x_k, x_l)$, each dot product used in a linear algorithm is replaced with a non-linear kernel function. In which, $\varphi(\cdot) : \mathcal{D} \to \Omega$ is a nonlinear mapping function that maps the input vector x from the input space $\mathcal{D}$ into the high dimensional feature space Ω. This causes the linear algorithm to operate in a different space. For nonseparable cases, we introduce a slack variable ξ to relax the margin constraints. Then, the optimization problem becomes

$$\begin{aligned} \min_{w,b,\xi} \mathcal{J}_P &= \frac{1}{2}w^T w + c\sum_{k=1}^{N} \xi_k \\ \text{such that} & \quad \begin{cases} y_k[w^T \varphi(x_k) + b] \geq 1 - \xi_k, k = 1, \ldots, N \\ \xi_k \geq 0, k = 1, \ldots, N \end{cases} \end{aligned} \tag{2.3.6}$$

The Lagrangian is constructed:

$$\mathcal{L}(w,b,\xi;\alpha,v) = \mathcal{J}_P(w,\xi) - \sum_{k=1}^{N} (\alpha_k y_k [w^T \varphi(x_k) + b] - 1 + \xi_k) - \sum_{k=1}^{N} v_k \xi_k \tag{2.3.7}$$

with Lagrange multipliers $\alpha_k \geq 0$, $v_k \geq 0$ for $k = 1, \ldots, N$. The solution is given by the saddle point of the Lagrangian:

$$\max_{a,v} \min_{w,b,\xi} \mathcal{L}(w,b,\xi;\alpha,v) \tag{2.3.8}$$

For SVMs, using the kernel trick makes the maximum margin hyperplane be fit in a feature space. The feature space is a non-linear map from the original input space, usually of much higher dimensionality than the original input space.

The dual problem becomes

$$\begin{aligned} \max_{\alpha} \mathcal{J}_D(\alpha) &= -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l K(x_k, x_l)\alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \\ \text{such that} & \quad \begin{cases} \sum_{k=1}^{N} \alpha_k y_k = 0 \\ 0 \le \alpha_k \le c, k = 1, \ldots, N \end{cases} \end{aligned} \tag{2.3.9}$$

In this way, the nonlinear SVM classifier can be created with the form as follows,

$$y(x) = \text{sign}[\sum_{k=1}^{N} \alpha_k y_k K(x, x_k) + b] \tag{2.3.10}$$

with α_k positive real constants which are the solution to a QP problem.

It has to be noted that as for the SVM classifier case many α_k values are equal to zero in the solution vector. In the dual space the nonlinear SVM classifier takes the form

$$y(x) = \text{sign}[\sum_{k=1}^{\#SV} \alpha_k y_k K(x, x_k) + b] \tag{2.3.11}$$

where the sum is taken over the non-zero α_k values which correspond to support vectors x_k of the training data set, $\#SV$ is the number of support vectors..

From Eq. (2.3.11), it is clear that the construction of the SVM classifier only depends on SVs, so SVs are the most important samples in SVMs. Actually, SVs usually lie near the separation boundary. Especially in nonseparable cases, they are always in the overlapping areas between different classes.

2.4 Separation Boundary Detection based Fast SVM Training

2.4.1 Separation Boundary Detection

This research proposes a separation boundary detection technique in SVM to select samples with higher likelihood of being SVs. It is like the edge detection in image processing. *Edge Detection* is a terminology in digital image processing and computer vision. Edges characterize boundaries and are therefore fundamentally important in image processing. Similarly, in classification problems, the samples close to the decision boundary are also very important. These samples have a higher likelihood of being SVs, so they are needed to be reserved in the training data reduction process.

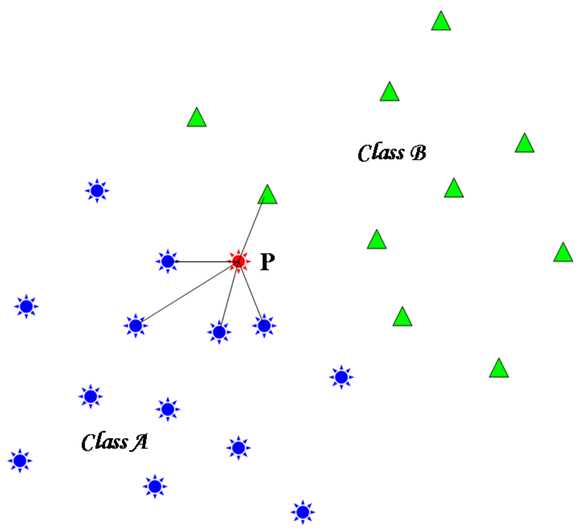


Figure 2.2: Separation boundary detection in classification problem.

It has to be noted, however, that decision boundary of SVM and SVs are all defined in feature space while this fast SVM training approach tries to detect the samples in input space. If neighborhood relation in input space is not preserved in feature space, the proposed approach may not be effective [50]. The neighborhood relation can be proven invariant under input to feature space mapping in SVMs. The proofs are given in the next section. Therefore, the samples close to the decision boundary in input space also lie around the decision boundary in feature space. Therefore, the separation boundary detection used in input space is feasible to scale down the training data set.

In image processing, edge detection technique aims at identifying points in a digital image at which the image brightness changes sharply or more formally has discontinuities. Edges are areas with strong *intensity contrasts* (a jump in intensity from one pixel to the next) [48]. Edge detection significantly reduces the amount of data and preserves the important properties.

In classification problems, our purpose of detecting sharp changes between different classes is to capture important samples around the boundary, so we only consider the change of class label.

A typical edge might for instance be the border between a block of red color and a block of yellow. Similarly, a typical classification problem might be binary. In image processing, we need to scan the neighbor pixels of a pixel to detect the sharp changes of brightness and color. In the proposed separation boundary detection model, this rule is changed to find the m nearest neighbor samples. As shown in Fig.2.2, assume that $m = 5$, for a certain training data sample P, if one of its neighbor samples has a different class label, then P is selected as an sample around the separation boundary to be reserved.

In this case, the result of applying the separation boundary detection on the training dataset may lead to a set of samples that indicate the data around the separation boundary between classes. Thus, applying a separation boundary detector on the training dataset may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties close to the separation boundary. Since SVs are the samples lie around the boundary, the samples reserved by the separation boundary detector can also keep SVs and make no bad effect on the classification result.

2.4.2 Process of Training Data Reconstruction

Sometimes, only using the samples reserved by the separation boundary detection to train SVM may lead the classifier to overfitting. In other words, the classifier may be not suitable for solving the whole data. Hence, some more information need to be added to describe the structural properties of the entire dataset. Therefore, clustering technique is used in each class, and the centroids of clusters are also reserved. As an example, K-means clustering is adopted. K-means is an algorithm which groups objects based on attributes into k number of groups, where k is a positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroids [103]. Of course, other clustering algorithms can also be used to get some better performances, such as k-center, affinity propagation, and so on.

Samples selected by the separation boundary detector and the cluster centroids selected by K-means are all used to reconstruct the training data. There are 4 steps to do this process:

Step 1 Determine the values of m = number of neighbor samples.

Step 2 Use separation boundary detector to select the samples around the separation boundary.

- Find the nearest m neighbor samples for each sample in training dataset.
- Check the class labels of neighbor samples. If one of them is different from the class label of the test sample, then the test sample is reserved, else it is removed.

Step 3 Determine the value of k = number of clusters and do K-means process to calculate the centroids of clusters.

Step 4 Reconstruct the training dataset by using the samples selected by the separation boundary detector and the centroids calculated by K-means clustering method.

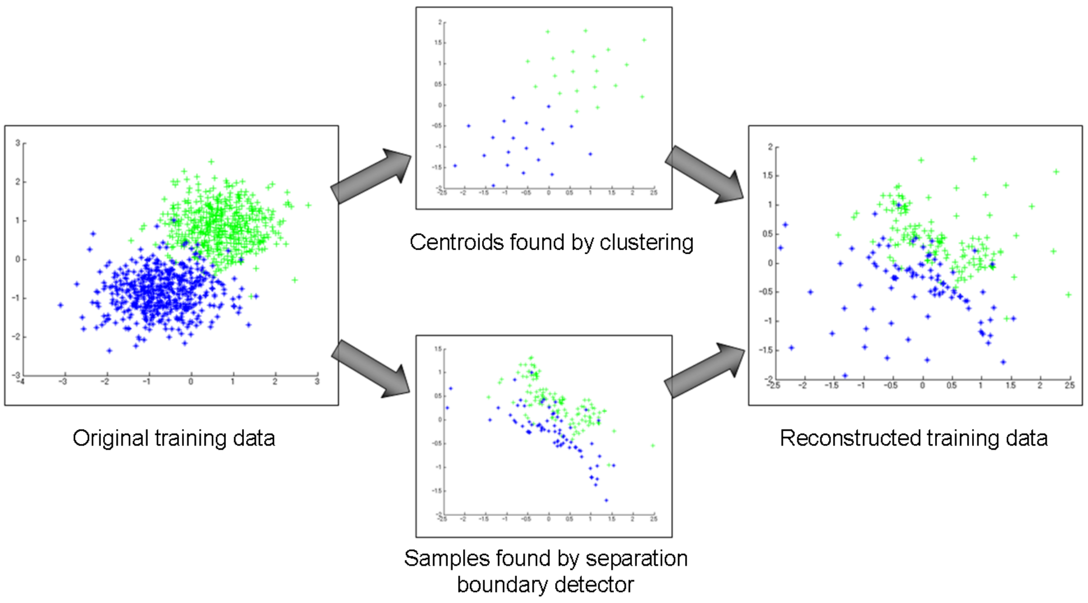


Figure 2.3: Scaling down the training data.

The process is clearly shown in Fig.2.3. Obviously, the size of the reconstructed training data is much smaller than the original training data. Further, the complexities of SVM training are also much smaller than classical algorithms.

2.5 Proofs on Validity of Separation Boundary Detection in Input Space

As described in Section 2.4, the separation boundary detection operates in input space $(\mathcal{D})$ using local information there. However, decision boundary of SVM and SVs are all defined in feature space (Ω). Since the mapping $\mathcal{D} \rightarrow \Omega$ is highly nonlinear as well as dimension expanding, we have to ensure that neighborhood relation in input space be preserved in feature space. We now provide proofs on that the k nearest neighbors of a pattern in the input space $(\mathcal{D})$ are also the k nearest neighbors of the pattern in the feature space (Ω).

Definition 2.5.1. (kNN Invariance) Let $kNN_{\mathcal{D}}(x)$ be the set of k nearest neighbors of a pattern x in the input space $(\mathcal{D})$, and $kNN_{\Omega}(x)$ be that of the pattern $\varphi(x)$ in the feature space Ω. If both sets are identical

$$kNN_{\mathcal{D}}(x) = kNN_{\Omega}(x), \forall k > 0, \forall x \tag{2.5.1}$$

the invariance of the k nearest neighbors (NN) holds.

Finding the nearest neighbors necessarily implies distance calculation. In terms of the squared Euclidean distance which is the most commonly used distance measure, the distance among patterns in the input space $\mathcal{D}$ is

$$\|x-y\|^2 = x \cdot x + y \cdot y - 2x \cdot y \tag{2.5.2}$$

The distance in the feature space Ω is similarly drawn as

$$\|\varphi(x)-\varphi(y)\|^2 = \varphi(x) \cdot \varphi(x) + \varphi(y) \cdot \varphi(y) - 2\varphi(x) \cdot \varphi(y) \tag{2.5.3}$$

where $\varphi(\cdot)$ is a mapping function from the input space to the feature space, $\varphi(\cdot) : \mathcal{D} \rightarrow \Omega$. One might obtain $\varphi(x)$ directly but the formula is extremely complicated. Thanks to the fact that the mapping $\varphi(\cdot)$ always appears within a form of inner product during SVM QP calculation, one thus uses kernel trick which substitutes the inner product to a kernel function, $\varphi(x) \cdot \varphi(y) = K(x, y)$. If this kernel trick is applied to Eq. (2.5.3), then the distance in the feature space becomes

$$\|\varphi(x)-\varphi(y)\|^2 = K(x,x) + K(y,y) - 2K(x,y) \tag{2.5.4}$$

We will consider the following typical kernel functions:

$$\text{RBF:} \quad K(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \tag{2.5.5}$$

$$\text{polynomial:} \quad K(x,y) = (x \cdot y + 1)^p \tag{2.5.6}$$

As long as the relative distance magnitude of the input space is preserved in the feature space Ω for all patterns, the composition of the k nearest neighbors of a pattern will be invariant. We now define proximity invariance.

Definition 2.5.2. (Proximity Invariance) For the patterns x, y_1 and y_2 ($x \neq y_1$,$x \neq y_2$, and $y_1 \neq y_2$) in the input space $\mathcal{D}$ satisfying

$$\|x-y_1\|^2 < \|x-y_2\|^2 \tag{2.5.7}$$

the invariance of proximity holds if they preserve their relative distances in the feature space Ω

$$\|\varphi(x)-\varphi(y_1)\|^2 < \|\varphi(x)-\varphi(y_2)\|^2 \tag{2.5.8}$$

It is obvious that kNN invariance holds if proximity invariance holds. The following two theorems provide proofs on kNN invariance for the two kernels, RBF and polynomial, by inducing proximity invariance for each of them.

Theorem 2.5.1. *(kNN Invariance for RBF Kernel) kNN invariance holds when the mapping function $\varphi(x)$ is defined such that*

$$\varphi(x) \cdot \varphi(y) = K(x, y) = \exp(-\frac{\|x - y\|^2}{2\sigma^2}) \tag{2.5.9}$$

Proof. Let y_1 and y_2 be two distinct neighbors of x with $\|x - y_1\|^2 < \|x - y_2\|^2$, i.e., y_1 is closer to x than y_2. Suppose the invariance of proximity does not hold for mapping function $\varphi(x)$, i.e., $\|\varphi(x) - \varphi(y_1)\|^2 \geq \|\varphi(x) - \varphi(y_2)\|^2$. Using Eq. (2.5.4), one can rewrite the inequality as

$$K(x, x) + K(y_1, y_1) - 2K(x, y_1) \geq K(x, x) + K(y_2, y_2) - 2K(x, y_2) \tag{2.5.10}$$

Since kernel is a inner product of mapping function, so $K(\vec{a}, \vec{a}) = 1$ and $K(\vec{a}, \vec{b}) > 0, \forall \vec{a}, \vec{b}$, the inequality is simplified as

$$K(x, y_1) \leq K(x, y_2) \tag{2.5.11}$$

Plugging the definition of RBF kernel, we obtain

$$\exp(-\frac{\|x - y_1\|^2}{2\sigma^2}) \leq \exp(-\frac{\|x - y_2\|^2}{2\sigma^2}) \tag{2.5.12}$$

which in turn can be simplified into

$$\|x - y_1\|^2 \geq \|x - y_2\|^2 \tag{2.5.13}$$

This is contradictory to our initial assumption that y_1 is closer to x than y_2. Thus the assumption that the invariance of proximity does not hold is not true. Therefore, *kNN invariance holds for RBF kernel.* □

Theorem 2.5.2. *((kNN Invariance for Polynomial Kernel) kNN invariance holds when the mapping function $\varphi(x)$ is defined such that*

$$\varphi(x) \cdot \varphi(y) = K(x, y) = (x \cdot y + 1)^p \tag{2.5.14}$$

and training patterns are all norm vectors ($\| \cdot \|$=1).

Proof. Let y_1 and y_2 be two distinct neighbors of x with $\|x - y_1\|^2 < \|x - y_2\|^2$, i.e., y_1 is closer to x than y_2. Suppose the invariance of proximity does not hold for mapping function $\varphi(x)$, i.e., $\|\varphi(x) - \varphi(y_1)\|^2 \geq \|\varphi(x) - \varphi(y_2)\|^2$. Using Eq. (2.5.4), one can rewrite the inequality as

$$K(x, x) + K(y_1, y_1) - 2K(x, y_1) \geq K(x, x) + K(y_2, y_2) - 2K(x, y_2) \tag{2.5.15}$$

Since kernel is a inner product of mapping function, so $K(\vec{a},\vec{a}) = 2^p$ from $\vec{a}\cdot\vec{a} = 1$, the inequality is simplified as

$$K(x, y_1) \leq K(x, y_2) \tag{2.5.16}$$

Plugging the definition of polynomial kernel, we obtain

$$(x \cdot y_1 + 1)^p \leq (x \cdot y_2 + 1)^p \tag{2.5.17}$$

The polynomial degree p can be eliminated from both sides since norm vectors always satisfy $-1 < \vec{a}\cdot\vec{b} < 1, \forall \vec{a} \neq \vec{b}$, and $(\vec{a}\cdot\vec{b} + 1) > 0$. Therefore, we get

$$x \cdot y_1 \leq x \cdot y_2 \tag{2.5.18}$$

The inner product between the patterns can be represented

$$\|x\|\|y_1\|\cos\theta_1 \leq \|x\|\|y_2\|\cos\theta_2 \tag{2.5.19}$$

where θ_1 and θ_2 denote the angles between x and y_1, and between x and y_2, respectively. Since $\|x\| = \|y_1\| = \|y_2\| = 1$, finally one has

$$\cos\theta_1 \leq \cos\theta_2 \tag{2.5.20}$$

and hence $\theta_1 \leq \theta_2$. In other words,

$$\|x - y_1\|^2 \geq \|x - y_2\|^2 \tag{2.5.21}$$

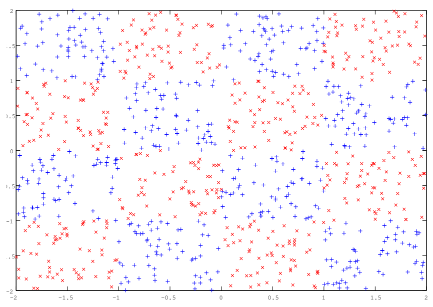
This is contradictory to our initial assumption that y_1 is closer to x than y_2. Thus the assumption that the invariance of proximity does not hold is not true. Therefore, *kNN invariance holds for polynomial kernel.* □

According to the proofs, we can find that the neighborhood relation is invariant under input to feature space mapping. Therefore, the separation boundary detection is practicable in the input space.

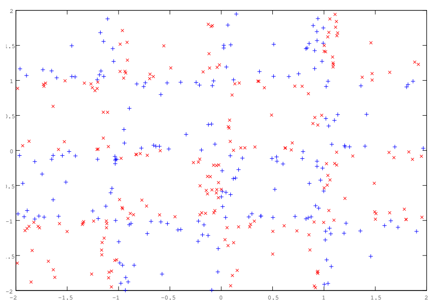
2.6 Simulations

2.6.1 Classification of Checkerboard Data

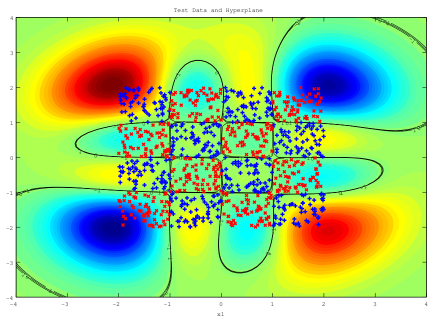
Firstly, we experiment on the 4×4 checkerboard data, which is commonly used for evaluating large-scale SVM implementations. The original training data and testing data are created randomly.



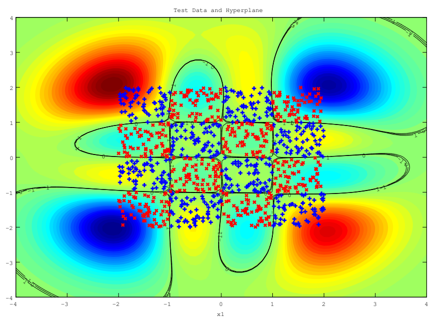
(a)



(b)



(c)

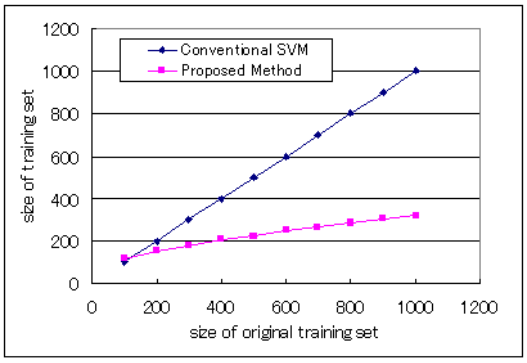


(d)

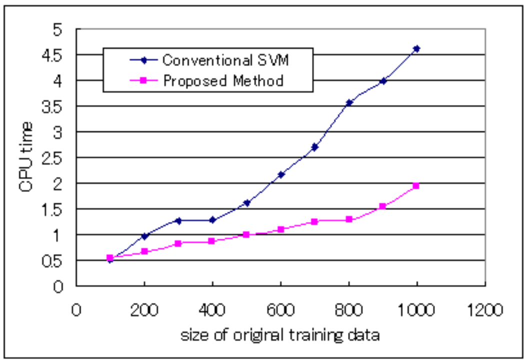
Figure 2.4: Checkerboard test dataset : (a) – the original training data, (b) – the reconstructed training data, (c) – separation boundaries built by the original data, (d) – separation boundaries built by the reconstructed data.

We use training sets with a maximum of 1000 samples and 1,000 samples for testing. This problem does not need so many points, but it is convenient for testing the scaling properties. As the default setting of the toolbox, we set the parameter $C = 10$ in SVM. Since we focus on the nonlinear kernels training algorithms, the RBF kernel is adopted. In the simulations shown in this thesis, the proposed approach and conventional SVM are all adapted from the SVM-KMToolbox (in MATLAB) [49].

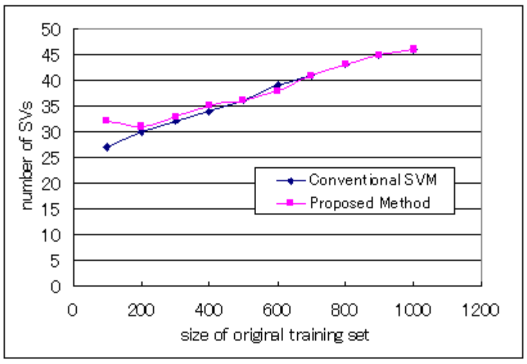
Experimentally, conventional SVM is also implemented to compare with our proposed approach. Training data and their classification results are shown in Fig.2.4 graphically. Figure 2.4(a) is the original training data. Figure 2.4(b) is the reconstructed training data by using the proposed approach. Figure 2.4(c) shows the test data and separation boundaries built by using the original



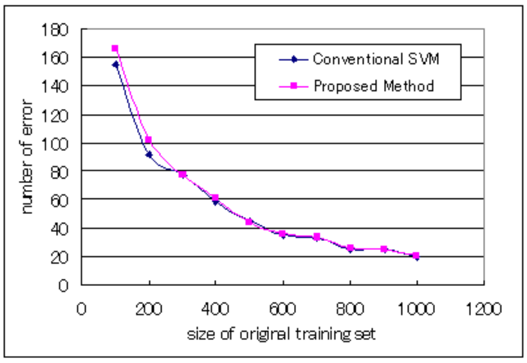
(a)



(b)



(c)

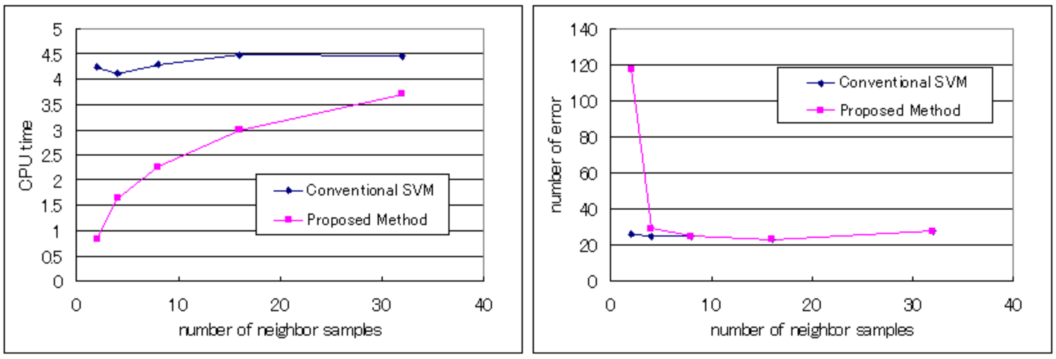


(d)

Figure 2.5: Comparison between conventional SVM and proposed method : (a) – size of training set, (b) – CPU time cost, (c) – number of support vectors, (d) – number of error.

data in Fig.2.4(a). Figure 2.4(d) shows the test data and separation boundaries built by using the reconstructed data in Fig.2.4(b). Obviously, the size of training data reconstructed by our proposed approach is much smaller than the size of original training data. From Fig.2.4(c) and Fig.2.4(d), we can also find that the proposed approach basically maintains the same separation boundaries.

Results of different size of original training data are shown in Fig.2.5. As can be seen, our proposed fast SVM training method based on separation boundary detection technique is as accurate as the conventional SVM. In addition, the number of SVs in the proposed approach also approximatively equals the number of SVs in conventional SVM. In other words, SVs are mostly kept. In Fig.2.5(b), the CPU time cost of the proposed approach consists of the time cost of the training data



(a) **(b)**

Figure 2.6: Performances corresponding to the number of neighbor samples (30 clusters): (a) – CPU time cost, (b) – number of error.

reduction and the time cost of SVM learning. We can also find that the proposed approach is much faster and produces far smaller size of training data on large datasets.

In particular, the performances corresponding to the number of neighbor samples and the performances corresponding to the number of clusters are shown in Fig.2.6 and Fig.2.7. As shown in Fig.2.6(b), if the number of neighbor samples is too small, the accuracy of SVM will be decreased. Therefore, for this dataset, the suitable number of neighbor samples is determined as 8.

2.6.2 Classification of Gaussian Random Binary Data

Gaussian random binary dataset is generated randomly. We also use training sets with a maximum of 1000 samples and 1,000 samples for testing. Figure 2.8 shows the intuitional results of conventional SVM and our proposed method. Figure 2.8(a) and Figure 2.8(b) are the training data used in conventional SVM and the proposed method. Figure 2.8(c) shows the test data, the separation boundary and bonding planes built by conventional SVM. Figure 2.8(d) shows the test data, the separation boundary and bonding planes built by the proposed method. In this case, the size of training data reconstructed by our proposed approach is also much smaller. The proposed approach also maintains the separation boundaries. Numerically, the proposed method as well as the conventional SVM almost has the same number of SVs and the same accuracy. However, the CPU time cost of the proposed approach is much smaller than the conventional SVM. We can see it clearly in Fig.2.9. From the performances corresponding to the number of neighbor samples and the number

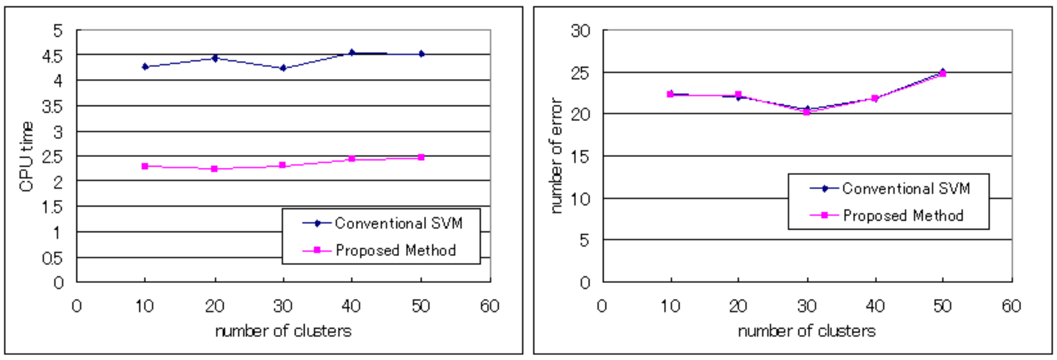


Figure 2.7: Performances corresponding to the number of clusters (number of neighbor samples = 8): (a) – CPU time cost, (b) – number of error

of clusters as shown in Fig.2.10 and Fig.2.11, we can get a similar result with the fist simulation.

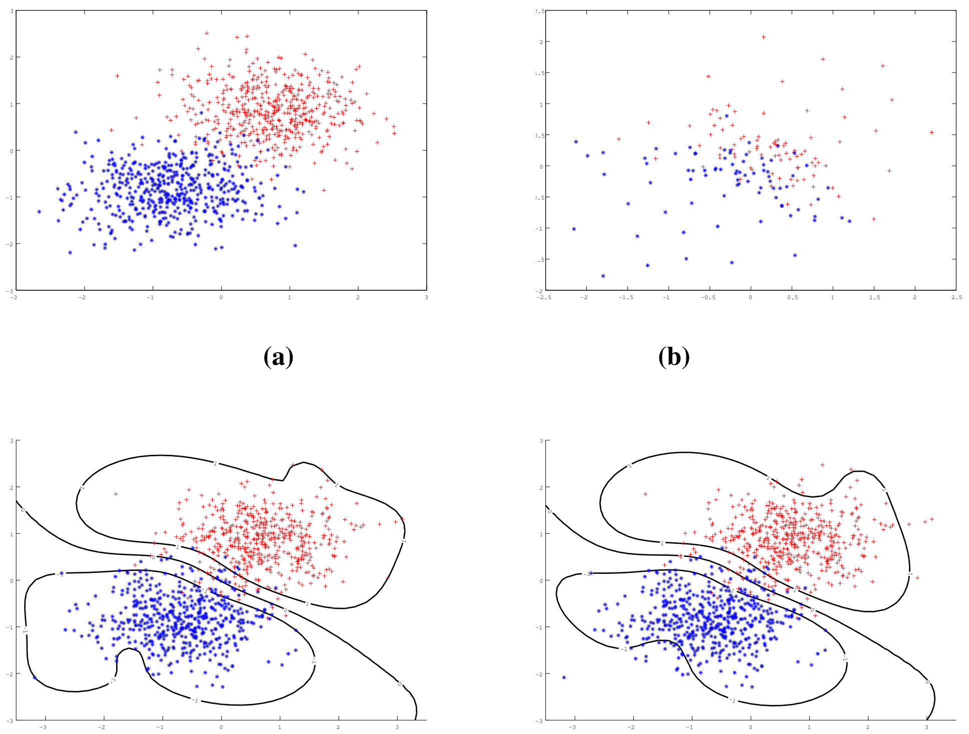
2.6.3 Classification of Real-world Data

We also experiment on some real datasets from UCI machine learning repository [104]. Iris dataset, monks-problems dataset and forest cover type dataset are used to test our proposed approach.

The Iris dataset is a collection of 150 Iris flowers of 3 kinds, with 4 attributes. Three classes are the iris-setosa, iris-versicolor and iris-virginica. 75 samples are used as the training set and the remaining 75 are used as the test set. For comparison, we also run Reduced Training Set (RTS) fast SVM, which is an effective fast SVM training method proposed by Ravindra Koggalage and Saman Halgamuge.

Table 2.1 is the results summary for IRIS-one class vs others. The first row of Table 2.1 shows values of performance metrics obtained for the conventional SVM. The following rows show the results of RTS fast SVM and our proposed fast SVM based on separation boundary detection. The first column is the accuracy of classification. The second column shows the percentage number of samples reserved in training. In other words, it is the size of training data. The last column shows the percentage time taken for the whole training process. In our proposed method, it consists of the time taken for training data reduction and the time taken for SVM training.

As shown in Table 2.1, using the proposed BD-FSVM, we can obtain the same accuracies with SVM and RTS fast SVM in the cases of IRIS-Setosa vs others and IRIS-Versicolor vs others. The



(a) **(b)**

(c) **(d)**

Figure 2.8: Gaussian random binary test dataset : (a) – the original training data, (b) – the reconstructed training data, (c) – separation boundaries built by the original data, (d) – separation boundaries built by the reconstructed data.

classification accuracy of the case of IRIS-Virginica vs others obtained by BD-FSVM is 96.0%, with an increase of 9.3% compared to SVM and RTS fast SVM. In the aspects of scaling down the training data and reducing the time cost, the proposed BD-FSVM obtains best values in all cases. The sizes of training data are scaled down to 4.0%, 22.1% and 16.0% of the sizes of original training data, respectively. And the time costs are reduced to 3.1%, 22.1% and 17.3% of the time costs in conventional SVM. Compared to RTS fast SVM, the performances are also significantly improved by BD-FSVM.

The three problem sets defined for Monk's data were also used in the experiment. The problem 1 is in standard disjunctive normal form and is supposed to be easily learnable by most of the

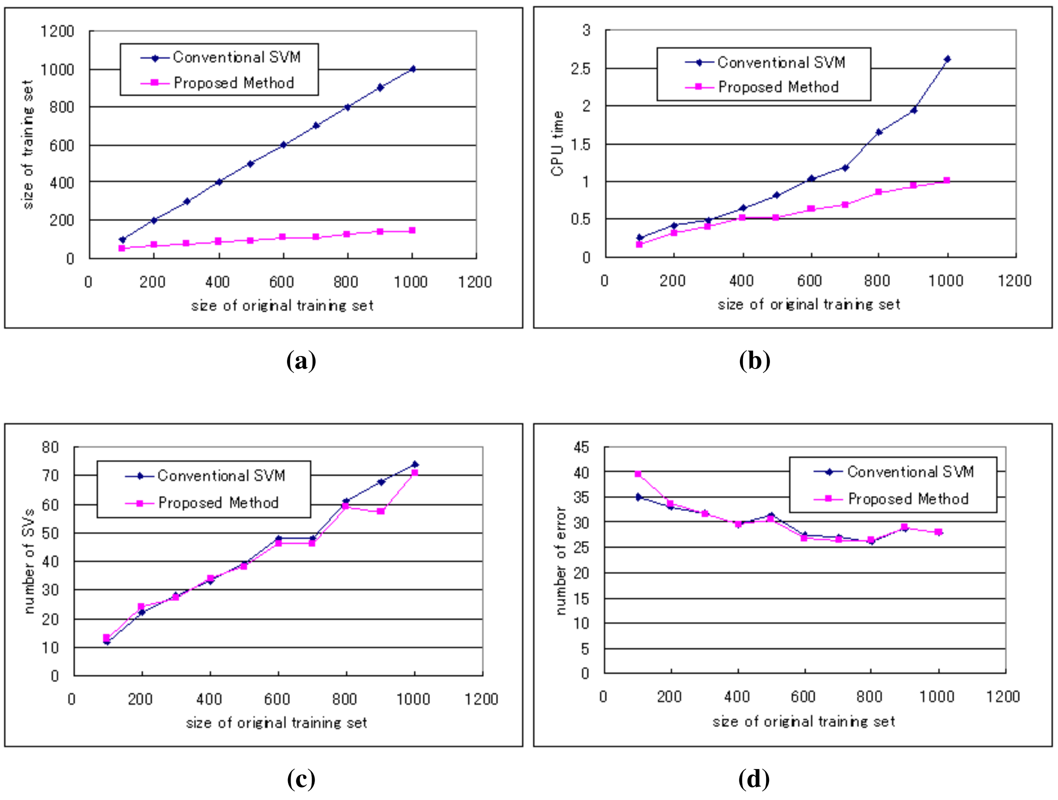
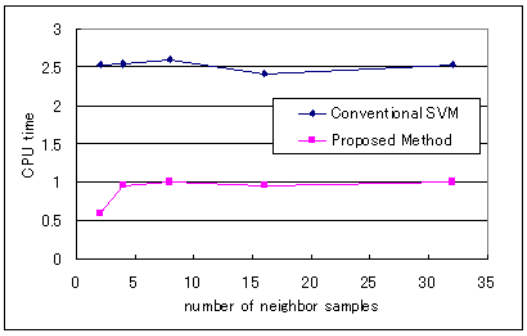


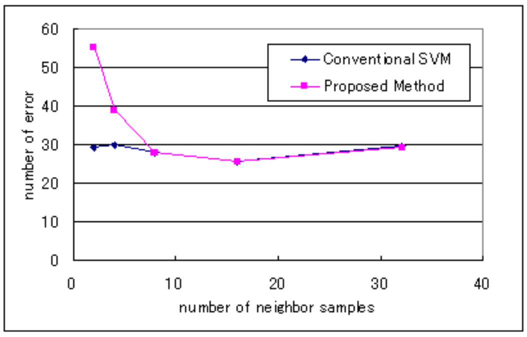
Figure 2.9: Comparison between conventional SVM and proposed method: (a) – size of training set, (b) – CPU time cost, (c) – number of support vectors, (d) – number of error.

algorithms and decision trees. Conversely, Monk's problem 2 is similar to parity problems. It combines different attributes in a way that makes it complicated to describe using the given attributes only. Thus, it is harder to solve than other two problems. Monk's problem 3 has noise added. Results are shown in Table 2.2.

On these problems, we obtain similar results with the Iris dataset. The sizes of training data are scaled down to 74.2%, 51.5% and 54.9% of the sizes of original training data, and the time costs are reduced to 44.2%, 44.7% and 44.5% of the time costs in conventional SVM. Although the accuracies of classification are decreased from 85.7% and 95.4% to 85.4% and 94.7%, the classification ability of BD-FSVM is still in an acceptable range and better than RTS fast SVM in most problems.

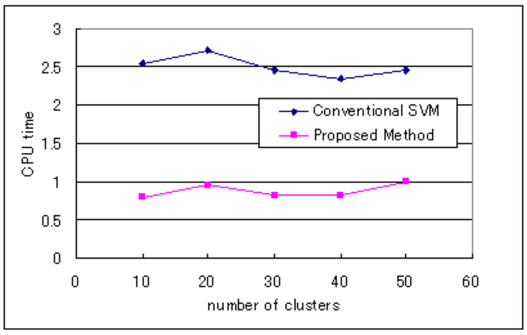
We also test a large actual dataset – forest cover type dataset, which is determined from the US Forest Service (USFS) and Resource Information System (RIS) data. Initial 10,000 samples are

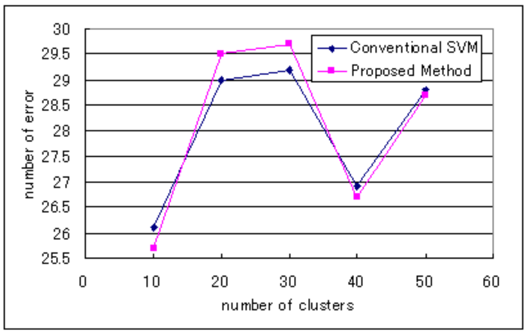




(a) **(b)**

Figure 2.10: Performances corresponding to the number of neighbor samples (40 clusters): (a) – CPU time cost, (b) – number of error.





(a) **(b)**

Figure 2.11: Performances corresponding to the number of clusters (number of neighbor samples = 8): (a) – CPU time cost, (b) – number of error.

Table 2.1: Results summary for IRIS-one class vs others.

Classifiers	IRIS-Setosa vs Others			IRIS-Versicolor vs Others			IRIS-Virginica vs Others		
	Correctly classified (%)	Percentage number of samples reserved (%)	Percentage time taken for training (%)	Correctly classified (%)	Percentage number of samples reserved (%)	Percentage time taken for training (%)	Correctly classified (%)	Percentage number of samples reserved (%)	Percentage time taken for training (%)
SVM	**100**	100	100	**97.3**	100	100	86.7	100	100
RTS SVM	**100**	13.3	5.0	**97.3**	53.3	27.9	86.7	42.7	19.0
BD-FSVM	**100**	**4.0**	**3.1**	**97.3**	**21.3**	**22.1**	**96.0**	**16.0**	**17.3**

Legends: Better results are in bold-type.

selected and divided into two datasets of 5,000 each, randomly. One set is the training set and the other is the test set. Cover types 1,2 and 5 are considered for the classification. Results obtained for the forest cover type data can be found in Table 2.3.

The classification accuracies obtained by BD-FSVM are as same as the accuracies of SVM and RTS fast SVM. Similar with IRIS data and Monks problems, the proposed approach also obtains the best performances in terms of accuracy and time cost on the forest cover type dataset.

2.6.4 Discussions

In our simulations, the time cost of the proposed approach consists of the computational cost of the training data reduction and the computational leaning cost of SVM. As shown in Fig.2.5(b), Fig.2.9(b), Table2.1, Table2.2 and Table2.3, our proposed approach is faster than other methods for all datasets. In the simulations of the forest cover type dataset, because the size of dataset and the number of attributes are very large, the time cost of training data reduction becomes larger than the time cost of SVM training. However, the total time cost of the proposed approach is still smaller than the conventional SVM and RTS fast SVM method.

Based on simulations, we found that 8 is a suitable number of neighbor samples for most cases. But this is not enough, so we will introduce some methods to find the optimal parameters in our future research.

Table 2.2: Results summary for Monks problems.

Classifiers	Monks Problem 1			Monks Problem 2			Monks Problem 3		
	Correctly classified (%)	Percentage number of samples reserved (%)	Percentage time taken for training (%)	Correctly classified (%)	Percentage number of samples reserved (%)	Percentage time taken for training (%)	Correctly classified (%)	Percentage number of samples reserved (%)	Percentage time taken for training (%)
SVM	**85.7**	100	100	69.4	100	100	**95.4**	100	100
RTS SVM	84.9	81.9	57.8	69.7	89.9	51.8	95.3	77.3	61.6
BD-FSVM	85.4	**74.2**	**44.2**	**70.8**	**51.5**	**44.7**	94.7	**54.9**	**44.5**

Legends: Better results are in bold-type.

Table 2.3: Results summary for forest cover type vs the rest.

Classifiers	Forest cover type 1 vs the rest			Forest cover type 2 vs the rest			Forest cover type 5 vs the rest		
	Correctly classified (%)	Percentage number of samples reserved (%)	Percentage time taken for training (%)	Correctly classified (%)	Percentage number of samples reserved (%)	Percentage time taken for training (%)	Correctly classified (%)	Percentage number of samples reserved (%)	Percentage time taken for training (%)
SVM	**86.4**	100	100	**89.7**	100	100	**87.2**	100	100
RTS SVM	**86.4**	79.2	77.6	**89.7**	94.3	93.2	**87.2**	73.1	40.9
BD-FSVM	**86.4**	**25.0**	**61.2**	**89.7**	**50.1**	**80.2**	**87.2**	**2.80**	**37.9**

Legends: Better results are in bold-type.

2.7 Conclusions

This chapter introduces an improved SVM training method to overcome the problems with large training datasets. When the training set is too large, it is impossible to use the entire original data for training SVM due to the high space cost and computation cost.

In SVM theory, SVs are samples close to the decision boundary. Here we introduced a separation boundary detection technique from image processing to SVM training process. Samples detected to be reserved by the separation boundary detection are more likely to be SVs, because they are the samples around the separation boundary (i.e. the samples in the overlapping areas are reserved). In our proposed approach, the separation boundary detection is implemented in the input space. Its validity is proven by proving the invariance of neighborhood relation under input space to feature space mapping in SVMs.

In order to maintain the structure property of the entire data and avoid overfitting, we also used a clustering method to compute the centroids of clusters. These centroids are also reserved to reconstruct the training data.

In the simulation part, we compared the proposed approach with conventional SVM and RTS fast SVM. As shown in the results summary tables, best performances in terms of success rate, the size of training data and the time taken for SVM calculation are all obtained by our proposed fast SVM training based on the separation boundary detection for both numerical data and real-world data.

Chapter 3

Feature Selection Using Correlation-based SVM Filter

3.1 Introduction

In machine learning, it is important to build a robust learning model for high dimensional data. One of the main tasks is dimensionality reduction, which can be divided into *Feature Selection* and *Feature Extraction* [52].

Feature selection tries to find a subset of the original variables. Feature extraction tries to map the multidimensional space into a space of fewer dimensions. Feature selection has been proven faster and more suitable for the data with some redundant features [53]. Especially in some real-world pattern recognition applications, feature selection is effective in reducing dimensionality, removing irrelevant data, increasing learning accuracy, and improving result comprehensibility.

Feature selection algorithms typically fall into two categories: *Feature Ranking* and *Subset Selection*. Feature Ranking ranks the features by a metric and eliminates all features that do not achieve an adequate threshold [54]. Subset Selection searches the set of possible features for the optimal subset [55]. Feature ranking is useful in the expatiation of relation between input features and the output. Subset selection reflects the performance of the combination of features in a certain extent.

Subset selection algorithms can be mainly divided into *Wrappers*, and *Filters* [56]. Wrappers apply an unsupervised learning algorithm to each subset of features and then evaluate the subset of features by criterion functions, such as evolutionary algorithms [57]. The disadvantages of wrappers are the high computationally cost and the risk of overfitting, since they must repeatedly call the induction algorithm along with a statistical re-sampling technique and must be rerun when a different induction algorithm is used [58].

Filters are similar to Wrappers in the search approach, but instead of evaluating against a model, a simpler filter is evaluated. In recent years, data has become increasingly larger in number of features in many applications such as genome projects, text categorization, image retrieval, and customer relationship management [59]. As discussed in many literatures, filters are ultimately more feasible than Wrappers in these cases [60]. Therefore, in this work, a filter model is adopted to work on the feature selection problems.

Some existing feature ranking algorithms, evolutionary algorithms, and some other evaluation measures have been shown effective in feature selection on training and testing data, for example, sequential forward search (SFS), plus and take away (PTA), gradient-based feature selection (GFS), genetic algorithms (GAs), Chaotic BPSO with K-NN (CBPSO-KNN), and PSO-SVM [61, 62, 63, 64, 65, 58, 66]. However, most of them only score subsets of features according to their predictive power but ignore the correlation analysis of features. As a result of ignoring the correlation analysis, these methods can not preserve the diversity of features, thus they cannot generalize well for actual problems.

A hypothesis is given as a rule to select good features in some literatures: ***a good feature subset is one that contains features highly correlated to the class, yet uncorrelated to each other*** [51]. In M. A. Hall et al. and L. Yu et al.'s papers, correlation measures are also applied to evaluate the goodness of feature subsets [51, 60, 56]. Although the correlation analysis is applied in their approaches, the classification performances are not good enough. That is because their approaches are not constructed based on any classifiers, the selected features' predictive power is not optimal and cannot be measured.

To overcome the problems of these algorithms and meet the demand for feature selection for high dimensional data, this work develops a hybrid approach which contains a supervised learning process and is designed based on the correlation analysis. The supervised learning is also implemented as a feature ranking and eliminates features with low influence quantities on the output. The correlation analysis is implemented as a correlation-based clustering to divide the features into some clusters. From each clusters, the feature who is ranked first in its cluster is selected.

Two main characteristics of the proposed method are that,

- The predictive power of selected features is evaluated by feature ranking.
- The diversity of selected features is preserved by correlation analysis.

In some existing feature selection methods, Neural Networks (NNs) based feature ranking has been used and proven effective [67, 68]. However, it does not stand mass noise and easily falls into a local optimum. In order to obtain better performances, Support Vector Machine (SVM) is chosen to develop an improved feature ranking approach. As an algorithm with outstanding performances, SVMs are not negatively affected by high dimensionality and can overcome the overfitting. In the proposed SVM feature ranking, a feature sensitivity is defined and calculated based on the Lagrange multipliers and SVs calculated from the training process. Features are ranked by their sensitivities. A threshold is set to eliminate features with very low sensitivities. In this approach, the rank of features reflects the influence quantities of features on the output.

After the feature ranking, a clustering method is used to divide and select features. Many existing methods can be used to do the feature clustering. Here, we select a new method called affinity propagation [69], which takes as input measures of similarity between pairs of data points. Real-valued messages are exchanged between data points until a high-quality set of exemplars and corresponding clusters gradually emerges. Affinity propagation can find clusters with much lower error than other methods, and it is also faster. In this clustering method, we defined two kinds of correlation measures:*linear correlation coefficient* and *information gain*. Linear correlation coefficient is suitable for most problems, but it is not able to capture correlations that are not linear in nature. Therefore, the information gain is introduced as another correlation measure for some real-world data. From each cluster, the feature who is ranked first is selected as a significant feature.

The main improvements made in this approach are listed as follow:

- The sensitivity in SVM is defined as the criterion of feature ranking.
- The objects of the clustering are the features but not the input vectors in common cases.
- Information gain is introduced in correlation analysis as well as linear correlation coefficient to demonstrate the proposed approach is universally valid.

As a combination of the SVM-based feature ranking and the correlation-based clustering, the proposed approach can preserve the diversity of selected features and improve the adaptability for classifiers. In comprehensive experiments, better performances are obtained by the proposed approach on several actual problems.

The remainder of this chapter is organized as follows: Section 3.2 describes the structure of the proposed feature selection approach. Section 3.3 provides the idea and the introduction of the SVM-based feature ranking. In Section 3.4, the algorithm of correlation-based clustering is introduced in detail. In Section 3.5, the effectiveness of the proposed approach is evaluated via experiments on various real-world data sets, and discuss the implications of the findings. Section 3.6 concludes this work with some possible extensions.

3.2 Structure of the Approach

As mentioned in the previous section, correlation analysis is important in feature selection. If we adopt the correlation between two features as a criterion, the purpose of feature selection becomes selecting a feature subset, in which each feature is highly correlated to the output but uncorrelated to any other features. In other words, if a feature has a high enough influence quantity on the output to make it predictive and is uncorrelated to any other relevant selected features, it will be regarded as a suitable feature to be selected for the classification task. In this sense, the problem of feature selection boils down to *finding a suitable measure of correlations between features* and *a selection procedure to select features from the relevant features*.

In order to solve this problem, we proposed a two-stage modular model. The proposed model consists of two main modules: *SVM-based feature ranking module* and *correlation-based clustering module*.

SVM-based feature ranking is proposed to rank the features based on their sensitivities on the output. The features with very low sensitivities are eliminated. The rank of features is also used after the correlation-based clustering to select features from clusters.

Correlation-based clustering is used to split the feature space into some fragments. In other words, it is used to divide the features of data into some clusters based on the correlation of features. In each cluster, a feature has high correlation with the other features, but low correlation with the features in the other clusters. So the features from the same cluster have similar characteristics. Therefore, we select a feature to delegate its cluster based on the rank of features. The flowchart of the proposed model is shown in Fig. 3.1.

3.3 SVM-Based Feature Ranking

In order to eliminate the features with low influence quantities on the output and give a criterion of feature selection after the clustering, we proposed a feature ranking approach based on SVM. That is because SVM has good performances in most real-world problems.

In previous chapters, the idea of finding a maximal margin in SVM is explained. The margin maximization process equals to finding the optimal separation boundary [70]. Because SVM implements the structural risk minimization principle, it can overcome the overfitting and obtain a global

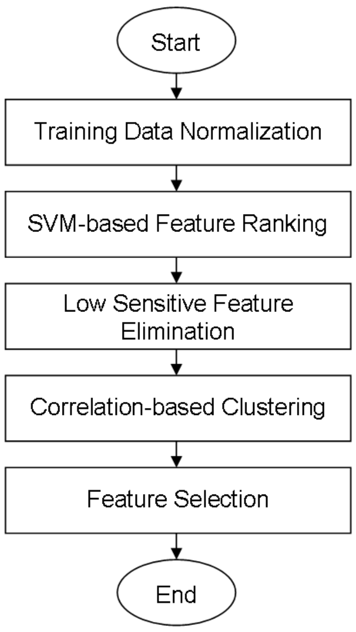

Start
Training Data Normalization
SVM-based Feature Ranking
Low Sensitive Feature Elimination
Correlation-based Clustering
Feature Selection
End

Figure 3.1: Flowchart of correlation-based SVM feature selection filter.

optimal solution.

3.3.1 Sensitivity in SVM

Assume that we have a binary data set denoted as $\{x_i, y_i\}$, where $x_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, N$. x_i is the i-th input vector and y_i is its class label (+1 or -1). The input data set is divided into two different classes A and B which have labels +1 and -1 respectively.

As introduced in Section 2.3, for nonlinear problems, by introducing a vector of Lagrange multipliers α and nonlinear mapping function $\varphi(\cdot)$, the problem (2.3.2) is rebuilt as a QP problem (2.3.9) in its dual space. In which, $K(x_k, x_l) = \varphi(x_k)^T\varphi(x_l)$ is the kernel function. After training, we obtain the vector of Lagrange multipliers α. If $\alpha_k \neq 0$, then sample-i is a support vector.

The nonlinear SVM classifier can be created with the form as follows,

$$y(x) = \text{sign}[\sum_{k=1}^{\#SV} \alpha_k y_k K(x, x_k) + b] \tag{3.3.1}$$

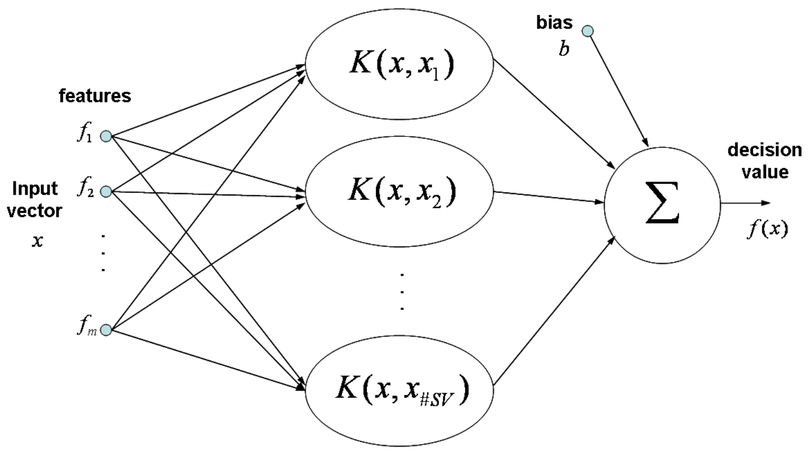


Figure 3.2: Structure of SVM.

where $\#SV$ is the number of support vectors. In which, we define the decision value $f(x)$ as:

$$f(x) = \sum_{k=1}^{\#SV} \alpha_k y_k K(x, x_k) + b \tag{3.3.2}$$

where x_k are SVs and x is the input vector. Only SVs have nonzero Lagrange multipliers, so using them to determine the separation boundary is sufficient. The structure of SVM can be drawn as a network shown in Fig. 3.2.

In order to rank features, we introduced the definition of sensitivity into SVM. In general, the sensitivity of function y to variable x is defined as follows [71],

$$\eta(x|y) = |\frac{\partial y}{\partial x}| \tag{3.3.3}$$

In SVM, we can consider that $f(x)$ is a mapping function of x. Then the sensitivity can be used as a measure of the feature's ability to influence the decision value $f(x)$. x is the set of features, so the sensitivity of SVM to x is defined by the formula,

$$\eta(x|f(x)) = |\frac{\partial f(x)}{\partial x}| = |\nabla_x f(x)| \tag{3.3.4}$$

According to the definition of decision value (3.3.2), the sensitivity is rewritten as follows,

$$\eta(x|f(x)) = |\sum_{k=1}^{\#SV} \alpha_k y_k \nabla_x K(x, x_k)| \tag{3.3.5}$$

Table 3.1: Kernels and their gradients.

Kernel	$K(x, x_k)$	$\nabla_x K(x, x_k)$
Linear	$x \cdot x_k$	x_k
Poly	$(1 + x \cdot x_k)^c$	$c(1 + x \cdot x_k)^{c-1} x_k$
RBF	$\exp(\frac{-\|x_k - x\|^2}{2\sigma^2})$	$\frac{(x_k - x)}{\sigma^2} \exp(\frac{-\|x_k - x\|^2}{2\sigma^2})$
Sigmoid	$\tanh(b(x \cdot x_k) - c)$	$\frac{b}{\cosh^2(x \cdot x_k - c)} x_k$

3.3.2 Sensitivity Based Feature Ranking

In the feature ranking process, the sensitivity of each feature is used as a criterion to rank and select features.

In SVMs, many kernels can be selected to calculate the feature sensitivity. As shown in Table 3.1, we can select linear kernel, ploy kernel, RBF kernel or sigmoid kernel to calculate the feature sensitivity. Their gradients are also shown in Table 3.1. Therefore, we can easily calculate the the feature sensitivity of each feature with these kernels, respectively.

As a practicable example, here we select a linear function as the kernel of SVM to explain the feature ranking process. The linear kernel is defined as the equation (3.3.6),

$$K(x, x_k) = x \cdot x_k \tag{3.3.6}$$

Assume that $x = [f_1, f_2, \ldots, f_m]$ and $x_k = [d_{k1}, d_{k2}, \ldots, d_{km}]$, the kernel function is rewritten as follows:

$$K(x, x_k) = x \cdot x_k = \sum_{j=1}^{m} f_j d_{kj} \tag{3.3.7}$$

Using the linear kernel, the formula of decision value is rewritten as

$$\begin{aligned} f(x) &= \sum_{k=1}^{\#SV} \sum_{j=1}^{m} \alpha_k y_k f_j d_{kj} + b \\ &= \sum_{j=1}^{m} (\sum_{k=1}^{\#SV} \alpha_k y_k d_{kj}) f_j + b \end{aligned} \tag{3.3.8}$$

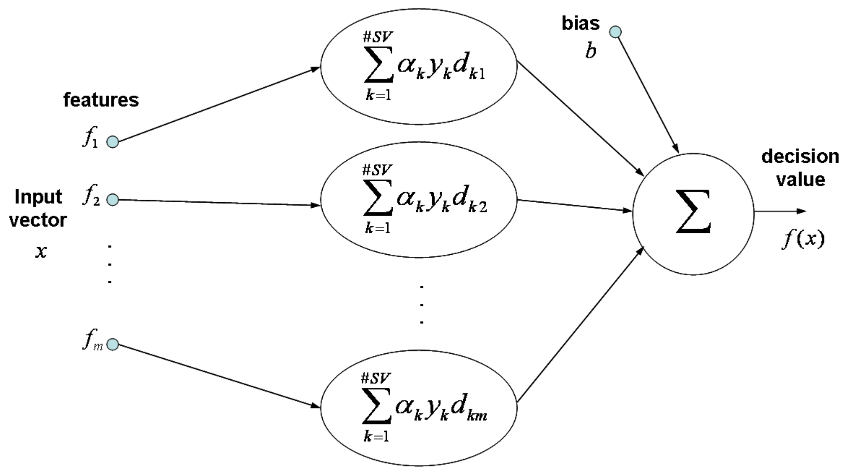

features
f_1
Input vector x
f_2
f_m
$\sum_{k=1}^{\#SV}\alpha_k y_k d_{k1}$
$\sum_{k=1}^{\#SV}\alpha_k y_k d_{k2}$
$\sum_{k=1}^{\#SV}\alpha_k y_k d_{km}$
bias b
Σ
decision value
$f(x)$

Figure 3.3: Structure of SVM with linear kernel.

Based on formulas (3.3.5) and (3.3.8), we can calculate the feature sensitivity of each feature:

$$\eta_j = |\sum_{k=1}^{\#SV} \alpha_k y_k d_{kj}|, \quad j = 1, 2, \ldots, m \tag{3.3.9}$$

Using linear kernel SVM, the feature sensitivity of each feature is calculated by the Lagrange multipliers α_k, the SVs x_k from the training data and their corresponding labels y_k, where $k = 1, \ldots, \#SV$. The feature sensitivities are used to rank features. The feature who has a higher feature sensitivity value is arranged to a higher position in sequence.

On the other hand, we can redraw the structure of SVM as Fig. 3.3 based on the formula (3.3.8). The figure shows that the decision value can be written as a linear combination of the features. Each feature has a weight. The weight of a feature is just its feature sensitivity. So we can also consider that the criterion in linear kernel SVM is the absolute value of each feature's weight.

As same as the linear kernel case, we can also calculate feature sensitivities based on other kernels. Because it is very hard to select appropriate types of kernel functions for a given problem in SVMs [72], we use both linear kernel and RBF kernel to calculate feature sensitivities in the experiments.

In order to make feature ranking feasible, we normalize the features before SVM training. Using the proposed approach, we rank the features based on their sensitivities. As same as some other feature ranking methods, we also define a threshold ε to eliminate the features with low sensitivities. In simulations, we normalize the sensitivities values to the range $[0, 1]$. Because we only want to eliminate the features who have very low predictive power, ε could be given any very small positive value. In the proposed approach, the classification results are not highly sensitive to the threshold ε. As an example, we set $\varepsilon = 0.03$ in our simulations.

3.4 Correlation-based Clustering

3.4.1 Affinity Propagation

In order to measure the correlations between reserved features after feature ranking, we select a new method called affinity propagation, which is proposed by B. J. Frey and D. Dueck [69]. Here, all features are simultaneously considered as potential exemplars. By viewing each feature as a node in a network, this method recursively transmits real-valued messages along edges of the network until a good set of exemplars and corresponding clusters emerges. Messages are updated on the basis of simple formulas that search for minima of an appropriately chosen energy function.

Affinity propagation takes as input a collection of real-valued similarities between data points, where the similarity $s(i,k)$ indicates how well the data point with index k is suited to be the exemplar for data point i.

There are two kinds of message exchanged between data points, and each takes into account a different kind of competition. Messages can be combined at any stage to decide which points are exemplars and, for every other point, which exemplar it belongs to.

- The "*responsibility*" $r(i,k)$, sent from data point i to candidate exemplar point k, reflects the accumulated evidence for how well-suited point k is to serve as the exemplar for point i, taking into account other potential exemplars for point i.
- The "*availability*" $a(i,k)$, sent from candidate exemplar point k to point i, reflects the accumulated evidence for how appropriate it would be for point i to choose point k as its exemplar, taking into account the support from other points that point k should be an exemplar.

$r(i,k)$ and $a(i,k)$ can be viewed as log-probability ratios. To begin with, the availabilities are initialized as $a(i,k) = 0$. Then, the responsibilities are computed using the rule

$$r(i,k) \leftarrow s(i,k) - \max_{k' \ s.t.\ k' \neq k} \{a(i,k') + s(i,k')\} \tag{3.4.1}$$

Whereas the above responsibility update lets all candidate exemplars compete for ownership of a data point, the following availability update gathers evidence from data points as to whether each candidate exemplar would make a good exemplar,

$$a(i,k) \leftarrow \min\{0, r(k,k) + \sum_{i' \ s.t. \ i' \notin \{i,k\}} \max\{0, r(i',k)\}\} \tag{3.4.2}$$

The "self-availability" $a(k,k)$ is updated differently:

$$a(k,k) \leftarrow \sum_{i' \ s.t. \ i' \neq k} \max\{0, r(i',k)\} \tag{3.4.3}$$

This message reflects accumulated evidence that point k is an exemplar, based on the positive responsibilities sent to candidate exemplar k from other points.

The above update rules require only simple, local computations that are easily implemented 3.4.2, and messages need only be exchanged between pairs of points with known similarities. At any point during affinity propagation, availabilities and responsibilities can be combined to identify exemplars.

Each iteration of affinity propagation consisted of (i) updating all responsibilities given the availabilities, (ii) updating all availabilities given the responsibilities, and (iii) combining availabilities and responsibilities to monitor the exemplar decisions and terminate the algorithm when these decisions did not change anymore.

In this method, we should define the similarity function $s(i,k)$ as the criterion to measure the correlation between each two features.

3.4.2 Criterions of Correlation Analysis

There exist broadly two approaches to define the similarity function $s(i,k)$ to measure the correlation between two features. One is based on classical *linear correlation* and the other is based on *information theory*.

linear correlation coefficient

Under the first approach, the most well known measure is linear correlation coefficient [73]. For a pair of variables (X,Y), the correlation coefficient r is given by the formula (3.4.4),

$$r = \mathcal{C}(X,Y) = \frac{\sum_i (x_i - \bar{x_i})(y_i - \bar{y_i})}{\sqrt{\sum_i (x_i - \bar{x_i})^2}\sqrt{\sum_i (y_i - \bar{y_i})^2}} \tag{3.4.4}$$

where $\bar{x_i}$ is the mean of x_i, and $\bar{y_i}$ is the mean of y_i. The correlation coefficient is a symmetrical measure for two variables. The value of r lies between -1 and 1, inclusive. If X and Y are completely correlated, r takes the value of 1 or -1; if X and Y are totally independent, r is zero.

Information Gain

Although linear correlation coefficient is applicable in most cases, it is not suitable for some real world data. That is because it is disable to capture correlations that are not linear in nature. In order to overcome this problem, in our approach we also adopt a criterion called information gain. In the information theory, the expected value of the information gain is a mutual information for a pair of variables (X, Y). It denotes the reduction in the entropy of X achieved by learning the state of the random variable Y. In other words, the information gain reflects the correlation between X and Y. The calculation of information gain is based on the information-theoretical concept of entropy. The entropy of a variable X is defined as

$$H(X) = -\sum_i P(x_i)\log_2(P(x_i)) \tag{3.4.5}$$

and the entropy of X after observing values of Y is defined as

$$H(X|Y) = -\sum_i \sum_j P(y_j)P(x_i|y_j)\log_2(P(x_i|y_j)) \tag{3.4.6}$$

where $P(y_j)$ is the prior probabilities of Y, and $P(x_i|y_j)$ is the posterior probabilities of X given the values of Y. Information gain is given as following

$$IG(X|Y) = H(X) - H(X|Y) \tag{3.4.7}$$

From this equation, it is clear that information gain is the amount by which the entropy of X decreases reflects additional information about X provided by Y. Thus, a feature Y is regarded more correlated to feature X than to feature Z, if $IG(X|Y) > IG(Z|Y)$.

3.4.3 Parameter Estimation in Feature Clustering

With using linear correlation coefficient and information gain as the correlation measurement function, affinity propagation is implemented to find feature clusters in the training data. In affinity propagation, a preferences parameter p can be changed to increase or decrease the number of identified clusters. As same as other feature selection approaches, the identification of the number of clusters is still a problem.

A common choice of parameter p is the median of similarity values $p' = median(s)$, s is the vector of similarity values. In our simulations, we tuned the preferences parameter p by a parameter selection methodology. The training data is fist divided into 5 partitions. Then the procedure of parameter p optimization follows R$\ddot{a}$tsch's methodology [74], which trains the algorithm with each candidate parameter by a 5-fold-cross validation procedure on the five training partitions of the training data and selects the model parameters to be the median over those five estimates. In our simulations, the candidate parameters are set to $\{\frac{p'}{3}, \frac{p'}{2}, p', 2p', 3p'\}$. The computational cost of this way to estimate the parameter is a little high, so we will commit ourself to solving this problem in our future work. However this method will make our comparison more robust and the results more reliable.

In each cluster produced by the correlation-based clustering, a feature will be selected, if it is ranked first. As a result of the proposed approach, the selected features are not highly correlated with each other, but have the highest influence quantity on the output in their clusters.

3.5 Simulations

In our simulations, we evaluate the proposed approach in terms of the number of selected features, classification accuracy, and the most important feature subset on several real-world data.

In this section, the proposed correlation-based SVM filter which uses the linear correlation coefficient as the correlation measure is abbreviated to *CSF-r*. And the proposed approach which uses the information gain as the correlation measure is abbreviated to *CSF-IG*. By using linear kernel and RBF kernel in SVM-based feature ranking, we construct 4 feature selection models: CSF-r (Lin), CSF-IG (Lin), CSF-r (RBF) and CSF-IG (RBF).

In order to demonstrate the effectiveness of the proposed approach, we employ some commonly used approaches (sequential forward search (SFS), plus and take away (PTA)), GAs (sequential Genetic Algorithm (SGA), hybrid Genetic Algorithm (HGA)), some novel evolutionary algorithms (Chaotic BPSO with K-NN (CBPSO-KNN), PSO-SVM), and a feature ranking method (gradient-based feature selection (GFS)) from the literatures in the comparison experiments.

3.5.1 Classification of Ionosphere Data

We first test our proposed approach on the Ionosphere dataset from the UCI database [104]. This radar data was collected by a system in Goose Bay, Labrador. The data format was arranged as shown in Table 3.2. The size of training data is 150 and the size of testing data is 201.

This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. "Good" radar

Table 3.2: Format of Ionosphere dataset.

Dataset	Instances	Classes	Features
Ionosphere	351 (201/150)	2	34

Legends: x/y: Number of test samples / Number of train samples.

Table 3.3: Clustering results (CSF-r).

Clusters	Features
I	1
II	3, 5
III	4
IV	8, 10, 12, 14, 16
V	6, 7, 9, 11, 13, 15, 17, 19, 21, 23
VI	18, 20, 22, 26, 30
VII	24
VIII	25, 27, 29, 31, 33
IX	28, 32, 34

returns are those showing evidence of some type of structure in the ionosphere. “Bad” returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this data base are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.

Using the proposed approach, 34 features are first ranked by the SVM based feature ranking (in simulations, we use two common kernels: linear kernel and RBF kernel). Some features with low sensitivities are eliminated. The reserved features are divided into some clusters by correlation-based clustering (in our simulations, we use two kinds of correlation measures: linear correlation coefficient and information gain). The number of clusters is identified by a preferences parameter p of affinity propagation. The value p is determined by R$\ddot{a}$tsch’s parameter selection methodology. And then, based on the rank, we select the features from the clusters.

The clustering results of CSF-r and CSF-IG are shown in Table 3.3 and Table 3.4, respectively. Features selected by the proposed 4 models (CSF-r (Lin), CSF-IG (Lin), CSF-r (RBF) and CSF-IG (RBF)) are shown in Table 3.5. The numbers of selected features of CSF-r and CSF-IG are 9 and 11. In some papers, the 2nd feature is commonly omitted in the experiments, because it is zero for all observations and thus does not contribute to the variance. In our simulations, the sensitivity of the 2nd feature is the lowest. In feature ranking, because its sensitivity is smaller than the threshold,

Table 3.4: Clustering results (CSF-IG).

Clusters	Features
I	1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 29, 31, 33, 34
II	11, 17
III	13, 15, 21
IV	16
V	18, 25, 27, 30
VI	20
VII	19, 22, 23
VIII	24
IX	26
X	28
XI	32

Table 3.5: Selected features.

Approaches	d*	Selected features
CSF-r (Lin)	9	1, **3**, 4, 7, 8, **22**, **24**, 27, 34
CSF-IG (Lin)	11	**3**, 11, 16, 20, 21, **22**, **24**, 26, 27, 28, 32
CSF-r (RBF)	9	1, **3**, 4, 7, 8, **22**, **24**, 29, 34
CSF-IG (RBF)	11	**3**, 11, 13, 16, 20, **22**, **24**, 26, 27, 28, 32

Legends: Features selected by four models (CSF-r (Lin), CSF-IG (Lin), CSF-r (RBF), CSF-IG (RBF)) are in bold-type.

it is eliminated.

We can find that the feature subset $\{3, 22, 24\}$ is selected by all 4 feature selection models. So in our simulations, we consider this subset as the most important feature subset. That means the features in this subset contains the most important information of dataset. Of course, we can not only use this subset to solve classification problems, some other selected significant features will also be used. Using the selected feature subset, an SVM classifier is trained to classify the test data. Table 3.6 compares the best experimental results obtained by other approaches with the proposed models. It is clearly that the proposed approach can obtain a higher accuracy than other approaches on the Ionosphere dataset.

3.5.2 Classification of Other Real-world Data

We also used some other datasets from the UCI Repository in our simulations. These datasets are all collected from the real-world. A summary of datasets is presented in Table 3.7. We use 10-fold

Table 3.6: Accuracy of classification for Ionosphere dataset.

Approaches	d*	accuracy(%)
SFS	7	93.5
PTA	7	93.5
SGA	7	95.4
HGA	7	95.7
CBPSO-KNN	15	97.3
PSO-SVM	15	97.3
GFS	12	97.3
CSF-r (Lin)	9	**98.7**
CSF-IG (Lin)	11	98.0
CSF-r (RBF)	9	97.3
CSF-IG (RBF)	11	97.3

Legends: The best result is in bold-type.

Table 3.7: Summary of classification problems.

Datasets	Instances	Classes	Features
Vowel	990	11	10
Wine	178	3	13
WDBC	569	2	30
Sonar	208	2	60

cross-validation method to evaluate the proposed approach. Table 3.8 compares the best experimental results obtained by other approaches with the proposed approach. The features selected by our proposed approach for each dataset are shown in Table 3.9.

3.5.3 Discussions

From Table 3.6 and Table 3.8, we can find that the proposed approach obtained the highest classification accuracy for the Ionosphere, Vowel, Wine and WDBC classification problems.

The classification accuracies of the Ionosphere dataset obtained by CSF-r (Lin) and CSF-IG (Lin) are 98.7% and 98.00%, respectively, an increase of 1.4% and 0.7% classification accuracy compared to the other existing approaches. The results of CSF-r (RBF) and CSF-IG (RBF) are 97.3% which equals to the accuracy obtained by CBPSO-KNN, PSO-SVM and GFS, but is better than other approaches. In Vowel classification problem, only CSF-IG (RBF) obtains the best result. Although the classification accuracies obtained by CSF-r (Lin), CSF-IG (Lin) and CSF-r (RBF) are worse than the accuracies of some other approaches, it is still comparable. For the Wine and WDBC classification problems, the proposed models all obtained good performances. In addition,

Table 3.8: Accuracy of classification for test datasets

	Vowel		Wine		WDBC		Sonar	
	d*	accuracy(%)	d*	accuracy(%)	d*	accuracy(%)	d*	accuracy(%)
SFS	8	**99.7**	8	95.5	18	94.0	48	91.8
PTA	8	**99.7**	8	95.5	18	94.2	48	92.3
SGA	8	**99.7**	5	95.5	12	94.4	24	95.7
HGA	8	**99.7**	5	95.5	6	94.9	24	**97.1**
CBPSO-KNN	6	97.9	8	99.4	8	97.5	27	93.3
PSO-SVM	7	99.5	8	**100**	13	95.6	34	96.2
GFS	5	99.5	3	**100**	7	**98.3**	39	93.3
CSF-r (Lin)	5	99.5	**2**	**100**	4	**98.3**	**9**	96.2
CSF-IG (Lin)	**4**	99.5	**2**	**100**	**2**	**98.3**	28	95.7
CSF-r (RBF)	6	99.5	**2**	95.5	4	**98.3**	11	96.2
CSF-IG (RBF)	**4**	**99.7**	**2**	**100**	**2**	**98.3**	43	92.3

Legends: Better results are in bold-type.

the numbers of selected features by using the proposed approach are much smaller than the other approaches. This means that only a few features are needed in these classification problems. For the Sonar classification problem, though the classification accuracy obtained by the proposed approach is worse than the accuracies of some other approaches, it is also still comparable.

From Table 3.9, we can also select the most important feature subsets of these problems. The selection of the most important feature subset may be useful for some other data analysis problems. The most important feature subsets of Vowel, Wine, and WDBC are $\{1, 2, 9\}$, $\{13\}$, and $\{24\}$.

The results indicate that for different classification problems, the proposed approach can serve as a preprocessing step and optimize the feature selection process. The features selected by the proposed approach can lead to an increase in classification accuracy effectively.

3.6 Conclusions

In this chapter, we propose a feature selection approach based on the correlation analysis and the theory of SVM.

The affinity propagation clustering approach is used to group the features with high correlations into clusters. In the clustering process, two correlation measures (linear correlation coefficient and information gain) are defined as the criterion of the correlation measure. For different data sets, the two correlation measures can give us different performances.

Based on the structure of SVM and the definition of sensitivity, we proposed a feature ranking

Table 3.9: Selected features for test datasets

datasets	Approaches	d*	Selected features
Vowel	CSF-r (Lin)	5	**1**, **2**, 3, 6, **9**
	CSF-IG (Lin)	4	**1**, **2**, 7, **9**
	CSF-r (RBF)	6	**1**, **2**, 5, 6, **9**, 10
	CSF-IG (RBF)	4	**1**, **2**, 5, **9**
Wine	CSF-r (Lin)	2	7, **13**
	CSF-IG (Lin)	2	4, **13**
	CSF-r (RBF)	2	7, **13**
	CSF-IG (RBF)	2	5, **13**
WDBC	CSF-r (Lin)	4	2, 17, **24**, 27
	CSF-IG (Lin)	2	4, **24**
	CSF-r (RBF)	4	17, 22, **24**, 27
	CSF-IG (RBF)	2	4, **24**
Sonar	CSF-r (Lin)	9	12, 18, 22, 27, 30, 31, 35, 44, 47
	CSF-IG (Lin)	28	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 28, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51
	CSF-r (RBF)	11	4, 8, 11, 14, 21, 23, 28, 34, 36, 43, 45
	CSF-IG (RBF)	43	1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 29, 30, 31, 32, 33, 34, 35, 36, 37, 39, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50

Legends: Features selected by four models (CSF-r (Lin), CSF-IG (Lin), CSF-r (RBF), CSF-IG (RBF)) are in bold-type.

approach to rank features, eliminate useless measurements and select the significant ones from the clusters built by correlation-based clustering.

The proposed approach is a general model for many problems, because it considers both correlation of features and their influence quantities on the output. The proposed approach is tested on some real-world benchmark problems. Simulation results show that our method simplifies features more effectively and obtains a higher classification accuracy compared to the other feature selection methods.

Chapter 4

Classification Using Fuzzy Decision-making SVM

4.1 Introduction

In most actual designs of pattern recognition, the classification problems are mostly non-separable. In order to deal with the non-separable cases, SVM algorithm uses a unique variable parameter denoted by c, which can weigh the tolerance of SVM in its training part. In other words, by the changing of c, the width of margin between neighbor classes is also changed.

Because the real-world databases are usually affected by interaction and many kinds of noises between different classes. So the classes are commonly overlapping and the separation boundaries are not clear and hard to be constructed. When using the variable parameter c to control the process of separation boundary construction, the situation of too wide margin may cause too many points to be included in the margin, and contrarily, the number of points included is too small. Because the width of margin controls the chosen of SVs, the changing of variable c can also effect the performance of SVM evidently. So choosing an appropriate value of c becomes very important, but this task is not easy. On the other side, in the decision-making part of SVM, it is used to apply the sign function to predict labels of test data, so the misclassification for the points around the boundary is unavoidable.

In order to reduce the influence of noise and interaction and make the pattern recognition model more robust for different choice of parameter c and different real-world problems, a fuzzy boundary is proposed in this chapter to take the place of the sign function in the decision-making part of SVM.

To clarify the availability of SVM, a series of real-world data sets have been used in simulations. The proposed method does not treat the binary-labeled test data sets as strict detached sets with a stiff boundary, but see them as fuzzy sets in which each test data vector with private decision making value has a proper believable value to predict its own label. Being different from the margin in training part of SVM, the proposed method can reduce the influence on the accuracy of classifier from changing c, and the misclassification caused by the hard decision-making boundary also can be modified effectively. This flexible method will let the classifier to be more regulable and stable for the change of parameters and the margin of SVM.

Another problem in actual classification applications is imbalance of data. That is, the size of one class is commonly much larger than the others. This phenomenon widely exists in the real-world and it is the main reason for causing the excursion of separation boundaries in SVM classifiers. Traditionally, for solving this problem, some methods are used to improve the quality of training. Genetic algorithm(GA), grid search techniques and derivative-free numerical optimizer are used for determining more suitable learning parameters. But because these models still use the conventional training framework, the performances are not significantly improved. S.X. Du and S.T. Chen (2005) introduced a weighted method called weighted SVM (WSVM) to solve the unbalance problem by distributing different weights to different classes [75]. Weights in the WSVM are determined by the proportion of different classes. However, the method may not be effective in the cases where data are not uniformly distributed, because for the nonuniform distribution data the proportion of different classes cannot describe correctly the unbalance of support vectors (SVs). Since the separation boundary in the SVM is determined by SVs that are the nearest samples around the boundary, it is natural to consider developing a weighted SVM based directly on the estimated unbalance of SVs.

In this chapter, we also introduce an offset parameter to the fuzzy function for modifying the boundary excursion. The offset parameter is estimated based on the decision values of SVs. Although a simple arithmetic mean of SVs' decision values may be applied to calculate the offset, it obviously is not accurate when the SVs are contaminated by noises. However, some SVs are inevitably influenced by noises in a real world. In nonseparable support vector classification (SVC), some samples influenced heavily by noises are also treated as SVs. As a result, it is hard to obtain an optimal offset by using the simple arithmetic mean. To reduce the effect of noise, we introduce a so called Weighted Harmonic Mean (WHM) algorithm to calculate the offset. Given that SVs influenced by noise are usually far away from the separation boundary, we use a Blackman function to give these SVs smaller weights and reduce their contribution to the offset. Using the WHM offset, we obtain a modified boundary.

The main improvements introduced in this chapter are that,

- A fuzzy decision-making separation boundary takes place of the classical hard sign separation boundary in SVM.
- An offset parameter is introduced into the fuzzy decision-making function to modify the separation boundary for solving the problem of data imbalance.
- The fuzzy decision-making function is constructed based on the statistics of results of the supervised learning.
- The offset parameter is calculated as the weighted harmonic mean of the decision values of SVs.

Simulation studies on different kinds of real-world unbalanced data sets are carried out to confirm the effectiveness of the proposed approach.

This chapter is organized as follows: Section 4.2 gives the definition of decision value in SVMs, which is the basis of calculation of the proposed approach. Section 4.3 propose a fuzzy decision-making function to replace the classical separation boundary. Section 4.4, an offset parameter is introduced to solve the problem of data imbalance, which is calculated as the weighted harmonic mean of the decision values of SVs. Numerical simulations are carried out in Section 4.5. Finally, Section 4.6 is devoted to discussions and conclusions.

4.2 Decision Value in SVM

In nonseparable problems, SVMs are basically constructed by introducing positive slack variable ξ_k in the constraints. In In the primal weight space the optimization problem becomes

$$\begin{aligned} \min_{w,b,\xi} \mathcal{J}_{\mathcal{P}} &= \frac{1}{2} w^T w + c \sum_{k=1}^{N} \xi_k \\ \text{such that} & \quad \begin{cases} y_k[w^T x_k + b] \geq 1 - \xi_k, k = 1, \ldots, N \\ \xi_k \geq 0, k = 1, \ldots, N \end{cases} \end{aligned} \tag{4.2.1}$$

In Eq. (4.2.1), c is a positive real constant. c is a regularization parameter which determines the trade-off between model complexity and training error. And it is the only changeable parameter in classical SVMs. Thus, it is important to construct a model robust with change of the parameter c.

In nonlinear cases, by applying the kernel trick, the optimization problem 4.2.1 is changed to be

$$\begin{aligned} \max_{\alpha} \mathcal{J}_D(\alpha) &= -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l K(x_k, x_l)\alpha_k \alpha_l + \sum_{k=1}^{N}\alpha_k \\ \text{such that} & \quad \left\{ \begin{array}{l} \sum_{k=1}^{N}\alpha_k y_k = 0 \\ 0 \le \alpha_k \le c, k = 1, \ldots, N \end{array} \right. \end{aligned} \tag{4.2.2}$$

As the solution of the QP problem, the nonlinear SVM classifier takes the form

$$y(x) = \text{sign}[\sum_{k=1}^{N}\alpha_k y_k K(x, x_k) + b] \tag{4.2.3}$$

In which, the function $f(x)$ formulated as follow,

$$f(x) = \sum_{k=1}^{N}\alpha_k y_k K(x, x_k) + b \tag{4.2.4}$$

is defined as the *decision value* in SVMs.

4.3 Fuzzy Decision-making Function

In this section, we construct the fuzzy decision-making function based on the fuzzy logic and the calculation of the decision values in SVMs, as well as statistic methods.

4.3.1 Fuzzy Logic in Real-world Problem

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth which means the truth values lie between ”completely true” and ”completely false”. It was introduced by Dr. Lotfi Zadeh of UC/Berkeley in the 1960’s as a means to model the uncertainty of natural language [76].

Dr. Zadeh says that rather than regarding fuzzy theory as a single theory, we should regard the process of ”fuzzification” as a methodology to generalize ANY specific theory from a crisp (discrete) to a continuous (fuzzy) form. Thus recently researchers have also introduced “fuzzy calculus”, “fuzzy differential equations”, and so on.

For introduction of the thought of fuzzy into our models, what is and how to use the fuzzy logic are necessary to be discussed. Just as the strong relationship exists between Boolean logic and the concept of a subset, there is also a similar strong relationship between fuzzy logic and fuzzy subset theory. So fuzzy theory are often used to analysis some set classification problems.

In classical set theory, a subspace S of a set A can be defined as a mapping from the elements of set A to a certain element in the set 0, 1,

$$S : A \rightarrow 0, 1 \tag{4.3.1}$$

Essentially, This mapping can be represented as a set of ordered pairs, with exactly one ordered pair present for each element of A. The first element of the ordered pair is an element comes from the set A, and the second element is an corresponding element of the set 0, 1. The value zero is used to represent that the element of A is a non-membership sample, and the value one is used to represent membership. The statement of the input element of A can be written as

- x from A is not in subspace S, when the ordered pair is (x, 0)
- x from A is in subspace S, when the ordered pair is (x, 1)

So actually, the second element of the ordered pair is a measurement of the statement whether x belongs to S. The statement is true if this element is 1, and the statement is false if it is 0.

Similarly, a fuzzy subset F of a set B can also be defined as a set of ordered pairs, each with the first element from S, and the second element corresponding to it is seated in the interval [0,1], with exactly one ordered pair present for each element of B.

This defines a mapping from the elements of the set A to a value zone denoted as the interval $[0, 1]$. The value zero is used to represent complete non-membership, the value one means complete membership, and values between this two values are used to represent intermediate degrees of membership. Then the set A is referred to as the universe of discourse for the fuzzy subset F. Usually, this mapping is described as a function, the membership function.

Simulate the statement of the crisp set, we can also get a statement for fuzzy set:

- (x is in subspace S) is completely false, when the ordered pair is (x, 0)
- (x is in subspace S) is partially believable
- (x is in subspace S) is completely true, when the ordered pair is (x, 1)

Apparently, the second element of the ordered pair is the degree of truth of the statement. In practice, the terms ”membership function” and fuzzy subset get used interchangeably.

Due to the gray zone built based on the fuzzy logic, the thought of fuzzy can be used to deal with some real-world data set powerfully. Such as description of some actual situations like height, age and so on, and some applications as OCR, handwritten Kanji character recognition, etc.. In

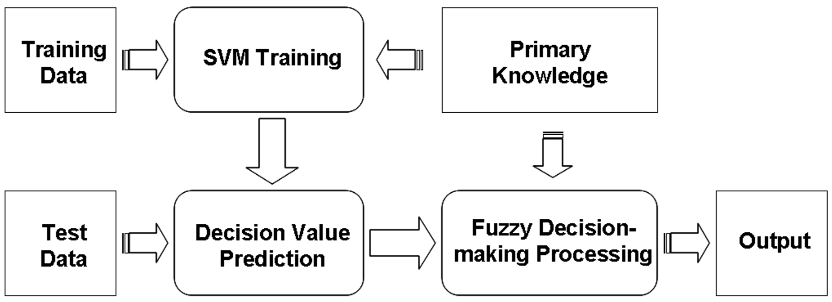


Figure 4.1: General structure of fuzzy decision-making SVM processing.

real-world applications, the construction of membership functions almost are not very simple. If it has more criteria, or to have the membership function depend on elements from many completely different universes of discourse, this kind of problem with a higher dimension will become more complex, but more exactly than classical methods.

Obviously, if you use just the values zero and one into the definitions of fuzzy sets, you could get the same truth tables as you would expect from conventional Boolean logic. This means that the classical results of Boolean logic are recovered from fuzzy logic when all fuzzy membership grades are restricted to the traditional set $\{0, 1\}$. In other words, all the traditional crisp sets could also be viewed as fuzzy sets with this very special type. So there is no conflict between fuzzy and crisp sight.

4.3.2 Structure of the Proposed Model

In our model, we introduce the thought of fuzzy logic to implement an accurate decision-making process, namely using a fuzzy expert systems in the last prediction part of our support vector machine classifiers. This fuzzy decision-making part can be considered as an expert system that uses a collection of fuzzy membership functions and rules, instead of boolean logic, to reason about data, which will be discussed in the following sections in details.

Decision-making SVM processing is a theory proposed based on analysis of database, SVM algorithm and a mass of experiments. It is an extension of traditional decision-making theory which is used to consider the data set as a crisp one. But as we mentioned above, in many real-world applications, each input point may not be separated exactly and during the period of SVM classifier processing not all input points are classified correctly. In other words, some characteristic input points are more easy to be classified but others has incoordinate difficulties.

To measure the degree of each input point whether to be classified correctly or be misclassified, decision value is distilled from conventional decision-making function of SVM, which is a sign function usually. Due to these values associated to different input points indicate the adjacent grades between these points and the optimal separate hyperplane, they can be used as an independent variables in fuzzy decision-making function proposed in this chapter. Modeling the general structure of fuzzy decision-making SVM processing, we can divide it into three main stages: SVM straining, decision value prediction, and at last, fuzzy decision-making processing. Fig 4.1 shows the structure of fuzzy decision-making SVM processing.

In fuzzy decision-making processing part, we construct a fuzzy model to rebuild the decision-making function of SVM with considering the clusters as fuzzy sets.

4.3.3 Fuzzy Decision-making Function

In conventional methods, sign function is always used to divide test data into different classes, so zero is the only one important value as a threshold. But in many real conditions, the relations between classes are not so simple. In both classes, especially around the boundary between them, interaction usually exists. Nearer the other class, the points are disturbed by the noise more acutely. Due to the influences from each other, the judgement on a certain decision value becomes interpretable no more, and zero becomes not so important as well as. Besides, since many classification methods are not good enough and some train datasets are not general for prediction, many misclassified cases occur in the neighbor of threshold. To replace this inflexible division, we build a fuzzy boundary between two fuzzy sets with binary labels. From the previous stage we can get decision value of each input vector, which distributes over the both sides of zero.

Assume that these two fuzzy sets are called A (the value in this set are deemed to be predicted to -1) and B (the values in this set are deemed to be predicted to +1), using decision value as independent variable, a fuzzy decision model could be constructed. In fuzzy theory, there is not a clear boundary between set A and set B, but a gray-zone replace it. Decision values in this zone are not classified to set A or set B determinately. In the method proposed, we built boundary functions using these decision values as independent variables, and the values of the function can indicate their reliability which handles the concept of belief problem (belief degree between 1 (completely believable) and 0 (completely false)). For approaching the real-world cases very well, a very flexible model is needed.

Since the real-world data has a slow shift property of reliability in the higher decision value part and lower decision value part, a function that is more accordant to this property is more effective to

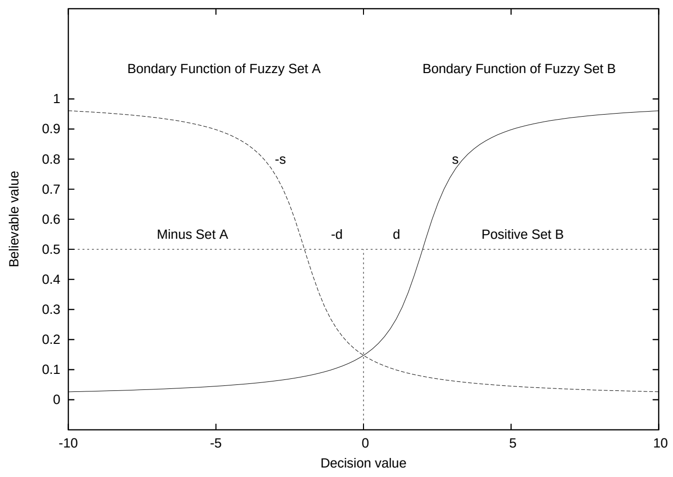


Figure 4.2: Curvilinear fuzzy decision-making model.

built fuzzy decision models. In another words, the value of this function should change by inches when the absolute decision value is small or large enough. To fit these conditions, arc-tangent function is taken into account. Accordingly, for two fuzzy sets A (minus set) and B (positive set) built before, arc-tangent functions are adopted as the boundary functions of these two fuzzy sets. Due to its curvilinear peculiarity, this model (Fig. 4.2) can be called curvilinear fuzzy decision model.

As shown in the figure above, d indicates amphibolous degree between sets, and s means the scale factor, both of which are positive. If the decision value $f(x)$ is denoted by v, and transforming the fuzzy decision value to the range from 0 to 1 directly, the boundaries of set A and set B can be defined as follows:

$$f_A(v) = \frac{\arctan(-v \cdot s - d \cdot s)}{\pi} + 0.5; \tag{4.3.2}$$

$$f_B(v) = \frac{\arctan(v \cdot s - d \cdot s)}{\pi} + 0.5. \tag{4.3.3}$$

4.4 Weighted Harmonic Mean Offset

4.4.1 Offset in Fuzzy Decision-making Function

In many real cases, the borderline is not as appropriate as the description in formulae (4.3.2) and (4.3.3). As a result of imbalance of train data, in many datasets the number of points in one class is much larger than in the other one, so the midpoint of the gray zone between two sets always not equals to zero. So an offset constant m is needed to be introduced to denote the distance between the real borderline and the theoretic one. One way to calculate m is to compute the mean of decision values of support vectors. Suppose that we have got many support vectors, then we can use them as test data to gain their decision values $f(SVs)$ expressed by $S_1, \ldots, S_n$, where n is the number of support vectors. A simple way to calculate the offset δ is computing the arithmetic mean of SVs' decision values. That is because SVs are samples nearest to the boundary. The formula can be written as follows:

$$\delta = \frac{\sum_{i=1}^{n} S_i}{n} \tag{4.4.1}$$

$f_A(v)$ and $f_B(v)$ become new referenced values concerning reliability to predict the output label instead of decision values. Thanks to the existence scale factor s, the method proposed becomes more adjustable. This parameter can be adjusted to find a suitable width of gray zone and the changing rate of function value. And then for transforming the fuzzy decision value to the range between 1 to 0, formulae (4.3.2) and (4.3.3) can be rewritten as:

$$f'_A(v) = \frac{\arctan(-v \cdot s - d \cdot s + \delta \cdot s)}{\pi} + 0.5 \tag{4.4.2}$$

$$f'_B(v) = \frac{\arctan(v \cdot s - d \cdot s - \delta \cdot s)}{\pi} + 0.5; \tag{4.4.3}$$

where d, and s are chosen from a great deal of experiments. We can use the values $f'_A(v)$ and $f'_B(v)$ to estimate whether an input vector should be labeled as +1 or -1. Based on the formulae (4.4.2) and (4.4.3), then the shells of sets in Fig. 4.2 are changed to the model with a modify constant (Fig. 4.3).

4.4.2 WHM offset

The arithmetic mean offset can modify the separation boundary into a better position from the conventional, but this offset will not be accurate if some SVs are contaminated by noise in the real-world datasets. In order to determine a more appropriate offset parameter, we must reduce the contribution from SVs affected by noise. For this purpose, we introduce a weighted harmonic mean (WHM) of SVs' decision values to replace the arithmetic mean offset. As same as the arithmetic mean algorithm, the WHM algorithm also uses SVs as the test data to obtain their decision values $S_1, \ldots, S_n$, where n is the number of SVs. Assume that $\beta_1, \ldots, \beta_n$ are weights corresponding to the SVs, then the offset is defined by

$$\delta = \frac{\sum_{i=1}^{n} \beta_i}{\sum_{i=1}^{n} \frac{\beta_i}{S_i}} \tag{4.4.4}$$

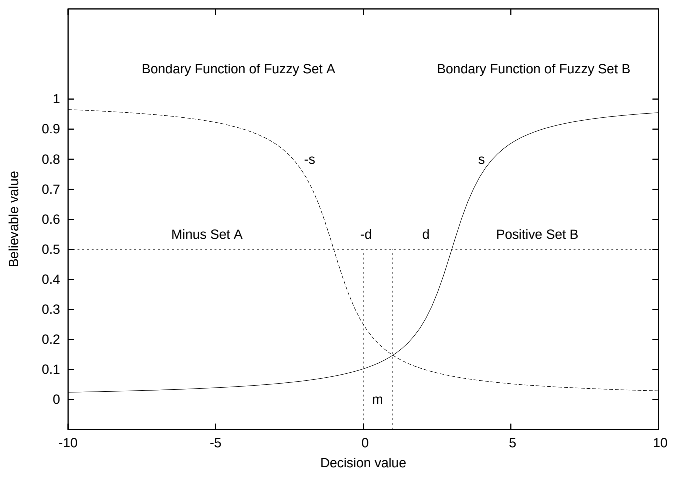


Figure 4.3: Curvilinear fuzzy decision model with an offset.

Before estimating weights $\beta_1, \ldots, \beta_n$, we first consider an example. Taking SVs into account, it is clear from Fig. 4.4 that SVs from subset A are the points of A ”on” or ”below” bounding plane $w^T x = b + 1$ and, similarly, SVs from B are those points of B ”on” or ”above” plane $w^T x = b - 1$. Both SV_1 and SV_2 are from subset A, but SV_1 is affected more by noises than SV_2. If some SVs are strongly influenced by noise, these samples will be apart from the separation boundary.

The decision value can denote the distance between the SV and the separation boundary. If we use decision values of SV contaminated by noise, such as SV_1, to calculate offset, the offset will be incorrect, so we must give a smaller weight to SV_1 to reduce its contribution to the offset calculation. We thus use adopt the Blackman function to calculate corresponding weights of SVs, the function is defined as follows:

$$\begin{aligned}\beta_i = 0.42 - 0.5\cos(&\frac{\pi(S_i + S_{max})}{S_{max}}) \\ +0.08\cos(&\frac{2\pi(S_i + S_{max})}{S_{max}})\end{aligned} \tag{4.4.5}$$

where S_{max} is the largest absolute value of all SVs' decision values, S_i is the i-th SV's decision value and β_i denotes its corresponding weight. The Blackman function is shaped as shown in Fig.4.5.

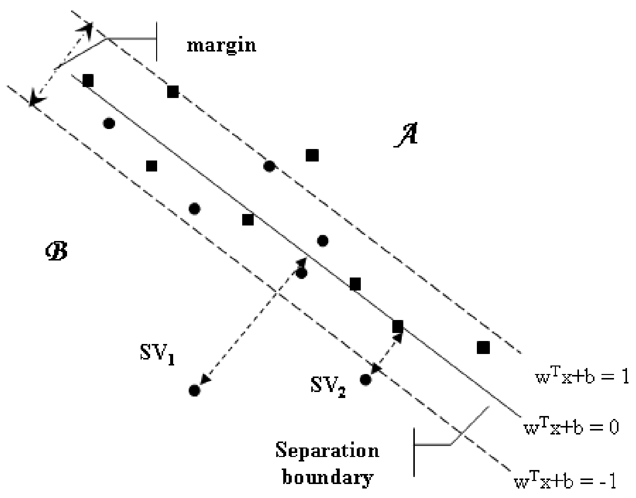


Figure 4.4: SVs in classification.

4.5 Simulations

4.5.1 Classification of Diabetes Diagnosis Data

Description of problem

We first tested our models on the Pima Indians diabetes diagnosis problem from the UCI database. The whole data consists of two classes (tested negative class and tested positive class). The total number of examples is 768, in which the number of samples in the negative class is 500 and the number of samples in the positive class is 268. It is a slightly unbalanced data. Each pair of input vector and output label can be denoted as $\{X(n), Y(n)\}$, where $X(n)$ is the input vector and $Y(n)$ is the output label with two poles: -1 (tested negative class) or +1 (tested positive class), $(n = 1, 2, \ldots, 768)$. Each sample in this data has 8 main attributes, so the vector $X(n) = (x_1(n), x_2(n), \ldots, x_8(n))$. These significations of these 8 attributes are: number of times pregnant, plasma glucose concentration a 2 hours in an oral glucose tolerance test, diastolic blood pressure $(mm\ Hg)$, triceps skin fold thickness (mm), 2-Hour serum insulin $(mu\ U/ml)$, body mass index (weight in $kg/(height\ in\ m)^2$), diabetes pedigree function, age.

In brief, our purpose is to predict the bipolar output $Y(n)$ with as few sign errors as possible, through the given input vectors $X(n)$.

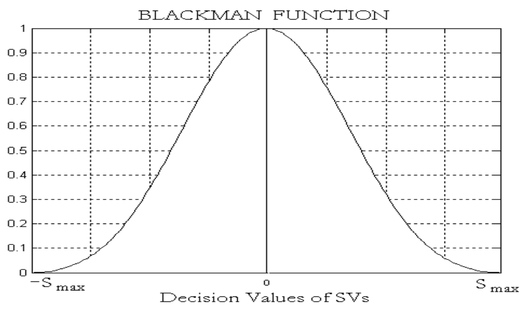


Figure 4.5: Shape of Blackman function.

Table 4.1: WHM offset δ in simulation 1.

offsets	$c = 1$	$c = 10$	$c = 100$
δ (RBF SVM)	0.5	1.2	0.8
δ (Poly SVM)	0.7	1.4	2.1

Results of Classification

Because the database used in this simulation comes from the real-world, it is nonseparable apparently. In SVM, not only the kernel function but also the regularization parameter C can affect the accuracy of classifiers. For the sake of confirming the applicability of proposed method, we let C equal to 1, 10, 100 in turn as common situations, and two different kinds of kernels are also used. They are RBF kernel (4.5.1) and Polynomial kernel (4.5.2).

$$K(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2}) \tag{4.5.1}$$

$$K(x_i, x_j) = (\langle x_i, x_j \rangle + 1)^d \tag{4.5.2}$$

where σ^2 is the common width of RBF and d is the exponential quantity of the polynomial.

Firstly, we calculate the WHM offset values for different combinations of kernels and values of C by using the decision values of SVs. And the results are shown in Table 4.1. By introducing these WHM offsets into SVM classifiers, two classification models are constructed: RBF-SVM classifier with WHM offset and Polynomial-SVM classifier with WHM offset.

Using the first half samples as the training data, and considering the other samples as the test data, the accuracies of these two proposed models are obtained. In order to test their performances,

Table 4.2: Models for unbalanced data in simulation 1.

models	$c = 1$	$c = 10$	$c = 100$
model A (*RBF Fuzzy Decision-making SVM classifier with WHM offset*)	**78.5%**	**78.8%**	**76.7%**
model B (*Poly Fuzzy Decision-making SVM classifier with WHM offset*)	76.3%	70.7%	72.2%
model C (*RBF WSVM classifier*)	78.2%	78.7%	76.1%
model D (*Poly WSVM classifier*)	76.1%	70.6%	72.0%

Legends: Better results are in bold-type.

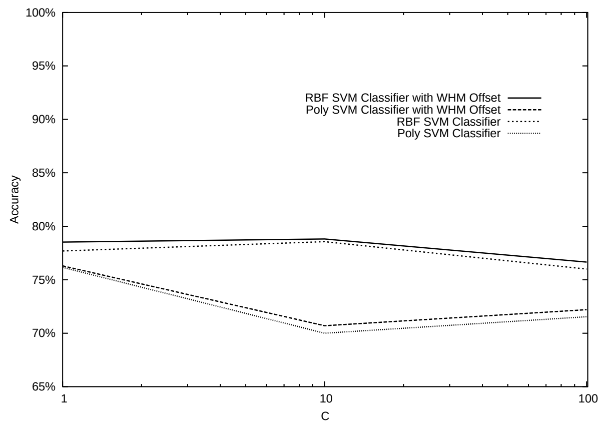


Figure 4.6: Accuracy curves for diabetes diagnosis.

we compare them with WSVM classifier models, the experiment results are shown in Table 4.2. Comparing model A (*RBF Fuzzy Decision-making SVM classifier with WHM offset*) with model C (*RBF WSVM classifier*) and comparing model B (*Poly Fuzzy Decision-making SVM classifier with WHM offset*) with model D (*Poly WSVM classifier*), we can find that the performances of the proposed models are a little better than classical approaches. Compared to corresponding WSVMs, we obtain an increase of 0.3%, 0.1% and 0.6% for RBF kernel, and an increase of 0.2%, 0.1% and 0.2% for polynomial kernel, respectively. The improvements of performances are not obvious in this Pima Indians diabetes diagnosis problem.

Comparing with classical RBF-SVM classifier, classical Polynomial-SVM classifier, the accuracies (y-axis) with three different values of parameter C (x-coordinate axis) are shown in Fig. 4.6.

In the aspect of parameter selection, we choose a standard RBF kernel with common width $\sigma^2 = 4$ and Polynomial kernel with $d = 3$.

Table 4.3: WHM offset δ in simulation 2.

offsets	$c = 1$	$c = 10$	$c = 100$
δ (RBF SVM)	1.1	3.0	2.6
δ (Poly SVM)	1.5	4.3	9.8

4.5.2 Classification of Heart Disease Detection Data

Description of problem

The second problem is heart disease detection, which is from Statlog database. It consists of two classes: absence class and presence class. The number of samples is 200, in which the size of the absence class is 150 and the size of the presence class is 50. Obviously, this database is unbalanced. Each sample has 13 main attributes, which are extracted from a larger set of 75. Assume that the vector $X(n) = (x_1(n), x_2(n), \dots, x_{13}(n))$, then the significations of these 13 elements are: age, sex, chest pain type, resting blood pressure, serum cholestoral in mg/dl, fasting blood sugar, resting electrocardiographic results, maximum heart rate, exercise induced angina, oldpeak, the slope of the peak exercise ST segment, number of major vessels and thal(3=normal; 6=fixed defect; 7=reversable defect). The output label $Y(n)$ has two poles: -1 (absence) or +1 (presence), $(n = 1, 2, \dots, 200)$.

Results of Classification

We also let C equal to 1, 10, 100 in turn, and use RBF kernel and Polynomial kernel. The WHM offset values for different combinations of kernels and values of C are shown in Table 4.3.

Using the first 70 samples as the training data, and considering the remainder as the test data, the accuracies of different classifiers are obtained. The performances of model A (*RBF Fuzzy Decision-making SVM classifier with WHM offset*), model B (*Poly Fuzzy Decision-making SVM classifier with WHM offset*), model C (*RBF WSVM classifier*) and model D (*Poly WSVM classifier*) are shown in Table 4.4. From the results of this problem, we can find that the performances of the proposed models are also better than classical approaches. Compared to corresponding WSVMs, we obtain an increase of 1.3%, 1.2% and 1.4% for RBF kernel, and an increase of 0.9%, 1.0% and 0.9% for polynomial kernel, respectively.

Comparing with classical RBF-SVM classifier and classical Polynomial-SVM classifier, the accuracies curves are shown in Fig. 4.7.

In the aspect of parameter selection, we also choose a standard RBF kernel with common width

Table 4.4: Models for unbalanced data in simulation 2.

models	$c = 1$	$c = 10$	$c = 100$
model A (*RBF Fuzzy Decision-making SVM classifier with WHM offset*)	**90.6%**	**94.4%**	**94.6%**
model B (*Poly Fuzzy Decision-making SVM classifier with WHM offset*)	89.4%	92.3%	93.0%
model C (*RBF WSVM classifier*)	89.3%	93.2%	93.2%
model D (*Poly WSVM classifier*)	88.5%	91.3%	92.1%

Legends: Better results are in bold-type.

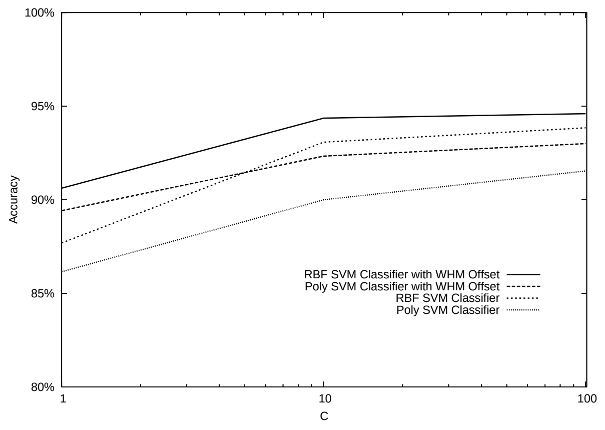


Figure 4.7: Accuracy curves for heart disease detection.

$\sigma^2 = 4$ and Polynomial kernel with $d = 3$.

4.5.3 Classification of Misfire Detection Data

Description of problem

The third database is also from real-world, which is a misfire detection problem in internal combustion engines. The whole database is divided into two subsets, one is used as the training data and the other is the test data. These data contain the information of time series of 50000 samples which are produced by physical system (10-cylinder internal combustion engine). Similarly as the simulation 1, each sample k of time series consists of four inputs elements and one output label, each pair of input vector and output label also can be written as $\{X(k), Y(k)\}$, where $X(k)$ is the input vector and $Y(k)$ is the output label, $(k = 1, 2, \ldots, 50000)$.

Table 4.5: WHM offset δ in simulation 3.

offsets	$c = 1$	$c = 10$	$c = 100$
δ (RBF SVM)	1.9	2.0	2.2
δ (Poly SVM)	1.9	5.7	11.6

Table 4.6: Comparison of two offsets in simulation 3.

models	$c = 1$	$c = 10$	$c = 100$
model A (*RBF Fuzzy Decision-making SVM classifier with WHM offset*)	**94.6%**	**94.7%**	94.8%
model B (*Poly Fuzzy Decision-making SVM classifier with WHM offset*)	92.1%	93.8%	**94.9%**
model C (*RBF WSVM classifier*)	93.3%	92.9%	92.8%
model D (*Poly WSVM classifier*)	92.0%	93.5%	94.6%

Legends: Better results are in bold-type.

The four elements of input vectors $x_1(k)$, $x_2(k)$, $x_3(k)$ and $x_4(k)$ represent cylinder identifier (first), engine crankshaft speed in Revolutions Per Minute (RPM) (second), load (third) and crankshaft acceleration (fourth) respectively. $Y(k)$ may have two values: -1 (normal firing) or +1 (misfire). The amounts of the normal cases in the training data and the test data are 45093 and 45395 and the numbers of samples in the misfire classes in the training and test data sets are 4907 and 4605. We can find that this database is extremely unbalanced. In this problem, Our purpose is also to detect the value of output $Y(k)$ for a certain given input vectors $X(k)$.

Results of Classification

As described in the simulation 1 and 2, we also let C equal to 1, 10, 100, and use RBF kernel and Polynomial kernel in our experiments respectively. The WHM offsets for different kernels and C are shown in Table 4.5. The comparison of proposed models and WSVM offsets is shown in Table 4.6. Model A, B, C and D have same definitions as Table 4.2 and 4.4. In this problem we can also obtain a better performance by using the proposed model with a WHM offset. The classification accuracies obtained by Model A are 94.6%, 94.7% and 94.8%, respectively, an increase of 1.3%, 1.8% and 2.0% classification accuracies compared to Model C. The performance of Model B is also better than Model D in this problem.

The accuracies of classical classifiers and the proposed models with WHM offset are shown in Fig. 4.8.

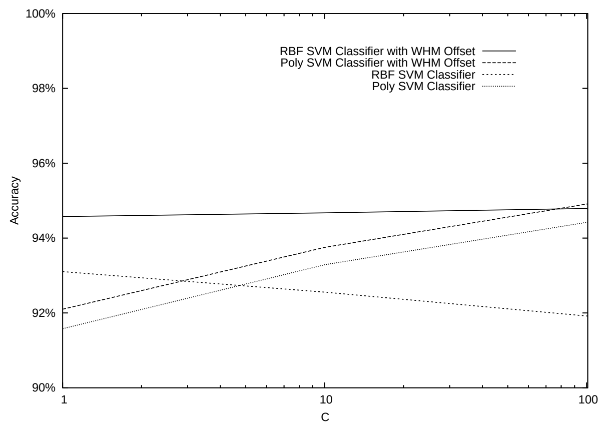


Figure 4.8: Accuracy curves for misfire detection problem.

In the aspect of parameter selection, we also set $\sigma^2 = 4$ for RBF kernel and $d = 3$ for Polynomial kernel.

4.5.4 Discussions

As shown in the simulation results, our proposed model presents better capabilities than classical SVM classifiers and other methods for unbalanced data classification problems. These performances indicate that the separation boundary can be modified to a better position by using WHM offset parameter. The fuzzy decision-making boundary is also suitable for real-world problems.

The three databases used in simulations are all unbalanced, but the degrees of data unbalance are different. We can find that the WHM offsets increase isochronously with the degrees of data unbalance. Furthermore, the effect of the improvement and the degree of data unbalance are in direct proportion. In other words, our proposed approach has a better performance on a more unbalanced problem.

In addition, it is difficult to design a certain SVM classifier with good performances for different problems. But from the three figures above (Fig. 4.6, Fig. 4.7 and Fig. 4.8), it is clear that the proposed model not only has a higher accuracy but also is more robust than classical ones.

As discussed, a more proper decision-making function, a more valid offset, a more appropriate regularization parameter c and a more suitable kernel would make the classifier more effective.

4.6 Conclusions

This chapter presented an fuzzy decision-making algorithm for building SVM classifiers. SVM is a promising technology for some real applications such as high-powered OCR systems, and thanks to its good nonlinear properties by using the kernel trick, it also has been used to solve many academic problems very well, for example, classification problems, regression problems and so on. The fuzzy decision-making models proposed in this thesis can improve the performances of SVM in the classification problems. Along with taking the concept of believable function into account, the prediction boundary is transformed from a straitlaced configuration to a flexible structure which is more similar with real-world cases. Thus the influence between data sets can be reduced and many misclassified points in the prediction part of conventional SVM classifier have a chance to be relabeled, and besides, with the introduction of an offset parameter m, a more correct boundary between each two sets can be confirmed and then the error caused by the imbalance of data sets could be overcome.

Chapter 5

Multiple Support Vector Classifier System

5.1 Introduction

"Divide-and-conquer" is an important algorithm design paradigm and widely used as a powerful tool for solving conceptually difficult problems [10]. A divide and conquer algorithm works by breaking down a difficult problem into two or more sub-problems. The solutions to the sub-problems are then combined to give a solution to the original problem. It has been shown by several researchers that multiple modular systems can result in effective solutions to difficult real-world tasks. Thus some multiple SVM systems are also proposed to solve classification and regression problems in this thesis.

In classifier modeling, multiple classifier systems are proposed based on the principle of "divide-and-conquer" and have been recently used with great success in difficult pattern recognition problems. A main idea of multiple classifier systems is using a committee of classifiers which are trained independently on the difficult training patterns, and combining their decisions to produce the final decision of the system [77]. The training patterns are usually generated by partitioning the original training data set into some sub training sets.

In multiple classifier systems, the original training data set is partitioned by using some unsupervised learning methods such as clustering methods and so on. Some drawbacks with using these classical clustering methods are listed as follows,

- The classification performances closely depend on the initial conditions [77].

- The training data sets for local classifiers are easily imbalanced.

The main reason of these drawbacks is that it is hard to guarantee that the centers of the sub training sets are around the separation boundary. As a result, it is usually required to run these algorithms several times with different initial conditions to discover the best performance.

Another drawback of the multiple classifier systems is that they are easier influenced by noises, because of the disadvantage of having less number of samples in the training subspace.

To address these problems, we try to construct a multiple classifier system that can reduce the influence of noises, and have suitable training data sets for local classifiers.

Differing from system identification problems, the main problem of classification is identifying the separation boundary. Therefore, it is very important to preserve the local characteristics of the separation boundary. With the idea of guaranteeing the centers of training sets around the separation boundary, *we formulate the task of classifier modeling as the piecewise approximation of the separation boundary*. The main difference between the propose model and classical systems is the partition method.

To be specific, the problem is changed to (i) identifying the area around the separation boundary, (ii) partitioning the area into some segments, (iii) finding the piecewise approximations of these segments, and (iv) combining the piecewise approximations. Therefore, we propose an improved multiple classifier system by introducing the piecewise interpolation scheme.

First, the mean filter algorithm is used as a preprocessor to reduce the noises and make the separation boundary clear. Then, a separation boundary detection technique is introduced to preserve the local characteristics around the separation boundary. By using the separation boundary detection algorithm, we identify the area around the separation boundary from the original training data set. And then the area around the separation boundary is partitioned into some disjoint segments based on the rough set theory. Because the segments are from the area around the separation boundary, their centers also lie around the separation boundary. In addition, for avoiding the over-fitting and preserving the distribution properties of the entire data, we also use a clustering algorithm to calculate the centroids of clusters for the entire data set. As an extra benefit, using the centroids from the entire data set can also overcome the problem of data imbalance. The training data sets for local classifiers are formed by the sub data sets of the extracted samples around the separation boundary and the centroids of clusters from the entire data set.

SVMs are used as the basis classifiers. The local SVMs are trained by the respective training data sets to identify the segments of the separation boundary [78]. In SVMs, the separation boundary is constructed based on SVs, which lie close to the separation boundary. Due to the separation boundary detection algorithm, samples that have a higher likelihood of being SVs are preserved.

Thus, the proposed approach will not reduce the performances of SVMs. In addition, the proposed approach can reduce the size of the training data sets. As a result, the training time complexity and space complexity of SVMs are also reduced [79].

Finally, the decisions of the local SVMs are combined to obtain the final classification decision. The decisions combination can be based on two general strategies, namely selection or fusion [80, 81, 82]. In the case of selection, one or more classifiers are nominated "local experts" based on their classification "expertise", whereas fusion assumes that all local classifiers have the expertise over the whole space. A variety of techniques have been applied to implement classifier fusion by combining the outputs of multiple classifiers. In our research, we use a fusion method by combining the decisions of the local SVMs in a probabilistic way.

Main techniques used in the proposed multiple SVM classifier system are that,

- In the preprocessing part, we use the mean filter approach to reduce noises and make the separation boundary clearer. This step is significant for reducing the influence of noises on the sub-tasks.
- In the partition part, the separation boundary detection technique is first used to extract the area around the separation boundary. It is helpful for the later steps to partition the separation boundary and guarantee the centers of training data sets are around the boundary.
- In the partition part, the rough set theory is also introduced as a clustering method. The centers calculated by this method are sparse and can make the sub-tasks balanced.
- In the recognition part, the decisions of local SVMs are combined in a probabilistic based fusion approach.

The remainder of this chapter is organized as follows: Section 5.2 shows a flowchart of the proposed model and introduces the structure of the approach in general. In Section 5.3, we propose a multiple SVM classifier system with piecewise interpolation and discuss the details of the techniques used in the model. Experimental results for the evaluation of the proposed system are presented in Section 5.4. Finally, Section 5.5 is devoted to discussions and conclusions.

5.2 Structure of Multiple SVC System

As stated in the previous section, we formulate the task of classifier modeling as a piecewise approximation of the separation boundary and propose a Multi-SVM Classifier System with Piecewise Interpolation (PMSVC).

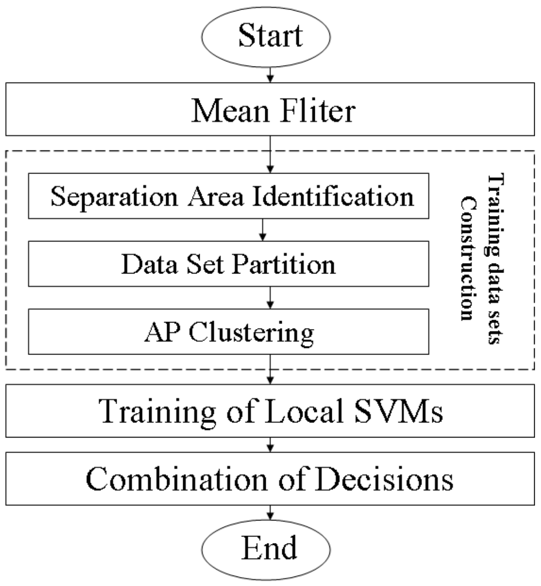


Figure 5.1: Flowchart of the proposed approach.

As shown in Fig. 5.1, the proposed model consists of the following main modules: *mean filter*, *construction of training data sets*, *training of local SVMs* and *combination of decisions*.

Mean filter is used as a preprocessing to reduce noises and make the separation boundary clearer.

After the mean filter, the training data sets for local SVMs are constructed based on the separation boundary detection, the rough set theory and AP clustering. Separation boundary detection identifies the area around the separation boundary. Then this area is partitioned based on the rough set theory. To avoid over-fitting and imbalance, AP clustering is used to calculate the centroids of clusters from the entire training data.

Using these training data sets, we train local SVMs for identifying the respective segments of the separation boundary.

Finally, the decisions of local SVMs are combined based on a probabilistic method to obtain the final decision.

5.3 Piecewise-Interpolation-Based Multi-SVM Classifier System

5.3.1 Mean Filter Algorithm

Consider a binary classification problem and an original training data set D of input-target training pairs $\{x_i, y_i\}$, where $x_i = [f_{i1}, f_{i2}, \ldots, f_{im}]$, and $y_i \in \{-1, +1\}$. f are the features, m is the number of features. If one or more features are influenced by noise, then some samples will be transferred into the region of another class and their corresponding labels will be different from their neighbor samples. Therefore, these samples can be seen as the nonuniform distribution salt and pepper noise in the data set [105].

Here, we introduce a mean filter to reduce this kind of noise. It calculates the mean of neighbor samples' labels and sets a positive threshold θ to calculate the final result [83]. The neighbor samples are selected by a radius r, and θ is used to control the degree of noise reduction. Assume that the number of samples which belong to class +1 is denoted as Num_+ and the umber of samples which belong to class -1 is denoted as Num_-, the function of mean filter can be written as following,

$$\psi_i = \begin{cases} -y_i, & if \ \frac{Num_- - Num_+}{Num_- + Num_+} y_i > \theta \\ y_i, & otherwise \end{cases} \tag{5.3.1}$$

After the mean filter, the original training data set D is changed to the data set D' of the input-target training pairs $\{x_i, \psi_i\}$.

Generally, median filters are more useful to reduce salt and pepper noise [106]. Because binary classification problems only have 2 different labels, the function of median filter can be written as following,

$$\phi_i = \begin{cases} 1, & if \ Num_+ > Num_- \\ -1, & if \ Num_+ < Num_- \\ y_i, & otherwise \end{cases} \tag{5.3.2}$$

Median filter can be seen as a special situation of mean filter when $\theta = 0$.

In simulations, the radius r and the threshold θ are optimized by R$\ddot{a}$tsch's methodology, which trains the algorithm with parameters by a 5-fold-cross validation and selects the parameter to be the median [74]. Of course, these parameters can also be changed manually. However, to avoid filtering some useful information, it is better not to set θ to a too small value.

5.3.2 Separation Boundary Detection

By using mean filter, the noises are reduced to a certain extent and the separation boundary becomes clear. Then we introduce a separation boundary detection method to identify the area around the separation boundary.

In classification problems, the separation boundary detection is implement by detecting the changes of class label between different classes. We first scan the m nearest neighbor samples of an object P. If one of its neighbor samples has a different class label, P is selected to be reserved as a sample around the separation boundary. Details of the separation boundary detection in classification is introduced in Section 2.4.1.

As a result of this separation boundary detection method, a subset D_E is extracted from the the data set D' as the area around the separation boundary. Because the subset D_E contains the samples around the separation boundary, SVs are reserved maximally and the local properties of separation boundary are preserved. Then this subset is partitioned to generate the training sets for local SVMs.

5.3.3 Data Set Partition

Here, we use a simple partition method based on the rough set theory, because the result of this partition method is dispersive and symmetrical. The rough set theory is constructed based on two key definitions: similarity and indiscernibility. The local properties of points depend on how similar they are to each other, thus the form of the similarity matrix S is dependent on the distance measure chosen to determine similarity $s(x_m, x_n)$ between pairs of objects. Here, we set the indiscernibility to a constant H. Assume that the data set D_E selected by the separation boundary detection has N samples x_i, where $i = 1, \ldots, N$, then we can get a mathematical equivalence relation as follows,

$$R = \{x | s(x_m, x_n) < H\}(m, n = 1, \ldots, N) \tag{5.3.3}$$

If the distance between a pair of objects is smaller than the indiscernibility constant H, then these two objects are seen as mathematical equivalence and replaced by their midpoint. In other words, each pair of neighbor samples are replaced by their midpoint. Repeat this process until the the distance between each pair of two objects are all larger than H.

Using these points $\mu_i, i = 1, \ldots, M$ as the centers, a number of sub data sets D_{Ei} are generated from the set D_E. These sub data sets can be used to describe the subtasks for identifying the segments of the separation boundary.

5.3.4 Training of Local SVMs

Sometimes, only using the samples reserved by the separation boundary detection to train SVMs may lead the classifier to over-fitting. In other words, the classifier may be not suitable for solving the entire data set. Hence, we also need to add some data to describe the structural properties of the entire data set. Therefore, we use clustering technique in each class, and the centroids of clusters are also reserved.

In order to have a faster and more effective clustering process, we select Affinity Propagation (AP) to calculate the centroids of clusters from the original training data set D. AP algorithm was introduced as an unsupervised learning algorithm for exemplar-based clustering [69]. In this algorithm, Real-valued messages are exchanged between data points until a high-quality set of exemplars and corresponding clusters gradually emerges. As a result, it has some advantages: speed, general applicability, and suitable for large number of clusters.

In our simulations, the negative Euclidean distance (squared error) was used to measure similarity as common cases. The shared value is set as the median of the input similarities. Then the centers $c_1, c_2, \ldots, c_K$ can be calculated, K is the number of clusters. Obviously, the set of centroids $D_C = \{c_1, \ldots, c_K\}$ is also a subset of the training data set.

Uniting the subsets D_{Ei} and the set of centroids D_C, the training data sets D_i for local SVMs are obtained as follows,

$$D_i = D_{Ei} \bigcup D_C, \;\; i = 1, \ldots, M \tag{5.3.4}$$

where M is the number of training data sets.

Assume that the input-target training pairs of the m-th training data set D_m is denoted as $\{x_{mi}, \psi_{mi}\}$, where $m = 1, \ldots, M$. x_{mi} is the i-th input vector in the m-th training data set and ψ_{mi} is its corresponding class label.

The QP optimization problem (5.3.5) in the dual space of SVM

$$\begin{aligned} \max_{\alpha} \mathcal{J}_D(\alpha) &= -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l K(x_k, x_l)\alpha_k\alpha_l + \sum_{k=1}^{N}\alpha_k \\ \text{such that} &\quad \begin{cases} \sum_{k=1}^{N}\alpha_k y_k = 0 \\ 0 \leq \alpha_k \leq c, \; \forall k \end{cases} \end{aligned} \tag{5.3.5}$$

would be written as:

$$\begin{aligned} \max_{\alpha} Q(\alpha) &= -\frac{1}{2}\sum_{i,l=1}^{N} \psi_{mi}\psi_{ml} K(x_{mi}, x_{ml})\alpha_i\alpha_l + \sum_{i=1}^{N}\alpha_i \\ \text{such that} &\quad \begin{cases} \sum_{i=1}^{N}\alpha_i \psi_{mi} = 0 \\ 0 \leq \alpha_i \leq c, \; \forall i \end{cases} \end{aligned} \tag{5.3.6}$$

where $K(x_{mi}, x_{ml}) = \varphi(x_{mi})^T\varphi(x_{ml})$ is the kernel function. In our simulations, we choose RBF as the kernel function. After training, we obtain the vector of Lagrange multipliers α. If $\alpha_i \neq 0$, then sample-i is an SV.

The general nonlinear SVM classifier takes the form

$$y(x) = \text{sign}[\sum_{k=1}^{N} \alpha_k y_k K(x, x_k) + b] \tag{5.3.7}$$

So the local SVM decision function by using the m-th training data set is obtained as follows:

$$y_m(x) = \text{sign}[\sum_{i=1}^{N} \alpha_i \psi_{mi} K(x, x_{mi}) + b] \tag{5.3.8}$$

where x_{mi} are samples from the training data and x is the test vector. The local SVMs are used as piecewise approximations of the separation boundary segments and trained in parallel in our simulations.

5.3.5 Combination of Decisions

Consider a test vector x, the decisions of local SVMs are $y_i(x),\ i = 1, \ldots, M.$ To obtain the final classification decision, $y_i(x)$ are combined in a probabilistic way. As usual, we adopt the approach to combine the results of the multiple classifiers by computing their fiducial probabilities $P_i,\ i = 1, \ldots, M$, M is the number of SVMs.

The probabilities are calculated by Frossyniotis' methodology, which is based on the Euclidean distances $d(x, \mu_i), i = 1, \ldots, M$ between the new input vector x and the centers μ_i calculated based on the rough set. The probabilities are calculated as follows,

$$P_i = \frac{1}{\sum_{k=1}^{M} \frac{d(x,\mu_i)}{d(x,\mu_k)}}, i = 1, \ldots, M \tag{5.3.9}$$

These probabilities are normalized in the sense that, for all SVMs,

$$\sum_{i=1}^{M} P_i = 1 \tag{5.3.10}$$

Finally, the final output is computed as follows:

$$g(x) = \sum_{i=1}^{M} P_i y_i(x) \tag{5.3.11}$$

where $y_i(x)$ are the prediction functions of SVMs. By using this method, we can construct a combined separation boundary. In the simulations, the effectiveness of the proposed approach is demonstrated through comparisons with some well-known approaches on both synthetic and real-world data sets.

5.4 Simulations

First, we present experimental results of PMSVC on the synthetic Gaussian data set in order to understand the behaviors of the proposed algorithm. Second, we carry out extensive experiments on 5 benchmark data sets.

5.4.1 Classification of Synthetic Data

In the first experiment, we compare PMSVC and the conventional SVM on the synthetic Gaussian data set. Gaussian random binary classification data set is generated randomly. In the generation function, a parameter $sigma$ is used to control the noise intensity of the data set.

In order to obtain the convictive results of performances, we generate 100 data sets for $sigma = 0, 1, \ldots, 10$, respectively. Each data set has 1000 samples. We randomly partitioned every data set into training and testing instances. Both the training and testing sets have 500 samples. All of the results are the average of 100 runs on the data sets. The experiments employ a Gaussian RBF kernel as the basis function.

Figure 5.2 illustrates the data processing of the proposed model PMSVC and the separation boundaries built by PMSVC and SVM. Figure 5.2(a) shows the original data set, where $sigma = 1$. Figure 5.2(b) is the reconstructed training data set by using the mean filter as a preprocessing function. The threshold of mean filter is set to 0.4. According to the figure, some data point far away from separation boundary are reduced and the separation boundary becomes clearer than original training data. Figure 5.2(c) shows the result of the separation boundary detection. The samples around the separation boundary between two classes are extracted. The groups obtained based on the rough set theory and the centroids calculated by AP clustering algorithm are shown in Fig. 5.2(d). The separation boundaries built by SVM and PMSVC are shown in Fig. 5.2(e) and Fig. 5.2(f), respectively.

The intuitional results of comparison between PMSVC and conventional SVM are shown in Figs. 5.3. We compare the classification accuracy, the number of SVs, the time cost of training and the time cost of prediction in this experiment to evaluate PMSVC and SVM. Figure 5.3(a) compare the classification accuracy between PMSVC and SVM. We can find that the proposed approach outperforms SVM in all cases, especially the data sets with high noise. Figure 5.3(b) shows the number of SVs used in PMSVC and SVM. According to the figure, PMSVC employs fewer SVs than the conventional SVM. Figure 5.3(c) shows the time cost of training. The time cost of SVM training in PMSVC is the summation of the time costs of local SVMs training. The total time cost of

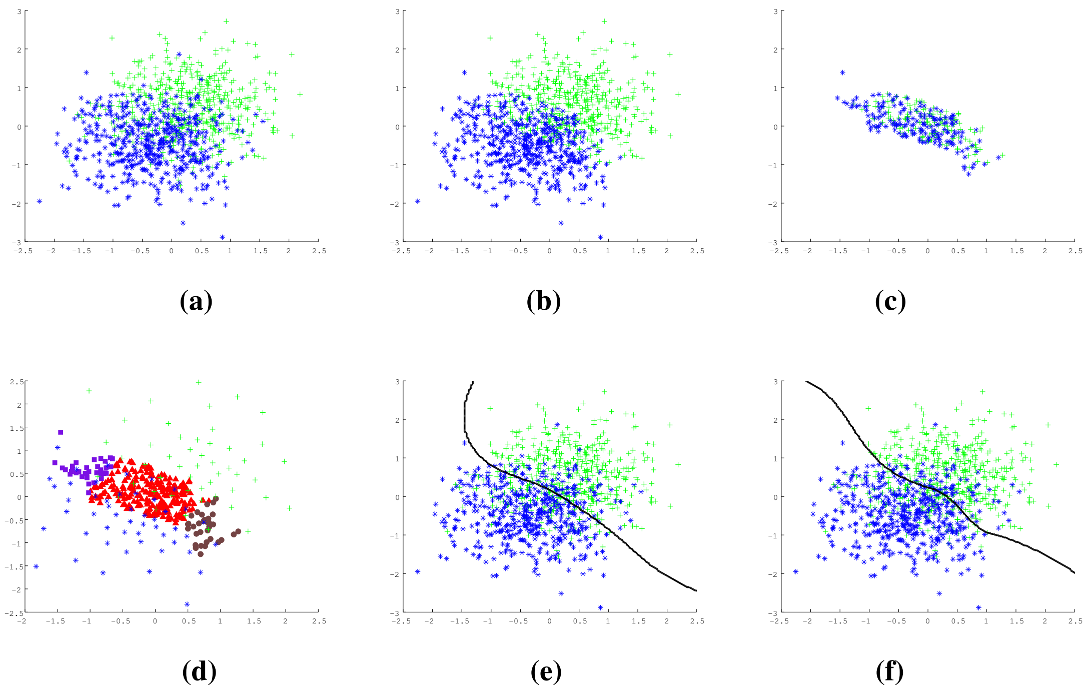


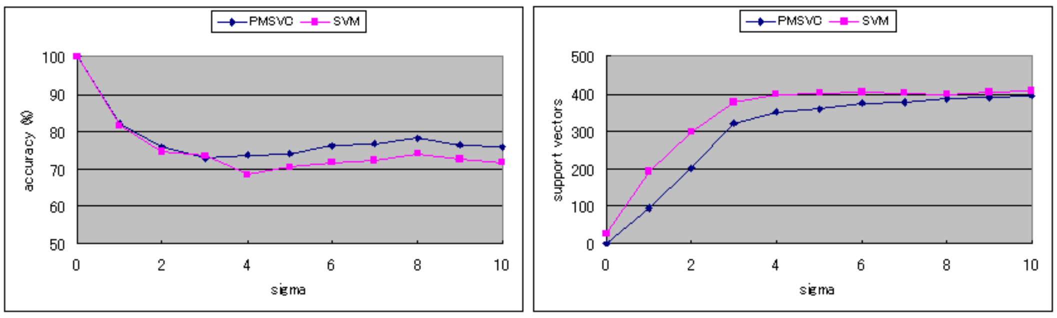
Figure 5.2: Illustration of classification process of PMSVC and SVM: (a)–original training data, (b)–result of mean filter, (c)–result of separation boundary detection, (d)–training data sets, (e)–classifier built by SVM, (f)–classifier built by PMSVC.

the whole training process in PMSVC includes the time costs of mean filter, training data generation and the training of SVMs. From the figure, we can find PMSVC is obviously better than SVM for the time cost of SVM training in most cases. Figure 5.3(d) shows the time costs of prediction in PMSVC and SVM. Because the decisions of SVMs are made in parallel in PMSVC, the time cost of prediction in PMSVC is also smaller than SVM.

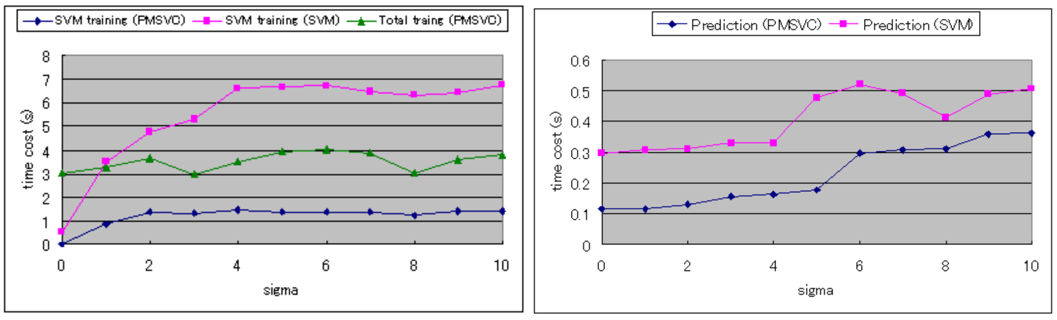
The results of PMSVC are promising on the synthetic Gaussian data set. PMSVC not only improves the classification accuracy, but also effectively controls the time costs of training and prediction.

5.4.2 Classification of Benchmark Data

In order to evaluate the performance of PMSVC further, we compare different algorithms on 5 well-known benchmark problems. These algorithms are one-nearest neighbor (1NN) and k-nearest



(a) (b)



(c) (d)

Figure 5.3: Comparison of performances between PMSVC and SVM: (a)–classification accuracy, (b)–number of support vectors, (c)–time cost of training, (d)–time cost of prediction.

neighbor (kNN), where the number of nearest neighbors k is selected from $\{1, 2, \ldots, 20\}$ by the parameter selection methodology, linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), soft-margin SVMs (SVM$_{soft}$), hard-margin SVMs (SVM$_{hard}$), relevance vector machines (RVMs), and PMSVC [84, 85, 86].

The data sets are all preprocessed for binary classification simulations. These data sets include 5 benchmark real-world data sets from the University of California at Irvine (UCI) Machine Learning Repository [104]. A summary of the data sets is presented in Table 5.1.

In order to obtain convictive results, we randomly partitioned every data set into 100 training and testing instances. All of the results are the average of 100 runs on the data sets. In addition, the data sets are all preprocessed to make the instances have zero mean and unit standard deviation.

For kNN, SVM$_{soft}$, SVM$_{hard}$, RVMs, we use a parameter selection methodology to select

Table 5.1: Summay of 5 benchmark data sets.

Data	No. Train	No. Test	Positive %	Negative %	d*
Cancer	200	77	29.28%	70.72%	9
Diabetics	468	300	34.90%	65.10%	8
German	700	300	30.00%	70.00%	20
Heart	170	100	44.44%	55.56%	13
Titanic	150	2051	58.33%	41.67%	3

Table 5.2: Comparison of 1NN, kNN, LDA, QDA, SVM$_{soft}$, SVM$_{hard}$, RVM, and PMSVC on 5 benchmark data sets, by percent error, and (standard deviation). These results are the average of 100 runs on the data sets.

Models	Cancer	Diabetics	German	Heart	Titanic
1NN	37.7 (4.8)	30.2 (2.2)	29.4 (2.4)	23.2 (3.5)	33.2 (11.5)
kNN	29.9 (3.2)	25.6 (2.1)	25.3 (2.6)	17.8 (3.3)	23.4 (4.6)
LDA	31.8 (4.3)	25.1 (2.1)	28.4 (2.5)	**16.4 (2.8)**	23.5 (4.3)
QDA	31.1 (4.8)	27.7 (1.8)	29.9 (2.8)	19.8 (3.4)	24.6 (6.4)
SVM$_{soft}$	26.3 (4.6)	24.4 (1.8)	25.0 (2.3)	17.9 (3.3)	23.7 (1.0)
SVM$_{hard}$	29.0 (4.8)	30.6 (2.2)	26.8 (2.3)	22.3 (3.4)	**22.3 (1.1)**
RVM	26.6 (4.7)	24.0 (1.7)	25.1 (2.3)	17.6 (3.5)	23.4 (1.0)
PMSVC	**25.8 (3.6)**	**23.3 (2.3)**	**24.6 (2.2)**	**16.4 (2.4)**	23.4 (1.1)

Legends: The best results are in bold-type.

the optimal parameters. The procedure of parameter optimization is implemented by training the algorithm with candidate parameters on the first five training sets and selecting the median over those five estimates as the optimal parameter. In affinity propagation part, the shared value is set as the median of the input similarities as common cases.

Table 5.2 shows the performance of these 8 algorithms on the 5 benchmark data sets with error. According to that table, the PMSVC performs better than other approaches. For example, it is observed that the PMSVC outperforms all other methods in 4 out of 5 benchmark data sets and comes the third in the remaining one. From these enlightening experimental results, we can find that the soft-margin SVM is better than the hard-margin SVM in most cases. But 1NN, kNN, LDA, and QDA, only perform well on one or two data sets. In other cases, they fail to compete with SVMs and PMSVC.

Table 5.3 shows the number of vectors employed by SVMs and the proposed PMSVC. It is very interesting that the proposed approach can achieve better performance by employing only a few of the data points. That means conventional SVMs employ too many useless data points which are influenced by noises.

Table 5.3: Comparison of SVM$_{soft}$, SVM$_{hard}$, RVM, and PMSVC on 5 benchmark data sets, by number of vectors and standard deviation. These results are the average of 100 runs on the data sets.

Data	No. Train	SVM$_{soft}$	SVM$_{hard}$	RVM	PMSVC
Cancer	200	119.9±6.4	105.7±6.4	9.4±2.0	94.9±7.2
Diabetics	468	270.6±7.9	376.0±7.0	8.4±2.0	255.1±7.0
German	700	520.9±10.9	514.4±12.4	27.4±4.3	362.4±15.2
Heart	170	111.7±4.8	104.8±6.1	9.3±1.8	90.1±5.8
Titanic	150	147.5±4.7	143.5±6.3	6.8±1.4	21.0±5.0

According to Table 5.3, the number of support vectors for SVM grows almost linearly with the size of training sets, while RVMs consistently use much fewer vectors. PMSVC employs more vectors than the RVM but fewer than SVM.

5.4.3 Statistical Analysis

In order to compare the PMSVC with other algorithms in a sound statistical context, we perform the statistical test for paired classifiers, e.g., 1NN vs. PMSVC and SVM vs. PMSVC, on the 5 benchmark data sets. The statistical test can provide the win-loss-tie summary based on the accuracy of classifiers. The threshold of the statistical tests is set to be 0.05.

Although t test has been used in most of the literatures to conduct statistical tests, it has been criticized for its type I/II error and low power for a long time [87]. Dietterich analyzes five statistical tests and proposes a new test, five replications of two fold cross-validation t test, i.e., 5×2 cv t test, which has a low type I error and a reasonable power [87].

However, 5×2 cv t test takes the statistics from only one fold as the numerator and may vary depending on factors that should not affect the test. Alpaydin improved 5×2 cv t test by combining multiple statistics to get a more robust test, 5×2 cv F test, which has a lower type I error and a higher power. Thus, we compare algorithms using 5×2 cv F test [88].

In the 5×2 cv F test, five replications of twofold cross-validation have been conducted. In each replication, the data set is divided into two equal-sized sets. $p_i^{(j)}$ is the difference between the error rates of the two classifiers on fold $j = 1, 2$ of replication $i = 1, 2, \ldots, 5$. The average on replication is $\bar{p}_i = (p_i^{(1)} + p_i^{(2)})/2$, and the estimated variance is $s_i^2 = (p_i^{(1)} - \bar{p}_i)^2 + (p_i^{(2)} - \bar{p}_i)^2$.

The 5×2 cv F test combines the results of the ten statistics p_i^j as the numerator, which makes the test more robust. Alpaydin pointed out that the following statistics:

$$f = \frac{\sum_{i=1}^{5} \sum_{j=1}^{2} (p_i^{(j)})^2}{2 \sum_{i=1}^{5} s_i^2} \tag{5.4.1}$$

Table 5.4: 5×2 cv F test for 5 data sets. For each metric, the second column is the win-loss-lie summary of the algorithm against PMSVC based on the mean value of error. The third column gives the statistical significance win-loss-lie summary based on 5 benchmark data sets.

Paired Models	Error Mean	Significance
1NN vs. PMSVC	0-5-0	0-5-0
kNN vs. PMSVC	0-5-0	0-4-1
LDA vs. PMSVC	0-5-0	0-3-2
QDA vs. PMSVC	0-5-0	0-5-0
SVM$_{soft}$ vs. PMSVC	0-5-0	0-4-1
SVM$_{hard}$ vs. PMSVC	1-4-0	1-4-0
RVM vs. PMSVC	0-5-0	0-4-1

Table 5.5: Comparison between SVM, and PMSVC on 5 benchmark data sets, by the time cost of SVM training, and the total time cost of the model construction. These results are the average of 100 runs on the data sets.

Time Cost of SVM Training	SVM	PMSVC
Cancer	1.125	0.634
Diabetics	38.43	1.032
German	156.1	0.905
Heart	1.765	0.513
Titanic	1.109	0.402

Total Time Cost	SVM	PMSVC
Cancer	1.125	0.963
Diabetics	38.43	29.50
German	156.1	137.8
Heart	1.765	1.225
Titanic	1.109	0.987

is approximately F distributed with ten and five degrees of freedom, $F(10, 5)$, and used this statistics to conduct the 5×2 cv F test.

Table 5.4 gives the win-loss-tie summary of the 5×2 cv F test based on 5 benchmark data sets. According to the significance tests, we find that SVM only wins once and loses four times. The RVM never wins. The performance of 1NN, kNN, LDA, QDA, SVM$_{soft}$, SVM$_{hard}$ and RVM are all not competitive against the proposed approach PMSVC.

5.4.4 Discussions

As discussed in previous sections, PMSVC partitions the training data into some subtasks. In each subtask, only the points around the separation boundary and the centroids calculated by AP clustering are used to train a local SVM in PMSVC. Training of the local SVMs is implemented independently. Therefore, the time cost of SVM training in PMSVC is smaller than the conventional SVM. This is an extra benefit obtained by the proposed model. Of course, the training process can also be implemented in parallel to obtain a lower time cost.

Table 5.5 shows the comparison between SVM and PMSVC on 5 benchmark data sets, by the

Table 5.6: Comparison between SVM, and PMSVC on 5 benchmark data sets, by the time cost of prediction. These results are the average of 100 runs on the data sets.

Time Cost of Prediction	SVM	PMSVC
Cancer	0.050	0.023
Diabetics	0.227	0.190
German	0.657	0.378
Heart	0.063	0.056
Titanic	0.093	1.025

time cost of SVM training, and the total cost of the model construction. The time cost of SVM training in PMSVC is the summation of the time costs of local SVMs training. The total time cost of the model construction in PMSVC includes the time costs of mean filter, training data sets generation and local SVMs training. According to the table, the time costs of SVMs training in PMSVCs are obviously smaller than SVMs. In PMSVC, AP is a fast clustering method, and the similarity matrix is communal in the training part. Therefore, the total time cost of PMSVC is also smaller than SVMs.

Table 5.6 shows the comparison between SVMs and PMSVC on 5 benchmark data sets, by the time cost of prediction. The time costs of PMSVCs are also smaller than SVMs in the test stage. That is because the prediction part of the proposed approach is implemented independently and the number of SVs in each subtask is small.

5.5 Conclusions

In this chapter, a multiple SVM classifier system with piecewise interpolation (PMSVC) is proposed for solving classification problems and obtaining better performances. PMSVC focuses on finding locally optimal fits of separation boundary for complex real-world data. In other words, in a local area of data space, PMSVC can construct a more suitable classifier than conventional SVM.

The main difference between classical multiple classifier systems and PMSVC is the approach of partition. PMSVC is implemented by detecting the area around the separation boundary and partitioning the area into some sub sets. These sub data set describe the segments of the area around the separation boundary. The set of centroids calculated by AP clustering on original data set and the sub data sets constructed after the partition form the training data sets to train local SVMs. The decisions of SVMs are combined based on a probabilistic interpretation. Because the multi-classifier modeling is easier influenced by the noise, mean filter algorithm is introduced into the proposed model. Mean filter is used as a preprocessor to filter noises. In the comprehensive study of PMSVC on synthetic Gaussian data set and 5 benchmark problems, simulation results confirm that the PMSVC performs very well on these data sets. Good performances are obtained in terms of accuracy and time cost.

Chapter 6

Multiple Support Vector Regression Network

6.1 Introduction

Based on the principle of "divide-and-conquer", we also propose a model for the time series prediction. The model is also a multiple SVM system called modular multiple support vector regression network. This research tries to provide a model which can be used in time series prediction, especially for dealing with financial time series.

Many approaches have been used for time series prediction which includes moving average (MA), auto-regressive integrated moving average (ARIMA), artificial neural networks (ANN) and so on [89]. However, few of them have proved convincible.

In the last two decades, ARIMA has been widely used in time series prediction. However, it inherently has its own limitation when doing non-linear prediction. Besides, it will face great uncertainty when it comes to a long-term prediction.

Many researchers have shown that ANN outperforms ARIMA model, especially for more irregular series and long-term prediction [89]. However, ANN can not stand mass noise which is common in financial data, so over-fitting is more likely to happen in ANN models.

Since the financial time series are usually with high noise, it is more suitable to use a prediction approach based on the statistical theory. Support vector machines (SVM) are closely related to statistical models and have been proved useful in a number of regression applications. The SVM used as a regression predictor is called Support Vector Regression (SVR). Compared with ANN, SVR implements the structural risk minimization principle while ANN adopts empirical risk minimization principle, so over-fitting is unlikely to occur with SVR [90][12]. Thus our proposed modular network model consists of SVRs.

Although SVR is better than most conventional regression methods, there are still some fundamental limitations and inherent difficulties when using SVR to do financial time series prediction. The problems are listed as follows,

- The first problem is that the learning process of modeling is usually ill-posed, i.e. there are infinitely many models which fit the training data well, but few of them generalize well. In order to form a more accurate model, it is desirable to use as large a training set as possible. However, for the case of highly non-stationary data, such as financial time series, increasing the size of the training set results in more data with statistics that are less relevant to the task. Therefore, we need to use a moving window to select training data, and the whole process should be online.
- The second problem is the selection of the input variables. Because time series prediction are usually subject to many factors and their influence quantities are difficult to be measured, most of them need to be adopted as input variables. These variables should include the past prices of some correlative foreign exchange rates and correlative commodities, and the information about current market situation as well as macroeconomic parameters.
- The third problem is the evaluation of the relation between inputs and the output. Actually, for different situations, the input variables have different influence quantities on the results of prediction, especially in different price regions of financial data. So it is a hard and complex task to evaluate the relationship between inputs and output. So it is naturally to consider about breaking the task into some subtasks, then employing the machine learning technique and constructing a multiple recognizor system.
- The fourth problem is how to partition data for constructing the multiple recognizor system. In time series prediction, the temporal relationship is often represented as random between inputs and output [91, 92]. However, in different price regions the relationship between inputs and output is also different. So we consider to partition the price domain into some subregions to solve this problem [93], but not implement the partition in the time domain.

Therefore, we propose a multiple SVM regression system based on the principle of "divide-and-conquer". The price domain is partitioned into some subregions. These subregions represent subtasks for different price levels. Local SVRs are trained as "local experts" to solve these subtasks. The outputs of these SVRs are combined by a *combinator* to give the final prediction result.

As an important and typical time series application, we use the foreign exchange (Forex) prediction as an example in the construction and simulations of our proposed model. Forex market is one of the most profitable markets around the world today. Forex prediction is a typical example of financial time series analysis problem which is challenging due to high noise, non-stationarity, and non-linearity [11].

In the construction of the proposed multiple SVM regression system, the main problem is how to partition the price domain. We can partition the price domain equationally or employ some prior knowledge. In this thesis, we try to employ some useful prior knowledge in the price domain partition. In other words, we want to find some important price levels in the price domain. Some literatures have introduced that the support and resistance levels are two of the most significant levels in the price domain. If a price breaks past a support level, support level often becomes a new resistance level. The opposite is true as well. In different subregions between the support and resistance levels, Forex time series have different properties. Therefore, we do the price domain partition based on these levels.

The support and resistance are highly discussed as attributes of technical analysis. Bill Poulos has proved Fibonacci Retracement is a very effective method to calculate the support and resistance levels [13]. As an example, we partition the price domain into several subregions based on Fibonacci support and resistance levels and build SVR model for each subregion. Four SVRs are trained as the "local experts" make up of the transformation layer of the model. Their outputs are used as the input of the prediction layer. In other words, these outputs from the transformation layer are the representations of the inputs for price subregions and the SVR used as the combinator in prediction layer combine them to get the final result. We apply the modular SVR network to give an online prediction of 5 different foreign exchange rates, and find significant predictability in comprehensive experiments.

The main improvements and benefits of the proposed approach are that,

- The influences of inputs on the output in different price domains are evaluated by local SVRs. The local SVRs can give a better prediction result.
- The partition is implemented in the price domain. Although some existing modular methods also split data up into some clusters [11, 94, 95], the partition methods in their model are based on the input variables. The dimension of the price is 1, so it is much smaller than classical approaches and the proposed approach is much more operable.

The remainder of this chapter is organized as follows: Section 6.2 shows the structure of the proposed model and introduces the idea of the approach in general. In Section 6.3, we propose a modular multiple SVM regression network with price domain partition and discuss the details of the techniques used in the model. Experimental results for the evaluation of the proposed system are presented in Section 6.4. Finally, Section 6.5 is devoted to discussions and conclusions.

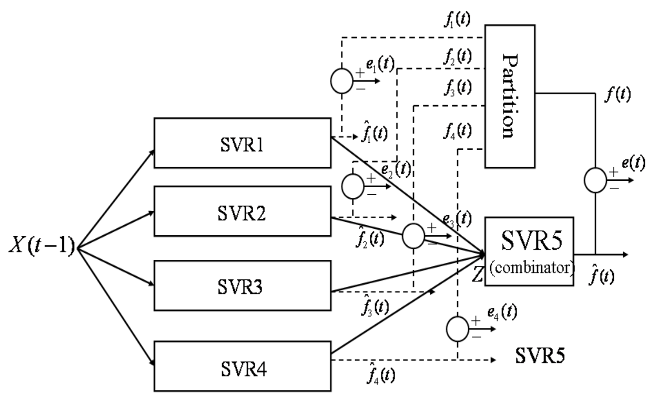
6.2 Structure of Modular SVR Network

In the proposed model, we are interested in using several predictors to fit the potential relationship between the inputs and the output in different price subregions. SVRs are able to construct spline approximations of given data independently from the number of input-dimensions regarding complexity during training and with only linear complexity. Because the input-dimension of the financial time series is commonly higher than other time series, it is suitable to choose SVR as the predictor to deal with this kind of data in the proposed approach.

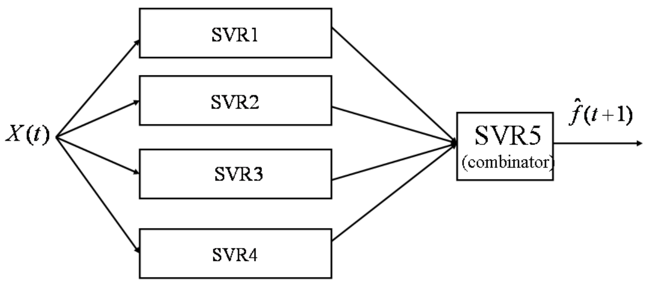
An example of the structure of the proposed modular SVR network is shown in Fig. 6.1. It is clearly that we restrict our attention to a modular network structure with a single transformation layer and a prediction layer. The transformation layer consists of several SVRs (in this example, we use 4 SVRs in the transformation layer). The SVRs transform the input into some symbolic representations. Then these representations, in other words outputs from transformation layer, are used as the input of the prediction layer.

SVRs in the transformation layer are trained by the inputs and the target outputs corresponding to price subregions, respectively. Assume that the modular SVR network has the n-dimension input $x(t) = (x_1, x_2, x_3, ..., x_n)$ at time t. $f(t)$ indicates the price of the a financial time series and $I_m(t)$ indicates other indicators at time t, where $m = 1, 2, \ldots$, our purpose is to predict a future price. For example, if we want to predict the next day's price $f(t+1)$, we can use the past 5 days' data as the input which can be written in details as $x(t) = (f(t), f(t-1), f(t-2), f(t-3), f(t-4), I_1(t), I_1(t-1), I_1(t-2), I_1(t-3), I_1(t-4), I_2(t), I_2(t-1), I_2(t-2), I_2(t-3), I_2(t-4), \ldots)$.

As shown in the exmple (Fig. 6.1), if the price domain is partitioned into 4 subregions, let $f_1(t)$, $f_2(t)$ $f_3(t)$ and $f_4(t)$ indicate the target outputs corresponding to the price subregions, $\hat{f}_1(t)$, $\hat{f}_2(t)$, $\hat{f}_3(t)$ and $\hat{f}_4(t)$ indicate actual outputs of the transformation layer. SVRs in the transformation layer



(a) Modular SVR network training



(b) Modular SVR network prediction

Figure 6.1: Structure of the modular SVR network.

are trained as “*Local Experts*” to minimize the errors

$$\begin{aligned} e_1(t) &= f_1(t) - \hat{f}_1(t) \\ e_2(t) &= f_2(t) - \hat{f}_2(t) \\ e_3(t) &= f_3(t) - \hat{f}_3(t) \\ e_4(t) &= f_4(t) - \hat{f}_4(t) \end{aligned} \tag{6.2.1}$$

.

In the prediction layer, we use another SVR to combine the outputs of the transformation layer to obtain the final prediction result. It is trained by the outputs of the transformation layer and the final target $f(t)$. Minimizing $e(t) = f(t) - \hat{f}(t)$, where $\hat{f}(t)$ is the actual output of SVR5, we can

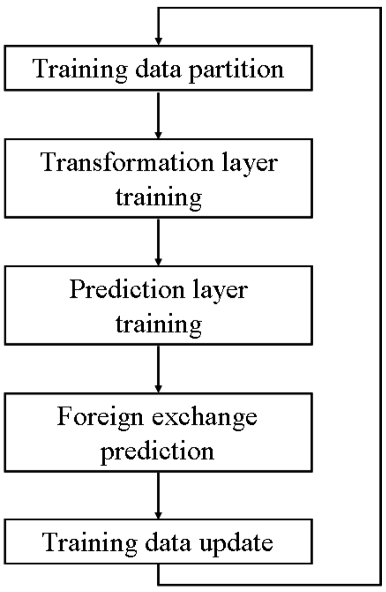

Training data partition
Transformation layer training
Prediction layer training
Foreign exchange prediction
Training data update

Figure 6.2: Flowchart of the online time series prediction.

construct the last SVR as a "*Combinator*".

The modular SVR network gives an online time series prediction. The flowchart of the time series prediction is shown in Fig. 6.2. The prediction process is online, so we update the training data and repeat all operations after the prediction [96].

6.3 Modular SVR Network

6.3.1 Inputs Selection

As a typical application of financial time series, Forex prediction problem is used as an example in this thesis. Forex time series prediction is a highly complicated task and has following properties:

- Forex time series is closed to a random-walk process as some papers mentioned, rendering the prediction impossible from a theoretical point of view.

- Forex time series are always subject to macroeconomic, regime shifting, macro-control and so on, i.e. statistical properties of Forex time series are different at different points in the time domain and the price domain [97].

- Forex time series are usually very noisy, i.e. there is a large amount of random (unpredictable) day-to-day variations [14].

The predictability of most common financial time series is a controversial issue and has been questioned in scope of the efficient market hypothesis (EMH) [98]. Many researches have proved Forex time series is predictable, but the non-stationarity of influence quantities of input variables is still a serious problem in Forex prediction [99].

Because Forex time series are subject to many factors and their influence quantities are difficult to be measured, how to select the input variables is also a problem. Several different types of data could be used as input variables in the Forex time series prediction, but at different region of price domain, their influence quantities are also different. Therefore, we choose several different kinds of data to reflect two main kinds of information: technical information and fundamental information.

- Technical information is usually referred to by Forex prediction. This kind of information includes such figures as past prices etc.

- Fundamental information is the information describing current economic activity in the world or a country whose currency price is to be predicted. Further, fundamental information include the information about current market situation as well as macroeconomic parameters.

In our experiments, we choose several indicators as the input variables. For example, in the prediction of USD/JPY, we choose its past prices, the past values of NASDAQ Composite index, the past prices of Light Sweet Crude Oil and the past values of NIKKEI 225 index as input variables [100, 107, 108]. Of course, this is just an example. We can also choose other variables as the input, if they could be considered influential on the output.

6.3.2 Price Domain Partition

As mentioned in the previous subsection, time series prediction problems are subject to several factors such as some economic indexes, correlative commodities, etc., but its statistical properties are different at different points in the price domain. In other words, it is hard to determine which input variable is more useful in a certain price zone. So it is natural to consider dividing the price domain into some subregions and building a regression model for each subregion. Most existing modular methods also partition the training data into some subsets, but their methods are based on the input variables. Differently, our method only divides the price domain into subregions. The dimension of price is only 1, which is much smaller than the dimension of the input, hence the partition in the proposed approach is much easier.

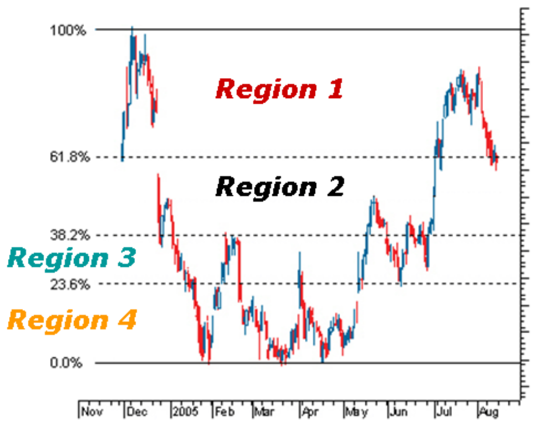


Figure 6.3: Fibonacci support and Fibonacci resistance.

How to partition the price domain is still a problem. Many methods can be used in this part, of course, we are also going to develop some more effective partition method in our future work. In this thesis, we try to employ some useful priori knowledge, so we do the partition based on the support and resistance levels.

In many applications of financial time series, support and resistance are the concepts in technical analysis that the price will tend to stop and reverse at a certain predetermined price level. Support and resistance are the most important levels in financial market, which represent key junctures where the forces of supply and demand meet. If the price breaks past a support level, that support level often becomes a new resistance level. These levels divide the price domain into several subregions.

Bill Poulos has proved Fibonacci Retracement is an effective method to identify the support and resistance levels. So in our research, we use Fibonacci Retracement to identify support and resistance levels at determined intervals. As shown in Fig. 6.3, the support and resistance levels are created by drawing a trendline between two extreme points and then dividing the vertical distance by the key Fibonacci ratios of 23.6%, 38.2%, 61.8% and 100%. Consider a training data set with output T, the maximum exchange rate is T_{Max} and the minimum is T_{Min}. The support and resistance levels are calculated as follows:

$$\begin{aligned} d_1 &= T_{Min} + (T_{Max} - T_{Min}) \times 23.6\% \\ d_2 &= T_{Min} + (T_{Max} - T_{Min}) \times 38.2\% \qquad (6.3.1) \\ d_3 &= T_{Min} + (T_{Max} - T_{Min}) \times 61.8\% \end{aligned}$$

Then the standard output in the training data is partitioned into following 4 regions:

- region 1: $(T_{max} > T > d_3)$
- region 2: $(d_3 > T > d_2)$
- region 3: $(d_2 > T > d_1)$
- region 4: $(d_1 > T > T_{min})$

As shown in Fig. 6.4, the significant data in each price region are reserved as the target outputs of subregions: $f_1(t)$, $f_2(t)$, $f_3(t)$ and $f_4(t)$. They are used in the training process of transformation layer.

6.3.3 Training of Modular SVR network

Following the Vapnik's method, we can estimate function $\hat{f}(t+1)$ using given training set and the result is:

$$\hat{f}(t+1) = \hat{g}(x(t)) = \sum_{k=1}^{N} (\alpha_k^* - \alpha_k) K(x(t), x_k) + b \tag{6.3.2}$$

where Lagrange multipliers α_k and α_k^* are associated with each data point x_k, and subject to the constraints $0 \leq \alpha_k^*, \alpha_k \leq C$ and $\sum_{k=1}^{N} (\alpha_k^* - \alpha_k) = 0$. Training points with nonzero Lagrange multipliers are called support vectors. The kernel function $K(\cdot)$ describes an inner product in the D-dimensional space as below and satisfies the Mercer's condition.

$$K(x(t), x_k) = \sum_{k=1}^{D} \Phi(x(t)) \Phi(x_k) \tag{6.3.3}$$

where $\Phi(x)$ is the nonlinear mapping function. The generalization performance (i.e., accuracy in predicting exchange rates in this study) also depends on the selection of kernel type. Common kernel types are linear, polynomial and Gaussian kernels. In the experiments, we choose linear kernel to build SVR models.

The coefficients α, α^* and the bias b are obtained by solving a quadratic programming problem. In the modular SVR network, assume that α_{1k}, α_{1k}^*, α_{2k}, α_{2k}^*, α_{3k}, α_{3k}^*, α_{4k} and α_{4k}^* denote the coefficients of SVRs in the transformation layer, b_1, b_2, b_3 and b_4 denote the biases, then their

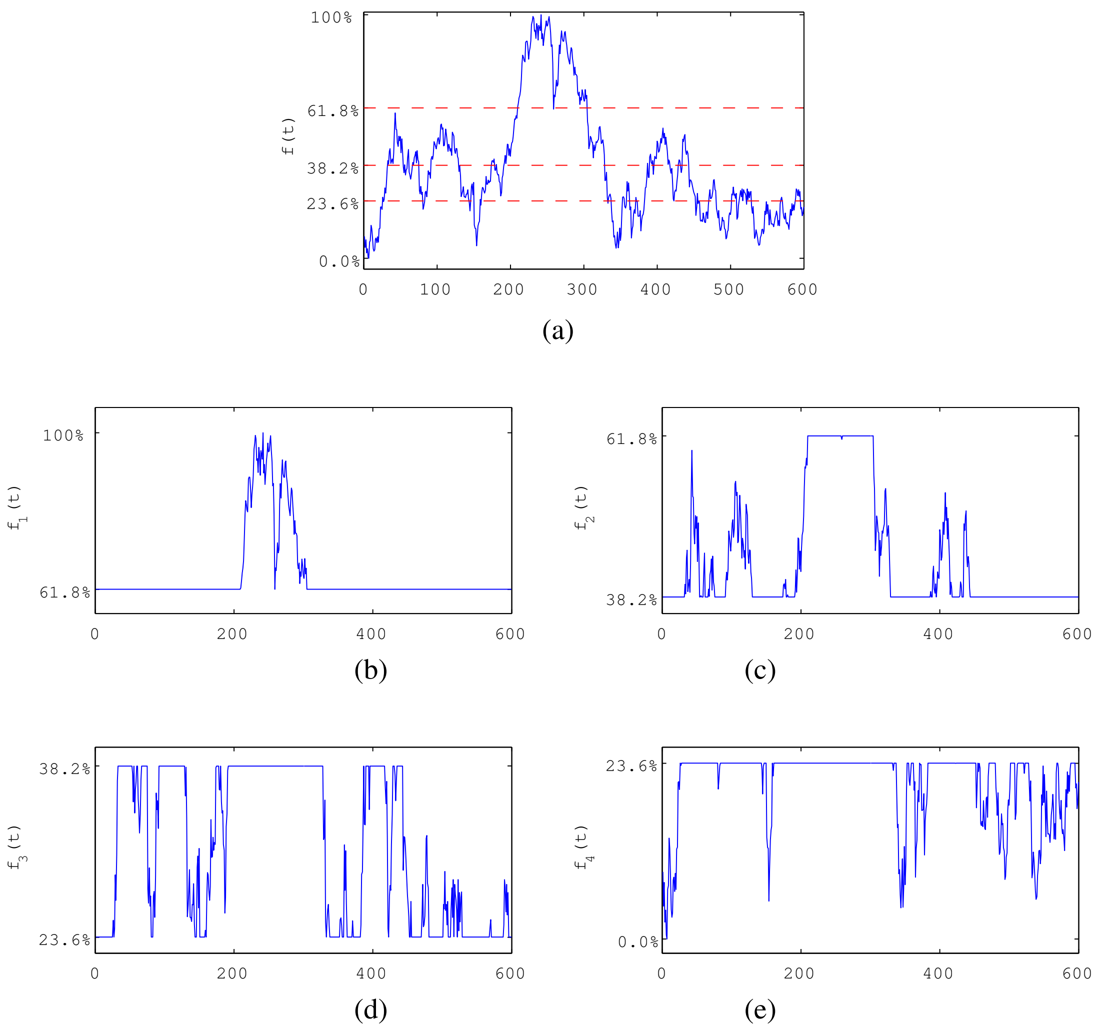


Figure 6.4: Price domain partition: (a) – target output, (b) – region 1, (c) – region 2, (d) – region 3, (e) – region 4.

outputs are written as follows:

$$
\begin{aligned}
\hat{f}_1(t+1) &= \sum_{k=1}^{N}(\alpha_{1k}^* - \alpha_{1k})K(x(t), x_k) + b_1 \\
\hat{f}_2(t+1) &= \sum_{k=1}^{N}(\alpha_{2k}^* - \alpha_{2k})K(x(t), x_k) + b_2 \\
\hat{f}_3(t+1) &= \sum_{k=1}^{N}(\alpha_{3k}^* - \alpha_{3k})K(x(t), x_k) + b_3 \\
\hat{f}_4(t+1) &= \sum_{k=1}^{N}(\alpha_{4k}^* - \alpha_{4k})K(x(t), x_k) + b_4
\end{aligned}
\tag{6.3.4}
$$

$\hat{f}_1(t+1)$, $\hat{f}_2(t+1)$, $\hat{f}_3(t+1)$ and $\hat{f}_4(t+1)$ are the outputs from the transformation layer as well as the input of the prediction layer. Let $z = (\hat{f}_1(t+1), \hat{f}_2(t+1), \hat{f}_3(t+1), \hat{f}_4(t+1))$ indicate the input of prediction layer, then the final solution of the SVR in prediction layer is given as follows:

$$\hat{f}(t+1) = \hat{g}(z) = \sum_{k=1}^{N}(\alpha_{5k}^* - \alpha_{5k})K(z, z_k) + b_5 \tag{6.3.5}$$

where α_{5k} and α_{5k}^* denote the coefficients of SVR5, and b_5 is the bias of SVR5.

6.4 Simulations

6.4.1 Foreign Exchange (Forex) Prediction

In our simulations, the proposed approach is applied on 5 different Forex time series: US Dollar exchange rates against EURO (EUR/USD), Japanese Yen exchange rates against US Dollar (USD/JPY), US Dollar exchange rates against Great Britain Pound (GBP/USD), Canada Dollar exchange rates against US Dollar (USD/CAD), Australian Dollar exchange rates against US Dollar (USD/AUD) and their 5 days moving average (MA) data. In order to avoid ignoring some important information, the selected input includes the past prices of some correlative foreign exchange rates, the past values of NASDAQ Composite index and the past prices of Light Sweet Crude Oil as input variables.

In the simulation of predicting the exchange rate of the next day, past 5 days data are used to train the modular SVR network. The input is represented as $x(t) = (f(t), f(t-1), f(t-2), f(t-3), f(t-4), I_1(t), I_1(t-1), I_1(t-2), I_1(t-3), I_1(t-4), I_2(t), I_2(t-1), I_2(t-2), I_2(t-3), I_2(t-4), \ldots)$, where $f(t)$ denotes the price of the Forex time series, and $I_m(t)$ denotes the value of the

Table 6.1: Performance metrics in simulations.

SSE	$SSE = \sum_k (y_k - \hat{y}_k)^2$
MAE	$MAE = \frac{1}{N} \sum_k \lvert y_k - \hat{y}_k \rvert$
CP	$CP = 100 \frac{\sum_k d_k}{\sum_k t_k}$ $d_k = \begin{cases} 1 & \text{if } (\hat{y}_k - \hat{y}_{k-1}) > 0, (y_k - y_{k-1})(\hat{y}_k - \hat{y}_{k-1}) \geq 0 \\ 0 & \text{otherwise} \end{cases}$ $t_k = \begin{cases} 1 & \text{if } (y_k - y_{k-1}) > 0 \\ 0 & \text{otherwise} \end{cases}$
CD	$CD = 100 \frac{\sum_k d_k}{\sum_k t_k}$ $d_k = \begin{cases} 1 & \text{if } (\hat{y}_k - \hat{y}_{k-1}) < 0, (y_k - y_{k-1})(\hat{y}_k - \hat{y}_{k-1}) \geq 0 \\ 0 & \text{otherwise} \end{cases}$ $t_k = \begin{cases} 1 & \text{if } (y_k - y_{k-1}) < 0 \\ 0 & \text{otherwise} \end{cases}$

indicator m, $m = 1, 2, \ldots$. The predicted value $f(t+1)$ is the exchange rate of the next day. Using the same method, we also predict $f(t+5)$ and $f(t+10)$, i.e., next week prediction and next two weeks prediction.

6.4.2 Evaluation of the Model

The prediction performances of the proposed model is evaluated against 4 widely used statistical metrics, namely, Sum of Square Error (SSE), Mean Absolute Error (MAE), Correct Up trend (CP) and Correct Down trend (CD). These criteria are defined in Table 6.1. SSE and MAE measure the deviation between the actual and predicted values. Smaller values of these metrics indicate that the model has a higher precision in the prediction. CP and CD measure the correctness of predicted up and down trend, respectively. In general, it is desirable for a prediction model to attain low SSE and MAE but high CP and CD. Larger values of these metrics indicate that the model has a higher accuracy in direction prediction. In practice, sometimes we may have a model that yields superior SSE and MAE but inferior CP and CD. Some researchers have argued that directional

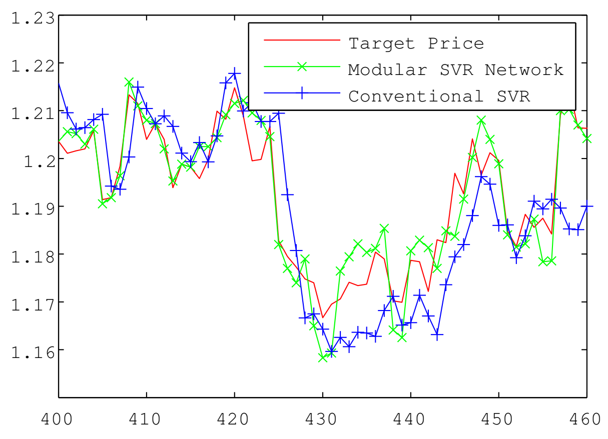


Figure 6.5: Forex prediction result.

change metrics may be a better standard for time series prediction. Therefore, we use these 4 metrics to evaluate the proposed approach.

6.4.3 Simulation Conditions

The data show the exchange rates during January 2000 to December 2007. Ignoring the missing values, we can get 1938 daily data for EUR/USD, GBP/USD, USD/CAD and USD/AUD, and 1865 daily data for USD/JPY. The first 600 daily data are used as the initial training dataset and the remaining daily data are used to validate the model and update the training dataset.

To build SVR models we should select the kernel type and the value of the parameters C and ϵ. Comparing with other kernels in simulations, the model was build using the simple linear kernel defined as $K(x_i, x_j) = \langle x_i \cdot x_j \rangle$ in this study, where $\langle \cdot \rangle$ is the inner product operator. Because over a wide range of value, the parameter c has very little impact on prediction performance for SVR model, we let c equals 10. In time sires prediction, the value of ϵ slightly affect the accuracy and does not cause any drastic degradation in performance, so we keep ϵ fixed at 0.001 as common cases. In the simulations, SVR predictors in the proposed approach are also adapted from the SVM-KMToolbox (in MATLAB) [49].

Table 6.2: Performance metrics of the currency exchange rates.

Future value	Criteria	EUR/USD		USD/JPY		GBP/USD		USD/CAD		USD/AUD	
prediction		SVR	M-SVRN	SVR	M-SVRN	SVR	M-SVRN	SVR	M-SVRN	SVR	M-SVRN
	SSE	3.321	**0.876**	6861	**1224**	6.769	**1.771**	4.365	**0.472**	2.081	**0.493**
next day	MAE	0.044	**0.022**	2.139	**0.871**	0.066	**0.033**	0.057	**0.018**	0.037	**0.017**
prediction	CP	63.4	**67.0**	61.4	**65.0**	63.0	**65.7**	57.2	**58.9**	63.6	**66.5**
(t+1)	CD	56.8	**57.1**	56.0	**58.2**	64.5	**64.8**	64.9	**65.8**	55.9	**57.1**
	SSE	4.268	**1.048**	8026	**1316**	7.742	**1.848**	5.182	**0.527**	2.398	**0.521**
next week	MAE	0.052	**0.025**	2.349	**0.896**	0.074	**0.036**	0.065	**0.019**	0.042	**0.018**
prediction	CP	58.5	**60.4**	56.2	**59.9**	61.5	**63.5**	59.3	**59.6**	60.1	**60.3**
(t+5)	CD	59.7	**60.7**	57.2	**61.4**	62.1	**63.9**	61.0	**61.7**	57.7	**58.0**
	SSE	4.18	**1.010**	8071	**1335**	7.150	**1.711**	5.038	**0.530**	2.279	**0.514**
next 2 weeks	MAE	0.052	**0.025**	2.364	**0.910**	0.073	**0.035**	0.066	**0.020**	0.041	**0.018**
prediction	CP	**56.3**	**56.3**	60.0	**64.7**	60.1	**62.5**	57.6	**59.0**	59.4	**60.4**
(t+10)	CD	62.8	**63.8**	60.6	**60.9**	59.3	**61.4**	62.1	**62.4**	58.9	**60.1**

Lengends: Better results are in bold-type.

Table 6.3: Performance metrics of the moving average data of currency exchange rates.

Future value	Criteria	MA (EUR/USD)		MA (USD/JPY)		MA (GBP/USD)		MA (USD/CAD)		MA (USD/AUD)	
prediction		SVR	M-SVRN	SVR	M-SVRN	SVR	M-SVRN	SVR	M-SVRN	SVR	M-SVRN
	SSE	3.902	**1.171**	8173	**1453**	6.743	**1.852**	5.398	**0.654**	2.244	**0.602**
next day	MAE	0.041	**0.022**	2.905	**1.139**	0.059	**0.029**	0.054	**0.017**	0.031	**0.016**
prediction	CP	59.0	**62.3**	65.5	**72.1**	62.4	**66.8**	61.2	**67.5**	59.1	**62.4**
(t+1)	CD	60.3	**62.9**	64.3	**69.8**	66.4	**70.1**	62.7	**69.0**	60.6	**63.6**
	SSE	4.933	**1.432**	9373	**1609**	7.891	**2.081**	6.121	**0.707**	2.590	**0.683**
next week	MAE	0.048	**0.025**	3.140	**1.204**	0.066	**0.032**	0.059	**0.019**	0.035	**0.017**
prediction	CP	60.1	**63.9**	62.5	**68.6**	64.6	**68.3**	62.4	**67.5**	59.6	**62.9**
(t+5)	CD	60.6	**63.1**	63.6	**71.1**	64.3	**68.5**	64.1	**70.2**	60.1	**64.3**
	SSE	5.008	**1.433**	9485	**1611**	7.888	**2.071**	6.115	**0.704**	2.658	**0.705**
next 2 weeks	MAE	0.048	**0.025**	3.185	**1.217**	0.067	**0.033**	0.060	**0.019**	0.036	**0.017**
prediction	CP	65.4	**70.1**	64.1	**72.0**	65.2	**71.6**	57.7	**66.4**	67.0	**73.5**
(t+10)	CD	62.1	**64.0**	61.8	**70.2**	61.4	**65.7**	63.6	**75.2**	63.3	**70.9**

Lengends: Better results are in bold-type.

6.4.4 Results and Discussions

Table 6.2 shows the results of performance metrics for the next day prediction, next week prediction and next 2 weeks prediction on the 5 different Forex rates, in which M-SVRN is the abbreviation for the modular SVR network. A comparison of results with conventional SVR model illustrates that the proposed modular SVR network can predict Forex rates better, both in terms of accuracy (SSE and MAE) and trend prediction (CP and CD). The average values of CP obtained by M-SVRN on currency exchange rates (EUR/USD, USD/JPY, GBP/USD, USD/CAD and USD/AUD) are 64.6%, 60.7% and 60.6%, respectively, for the next day, next week and next 2 weeks prediction. Compared to classical SVR, we obtain an increase of 2.9%, 1.6% and 1.3% of average values of CP. The average values of CD obtained by M-SVRN on currency exchange rates are 60.6%, 61.14% and 61.7%, respectively, for the next day, next week and next 2 weeks prediction. Compared to classical SVR, we obtain an increase of 1.0%, 1.6% and 1.0% of average values of CD. As discussed above, the modular SVR network model produces lower SSE and MAE and higher CP and CD than those of conventional SVR based model.

As shown in Table 6.2, although the proposed approach can obtain better performances, dithering existing in the daily Forex time series still makes the model difficult to do the trend prediction. Thus, Moving Average (MA) data of Forex time series are also used to test the proposed approach. MA data is smoother than the actual daily data, so it is easier to predict its trend. As shown in Table 6.3, the average values of CP obtained by M-SVRN on MA data of currency exchange rates (EUR/USD, USD/JPY, GBP/USD, USD/CAD and USD/AUD) are 66.2%, 66.3% and 70.7%, respectively, for the next day, next week and next 2 weeks prediction. Compared to classical SVR, we obtain an increase of 4.8%, 4.5% and 6.8% of average values of CP. The average values of CD obtained by M-SVRN on currency exchange rates are 67.1%, 67.5% and 69.2%, respectively, for the next day, next week and next 2 weeks prediction. Compared to classical SVR, we obtain an increase of 4.3%, 5.0% and 6.8% of average values of CD. Obviously, the proposed approach obtains much better performances of trend prediction on the MA data of Forex rates.

Figure 6.5 shows the actual and predicted time series by the modular SVR network and the conventional SVR. For example, in the prediction of US Dollar exchange rate against EURO (EUR/USD) between day 400$\sim$460 which has sudden sharp falls and steep rises, the proposed modular SVR network model could closely follow the actual rate whereas conventional SVR suffered from gross deviation. Compared with conventional SVR, the ability of the modular SVR network to closely follow the actual rate in such a situation will also make a significant difference in the outcome of financial gain in the trading system.

6.5 Conclusions

In time series prediction, it is important to build an efficient regression model for reflecting the potential relationship between the input variables and the output. In this chapter, we introduced a regression model – the modular SVR network, for solving this problem in the price domain. Differing from the existing methods, we use the price domain partition of data as the preprocess of training. The partition is implemented based on the support and resistance levels calculated by Fibonacci Retracement. Several SVRs are built to describe the characters of data and the relationship between input variables and the sub-targets in price subregions. The outputs of these SVRs are used as the input of the prediction layer. Using another SVR as a combinator, we can obtain the final result. Because the whole price domain is partitioned into some subregions and separately dealt with, the problem caused by the unstable statistical properties of Forex time series in price domain can be solved to a certain extent. The proposed approach was experimented on 5 different Forex time series. Experimental results show that our method performs better than the conventional SVR model.

Chapter 7

Conclusions

7.1 Summary

Pattern recognition techniques have evolved rapidly over the past several years. They have been used in many real-world applications. With the development of the data mining, image processing, signal processing and some other techniques in different real-world problems, pattern recognition techniques have to face some new challenges.

This work is motivated by recent trends which indicate that workloads on pattern recognition systems have changed dramatically over the past few years, with a greater emphasis on applications such as gene databases, media processing, communications, and so on.

Size of databases, dimension of samples, and kind of noises are all increased observably in the real-world applications. In addition, imbalance of data is also a challenge. As an algorithm with outstanding performances and constructed based on statistical theories, SVMs are widely used in real-world applications. However, it is still hard to construct an effective and efficient pattern recognition system for dealing with these real-world problems.

In this thesis, some improved algorithms and architectures of SVMs are proposed to solve these problems. Multi-SVM systems are constructed based on these improved single SVMs for obtaining better recognition performances. The following are the main conclusions of this thesis:

- A fast SVM training method based on the separation boundary detection technique is proposed for solving the challenge of large training data.
- A feature selection based on correlation-based feature clustering and SVM-based feature ranking is proposed for solving the challenge of high dimensional input.
- A classification model using a fuzzy decision-making separation boundary is proposed for

solving the challenges of noises and interaction.

- A weighted harmonic mean offset is proposed for solving the challenges of data imbalance.
- A multiple SVM classifier system based on piecewise partition and interpolation is proposed for gaining better performances in classification problems.
- A modular SVR predictor system based on the price domain partition is proposed for solving time series prediction problems.

7.2 Topics for Future Research

Although a lot of progress has been made, there are still many aspects that need further investigations.

- The separation boundary detection technique is quite crucial in fast SVM training. More research can be done in developing novel boundary detection methods in an optimal way.
- The identification of the number of clusters is also important in the proposed feature selection approach. The identification method might be improved in future research.
- In this thesis, we use correlation coefficient and information gain as the criterion of correlation analysis. The choice of the criterion or some more effective standards might be studied.
- Although fuzzy decision-making is a flexible separation boundary and has proven to be able to obtain better performance, an automatically adjustive model is expected. Therefore, the parameters and offset might be adjusted automatically.
- The use of more prior knowledge, in partition of training data, may give a better performance in multiple SVM pattern recognition systems.

Appendix A

A Review of Pattern Recognition Systems

In this appendix, we will give a brief review of pattern recognition systems and some useful definitions in these systems.

A.1 Patterns

Patterns are the means by which we interpret the world. We are performing the task of pattern recognition at every instant of our lives. The ease with which humans classify and describe patterns often leads to the incorrect assumption that this capability is easy to automate [2].

Pattern recognition is naturally based on patterns. A pattern can be as basic as a set of measurements or observations, perhaps represented in a vector or matrix notation. The use of measurements already presupposes some preprocessing and instrumentation system complexity. These measurements could be entities such as blood rate, temperature, volume, age, and the like. Moreover, a pattern can be converted from one representation to another.

A.2 Features

In general, features are any extractable measurement used. Examples of low-level features are signal frequency, sample weight. Features can be symbolic (e.g. color), numerical (e.g. length) or a combination of both. Applying feature extraction to the input data can result in features as well. Additionally, features may be on a higher level of entities: for example, geometric descriptors, such as aspect ratio of an ellipse. Features may be represented by binary variables (e.g. yes/no), discrete (limited values) or continuous variables.

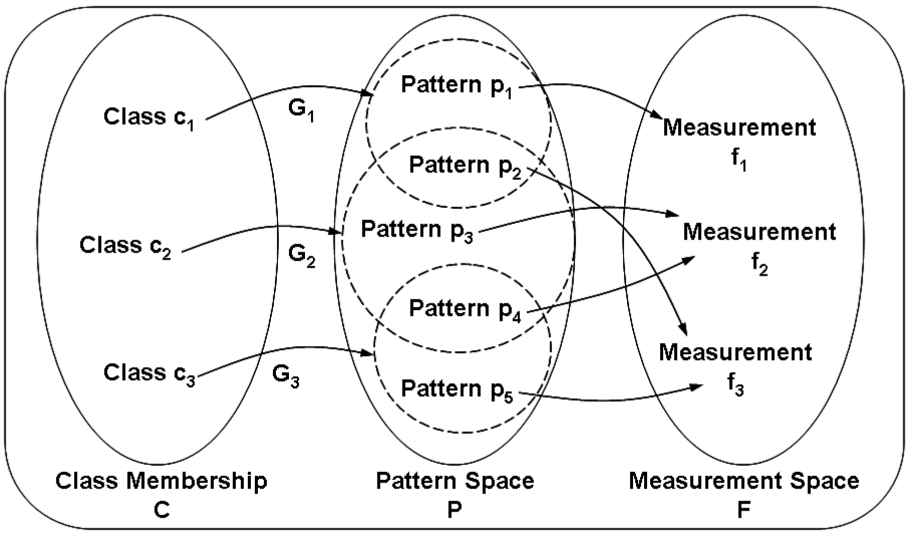


Figure A.1: Mapping in an abstract representation of pattern classification systems.

A.3 Feature Selection

Feature selection is one of the major tasks in any automatic pattern recognition system. The main objective of feature selection is to retain the optimum salient characteristic necessary for the recognition process and to reduce the dimensionality of the measurement space, so that effective and easily computable algorithms can be devised for efficient classification. It is important that the extracted features are relevant to the particular task at hand. Also note that errors or noise can be introduced when extracting features.

The problem of feature selection has two aspects: the selection of an optimal subset from the available features and the formulation of a suitable criterion to evaluate the goodness of a set of features. In some cases there are mathematical tools that help in feature selection. In other cases, simulation may aid in the choice of appropriate features.

A.4 Pattern Recognition

Basically, pattern recognition is about information mapping, information labeling or information reduction [2]. An abstract view of the pattern recognition classification problem is depicted in Figure A.1.

The class-membership space C denotes an abstract domain for classes c_1, c_2, c_3. Via a relation

$G_i\{i = 1, 2, 3\}$, each class c_i is mapped to a subset of patterns in pattern space P, where the i-th pattern is denoted by p_i. These subspaces may overlap, allowing patterns from different classes to share characteristics. Another relation, F, maps patterns from subspaces of P into features or measured patterns, denoted by f_i. Using this concept, the characterization of pattern recognition problems is simply that, given measurement f_i, we want to invert the mappings F and G for all i.

In practice, however, these mappings are rarely invertible, more often not 1:1 *(one-to-one)* functions. As illustrated in Figure A.1, identical measurements may result from different subsets of pattern p_i, which in turn correspond to different underlying classes. It should also be noted that patterns that are generated by the same class (e.g. p_1 and p_2, from c_1) although close in pattern space P, do not necessarily yield close measurements (f_1 and f_3 in this case). This is significant when a clustering of features is used to measure pattern similarity.

A.5 Correct Classification Rate

There are many statistical methods for evaluating and estimating the performance of a classifier. Some of the best known methods are leave-one-out (loo), N-fold cross validation and bootstrapping methods [3, 6]. The goal should be to obtain as honest as possible an estimation about the classification accuracy of the system. The correct classification rate, or sometimes called accuracy, shows the proportion of correct classifications to the total number of classification tests. However this test depends on the assessment setup.

It is always expected that a classifier can reach a performance as high as possible, but overfitting should be avoided. This problem might occur as the classifier fits too much to the training data, but cannot generalize well to new data. A common setup to fairly measure classification performance is by dividing the total number of data into a training set, a validation set and a test set. The classifier is trained with the training set, while the validation set is used to decide when to stop training (i.e. when a minimal error on this validation set is reached). The test data are only used after the training by means of which an independent classification performance can be measured. Figure A.2 depicts this classification setup.

Unfortunately in real-world problems, it is sometimes very difficult to have a large dataset. With the limited available number of data, a trade-off should be taken to divide the dataset into a training set and a test set, e.g. 2/3 of the dataset are used as training data while the rest is used later as test data.

Sometimes, leave-one-out cross validation also can be used as a method to estimate the classifier

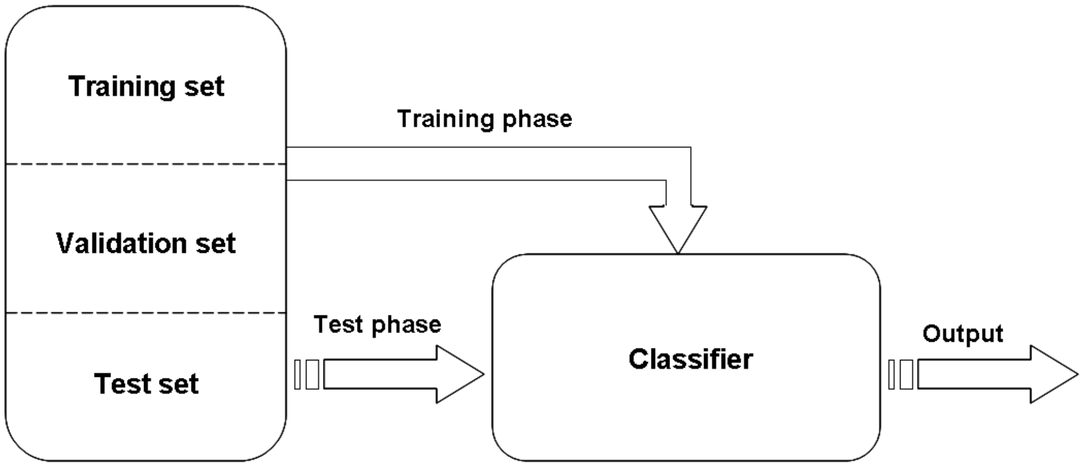


Figure A.2: The dataset is divided into a training, validation and test set. The validation set is used in the training process to obtain a good generalization, e.g. to avoid overfitting. The test set is used to measure the performance of the classifier on the independent/unseen data.

performance in unbiased manner. Here, each step one data point is left out and the classifier is trained using the rest and then the classifier is applied to the left out data point. This procedure is repeated such that each data point is left out once. Classification performance is calculated by taking the average of the number of correct classifications.

A.6 Conclusions

This appendix introduced the terminologies in a typical pattern recognition system. The flow of information from the observed object, features are extracted and results in recognition. The significance of classification performance measures are briefly explained. Some basic terms commonly used in pattern recognition are mentioned as they are used in this thesis.

Appendix B

Linear Classification

In this appendix, we will discuss the binary classification problem which is most common in pattern recognition, explain the basic idea of linear classification and introduces some approaches for linear cases.

B.1 Binary Classification Problems

A simple classification task is to categorize a set of objects, classifying them into two groups on the basis of whether they have some properties or not [15]. This is also called binary classification. Some typical examples are:

- medical testing to determine if a patient has a certain disease or not (the classification property is the disease)
- deciding whether an email is a spam or not (the classification property is having (un-)wanted content, in the simplest case)
- quality control in factories; i.e. deciding whether a new product is good enough to be sold, or if it should be discarded (the classification property is passing certain criteria)

Suppose we are given a task of classifying a group of 100 people into two classes: healthy people and unhealthy people. Surely this task is not straightforward as healthiness can be defined in different ways. A very simple approach will be taken for this example [15].

A first common measurement is the body temperature. Normal body temperature is considered to be 37 degrees Celsius [101]. Body temperature can vary a half degree Celsius above or below 37 °C. and still be considered “normal”.

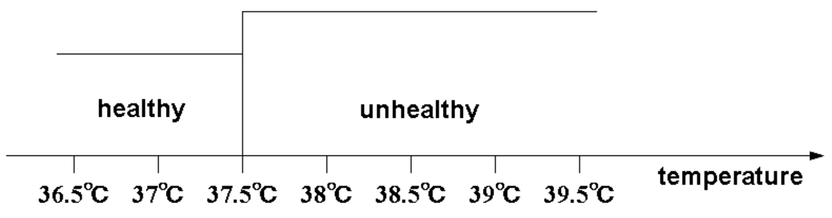


Figure B.1: Example of separating healthy and unhealthy persons based on the body temperature.

If we take as a criterion for an unhealthy person as having a higher body temperature (to have fever), and assume that starting from 37.5 degrees Celsius a human is considered as unhealthy, then a line can be drawn to roughly separate these two classes as shown in Figure B.1.

However, body temperature varies with many factors including level of activity. For example, food, extra clothing, excitement, and anxiety can all raise the body temperature. A womans menstrual cycle can also elevate temperature by one degree or more. Therefore rather than a crisp border, a more natural way to describe the problem is by drawing a distribution. Using the measurement on the population a mean temperature of 37 degree for healthy persons is taken, with half degree of variation, while for unhealthy persons a mean of 38.5 degree Celsius is taken with 1 degree of variation.

A histogram as shown in Figure B.2 gives a better view of the classification, but it is still very difficult to correctly classify a particular person, whose body temperature is located in the range covered by both histograms.

Additional measurements can be added to improve this. Although there are many possible signs of illness, e.g. pale face, red eyes, they should be quantifiable, such as body weight changes, blood pressure or a measure of concentration time. One can take the blood pressure as additional measurement, and assume a person is healthy for blood pressure up to 120 mmHg, otherwise he/she is considered abnormal (blood pressure above 120 mmHg). Now, a pair of values (or a vector) $[body\_temp \;\; blood\_press]^T$ can be used to represent each person as shown in Figure B.3. The additional element of the vector permits us to draw the classification problem in two-dimensions (2D). Extending the dimensionality adds a new degree of freedom. Separation can be done not only by a single threshold point, but by a line. In higher dimensions, it will be called separating hyperplane.

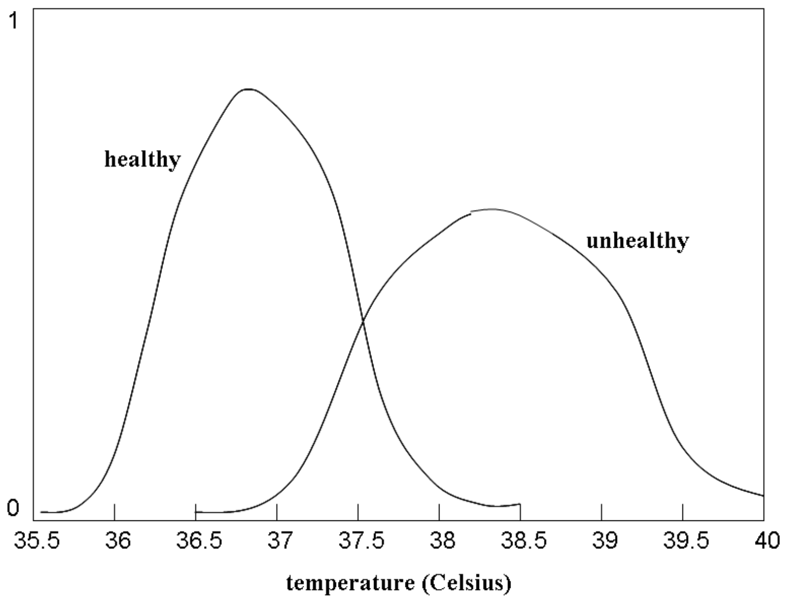


Figure B.2: Example of separating healthy and unhealthy persons based on the body temperature distribution.

B.2 Linear Classification

The simplest form of discriminant function is linear. Linear discriminant function $f(x)$ can be written as

$$f(x) = w^T x + b \tag{B.2.1}$$

where $w = [w_1, w_2, \dots w_l]^T$ is known as the *weight vector* and b as the *bias*. If x_1, x_2 are two points on the decision hyperplane, then the following is valid

$$0 = w^T x_1 + b = w^T x_2 + b \Rightarrow w^T (x_1 - x_2) = 0 \tag{B.2.2}$$

Since the difference vector $x_1 - x_2$ obviously lies on the decision hyperplane for any x_1, x_2, it is clear that the vector w is orthogonal to the decision hyperplane. Figure B.4 illustrates this linear classification.

The distance d and z can be easily calculated by

$$d = \frac{|b|}{\sqrt{w_1^2 + w_2^2}} \qquad z = \frac{|f(x)|}{\sqrt{w_1^2 + w_2^2}} \tag{B.2.3}$$

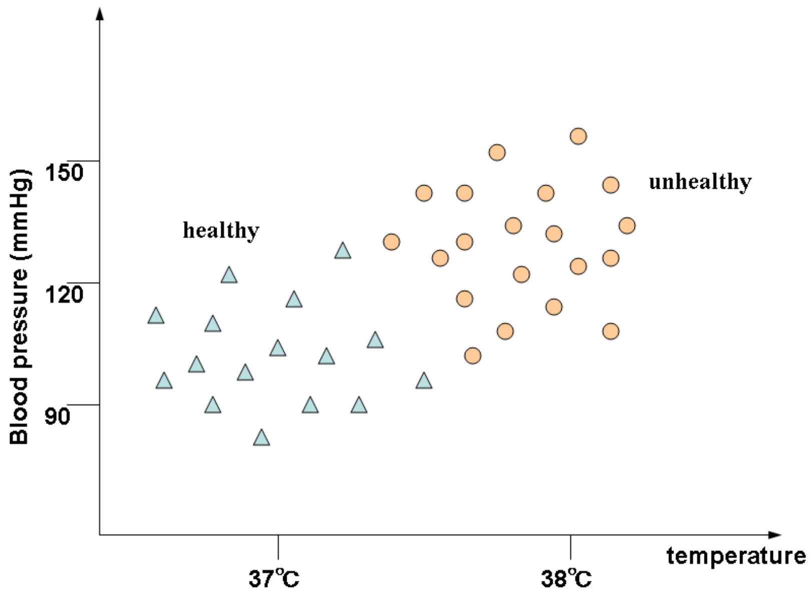


Figure B.3: Binary classification problem.

In other words, $|f(x)|$ is a measure of Euclidean distance of the point x from the decision hyperplane. On one side of the plane $f(x)$ has positive values and on the other negative. In the special case $b = 0$ the hyperplane passes through the origin.

Some criteria commonly used in classification are distance measure, Bayes decision rule, and likelihood. In the following those criteria are explained:

- Distance measure is the simplest and most direct approach to classify data points. Basically, the idea is to classify a data point into the class closest to it. The Euclidean distance and Mahalanobis distance are the two most common definitions. Suppose we have K classes with (μ_i, S_i) as the known parameter set of class i, where μ_i is the reference vector of class i and S_i is the covariance. The Euclidean distance of an observation vector x from class i is

$$d_i(x) = \sqrt{\|x - \mu_i\|^2} \tag{B.2.4}$$

while Mahalanobis distance of vector x can be written as

$$d_i(x) = \sqrt{(x - \mu_i)^T S_i^{-1} (x - \mu_i)} \tag{B.2.5}$$

The Euclidean distance is in fact a special case of the Mahalanobis distance.

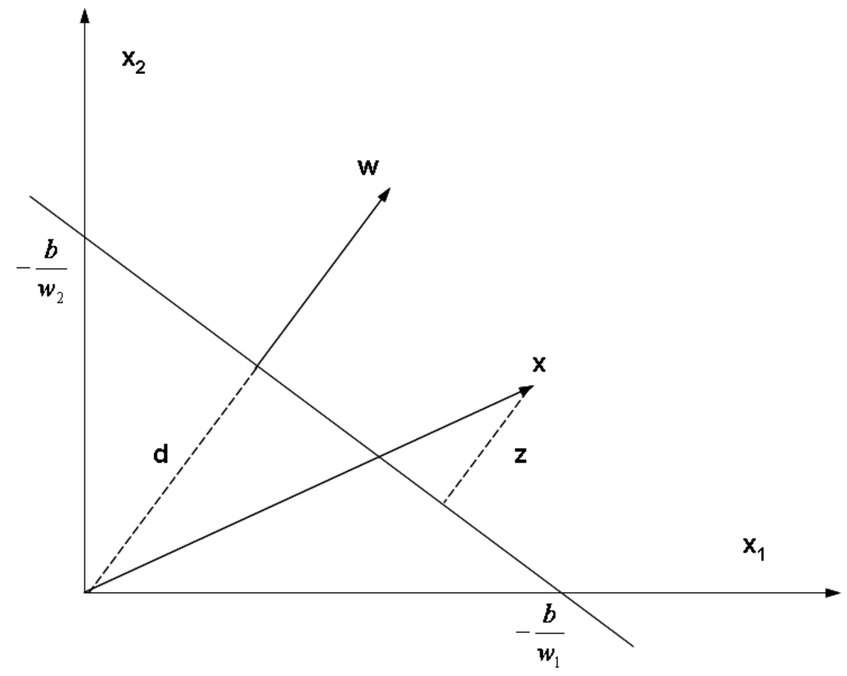


Figure B.4: Linear classification. $f(x) > 0$ above the line, and $f(x) < 0$ below the line.

- Bayes decision rule is based on the assumption that classification problems can be characterized in probabilistic terms and all of the relevant probabilities are known. An observation vector will be assigned to the class that has the largest posterior probability $p(c_j|x)$. Suppose there are k classes c_1, c_2, ..., c_k, and the known a priori probability of each class is denoted by $P(c_i)$, $i = 1, 2, \ldots, k$ and the conditional probability density by $p(x|c_i)$, $i = 1, 2, \ldots, k$, then the posteriori probability can be calculated with Bayes rule:

$$p(c_j|x) = \frac{p(x|c_j)P(c_j)}{p(x)} = \frac{p(x|c_j)P(c_j)}{\sum_{i=1}^{k} p(x|c_i)P(c_i)} \tag{B.2.6}$$

- The likelihood criterion is a special case of Bayes decision rule. Assuming that all of the a priori probabilities $P(c_i)$ are equal and the distribution of the classes are normal ($x \sim \mathcal{N}(\mu_i, S_i)$, $i = 1, 2, \ldots, k$. Then $p(c_j|x) = p(x|c_j)$ and we have

$$p(x|c_j) = \frac{1}{2\pi|S_i|^{1/2}} \exp^{-\frac{1}{2}(x-\mu_i)^T(x-\mu_i)} \tag{B.2.7}$$

If the parameters of a class are known, the likelihood is in fact the prob- ability density function of the class. The logarithm of the likelihood is usually taken to simplify the calculation.

The log-likelihood is written in the following form:

$$P_i(x) = -\frac{1}{2}\ln|S_i| - \frac{n}{2}\ln 2\pi - \frac{1}{2}(x-\mu_i)^T S_i^{-1}(x-\mu_i) \tag{B.2.8}$$

However, in practical applications the class probability densities as well as the prior class probabilities may be unknown. One approach is by estimating the class probability densities from the training data and use the estimated class probability densities together with the prior class distributions to construct the decision rule.

Another approach is by assuming that the classes are distributed normally. The major reasons for the popular use of this Gaussian distribution are its computational tractability and the fact that it models adequately a large number of cases. The mean and the covariance matrices can then be estimated from the given data set. If both classes are assumed to have also equal covariance matrices, a linear discriminant analysis can be applied for this problem.

B.3 Linear Discriminant Analysis (LDA)

The Linear Discriminant Analysis originally designed by Fisher for taxonomic classification [23]. Suppose L represents the linear transformation that maps the original n-dimensional space onto a m-dimensional feature subspace where normally $m \ll n$. The new feature vectors $z_k \in \mathbb{R}^m$ are defined by $z_k = L^T x, k = 1, \ldots, N$.

LDA searches for those vectors in the underlying space that best discriminate among classes (rather than those that best describe the data). More formally, given a number of independent features relative to which the data is described, LDA creates a linear combination of these which yields the largest mean differences between the desired classes.

For all samples of the classes, we define two measures [24]. The first one is called *within-class* scatter matrix, as given by

$$S_w = \sum_{j=1}^{n_C}\sum_{k=1}^{N_j}(x_k^j - \mu_j)^T \tag{B.3.1}$$

where x_k^j is the kth sample of class j, μ_j is the mean of class j, n_C is the number of classes, and N_j is the number of samples in class j.

The second measure is called *between-class* scatter matrix

$$S_b = \sum_{j=1}^{n_C}(\mu_j - \mu)(\mu_j - \mu)^T \tag{B.3.2}$$

where μ represents the mean of all classes.

The goal is to maximize the between-class measure while minimizing the within-class measure. A common way to do this is by maximizing the ratio $\frac{det(S_b)}{det(S_w)}$. It has been proven that if S_w is a nonsingular matrix then this ratio is maximized when the column vectors of the projection matrix, L, are the eigenvectors of $S_w^{-1}S_b$ [23]. However, it should be noted that there are at most $n_C - 1$ nonzero generalized eigenvectors and, so, an upper bound on m is $n_C - 1$. We also require at least $n + n_C$ samples to guarantee that S_w does not become singular. Problems arise when dealing with high dimensional data. The main difficulty in this case lies in the fact that the within-class scatter matrix is almost always singular, therefore the standard algorithm cannot be used. Another disadvantage is a high computational complexity of solving $S_w^{-1}S_b$ when working with the high dimensional input space. To solve this [25, 26] propose the use of an intermediate space. Principal Component Analysis is performed to reduce the dimensionality of the data, followed by applying LDA to this data in the reduced dimension. This method was shown efficient by empirical testing but had no theoretical background upon which it was proven correct [102].

B.4 Principal Component Analysis (PCA)

Principal Component Analysis is one of the simplest and common statistical methods to reduce high-dimensional data. It consists of a transformation from the original variables into a new set of uncorrelated variables, called principal components (PCs). These new variables are linear combinations of the original ones ranked in decreasing order of importance and the number of PCs is determined by the cumulative variance explained by the first components. If the original variables are highly correlated, then the first few PCs will account for most of the variation and the remaining PCs can be discarded with slight loss of information. The more PCs are used the better the reconstructed data fits the original noisy data.

Basically PCA maps data into an orthogonal space, where the axes of this new coordinate system lie along the direction of maximum variance of the original data. By this transformation, it allows the mapping of vectors $x \in \mathbb{R}^n$ into a lower dimensional vectors $z \in \mathbb{R}^m$ with $m < n$. The covariance matrix can be estimated by

$$\hat{S} = \frac{1}{N-1}\sum_{k=1}^{N}(x_k - \mu)(x_k - \mu)^T \tag{B.4.1}$$

where $\mu = \frac{1}{N}\sum_{k=1}^{N} x_k$ and computes the eigenvalue decomposition

$$\hat{S}u_i = \lambda_i u_i \tag{B.4.2}$$

Taking the m largest eigenvalues and their corresponding eigenvectors, the score or transformed variables can be calculated using

$$z_i = u_i^T(x - \mu), i = 1, \ldots, m \tag{B.4.3}$$

In signal processing, PCA is expected to eliminate a significant number of dimensions by partitioning the feature space into subspaces of signal and noise. This is assuming that most of the variance in the data set will be explained by the signal and small contribution by the noise. In this case, taking some PCs which cover a certain amount of variance, e.g. 75%, will significantly reduce the dimension while keeping most information and filtering out noise.

PCA is a useful method for reducing the dimensionality of spectral data because if often works well when the underlying dimensionality of the input space is small. However, it is only appropriate if the underlying relationship between the variables are linear, and most of the variance in the data corresponds to the signal rather than the noise.

B.5 Conclusions

A common way to categorize data into different classes is by creating a linear classification. However, in certain practical applications the dimensionality of the data in the problem space may be very high and it is computationally very expensive. If the underlying relationship between the variables are linear, PCA can be applied to reduce the dimensionality. Otherwise, for a more complex problem, a nonlinear mapping is needed to achieve a better classification.

Appendix C

Kernel-based Classification

In this appendix, we will focus on the kernel-based classifiers, especially the idea of SVM. And some criteria which commonly used are also mentioned.

C.1 Introduction

This section presents the kernel-based classification. The section introduces the idea of maximal margin classification, optimal separating hyperplane, followed by kernel methods as the basis for the extension towards nonlinear classification as introduced by Vapnik, which is called Support Vector Machines (SVM).

C.2 Maximal Margin Classifiers

Consider the class of hyperplanes $w^T x + b = 0$, $w \in \mathbb{R}^n$, $b \in \mathbb{R}$, corresponding to a decision function

$$f(x) = \text{sign}(w^T x + b) \tag{C.2.1}$$

First, we consider the case of linearly separable data. A hyperplane can separate two classes of data in many possible ways (see Figure C.1 left). There is no unique separating hyperplane, unless we add a criterion to decide which is the best or the *optimal separating hyperplane*.

Basically the idea of learning from examples is to recognize the pattern of a class by examining the training points corresponding to that class. New data points are assumed to lie somewhere around the known training data. Therefore, a hyperplane should be chosen such that a small shift of the data does not result in prediction change. If the distance between the separating hyperplane and

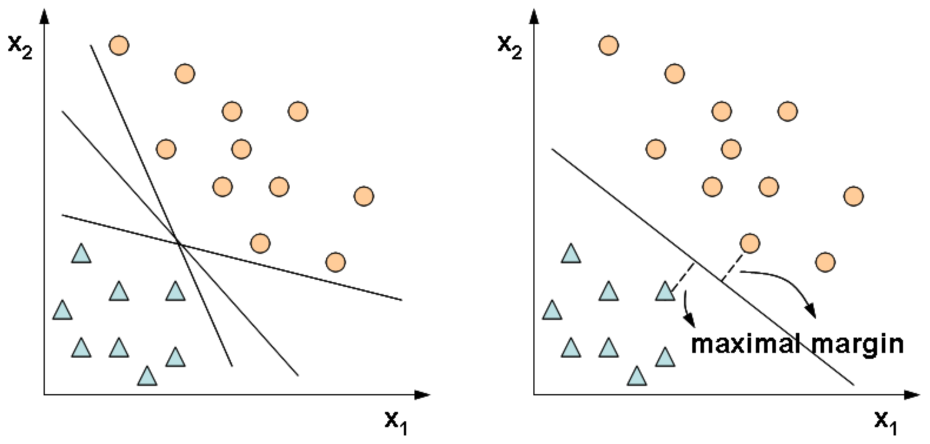


Figure C.1: Linear classification: (left) example of classification problem where the separating hyperplane is not unique, (right) example of a unique hyperplane which corresponds to a maximal margin of the closest points to the separating hyperplane.

the training points becomes too small, even test examples very close to the training samples may be classified incorrectly.

Based on this idea, Vapnik and Chervonenkis presumed that the generalization ability depends on the distance between the hyperplane and the training points. They introduced the *Generalized Portrait*, a learning algorithm for separable problems, by constructing a hyperplane which maximally separates the classes (*maximum margin*):

$$\max_{w,b} \min\{\|x - x_k\| : x \in \mathbb{R}^n, w^T x + b = 0, k = 1, \dots, N\} \tag{C.2.2}$$

To show how this hyperplane can be constructed in an efficient way, we need to start with some definitions.

Definition C.2.1. (Separability)
A training set $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N) : x_k \in \mathbb{R}^n, y_k \in \{-1, +1\}\}$ is called separable by a hyperplane $w^T x + b = 0$ if there exist both a unit vector w ($\|w\| = 1$) and a constant b such that the following equalities hold:

$$w^T x_k + b \geq +1 \quad \text{for} \quad y_k = +1 \tag{C.2.3}$$

$$w^T x_k + b \leq -1 \quad \text{for} \quad y_k = -1 \tag{C.2.4}$$

The hyperplane defined by w and b is called a *separating hyperplane*.

Definition C.2.2. (Margin)
Consider separating hyperplane H as defined by $w^T x + b = 0$.

- the margin $\zeta_k(w, b)$ of a training point x_k is defined as the distance between H and x_k:

$$\zeta_k(w, b) = y_k(w^T x + b) \tag{C.2.5}$$

- the margin $\zeta_{\mathcal{D}}(w, b)$ of a set of vectors $\mathcal{D} = x_1, \ldots, x_k$ is defined as the minimum distance from H to the vectors in $\mathcal{D}$:

$$\zeta_{\mathcal{D}}(w, b) = \min_{x_k \in \mathcal{D}} \zeta_k(w, b) \tag{C.2.6}$$

Now consider the unit vector w^* and the constant b^* which maximize the margin of the training set $\zeta(w, b)$ and also satisfy the condition (C.2.3) and (C.2.4). This pair of (w^*, b^*) defines the hyperplane which separates the positive from the negative examples with the largest margin. This hyperplane is also called *maximal margin hyperplane* or *optimal separating hyperplane*.

Definition C.2.3. (Optimal separating hyperplane)
The Optimal hyperplane of a training set $\mathcal{D}$ is defined by

$$(w^*, b^*) = \arg\max_{w,b} \zeta_{\mathcal{D}(w,b)} \tag{C.2.7}$$

Vapnik proves on the uniqueness of the optimal separating hyperplane in [7]. With a unique w^* we can describe b^* by

$$b^* = \frac{1}{2}(\min_{k \in I_+} w^* x_k - \max_{k \in I_-} w^* x_k) \tag{C.2.8}$$

where $I_+ = \{k | y_k = +1\}$ and $I_- = \{k | y_k = -1\}$.

C.3 Construction of the Optimal Hyperplane

Given a training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N) : x_k \in \mathbb{R}^n, y_k \in \{-1, +1\}\}$, the optimal separating hyperplane is the solution to the optimization problem

$$\begin{aligned} \max \quad & \zeta_{\mathcal{D}}(w, b) \\ \text{subject to} \quad & \zeta_{\mathcal{D}}(w, b) > 0 \\ & \|w\| = 1 \end{aligned} \tag{C.3.1}$$

However, solving (C.3.1) is not straightforward, it involves nonlinear constraints, while the objective function itself is neither linear nor quadratic. We can rewrite the optimization problem equivalently as

$$\begin{aligned} \min \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & w^T x + b \geq +1 \quad for \;\; y_k = +1 \\ & w^T x + b \leq -1 \quad for \;\; y_k = -1 \end{aligned} \tag{C.3.2}$$

Theorem C.3.1. *(Vapnik) [7] Vector w_0 that solves problem (C.3.2) is related to the vector w^* solving problem (C.3.1) by the equality*

$$w^* = \frac{w_0}{\|w_0\|} \tag{C.3.3}$$

This implies that the construction of the optimal hyperplane first needs to solve problem (C.3.2) with linear constraints, where the constraints can be simplified as

$$y_k(w^T x_k + b) - 1 \geq 0, k = 1, \ldots, N \tag{C.3.4}$$

The Lagrangian for this problem is

$$\mathcal{L}(w, b, e; \alpha) = \frac{1}{2} w^T w - \sum_{k=1}^{N} \alpha_k \{ y_k [w^T x_k + b] - 1 \} \tag{C.3.5}$$

with Lagrange multipliers $\alpha_k \geq 0$ for $k = 1, \ldots, N$. The solution is characterized by the saddle point of the Lagrangian

$$\max_{\alpha} \min_{w,b} \mathcal{L}(w, b; \alpha) \tag{C.3.6}$$

This leads to

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{k=1}^{N} \alpha_k y_k x_k \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \end{cases} \tag{C.3.7}$$

with resulting classifier

$$y(x) = \text{sign}[\sum_{k=1}^{N} \alpha_k y_k x_k^T x + b] \tag{C.3.8}$$

Elimination of w from (C.3.5) with (C.3.7) gives the following Quadratic Programming (QP) problem as the dual problem in the Lagrange multipliers α_k

$$\begin{aligned} \max_{\alpha} \mathcal{J}_D(\alpha) \quad &= \quad -\frac{1}{2} \sum_{k,l=1}^{N} y_k y_l x_k^T x_l \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \\ \text{such that} \quad & \sum_{k=1}^{N} \alpha_k y_k = 0 \end{aligned} \tag{C.3.9}$$

Note that this problem is solved in $\alpha = [\alpha_1 \ldots \alpha_N]$, and not in w. This QP problem has a number of interesting properties [8]:

- *global and unique solution*:
  The matrix related to the quadratic term in α in this quadratic form is positive definite or positive semidefinite. In the case that the matrix is positive definite (all eigenvalues strictly positive) the solution α to this QP problem is global and unique [9]. When the matrix is positive semidefinite (all eigenvalues are positive with some zero eigenvalues) then the solution is global but not necessarily unique. It may happen that the solution of (w, b) is unique but α not. In terms of $w = \sum_k \alpha_k y_k x_k$ this means that there may exist equivalent expansions of w which require fewer support vectors.

- *sparseness*:
  An interesting property is that many of the resulting α_k values are equal to zero. Hence the obtained solution is sparse. This means that in the resulting classifier the sum should be taken only over non-zero α_k values instead of all training data points:

$$y(x) = \text{sign}[\sum_{k=1}^{\#SV} \alpha_k y_k x_k^T x + b] \tag{C.3.10}$$

  where $\#SV$ is the number of support vectors.

- *geometric meaning of support vectors*: The support vectors obtained from the QP problem are located close to the decision boundary.

- *non-parametric/parametric issues*: Note that the size of the solution vector α grows with the number of data points N. It means, the dual problem actually corresponds to a non-parametric approach, while in the primal problem it is parametric because the size of w is fixed independently of the number of data points. As a consequence, for large data sets it might be advantageous to solve the primal problem but in higher dimensional input space it is better to solve the dual problem as the size of the solution vector α does not depend on the dimension n of the input space.

C.4 Optimal Hyperplane for Linearly Nonseparable Case

In the previous subsection the SVM solution of a linearly separable classification problem is explained. However, in most real life problems we deal with noisy data which will render simple linear separation impossible. No feasible solution to the margin maximization problem can be found, due to the fact that the objective function (i.e. the dual Lagrangian) is growing arbitrarily large.

To generalize the previous result with this case, we need to relax the separability constraints, tolerating some misclassifications in the overlapping region. For each violation of the constraints will be punished with a misclassification penalty.

Cortes and Vapnik in 1995 gave the extension of linear SVM to the nonseparable case [27]. Basically it is done by introducing positive slack variable ξ_k in the constraints. The inequalities are transformed to

$$y_k[w^T x_k + b] \geq 1 - \xi_k, k = 1, \ldots, N \tag{C.4.1}$$

We are interested in the smallest slack variable satisfying

$$\xi_k = \max\{0, 1 - y_k[w^T x_k + b]\} \tag{C.4.2}$$

It measures how many points fail to have a margin of $1/\|w\|$. The values of ξ_k indicate where x_k lies compared to the separating hyperplane.

- $\xi_k \geq 1$: $y_k[w^T x_k + b] < 0$, misclassification;
- $0 < \xi_k < 1$: x_k is classified correctly, but lies inside the margin;
- $\xi_k = 0$: x_k is classified correctly, and lies outside the margin or exactly on the margin boundary.

In the primal weight space the optimization problem becomes

$$\begin{aligned} \min_{w,b,\xi} \mathcal{J}_{\mathcal{P}} &= \frac{1}{2} w^T w + c \sum_{k=1}^{N} \xi_k \\ \text{such that} & \quad \begin{cases} y_k[w^T x_k + b] \geq 1 - \xi_k, k = 1, \ldots, N \\ \xi_k \geq 0, k = 1, \ldots, N \end{cases} \end{aligned} \tag{C.4.3}$$

where c is a positive real constant. We should then consider the following Lagrangian

$$\mathcal{L}(w, b, \xi; \alpha, v) = \mathcal{J}_{\mathcal{P}}(w, \xi) - \sum_{k=1}^{N} \alpha_k (y_k[w^T x_k + b] - 1 + \xi_k) - \sum_{k=1}^{N} v_k \xi_k \tag{C.4.4}$$

with Lagrange multipliers $\alpha_k \geq 0, v_k \geq 0$ for $k = 1, \ldots, N$. The second set of Lagrange multipliers is needed due to the additional slack variables ξ_k. The solution is given by the saddle point of the Lagrangian:

$$\max_{a,v} \min_{w,b,\xi} \mathcal{L}(w, b, \xi; \alpha, v) \tag{C.4.5}$$

One obtains

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{k=1}^{N} \alpha_k y_k x_k \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_k} = 0 \rightarrow 0 \leq \alpha_k \leq c, k = 1, \ldots, N \end{cases} \tag{C.4.6}$$

which gives the following dual QP problem after substituting (C.4.6) into (C.4.4):

$$\max_{\alpha} \mathcal{J}_{\mathcal{D}} = -\frac{1}{2} \sum_{k,l=1}^{N} y_k y_l x_k^T x_l \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \tag{C.4.7}$$

$$\text{such that} \quad \begin{cases} \sum_{k=1}^{N} \alpha_k y_k = 0 \\ 0 \leq \alpha_k \leq c, k = 1, \ldots, N \end{cases}$$

In comparison with the linearly separable case (C.4.1) this problem has additional box constraints.

C.5 Kernel-based Classifiers

In the previous sections we have seen how linear classifiers in the input space can easily be solved by standard optimization techniques. Although linear classifiers are easy to handle, they sometimes pose severe restrictions on the learning task. The target concept may be too complex to be expressed as a linear combination of the given attributes. This problem can be overcome by an approach called kernel technique, which dates back to the early 1960s and was introduced as the method of potential functions by Aizerman et al. [28]

Let $x \in \mathcal{D} \subseteq \mathbb{R}^n$ denote a real valued random input vector, and $y \in \{-1, +1\}$ discrete real valued random output variable and let $\Omega \subseteq \mathbb{R}^{n_\mathcal{H}}$ denote a high dimensional feature space. The SVM method basically maps the input vector x into the high dimensional feature space Ω through some nonlinear mapping $\varphi : \mathcal{D} \rightarrow \Omega$. In this feature space, one consider the linear function

$$f(x) = \text{sign}[w^T \varphi(x) + b] \tag{C.5.1}$$

This linear function may generalize well in solving classification problems, however, it remains a problem to solve the calculation in the high dimensional feature space. Interestingly, no explicit construction of the nonlinear mapping $\varphi(x)$ is needed. This is motivated by the following result. The inner product in the feature space $\varphi(x_k)^T \varphi(x_l)$ can be replaced with the corresponding kernel $K(x_k, x_l)$ satisfying Mercer's condition.

Theorem C.5.1. *(Mercer) [29] Let $K \in L^2(C)$,$g \in L^2(C)$ where C is a compact subset of $\mathbb{R}^d$. To guarantee that a continuous symmetric function K has an expansion*

$$K(t, z) = \sum_{k=1}^{\infty} a_k \phi(t) \phi(z) \tag{C.5.2}$$

with positive coefficients $a_k > 0$, it is necessary and sufficient that the condition

$$\int_C \int_C K(t,z)g(t)g(z)dtdz \geq 0 \tag{C.5.3}$$

be valid for all $g \in L^2(C)$.

Suppose $w = \sum_{k=1}^N \alpha_k \varphi(x_k)$, and based on the Mercer theorem above, the linear function in the feature space (C.5.1) has the following equivalent representation in input space

$$f(x) = \text{sign}[\sum_{k=1}^N \alpha_k K(x, x_k) + b], b \in \mathbb{R}, \alpha_k \in \mathbb{R} \tag{C.5.4}$$

where x_k are vectors and $K(x, x_k)$ is a given function satisfying Mercer's condition. While in the primal space the parameter w may have a range over an "infinite dimensional" parameter set, in the dual problem the dimension of the parameter vector α grows with the number of data points.

Using Mercer's theorem to replace the inner product $\varphi(x_k)^T\varphi(x_l)$ with its corresponding kernel $K(x_k, x_l)$ is often called the kernel trick. It enables us to work in a huge dimensional feature space without actually having to do explicit computations in this space. Computations are done in another space after applying this kernel trick. In the case of support vector machines, one starts from a formulation in the primal weight space with a high dimensional feature space by applying transformations $\varphi(\cdot)$. The solution is calculated not in this primal weight space, but in the dual space of Lagrange multipliers after applying the kernel trick. In this way classification is done implicitly in a high dimensional feature space rather than in the original input space.

C.6 Nonlinear SVM Classifiers

By combining the idea of an optimal separating hyperplane with a kernel induced mapping to a high dimensional feature space, we extend the idea from linear to nonlinear classifiers. One can formally replace x by $\varphi(x)$ and apply the kernel trick where possible. However, notice that $\varphi(x)$ can be infinite dimensional, and hence also the w vector. While for linear SVM one can in fact equally well solve the primal problem in w as the dual problem in the support values α, this is no longer the same for the nonlinear SVM case because in the primal problem the unknown w can be infinite dimensional.

With slight modification, for the nonlinear case we can write

$$w^T\varphi(x_k) + b \geq +1 \quad \text{for} \quad y_k = +1 \tag{C.6.1}$$

$$w^T\varphi(x_k) + b \leq -1 \quad \text{for} \quad y_k = -1 \tag{C.6.2}$$

which is equivalent to

$$y_k[w^T\varphi(x_k)+b] \geq 1, k=1,\ldots,N \tag{C.6.3}$$

in the case of separable data. The classification can be written as

$$y_k = \text{sign}[w^T\varphi(x)+b] \tag{C.6.4}$$

The optimization problem becomes

$$\begin{aligned} \min_{w,b,\xi} \mathcal{J}_P &= \frac{1}{2}w^Tw + c\sum_{k=1}^{N}\xi_k \\ \text{such that} & \quad \begin{cases} y_k[w^T\varphi(x_k)+b] \geq 1-\xi_k, k=1,\ldots,N \\ \xi_k \geq 0, k=1,\ldots,N \end{cases} \end{aligned} \tag{C.6.5}$$

The Lagrangian is constructed:

$$\mathcal{L}(w,b,\xi;\alpha,v) = \mathcal{J}_P(w,\xi) - \sum_{k=1}^{N}(\alpha_k y_k[w^T\varphi(x_k)+b]-1+\xi_k) - \sum_{k=1}^{N} v_k\xi_k \tag{C.6.6}$$

with Lagrange multipliers $\alpha_k \geq 0$, $v_k \geq 0$ for $k=1,\ldots,N$. The solution is given by the saddle point of the Lagrangian:

$$\max_{a,v}\min_{w,b,\xi} \mathcal{L}(w,b,\xi;\alpha,v) \tag{C.6.7}$$

This leads to

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{k=1}^{N}\alpha_k y_k \varphi(x_k) \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{k=1}^{N}\alpha_k y_k = 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_k} = 0 \rightarrow 0 \prec \alpha_k \prec c, k=1,\ldots,N \end{cases} \tag{C.6.8}$$

The dual problem becomes

$$\begin{aligned} \max_{\alpha} \mathcal{J}_D(\alpha) &= -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l K(x_k,x_l)\alpha_k\alpha_l + \sum_{k=1}^{N}\alpha_k \\ \text{such that} & \quad \begin{cases} \sum_{k=1}^{N}\alpha_k y_k = 0 \\ 0 \leq \alpha_k \leq c, k=1,\ldots,N \end{cases} \end{aligned} \tag{C.6.9}$$

In this quadratic form, the kernel trick is applied

$$K(x_k,x_l) = \varphi(x_k)^T\varphi(x_l) \tag{C.6.10}$$

for $k=1,\ldots,N$. Finally the nonlinear SVM classifier takes the form

$$y(x) = \text{sign}[\sum_{k=1}^{N}\alpha_k y_k K(x,x_k)+b] \tag{C.6.11}$$

with α_k positive real constants which are the solution to a QP problem. The next problem is the determination of the constant b. Karush-Kuhn-Tucker (KKT) complementarity conditions state that the product of the dual variables and the constraints should be zero at the optimal solution [9]. Therefore, using the KKT conditions, it yields

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{k=1}^{N} \alpha_k y_k \varphi(x_k) \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_k} = 0 \rightarrow c - \alpha_k - v_k = 0 \\ \alpha_k\{y_k[w^T\varphi(x_k) - 1 + \xi_k]\} = 0, k = 1, \ldots, N \\ v_k \xi_k = 0, k = 1, \ldots, N \\ \alpha_k \geq 0, k = 1, \ldots, N \\ v_k \geq 0, k = 1, \ldots, N \end{cases} \tag{C.6.12}$$

From $v_k\xi_k = 0$ we have for the solutions w^*, b^*, ξ^*, α^*, v^* to this problem that $\xi_k^* = 0$ for $\alpha_k^* \in (0, c)$. Hence

$$y_k[w^T\varphi(x_k) + b] - 1 = 0 \quad \text{for } \alpha_k \in (0, c) \tag{C.6.13}$$

which means that one can take any training data point for which $0 \leq \alpha_k \leq c$ and use that equation to compute the bias term b.

Some properties of the nonlinear SVM classifier and its solution are the following [8]:

- *choice of kernel function*:
  Four common choices of kernels are

  1. $K(x, z) = x^T z$ (linear SVM))
  2. $K(x, z) = (\tau + x^T z)^d$ (polynomial SVM of degree d)
  3. $K(x, z) = \exp(-\|x - z\|_2^2/\sigma^2)$ (RBF kernel)
  4. $K(x, z) = \tanh(\kappa_1 x^T z + \kappa_2)$ (Multilayer Perceptron/MLP)

  The Mercer condition holds for all σ values in the RBF kernel case and τ values in the polynomial case but not for all possible choices of κ_1, κ_2 in the MLP case.

- *global and unique solution*:
  As in the linear SVM case the solution to the convex QP problem is again global and unique provided that one chooses a positive definite kernel for $K(\cdot, \cdot)$. This choice guarantees that the matrix involved in the QP problem is positive definite as well, and the kernel trick is applicable. For a positive semidefinite kernel the solution to the QP problem is global but not necessarily unique.

- *sparseness*:
  As for the linear SVM classifier case many α_k values are equal to zero in the solution vector. In the dual space the nonlinear SVM classifier takes the form

$$y(x) = \text{sign}[\sum_{k=1}^{\#SV} \alpha_k y_k K(x, x_k) + b] \tag{C.6.14}$$

  where the sum is taken over the non-zero α_k values which correspond to support vectors x_k of the training data set.

- *geometric meaning of support vectors*:
  The support vectors obtained from the QP problem are located close to the nonlinear decision boundary.

- *non-parametric/parametric issues*:
  Both the primal and the dual problem have neural network representation interpretations. The problem in the primal weight space is parametric, while the dual is non-parametric. Note that in the dual problem the size of the QP problem is not influenced by the dimension n of the input space.

C.7 Conclusions

Support Vector Machines (SVM) is a method of calculating the optimal separating hyperplane in the feature space. Optimal separating hyperplane is defined as the maximum-margin hyperplane in the higher dimensional feature space.

The use of the maximum-margin hyperplane is motivated by statistical learning theory, which provides a probabilistic test error bound which is minimized when the margin is maximized.

The parameters of the maximum-margin hyperplane are derived by solving a quadratic programming (QP) optimization problem. There exists several specialized algorithms for quickly solving the QP problem that arises from SVMs.

The original SVM was a linear classifier. However, Vapnik suggested using the kernel trick (originally proposed by Aizerman). In the kernel trick, each dot product used in a linear algorithm is replaced with a non-linear kernel function. This causes the linear algorithm to operate in a different space. For SVMs, using the kernel trick makes the maximum margin hyperplane be fit in a feature space. The feature space is a non-linear map from the original input space, usually of much higher dimensionality than the original input space. In this way, non-linear SVMs can be created. If the kernel used is a radial basis function, the corresponding feature space is a Hilbert space of infinite dimension.

Publication List

1. Boyang Li, Qiangwei Wang and Jinglu Hu, "Feature Subset Selection: A Correlation-based SVM Filter Approach", in *IEEJ Transactions on Electrical and Electronic Engineering (TEEE), Vol.6, No.2*, in press, 2010.

2. Boyang Li, Jinglu Hu and Kotaro Hirasawa, "Support Vector Machine Classifier with WHM Offset for Unbalanced Data", in *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol.12, No.1*, pp. 94-101, 2008.

3. Qiangwei Wang, Boyang Li and Jinglu Hu, "Human Resource Selection based on Performance Classification using Weighted Support Vector Machine", in *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol.13, No.4*, pp. 407-415, 2009.

4. Boyang Li, Qiangwei Wang and Jinglu Hu, "A Fast SVM Training Method for Very Large Datasets", in *Proc. of IEEE International Joint Conference on Neural Networks 2009 (IJCNN 2009)*, pp. 1784-1789, Atlanta, Georgia, USA, 2009.

5. Qiangwei Wang, Boyang Li and Jinglu Hu, "Feature Selection for Human Resource Selection Based on Affinity Propagation and SVM Sensitivity", in *Proc. of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009)*, pp. 31-36, Coimbatore, India, 2009.

6. Boyang Li, Jinglu Hu and Kotaro Hirasawa, "Financial Time Series Prediction Using A Support Vector Regression Network", in *Proc. of IEEE World Congress on Computational Intelligence 2008 (WCCI 2008) – International Joint Conference on Neural Networks 2008 (IJCNN 2008)*, pp. 621-627, Hong Kong, China, 2008.

7. Boyang Li, Jinglu Hu and Kotaro Hirasawa, "An Improved Support Vector Machine with Soft

Decision-Making Boundary", in *Proc. of IASTED International Conference on Artificial Intelligence and Applications (AIA2008)*, pp. 40-45, Innsbruck, Austria, 2008.

8. Boyang Li, Liangpeng Ma, Jinglu Hu and Kotaro Hirasawa, "Gene classification using an improved SVM classifier with soft decision boundary", in *Proc. of SICE Annual Conference 2008*, pp. 2476-2480, Tokyo, Japan, 2008.

9. Qiangwei Wang, Boyang Li and Jinglu Hu, "Human Resource Selection based on Performance Classification using Weighted Support Vector Machine", in *Proc. of Joint 4th International Conference on Soft Computing and Intelligent Systems and. 9th International Symposium on advanced Intelligent Systems (SCIS&ISIS 2008)*, pp.1837-1842, Nagoya, Japan, 2008.

10. Boyang Li, Jinglu Hu and Kotaro Hirasawa, "Fuzzy Decision-making SVM with An Offset for Real-world Lopsided Data Classification", in *Proc. of SICE-ICASE International Joint Conference 2006*, pp. 143-148, Pusan, Korea, 2006. (Young Author Award)

11. Boyang Li, Jinglu Hu, Kotaro Hirasawa, Pu Sun and Kenneth Marko, "Support Vector Machine with Fuzzy Decision-Making for Real-world Data Classification", in *Proc. of IEEE World Congress on Computational Intelligence 2006 (WCCI 2006) – International Joint Conference on Neural Networks 2006 (IJCNN 2006)*, pp. 1314-1319, Vancouver, Canada, 2006.

Bibliography

[1] V. Vapnik. The Nature of Statistical Learning Theory. Springer, New York (1995).

[2] R. Schalkoff. Pattern Recognition: statistical, structural and neural approaches. John Wiley & Sons, Inc. (1992).

[3] R. O. Duda, P. E. Hart and D. G. Stork. Pattern Classification. 2nd ed. John Wiley & Sons, (2001).

[4] B. D. Ripley. Pattern Recognition and Neural Networks. Cambridge University Press (1996).

[5] J. Tou and R. Gonzalez. Pattern Recognition Principles. Addison Wesley, Reading Mass. (1979).

[6] C. M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press (1995).

[7] V. Vapnik. Statistical Learning Theory. John Wiley & Sons (1998).

[8] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor and J. Vandewalle. Least Squares Support Vector Machines. World Scientific Publishing Co. (2002).

[9] R. Fletcher. Practical methods of optimization. Chichester and New York: John Wiley and Sons (1987).

[10] T. H. Cormen, C. E. Leiserson and R. L. Rivest. Introduction to Algorithms. MIT Press (2000).

[11] N. F. Johnsona, D. Lampera, P. Jeeriesa, M. L. Harta and S. Howisonb. Application of multi-agent games to the prediction of financial time-series. Elsevier (2001).

[12] H. Li and R. Kozma. A Dynamic Neural Network Method for Time Series Prediction Using the KIII Model (2003).

[13] B. Poulos. The truth about Fibonacci Trading (2004).

[14] D. Pissarenko. Neural Networks For Financial Time Series Prediction: Overview Over Recent Research, (2001-2002).

[15] L. Lukas. Least Squares Support Vector Machines Classification Applied to Brain Tumor Recognition Using Magnetic Resonance Spectroscopy. In *Ph.D. Dissertation*, Katholieke Universiteit Leuven (2003).

[16] F. T. Allen, J. M. kinser and H. J. Gaulfield. A neural bridge from syntactic pattern recognition to statistical pattern recognition. In *Neural Networks* 12, 519-526 (1999).

[17] D. Marr. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. W. H. Freeman. San Francisco, CA: (1982).

[18] J. M. Keller. Fuzzy sets in computer vision. In *Proc. VI IFSA World Congress*, Sap Paulo, Brazil, Vol.I, 7-10, (1995).

[19] KS Ray, J. Ghoshal. Approximate reasoning approach to pattern recognition. In *Fuzzy Sets and Systems*, 77:125-150 (1996).

[20] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. In *Automation and Remote Control*, 24 (1963).

[21] V. Vapnik and A. Chervonenkis. A note on class of perceptron. In *Automation and Remote Control*, 24 (1964).

[22] A. J. Smola and B. Schollköpf. A tutorial on support vector regression. In *NeuroCOLT2 Technical Report Series NC2-TR-1998-030*, ESPRIT working group on Neural and Computational Learning Theory, NeuroCOLT2 (1998).

[23] R. A. Fisher. The use of multiple measurements in taxonomic problems. In *Annals of Eugenics* 7, 179-188 (1936).

[24] A. M. Martinez and A. C. Kak. PCA versus LDA. In *IEEE Trans. Pattern Analysis and Machine Intelligence* 23(2), 228-233 (2001).

[25] P. N. Belhumeur, J. P. Hespanha and D. J. Kriegman. Eigenfaces vs. Fisher-face: recognition using class specific linear projection. In *IEEE Trans. Pattern Analysis and Machine Intelligence* 19(7), 711-720 (1997).

[26] D. L. Swets and J. J. Weng. Using discriminant eigenfeatures for image retrieval. In *IEEE Trans. Pattern Analysis and Machine Intelligence* 18(8), 831-836 (1996).

[27] C. Cortes and V. Vapnik. Support Vector Networks. In *Machine Learning*, 20, 273-297 (1995).

[28] M. A. Aizerman, E. M. Braverman and L. I. Rozonoer. Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning. In *Autom. Remote Control* 25 (1964).

[29] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. In *Philos. Trans. Roy. Soc. London* 209, 415-446 (1909).

[30] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machine. In *Advances in Neural Information Processing Systems*, Vol.13, Cambridge, MA, MIT Press (2001).

[31] A. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference onMachine Learning*, 911-918, Stanford, USA (2000).

[32] D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. In *Advances in Neural Information Processing Systems*, Vol.14, Cambridge, MA, MIT Press (2002).

[33] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. In *Journal of Machine Learning Research*, 243C264 (2001).

[34] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of Computer Vision and Pattern Recognition*, 130-136 (1997).

[35] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods – Support Vector Learning*, 185-208, MIT Press (1999).

[36] T. Joachims. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods – Support Vector Learning*, MIT Press (1999).

[37] I. W. Tsang, J. T. Kwok, and P-M. Cheung. Core Vector Machines: Fast SVM Training on Very Large Data Sets. In *Journal of Machine Learning Research*, 6:363-392 (2005).

[38] W-C. Kao, K-M. Chung, C-L. Sun, and C-J. Lin. Decomposition methods for linear support vector machines. In *Neural Computation*, 1689C1704 (2004).

[39] C. Yang, R. Duraiswami, and L. Davis. Efficient kernel machines using the improved fast Gauss transform. In *In Advances in Neural Information Processing Systems*, Vol.17, Cambridge, MA, MIT Press (2005).

[40] G. Fung and O. L. Mangasarian. Finite Newton method for Lagrangian support vector machine classification. In *Neurocomputing*, 39-55 (2003).

[41] O. L. Mangasarian and D. R. Musicant. Active set support vector machine classification. In *Advances in Neural Information Processing Systems*, Vol.13, 577-583, Cambridge, MA, MIT Press (2001).

[42] O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. In *Journal of Machine Learning Research*, 161-177 (2001).

[43] H. Shin and S. Cho. Fast pattern selection for support vector classifiers. In *Lecture Notes in Computer Science*, Vol. 2637, pp. 376-387. Springer (2003).

[44] D. Pavlov, J. Mao, and B. Dom. Scaling-up support vector machines using boosting algorithm. In *Proceedings of the International Conference on Pattern Recognition*, vol.2, 2219-2222, Barcelona, Spain (2000).

[45] R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of SVMs for very large scale problems. In *Neural Computation*, 1105C1114 (2002).

[46] Y-J. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. In *Proceeding of the First SIAM International Conference on Data Mining* (2001).

[47] R. Koggalage, and S. Halgamuge. Reducing the Number of Training Samples for Fast Support Vector Machine Classification. In *Neural Information Processing*, Vol.2, No.3, 57-65 (2004).

[48] D. Ziou, S. Tabbone. Edge detection techniques – an overview. In *International Journal of Pattern Recognition and Image Analysis* (1997).

[49] S. Canu, Y. Grandvalet, V. Guigue and A. Rakotomamonjy. SVM and Kernel Methods Matlab Toolbox. In *Perception Systems et Information*, INSA de Rouen, Rouen, France (2005).

[50] H. Shin, S. Cho. Invariance of neighborhood relation under input space to feature space mapping. In *Pattern Recognition Letters* 26, 707-718 (2005).

[51] M. A. Hall. Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 359-366 (2000).

[52] A. Gorban, B. Kegl, D. Wunsch and A. Zinovyev. Principal Manifolds for Data Visualisation and Dimension Reduction. In *Lecture Notes in Computational Science and Engineering*, Vol. 58, Springer (2007).

[53] R. Huber and L. V. Dutra. Feature Selection for ERS-1/2 InSAR Classification: High Dimensionality Case. In *Proceedings of the International Geoscience and Remote Sensing Symposium*, 1605-1607 (1998).

[54] K. Jong, J. Mary, A. Cornujeols, E. Marchiori and M. Sebag. Ensemble Feature Ranking. In *Lecture Notes in Computer Science*, Vol. 3202/2004, 267-278 (2004).

[55] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. In *Journal of Machine Learning Research 3*, 1157-1182 (2003).

[56] L. Yu and H. Liu. Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 856-863, Washington DC (2003).

[57] H. Liu, E. R. Dougherty, J. G. Dy, K. Torkkola, E. Tuv, H. Peng, C. Ding, F. Long, M. Berens and L. Parsons. Evolving feature selection. In *IEEE Intelligent Systems 20*, No. 6, 64-76 (2005).

[58] L. Y. Chuang, J. C. Li and C. H. Yang. Chaotic Binary Particle Swarm Optimization for Feature Selection using Logistic Map. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2008*, Vol. I, IMECS (2008).

[59] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. In *Artificial Intelligence*, 245-271 (1997).

[60] M A. Hall and L. A. Smith. Feature Subset Selection: A Correlation Based Filter Approach. In *International Conference on Neural Information Processing and Intelligent Information Systems 1997*, 855-858, Berlin (1997).

[61] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn and A. K. Jain. Dimensionality Reduction Using Genetic Algorithms. In *IEEE Trans. Evolutionary Computation*, Vol. 4, No. 2, 164-171 (2000).

[62] P. M. Narendra and K. Fukunage. A Branch and Bound Algorithm for Feature Subset Selection. In *IEEE Trans. Computers*, Vol. 6, No. 9, 917-922 (1977).

[63] P. Pudil, J. Novovicova and J. Kittler. Floating Search Methods in Feature Selection. In *Pattern Recognition Letters*, Vol. 15, 1119-1125 (1994).

[64] B. Roberto. Using mutual information for selecting features in supervised neural net learning. In *IEEE Transactions on Neural Networks*, 5(4):537-550 (1994).

[65] C. J. Tu, L. Y. Chuang, J. Y. Chang and C. H. Yang. Feature Selection using PSO-SVM. In *IAENG International Journal of Computer Science*, 33:1, IJCS-33-1-18 (2007).

[66] A. Rakotomamonjy. Variable selection using SVM-based criteria. In *Journal of Machine Learning Research* 3, 1357-1370 (2003).

[67] P. Gallinari and P. Gallinari. Feature selection with neural networks. In *Behaviormetrika*, Vol. 26, 16-6 (1999).

[68] D. W. Ruck, S. K. Rogers and M. Kabrisky. Feature selection using a multilayer perceptron. In *Neural Network Comput*, Vol. 2, 40-48 (1990).

[69] B. J. Frey and D. Dueck. Clustering by Passing Messages Between Data Points. In *Science*, Vol. 315, 972–976 (2007).

[70] O. Chapelle and V. Vapnik. Model selection for Support Vector Machines. In *S. Solla, T. Leen, and K.-R. Müler, editors, Adv. Neural Inf. Proc. Syst.* 12, Cambridge, MA, MIT Press (2000).

[71] K. Q. Shen, C. J. Ong, X. P. Li and E. W. Smith. Feature selection via sensitivity analysis of SVM probabilistic outputs. In *Machine Learning*, Vol. 70, No. 1, 1-20 (2008).

[72] Huaqing Li , Shaoyu Wang, and Feihu Qi. SVM Model Selection with the VC Bound. In *Lecture Notes in Computer Science*, Vol. 3314/2005, 1067-1071 (2005).

[73] J. Cohen, P. Cohen, S. G. West and L. S. Aiken. Applied multiple regression/correlation analysis for the behavioral sciences. (3rd ed.) In *Hillsdale, NJ: Lawrence Erlbaum Associates* (2003).

[74] G. Rätsch, T. Onoda and K. R. Müller. Soft margins for adaboost. In *Mach. Learn.*, Vol. 42, No. 3, pp. 287-319 (2001).

[75] S. X. Du and S. T. Chen. Weighted Support Vector Machine for Classification. In *IEEE International Conference on Systems, Man and Cybernetics* (2005).

[76] Lotfi A. Zadeh. Fuzzy Sets Information and Control 8(3): 338-353 (1965).

[77] D. S. Frossyniotis and A. Stafylopatis. A Multi-SVM Classification System. In *Lecture Notes in Computer Science*, Vol. 2096/2001, 198-207 (2001).

[78] S. E. Ryan and L. S. Porth. A tutorial on the piecewise regression approach applied to bedload transport data. In *USDA Forest Service RMRS-GTR-189* (2007).

[79] B. Li, Q. Wang and J. Hu. A fast SVM training method for very large datasets. In *IJCNN 2009 International Joint Conference on Neural Networks*, pp.1784-1789 (2009).

[80] E. Alpaydin. Techniques for combining multiple learners. In *Proceedings of Engineering of Intelligent Systems*, Vol.2, 6-12, ICSC Press (1998).

[81] Robert P. W. Duin and David M. J. Tax. Experiments with Classifier Combining Rules. In *Lecture Notes in Computer Science, Multiple Classifier Systems, Springer Berlin/Heidelberg*, Vol.1857/2000, 16-29 (2000).

[82] F. Roli and G. Fumera. Analysis of Linear and Order Statistics Combiners for Fusion of Imbalanced Classifiers. In *Lecture Notes in Computer Science, Multiple Classifier Systems*, Springer Berlin/Heidelberg, Vol.2364/2002, 252-261 (2002).

[83] S. Rakshita, A. Ghosh and B. U. Shankara. Fast mean filtering technique (FMFT). In *Pattern Recognition Society* Published by Elsevier B.V., Vol. 40, Issue 3, 890-897 (2007).

[84] H. Chen, P. Tǐno and Xin Yao. Probabilistic Classification Vector Machines. In *IEEE transactions on neural networks*, Vol. 20, No. 6, 901-914 (2009).

[85] K. Nishida and T. Kurita. Pedestrian Detection by Boosting Soft-Margin SVM with Local Feature Selection. In *Conference on Machine VIsion Applications*, 402-405 (2005).

[86] Q. Wu and D. Zhou. SVM Soft Margin Classifiers: Linear Programming versus Quadratic Programming. In *Neural Computation*, Vol. 17, No. 5, pp. 1160-1187 (2005).

[87] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. In *Neural Comput.* Vol. 10, No. 7, pp. 1895C1923 (1998).

[88] E. Alpaydin. Combined 5$\times$2 cv f test for comparing supervised classification learning algorithms. In *Neural Comput.* Vol. 11, no. 8, pp. 1885-1892 (1999).

[89] J. Kamruzzaman and R. A. Sarker. Forecasting of currency exchange rates using ANN: a case study. In *Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on*, Vol. 1, 793-797 (2003).

[90] L. Vanajakshi and L. R. Rilett. Support Vector Machine Technique for the Short Term Prediction of Travel Time. In *Intelligent Vehicles Symposium, 2007 IEEE*, 600-605 (2007).

[91] M. A. H. Dempster and C. M. Jones. Trading on the Edge: Neural, Genetic and Fuzzy Systems for Chaotic Financial Markets. In *New York Wiley* (1994).

[92] J. Huang, J. Lin, X. He and M. Dai. The Algorithm for Detecting Hiding Information Based on SVM. In *Advances in Neural Networks – ISNN* (2004).

[93] B. Li, J. Hu and K. Hirasawa. Financial time series prediction using a support vector regression network. In *IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence)*, 621-627 (2008).

[94] C. L. Giles, S. Lawrence and A. C. Tsoi. Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference. In *Machine Learning*, Vol. 44, 161-183 (2001).

[95] R. Fernandez. Predicting Time Series with a Local Support Vector Regression Machine. In *ACAI* (1999).

[96] M. A. H. Dempster and C. M. Jones. A real-time adaptive trading system using genetic programming. In *Quantitative Finance* Vol. 1, 397-413 (2001).

[97] D. Touretzky and K. Laskowski. Neural Networks for Time Series Prediction. In *Artificial Neural Networks* (2006).

[98] Y. Chen, B. Yang and J. Dong. Time-series prediction using a local linear wavelet neural network. In *Neurocomputing*, 449-465 (2006).

[99] M. Small and C. K. Tse. Minimum description length neural networks for time series prediction. In *PHYSICAL REVIEW* (2002).

[100] J. Wolfers and E. Zitzewitz. Prediction Markets. In *Journal of Economic Perspectives*, Vol. 18 (2004).

[101] National Institute on Aging. Available: http://www.healthandage.net/html/min/nih/content/hyperthermia.htm.

[102] S. Fidler and A. Leonardis. Robust LDA classification by subsampling. Available: http://www.cse.lehigh.edu/ rjm2/SACV/.

[103] K. Teknomo. K-Means Clustering Tutorials. Available: http://people.revoledu.com/kardi/tutorialh/kMean.

[104] UCI Repository. Available: http://archive.ics.uci.edu/ml/.

[105] Salt and pepper noise. Available: http://en.wikipedia.org/wiki/Salt_and_pepper_noise.

[106] Median filter. Available: http://en.wikipedia.org/wiki/Median_filter.

[107] Federal Reserve Board of Governors Statistics: Releases and Historical Data. http://www.federalreserve.gov/releases/.

[108] YahooFinance. http://finance.yahoo.com.