Large margin classifications and their applications in computer vision

Li, Xuchun

2007

Li, X. (2007). Large margin classifications and their applications in computer vision.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/4770>

<https://doi.org/10.32657/10356/4770>

Nanyang Technological University

Downloaded on 26 Apr 2023 18:40:01 SGT

Large Margin Classifications and Their Applications in Computer Vision

Xuchun Li

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2007

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

.....
Date

.....
Xuchun Li

Acknowledgments

I would like to express my deepest appreciation to my supervisor, Dr. Eric Sung, for his guidance, patience and many invaluable comments and suggestions throughout my Ph.D. study.

I also take this opportunity to thank the Nanyang Technological University for the award of a research scholarship which enabled me to undertake this research.

I wish to acknowledge my family members who showed so much concern and care for me in the course of my studies. Their encouragement and unyielding support have contributed to the completion of this project.

Special thanks also go to all the people in the Intelligent Machine Research Laboratory, especially Dr. Lei Wang, for their generous assistance provided throughout the entire study.

Summary

The concept of large margin classification is used as a principle for analyzing many different algorithms, such as Support Vector Machine (SVM) and Boosting. Here, it is the margin instead of raw training error that is used in designing classifiers. The two most popular large margin classification algorithms are SVM and Boosting. Both of them have been widely used in solving different kinds of computer vision problems, such as object recognition and image classification. Although large margin classification algorithms are proposed based on statistical learning theory and great success has been achieved, when they are used in some real applications, some difficulties still exist. The main problem of using SVM is its low training speed when handling large number of training samples. Meanwhile, the problem of designing Boosting is how we can well handle its diversity/accuracy dilemma. SVM-based active learning has achieved success in some binary classification problems. However, does it still work well for multi-label classification problems, which is often met in the task of image classification?

In this thesis, firstly, an SVM training-accelerating algorithm is proposed. This algorithm is based on the analysis of SVM in version space and the selection of

SBA samples. This algorithm can effectively speed up the training process of SVM when handling large data sets. For some data sets, the proposed algorithm exhibits speed-ups in the hundreds over conventional SVM. Secondly, a framework is proposed to handle the diversity/accuracy dilemma in Boosting. Based on this framework, two realizations of the framework are proposed to solve binary-class and multi-class classification problems, separately. Great improvement of generalization performance was obtained on both machine learning benchmark data sets and real face recognition databases. Finally, a multi-label SVM active learning algorithm is proposed to help decrease the needed number of training samples for multi-label image classification.

Contents

Acknowledgements	i
Summary	ii
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Motivation	2
1.1.1 Machine Learning	2
1.1.2 Large Margin Classification	4
1.1.3 Several Problems Pertaining Large Margin Classification Al- gorithms	5
1.2 Objectives	8
1.2.1 Speeding Up the Training Process of SVM	8
1.2.2 Handling the Diversity/Accuracy Dilemma in Boosting	9

1.2.3	Decreasing the Number of Training Samples for Multi-label SVM Classification	10
1.3	Major Contributions of the Thesis	12
1.3.1	A New SVM Training Accelerator	12
1.3.2	A New Framework for Handling the Diversity/Accuracy Dilemma in Boosting	14
1.3.3	A New SVM-based Active Learning Method for Multi-label SVM Classification	15
1.4	Organization of the Thesis	16
2	Background Principles and Literature Review	18
2.1	Large Margin Classification	19
2.1.1	Problem Setting	19
2.1.2	Vapnik Chervonenkis (VC) Theory	21
2.2	Support Vector Machines	26
2.2.1	Support Vector Machines Algorithm	26
2.2.2	Soft Margin Bound of SVM	34
2.2.3	Applications of SVM	35
2.2.4	The Training Time Problem of SVM	36
2.3	Boosting	38
2.3.1	Boosting Algorithm	39
2.3.2	Explanation of Boosting by Margin	42
2.3.3	Other Methods to Analyze Boosting	43

2.3.4	Applications of Boosting	44
2.3.5	The Diversity/Accuracy Dilemma in Boosing	45
2.4	SVM Active Learning	47
3	Sequential Bootstrapped SVM: A SVM Training Accelerator	48
3.1	Literature Review of SVM Training Accelerating Methods	49
3.2	General Approach of SeqSVM	53
3.3	Version Space	55
3.4	Existence of SeqSVM's Version Space	57
3.5	Basic Idea of SeqSVM	60
3.6	SBA Sample in the Version Space	62
3.7	Proposed SeqSVM Algorithm	65
3.8	Why the SeqSVM can Quickly Decrease the Version Space?	66
3.9	Convergence Proof	67
3.10	Speeding Up SeqSVM by using Online SVM	68
3.11	Initialization	70
3.11.1	The Best Pair	70
3.11.2	A Set of Training Samples	70
3.11.3	A Set of Initial Support Vectors	72
3.12	Time Complexity	73
4	Experimental Results of Sequential Bootstrapped SVM	75
4.1	General Information about Experiment	76

4.2	Artificial Data Set I: Checkerboard Data Set	78
4.3	Artificial Data Set II: Sine Cube Data Set	82
4.4	Benchmark Data Sets	86
4.5	Conclusion	89
5	BoostingDA Framework: Handling the Diversity/Accuracy Dilemma	
	in Boosting Methods	90
5.1	Introduction	91
5.2	Diversity Augmentation Methodology	94
5.3	Proposed BoostingDA Framework	95
5.4	Realization of Proposed Framework	97
5.4.1	Realization I: BoostingDA-SVM	97
5.4.2	Realization II: BoostingDA-LDA	101
6	Experimental Results of BoostingDA	105
6.1	General Information about Experiment	106
6.2	Test Diversity Augmentation Methodology	108
6.3	Test the Performance of BoostingDA-SVM	112
6.3.1	Compare BoostingDA-SVM with Bagging, AdaBoost and SVM	112
6.3.2	Comparison on Unbalanced Data Sets	114
6.4	Test the Performance of BoostingDA-LDA	120
6.4.1	The performance on the ORL database	121
6.4.2	The performance on the Yale face database B	122

6.5	Conclusion	123
7	Multi-label SVM Active Learning	124
7.1	Introduction	125
7.2	Background: Multi-label SVM	128
7.3	Proposed Multi-label SVM Active Learning	129
7.4	Experimental Results	133
7.4.1	Artificial Database	133
7.4.2	Real Color Image Database	133
7.5	Conclusion	137
8	Conclusion and Recommendations	138
8.1	Conclusion	139
8.2	Recommendations for further research	141
	Author's Publications	143
	Bibliography	145

List of Figures

2.1	Under-fitting and Over-fitting	21
2.2	three points can be shattered by oriented straight lines in the two dimensioned space.	22
2.3	The VC bound on the expected risk (2.7) is the sum of empirical risk and the confidence interval. The Structural Risk Minimization principle chooses the smallest bound at an appropriate point f^* by considering these two factors together.	24
2.4	SVM separating hyperplane for linearly separable data distribution .	27
2.5	SVM separating hyperplane for non-separable data distribution . . .	31
2.6	Kernel Mapping	33
2.7	Diversity vs Accuracy	46
3.1	Version Space Changes from I to II by Adding the Sample $\mathbf{x}_s$	63
3.2	Version Space Changes from I to II by Adding the Sample c	64
3.3	Flow Chart of SeqSVM Improved by Online SVM	69
4.1	Artificial Data Sets	78

4.2	Test Accuracy Comparison on Checkboard Data Sets	79
4.3	Training Time Comparison on Checkboard Data Sets	79
4.4	Number of Support Vectors Comparison on Checkboard Data Sets . .	80
4.5	Test Accuracy Comparison on Sine Cube Data Sets	84
4.6	Training Time Comparison on Sine Cube Data Sets	84
4.7	Number of Support Vectors Comparison on Sine Cube Data Sets . . .	85
5.1	The Test Error of SVM vs. σ and C Values	98
5.2	PCA Dimension vs Recognition Accuracy	102
6.1	Diversity-Accuracy of BoostingDA-SVM and BoostingDA-SVM- σ_{middle}	111
6.2	Compare BoostingDA-SVM and SVM on ‘Splice’ Dataset	116
6.3	10 Images of 2 Persons in ORL Database	121
6.4	Recognition Accuracy on ORL Database	121
6.5	20 Images of 2 Persons in Yale Face Database B	122
6.6	Recognition Accuracy on Yale Face Database B	122
7.1	The images with multi-label	126
7.2	Comparison Result of Artificial Data Classification	134
7.3	Comparison result of real image classification.	135
7.4	the influence of parameters	136

List of Tables

2.1	AdaBoost	41
3.1	Sequential Bootstrapped SVM	65
3.2	Search Initial Pair	71
4.1	General Information of Data Sets	77
4.2	Comparison on Benchmark Data Sets	87
4.3	Comparison on Benchmark Data Sets (continue)	88
5.1	Proposed BoostingDA Framework	96
5.2	Realization I for Solving Binary-class Classification Problems	100
5.3	Realization II for Solving Multi-class Classification Problems	104
6.1	General Information of Data Sets.	107
6.2	Comparison of Diversity Values, Generalization Errors with Standard Deviation Are in the Parenthesis in %. w_{div} Are Fixed As 0.3.	109

6.3	Generalization Errors with Standard Deviation of BoostingDA-SVM (BDA-SVM), BoostingDA-SVM with single best parameters for SVM (BDA-S-SVM), Bagging, AdaBoost and SVM Algorithms on Bench- mark Data Sets in %	113
6.4	general characteristics of data sets and the amount of over-sampling of the minority class we did for the SMOTE algorithm	117
6.5	g-means metric results on the five uci imbalanced data sets	119
6.6	AUS results on the five UCI imbalanced data sets	119

Chapter 1

Introduction

1.1 Motivation

In this section, the background of machine learning and the four research periods of learning problems are introduced firstly. Within the last decade, *large margin classification* technique has been investigated widely as the result of the statistical learning theory. Recent theoretical and experimental results show that these larger margin classification algorithms have achieved better generalization performance than previous ones. Hence, following the introduction of machine learning, large margin classification technique is briefly introduced with its several problems.

1.1.1 Machine Learning

The objective of machine learning [120] is whether computers can be designed to learn by themselves? If we could realize this objective, the influence would be dramatic. Imagine if computerized medical treatment programmes can learn from medical records so that new diseases can be combated by treatments more easily, house building program can learn from previous building experience to optimize energy costs based on the particular usage patterns and so on. Furthermore, a deeper understanding of machine learning could in turn lead to a better understanding of human learning abilities. In the field of machine learning, there are lots of branches: Decision Tree Learning [141] [142] [26], Artificial Neural Networks [45] [138] [100] [147], Bayesian Learning [119] [83] [51], Computational Learning [176] [21] [12] [127], Instance-based Learning [39] [5] [7] [180], Genetic Algorithm [13] [95] [159] [74], Reinforcement Learning [18] [86] [190] and so on. In the research history of learning

problems, there are four periods which can be characterized by four events [178]: In 1962, F. Rosenblatt proposed the first model of a learning machine, called Perceptron [70]. The Perceptron was investigated to solve pattern recognition problems. In the second period, fundamentals of the learning theory are constructed. In 1971, the bounds for the rate of convergence were obtained [187], these bounds made the introduction of a novel inductive principle - Structural Risk Minimization [188], which completed the development of pattern recognition theory. In 1989, necessary and sufficient condition for consistency of the empirical risk minimization inductive principle and maximum likelihood method were proposed [186], which completed the analysis of empirical risk minimization inductive inference. In the third period, Neural Networks were developed. In 1986, Neural Networks were proposed by several authors independently for simultaneously constructing the vector coefficients for all neurons of the Perceptron using the back-propagation method [103] [52]. We are currently at the fourth period. In this period, more attention is focused on the fundamentals and statistical learning theory played a more important role. Many new algorithms were proposed which aimed to minimize the theoretical bounds on the error rate. These algorithms not only showed nice mathematical properties (such as uniqueness of the solution, independence of the dimensionality of data), but also exhibited excellent performance. The typical algorithms are the so-called large margin algorithms, such as Support Vector Machines [44] and Boosting [67].

The machine learning problems can be roughly categorized as either *supervised learning* or *unsupervised learning*. In the field of supervised learning, there is a label

associated with each data. The goal of supervised learning is to predict the label of a new unseen data based on the current obtained training data with their labels. Hence, supervised learning is to generalize unseen data. On the other hand, in the field of unsupervised learning, there is no label associated with data. Hence, the goal of unsupervised learning is to describe how the data are organized. In this thesis, supervised learning is the main concern.

1.1.2 Large Margin Classification

The concept of large margin classification is used as a unifying principle for analyzing many different algorithms, such as SVM and Boosting. Here, it is the *margin* instead of raw training error that is used when designing these classifiers. At present, algorithms that explicitly or implicitly exploit the margins are among the most promising methods to learning from data. The two most popular large margin classification algorithms are SVM and Boosting. SVM was developed from the theory of structural risk minimization while Boosting was developed from the theory of divide and conquer. Both of them have been widely used in solving different kinds of problems, such as object recognition [123], image classification [130] [166] [116] [64], gait classification [16], signal classification [46], audio classification [75], call classification [148], gene selection [59], texture classification [91], land cover classification [35], frontal face authentication [124], image fusion [107], active learning for image retrieval [171] and so on.

1.1.3 Several Problems Pertaining Large Margin Classification Algorithms

Although large margin classification algorithms are proposed based on statistical learning theory and great success has been achieved, when they are used to solve some real applications, some problems still exist. These problems are introduced briefly as follows.

The problem of using SVM is its low training speed when handling large number of training samples. Since during the training process, the standard SVM has to solve a large quadratic programming (QP) optimization problem [121] whose time complexity is $O(N^3)$ and space complexity is $O(N^2)$, where N is the number of training samples, training time and memory increase dramatically with the increase of the number of training samples. Therefore, it is difficult to solve large training set problems using the standard SVM. Some methods were proposed to speed up the training process of SVM, such as Chunking [42], SMO [157] (detailed review of these methods is given in chapter 3). Although these methods can speed up the SVM training process, they still need a long training period when handling very large training sets, for instance, more than 100,000 training samples. Therefore, the training process is still expected to be further speeded up when the SVM is used to handle very large training set problems.

The Boosting algorithm finds a highly accurate classifier by combining many moderately accurate component classifiers. It is known that the generalization performance of a Boosting classifier is affected by both its component classifiers' ac-

curacy and their diversity [193]. Here, the accuracy is the training accuracy of component classifiers and we refer to the measure of disagreement among different component classifiers as the diversity [117]. In [98], 10 different measures of diversity are compared and it is found that most of them are highly correlated. In this thesis, the measure of diversity in [117] is used. In practical design of Boosting classifiers, if the component classifiers are too accurate, it is difficult to find diverse ones, and combining these accurate but non-diverse classifiers often leads to very limited improvement. On the other hand, if the component classifiers are too inaccurate, although we can easily find diverse ones, the combined result may be worse than that of combining both more accurate and diverse component classifiers. This is because the combined result is dominated by too many inaccurate component classifiers, and will be wrong most of the time, leading to a poor generalization result. These problems result in the dilemma between component classifiers' accuracy and diversity [54]: How can we maximize the diversity under the condition of obtaining a fairly good component classifiers' accuracy for Boosting methods?

Multi-label classification problems are often met in real applications, such as document classification [115] and functional gene classification [61] problems. Here, multi-label means that a sample can belong to one or several different classes. For example, a multi-label image can belong to both "sunset" class and "mountain" classes (detailed introduction is given in chapter 7). When labelling these multi-label samples, it is very time-consuming and costly. Fortunately, in many cases, we can carefully select images used to query the oracle¹ instead of randomly picking

¹Oracle knows the ground truth of the selected image. In practice, oracle can be a group of

images. If the selection strategy is well-designed, this approach can give rise to better classification result under a given number of labelling chances. This procedure can be categorized into active learning. SVM-based active learning has achieved success when handling some binary classification problems [156] [32] [172]. However, does the SVM still work well for the aforementioned multi-label classification problems?

1.2 Objectives

As discussed in the previous section, when large margin classification algorithms are used in some real applications, some difficulties still exist. Hence, the main objective of my Ph.D. work is to solve these problems.

1.2.1 Speeding Up the Training Process of SVM

Support Vector Machine was developed from the principle of Structural Risk Minimization. Due to its impressive generalization performance on classification problems, SVM has yielded much success in a variety of domains, such as image processing [90], micro-array data analysis [198] and audio classification [75]. Large data set problems are often met in real applications, such as automatic document classification. Unfortunately, when one tries to scale up the standard SVM to large data sets, the training time increases greatly with the standard SVM algorithm. It is known that it is the support vectors that determines the decision boundary of the SVM classifier while all other training samples can be removed. Hence, can we only select support vectors for training to speed up SVM? In this thesis, this question will be given a answer with a detailed analysis of the SVM training process.

1.2.2 Handling the Diversity/Accuracy Dilemma in Boosting

One of the major developments in machine learning in the past decade is the Boosting method. It is known that the generalization performance of a Boosting classifier is affected by both component classifiers' accuracy and their diversity. There exists a dilemma between component classifiers' accuracy and diversity as introduced in the above section: How can we maximize the diversity under the condition of obtaining a fairly good component classifiers' accuracy for Boosting methods?

In order to handle this dilemma, we want to maximize the diversity under the condition of obtaining a fairly good component classifiers' accuracy. A fact is that if the diversity of candidate component classifiers is constrained into a small scale, we cannot ensure that the selected component classifiers with satisfactory diversity level can be really found. Hence, diversity over candidate component classifiers is preferred to be as large as possible. On the other hand, it is known that the diversity between two classifiers on a certain data set is decided by their classification results on this data set. The classification result of a classifier is in turn affected by two factors: the training data and this classifier's learning ability. Although current Boosting methods can create diversity by setting different weights to different training data, it does not consider the diversity resulting from using classifiers with different learning abilities. Instead, they only used those having same learning abilities as component classifiers. Hence, an effective diversity augmentation method for candidate component classifiers is needed, especially from the perspective of

using component learning classifiers with different learning abilities. It is hoped that this diversity augmentation method can offer more suitable candidate component classifiers for Boosting methods to better handle the dilemma.

1.2.3 Decreasing the Number of Training Samples for Multi-label SVM Classification

Multi-label classification is a case in which the data can have one or more labels. Therefore, multi-class classification can be viewed as a special case of multi-label classification. Multi-label classification has been applied to document classification [115] and functional gene classification [61] problems. In real applications, labelling multi-label samples is often a time-consuming and costly task. Fortunately, in many cases, we can carefully select samples to query the oracle ² instead of randomly picking samples. If the selection strategy is well-designed, this approach can give rise to better classification result under a given number of labelling chances. This procedure can be categorized into active learning. SVM-based active learning algorithm has been widely investigated and great success has been achieved to improve the classification performance [156] [32] [172] (detailed review of SVM active learning is given in chapter 2). However, none of them addressed the multi-label classification problem. Hence, can we use active learning to help multi-label classification decrease the number of needed training samples?

²Oracle knows the ground truth of the selected sample. In practice, oracle can be a group of human observers or annotators.

Generally speaking, the objectives of this thesis are solving the above mentioned problems: accelerating the training process of SVM when handling large data sets; handling the diversity/accuracy dilemma in Boosting by increasing the diversity among component classifiers; using SVM-based active learning technique to decrease the number of training samples for multi-label classification.

1.3 Major Contributions of the Thesis

In this thesis, firstly, an SVM training accelerating algorithm is proposed. This algorithm is based on the analysis of SVM in version space and the selection of SBA samples. This algorithm can effectively speed up the training process of SVM when handling large data sets. For some data sets, the proposed algorithm exhibits speed-ups in the hundreds over conventional SVM. Secondly, a framework is proposed to handle the diversity/accuracy dilemma in Boosting. Based on this framework, two realizations of the framework are proposed to solve binary-class and multi-class classification problems, separately. Great improvements were obtained on both machine learning benchmark data sets and real face recognition databases. Finally, a multi-label SVM active learning algorithm is proposed to help decrease the needed number of training samples for multi-label image classification. The main contributions of this thesis are summarized as follows:

1.3.1 A New SVM Training Accelerator

Support Vector Machines has been proved to be one of the most successful machine learning algorithms. However, it requires to solve a quadratic optimization(QP) problem so that its training time increases dramatically with the increase in the number of training samples. Hence, the standard SVM with batch learning has difficulty in handling large scale problems. In this thesis, a SVM training accelerator is proposed, called the Sequential Bootstrapped SVM (SeqSVM), to speed up the training process of SVM. The SeqSVM is an attempt to find an approximation to

the SVM that will significantly reduce the training time while achieving similar generalization performance. In the beginning, the SeqSVM trains a SVM classifier on a small part of all training samples and then keeps on selecting the so-called SBA samples from the given large training set to retrain this SVM until all SBA samples are selected. The key idea of SeqSVM is to help SVM to pick the next best sample to learn from. It does this by finding the sample that is furthest away from and wrongly classified by the current optimal hyperplane. Moreover, because the most extreme sample is picked that most disagrees (most uncertainty) with the current hyperplane, the learning will be the fastest on average. Two difficulties have to be overcome. The first is that the number of SeqSVM iterations is proportional to the number of SBA samples so that if the number of SBA samples is large, a large number of training iterations are needed leading to a long training time. The second is that when the class distributions are not separable, it is not easy to pick SBA samples. In this thesis, we show how they have been overcome. Experimental results on both artificial data sets and benchmark data sets demonstrated the effectiveness of proposed algorithm to reduce the learning time of SVM on the whole training set. For some data sets, the SeqSVM exhibits speed-ups in the hundreds over the conventional SVM.

1.3.2 A New Framework for Handling the Diversity/Accuracy

Dilemma in Boosing

A new diversity augmentation methodology is also proposed by purposely expanding the scale of learning ability over candidate component classifiers. This methodology can effectively improve the diversity and give more suitable candidate component classifiers to Boosting methods, leading to better generalization performance. Then, based on this methodology, we design a framework, named ***Boosting** integrating **Diversity** and **Accuracy***, BoostingDA for short, to handle the diversity/accuracy dilemma in Boosting methods. Finally, two realizations of proposed BoostingDA framework are developed by using Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA) as component classifiers, separately. They are named as BoostingDA-SVM and BoosingDA-LDA. These two realizations are designed to handle binary-class and multi-class classification problems separately. In the BoostingDA-SVM, the proposed diversity augmentation methodology is realized by training SVM component classifiers with different model parameters. While in the Boosting-LDA, it is realized by training LDA in the Principal Components Analysis (PCA) subspace with different dimensions. The effectiveness of the proposed diversity augmentation methodology and the two realizations are experimentally demonstrated on machine learning benchmark data sets and face recognition databases.

1.3.3 A New SVM-based Active Learning Method for Multi-label SVM Classification

Image classification is an important task in computer vision. However, how to assign suitable labels to images is a subjective matter, especially when some images can be categorized into multiple classes simultaneously. Multi-label image classification focuses on the problem that each image can have one or multiple labels. It is known that manually labelling images is time-consuming and expensive. In order to reduce the human effort of labelling images, especially multi-label images, a multi-label SVM active learning method is proposed. Two selection strategies are also proposed: Max Loss strategy and Mean Max Loss strategy. Experimental results on both artificial data and real-world images demonstrate the advantage of proposed method.

1.4 Organization of the Thesis

In the chapter 2, large margin classification technique is briefly reviewed.

In the chapter 3, the Sequential Bootstrapped SVM (SeqSVM) algorithm is proposed. Following the introduction of the general approach of SeqSVM algorithm, a way to find the SBA samples that have the most probability to be support vectors is explained and we explain by way of version space why they induce the fastest convergence to the optimal hyperplane. Then, the outline of the SeqSVM algorithm is described in detail. Afterwards, the convergence proof of SeqSVM is given. Finally, online SVM is used to speed up the SeqSVM and the time complexity is given.

In the chapter 4, two artificial data sets and ten benchmark data sets with different number of training samples are used to compare the performance of proposed SeqSVM algorithm, standard SVM algorithm, CVM algorithm, and active learning with SVM algorithm.

In the chapter 5, proposed diversity augmentation methodology for Boosting methods will be presented firstly. Then, why we need a diversity augmentation method when handling the diversity/accuracy dilemma will be explained. Finally, based on proposed diversity augmentation methodology, our framework to handle the diversity/accuracy dilemma in Boosting methods will be introduced in detail.

In the chapter 6, experimental comparisons among proposed BoostingDA-SVM, Bagging, AdaBoost and SVM on machine learning benchmark data sets are given. Furthermore, proposed BoostingDA-LDA algorithm are also compared with other algorithms dealing with the face recognition problem on several face recognition

databases.

In the chapter 7, a multi-label SVM active learning method was proposed for multi-label image classification. Based on theoretic analysis, we proposed two selection strategies: Max Loss strategy and Mean Max Loss strategy. To our best knowledge, we are the first one who use active learning to help multi-label classification. The experimental results on both artificial and real color image databases demonstrated that the two proposed methods can provide better classification performance than original multi-label SVM.

Conclusions and recommendations for further research are presented in chapter 8.

Chapter 2

Background Principles and Literature Review

2.1 Large Margin Classification

2.1.1 Problem Setting

For the case of binary-class classification problems, the task can be formulated in the following ways: given a set of separating functions, F :

$$F = \{f(\theta), \theta \in \Psi\}, \quad f(\theta) : \mathbf{R}^m \rightarrow \{-1, +1\} \quad (2.1)$$

where θ is the parameter of function f . Also given a set of training samples, I , with labels

$$I = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}, \quad \mathbf{x}_i \in \mathbf{R}^m, \quad y_i \in \{-1, +1\} \quad (2.2)$$

These training samples are obtained independent identically distributed (i.i.d.) from an unknown probability density distribution $P(\mathbf{x}, y)$ containing the underlying dependency. The test samples are assumed to be generated from the same probability density distribution as the training samples. We want to learn a separating function, f^* , from F to satisfy that f^* provides the smallest expected risk (also *generalization error*)

$$R[f] = \int l(f(\mathbf{x}), y) dP(\mathbf{x}, y) \quad (2.3)$$

where l denotes a loss function, such as the 0/1-loss

$$l(f(\mathbf{x}), y) = \Theta(yf(\mathbf{x})) \quad (2.4)$$

where $\Theta(z) = 0$ if $z > 0$, otherwise 1. The most commonly used loss function is the squared loss:

$$l(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2 \quad (2.5)$$

More loss functions can be found in [8]. However, the expected error cannot be minimized directly. This is because the underlying probability density distribution $P(\mathbf{x}, y)$ is unknown. The only available information is contained in the training set (2.2). Hence, the *empirical risk* is minimized instead

$$R_{emp}[f, I] = \frac{1}{N} \sum_{i=1}^N l(f(\mathbf{x}_i), y_i) \quad (2.6)$$

This principle is called the *empirical risk minimization* (ERM) inductive principle. It is often used. For example, the least-squares method or the maximum likelihood method are realizations of the ERM inductive principle.

Obviously, if the training set is very large or even infinite, the empirical risk will converge toward the expected risk. Unfortunately, it is not possible in the real-world problems. Therefore, under-fitting or over-fitting problems sometimes might occur. An illustration is shown in Figure 2.1. In this figure, if we only know a small training set(left), both the line and the sine curve may be correct. However, only when a large set of training samples are given, we can decide which decision classifies the true data distribution more accurately. For example, if the true data distribution looks like the one in Figure 2.1 (middle), the line decision is *under-fitting*. While if the true data distribution looks like the one in Figure 2.1 (right), the sine curve decision is *over-fitting*. Hence, how should we minimize the expected risk (2.3) on the basis of empirical risk (2.6) and avoid the above under-fitting or over-fitting problems? In the following sections, Vapnik-Chervonenkis (VC) theory and large margin classification algorithms are introduced to give further insight into such problems.

Figure 2.1: Under-fitting and Over-fitting

2.1.2 Vapnik Chervonenkis (VC) Theory

In this subsection, the Vapnik Chervonenkis (VC) theory is briefly introduced, which includes the definition of VC dimension, the bound of the generalization error, and the Structural Risk Minimization (SRM) theory.

• Shattering points with oriented hyperplanes

Considering a binary-class classification problem, given a set of N points, $I = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, if these N points can be labelled in all possible 2^N ways, and for each labelling way, a separating function, $f(\mathbf{x})$, from the set of separating functions, $F = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots\}$, can be found to correctly label these points. Then, the point set I can be shattered by the function set F . An illustration is shown in Figure 2.2. In this figure, there are totally three points (black point and circle) in a two dimensional space. The separating function set consists of a set of oriented straight lines in the two dimensional space. For a given oriented straight line, all points on one side are assigned the label 1, and all points on the other side, the label

Figure 2.2: three points can be shattered by oriented straight lines in the two dimensional space.

–1. It can be found that these three points can be shattered by the set of oriented straight lines.

• VC dimension of a set of functions

The VC dimension was proposed by Vapnik and Chervonenkis [178]. It is a property of a set of functions $F = \{f(\theta)\}$. Here, θ is a generic set of parameters so that a choice of θ corresponds to a particular function. For a binary-class classification problem, the VC dimension for the function set F is defined as the maximum number of training samples that can be shattered by F . For example, in Figure 2.2, since it is possible to find three points that can be shattered by a set of straight lines in the two dimensional space and not possible to find four points, the VC dimension of the straight lines in the two dimensional space is three.

• VC Bounds

Recall that in the Problem Setting subsection 2.1.1, we met the problem that how we could minimize the expected risk $R[f]$ (2.3) on the basis of empirical risk $R_{emp}[f, I]$ (2.6) and avoid the under-fitting or over-fitting problems. In the following, the VC Bounds [178] [126] and Structural Risk Minimization(SRM) [179] [178] principle will be introduced to handle such problems.

Let ϱ denote the VC dimension of the function set F . Given a set of N samples, I , the empirical risk, $R_{emp}[f, I]$, uses the 0/1 loss (2.4). For all $\varphi > 0$ and $f \in F$, the following inequality bounding the expected risk $R[f]$ holds with probability of at least $1 - \varphi$ for $N > \varrho$.

$$R[f] \leq R_{emp}[f, I] + \sqrt{\frac{1}{N} \left[\varrho \left(\ln \frac{2N}{\varrho} + 1 \right) - \ln \left(\frac{\varphi}{4} \right) \right]} \quad (2.7)$$

Similar bounds by using other kinds of loss [179] or other kind of complexity measure [192] have also been proposed. From this bound, it can be found that if we want to minimize the expected risk, both the empirical risk and the VC dimension of the set of functions should be considered. This leads to the following Structural Risk Minimization principle.

• Structural Risk Minimization Principle

First, let us construct a nested family of function sets

$$F_1 \subset F_2 \subset \cdots \subset F_m \quad (2.8)$$

where the VC dimension, ϱ_i , of function set, F_i , satisfies

$$\varrho_1 \leq \varrho_2 \leq \cdots \leq \varrho_m \quad (2.9)$$

Then, assume f_i be the solution of the empirical risk minimization (2.6) in F_i for a certain training data set. As the index i increases, the empirical risk decrease. However, the term responsible for the confidence interval (the second summand in the inequality (2.7)) increases. This is because of the increase of VC dimension. Therefore, the Structural Risk Minimization principle defines a trade-off between the empirical risk and the VC dimension. SRM principle chooses the function set F^* which can attain the minimal VC bounds (2.7) on the expected risk by taking both factors into account. An illustration is given in Figure 2.3.

Figure 2.3: The VC bound on the expected risk (2.7) is the sum of empirical risk and the confidence interval. The Structural Risk Minimization principle chooses the smallest bound at an appropriate point f^* by considering these two factors together.

Now, the task is how we can realize the Structural Risk Minimize principle. Since it is difficult to accurately calculate the VC dimension of function set, how do we find the f^* ? A good alternative method is to use *margin* as criteria. Let us give the

definition of margin first.

We define the margin ρ of a sample $\mathbf{x}$, whose label is y , with respect to a function f as the product of y and the decision value of f on $\mathbf{x}$ [44]

$$\rho(\mathbf{x}) = yf(\mathbf{x}) \quad (2.10)$$

Note that $\rho > 0$ implies that correct classification of $\mathbf{x}$ by f . The margin $\rho(f)$ of f with respect to the training set I is the minimal margin of $\mathbf{x}$, where $\mathbf{x} \in I$

$$\rho(f) = \min_{\mathbf{x} \in I} \rho(\mathbf{x}) \quad (2.11)$$

The two successful learning algorithms based on margin bounds to minimize the expected risk, which are called large margin classification algorithms, are Support Vector Machines [44] and Boosting [69]. They are the focuses of this thesis and will be introduced in the next two sections, separately.

The margin bounds of SVMs [162] [163] should be introduced.

2.2 Support Vector Machines

Support Vector Machines (SVM) [22] [42] [178] [179] is an approximate implementation of the Structural Risk Minimization principle. This induction principle is based on the fact that the generalization error is bounded by the sum of the training error and a term that depends on the Vapnik-Chervonenkis (VC) dimension. For a linearly separable data distribution, SVM can produce zero value for the first term of this bound and simultaneously be able to minimize the second term. Hence, it can provide a good generalization performance. For a non-linearly separable data distribution, a kernel mapping technique is used to find the optimal separating hyperplane in the induced kernel space. Detailed introduction of SVM is given in this section.

2.2.1 Support Vector Machines Algorithm

In a binary-class classification problem, a training set I of size N is given,

$$I = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbf{R}^d, y_i \in \{+1, -1\}, i = 1, 2, \dots, N\} \quad (2.12)$$

where $\mathbf{x}_i$ is a training sample and its label is y_i , denotes the class labels +1 and -1 separately. These samples are assumed to be i.i.d. according to some unknown fixed distribution $P(\mathbf{x})$, their labels are distributed according to some unknown conditional distribution $P(y|\mathbf{x})$. To begin with, we assume that the two classes are linearly separable. Hence, the equation of a separating hyperplane is

$$\mathbf{w}^\top \mathbf{x} + b = 0 \quad (2.13)$$

where $\mathbf{w}$ is the normal vector of the separating hyperplane; b is the offset. A data, $\mathbf{x}$, will be given the label $+1$ if $\mathbf{w}^\top \mathbf{x} + b > 0$ while -1 if $\mathbf{w}^\top \mathbf{x} + b \leq 0$. The assumption of linearly separable data distribution is made here in order to explain the basic idea of SVM. This assumption will be relaxed later. An illustration of SVM separating hyperplane for the linearly separable data distribution is shown in Figure 2.4.

Figure 2.4: SVM separating hyperplane for linearly separable data distribution

In this figure, the distance of the closest data, $\mathbf{x}$, from the SVM separating hyperplane is called *margin*, denoted by ρ . The goal of SVM is to find the particular hyperplane whose margin is maximal. Due to the fact that the function associated with the hyperplane $\mathbf{w}^\top \mathbf{x} + b = 0$ does not change if we re-scale the hyperplane to $(\lambda \mathbf{w}, \lambda b)$, for $\lambda \in \mathbf{R}^+$. By requiring the scaling of $\mathbf{w}$ and b to be such that the data closest to the separating hyperplane satisfies $\mathbf{w}^\top \mathbf{x} + b = \pm 1$, we obtain the *canonical form*. In this case, the data, which precisely satisfy $\mathbf{w}^\top \mathbf{x} + b = \pm 1$, are called *support*

vectors. Support vectors satisfy

$$\mathbf{w}^\top \mathbf{x}_{sv} + b = \begin{cases} -1 : \text{if } y_{sv} = -1 \\ +1 : \text{if } y_{sv} = +1 \end{cases} \quad (2.14)$$

where y_{sv} is the label of the support vector $\mathbf{x}_{sv}$. Hence, the margin ρ can be calculated from two support vectors from opposite classes as follows:

$$\begin{aligned} \rho &= \frac{1}{2} \left(\left(\frac{\mathbf{w}}{\|\mathbf{w}\|_2} \right)^\top \mathbf{x}_{sv+} + b \right) - \left(\left(\frac{\mathbf{w}}{\|\mathbf{w}\|_2} \right)^\top \mathbf{x}_{sv-} + b \right) \\ &= \frac{1}{2 \|\mathbf{w}\|_2} \left((\mathbf{w}^\top \mathbf{x}_{sv+} + b) - (\mathbf{w}^\top \mathbf{x}_{sv-} + b) \right) \\ &= \frac{1}{\|\mathbf{w}\|_2} \end{aligned} \quad (2.15)$$

From this equation, it can be found that maximizing the margin is equivalent to minimizing the Euclidean norm of $\mathbf{w}$. Furthermore, when the data distribution is linearly separable, the goal of the SVM is to find the optimal hyperplane with training samples subject to the following constraint,

$$y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1, \quad i = 1, 2, \dots, N \quad (2.16)$$

Hence, for a linearly separable data distribution, the optimal parameters $\mathbf{w}^*$ and b^* corresponding to the optimal classifier $f^*(\mathbf{x})$ with maximal margin can be obtained by solving the following optimization problem,

$$\text{minimize:} \quad g(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 \quad (2.17)$$

$$\text{subject to:} \quad y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1, \quad i = 1, 2, \dots, N \quad (2.18)$$

The solution to this optimization problem is given by the saddle point of the Lagrange (primal) function:

$$L(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i \{y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1\} \quad (2.19)$$

where α_i is a Lagrange multiplier corresponding to the sample $\mathbf{x}_i$. This function will be maximized with respect to $\mathbf{w}$ and b . The derivatives of L with respect to the primal variables must vanish,

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \quad (2.20)$$

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \quad (2.21)$$

leading to

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i \quad (2.22)$$

$$\sum_{i=1}^N \alpha_i y_i = 0 \quad (2.23)$$

According to the Karash-Kuhn-Tucker(KKT) theorem of optimization theory [55], only the Lagrange multipliers α_i can be nonzero which correspond to the constraints (2.18) which are precisely met:

$$\alpha_i [y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) - 1] = 0 \quad (2.24)$$

The sample $\mathbf{x}$ whose corresponding α_i satisfies $\alpha_i > 0$ is a support vector. The corresponding dual objective function is found by substituting the obtained relations (2.22)(2.23) into the above primal function (2.19):

$$\text{maximize: } L(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j \quad (2.25)$$

$$\text{subject to: } \sum_{i=1}^N y_i \alpha_i = 0, \quad \forall i : 0 \leq \alpha_i \quad (2.26)$$

Note that this dual problem is cast entirely in terms of the training samples. Hence, having determined the optimal Lagrange multipliers, denoted by $\hat{\alpha}_i$, we can compute the optimal $\mathbf{w}_o$ using equation (2.22)

$$\mathbf{w}_o = \sum_{i=1}^N \hat{\alpha}_i y_i \mathbf{x}_i \quad (2.27)$$

To compute the optimal bias b_o , we can use the $\mathbf{w}_o$ and a positive support vector

$$b_o = 1 - \mathbf{w}_o^\top \mathbf{x}_{sv+} \quad (2.28)$$

In practice, a linearly separable distribution often does not exist. This leads to the following constraint being violated.

$$y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1, \quad i = 1, 2, \dots, N \quad (2.29)$$

The violation can arise in one of two ways:

- The data falls inside the region of separation but on the correctly classified side of the separating hyperplane.
- The data falls on the wrong side of the separating hyperplane.

Hence, the following slack variables are used to handle the non-separable data distribution [179]:

$$\xi_i \geq 0, \quad i = 1, 2, \dots, N \quad (2.30)$$

and use a relaxed separation constraints instead of (2.18):

$$y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1 - \xi_i, \quad i = 1, 2, \dots, N \quad (2.31)$$

If $0 \leq \xi_i \leq 1$, this data falls inside the region of separation but on the correctly classified side of the separating hyperplane. If $\xi_i > 1$, this data falls on the wrong side of the separating hyperplane. The support vectors are those data which satisfy the inequality (2.31). An illustration is shown in Figure 2.5.

Figure 2.5: SVM separating hyperplane for non-separable data distribution

Hence, for non-separable data distributions, the soft margin SVM [179] should solve the following optimization problem instead:

$$\text{minimize: } g(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad (2.32)$$

$$\text{subject to: } y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (2.33)$$

where, ξ_i is a slack variable and C is the regularization parameter. Therefore, The Lagrange (primal) function is:

$$L(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i [y_i(\mathbf{w}^\top \mathbf{x} + b) - (1 - \xi_i)] - \sum_{i=1}^N \mu_i \xi_i \quad (2.34)$$

where α_i and μ_i are Lagrange multipliers which correspond to $\mathbf{x}_i$. This function will be maximized with respect to $\mathbf{w}$, b and ξ . Setting the respective derivatives to zero, we get

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i \quad (2.35)$$

$$\sum_{i=1}^N \alpha_i y_i = 0 \quad (2.36)$$

$$\alpha_i = C - \mu_i \quad (2.37)$$

The corresponding dual objective function is found by substituting the above obtained relations into the above primal function (2.34):

$$\text{maximize: } L(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j \quad (2.38)$$

$$\text{subject to: } \sum_{i=1}^N y_i \alpha_i = 0, \quad \forall i : 0 \leq \alpha_i \leq C \quad (2.39)$$

Note that the slack variables do not appear in this dual problem. The dual problem for the case of non-separable data distribution is similar to that of separable data distribution. The only difference is that the constraint $0 \leq \alpha_i$ for the separable data distribution is replaced by a more stringent constraint $0 \leq \alpha_i \leq C$. Hence, having determined the optimal Lagrange multipliers, denoted by $\hat{\alpha}_i$, we can compute the optimal $\mathbf{w}_o$:

$$\mathbf{w}_o = \sum_{i=1}^N \hat{\alpha}_i y_i \mathbf{x}_i \quad (2.40)$$

To compute the optimal bias b_o , we can use the procedure similar to the one used for separable data distribution. Specifically, the Kuhn-Tucker conditions are defined by

$$\alpha_i [y_i (\mathbf{w}^\top \mathbf{x} + b) - (1 - \xi_i)] = 0, \quad i = 1, 2, \dots, N \quad (2.41)$$

$$\xi_i \mu_i = 0, \quad i = 1, 2, \dots, N \quad (2.42)$$

By combining equation (2.37) and (2.42), we obtain that

$$\xi_i = 0, \quad \text{if } \alpha_i < C \quad (2.43)$$

Hence, we may calculate the optimal bias b_o by taking any data whose Lagrange multiplier less than C and therefore $\xi_i = 0$, and then substituting this data in equation (2.41). However, it is better to take the mean value of b_o obtained from all such data.

How can the above methods be generalized to the case where the decision function is not a linear function of the training data? In [22], the kernel trick was introduced to solve such problem. There, before finding the optimal separating hyperplane, the data were first mapped to a high-dimensional space, named feature space. This mapping is called $\phi(\cdot)$. Data $\mathbf{x}$ is mapped to the feature space as $\phi(\mathbf{x})$ in Figure 2.6.

Figure 2.6: Kernel Mapping

The only way in which the training data appears in the training problem (equation (2.38)) is in the form of dot products. Hence, if there is a kernel function K , such that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$, we would only need to use K in the training process and would never need to explicitly even know what $\phi(\cdot)$ is. Some kernel functions investigated for the pattern recognition problems are:

$$k(\mathbf{x}_i, \mathbf{x}) = [\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) + 1]^d \quad (2.44)$$

$$k(\mathbf{x}_i, \mathbf{x}) = \exp\left(-\frac{\|\phi(\mathbf{x}_i) - \phi(\mathbf{x})\|^2}{2\sigma^2}\right) \quad (2.45)$$

Equation (2.44) results in a classifier which is a polynomial of degree d in the data and Equation (2.45) results in a Gaussian radial basis function classifier. After using kernel mapping, the new Lagrange objective function should be maximize in order to find the optimal separating hyperplane for the nonlinear case

$$L(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (2.46)$$

Consequently, everything that has been said about the linear case also applies to nonlinear case obtained by the kernel mapping. Hence, the new final optimal separating function is

$$f(\mathbf{x}) = \sum_{i=1}^N \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \quad (2.47)$$

2.2.2 Soft Margin Bound of SVM

The soft margin bound [162] [163] of SVM can be derived as follows:

In the linear function classes case, the probability ϵ of misclassification of a

randomly chosen test data is bounded by

$$\varepsilon \leq \tilde{O}\left(\frac{(R + \|\xi\|_2)^2}{|I|\rho^2}\right) \quad (2.48)$$

where $\tilde{O}$ notation indicating asymptotics ignoring log factors, R is the radius of a ball about the origin which contains the support of the input probability distribution, I is the training set and ρ is the margin.

While in the non-linear function classes case, if 2-norm of the slack variables are used, the bound is

$$\varepsilon \leq \tilde{O}\left(\frac{\text{fat}(\rho/16) + \|\xi\|_2^2/\rho^2}{|I|}\right) \quad (2.49)$$

where $\text{fat}(\rho)$ is the fat-shattering dimension.

2.2.3 Applications of SVM

In this subsection, some real-world problems solved by SVM are briefly introduced. In [85] [60] [56], the Reuters news stories are used for text categorization by SVM. Pontil and Verri [123] used SVM for aspect independent object recognition. However, the experiments in [123] relied only on grey level pixel information and ignored the color information which can be exploited for classification with SVM. Olivier Chapelle et al. [130] used color and luminescence information and obtained better results. SVM have been applied to many computer vision problems, such as optical pattern and object recognition [104] [31] [47] [20], 3D object recognition [123] [146] [145], time-series prediction [125] [114], face authentication [133] [124], image fusion [107] and bioinformatics [29] [77] [29] [105]. SVM has also been used to solve handwritten digit recognition problem. This problem has been tackled by Vapnik and his

co-workers [22] [42] [30] both with maximal margin and with soft margin machines, mainly with polynomial and Gaussian kernels. Furthermore, SVM was used to solve some bioinformatics problems [72] [203] [174] [139]. In [174], Jaakkola and Haussler used SVM to classify protein families into their super-family. Brown et al. [139] described a successful use of SVM applied to gene expression data for both the task of classifying genes and the equally important one of cleaning existing data.

2.2.4 The Training Time Problem of SVM

There exists a training speed problem when a SVM classifier is trained on a large number of training samples. By defining the matrix $\mathbf{Q}$ as $\mathbf{Q}_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$, the SVM maximization problem (2.25) can be rewritten as

$$\text{maximize: } W(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} \quad (2.50)$$

$$\text{subject to: } \boldsymbol{\alpha}^T \mathbf{y} = 0, \quad \mathbf{0} \leq \boldsymbol{\alpha} \quad (2.51)$$

Therefore, to train an SVM, we need to solve a quadratic optimization problem, where the number of parameters is N . Since the size of the matrix $\mathbf{Q}$ is $N \times N$, the above optimization problem depends on the square of the number of training samples. This makes the use of SVM for a large training set difficult: computing $k(\mathbf{x}_i, \mathbf{x}_j)$ for every training pair would require $O(N^2)$ computation. However, based on the optimal solution α , the decision function can also be written as

$$f(\mathbf{x}) = \sum_{\mathbf{x}_i \in I_{sv}} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \quad (2.52)$$

where I_{sv} is the set of support vectors which have nonzero α 's. It is the support vectors that determines the decision boundary and all other training samples can be removed. Hence, can we only select support vectors for training to speed up SVM? This problem will be further investigated in the later chapter and a solution is proposed there as well.

2.3 Boosting

Boosting combines a number of weak component classifier into a strong classifier. Because finding many rough rules of thumb can be a lot easier than finding a single, highly accurate prediction rule, Boosting is based on the principle of divide and conquer [79]. According to this principle, a computationally complex problem will be divided into a set of computationally simple sub-problems and then we combine the sub-solutions to form a final solution. To apply the Boosting approach, we start with a method or algorithm for finding the rough rules of thumb, which is called component learning algorithm. The Boosting algorithm calls this component learning algorithm repeatedly, each time feeding it a different distribution or weighting over the training examples. Each time it is called, the component learning algorithm generates a new component prediction rule, and after many rounds, the Boosting algorithm must combine these component rules into a single prediction rule that, hopefully, will be much more accurate than any one of the component rules. In the Boosting method, computational simplicity is achieved by distributing the learning task among a number of component classifiers.

The original idea of Boosting was introduced by Kearns and Valiant [88]. By working in Valiant's PAC (probably approximately correct) model [177], they provide an algorithm that was devised for directly converting a base learning model that performs just slightly better than randomly guessing into a strong learning model that can be arbitrarily accurate. Later, Schapire [151] provided the first provable polynomial-time Boosting algorithm in 1989. In this algorithm, simple voting was

used to evaluate the performance of the Boosting on the test samples. Specifically, a test sample is presented to the Boosting. If the first and second experts in the Boosting agree in their respective decisions, that class label is used. Otherwise, the class label discovered by the third expert is used. A year later, Freund [65] developed a much more efficient Boosting algorithm which, although optimal in a certain sense, nevertheless suffered like Schapire's algorithm from certain practical drawbacks. The first experiments with these early Boosting algorithms were carried out by Drucker, Schapire and Simard [57] on an OCR task. In 1995, AdaBoost [69] was introduced which is a significant contribution for Boosting-like method.

2.3.1 Boosting Algorithm

In this subsection, we will briefly introduce the AdaBoost algorithm. There are two fundamental questions that AdaBoost must solve: first, how should each distribution of the training samples be chosen on each round, and second, how should the base rules be combined into a single rule? For the first point, the core idea of AdaBoost, after each iteration, misclassified training sample will be adaptively given more weight in the next iteration according to whether this training sample is classified correctly or not. That means AdaBoost will subsequently focus more on the misclassified training samples. For the second point, AdaBoost gives a weight to each component classifier according to its training error and then takes a weighted majority vote to obtain the final rule. Hence, given a set of training samples, $I = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i$ is a training sample and

y_i is the label of $\mathbf{x}_i$, AdaBoost (Table 2.1) maintains a probability distribution, W , over these samples. This distribution is initially uniform. Then, AdaBoost calls ComponentLearn algorithm repeatedly in a series of iterations. At iteration t , AdaBoost provides training samples with a distribution W_t to the ComponentLearn algorithm. In response, the ComponentLearn trains a component classifier h_t . The distribution W_t is updated after each iteration according to the prediction results on the training samples. “Easy” samples that are correctly classified by the component classifier, h_t , get low weights, and “hard” samples that are misclassified get higher weights. Thus, AdaBoost focuses on the samples with more weights, which seem to be harder for ComponentLearn. This process continues for T iterations, and at last, AdaBoost uses weighted vote to combine all the obtained component classifiers into a single final classifier f . Greater weights are given to component classifiers with lower errors. The important theoretical property of AdaBoost is that if the component classifiers consistently have accuracy only slightly better than half, then the error of the final classifier drops to zero exponentially fast. This means that the component classifiers need be only slightly better than random.

The most basic theoretical property of AdaBoost concerns its ability to reduce the training error. In [153], Schapire and Singer proved that the training error of the final classifier is bounded as

$$\frac{1}{N} |\{i : f(\mathbf{x}_i) \neq y_i\}| \leq \frac{1}{N} \sum_i \exp(-y_i f(\mathbf{x}_i)) = \prod_t C_t \quad (2.53)$$

where N is the number of training samples, y_i is the label of training sample $\mathbf{x}_i$, C_t

Table 2.1: AdaBoost

Algorithm: AdaBoost [153]

1. Input: a set of training samples with labels $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$,

ComponentLearn algorithm, the number of iterations T .

2. Initialize: the weight of training samples: $w_i^1 = 1/N$, for all $i = 1, \dots, N$.

3. Do for $t = 1, \dots, T$

(1) Use ComponentLearn algorithm to train a component classifier h_t .

(2) Calculate the training error of h_t : $\epsilon_t = \sum_{i=1}^N w_i^t, y_i \neq h_t(\mathbf{x}_i)$.

(3) Set weight of component classifier h_t : $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$.

(4) Update training samples' weights: $w_i^{t+1} = \frac{w_i^t \exp\{-\alpha_t y_i h_t(\mathbf{x}_i)\}}{C_t}$

where C_t is a normalization constant, and $\sum_{i=1}^N w_i^{t+1} = 1$.

4. Output: $f(\mathbf{x}) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}))$.

is a normalization factor which is chosen so that the next weight of training samples will be normalized, the AdaBoost classifier $f(\mathbf{x})$ is

$$f(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x})\right) \quad (2.54)$$

This bound shows that if we want to decrease the training error of AdaBoost, we should only choose suitable α_t and h_t at each iteration round to minimize C_t .

In the case of binary-class classification, Freund and Schapire [69] proved that the training error bound of AdaBoost can also be written in the form

$$\varepsilon \leq \prod_{t=1}^T \sqrt{1 - 4\gamma_t^2} \leq (-2 \sum_{t=1}^T \gamma_t^2) \quad (2.55)$$

where ε is the training error, T is the number of component classifiers, γ is defined as $\gamma_t = \frac{1}{2} - \varepsilon_t$. This inequality shows that when each weak learner is slightly better than random selection, the training error will drop exponentially fast.

Later, in some experiments, it was observed that if AdaBoost was continually trained after the training error was zero, the generalization accuracy continued to improve even after the training error was zero. Then in 1998, following the work of Barlett, Schapire et al. [150] used a margin as a measure to analyze the classification performance of AdaBoost. They gave a classification bound based on the margin and proved that larger margins on the training sample set could lead to higher classification accuracy.

2.3.2 Explanation of Boosting by Margin

The reason for the success of Boosting is explained based on margin in [150]. First give the definition of margin for Boosting. The margin ρ of a sample $\mathbf{x}$ is defined as:

$$\rho(\mathbf{x}) = yf(\mathbf{x}) \quad (2.56)$$

It has been found in [150] that with probability at least $1 - \delta$ over the random choice of training set I , the AdaBoost classifier satisfied the following generalization error bound for all $\theta > 0$:

$$\mathbf{P}_{\mathcal{D}}[yf(\mathbf{x}) \leq 0] \leq \sum_{i=1}^N \mathbf{I}(\rho(\mathbf{x}_i) < \theta) + O\left(\sqrt{\frac{d \log^2(N/d)}{N\theta^2}} + \frac{\log(\frac{1}{\delta})}{N}\right) \quad (2.57)$$

This bound is based on the work of [187] [101] [137]. The idea is that the effective size of the function class is much smaller if there is a separating margin between

the classes. In [150], it was claimed that the reason for the success of AdaBoost lies in that the AdaBoost produces hypotheses with large margins on all samples, say larger than some θ . Then, the first term on the right of this bound is zero and the second term is small. The larger the θ is, the smaller the second term is. Hence, the most important quantity is the margin. The second term would be smallest if the margin is as large as possible.

2.3.3 Other Methods to Analyze Boosting

Freund and Schapire [66][68] also used game-theoretic setting to analyze the behavior of AdaBoost. They proposed an algorithm for learning to play repeated games based on the on-line prediction methods of Littlestone and Warmuth. They showed that Boosting is obtained by applying the same algorithm to the dual of this game. Leo Breiman [25] proved a bound on the generalization error for the combined component classifiers in terms of their maximum error and showed that arcing algorithms converged to the optimal strategy. The problem of solving a zero-sum game is solvable by using linear programming. Hence, Ratsch et al. [143], Demiriz et al.[50] used linear programming to solve it. In [143], Ratsch et al. observed that Boosting algorithms like AdaBoost are iterative strategies to minimize a constrained objective function, equivalent to Barrier algorithms. Hence, they introduced a new Boosting technique for regression, ε -Boost. Demiriz et al. [50] proved that for classification, minimizing the 1-norm soft margin error function directly optimize a generalization error bound and then they provided algorithms for

soft margin classification, confidence-rated, and regression Boosting problems. The optimal solutions of LPBoost [50] are very sparse in contrast with gradient based methods. Computationally, LPBoost is competitive in quality and computational cost to AdaBoost. Following up on the work by Kivinen and Warmuth [93] and Lafferty [99], Collins, Schapire and Singer [41] derived an algorithm using a unification of logistic regression and Boosting based on Bregman distances. They also gave unified proofs of convergence to optimality for a family of new and old algorithms, including AdaBoost, for both the exponential loss used by AdaBoost and the logistic loss used for logistic regression. This work further connects Boosting to the maximum-entropy literature [49]. By replacing the logistic loss function with one that incorporates prior knowledge, Schapire [152] proposed an algorithm that permits the incorporation of prior human knowledge as a means of compensating for a shortage of training data. In the later work of Lebanon and Lafferty [102], they showed logistic regression and Boosting were in fact solving the same constrained optimization problem, except that in Boosting, certain normalization constraints had been dropped.

2.3.4 Applications of Boosting

Boosting methods have been widely used in many real application problems. In [154], Schapire et al. compared AdaBoost and Rocchio for text filtering over three large text filtering tasks. Their experimental result showed that AdaBoost produced better classifiers than Rocchio when the training collection contains a very

large number of relevant documents. Boosting has been used in natural language processing [84] and call classification [148] [144]. In [84], Pedro J. Moreno et al. derived feature vectors from speech recognition lattices and feed them into a Boosting classifier. By comparing experimental result gotten from Boosting with those from SVM and Regression Trees, significant improvements were obtained. Kinh Tieu and Paul Viola [170] used AdaBoost to solve image retrieval problems. The most important application of Boosting methods in computer vision is object detection [182] [183]. Some other fields include auction [155] [135], document routing [82], cancer identification [202] [53], medical diagnosis [118], monitoring system [131], word sense disambiguation [73], document routing [82], text categorization [149] [161], character recognition [160], and customer monitoring [122].

2.3.5 The Diversity/Accuracy Dilemma in Boosing

It is known that the generalization performance of Boosting methods is affected by both component classifiers' accuracy and their diversity [193]. There exists a dilemma between component classifiers' accuracy and diversity [54]. We will use the Diversity-Accuracy diagram of AdaBoost in Figure 2.7 to explain this dilemma. This Diversity-Accuracy diagram is a scatterplot where each point corresponds to a component classifier. A point's x coordinate value is its corresponding component classifier's diversity value while y coordinate value is the classifier's accuracy value. From this figure, it can be observed that if the component classifiers are too accurate, it is difficult to next find very diverse ones, and combining these accurate but non-

diverse classifiers often leads to very limited improvement [193]. On the other hand, if the component classifiers are too inaccurate, although we can find diverse ones, the combined result may be worse than that of combining both more accurate and diverse component classifiers. This is because the combination result is dominated by too many inaccurate component classifiers, and the combination result will be wrong most of the time, leading to poor classification result. Hence, the diversity/accuracy dilemma is how we could maximize the diversity under the condition of obtaining a fairly good component classifiers accuracy for Boosting methods? This problem will be investigated in detailed and a framework to handle this dilemma will also be proposed in the following chapter.

Figure 2.7: Diversity vs Accuracy

2.4 SVM Active Learning

In some classification problems, it is difficult or very expensive to collect many labelled samples. Meanwhile, there are many unlabelled samples and we are given the opportunity to query the oracle the ground truth of the selected unlabelled samples. Active learning algorithm is designed to select the most informative unlabelled sample from the unlabelled sample pool and then query the oracle the ground truth of this sample so as to train a classifier as good as possible. SVM-based active learning algorithms have been widely investigated. A similar SVM-based active learning was independently developed [156] [32] [172]. Tong et al. [172] used version space to analyze SVM and gave an optimal selection strategy.

Image classification [128] [37] [130] [108] is an important task in computer vision. It can give much help to image indexing [140] [10] [111] [136] and retrieval tasks [199] [168] [19] [169], because both of them need correctly label images. Multi-label image classification is a case in which the image can have one or more labels. Therefore, multi-class classification can be viewed as a special case of multi-label classification. Since SVM-based active learning has achieved great success in binary classification problems, can we use it to help decrease the number of needed multi-label image training samples? The following chapters will address these issues and find solutions.

Chapter 3

Sequential Bootstrapped SVM: A SVM Training Accelerator

3.1 Literature Review of SVM Training Accelerating Methods

Support Vector Machines (SVM) [179] was developed from the theory of structural risk minimization. Due to its impressive generalization performance on classification problems, SVM has yielded much success in a variety of domains, such as image processing [90], micro-array data analysis [198] and audio classification [75]. Large data set problems are often met in real applications, such as automatic document classification. Unfortunately, when one tries to scale up the standard SVM to large data sets, certain problems arise. Since the standard SVM has to solve a quadratic optimization(QP) problem whose time complexity is $O(N^3)$ and space complexity is $O(N^2)$, where N is the number of training samples, training time and memory increase dramatically with the increase of the training set. Therefore, it is difficult to solve large training set problems using the standard SVM.

Many implementation techniques and algorithms have been developed for training SVM more efficiently and quickly. Most techniques attempt to speed up the SVM computations by dividing the original QP problem into smaller parts in order to reduce the size of each QP problem. The Chunking algorithm [42] is an iterative method that at each iteration solves a small sub-problem. It uses the support vectors found in previous batches for the use in the next batch of samples. Osuna et al. [133] uses only a subset of the samples as a working set and optimizes the problem with respect to them while freezing the others. Going to the extreme case,

the Sequential Minimum Optimization algorithm (SMO) [157] only uses two variables in each sub QP problem. Because at each iteration, the SMO only involves two variables in the optimization, this QP problem is analytic and is thus very fast. However, the convergence of the SVM is slow [87]. O. L. Mangasarian and David R. Musicant [112] proposed the first method to handle data sets with more than 10^6 data. In [112], successive overrelaxation (SOR) for symmetric linear complementarity problems and quadratic programs is used to train a SVM. Because SOR handles one point at a time, similar to SMO, SOR can process very large data sets that need not reside in memory. Hence, SOR need less memory when handling large training data set. The SimpleSVM algorithm [184] works by incrementally changing a candidate support vector set using a greedy approach. However, storage is still a problem when the SimpleSVM is applied to generic dense matrices on large noisy data sets as pointed in [184]. Although by using these decomposition methods, the training time complexity of SVM can be reduced from $O(N^3)$ to $O(N^2)$, the training time is still long when handling large number of training samples.

SVM approximation [71] attempts to improve the computational efficiency by altering the QP formulation to the extent that it keeps a similar semantic of the original SVM while it is faster to be solved by a QP solver. Mangasarian et. al [113] proposed a Lagrangian SVM (LSVM) to avoid the QP problem. However, LSVM still requires the inversion of a $N \times N$ matrix [173]. Other kinds of approaches focus on obtaining low-rank approximations on the kernel matrix, such as greedy matrix approximation [165], Nystrom method [191], sampling techniques [6] and low-rank

kernel representations [63]. However, for very large data sets, the reduced rank of kernel matrix is still too high.

Random sampling [106] uses a randomly selected subset of the whole data set to obtain a SVM separating hyperplane. An improved random sampling technique [14] is developed to reduce the training time of SVM based on the idea that the sample near the boundary with higher probabilities will be sampled. However, the true distribution of the data set may not be well represented by the sampled training samples so as to degrade the generalization performance sometimes.

Based on the fact that it is the support vectors that determine the optimal separating hyperplane, some approaches try to identify the support vectors efficiently. In [164], a fast training algorithm based on quick identification of support vectors was proposed, however, this algorithm appears to have some difficulties in dealing with linearly separable training set. In addition, Keerthi et al. [89] proposed an algorithm based on observations about the geometrical properties of SVM. Clustering-based SVM [201] is proposed to speed up SVM training time by considering the cluster information in reducing the training samples. However, the distance is measured in the feature space while the clustering is performed in the input space, which introduces some mistakes in selecting valuable training samples.

Some other related work include Online SVM [92], incremental and decremental SVM [33], and active learning with SVM [172] [189]. Online SVM, incremental and decremental SVM aim to dynamically handle incoming samples. Active Learning focuses on using unlabelled samples to help select the most informative samples for

training.

In this chapter, a sequential method named *Sequential Bootstrapped SVM* (SeqSVM) to speed up SVM is proposed. This method is motivated by the fact that all support vectors lie on the convex hull of each class distribution in the case of linearly separable classes. The SeqSVM selects the so-call SBA samples which have more probability to be support vectors than other samples, and then focused on training them to decrease the needed number of training samples. The SeqSVM method can be described as follows: At the beginning, the large training set is divided into two disjoint subsets: *TrainActual* and *TrainPool*. Then, a SVM classifier can be trained on the current *TrainActual* set. With the derived SVM, the SBA sample is selected in the *TrainPool* set that is furthest away from and on the wrong side of the current SVM hyperplane. This SBA sample is transferred to the *TrainActual* set and the SVM is retrained again. It is found that to apply this simple principle is not straight forward and it is necessary to refine it to take care of computational efficiencies and non-separable cases. The details are described in this chapter. The SeqSVM runs on top of the techniques, such as Chunking, decomposition and SMO, to handle large scale problems.

3.2 General Approach of SeqSVM

In the following several sections, general approach of proposed Sequential Bootstrapped SVM algorithm, SeqSVM for short, is introduced firstly. Next, we explain a way to find the SBA samples. These SBA samples have the most probability to be support vectors and we explain by way of version space why they induce the fastest convergence to the optimal hyperplane. Then, the outline of the SeqSVM algorithm is described in detail. Afterwards, the convergence proof of SeqSVM is given. Finally, online SVM is used to speed up the SeqSVM and the time complexity is given.

Given a large training set $I = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$, where each sample $\mathbf{x}_i \in \mathbf{R}^d$ and its label $y_i \in \{+1, -1\}$. These samples are assumed to be i.i.d. distributed according to some unknown fixed distribution $P(\mathbf{x})$, their labels are distributed according to some unknown conditional distribution $P(y|\mathbf{x})$. Since only the support vectors determine the final decision boundary of SVM, if we could train an SVM classifier only using the support vectors from the large training set, we can decrease the training time greatly. Hence, in the SeqSVM, we aim to focus on training the samples which have more probability to be support vectors and pay less attention to the samples which have less probability. Initially, the SeqSVM separates the large training set, I , into two disjoint subsets: a small initial *TrainActual* set, A_0 , and a relatively large initial *TrainPool* set, P_0 . Then, based on A_0 , an initial SVM classifier, f_0 , is trained. After that, the most informative training sample to classification, which we call SBA sample, $\mathbf{x}_c$, the one in P_0 that is furthest away from

and wrongly classified by f_0 , is selected. This sample has the most disagreement with the current SVM solution. Its transfer to A_0 will, with greatest probability, give the most drastic change in the SVM hyperplane solution. After that, a new training-selecting-transferring iteration begins. This iteration stops if no SBA sample could be found. The SVM classifier obtained in the last SeqSVM iteration is the final output classifier. Hence, by focusing on training with the SBA samples, the SeqSVM can pay more attention to the samples, which have more probability to be support vectors. This can greatly decrease the total number of needed training samples so as to decrease the SVM's training time on the whole large training set.

3.3 Version Space

In this section, we will briefly introduce the concept of version space [120], which will be used to analyze SBA samples later. It is known that by using the kernel trick [179], the SVM maps training samples from the input space, $\mathcal{I}$, to a high-dimensional feature space, $\mathcal{F}$, and then finds an optimal separating hyperplane there. In this way, the decision function of SVM can be written as¹

$$h(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) \quad (3.1)$$

where $\mathbf{w}$ is a vector in the coefficient space, $\mathcal{W}$, and $\phi(\mathbf{x})$ is a mapping of sample $\mathbf{x}$ from $\mathcal{I}$ to $\mathcal{F}$. $h(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$ represents a hyperplane in $\mathcal{F}$. A point $\mathbf{x}$ that the SVM maps to $(-\infty, 0]$ is given the class of -1 , and if the SVM maps it to $(0, +\infty)$, this point is given a class of 1 . Obviously, the dimensions of $\mathcal{F}$ and $\mathcal{W}$ are the same. The version space of SVM is explored as follows.

Given a set of training samples, $I = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$, where each sample $\mathbf{x}_i \in \mathbf{R}^d$ and its label $y_i \in \{+1, -1\}$, the version space can be viewed as a set of hypotheses, each of which correctly classifies all the training samples in I [120]. In the case of SVM, the version space, V_I , is defined as a set of SVM classifiers, h , under the condition that each h can correctly classify all the samples in I :

$$V_I = \{h | h(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}), y h(\mathbf{x}) > 0, \forall \mathbf{x} \in I, y \in \{+1, -1\}, \mathbf{w} \in \mathcal{W}\} \quad (3.2)$$

Since each h in V_I is uniquely defined by its coefficient vector, $\mathbf{w}$, V_I can be redefined based on $\mathbf{w}$, instead of h , under the condition that the corresponding classifier of

¹If a bias weight is needed, one can alter the kernel or input space to fit it.

each $\mathbf{w}$ can correctly classify all the training samples:

$$V_I = \{\mathbf{w} \mid y(\mathbf{w}^\top \phi(\mathbf{x})) > 0, \forall \mathbf{x} \in I, y \in \{+1, -1\}, \mathbf{w} \in \mathcal{W}\} \quad (3.3)$$

Similarly, we define V_{A_i} as the version space corresponding to the *TrainActual* set, A_i , formed in the i -th SeqSVM iteration:

$$V_{A_i} = \{\mathbf{w} \mid y(\mathbf{w}^\top \phi(\mathbf{x})) > 0, \forall \mathbf{x} \in A_i, y \in \{+1, -1\}, \mathbf{w} \in \mathcal{W}\} \quad (3.4)$$

Since the SVM can be written as maximizing an objective function subject to $\|\mathbf{w}\|_2 = 1$ [80], version space is a connected region on the surface of the hypersphere $\|\mathbf{w}\|_2 = 1$ in the coefficient space $\mathcal{W}$ [189].

3.4 Existence of SeqSVM's Version Space

From the definition of version space, it can be found that to ensure the existence of SeqSVM's version space, a requirement has to be taken that the *TrainActual* set is linearly separable in the feature space $\mathcal{F}$. This is because only if the *TrainActual* set is linearly separable in $\mathcal{F}$, an SVM separating hyperplane, which satisfies equation (3.2), can be found in $\mathcal{F}$. For the case of linearly separable problems, this is met. When the classes are nonlinearly separable, kernel mapping can achieve this requirement. For the case of non-separable problems, we can use the soft margin SVM with quadratic penalization of errors. This is because the soft margin SVM with quadratic penalization of errors can be considered as a special case of the hard margin version with the modified kernel [44]:

$$\mathbf{K} \leftarrow \mathbf{K} + \frac{1}{C}\mathbf{I} \quad (3.5)$$

where $\mathbf{I}$ is the identity matrix and C is a positive regularization constant. Hence, for the non-separable data distributions, by a proper transformation of kernel matrix (equation (3.5)), the data will be linearly separable in the new induced feature space. Thus, the existence of version space is guaranteed. We give an explanation of why the soft margin SVM with quadratic penalization of errors can be considered as a special case of the hard margin version with the modified kernel as follows.

Different from solving the optimization problem for the separable data distributions, which is introduced in the background part of this chapter (equation (2.17)), for non-separable data distributions, the soft margin SVM with quadratic penaliza-

tion of errors should solve the following optimization problem instead:

$$\text{minimize: } g(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i^2 \quad (3.6)$$

$$\text{subject to: } y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (3.7)$$

where, ξ_i is a slack variable and C is the regularization parameter. The quadratic penalization of errors are used here. Note that if $\xi_i < 0$, the constraint $y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i$ still hold if we set $\xi_i = 0$. This change will also reduce the value of the objective function. Hence, the constraint $\xi_i \geq 0$ can be removed:

$$\text{minimize: } g(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i^2 \quad (3.8)$$

$$\text{subject to: } y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \quad (3.9)$$

Therefore, The Lagrange (primal) function is:

$$L(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{2} \sum_{i=1}^N \xi_i^2 - \sum_{i=1}^N \alpha_i [y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) - (1 - \xi_i)] \quad (3.10)$$

where α_i is a Lagrange multiplier which corresponds to $\mathbf{x}_i$. This function will be maximized with respect to $\mathbf{w}$, b and $\boldsymbol{\xi}$. The corresponding dual objective function is found by differentiating with respect to $\mathbf{w}$, b and $\boldsymbol{\xi}$, and then substituting the obtained relations into the above primal function:

$$\text{maximize: } L(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{2C} \langle \boldsymbol{\alpha}, \boldsymbol{\alpha} \rangle \quad (3.11)$$

$$= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \left\{ k(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{C} \delta(i, j) \right\}$$

$$\text{subject to: } \sum_{i=1}^N y_i \alpha_i = 0, \quad \forall i : 0 \leq \alpha_i \leq C \quad (3.12)$$

where $\delta(i, j)$ is defined to be 1 if $i = j$ and 0 otherwise. Hence, this problem can be viewed as a hard margin SVM problem as simply a change of kernel:

$$\mathbf{K} \leftarrow \mathbf{K} + \frac{1}{C} \mathbf{I} \quad (3.13)$$

3.5 Basic Idea of SeqSVM

In this section, we will introduce the basic idea about how the proposed SeqSVM algorithm speeds up the SVM training process when training on large data sets. Let h^* be the optimal SVM classifier which is trained on the whole large training set, I . The coefficient vector of h^* is denoted as $\mathbf{w}^*$. In the i -th SeqSVM iteration, let h_i^* be the SVM classifier which is trained on the *TrainActual* set, A_i . The coefficient vector of h_i^* is denoted as $\mathbf{w}_i^*$.

Normally, the SVM training time increases with the increase of the number of training samples. Hence, the basic idea of the proposed SeqSVM algorithm is *to let the obtained classifier h_i^* trained on the TrainActual set, A_i , which is a part of I , have the similar generalization performance as the optimal classifier h^* trained on the whole training set I* . Therefore, the key point of SeqSVM is how to select the *TrainActual* set, A_i , from the whole large training set, I .

From the equation (3.1), we can see that the classifiers h^* and h_i^* can be uniquely decided by their coefficient vectors $\mathbf{w}^*$ and $\mathbf{w}_i^*$. Hence, the above basic idea equals to letting $\mathbf{w}_i^*$ approach $\mathbf{w}^*$. From the existence of SeqSVM's version space in last subsection, we know that in the i -th SeqSVM iteration, the resulting SVM classifier, h_i^* , which is trained on A_i , can guarantee zero prediction error on A_i . Hence, h_i^* lies in the version space corresponding to A_i and its coefficient vector, $\mathbf{w}_i^*$, lies in the version space, V_{A_i} . Because of the same reason, the coefficient vector of the optimal classifier, $\mathbf{w}^*$, always lies in the series of version spaces, V_{A_i} , $i = 1, 2, \dots$, formed in each SeqSVM iteration. Since both $\mathbf{w}_i^*$ and $\mathbf{w}^*$ lie in V_{A_i} , if we can let the size

of V_{A_i} be small enough, we can let $\mathbf{w}_i^*$ and $\mathbf{w}^*$ be much closer, which is the goal of proposed SeqSVM algorithm. Therefore, the key point of the SeqSVM is to transfer proper samples from the *TrainPool* set, P_i , to A_i which can reduce the size of V_{A_i} as greedily as possible. This is because by doing so, we can make $\mathbf{w}_i^*$ converge to $\mathbf{w}^*$ as fast as possible. Thus, the basic idea of the SeqSVM can be described as follows.

$$\text{training time} \downarrow \Leftarrow h_i^* \text{ approach } h^* \Leftarrow \mathbf{w}_i^* \text{ close to } \mathbf{w}^* \Leftarrow \text{size}(V_{A_i}) \downarrow \quad (3.14)$$

3.6 SBA Sample in the Version Space

In this section, we want to explain how to find the SBA samples and why SBA samples can induce the fastest convergence to the optimal SVM hyperplane by way of version space.

As mentioned before, $h(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$ can be seen as a hyperplane in the feature space, $\mathcal{F}$, where $\mathbf{w}$ is the coefficient vector and $\phi(\mathbf{x})$ is the variable. By duality, it can also be seen as a hyperplane in the coefficient space, $\mathcal{W}$, where $\phi(\mathbf{x})$ is the coefficient vector and $\mathbf{w}$ is the variable. Hence, a training sample corresponding to a point in $\mathcal{F}$, also corresponds to a hyperplane in $\mathcal{W}$, and vice versa. Recall that the version space is a connected region on the surface of the hypersphere $\|\mathbf{w}\|_2 = 1$ in the coefficient space $\mathcal{W}$ [189]. In the i -th SeqSVM iteration, after transferring a selected sample, $\mathbf{x}_s$, from the *TrainPool* set, P_i , to the *TrainActual* set, A_i , the version space V_{A_i} is partitioned into two version subspaces in $\mathcal{W}$ by the hyperplane $h(\mathbf{w}) = (\phi(\mathbf{x}_s))^\top \mathbf{w}$. This hyperplane is induced by $\mathbf{x}_s$. Since $\mathbf{w}$ s in one version subspace let $(\phi(\mathbf{x}_s))^\top \mathbf{w}$ positive, while negative in the other version subspace, only the $\mathbf{w}$ s in one version subspace can correctly classify $\mathbf{x}_s$. Furthermore, $\mathbf{w}^*$ lies in the version space V_{A_i} and can correctly classify $\mathbf{x}_s$ so that $\mathbf{w}^*$ still lies in the next version space $V_{A_{i+1}}$. Hence, after adding $\mathbf{x}_s$, $V_{A_{i+1}}$ will be the version subspace of V_{A_i} if and only if $\mathbf{w}^*$ lies in this version subspace. An illustration is given in Figure 3.1.

It is also known that $\mathbf{w}_i^*$ lies in V_{A_i} . Hence, if the corresponding hyperplane of $\mathbf{x}_s$ is farthest from $\mathbf{w}_i^*$ and after adding $\mathbf{x}_s$, $V_{A_{i+1}}$ does not include $\mathbf{w}_i^*$, the smallest

Figure 3.1: Version Space Changes from I to II by Adding the Sample $\mathbf{x}_s$

version subspace of V_{A_i} will be selected to form $V_{A_{i+1}}$ so as to decrease V_{A_i} fastest with greatest probability. An example is shown in Figure 3.2. We assume that the ‘Version Space I’(represented by the rectangle) is V_{A_i} and line **a**, **b**, **c**, **d** correspond to four training samples in the *TrainPool* set. After adding one of these samples, $V_{A_{i+1}}$ is the part of ‘Version Space I’ directed by the arrowhead corresponding to that sample. From this figure, it can be found that compared with selecting other samples, if **c** is transferred to A_i , V_{A_i} decreases greatest and $V_{A_{i+1}}$ is the smallest represented by the triangle ‘Version Space II’. This is because compared with **a**, **b** and **d**, sample **c** is farthest from $\mathbf{w}_i^*$ and after adding **c**, $V_{A_{i+1}}$ does not include $\mathbf{w}_i^*$.

Because $\|\phi(\mathbf{x})\|_2 = 1$ for the RBFSVM (SVM with the RBF kernel), which will be used in the SeqSVM, the distance of a point, $\mathbf{w}$, from the hyperplane induced by $\mathbf{x}_s$ in $\mathcal{W}$, can be calculated as

$$d_{\mathbf{x}_s}(\mathbf{w}) = \frac{|(\phi(\mathbf{x}_s))^\top \mathbf{w}|}{\|\phi(\mathbf{x}_s)\|_2} = |(\phi(\mathbf{x}_s))^\top \mathbf{w}| \quad (3.15)$$

Figure 3.2: Version Space Changes from I to II by Adding the Sample c

and the $\mathbf{w}$ whose corresponding classifier $h(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$ misclassifies $\mathbf{x}_s$, should satisfy

$$y_s \times (\mathbf{w}^\top \phi(\mathbf{x}_s)) < 0 \quad (3.16)$$

where y_s is the label of $\mathbf{x}_s$. In the i -th SeqSVM iteration, an SVM classifier, $h_i(\mathbf{x}) = (\mathbf{w}_i)^\top \phi(\mathbf{x})$, is trained on the *TrainActual* set, A_i . Then, the SeqSVM will select a sample in the *TrainPool* set, P_i . This sample is furthest away from and wrongly classified by h_i . This sample is the one with the greatest disagreement with h_i . We call it as SBA sample, $\mathbf{x}_c^i$,

$$\mathbf{x}_c^i = \arg \max_{\mathbf{x} \in P_i} d_{\mathbf{x}}(\mathbf{w}_i) = \arg \max_{\mathbf{x} \in P_i} |(\phi(\mathbf{x}))^\top \mathbf{w}_i|, \quad (3.17)$$

$$\text{subject to: } y_c^i \times ((\mathbf{w}_i)^\top \phi(\mathbf{x}_c^i)) < 0 \quad (3.18)$$

where y_c^i is the label of $\mathbf{x}_c^i$.

3.7 Proposed SeqSVM Algorithm

Hence, the proposed SeqSVM algorithm can be described as:

Table 3.1: Sequential Bootstrapped SVM

Algorithm: Sequential Bootstrapped SVM

1. Input: a large training set $I = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_N, y_N)\}$,

$\mathbf{x}_i \in \mathbf{R}^d, \forall y_i \in \{+1, -1\}$.

2. Initialize: separate I into two disjoint subsets: a relatively small *TrainActual* set, A_0 , and a fairly large *TrainPool* set, P_0 .

3. Do While (new SBA sample is selected)

(1)Train SVM: in the i -th iteration, train a soft margin RBFSVM classifier with quadratic penalization of errors, based on A_i : $h_i(\mathbf{x}) = (\mathbf{w}_i)^\top \phi(\mathbf{x})$.

(2)Search the SBA sample:

$$\mathbf{x}_c^i = \arg \max_{\mathbf{x} \in P_i} d_{\mathbf{x}}(\mathbf{w}_i), \text{ subject to: } y_c^i \times ((\mathbf{w}_i)^\top \phi(\mathbf{x}_c^i)) < 0$$

(3)Transfer $\mathbf{x}_c^i$ from P_i to A_i to form the next *TrainActual* set A_{i+1} .

4. Output: the RBFSVM classifier obtained in the last iteration:

$$h(\mathbf{x}) = (\mathbf{w}_{last})^\top \phi(\mathbf{x}).$$

3.8 Why the SeqSVM can Quickly Decrease the Version Space?

Given a large training set, in the i -th SeqSVM iteration, after training an SVM classifier on the *TrainActual* set, A_i , it is assumed that a random sample is transferred from the *TrainPool* set P_i to A_i , instead of the SBA sample. From the above analysis of version space, it is known that only when the corresponding hyperplane of this random sample crosses the current version space, V_{A_i} , such as samples **b**, **c**, **d** in Figure 3.2, this hyperplane can cut V_{A_i} into two version subspaces and one of them forms the next new version space, $V_{A_{i+1}}$. If this hyperplane does not cross V_{A_i} , which means all of **ws** in V_{A_i} can correctly classify this sample, such as **a** in Figure 3.2, after adding this sample, V_{A_i} cannot be decreased. Hence, if we want the current version space to decrease by transferring a new sample from P_i to A_i , this sample should cut the current version space, such as **b**, **c**, **d**. Furthermore, some samples' selected version subspaces are overlapped. For instance, the induced version space by **b** totally covers that by **d**. Hence, **d** is more effective for decreasing the version space than **b** and **b** could be ignored if **d** is selected. There is no need to use all the training samples to decrease the version space. Therefore, we focus on using SBA samples, which are more efficient in cutting off the version space.

3.9 Convergence Proof

Observe that the version space must strictly decrease in each SeqSVM iteration. This is because in each SeqSVM iteration, a SBA sample is transferred from the *TrainPool* set to the *TrainActual* set. This SBA sample is misclassified by current obtained SVM classifier. Furthermore, since the current obtained SVM classifier lies in current version space, this SBA sample cuts the current version space and only a part of the current version space is preserved to form the next version space. Therefore, the version space keeps decreasing during SeqSVM iterations. If the version space decreases sufficiently, the coefficient vector of the obtained SVM classifier and that of the optimal classifier will be close enough so that the obtained SVM classifier approaches the optimal SVM classifier.

Next, note that for the hard-margin SVM or the soft margin SVM with quadratic penalization of errors, after several SeqSVM iterations, if no SBA sample can be found, this means the prediction error of current obtained SVM classifier on the whole training set is zero. Furthermore, the current SVM classifier is obtained by solving the optimization problem (such as equation (2.17) or (3.6)) and therefore satisfies the Structural Risk Minimization [179]. Hence, this SVM classifier is the optimal SVM classifier we are searching. The SeqSVM would be particularly useful if the number of support vectors was comparatively small with respect to the overall size of the training data set.

3.10 Speeding Up SeqSVM by using Online SVM

Online SVM [92] trains a new SVM classifier based on the decision function of the obtained SVM classifier and a newly added training sample, $\mathbf{x}_a$, by modifying the obtained decision function instead of retraining all the samples. Online SVM can yield a fairly good SVM classifier with much less training time than the standard SVM. It calculates the $(j+1)$ -th SVM based on the j -th SVM and the newly added sample, $\mathbf{x}_a$, as:

$$\alpha_i^{j+1} = (1 - \eta)\alpha_i^j, \quad \alpha_{n+1}^{j+1} = \eta\sigma_a y_a. \quad (i = 1, 2, \dots, n) \quad (3.19)$$

where α_i^j is the i -th support vector's Lagrange multiplier of the j -th SVM classifier. Support vectors of the j -th SVM will still be the support vectors of the $(j+1)$ -th SVM, but with different Lagrange multipliers. y_a is the label of $\mathbf{x}_a$. σ_a equals 1 if $\mathbf{x}_a$ is misclassified by the j -th SVM. η is the learning rate of online SVM. Using online SVM to predict decision values need much less time than what the standard SVM needs. It can be expressed as follows by kernel expansion.

$$\begin{aligned} h^{j+1}(\mathbf{x}) &= \sum_{i=1}^{n+1} \alpha_i^{j+1} k(\mathbf{x}_i, \mathbf{x}) \\ &= \sum_{i=1}^n \alpha_i^{j+1} k(\mathbf{x}_i, \mathbf{x}) + \alpha_{n+1}^{j+1} k(\mathbf{x}_{n+1}, \mathbf{x}) \\ &= (1 - \eta)h^j(\mathbf{x}) + \eta y_a k(\mathbf{x}_{n+1}, \mathbf{x}) \end{aligned} \quad (3.20)$$

where h^j is the j -th SVM classifier obtained by online SVM.

One potential danger in applying the SeqSVM is that it should not take too many iterations to reach the optimal solution, otherwise the computational cost will become greater than that of the standard SVM. Hence, we adopt the above Online

SVM. Since Online SVM gives reasonable performance at a very fast speed, we allow it to run for several iterations during the SeqSVM training process. Then, by checking a criterion, the standard SVM is invoked to obtain accurate hyperplane. This is like a re-calibration process. The training process of proposed SeqSVM algorithm speeded up by using online SVM is shown in Figure 3.3. There, the RBFSVM is trained instead of the online SVM when the number of training online SVM, N_{online}^i , between training the i -th and the $i + 1$ -th RBFSVM satisfies $1 - (1 - \eta)^{N_{online}^i} < 0.8$. This can ensure that the obtained online SVM classifier is not far away from the RBFSVM classifier.

Figure 3.3: Flow Chart of SeqSVM Improved by Online SVM

3.11 Initialization

A good initial *TrainActual* set can not only speed up the convergence of SeqSVM so as to reduce its training time, but also avoid the generalization performance fluctuation at the beginning of the SeqSVM training process. We introduce the following methods to create the *TrainActual* set.

3.11.1 The Best Pair

As discussed in [184], an iterative scheme can be used to find the pair of training samples ($\mathbf{x}_+$, $\mathbf{x}_-$) from opposing classes closest to each other as follows (Table 3.2). However, this search need log-linear time so that it is not acceptable to be used as initial *TrainActual* set when training large training sets. An alternative method used in [184] chooses the best pair from a set of random pairs. This is based on the fact that roughly 59 pairs are enough for a pair which is better than 95% of all pairs with 0.95 probability [158]. However, for the SeqSVM, if only two training samples are used as initial *TrainActual* set, the SeqSVM need more iterations which leads to an increase of training time. Furthermore, we also observed that if only two training samples were used as the initial *TrainActual* set, the generalization performance of SeqSVM sometimes fluctuated at the beginning of its training process.

3.11.2 A Set of Training Samples

In order to avoid too many SeqSVM iterations resulting from the above best-pair method, we will use a set of training samples instead of two as the initial *TrainActual*

Table 3.2: Search Initial Pair

Algorithm: Search Initial Pair

1. Input:

A training set $I = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_N, y_N)\}$, $\mathbf{x}_i \in \mathbf{R}^d$, $\forall y_i \in \{+1, -1\}$.

2. Initialize:

Randomly select a pair of training samples $(\mathbf{x}_+, \mathbf{x}_-)$ from the two opposing classes. $\mathbf{x}_+$ has label $+1$ and $\mathbf{x}_-$ has label -1 .

3. Do While $((\mathbf{x}_+, \mathbf{x}_-)_{new} = (\mathbf{x}_+, \mathbf{x}_-)_{old})$

(1) $(\mathbf{x}_+, \mathbf{x}_-)_{old} \leftarrow (\mathbf{x}_+, \mathbf{x}_-)$

(2) $\mathbf{x}_- \leftarrow \arg \min_{\mathbf{x} \in I} distance(\mathbf{x}_+, \mathbf{x}), \forall y = -1$

(3) $\mathbf{x}_+ \leftarrow \arg \min_{\mathbf{x} \in I} distance(\mathbf{x}_-, \mathbf{x}), \forall y = +1$

(4) $(\mathbf{x}_+, \mathbf{x}_-)_{new} \leftarrow (\mathbf{x}_+, \mathbf{x}_-)$

4. Output: $(\mathbf{x}_+, \mathbf{x}_-)$

set. Furthermore, by doing so, the generalization performance fluctuation problem at the beginning of SeqSVM training process can also be solved. This is because a relatively good initial SVM classifier can be obtained on this randomly selected set of training samples. For very large data sets (more than 20,000 training samples), we randomly select one hundredth of the whole training set as the initial *TrainActual* set. For large data sets (less than 20,000 training samples), we randomly select one fiftieth of the whole training set as the initial *TrainActual* set.

3.11.3 A Set of Initial Support Vectors

Differing from the above method, this method uses preliminary support vectors as the initial *TrainActual* set. These support vectors correspond to the SVM classifier trained on the above randomly selected set of training samples. By doing so, we can obtain a fairly good initial *TrainActual* set with a relatively small size. Hence, in the following experimental part, for very large data sets (more than 20,000 training samples), we randomly select one hundredth of the whole training set to train an initial SVM classifier and then use its support vectors as the initial *TrainActual* set. For large data sets (less than 20,000 training samples), we randomly select one fiftieth of the whole training set to train an initial SVM classifier and then use its support vectors as the initial *TrainActual* set.

3.12 Time Complexity

In this subsection, we will derive the time complexity of the proposed SeqSVM algorithm when using the online SVM. Assuming that the size of the whole training set is N . When training the i -th RBFSVM, the size of the i -th *TrainActual* set is n_i and the size of its support vector set is n_i^{sv} . The total number of training RBFSVM is N_{svm} . The number of training online SVM between training the i -th and the $i + 1$ -th RBFSVM is N_{online}^i . The time complexity is calculated based on the number of kernel evaluations. It is known that the standard SVM training on N samples takes $O(N^3)$ time. The time complexity of the SeqSVM can be calculated as follows.

Training the i -th RBFSVM takes $O((n_i)^3)$ time, which of course can be speeded up by some sophisticated methods, such as SMO [157]. Predicting decision values of the *TrainPool* set by the i -th RBFSVM takes $O((N - n_i) \times n_i^{sv})$ time. The time for searching the SBA sample can be ignored compared with that of training RBFSVM and of predicting decision values of the *TrainPool* set. Updating decision values of the *TrainPool* set by online SVM takes time less than $\sum_{i=1}^{N_{svm}} O((N - n_i) \times N_{online}^i)$. Hence, the total time is:

$$\begin{aligned}
T &= \sum_{i=1}^{N_{svm}} O((n_i)^3) + \sum_{i=1}^{N_{svm}} O((N - n_i) \times n_i^{sv}) + \sum_{i=1}^{N_{svm}} O((N - n_i) \times N_{online}^i) \\
&= \sum_{i=1}^{N_{svm}} \{O((n_i)^3) + O((N - n_i) \times n_i^{sv}) + O((N - n_i) \times N_{online}^i)\} \\
&= \sum_{i=1}^{N_{svm}} \{O((n_i)^3) + O((N - n_i) \times (n_i^{sv} + N_{online}^i))\} \\
&< \sum_{i=1}^{N_{svm}} \{O((n_i)^3) + O((N - n_i) \times (n_i + N_{online}^i))\} \\
&= \sum_{i=1}^{N_{svm}} \{O((n_i)^3) + O(N \times (n_i + N_{online}^i)) - O(n_i \times N_{online}^i)\}
\end{aligned} \tag{3.21}$$

Normally, N_{online}^i is not larger than n_i , hence

$$T = \sum_{i=1}^{N_{svm}} \{O((n_i)^3) + O(N \times n_i)\} \tag{3.22}$$

From this equation, it can be found that given a large number of training samples (assuming N training samples), the training time of the SeqSVM depends on the total number of training RBF SVM, N_{svm} , and the size of each *TrainActual* set, n_i . Since the SeqSVM trains online SVM instead of directly training RBF SVM in some SeqSVM iterations, the N_{svm} is well controlled. Furthermore, because the SeqSVM focuses on training the SBA samples, which are a small part of all the training samples, n_i is much smaller than N . Hence, the training time of the SeqSVM (equation (3.22)) is smaller than that of the standard SVM.

Chapter 4

Experimental Results of

Sequential Bootstrapped SVM

4.1 General Information about Experiment

Two artificial data sets and ten benchmark data sets with different number of training samples are used in this chapter to compare the computational advantage and classification accuracy performance of proposed SeqSVM algorithm, standard SVM algorithm [179], CVM algorithm [173], and active learning with SVM algorithm [172], ASVM for short. Table 6.1 gives the general characteristics of these data sets¹.

The Gaussian kernel was used for the SVM in our experiments. The parameter values of SVM were obtained by doing 10-fold cross validation unless otherwise specified. The learning rate η used in the SeqSVM was $\frac{1}{n_i}$ as the one used in [92], where n_i is the number of samples in A_i . The value of ε used in CVM was fixed at 10^{-6} in all the experiments as reported in [173]. The initial training set of ASVM has one hundredth of the whole training set by random selection unless otherwise specified. LIBSVM version 2.71 was used for the implementation of SVM. The implementation of SeqSVM was based on LIBSVM. The CVM implementation was also adapted from the LIBSVM. The source codes of LIBSVM can be down-

¹‘king-rook-vs-king’ can be found at <http://www.ics.uci.edu/~mlearn/MLRepository.html>. ‘Adult’ data set comes from UCI Machine Learning Repository and an easy-used format of this data set can be downloaded from <http://research.microsoft.com/users/jplatt/smo.html>. ‘Splice’, ‘Image’, ‘Waveform’, ‘Twonorm’ and ‘Ringnorm’ data sets can be downloaded from <http://mlg.anu.edu.au/~raetsch/data/>. For ‘Waveform’, ‘Twonorm’ and ‘Ringnorm’ data sets, since training set and test set have the same distribution, in order to increase the number of training samples, we exchange training set and test set.

Table 4.1: General Information of Data Sets

DATASET	DIMENSION	MAX TRAINING SET SIZE	TEST SET SIZE
Checkerboard	2	10000	1000
Sincube	50	200000	1000
Splice	60	1000	2170
Image	18	1300	1010
Waveform	21	4600	400
Twonorm	20	7000	400
Ringnorm	20	7000	400
King-rook-vs-king	6	8000	747
Adult-6a	123	11221	21341
Adult-7a	123	16101	16461
Adult-8a	123	22697	9865
Adult-a	123	32562	16281

loaded from <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>, and CVM implementation can be downloaded from <http://www.cs.ust.hk/ivor/cvm.html>. Experiments were performed on a Pentium 4 2.4Ghz machine with 512MB memory running Windows 2000.

4.2 Artificial Data Set I: Checkerboard Data Set

The 4×4 checkerboard data set (Figure 4.1(a)) has been used for evaluating the large-scale SVM implementations by some previous work [113] [173]. In this subsection, we did a set of experiments on the checkerboard data set with different number of training samples, which varies from 2,000 to 10,000. The number of test samples is fixed at 2000. For the SVM, we set the same σ and C values as those used in [173]. The initial training set of ASVM has one tenth of the whole training set by random selection, because if one hundredth of the whole training set was used as initial set for ASVM, we found that the ASVM converged very slowly. The experimental results are shown in Figure 4.2, 4.3 and 4.4.

Figure 4.1: Artificial Data Sets

Figure 4.2: Test Accuracy Comparison on Checkerboard Data Sets

Figure 4.3: Training Time Comparison on Checkerboard Data Sets

Figure 4.4: Number of Support Vectors Comparison on Checkerboard Data Sets

In these figures, the SeqSVM1 means the SeqSVM algorithm with online SVM, while the SeqSVM2 means the SeqSVM algorithm without online SVM. From these figures, it can be observed that both SeqSVM1 and SeqSVM2 are as accurate as the standard SVM and CVM algorithms, while better than ASVM. Besides, the SeqSVM1 is much faster than the standard SVM and the ASVM. The training speed of the SeqSVM1 is comparable with that of the CVM. The SeqSVM2 is slower than the SeqSVM1 and CVM which can demonstrate the effectiveness of using online SVM in the SeqSVM. Since the training speed of ASVM normally is at least 100 times slower than that of SeqSVM, it is not drawn on the Figure 4.3. The number of support vectors produced by the both SeqSVM1 and SeqSVM2 are comparable with that of the CVM and much less than that of the standard SVM, which implies faster testing. The training time of the ASVM is much longer than that of others

because in each active learning iteration, predicting all the unlabelled samples is too time-consuming for ASVM. However, for the SeqSVM1, the online SVM is used to solve such problem and therefore effectively speeds up the training process while maintaining a good performance.

4.3 Artificial Data Set II: Sine Cube Data Set

The second artificial data set is nonlinearly separable in the first two dimensions with a multiple sine-wave liked margin, and other dimensions are randomly generated in a unit cube. The data dimension is 50. Figure 4.1(b) shows the data distribution in the first and second dimensions. Figure 4.5, 4.6 and 4.7 show the performance comparison between SeqSVM1, SeqSVM2, SVM, CVM and ASVM on different numbers of training samples from 500 to 200,000. Of course, this problem does not need so many training samples, but it is convenient for illustrating the scaling properties. From Figure 4.5, it can be observed that the generalization accuracy of the SeqSVM1 and SeqSVM2 are comparable with that of the standard SVM. The generalization accuracy of CVM is less than those of the SeqSVM1 and SeqSVM2 and increases very slowly with respect to the increase of the training size. When the training size reaches 200,000, the generalization accuracy of the CVM is even less than that of SeqSVM1 up to about 3%. The training speed of the SeqSVM outperforms that of the standard SVM increasingly with respect to the increase of the size of training sets. The reason why the training time of the standard SVM is not shown when the training set is larger than 50,000 is because it is too large. Furthermore, the number of support vectors of the SeqSVM is again less than that of the standard SVM. Since the training speed of ASVM normally is at least 10 times slower than that of SeqSVM, it is not drawn on the Figure 4.6. The results of ASVM on data sets with more than 10,000 training samples are not shown because they are too time-consuming. It can also be seen that SeqSVM1 is much faster

than SeqSVM2 which again demonstrates the effectiveness of using online SVM in SeqSVM.

Figure 4.5: Test Accuracy Comparison on Sine Cube Data Sets

Figure 4.6: Training Time Comparison on Sine Cube Data Sets

Figure 4.7: Number of Support Vectors Comparison on Sine Cube Data Sets

4.4 Benchmark Data Sets

In this subsection, we will use 10 benchmark data sets to further evaluate the performance of proposed SeqSVM algorithm comparing with other three algorithms: SVM, CVM and ASVM. Since the ‘Twonorm’ and ‘Ringnorm’ data sets are very easy for classification, we used one thousand-th of whole training set as initial training set for ASVM instead of one hundredth in the default setting. The results are shown in Table 4.2 and Table 4.3.

Although the test accuracy of the SeqSVM is slightly worse than that of the standard SVM on the ‘Adult’ data sets (while comparable on other data sets), the training process of the SeqSVM is much faster than that of the standard SVM there. Actually, the training process of the SeqSVM is faster than that of the standard SVM on all the data sets. Furthermore, again, the numbers of support vectors of the SeqSVM are still far less than those of the standard SVM on the ‘Adult’ data sets. This leads to a much faster testing process. On other data sets except the ‘Adult’ data sets, the test accuracy of the SeqSVM is comparable with that of the standard SVM, but with a less training time. The test accuracy of the SeqSVM is comparable with that of the CVM on the ‘Adult’ data sets. However, the training process of the SeqSVM is much faster than that of the CVM and furthermore, the number of support vectors of the SeqSVM is much less than that of the CVM on the ‘Adult’ data sets. The training time of the ASVM on the ‘King-rook-vs-king’ data set is too long so that we do not report it.

Table 4.2: Comparison on Benchmark Data Sets

DATASET		SeqSVM	SVM	CVM	ASVM
Splice	Test accuracy	89.43%	89.12%	88.94%	88.78%
	Training time (sec.)	1.75	1.89	7.81	259.83
	# of support vectors	475.2	691.7	666.5	508.2
Image	Test accuracy	96.98%	97.04%	97.01%	96.53%
	Training time (sec.)	1.87	3.03	1.84	206.53
	# of support vectors	166.5	163.9	179.3	257.6
Waveform	Test accuracy	90.75%	91.03%	90.95%	89.87%
	Training time (sec.)	5.40	7.82	19.52	187.91
	# of support vectors	129.6	1178.4	109.1	247.3
Twonorm	Test accuracy	97.55%	97.47%	96.02%	97.03%
	Training time (sec.)	6.31	9.29	13.47	67.58
	# of support vectors	56.3	1172.3	769.4	77.9
Ringnorm	Test accuracy	98.22%	98.45%	97.73%	97.96%
	Training time (sec.)	4.67	8.63	7.27	45.13
	# of support vectors	52.6	1121.6	472.8	72.7

Table 4.3: Comparison on Benchmark Data Sets (continue)

DATASET		SeqSVM	SVM	CVM	ASVM
King-rook-vs-king	Test accuracy	92.97%	93.08%	92.53%	-
	Training time (sec.)	78.55	122.18	156.32	-
	# of support vectors	2802.1	2998.7	2783.1	-
Adult-6a	Test accuracy	83.67%	84.55%	83.51%	83.24%
	Training time (sec.)	87.54	152.74	351.22	4434.1
	# of support vectors	651.4	4995.3	5187.5	731.8
Adult-7a	Test accuracy	83.71%	84.80%	83.50%	83.52%
	Training time (sec.)	104.24	273.63	507.41	9013.3
	# of support vectors	701.3	6951.8	7061.3	832.9
Adult-8a	Test accuracy	84.07%	85.03%	84.22%	83.65%
	Training time (sec.)	183.94	533.12	701.52	24496.3
	# of support vectors	874.2	8481.3	8624.1	925.8
Adult-a	Test accuracy	84.29%	85.20%	84.37%	83.86%
	Training time (sec.)	319.82	1051.23	1835.6	57849.4
	# of support vectors	1005.3	12057.2	14010.4	1254.4

4.5 Conclusion

An algorithm, named Sequential Bootstrapped SVM, to speed up the training process of SVM, is presented. We proposed to select the so-call SBA samples which have more probability to be support vectors than other samples and then focus on training SVM classifier based on these samples to decrease the needed number of training samples. The advantage of SeqSVM is experimentally demonstrated on both artificial data sets and benchmark data sets in terms of training time. We claim that the strength of SeqSVM lies in reducing the needed number of training samples so as to speed up the training process of SVM on large data sets.

Chapter 5

BoostingDA Framework: Handling the Diversity/Accuracy Dilemma in Boosting Methods

5.1 Introduction

One of the major developments in machine learning in the past decade is the Ensemble method, which finds a highly accurate classifier by combining many moderately accurate component classifiers. Bagging [24] and Boosting [153] are the two most successful techniques for constructing Ensemble classifiers. *Diversity* is known to be an important factor in affecting the generalization performance of Ensemble classifiers [98]. When each component classifier is moderately accurate and these component classifiers disagree with each other, the uncorrelated errors among them will be removed by the voting process so as to achieve good ensemble results [27]. Therefore, diverse ensembles seem to have a better potential for improving the generalization performance than non-diverse ensembles. It was found that Boosting can produce more diverse component classifiers than Bagging [97]. This is because Boosting works by inducing diversity while the creation of the diversity of Bagging solely relies on the natural diversity of independent bootstrap samples. It is known that the generalization performance of a Boosting classifier is affected by both accuracy and diversity of component classifiers [193]. In practical design of Ensemble classifiers, there is a dilemma between the accuracy and diversity of component classifiers [54]: How can we maximize the diversity under the condition of obtaining a fairly good accuracy of component classifiers for Boosting methods?

Several methods have been proposed to consider this dilemma. In [4], the Neural Network ensembles are proposed to improve the generalization accuracy. The ambiguity is defined as the variation of the output of ensemble members averaged over

unlabelled data and it is used to give an estimate of the ensemble generalization error. The Genetic Ensemble Feature Selection algorithm [132] promoted diversity among the component classifiers by using different feature subsets. It trained neural networks component classifiers on these feature subsets and then used a genetic algorithm with a diversity and accuracy combination fitness function to select component classifiers. But this method is time-consuming and difficult to do parameter tuning. The Negative Correlation Learning algorithm [28] combined a set of neural networks with the same models by incorporating diversity and accuracy into the backpropagation weight updates of neural networks. But it is only suitable for neural networks. In [81], the Ensemble classifier was the combination of two selected clusters of decision trees. Each decision tree was trained on a bootstrapped training set. One cluster had highest average accuracy and the other cluster had the largest diversity between the first cluster. However, this method cannot guarantee a good diversity between decision trees in the same cluster and its measure of distance of two cluster is only suitable for decision tree. Since each of the above method is only concerned for a certain type of component classifier to handle the dilemma, a general framework which is applicable to different types of component classifiers is desired to design and analyze Boosting methods more conveniently and effectively when dealing with the diversity/accuracy dilemma.

The classification result of a classifier is affected by two factors: the training data and this classifier's learning ability. However, all of the afore-mentioned methods did not consider the diversity resulting from using component classifiers with

different learning abilities. Instead, they only used those having same learning abilities as component classifiers. If the diversity of candidate component classifiers is constrained into a small scale, we cannot ensure that the component classifiers with satisfactory diversity level can be really found. To get around this problem, *an effective diversity augmentation method for candidate component classifiers is needed, especially from the perspective of using component classifiers with different learning abilities.* In this chapter, a new diversity augmentation methodology is proposed by purposely expanding the scale of learning ability over candidate component classifiers. By doing so, it can offer more suitable candidate component classifiers with larger diversity to Boosting methods. Then, based on this methodology, a general framework, BoostingDA, is proposed to deal with the diversity/accuracy dilemma. Next, two realization of proposed framework, named as BoostingDA-SVM and BoostingDA-LDA, are given to handle binary-class and multi-class classification problems respectively. Finally, experimental results on machine learning benchmark data sets demonstrate that the BoostingDA-SVM can effectively improve the diversity and achieve better generalization performance than Bagging and AdaBoost. The BoostingDA-SVM also outperforms SVM on unbalanced data sets. For proposed BoostingDA-LDA algorithm, better performance than other algorithms dealing with the face recognition problem has also been obtained on several face recognition databases.

5.2 Diversity Augmentation Methodology

As mentioned before, the most difficulty in handling the diversity/accuracy dilemma lies in how we could maximize the diversity under the condition of obtaining fairly good accuracy of component classifiers? In order to offer more component classifiers with large diversity and fairly good accuracy to Boosting, we proposed a diversity augmentation methodology from the perspective of learning ability of classifier. It is achieved by purposely expanding the scale of learning ability over candidate component classifiers: Given a training set,

$$I = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\} \quad (5.1)$$

where $\mathbf{x}_i$ is a training sample and y_i is the label of $\mathbf{x}_i$, instead of using candidate component classifier with fixed learning ability, proposed diversity augmentation methodology will use a candidate component classifier set, G , including a number of candidate component classifiers with different learning abilities, to construct a Boosting classifier:

$$G = \{CL_1, \dots, CL_i, \dots, CL_m\} \quad (5.2)$$

where CL_i is the i -th candidate component classifier and G is designed to satisfy that any two classifiers in G have different learning abilities. By doing so, we can further explore the diversity among candidate component classifiers from a new perspective.

5.3 Proposed BoostingDA Framework

The proposed BoostingDA framework works as follows: Firstly, by training the classifiers from the candidate component classifier set, G , a candidate component classifier set, $H = \{h_1, h_2, \dots\}$, can be constructed over the training set, I . Then, define a diversity-accuracy evaluation function, E , for judging the contribution of a candidate component classifier, h , to the final Boosting classifier:

$$E(h) = E(A(h), D(h), \theta_E), \quad h \in H \quad (5.3)$$

where E is a function of $A(h)$, $D(h)$ and θ_E ; $A(h)$ is a measure of the accuracy of h ; $D(h)$ is a measure of the diversity of h ; and θ_E is the parameter set of E . In the BoostingDA framework, the h^* which can maximize the value of diversity-accuracy evaluation function is selected at each Boosting iteration:

$$h^* = \arg \max_{h \in H} E(A(h), D(h), \theta_E) \quad (5.4)$$

The final output classifier of BoostingDA is a fusion of those h^* s obtained at each iteration. Therefore, the BoostingDA framework works as follows (Table 5.1): Firstly, a training sample set, I_s , is created from I by sub-sampling or weighted sampling techniques. Secondly, a candidate component classifier set, $H = \{h_1, h_2, \dots\}$, is trained by using algorithms in G on the training set I_s . Thirdly, a diversity-accuracy evaluation value, $E(h_i)$, is calculated for each h_i , where $h_i \in H$. Fourthly, the classifier in H with maximal evaluation value is selected as a component classifier of the final Boosting. Afterwards, a new I_s is created and a new iteration begins until a given stop criterion, S , is satisfied. The final BoostingDA output, f , is a

combination of selected component classifiers by using a fusion method F .

Table 5.1: Proposed BoostingDA Framework

Framework: BoostingDA

1. Input: a training set, $I = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$; a candidate component classifier set with different learning abilities, $G = \{CL_1, \dots, CL_m\}$; a stop criteria, S ; a measure of accuracy, A ; a measure of diversity, D ; a diversity-accuracy evaluation function, E ;

2. Do while(S is not satisfied)

- (1) Create training sample set I_s from I by sub-sampling or weighted sampling.
- (2) Train a set of candidate component classifiers on I_s by G : $H = \{h_1, h_2, \dots\}$.
- (3) Compute diversity-accuracy evaluation: $E(h_i) = E(A(h_i), D(h_i), \theta_E), h_i \in H$.
- (4) Select a component classifier: $h_t^* = \arg \max E(h_i), h_i \in H$.

3. Output: $f(\mathbf{x}) = F(h_1^*, h_2^*, \dots, h_T^*)$.

5.4 Realization of Proposed Framework

The proposed diversity augmentation methodology works by purposely expanding the scale of learning ability over candidate component classifiers. This methodology only gives us a principle about how to design component classifiers. However, it depends on which kind of component classifier is used. For different kinds of component classifiers, we may use different methods to realize this methodology. Hence, when using this methodology, the way to achieve different learning abilities among component classifiers is the most challenging. In this section, we will give two examples about how to obtain different learning abilities among component classifiers. One example is based on SVM component classifiers for solving binary-class classification problems and the other is based on LDA component classifiers for solving multi-class classification problems.

5.4.1 Realization I: BoostingDA-SVM

It is known that the RBFSVM (SVM with the Radial Basis Function kernel) with different hyper-parameters C and σ (where C is a penalty parameter and σ is the Gaussian width) can show different learning abilities. Furthermore, although the RBFSVM cannot learn well when a very low value of parameter C is used, its performance largely depends on the value of parameter σ if a roughly suitable C is given [175]. For a given suitable parameter C , the RBFSVM can demonstrate different learning ability from very weak level to very strong level by simply adjusting its parameter σ : in a certain range, a larger σ often gets the learning model complex-

ity reduced and lowers down the classification performance; also, a smaller σ often increases the learning complexity and leads to higher classification performance in general. An example is shown in Figure 5.1, which plots the test error of SVM against the values of C and σ on a non-separable data set used in [15]. Clearly, changing σ value leads to larger variation on test error than changing C . This means that, over a large range of C , the performance of RBFSVM can be adjusted by simply changing the value of σ .

Figure 5.1: The Test Error of SVM vs. σ and C Values

Hence, in the first realization of proposed BoostingDA framework, we will use the RBFSVM as component classifier and we call this realization as BoostingDA-SVM. The advantage of using RBFSVM as component classifier lies in that the learning ability of RBFSVM is easy to control, which is achieved by only adjusting the σ .

Therefore, in the BoostingDA-SVM, we use the RBFSVM with different hyper-parameters to obtain a set of component classifiers with different learning abilities.

The candidate component classifier of BoostingDA-SVM, CL_i , is a RBFSVM classifier with hyper-parameters C and σ_i :

$$CL_i \doteq RBF SVM(C, \sigma_i) \quad (5.5)$$

As the AdaBoost, proposed BoostingDA-SVM algorithm trains a set of SVM component classifiers on the weighted training sample set. Hence, the accuracy of a component classifier is a weighted training accuracy:

$$A(h_t) = \sum_{i=1}^N a_i, \quad \forall a_i = \begin{cases} 0 & \text{if } y_i \neq h_t(\mathbf{x}_i) \\ w_i^t & \text{if } y_i = h_t(\mathbf{x}_i) \end{cases} \quad (5.6)$$

where w_i^t is the weight of training sample $\mathbf{x}_i$ when training classifier h_t .

In the realization of proposed framework, we mainly aim to show the effectiveness of this framework, instead of finding the most suitable diversity measure. Hence, we just take one kind of diversity measure [117] without any bias. However, our framework is open for other kinds of diversity measures. In the BoostingDA-SVM, The diversity of the t -th classifier h_t is defined as follows: If $h_t(\mathbf{x}_i)$ is the prediction label of h_t on sample $\mathbf{x}_i$, and $f(\mathbf{x}_i)$ is the combined prediction label of all the existing classifiers, the diversity of h_t is defined as:

$$D(h_t) = \frac{1}{N} \sum_{i=1}^N d_t(\mathbf{x}_i), \quad \forall d_t(\mathbf{x}_i) = \begin{cases} 0 & : \text{if } h_t(\mathbf{x}_i) = f(\mathbf{x}_i) \\ 1 & : \text{if } h_t(\mathbf{x}_i) \neq f(\mathbf{x}_i) \end{cases} \quad (5.7)$$

The diversity-accuracy evaluation function of BoostingDA-SVM is defined as a weighted linear combination of diversity and accuracy, although more complex function can be used. The diversity-accuracy evaluation value of h_t is:

$$E(h_t) = (1 - w_{div}) \times A(h_t) + w_{div} \times D(h_t) \quad (5.8)$$

where w_{div} is the weight of diversity, ranging from 0 to 1. w_{div} is used to balance the diversity and accuracy in this realization. Hence, the BoostingDA-SVM algorithm is showed in Table 5.2.

Table 5.2: Realization I for Solving Binary-class Classification Problems

Realization I: BoostingDA-SVM

1. Input: training set $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$; the initial σ , σ_{ini} ; the minimal σ , σ_{min} ; the step of σ , σ_{step} ; the weight of diversity w_{div} ; the threshold Υ .

2. Initialize: the weights of training samples: $w_i^1 = 1/N$, $i = 1, \dots, N$.

3. Do While ($\sigma > \sigma_{min}$)

(1)Train a RBFSVM classifier, h_t , on the weighted training set.

(2)Calculate the accuracy of h_t : $A(h_t) = \sum_{i=1}^N a_i$, see eqn (5.6).

(3)Calculate the diversity of h_t : $D(h_t) = \frac{1}{N} \sum_{i=1}^N d_t(\mathbf{x}_i)$, see eqn (5.7).

(4)Compute the diversity-accuracy evaluation value of h_t :

$$E(h_t) = (1 - w_{div}) \times A(h_t) + w_{div} \times D(h_t).$$

(5)If: ($E(h_t) < \Upsilon$) or $A(h_t) < 50\%$, Then: $\sigma = \sigma - \sigma_{step}$, goto (1).

(6)Set the weight of h_t : $\alpha_t = \frac{1}{2} \ln\left(\frac{A(h_t)}{1-A(h_t)}\right)$.

(7)Update the weights of training samples: $w_i^{t+1} = \frac{w_i^t \exp\{-\alpha_t y_i h_t(\mathbf{x}_i)\}}{C_t}$

$$(C_t \text{ is a normalization constant, such that } \sum_{i=1}^N w_i^{t+1} = 1.)$$

4. Output: $f(\mathbf{x}) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}))$.

This algorithm can be described as follows: Firstly, a large value is set to σ , which corresponds to a RBFSVM classifier with very weak learning ability. Then, the

RBFSVM with this σ will be selected as a component classifier of the BoostingDA-SVM as long as the diversity-accuracy evaluation function value of this classifier is larger than a predefined threshold Υ . Otherwise, this σ value is decreased slightly in order to obtain a new RBFSVM classifier with a higher learning ability, and a RBFSVM with the new σ is checked. This process repeats until the σ reaches a given minimal σ value. The final BoostingDA-SVM is a weighted combination of selected RBFSVM component classifiers.

5.4.2 Realization II: BoostingDA-LDA

In this subsection, we will use LDA as component classifier to realize proposed framework. LDA is a well-known scheme for feature extraction and dimension reduction. It has been widely used in many applications such as face recognition [17], image retrieval [168], etc. The reason why we chose LDA as component classifier is because its good performance on multi-class classification problems, especially on the face recognition problem. Recently, many LDA-related algorithms have been proposed to handle the face recognition problem [17] [110] [197] [34]. Among these LDA extensions, the two-stage LDA (PCA+LDA) has received a lot of attention [17]. In the two-stage LDA, an intermediate PCA step is implemented before the LDA step and then LDA is performed in the PCA subspace. Since a PCA subspace with different dimensions includes different principal component information, the LDA classifiers trained in the PCA subspace with different number of dimensions can show different learning abilities. An example is shown in Figure 5.2, where there

are 4 training samples for each person in ORL face recognition database. From this figure, it can be clearly observed that LDA trained in the PCA subspace with different dimensions can show different recognition accuracy. Since these LDA are trained on the same training samples, the different recognition accuracy is obviously resulted from different learning abilities among LDAs. Hence, in the second realization of the proposed BoostingDA framework, LDA is used as component learning algorithm and the different learning abilities among LDA component classifiers are obtained by training LDA classifiers in the PCA subspace with different dimensions. We call this realization as BoostingDA-LDA. By training a set of LDA classifiers with different learning abilities, larger instability and diversity are expected to obtain.

Figure 5.2: PCA Dimension vs Recognition Accuracy

Therefore, in the BoostingDA-LDA, we use LDA classifier trained in PCA subspace with different dimensions to obtain a set of component classifiers with different learning abilities:

$$CL_i \doteq LDA(\theta) \quad (5.9)$$

where θ is the number of dimension of the PCA subspace on which a LDA classifier is trained. As in the AdaBoost.M1 [67], the BoostingDA-LDA algorithm trains a set of LDA component classifiers on the weighted training set. Hence, the accuracy of a component classifier, h_t , is a weighted training accuracy. The diversity calculation method and the diversity-accuracy evaluation function are the same as those used in the proposed BoostingDA-SVM algorithm.

For a multi-class classification problem, given a set of training samples with labels, $I = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$, where label $y_i \in \{1, 2, \dots, m\}$, proposed BoostingDA-LDA algorithm is described as follows (Table 5.3)¹: Firstly, by using Boosting by subsampling technique [79], a training set, I_s , is sampled from I according to the weight distribution of training samples, where the number of samples in I_s is N_s . Secondly, a PCA transformation on the I_s is done to obtain $N_s - 1$ eigenvectors corresponding to the top $N_s - 1$ eigenvalues. Thirdly, a set of LDA candidate component classifiers, $H = \{h_{t1}, h_{t2}, \dots\}$, is trained in the PCA subspace with the top $n_{ini} + i$ dimensions on I_s , where $i = 1, 2, \dots, N_s - 1 - n_{ini}$ and n_{ini} is the number of initial sub-features corresponding to the top n_{ini} eigenvalues obtained by previous PCA transformation. Fourthly, the diversity-accuracy evaluation value of each LDA candidate component classifier in H is calculated and the one with largest diversity-accuracy evaluation value is selected as a component classifier of the final BoostingDA-LDA classifier. Afterwards, a new I_s is created and a new iteration begins. The final BoostingDA-LDA classifier is a weighted combination of

¹We define $\lceil \Psi \rceil = 1$ if Ψ holds and 0 otherwise.

the selected LDA component classifiers.

Table 5.3: Realization II for Solving Multi-class Classification Problems

Realization II: BoostingDA-LDA

1. Input: a multi-class training set I ; the number of initial sub-features, n_{ini} ;

the diversity weight w_{div} ; the number of iterations T .

2. Initialize: Set the weights of training samples: $w_i^1 = 1/N$, $i = 1, \dots, N$.

3. Do for $t = 1, 2, \dots, T$

(1)Sample a sample set I_s from I according to the weight distribution of training samples, the number of samples in I_s is N_s .

(2)Do PCA transformation on the I_s and obtain $N_s - 1$ eigenvectors corresponding to the top $N_s - 1$ eigenvalues.

(3)Train LDA candidate component classifiers, $H = \{h_{t1}, h_{t2}, \dots\}$, in PCA subspace with top $n_{ini} + i$ dimensions on I_s , where $i = 1, \dots, N_s - 1 - n_{ini}$.

(4)Calculate accuracy of h_{tj} : $A(h_{tj}) = \sum_{i=1}^N w_i^t [h_{tj}(\mathbf{x}_i) = y_i]$, $h_{tj} \in H$.

(5)Calculate diversity of h_{tj} : $D(h_{tj}) = \sum_{i=1}^N d_t(\mathbf{x}_i)$, $h_{tj} \in H$, see eqn (5.7).

(6)Calculate diversity-accuracy evaluation value of h_{tj} :

$$E(h_{tj}) = (1 - w_{div}) \times A(h_{tj}) + w_{div} \times D(h_{tj}), h_{tj} \in H.$$

(7)Select component classifier for the BoostingDA-LDA:

$$h_t = \arg \max E(h_{tj}), h_{tj} \in H \text{ and } A(h_{tj}) > 50\%.$$

(8)Update the weights of training samples: $w_i^{t+1} = \frac{w_i^t (\frac{1-A(h_t)}{A(h_t)})^{1-[h_t(\mathbf{x}_i) \neq y_i]}}{C_t}$

4. Output: $f(\mathbf{x}) = \arg \max_{y \in Y} \sum_{t=1}^T (\log \frac{A(h_t)}{1-A(h_t)}) [h_t(\mathbf{x}) = y]$.

Chapter 6

Experimental Results of BoostingDA

6.1 General Information about Experiment

In the following several sections, firstly, the effectiveness of the proposed diversity augmentation methodology is demonstrated on a large scale of benchmark data sets. Then, the proposed BoostingDA-SVM algorithm is compared with three state-of-the-art classification algorithms, Bagging, AdaBoost and SVM, on these benchmark data sets. Next, the BoostingDA-SVM is further compared with SVM algorithm on several unbalanced data sets. Finally, comparison between proposed BoostingDA-LDA algorithm and several popular face recognition algorithms are given on two famous face recognition databases.

The 13 benchmark data sets came from UCI Repository, DELVE, and STAT-LOG. The dimensions of these data sets range from 2 to 60, the numbers of training samples range from 140 to 1300, and the numbers of test samples range from 75 to 7000. More information can be found in Table 6.1 and data sets are available at: <http://mlg.anu.edu.au/~raetsch/data>. For each data set, 100 partitions were generated into training and test set (mostly $\cong 60\% : 40\%$). On each partition, for each algorithm, a classifier was trained and then test error was calculated based on this classifier. Hence, the final performance of each type of algorithm on a data set is its average performance over these 100 partitions of this data set.

Table 6.1: General Information of Data Sets.

DATASET	DIMENSION	#TRAIN	#TEST
Banana	2	400	4900
B. Cancer	9	200	77
Diabetes	8	468	300
German	20	700	300
Heart	13	170	100
Image	18	1300	1010
Ringnorm	20	400	7000
F. Solar	9	666	400
Splice	60	1000	2170
Thyroid	5	140	75
Titanic	3	150	2051
Twonorm	20	400	7000
Waveform	21	400	4600

6.2 Test Diversity Augmentation Methodology

In this section, we will test whether the proposed diversity augmentation methodology can increase the diversity and then lead to an improved generalization performance. The proposed diversity augmentation methodology works by purposely expanding the scale of learning ability over candidate component classifiers. Therefore, we compared proposed BoostingDA-SVM with three deliberately altered versions of BoostingDA-SVM for illustration. The parameter σ of these three altered versions are constrained into the ranges smaller than that used by BoostingDA-SVM. By doing so, the scales of learning abilities of candidate component classifiers in these three altered versions are smaller than that in the BoostingDA-SVM. We call these three altered versions as BDA-SVM- σ_{small} , BDA-SVM- σ_{middle} , and BDA-SVM- σ_{large} (Table 6.2). They have the same parameter settings as the BoostingDA-SVM except that the ranges of σ are set to be $[0.3\sigma^*, 0.1\sigma^*]$, $[1.2\sigma^*, 0.8\sigma^*]$ and $[4\sigma^*, 3.5\sigma^*]$, respectively. The value of σ^* was obtained by using 5-fold cross validation. The value of w_{div} was fixed as 0.3 for all the four methods. It is known that the parameter σ mainly affects the performance of RBFSVM component classifier compared with the other parameter C [175]. Hence, in the following experiments, the C is set as a value within 10 to 100 for the four methods. The σ_{min} is set as the average minimal distance between any two training samples and the σ_{ini} is set as the scatter radius of the training samples in the input space for the BoostingDA-SVM. Although the value of σ_{step} can affect the number of learning iteration of BoostingDA-SVM, it has less influence on the final generalization performance. Therefore, σ_{step} is set as

a value within 1 to 3 by doing cross validation. The threshold Υ is set as 0.5. In Table 6.2, we give the obtained diversity values.

Table 6.2: Comparison of Diversity Values, Generalization Errors with Standard Deviation Are in the Parenthesis in %. w_{div} Are Fixed As 0.3.

DATASET	BDA-SVM- σ_{small} $\sigma \in [0.1\sigma^*, 0.3\sigma^*]$	BDA-SVM- σ_{middle} $\sigma \in [0.8\sigma^*, 1.2\sigma^*]$	BDA-SVM- σ_{large} $\sigma \in [3.5\sigma^*, 4\sigma^*]$	BDA-SVM $\sigma \in [\sigma_{min}, \sigma_{ini}]$
Banana	0.38(16.2 $\pm$ 2.1)	0.41(14.1 $\pm$ 2.2)	0.48(15.4 $\pm$ 2.3)	0.58(11.7 $\pm$ 1.4)
B. Cancer	0.44(31.4 $\pm$ 6.0)	0.45(29.3 $\pm$ 5.7)	0.45(30.8 $\pm$ 5.4)	0.67(25.2 $\pm$ 4.7)
Diabetes	0.42(29.1 $\pm$ 2.3)	0.43(27.5 $\pm$ 2.5)	0.47(27.1 $\pm$ 2.1)	0.63(24.1 $\pm$ 2.1)
German	0.40(25.7 $\pm$ 2.7)	0.40(26.7 $\pm$ 2.5)	0.42(25.3 $\pm$ 2.5)	0.54(22.6 $\pm$ 2.1)
Heart	0.39(20.2 $\pm$ 3.8)	0.38(18.3 $\pm$ 3.8)	0.45(21.9 $\pm$ 3.2)	0.65(15.4 $\pm$ 3.1)
Image	0.45(5.8 $\pm$ 1.3)	0.44(6.2 $\pm$ 1.1)	0.49(5.3 $\pm$ 1.1)	0.59(2.6 $\pm$ 0.6)
Ringnorm	0.44(7.6 $\pm$ 0.4)	0.45(5.9 $\pm$ 0.9)	0.46(7.4 $\pm$ 0.6)	0.50(1.9 $\pm$ 0.3)
F. Solar	0.41(36.2 $\pm$ 2.6)	0.42(35.4 $\pm$ 3.7)	0.44(35.2 $\pm$ 2.5)	0.66(32.6 $\pm$ 1.8)
Splice	0.41(18.4 $\pm$ 1.3)	0.45(14.8 $\pm$ 1.8)	0.48(16.1 $\pm$ 1.2)	0.65(11.0 $\pm$ 1.1)
Thyroid	0.43(9.6 $\pm$ 2.4)	0.46(8.2 $\pm$ 2.9)	0.47(7.3 $\pm$ 2.7)	0.57(4.0 $\pm$ 2.2)
Titanic	0.42(25.7 $\pm$ 1.9)	0.44(24.9 $\pm$ 2.5)	0.45(26.4 $\pm$ 1.5)	0.57(22.3 $\pm$ 1.5)
Twonorm	0.43(8.5 $\pm$ 1.1)	0.45(7.2 $\pm$ 0.8)	0.46(9.9 $\pm$ 1.1)	0.60(3.2 $\pm$ 0.5)
Waveform	0.46(13.8 $\pm$ 1.7)	0.47(14.7 $\pm$ 1.4)	0.42(12.8 $\pm$ 1.2)	0.64(10.3 $\pm$ 1.1)
AVERAGE	0.42(19.1 $\pm$ 2.3)	0.43(17.9 $\pm$ 2.5)	0.46(18.5 $\pm$ 2.1)	0.60(14.4 $\pm$ 1.7)

From this table, it can be found that the proposed BoostingDA-SVM algorithm achieved highest average diversity value. Table 6.2 (in the parenthesis) also shows that the BoostingDA-SVM achieved better generalization performance than other three up to 4.7% ($19.1\%-14.4\%=4.7\%$) on average. The only difference among these four methods is that proposed BoostingDA-SVM algorithm has a larger scale of learning ability over candidate component classifiers than the other three methods. Hence, the effectiveness of proposed methodology to increase the diversity by increasing the scale of learning ability can be clearly seen. The effectiveness of the proposed diversity augmentation methodology to improve the generalization performance can be demonstrated as well.

Furthermore, the Diversity-Accuracy diagram is also used to visualize the diversity and accuracy distribution over the finally selected component classifiers so as to investigate why the proposed diversity augmentation methodology can improve the generalization performance. We report the diagrams of BoostingDA-SVM and BoostingDA-SVM- σ_{middle} on the ‘Splice’ data set in Figure 6.1 for illustration.

From these figures, we can observe that the component classifiers of BoostingDA-SVM and BoostingDA-SVM- σ_{middle} achieved similar average accuracy. But the component classifiers of BoostingDA-SVM achieved much larger diversity. This makes BoostingDA-SVM more efficient in handling the diversity/accuracy dilemma by selecting component classifiers with larger diversity under the condition of obtaining fairly good accuracy of component classifiers. Hence, BoostingDA-SVM showed better generalization performance. Similar experimental results have also been observed

Figure 6.1: Diversity-Accuracy of BoostingDA-SVM and BoostingDA-SVM- σ_{middle} in the comparison between BoostingDA-SVM- σ_{large} and BoostingDA-SVM- σ_{small} .

6.3 Test the Performance of BoostingDA-SVM

6.3.1 Compare BoostingDA-SVM with Bagging, AdaBoost and SVM

In order to test the generalization performance of proposed BoostingDA-SVM algorithm, we did a set of experiments to compare BoostingDA-SVM with three state-of-the-art classification methods, Bagging, AdaBoost and SVM, on the 13 benchmark data sets. Table 6.3 shows the generalization errors (with standard deviation) for the four methods. The parameters of BoostingDA-SVM were set the same as those used in section 3.8 except that the w_{div} was obtained by doing cross validation. From these results, it can be found that the BoostingDA-SVM dominates the minimal test errors highlighted in bold. Among the three Ensemble methods (Bagging, AdaBoost and BoostingDA-SVM), proposed BoostingDA-SVM performed better than the other two, achieving about 3.3% and 2% improvements respectively. We think that this is because the BoostingDA-SVM considered the diversity/accuracy dilemma in Boosting method. Also, the BoostingDA-SVM performed slightly better than SVM algorithm in general.

Table 6.3: Generalization Errors with Standard Deviation of BoostingDA-SVM (BDA-SVM), BoostingDA-SVM with single best parameters for SVM (BDA-S-SVM), Bagging, AdaBoost and SVM Algorithms on Benchmark Data Sets in %.

DATASET	BDA-SVM	BDA-S-SVM	BAGGING	ADABOOST	SVM
Banana	11.2±1.3	14.2±0.6	14.1±0.6	12.3±0.7	11.5±0.7
B. Cancer	24.7±4.4	30.4±4.4	29.8±4.9	30.4±4.7	26.0±4.7
Diabetes	23.8±2.0	24.8±2.0	27.7±2.4	26.5±2.3	23.5±1.7
German	22.3±2.1	25.8±1.9	26.9±2.7	27.5±2.5	23.6±2.1
Heart	14.8±2.8	19.2±3.5	22.6±3.2	20.3±3.4	16.0±3.3
Image	2.4±0.5	6.2±0.7	4.3±0.9	2.7±0.7	3.0±0.6
Ringnorm	1.7±0.3	5.1±0.2	3.9±0.4	1.9±0.3	1.7±0.1
F. Solar	32.3±1.3	36.8±1.5	35.7±1.9	35.7±1.8	32.4±1.8
Splice	10.4±1.0	14.3±0.5	12.1±1.1	10.1±0.5	10.9±0.7
Thyroid	3.7±2.1	8.5±2.1	6.5±2.3	4.4±2.2	4.8±2.2
Titanic	21.8±1.4	25.6±1.2	24.0±1.3	22.6±1.2	22.4±1.0
Twonorm	2.5±0.4	5.7±0.3	4.3±0.2	3.0±0.3	3.0±0.2
Waveform	10.0±1.1	12.7±0.4	12.4±0.7	10.8±0.6	9.9±0.4
AVERAGE	14.0±1.5	17.7±1.5	17.3±1.7	16.0±1.6	14.5±1.5

6.3.2 Comparison on Unbalanced Data Sets

We also found that proposed BoostingDA-SVM algorithm performed much better than SVM algorithm when facing unbalanced classification problems. In the case of binary-class classification, unbalanced problem means that the number of positive samples is much larger than that of negative samples, or vice versa[36] [9], such as unbalanced document categorization [48], unbalanced clustering for microarray data [134], detecting credit card fraud [62] and many data mining tasks [1] [2] [3]. Proposed BoostingDA-SVM algorithm includes many moderately accurate SVM component classifiers, some of which can focus on the misclassified samples of the minority class. By combining these classifiers, the BoostingDA-SVM can pay more attention on the minority class than standard SVM does, leading to a better generalization performance than standard SVM on unbalanced problems. In the following experiments, we will show the performance of proposed BoostingDA-SVM algorithm on unbalanced classification problems and compare it with several state-of-the-art algorithms specifically designed to solve unbalanced classification problems.

Review of current algorithms dealing with unbalanced problems

A common method for handling unbalanced problems is to re-balance them artificially by under-sampling [96] (ignoring instances from the majority class) or over-sampling [36] (replicating instances from the minority class) or combination of both under-sampling and over-sampling [109]. The most famous one is the SMOTE algorithm [36], which synthetically over-sampled the minority class. Another type

of algorithms focused on biasing the SVM algorithm to deal with the unbalanced problems. Several different ways were used to bias the SVM algorithm: in [181], different penalty constants were used for different classes to control the balance between false positive instances and false negative instances. Cristianini et al [43] used kernel alignment to adjust the kernel matrix to fit the training data. G. Wu and E. Y. Chang [194] realized this by kernel boundary alignment. Furthermore, in [9], SMOTE and different error costs were combined for SVM to better handle unbalanced problems. Decision tree [58] and multiplayer perceptron [129] were also modified to improve the learning performance on unbalanced data sets. In [76], Boosting combined with data generation was used to solve unbalanced problems. Another similar work [196] used SVM ensemble to predict rare classes in scene classification.

Generalization performance on unbalanced data sets

Firstly, we compared proposed BoostingDA-SVM algorithm with standard SVM algorithm on the ‘Splice’ data set from UCI Machine Learning Repository with fixed number of negative samples and varied number of positive samples. ‘Splice’ data set has 483 positive training samples, 517 negative training samples and 2175 test samples. In the following experiments, the number of negative samples is fixed as 500 and the number of positive samples is reduced from 150 to 30 step by step.

From Figure 6.2, it can be found that along with the decreasing ratio of the number positive samples to that of negative samples, the improvement of BoostingDA-

Figure 6.2: Compare BoostingDA-SVM and SVM on 'Splice' Dataset

SVM over SVM becomes more and more. When the ratio reaches 30:500, standard SVM algorithm almost cannot work and performs like random guess, while proposed BoostingDA-SVM algorithm can still work with a relatively good performance.

In the following part, proposed BoostingDA-SVM algorithm were compared with four algorithms, which are standard SVM [179], SVM with different penalty constants (SVM-DPC for short) [181], Under-sampling (US for short) [96] and SMOTE [36] on five UCI imbalanced data sets: Car(3), Glass(7), Letter(26), Segment(1) and Soybean(12). The class-label in the parentheses indicates the class we selected. For each data set, we randomly split it into training and test sets in the ratio of 7 to 3 while ensuring that training and test sets had the same negative to positive ratio [96]. In Table 6.4, the general characteristics of these data sets are given including the number of attributes, the number of positive instances and the number of nega-

tive instances. It also lists the amount of over-sampling of the minority class we did for the SMOTE algorithm as suggested in [194]. For the SVM-DPC algorithm, we used the suggestion in [9] to decide the penalty constants, which worked by setting the cost ratio to the inverse of the imbalanced ratio. For the under-sampling algorithm, the majority class was under-sampled by randomly removing samples from the majority class until it had the same number of instances as minority class.

Table 6.4: general characteristics of data sets and the amount of over-sampling of the minority class we did for the SMOTE algorithm

Data set	# Attribute	# Minority Class	# Majority Class	% Over Sampled
Segment1	19	330	1980	200
Glass7	10	29	185	200
Soybean12	35	44	639	400
Car3	6	69	1659	400
Letter26	17	734	19266	400

The commonly used “Sensitivity” and “Specificity” were taken to measure the performance of each algorithm on the imbalanced data sets. They are defined as:

$$Sensitivity = \frac{\#true_positive}{\#true_positive + \#false_negative} \quad (6.1)$$

$$Specificity = \frac{\#true_negative}{\#true_negative + \#false_positive} \quad (6.2)$$

Based on these, the g-means metric in [96] was calculated to evaluate these five

algorithms on the imbalanced data sets. It is defined as:

$$g = \sqrt{\text{sensitivity} * \text{specificity}} \quad (6.3)$$

Several researchers [96] [194] [9] have used the g-means metric to evaluate the algorithm performance on imbalanced problems because g-means metric combines both the sensitivity and specificity by taking their geometric mean. The obtained g-means metric of all the five algorithms on the UCI data sets are shown in Table 6.5. From this table, it can be found that proposed BoostingDA-SVM algorithm performed best among the five algorithms. It achieved the highest g-means metric in four out of the five data sets and also obtained the highest average g-means metric among the five algorithms. The success of proposed algorithm lies in its Boosting mechanism forcing part of SVM component classifiers to focus on the misclassified instances of the minority class, which can prevent the minority class from being wrongly recognized as a noise of the majority class and classified into it. Hence, the BoostingDA-SVM can achieve better generalization performance on the unbalanced data sets.

Furthermore, the Receiver Operating Characteristic (ROC) analysis was used. We used Area Under the ROC curve (AUC) [23] to compare these five algorithms on the UCI data sets. The AUC is defined as the area under an ROC curve. It is known that larger AUC values indicate generally better classifier performance [78]. The AUCs listed in the Table 6.6 shows that proposed BoostingDA-SVM algorithm achieved the highest average AUCs in all the five data sets. Statistically, the higher AUC values obtained by proposed BoostingDA-SVM means that proposed algorithm

Table 6.5: g-means metric results on the five uci imbalanced data sets

Data set	SVM	SVM-DPC	US	SMOTE	BOOSTINGDA-SVM
Segment1	0.995	0.956	0.945	0.958	0.965
Glass7	0.867	0.874	0.835	0.863	0.885
Soybean12	0.926	0.993	0.921	0.975	0.997
Car3	0	0.382	0.925	0.966	0.975
Letter26	0.818	0.934	0.913	0.927	0.945
Average	0.721	0.827	0.908	0.938	0.953

would favor in classifying a positive (target) instance with a higher probability than other algorithms so that it can better handle the unbalanced problems.

Table 6.6: AUS results on the five UCI imbalanced data sets

Data set	SVM	SVM-DPC	US	SMOTE	BoostingDA-SVM
Segment1	0.998	0.982	0.978	0.985	0.991
Glass7	0.943	0.956	0.938	0.941	0.963
Soybean12	0.960	0.998	0.955	0.989	1.000
Car3	0.631	0.762	0.962	0.984	0.997
Letter26	0.933	0.978	0.950	0.964	0.985
Average	0.893	0.935	0.957	0.972	0.987

6.4 Test the Performance of BoostingDA-LDA

In this section, proposed BoostingDA-LDA algorithm is used for solving face recognition problem and tested on two face databases: ORL database ¹ and Yale face database B ². The ORL database is used to evaluate the performance of BoostingDA-LDA under conditions where the pose, face expression and face scale vary. The Yale face database B is used to examine the performance when illumination varies extremely. The number of initial sub-features, n_{ini} , is set as 10 in order to obtain an initial LDA classifier with more than half training accuracy, the weight of diversity w_{div} is obtained by doing cross validation and the number of iterations T is 100. On these two face databases, proposed BoostingDA-LDA algorithm is compared with other five popular face recognition methods: Fisherface [17], Direct-LDA (D-LDA) [200], Null-space based LDA (N-LDA) [38], Two-Dimensional Fisher Discriminant Analysis (U2DFDA and B2DFDA) [94]. For the Fisherface, the PCA subspace is constrained to $(N - C)$, the final classification dimension is set to $(C - 1)$, where N is the total number of training samples, C is the number of classes. For D-LDA and N-LDA, the final classification dimensions are both set to $(C - 1)$. For U2DFDA, we select the optimal numbers of Fisher feature vector for classification. For B2DFDA, the numbers of the right- and left-multiplying Fisher feature vector are set to be equal, both being the optimal number in U2DFDA.

¹ORL database: <http://www.uk.research.att.com/facedatabase.html>

²Yale face database B: <http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html>

6.4.1 The performance on the ORL database

The ORL database contains images from 40 persons, each providing 10 different images. All images are grayscale and normalized to a resolution of 46×56 pixels.

Ten sample images of two persons from the ORL database are shown in Figure 6.3.

We test the recognition performance with different training numbers. k ($3 \leq k \leq 9$)

Figure 6.3: 10 Images of 2 Persons in ORL Database

images of each person are randomly selected for training and the remaining $10-k$ images of each person for testing. For each number k , 50 runs are performed with different random partition between training set and testing set. Figure 6.4 shows the average recognition rates of the six methods. From this figure, it can be seen that the performance of BoostingDA-LDA is much better than all the others, no matter the size of the training set.

Figure 6.4: Recognition Accuracy on ORL Database

6.4.2 The performance on the Yale face database B

In our experiment, altogether 640 images of 10 persons from Yale face database B are used (64 illumination conditions under the same frontal pose). The image size is 50×60 . Twenty sample images of two persons are shown in Figure 6.5.

Figure 6.5: 20 Images of 2 Persons in Yale Face Database B

The recognition performance is tested with different training numbers. k ($3 \leq k \leq 12$) images of each person are randomly selected for training and the remaining $64-k$ images of each person for testing. For each number k , 100 runs are performed with different random partition between training set and testing set. Figure 6.6 shows the average recognition rate of the six methods. Similarly, from this figure, it can be seen that BoostingDA-LDA is the best one among all the methods.

Figure 6.6: Recognition Accuracy on Yale Face Database B

6.5 Conclusion

Based on the proposed diversity augmentation methodology, we designed a general framework, BoostingDA, to handle the diversity/accuracy dilemma in Ensemble methods. The proposed diversity augmentation methodology creates diversity from the perspective of using component classifiers with different learning abilities. Furthermore, two realization of BoostingDA, named BoostingDA-SVM and BoostingDA-LDA, are presented by using SVM and LDA as component classifiers respectively. A large scale of experiments on benchmark data sets and face recognition databases demonstrated the effectiveness of the proposed diversity augmentation methodology and the two realizations of proposed framework.

Chapter 7

Multi-label SVM Active Learning

7.1 Introduction

Image classification is an important task in computer vision. It can give much help to image indexing and retrieval, because both of them need correctly label images. In an image classification task, some images can easily be assigned with unique labels, such as Figure 7.1(a) and (c), they can be labelled as “Mountain” and “Wave”, respectively. The classification over these images is known as multi-class image classification. However, for the images such as Figure 7.1(b) and (d), the case is different. How should we label them? “Mountain”, “Sunset”, or “Wave”? Due to the subjectivity of human perception, each of them may be a correct label. In this case, if multi-class classification method is still used, these images will be rigidly grouped into only one of the classes. This can be harmful to the applications such as image retrieval. Multi-label classification is a case in which the data can have one or more labels. Therefore, multi-class classification can also be viewed as a special case of multi-label classification. Multi-label classification has been applied to document classification and functional gene classification problems. In this chapter, we focus on multi-label image classification problem or image annotation problem [185].

In real applications, labelling multi-label images is often a time-consuming and costly task. Fortunately, in many cases, we can carefully select images used to query the oracle instead of randomly picking images. If the selection strategy is well-designed, this approach can give rise to better classification result under a given number of labelling chances. This procedure can be categorized into active learning. Active learning is a mechanism which aims to optimize the classification

(a) Mountain

(b) Sunset & Mountain

(c) Wave

(d) Sunset & Wave

Figure 7.1: The images with multi-label

performance while minimizing the number of needed labelled data for training. The first active learning mechanism [40] was introduced by Cohn et al. A similar active learning with SVM was independently developed [156] [32] [172]. Tong et al. [172] used version space to analyze SVM and gave an optimal selection strategy. In [195], Yan et al. proposed a multi-class active learning model for video labelling. Although these methods achieved success in some applications, none of them addressed multi-label classification problems.

Therefore, in this chapter, we proposed a multi-label SVM active learning method

and used it to solve multi-label image classification problem. The key issue of active learning mechanism is the optimization of selection strategy for fastest learning rate. Hence, we proposed two selection strategies: Max Loss strategy and Mean Max Loss strategy. Experimental results on both artificial and real-world image databases demonstrated that the proposed method can effectively reduce the demands of labelled images while maintaining a good classification performance.

7.2 Background: Multi-label SVM

Now the multi-label SVM is introduced, which will be used for active learning in this chapter. Multi-label SVM uses the one-versus-all method to combine predictions of multiple binary SVM classifiers. Let $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$, where $\mathbf{x}_j$ ($\mathbf{x}_j \in \mathbf{R}^n$) denotes a n -dimensional feature vector, its p -dimensional label vector is $\mathbf{y}_j$ ($\mathbf{y}_j \in \mathbf{Y}^p$) which consists of $\{-1, +1\}$, where $+1$ means the data belongs to this class and -1 mean the data does not. p is the size of label set. $\mathbf{y}_j^i$ is the i -th component of $\mathbf{y}_j$ and corresponds to the output of i -th binary SVM. Given $\mathcal{D}$, each class is separated from all other classes by a binary SVM classifier. Then p binary SVM classifiers are obtained in total. For i -th ($i = 1, \dots, p$) classifier, it can be written as $\langle \mathbf{w}_i^*, \mathbf{x} \rangle + b_i^* = 0$. $\mathbf{w}_i^*$ and b_i^* can be found by:

$$\text{minimize} : \Phi(\mathbf{w}_i) = \frac{1}{2} \|\mathbf{w}_i\|^2 + C \sum_{j=1}^m \xi_i^j \quad (7.1)$$

$$\text{subject to} : y_j^i (\langle \mathbf{w}_i, \mathbf{x}_j \rangle + b_i) \geq 1 - \xi_i^j, \quad j = 1, \dots, m \quad (7.2)$$

where ξ_i^j ($\xi_i^j \geq 0$) is the j -th slack variable and C is the parameter controlling the trade-off between function complexity and training error. For each binary SVM classifier f_i , a threshold t_i will be set. This will be discussed in the next section.

7.3 Proposed Multi-label SVM Active Learning

Assume that $\mathcal{I}$, a large set of multi-label images consists of a small set of labelled images, $\mathcal{L}^0$, and an unlabelled image set, $\mathcal{U}^0$. The proposed method focuses on the problem, that given a limited number of labelling chances, of selecting the most informative images from $\mathcal{U}^0$ to be labelled to train a classifier which can give the best classification performance on $\mathcal{I}$. Its procedure is described as follows. At the beginning, a multi-label SVM classifier, f^0 , is trained on $\mathcal{L}^0$. After that, the most informative unlabelled images are selected from $\mathcal{U}^0$ by using the proposed selection strategy. After being labelled, they will be added into $\mathcal{L}^0$ to form a new set, $\mathcal{L}^1$. Afterwards, a new training-selecting-labelling learning cycle will begin based on $\mathcal{L}^1$ until the labelling chances are used up.

The key issue of active learning mechanism is how to select the most informative data per learning cycle to obtain the maximal improvement of classification performance. It is known that, by minimizing the expected loss over the data distribution, SVM can obtain the best classification performance [11]. Hence, we could use the decrease of the expected loss between two contiguous learning cycles as an indicator of the improvement of classification performance. Therefore, the objective of the proposed selection strategy is to maximize the decrease of the expected loss on $\mathcal{I}$ at the i -th learning cycle. Let $\mathcal{U}_s^{i*}$ denote the set of selected images, and $\mathcal{L}^i, \mathcal{U}^i$ denote the labelled and unlabelled image sets at i -th learning cycle. Hence, $\mathcal{U}_s^{i*}$ should

satisfy

$$\mathcal{U}_s^{i*} = \arg \max_{\mathcal{U}_s^i \subset \mathcal{U}^i} \left[\sum_{\mathbf{x} \in \mathcal{I}} L^i(\mathbf{x}) - \sum_{\mathbf{x} \in \mathcal{I}} L^{i+1}(\mathbf{x}, \mathcal{U}_s^i) \right] \quad (7.3)$$

where $L^i(\mathbf{x})$ is the loss value of image $\mathbf{x}$ at the i -th learning cycle. $L^{i+1}(\mathbf{x}, \mathcal{U}_s^i)$ is the loss value of $\mathbf{x}$ at $(i+1)$ -th learning cycle. $L^{i+1}(\mathbf{x}, \mathcal{U}_s^i)$ is a function of selected image set $\mathcal{U}_s^i$ because different $\mathcal{U}_s^i$ leads to different multi-label SVM classifier, f^{i+1} , at $(i+1)$ -th learning cycle and then leads to different loss value for $\mathbf{x}$. But for the present learning cycle i , the classifier, f^i , has been fixed before selecting $\mathcal{U}_s^i$. Hence, $L^i(\mathbf{x})$ can be calculated before selecting $\mathcal{U}_s^i$.

By modifying the kernel of SVM, the images in the induced kernel space can become linearly separable [172]. This guarantees that the training error is zero and then the loss values of the images used for training are zero. Note that there is $\mathcal{I} = \mathcal{L}^i + \mathcal{U}^i$ ($i = 0, 1, 2, \dots$). Therefore, Equation (7.3) becomes

$$\begin{aligned} \mathcal{U}_s^{i*} &= \arg \max_{\mathcal{U}_s^i \subset \mathcal{U}^i} \left[\left(\sum_{\mathbf{x} \in \mathcal{L}^i} L^i(\mathbf{x}) + \sum_{\mathbf{x} \in \mathcal{U}^i} L^i(\mathbf{x}) \right) - \left(\sum_{\mathbf{x} \in \mathcal{L}^{i+1}} L^{i+1}(\mathbf{x}, \mathcal{U}_s^i) + \sum_{\mathbf{x} \in \mathcal{U}^{i+1}} L^{i+1}(\mathbf{x}, \mathcal{U}_s^i) \right) \right] \\ &= \arg \max_{\mathcal{U}_s^i \subset \mathcal{U}^i} \left[\sum_{\mathbf{x} \in \mathcal{U}^i} L^i(\mathbf{x}) - \sum_{\mathbf{x} \in \mathcal{U}^{i+1}} L^{i+1}(\mathbf{x}, \mathcal{U}_s^i) \right] \end{aligned} \quad (7.4)$$

In the i -th learning cycle, $\mathcal{U}^i$ consists of selected image set, $\mathcal{U}_s^i$, and non-selected image set, $\mathcal{U}_s^i$. It is known that $\mathcal{U}^{i+1}$ will be $\mathcal{U}_s^i$ because $\mathcal{U}_s^i$ will be removed from $\mathcal{U}^i$ after selection. As in [32], we assume that all the expected losses of $\mathbf{x}$ in $\mathcal{U}_s^i$ have an equal influence between two contiguous learning cycles. Thus, Equation (7.4) can be rewritten as:

$$\begin{aligned} \mathcal{U}_s^{i*} &= \arg \max_{\mathcal{U}_s^i \subset \mathcal{U}^i} \left[\left(\sum_{\mathbf{x} \in \mathcal{U}_s^i} L^i(\mathbf{x}) + \sum_{\mathbf{x} \in \mathcal{U}_s^i} L^i(\mathbf{x}) \right) - \sum_{\mathbf{x} \in \mathcal{U}_s^i} L^{i+1}(\mathbf{x}, \mathcal{U}_s^i) \right] \\ &\approx \arg \max_{\mathcal{U}_s^i \subset \mathcal{U}^i} \sum_{\mathbf{x} \in \mathcal{U}_s^i} L^i(\mathbf{x}) \end{aligned} \quad (7.5)$$

This function indicates that the selected image set, $\mathcal{U}_s^{i*}$, consists of the images the sum of whose expected loss values are largest. In practice, these images can be selected as those which have the larger $|\mathcal{U}_s^{i*}|$ loss values in $\mathcal{U}^i$. ($|\mathcal{U}_s^{i*}|$ is the size of $\mathcal{U}_s^{i*}$)

The distinctive feature of multi-label image classification lies in that each image, $\mathbf{x}$, can have one or multiple labels. To achieve an accurate estimation of the loss value, $L^i(\mathbf{x})$, of $\mathbf{x}$ in Equation (7.5), the multi-label information has to be incorporated. Hence, in this chapter, multi-label classification is viewed as a combination of n multi-class classification, where n is the number of predicted classes of $\mathbf{x}$. $L^i(\mathbf{x})$ is averaged over the n predicted classes. Thus, we define the loss value of $\mathbf{x}$ as

$$\begin{aligned} L(\mathbf{x}) &= \frac{1}{n} \sum_{y=l_1}^{l_n} \sum_{j=1}^p D(m_{yj} f_j(\mathbf{x})) \\ &= \frac{1}{n} \sum_{y=l_1}^{l_n} \sum_{j=1}^p \max[(1 - m_{yj} f_j(\mathbf{x})), 0] \end{aligned} \quad (7.6)$$

where $\{l_1, \dots, l_n\}$ are the indexes of predicted classes of $\mathbf{x}$. p is the total number of classes. m_{yj} is a component of a coding matrix M [11] which is a $p \times p$ matrix with diagonal components 1 and others -1. $f_j(\mathbf{x})$ is the output of $\mathbf{x}$ on the j -th binary SVM classifier and it is a real value. $D(m_{yj} f_j(\mathbf{x})) = \max\{(1 - m_{yj} f_j(\mathbf{x})), 0\}$ is the loss function of $\mathbf{x}$ classified by $f_j(\mathbf{x})$ and m_{yj} is used as the true label of $\mathbf{x}$. This follows the assumption in [32] that the prediction of $f_j(\mathbf{x})$ is positively correlated to the true label of $\mathbf{x}$.

In order to get the loss value of $\mathbf{x}$ on the predicted n classes, we need to decide the n predicted classes of $\mathbf{x}$. To achieve this, we should estimate the threshold of

loss value on each binary SVM classifier. If the loss value of $\mathbf{x}$ on the J -th class is smaller than the threshold on the J -th binary SVM classifier, $\mathbf{x}$ will belong to class J . We estimate the J -th threshold t_J as:

$$t_J = \arg \min_{\mathbf{x}} \sum_{j=1}^p D(m_{Jj} f_j(\mathbf{x})), \quad \forall \mathbf{x} \in \{\mathbf{x} | \mathbf{y}(J) = 1\} \quad (7.7)$$

where $\mathbf{y}$ is the label vector of $\mathbf{x}$ and $\mathbf{y}(J)$ is $\mathbf{y}$'s J -th component. $\mathbf{y}(J) = 1$ means $\mathbf{x}$ belongs to class J .

Hence, by substituting the Equation (7.6) into (7.5), the proposed Mean Max Loss(MML) Selection Strategy will select the image set $\mathcal{U}_s^{i*}$ satisfying:

$$\mathcal{U}_s^{i*} = \arg \max_{\mathcal{U}_s^i \in \mathcal{U}^i} \sum_{\mathbf{x} \in \mathcal{U}_s^i} \frac{1}{n} \left\{ \sum_{y=l_1}^{l_n} \sum_{j=1}^p \max[(1 - m_{yj} f_j(\mathbf{x})), 0] \right\} \quad (7.8)$$

Although in multi-label image classification task, one image may have more than one label, we could get a fairly simple selection strategy if we calculate the loss value only on the most certainly predicted class of this image. Hence, the proposed Max Loss(ML) Selection Strategy will select the image set $\mathcal{U}_s^{i*}$ as:

$$\mathcal{U}_s^{i*} = \arg \max_{\mathcal{U}_s^i \in \mathcal{U}^i} \sum_{\mathbf{x} \in \mathcal{U}_s^i} \sum_{j=1}^p \max\{(1 - m_{yj} f_j(\mathbf{x})), 0\} \quad (7.9)$$

where y is the index of the most certainly predicted class of $\mathbf{x}$. It can be found that the selection strategy proposed in [195] for multi-class active learning is similar to the proposed ML selection strategy and it is only a special case of the proposed MML selection strategy when the most certainly predicted class is involved only.

7.4 Experimental Results

7.4.1 Artificial Database

Figure 7.2(a) shows the artificial database consisting of three 2D circles with the same center. The data's multi-label information can be found from the legend of Figure 7.2(a). Figure 7.2(b) shows the comparison result of multi-label SVM among random selection, the proposed ML and MML selection strategies on the artificial database. From this figure, it can be seen that the two proposed strategies have a clear advantage over random selection. To achieve the same accuracy, for example, 90% test accuracy, nearly 35 data are saved by using the ML selection strategy and nearly 50 data are saved by using the MML selection strategy than random selection. Based on the same number of labelled training data, maximal 10% and 17% improvement are obtained by the two proposed strategies, respectively. Obviously, MML strategy is more effective than ML strategy. This is because we consider the multi-label information in MML strategy.

7.4.2 Real Color Image Database

The real color image database includes 400 general color images composed from *VisTex* of MIT and *Corel* Stock Photos. Four image classes are defined based on high-level semantics (i.e. defined by several human observers): “People”, “Ocean”, “Mountain”, “Sunset”. Each image belongs to one or several classes. For instance, if an image is a sunset over ocean as Figure 7.1(d), then this image belongs to

Figure 7.2: Comparison Result of Artificial Data Classification

both “Sunset” and “Ocean” classes. This database provides the ground truth for evaluation. A perceptually uniform color space, $CIE - Lab$, is used to represent general color images. Based on this color space, a feature vector of color Moments [167] is defined for each image. This feature vector consists of the mean, variance, and skewness of the pixel values of an image along L , a , and b axes, respectively. The dimensions of this feature vector is 3×3 (9 in total). In Figure 7.3, we plot the experimental result on the real color image database. Here, the numbers of initial training images are 20, 20 and 30, respectively. At each active learning cycle, the numbers of selected unlabelled images are 1, 5 and 1, respectively. It can be seen that the proposed MML selection strategy still shows the best performance.

The Figure 7.4 (a) shows the relationship between the number of selected images per learning cycle and the maximal classification accuracy improvement achieved

(a)Initial 20, 1 Each

(b)Initial 20, 5 Each

(c)Initial 30, 1 Each

Figure 7.3: Comparison result of real image classification.

by proposed MML selection strategy over random selection if the same number of training data are used. It shows that the less the selected images per learning cycle, the higher the improvement of the MML strategy and that the MML strategy performs best when only one image is selected per learning cycle. This is because

that selecting only one image per learning cycle gives the proposed active learning method more chances to interact with the oracle. This makes the selected images more informative for the subsequent classification. In Figure 7.4 (b), we show the relationship between the number of initial training images and the maximal classification accuracy improvement achieved by the MML selection strategy over random selection if the same number of training data are used. It can be found that within the range from 20 to 50, the less the number of initial training images, the higher the improvement of the proposed strategy.

Figure 7.4: the influence of parameters

7.5 Conclusion

In this chapter, a multi-label SVM active learning method was proposed for multi-label image classification. Based on theoretic analysis, we proposed two selection strategies: Max Loss strategy and Mean Max Loss strategy. To our best knowledge, we are the first one who use active learning for multi-label classification. The experimental results on both artificial and real color image databases demonstrated that the two proposed methods can provide better classification performance than original multi-label SVM.

Chapter 8

Conclusion and Recommendations

8.1 Conclusion

Within the last decade, large margin classification technique has been investigated widely as the result of the statistical learning theory. In this thesis, we considered the large margin classification methods, especially the Support Vector Machines and Boosting. Although both Support Vector Machines and Boosting have achieved great success in many areas, they still exit some problems: For SVM, its training process takes so long when handling large training set problems. For Boosting, the way to effectively balance the diversity and accuracy is still an open problem. For multi-label SVM classification, can we use active learning to decrease the needed number of training samples? Hence, in this thesis, these problems have been addressed and solutions offered.

In chapter 3 and 4, by analyzing the learning performance of SVM in the version space, we proposed a method to speed up the training process of SVM by mainly training on the SBA samples. These SBA samples have more probability to be support vectors than other samples. The strength of proposed method lies in reducing the needed number of training samples greatly so as to speed up the training process of SVM on large data sets. A broad experiments on both artificial data sets and benchmark data sets demonstrated its fast training process.

In chapter 5 and 6, a new diversity augmentation methodology was proposed for Boosting methods. This diversity augmentation methodology is very useful when facing the diversity/accuracy dilemma in Boosting methods. Proposed methodology works from a new perspective to augment diversity other than current method,

which aims to enlarge the scale of learning abilities among candidate component classifiers of Boosting. By doing so, larger diversity among component classifiers can be achieved leading to a better generalization performance, which was demonstrated on a large scale of benchmark data sets. To further illustrate the usefulness of proposed methodology in real-world problems, we also tested our realization on face recognition problems. Great improvement has been obtained as well.

In chapter 7, a multi-label SVM active learning method was proposed for multi-label image classification. Based on theoretic analysis, we proposed two selection strategies: Max Loss strategy and Mean Max Loss strategy. To our best knowledge, we are the first one who use active learning to help multi-label classification. The experimental results on both artificial and real color image databases demonstrated that the two proposed methods can provide better classification performance than original multi-label SVM.

Summarizing, the main contribution of this thesis is to propose new algorithms to handle the difficulties when using large margin classification methods for some real problems: One is to speed up the training process of SVM and the second one is a new framework and realizations for handling the diversity/accuracy dilemma of Boosting methods. The third one is a multi-label SVM active learning algorithm for image classification.

8.2 Recommendations for further research

1. Although proposed BoostingDA framework has achieved good performance by using SVM and LDA as component classifiers, one potential future research is realize proposed framework by other component classifiers, such as Neural Networks or Decision Tree. Another good research direction is on how to measure the diversity.

2. One hurdle in many supervised learning tasks is the difficulty in obtaining enough labelled samples. It is often infeasible to obtain labels for large training samples since assigning labels sometimes can require expensive resources such as human labor or laboratory tests. For example, it is expensive to collect image databases of only faces or to assign a medical image. Furthermore, in some cases, ground truth labels are impossible to obtain, e.g. if the necessary measurements can no longer be made, or if the labels will be given only in the future. On the other hand, unlabelled samples may be relatively easy to obtain, but traditionally it was ignored in supervised learning. For instance, we generate more and more data with digital sensors, devices, cameras, computers, and networks. We preserve more data since storage has become cheaper and more plentiful. While it is expensive to collect image databases of only faces, it is cheap to collect arbitrary imagery with occasional faces, e.g. by crawling the world wide web, or by pointing a video camera out the window. Although we interact more with computers, we are willing to give a few examples of our preferences and want computers to learn in an unobtrusive, unsupervised way. Hence, studying the process of learning from partially labelled samples becomes very important and meaningful. We want to use semi-supervised

learning technique to decrease the required number of labelled samples as much as possible. With semi-supervised learning technique, we could provide a few labelled samples describing our purposes and have the system use the remaining unlabelled samples to build a classifier.

Author's Publications

- [1] Xuchun Li, Yan Zhu, and Eric Sung, “SVM Training Accelerator: Sequential Bootstrapped Support Vector Machines”, under review of *IEEE Transactions on Neural Networks*.
- [2] Xuchun Li, Lei Wang, and Eric Sung, “A New Diversity Augmentation Methodology for Handling Diversity/Accuracy Dilemma in Boosting”, under review of *IEEE Transactions on Neural Networks*.
- [3] Xuchun Li, Lei Wang, and Eric Sung, “AdaBoost with SVM Based Component Classifiers”, under review of *IEEE Transactions on System, Man and Cybernetics, part B*.
- [4] Xuchun Li, Lei Wang, and Eric Sung, “Mutli-label Active Learning”, to be submitted to *IEEE Transactions on System, Man and Cybernetics, part B*.
- [5] Xuchun Li, Yan Zhu, and Eric Sung, “Sequential Bootstrapped Support Vector Machines-A SVM Accelerator”, *IEEE International Joint Conference on Neural Networks*, Montreal, Canada, 2005.

- [6] Xuchun Li, Lei Wang, and Eric Sung, “A Study of AdaBoost with SVM Based Weak Learners”, *IEEE International Joint Conference on Neural Networks*, Montreal, Canada, 2005.
- [7] Xuchun Li, Lei Wang, and Eric Sung, “Multi-label SVM Active Learning for Image Classification”, *IEEE International Conference on Image Processing*, Singapore, 2004.
- [8] Xuchun Li, Lei Wang, and Eric Sung, “Improving AdaBoost for Classification on Small Training Sample Sets with Active Learning”, *The Sixth Asian Conference on Computer Vision*, Korea, 2004.

Bibliography

- [1] Aaai'2000 workshop on learning from imbalanced data sets. 2000.
- [2] Icml'2003 workshop on learning from imbalanced data sets (ii). 2003.
- [3] Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explorations*, 6, 2004.
- [4] Krogh A and Vedelsby J. Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems*, 7.
- [5] A. Aamodt and E. Plazas. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–52, 1994.
- [6] D. Achlioptas, F. McSherry, and B. Scholkopf. Sampling techniques for kernel methods. In *Advances in Neural Information Processing Systems 14*, 2002.
- [7] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

- [8] A.J.Smola and B.Scholkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22:211–231, 1998.
- [9] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *Proceedings of the 15th European Conference on Machine Learning*, pages 39–50, 2004.
- [10] E. Albuz, E. Kocalar, and A. A. Khokhar. Scalable color image indexing and retrieval using vector wavelets. *IEEE Transactions on Knowledge and Data Engineering*, 13(5):851–861, 2001.
- [11] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, Dec 2000.
- [12] M. Anthony and N. Biggs. *Computational learning theory: an introduction*. Cambridge University Press, Cambridge, UK, 1992.
- [13] T. Back. *Evolutionary algorithms in theory and practice*. Oxford University Press, Oxford, UK, 1996.
- [14] Jose L. Balcazar, Yang Dai, and Osamu Watanabe. A random sampling technique for training support vector machines. In *Proceedings of the 12th International Conference on Algorithmic Learning Theory*, pages 119–134, 2001.

- [15] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12:2385–2404, 2000.
- [16] R. K. Begg, M. Palaniswami, and B. Owen. Support vector machines for automated gait classification. *IEEE Transactions on Biomedical Engineering*, 52(5):828–838, 2005.
- [17] P.N. Belhumeur, J. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, July 1997.
- [18] R. E. Bellman. *Dynamic programming*. Princeton University Press, Princeton, NJ, 1957.
- [19] A. Del Bimbo and P. Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):121–132, 1997.
- [20] V. Blanz, B. Scholkopf, H. Blthoff, C. J. C. Burges, V. N. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3-d models. *Artificial Neural Networks ICANN96*, 1112:251–256, 1996.
- [21] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

- [22] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.
- [23] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.
- [24] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
- [25] Leo Breiman. Prediction games and arcing classifiers. *Neural Computation*, 11:1493–1517, 1999.
- [26] C. E. Brodley and P. E. Utgoff. Multivariate decision tree. *Machine Learning*, 19:45–77, 1995.
- [27] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6:5–20, 2005.
- [28] Gavin Brown and Jeremy L. Wyatt. The use of the ambiguity decomposition in neural network ensemble learning methods. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 67–74, Washington, DC, USA, 2003.
- [29] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, M. Ares, and D. Haussler. Support vector machine classification of microarray gene expression data. *Technical Report UCSC-CRL-99-09, University of California, Santa Cruz*, 1999.

- [30] B.Scholkopf, C.Burges, and V.Vapnik. Extracting support data for a given task. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*, 1995.
- [31] C. J. C. Burges and B. Scholkopf. Improving the accuracy and speed of support vector learning machines. *Advances in Neural Information Processing Systems 9*, pages 375–381, 1999.
- [32] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceeding of the 17th International Conference on Machine Learning*, pages 111–118, 2000.
- [33] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *14th Annual Conference on Neural Information Processing Systems*, pages 409–415, 2000.
- [34] H. Cevikalp, M. Neamtu, M. Wilkes, and A. Barkana. Discriminative common vectors for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):4–23, Jan 2005.
- [35] Jonathan Cheung-Wai Chan, Chengquan Huang, and R. DeFries. Enhanced algorithm performance for land cover classification from remotely sensed data using bagging and boosting. *IEEE Transactions on Geoscience and Remote Sensing*, 39(3):693–695, 2001.

- [36] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [37] E-Liang Chen, Pau-Choo Chung, Ching-Liang Chen, Hong-Ming Tsai, and Chein-I Chang. An automatic diagnostic system for ct liver image classification. *IEEE Transactions on Biomedical Engineering*, 45(6):783–794, 1998.
- [38] Li-Fen Chen, Hong-Yuan Mark Liao, Ming-Tat Ko, Ja-Chen Lin, and Gwo-Jong Yu. A new lda-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, 33(10):1713–1726, 2000.
- [39] T. Cober and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.
- [40] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.
- [41] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Machine learning*, 48:253–285, 2002.
- [42] C. Cortes and V.N.Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
- [43] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz S. Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems*, pages 367–373, 2001.

- [44] Nello Cristianini and John Shawe Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge UK, 2000.
- [45] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.
- [46] M. Davy, A. Gretton, A. Doucet, and R. J. W. Rayner. Optimized support vector machines for nonstationary signal classification. *IEEE Signal Processing Letters*, 9(12):442–445, 2002.
- [47] D. DeCoste and B. Scholkopf. Training invariant support vector machines. *Machine Learning*, 2001.
- [48] M. Dolores del Castillo and Jose Ignacio Serrano. A multistrategy approach for digital text categorization from imbalanced documents. In *ACM SIGKDD Explorations: Special issue on learning from imbalanced datasets*, pages 70–39, 2004.
- [49] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 19:380–393, 1997.
- [50] Ayhan Demiriz, kristin P.Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

-
- [51] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.
- [52] D.E.Rumelhart, G.E.Hinton, and R.J.Williams. Learning internal representations by error propagation. *Parallel distributed processing: explorations in macrostructure of cognition*, I:318–362, 1986.
- [53] M. Dettling and P. Buhlmann. How to use boosting for tumor classification with gene expression data. *Preprint*.
- [54] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, Aug 2000.
- [55] D.P.Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.
- [56] H. Drucker, D.Wu, and V. N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, pages 1048–1054, 1999.
- [57] Harris Drucker, Robert Schapire, and Patrice Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705–719, 1993.

- [58] C. Drummond and R. Holte. Exploiting the cost (in)sensitivity of decision tree splitting criteria. In *Proceedings of the seventeenth International Conference on Machine Learning*, pages 239–246, 2000.
- [59] Kai-Bo Duan, J. C. Rajapakse, Haiying Wang, and F. Azuaje. Multiple svm-rfe for gene selection in cancer classification with expression data. *IEEE Transactions on NanoBioscience*, 4(3):228–234, 2005.
- [60] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *7th International Conference on Information and Knowledge Management*, 1998.
- [61] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *15th Annual Conference on Neural Information Processing Systems*, pages 681–687, 2001.
- [62] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.
- [63] Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.
- [64] G. M. Foody and A. Mathur. A relative evaluation of multiclass image classification by support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(6):1335–1343, 2004.

- [65] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121:256–285, 1995.
- [66] Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. *Computational Learning Theory*, pages 325–332, 1996.
- [67] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [68] Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.
- [69] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.
- [70] F. Rosenblatt. *Principles of neurodynamics: Perceptron and theory of brain mechanisms*. Spartan Books, Washington D.C., 1962.
- [71] Glenn Fung and O. L. Mangasarian. Proximal support vector classifiers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86, 2001.
- [72] T. Furey, N. Cristianini, N. Duffy, D. Bednarski, M. Schummer, and D. Hausler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, page 906914, 2000.

-
- [73] L. Marquez G. Escudero and G. Rigau. Boosting applied to word sense disambiguation. In *Proceedings of the 12th European Conference on Machine Learning*, pages 129–141, 2000.
- [74] D. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA, 1989.
- [75] Guodong Guo and S. Z. Li. Content-based audio classification and retrieval by support vector machines. *IEEE Transactions on Neural Networks*, 14(1):209–215, 2003.
- [76] Hong Guo and Herna L. Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM SIGKDD Explorations: Special issue on learning from imbalanced datasets*, pages 30–39, 2004.
- [77] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.
- [78] D.J. Hand. *Construction and Assessment of Classification Rules*. John Wiley and Sons, UK, 1997.
- [79] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, 1999.

- [80] Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, Aug 2001.
- [81] H.W.Shin and S.Y.Sohn. Selected tree classifier combination based on both accuracy and error diversity. *Pattern Recognition*, 38:191–197, 2005.
- [82] Raj D. Iyer, David D. Lewis, Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting for document routing. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, 2000.
- [83] F. V. Jensen. *An introduction to Bayesian networks*. Springer Verlag, New York, USA, 1996.
- [84] Pedro J.Moreno, Beth Logan, and Bhiksha Raj. A boosting approach for confidence scoring. *Proceedings of the 7th European Conference on Speech Communication and Technology*, 2001.
- [85] T. Joachims. Text categorization with support vector machines. In *Proceedings of European Conference on Machine Learning*, 1998.
- [86] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of AI Research*, 4:237–285, 1996.
- [87] Wei-Chun Kao, Kai-Min Chung, Chia-Liang Sun, and Chih-Jen Lin. Decomposition methods for linear support vector machines. *Nueral Computation*, 16:1689–1704, 2004.

- [88] M. Kearns and L. G. Valiant. Learning boolean formulae or finite automata is as hard as factoring. *Technical Report TR-14-88, Harvard University Aiken Computation Laboratory*, 1988.
- [89] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, Jan 2000.
- [90] Kwang In Kim, Keechul Jung, and Jin Hyung Kim. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1631–1639, Dec 2003.
- [91] Kwang In Kim, Keechul Jung, Se Hyun Park, and Hang Joon Kim. Support vector machines for texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1542–1550, 2002.
- [92] Jyrki Kivinen, Alexander J. Smola, and Robert C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, Aug 2004.
- [93] Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. *Computational Learning Theory*, pages 134–144, 1999.
- [94] H. Kong, L. Wang, E.K. Teoh, J.G. Wang, and R. Venkateswarlu. A framework of 2d fisher discriminant analysis: Application to face recognition with small

- number of training samples. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1083–1088, 2005.
- [95] J. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, 1992.
- [96] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the 14th International Conference on Machine Learning*, pages 179–186, 1997.
- [97] L. I. Kuncheva, M. Skurichina, and R. P. W. Duin. An experimental study on diversity for bagging and boosting with linear classifiers. *Information Fusion*, 3:245–258, 2002.
- [98] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, May 2003.
- [99] John Lafferty. Additive models, boosting and inference for generalized divergences. *Proceedings of the 12th Annual Conference on Computational Learning Theory*, pages 125–133, 1999.
- [100] K. J. Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Nueral Networks*, 3:33–43, 1990.
- [101] L.Devroye. Bounds for the uniform deviation of empirical measures. *Journal of Multivariate Analysis*, 16(12):72–79, 1982.

- [102] Guy Lebanon and John Lafferty. Boosting and maximum likelihood for exponential models. *Advances in Neural Information Processing Systems*, pages 447–454, 2002.
- [103] Y. LeCun. Learning processes in an asymmetric threshold network. *Disordered systems and biological organizations*, pages 233–240, 1986.
- [104] Y. A. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Mller, E. Sackinger, P. Y. Simard, and V. N. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks*, pages 261–276, 1995.
- [105] Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Technical Report 1064, Department of Statistics, University of Wisconsin*, 2002.
- [106] Y.-J. Lee and O. Mangasarian. Rsvm: Reduced support vector machines. In *Proceedings of the first SIAM International Conference on Data Mining*, 2001.
- [107] Shutao Li, James Tin-Yau Kwok, Ivor Wai-Hung Tsang, and Yaonan Wang. Fusing images with different focuses using support vector machines. *IEEE Transactions on Neural Networks*, 15(6):1555–1561, 2004.
- [108] Chin-Teng Lin, Yin-Cheung Lee, and Her-Chang Pu. Satellite sensor image classification using cascaded architecture of neural fuzzy network. *IEEE Transactions on Geoscience and Remote Sensing*, 38(2):1033–1043, 2000.

-
- [109] Charles X. Ling and Chenghui Li. Data mining for direct marketing problems and solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998.
- [110] Juwei Lu, K.N. Plataniotis, and A.N. Venetsanopoulos. Face recognition using lda-based algorithms. *IEEE Transactions on Neural Networks*, 14(1):195–200, Jan 2003.
- [111] M. K. Mandal, T. Aboulnasr, and S. Panchanathan. Image indexing using moments and wavelets. *IEEE Transactions on Consumer Electronics*, 42(3):557–565, 1996.
- [112] O. L. Mangasarian and David R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10:1032–1037, 1999.
- [113] O. L. Mangasarian and David R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001.
- [114] D. Mattera and S. Haykin. Support vector machines for dynamic reconstruction of a chaotic system. *Advances in Kernel MethodsSupport Vector Learning*, pages 211–242, 1999.
- [115] A. K. McCallum. Multi-label text classification with a mixture model trained by em. In *AAAI’99 Workshop on Text Learning*, 1999.

- [116] F. Melgani and L. Bruzzone. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(8):1778–1790, 2004.
- [117] Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, Mar 2005.
- [118] Stefano Merler, Cesare Furlanello, Barbara Larcher, and Andrea Sboner. Tuning cost-sensitive boosting and its application to melanoma diagnosis. *Multiple Classifier Systems 2001*, 2096:32–42, 2001.
- [119] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine learning, neural and statistical classification*. Ellis Horwood, New York, USA, 1994.
- [120] Tom M. Mitchell. *Machine Learning*. McGraw Hill Inc, 1997.
- [121] J. J. More and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Optimization*, 1(1):93–113, 1991.
- [122] M.C. Mozer, R. Wolniewicz, D.B. Grimes, E. Johnson, and H. Kaushansky. Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry. *IEEE Transactions on Neural Networks*, 11(3):690 –696, 2000.
- [123] M.Pontil and A.Verri. Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:637–646, 1998.

- [124] M.Pontil and A.Verri. Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):735–746, 2001.
- [125] K.-R. Muller, A. J. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, and V. N. Vapnik. Predicting time series with support vector machines. *Artificial Neural Networks ICANN97*, pages 999–1004, 1997.
- [126] Klaus-Robert Muller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernhard Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.
- [127] B. K. Natarajan. *Machine learning: a theoretical approach*. Morgan Kaufmann, San Mateo, CA, 1991.
- [128] R. Nishii and S. Eguchi. Supervised image classification by contextual adaboost based on posteriors in neighborhoods. *IEEE Transactions on Geoscience and Remote Sensing*, 43(11):2547–2554, 2005.
- [129] A. Nugroho, S. Kuroyanagi, and A. Iwata. A solution for imbalanced training sets problem by combnet-ii and its application on fog forecasting. *IEICE Transaction on Information and Systems*, pages 1165–1174, 2002.
- [130] O.Chapelle, P.Haffner, and V.Vapnik. Svms for histogram-based image classification. *IEEE Transactions on Neural Networks*, 1999.

-
- [131] T. Onoda, G. Ratsch, and K.-R. Muller. Applying support vector machines and boosting to a non-intrusive monitoring system for household electric appliances with inverters. In *Proceedings of NC'2000*, 2000.
- [132] David W. Opitz. Feature selection for ensembles. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence*, pages 379–384, Orlando, Florida, USA, 1999.
- [133] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of Computer Vision and Pattern Recognition*, pages 130–136, 1997.
- [134] Ronald Pearson, Gregory Goney, and James Shwaber. Imbalanced clustering for microarray time-series. In *ICML'2003 Workshop on Learning from Imbalanced Data Sets (II)*, 2003.
- [135] Michael L. Littman Janos A. Csirik Peter Stone, Robert E. Schapire and David McAllester. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. *Journal of Artificial Intelligence Research*, 19:209–242, 2003.
- [136] Ming Hong Pi and Chun-Hung Li. A low-complexity index for fractal image indexing. *Canadian Journal of Electrical and Computer Engineering*, 30(2):89–92, 2005.

- [137] P.L.Bartlett. The sample complexity of pattern classification with neural networks: the size of the weight is more important than the size of the network. *IEEE Transactions on Information Geometry*, 44(2):525–536, 1998.
- [138] D. A. Pomerleau. *Knowledge-based training of artificial neural networks for autonomous robot driving*. Kluwer Academic Publishers, Boston, USA, 1993.
- [139] Michael P.S.Brown, William Noble Brundy, David Lin, Nello Cristianini, Charles Sugnet, Terrence S. Furey, Manuel Ares Jr., and David Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. In *Proceedings of the National Academy of Sciences*, 1998.
- [140] Guoping Qiu. Color image indexing using btc. *IEEE Transactions on Image Processing*, 12(1):93–101, 2003.
- [141] J. Ross Quinlan. Introduction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
- [142] J.Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, San Francisco, CA, USA, 1993.
- [143] Gunnar Ratsch, Manfred Warmuth, Sebastian Mika, Takashi Onoda, Steven Lemm, and Klaus Robert Muller. Barrier boosting. *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 170–179, 2000.

- [144] M. Rochery, R. Schapire, M. Rahim, N. Gupta, G. Riccardi, S. Bangalore, H. Alshawhi, and S. Douglas. Combining prior knowledge and boosting for call classification in spoken language dialogue. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2002.
- [145] D. Roobaert. Improving the generalization of linear support vector machines: an application to 3d object recognition with cluttered background. In *Proceedings of SVM Workshop at IJCAI'99*, 1999.
- [146] D. Roobaert and M. M. Van Hulle. View-based 3d object recognition with support vector machines. In *IEEE Neural Networks for Signal Processing Workshop*, 1999.
- [147] D. Rumelhart, B. Widrow, and M. Lehr. The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–92, 1994.
- [148] R. E. Schapire, M. Rochery, M. Rahim, and N. Gupta. Boosting with prior knowledge for call classification. *IEEE Transactions on Speech and Audio Processing*, 13(2):174–181, 2005.
- [149] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39:135–168, 2000.
- [150] R. E. Schapire, Y. Singer, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

-
- [151] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
- [152] Robert E. Schapire, Marie Rochery, Mazin Rahim, and Narendra Gupta. Incorporating prior knowlege into boosting. *Proceeding of 19th International Conference on Machine Learning*, pages 538–545, 2002.
- [153] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, Dec 1999.
- [154] Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting and rocchio applied to text filtering. *Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval*, pages 215–223, 1998.
- [155] Robert E. Schapire, Peter Stone, David McAllester, Michael L. Littman, and Janos A. Csirik. Modeling auction price uncertainty using boosting-based conditional density estimation. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.
- [156] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceeding of the 17th International Conference on Machine Learning*, pages 839–846, 2000.
- [157] Bernhard Scholkopf, Chris Burges, and Alex J. Smola. *Advances in Kernel Methods - Support Vector Learning*. MIT press, Cambridge, Mass, 1999.

- [158] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2001.
- [159] H. P. Schwefel. *Evolution and optimum seeking*. John Wiley & Sons, New York, USA, 1995.
- [160] H. Schwenk and Y. Bengio. Adaptive boosting of neural networks for character recognition. *Technical Report 1072, Department d'Informatique et Recherche Operationelle, Univerite de Montreal*, 1997.
- [161] Fabrizio Sebastiani, Alessandro Sperduti, and Nicola Valdambrini. An improved boosting algorithm and its application to automated text categorization. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 78–85, 2000.
- [162] J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.
- [163] J. Shawe-Taylor and N. Cristianini. On the generalisation of soft margin algorithms. *IEEE Transactions on Information Theory*, 48(10):2721–2735, 2002.
- [164] Hyunjung Shin and Sungzoon Cho. Fast pattern selection for support vector classifiers. In *Advances in Knowledge Discovery and Data Mining, 7th Pacific-Asia Conference*, pages 376–387, 2003.

- [165] Alex J. Smola and Bernhard Schokopf. Sparse greedy matrix approximation for machine learning. In *Proceeding of the 17th International Conference on Machine Learning*, pages 911–918, 2000.
- [166] Qing Song, Wenjie Hu, and Wenfang Xie. Robust support vector machine with bullet hole image classification. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 32(4):440–448, 2002.
- [167] M. Stricker and M. Orengo. Similarity of color images. In *Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, pages 381–392, 1995.
- [168] D.L. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.
- [169] D.L. Swets and Juyang Weng. Hierarchical discriminant analysis for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):386–401, 1999.
- [170] Kinh Tieu and Paul Viola. Boosting image retrieval. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1:13–15, 2000.
- [171] Simon Tong. Active learning: Theory and application. *PHD Thesis*, August 2001.

-
- [172] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.
- [173] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.
- [174] T.S.Jaakkola and D.Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, 11, 1998.
- [175] Giorgio Valentini and Thomas G. Dietterich. Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. *Journal of Machine Learning Research*, 5:725–775, Jul 2004.
- [176] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.
- [177] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, November 1984.
- [178] Vladimir Vapnik. *The nature of statistical learning theory*. Springer-Verlag, New York, 1995.
- [179] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley and Sons Inc., New York, 1998.

- [180] M. M. Veloso. *Planning and learning by analogical reasoning*. Springer Verlag, Berlin, 1992.
- [181] K. Veropoulos, C. Campbell, and N Cristianini. Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on Artificial Intelligent*, pages 55–60, 1999.
- [182] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1:511–518, 2001.
- [183] Paul Viola and Michael J. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
- [184] S.V.N. Vishwanathan, Alexander J. Smola, and M. Narasimha Murty. *simplesvm*. In *Proceeding of the 20th International Conference on Machine Learning*, pages 760–767, 2003.
- [185] V.Lavrenko, R.Manmatha, and J.Jeon. A model for learning the semantics of pictures. In *17th Annual Conference on Neural Information Processing Systems*, 2003.
- [186] V.N.Vapnik and A.Ja.Chervonenkis. The necessary and sufficient conditions for consistency of the method of empirical risk minimization (in russian). *Yeabook of the Academy of Sciences of the USSR on Recognition, classification and Forecasting 2*, pages 217–249, 1989.

- [187] V.N.Vapnik and A.Y.Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–280, 1971.
- [188] V.Vapnik and A.Chervonenkis. *Theory of Pattern Recognition (in Russian)*. Nauka, Moscow, 1974.
- [189] Lei Wang and Kap Luk Chan. Bootstrapping svm active learning by incorporating unlabelled images for image retrieval. *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.
- [190] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
- [191] C. K. I. Williams and M. Seeger. Using the nystrom method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, 2001.
- [192] R.C. Williamson, A.J. Smola, and B. Scholkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transactions on Information Theory*, 47(6):2516–2532, 2001.
- [193] Terry Windeatt. Diversity measures for multiple classifier system analysis and design. *Information Fusion*, 6:21–36, 2005.

-
- [194] Gang Wu and Edward. Y. Chang. Kba: Kernel boundary alignment considering imbalanced data distribution. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):786–795, 2005.
- [195] R. Yan, J. Yang, and A. Hauptmann. Automatically labeling video data using multi-class active learning. In *The 9th International Conference on Computer Vision*, pages 516–523, 2003.
- [196] Rong Yan, Yan Liu, Rong Jin, and Alex Hauptmann. On predicting rare class with svm ensemble in scene classification. In *Proceeding of the IEEE International Conference on Acoustics, Speech, and Signal 2003*, pages III – 21–4, Apr 2003.
- [197] Jian Yang, A.F. Frangi, Jing-Yu Yang, David Zhang, and Zhong Jin. Kpca plus lda: a complete kernel fisher discriminant framework for feature extraction and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):230–244, Feb 2005.
- [198] Zheng Rong Yang and Kuo-Chen Chou. Bio-support vector machines for computational proteomics. *Bioinformatics*, 20:735–741, Mar 2004.
- [199] Peng-Yeng Yin, B. Bhanu., Kuang-Cheng Chang, and Anlei Dong. Integrating relevance feedback techniques for image retrieval using reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1536–1551, 2005.

-
- [200] Hua Yu and Jie Yang. A direct lda algorithm for high-dimensional data with application to face recognition. *Pattern Recognition*, 34(10):2067–2070, 2001.
- [201] Hwanjo Yu, Jiong Yang, and Jiawei Han. Classifying large data sets using svm with hierarchical clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315, 2003.
- [202] Z.-H. Zhou, Y. Jiang, Y.-B. Yang, and S.-F. Chen. Lung cancer cell identification based on artificial neural network ensembles. *Artificial Intelligence in Medicine*, 24(1):25–36, 2002.
- [203] A. Zien, G. Rtsch, S. Mika, B. Schlkopf, T. Lengauer, , and K.-R. Mller. Engineering support vector machine kernels that recognize translation initiation sites in dna. *Bioinformatics*, page 799807, 2000.