SVM Classification for Imbalanced Datasets with Multi Objective Optimization Framework

by

Ayşegül Öztürk

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Industrial Engineering

Koç University

July, 2009

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Ayşegül Öztürk

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Serpil Sayın(Advisor)

_____

Asst. Prof. Özden Gür Ali

_____

Asst. Evrim Didem Güneş

Date: _____

ABSTRACT

Classification of imbalanced datasets in which negative instances, also called majority class, outnumber the positive instances, also called minority class, is a significant challenge. These kind of datasets are commonly encountered in real-life problems. However, performance of well-known classifiers are limited in case there exists imbalance in the dataset. Various solution approaches are proposed in the literature, applied on either data-level or algorithm-level to address the problems that arise in case of imbalance. Data-Level approaches mainly aim to balance the distribution of the dataset either by eliminating some instances of majority class or by replicating some instances of minority class. On the other hand, Algortihm-Level approaches either bias the algorithm proposed or adjust some parameters in order to bias the underlying model. Support Vector Machines (SVMs) that have a solid theoretical background also encounter a dramatic decrease in performance when the distribution of the datasets is imbalanced.

The objective of this study is to improve the classification performance of SVMs for imbalanced datasets. The method proposed is based on modifying L_1 Norm SVM formulation to create a three objective optimization problem so as to incorporate into the formulation the error sums for the two classes independently. Motivated from the multi objective nature of the SVMs, the solution approach uses the fundamentals of Multi Objective Optimization. The proposed method suggests to reduce the problem formulation into two criteria variations and to investigate the efficient frontier systematically. Investigating the efficient frontier by a systematic procedure leads the method to evaluate the problem for a remarkable set of parameters rather than adjusting a few parameters empirically as in the existing approaches. Therefore the proposed method improves the performance of a SVM by decreasing the computational effort needed for evaluating the problem for the same amount of parameters. The results are reported in terms

of three widely used metrics and computational experiments are discussed in detail.

ÖZETÇE

Negatif sınıf (çoğunluk sınıfı) örneklerinin pozitif sınıf (azınlık sınıfı) örneklerinden fazla olduğu Dengesiz Veri Kümelerinde sınıflandırma önem taşımaktadır. Bu tip, dağılımları dengesiz olan veri kümelerine gerçek hayat problemlerinde sıklıkla rastlanmaktadır. Ancak veri kümesinin dengesiz olması durumunda sıklıkla bilinen sınıflandırma algoritmalarının performansı düşmektedir. Literatürde, dağılımın dengesiz olması durumunda ortaya çıkan problemleri çözmek için veri seviyesinde veya algoritma seviyesinde çeşitli çözüm yaklaşımları sunulmuştur. Veri seviyesindeki yaklaşımlar genellikle, çoğunluk sınıfından bazı örnekler eleyerek veya azınlık sınıfındaki bazı örnekleri yineleyerek veri kümesinin dağılımını dengelemeyi amaçlarlar. Algoritma seviyesindeki yaklaşımlar ise önerilen algoritmaya veya altında yatan modele sapma ekleyerek sınıflandırma performansını arttırmayı amaçlarlar. Güçlü bir teorik alt yapıya sahip olan SVM' lerde de veri kümesinde dengesizlik olduğu durumlarda performansta ciddi bir düşüş gözlenmektedir.

Bu çalışma SVM' lerin dengesiz veri kümelerindeki sınıflandırma performansını iyileştirmeyi amaçlamaktadır. Sunulan yöntem L_1 Norm SVM formulasyonuna her iki sınıf için hata toplamlarını birbirinden ayrı şekilde dahil ederek üç ölçüt fonksiyonuna sahip bir optimizasyon problemi yaratmaktadır. SVM' lerin çok ölçütlü yapısından dolayı çözüm yaklaşımı Çok Ölçütlü Optimizasyon temelleri üzerine kurulmuştur. Sunulan yöntem, problem formulasyonunu alternatif iki ölçüt fonksiyonlu formulasyonlara indirgemeyi ve etkin kümeyi sistematik bir şekilde incelemeyi önermektedir. Etkin kümeyi sistematik bir şekilde incelemek, yalnızca empirik olarak belirlenmiş kısıtlı sayıda parametreye değer vererek problemi değerlendiren yöntemlerin aksine, önemli sayıda parametreyle sistematik olarak problemi değerlendirmeyi sağlar. Böylece, önerilen yöntem daha az hesaplama zamanı ile daha çok parametre değeri kullanarak SVM' in performansında artış sağlamaktadır. Sonuçlar sıklıkla kullanılan üç metrik cinsinden rapor edilmiştir ve tüm deneyler ayrıntılı şekilde tartışılmıştır.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my deepest gratitude to my advisor Serpil Sayın for her insightful discussions and never ending support during this work. I have learned and achieved more than I ever dreamed of in two-years by her precious guiding and help as well as her inspiration and encouragement. I feel myself very lucky and very proud of being one of her students.

I am grateful to Özden Gür Ali and Evrim Didem Güneş for taking part in my thesis committee, for critical reading of this thesis and for their supportive suggestions and comments.

I also thank to TUBITAK (The Scientific and Technological Research Council of Turkey) for their generous financial support.

I would like to thank to all my friends at Koç University, especially Derya Kunduzcu, Deniz Kubalı and Nihan Çömden for their valuable friendship , for their encouragement when I needed and for all the fun and good times we shared together as officemates. Also I would like to thank Umut ınetaş not only for his valuable friendship for many years but also his encouragement and support for any problem I have ever encountered. Special thanks go to Doğan Aşkan who has stand by me the whole time as my life support unit. And the last but not least, I would like thank to my parents and my brother for believing me and trusting me at every step I have taken so far and for their endless and supporting love.

TABLE OF CONTENTS

List of Tables **ix**

List of Figures **x**

Chapter 1: Introduction and Background **1**

1.1 Introduction 1

1.2 Fundamental Concepts in Data Mining 3

1.3 Support Vector Machine Classification 6

1.3.1 L_1 Norm SVM Formulation 11

Chapter 2: Literature Review **13**

2.1 The Imbalanced Datasets 13

2.2 Handling the Imbalanced Datasets 15

2.2.1 Data Level Solution Approaches 15

2.2.2 Algorithm Level Solution Approaches 17

Chapter 3: Alternative Method Handling the Imbalance **20**

3.1 Obtaining Efficient Frontier in the Two Criteria Case 25

3.2 Grid Search Model Reducing Three Criteria to Two Criteria 28

3.3 Datasets 34

3.3.1 Soybean Dataset 34

3.3.2 Yeast Dataset 34

3.3.3 Abalone Dataset 35

3.3.4 Statlog Dataset 35

3.3.5 Letter Recognition Dataset 36
3.4 Evaluation Metrics 36

Chapter 4: Experiments **47**
4.1 Experiment Settings 47
4.2 Results 48

Chapter 5: Reducing Computational Time **57**

Chapter 6: Conclusion **61**

Bibliography **64**

Vita **70**

LIST OF TABLES

3.1 Properties of Soybean Dataset 34
3.2 Properties of Yeast Dataset 35
3.3 Properties of Abalone Dataset 35
3.4 Properties of Statlog Dataset 36
3.5 Properties of Letter Recognition Dataset 36

4.1 Evaluation for soybean dataset 48
4.2 Evaluation for yeast dataset 49
4.3 Evaluation for abalone dataset 49
4.4 Evaluation for statlog dataset 50
4.5 Evaluation for letter recognition dataset 51

5.1 Reduced Evaluation for Soybean Dataset 58
5.2 Reduced Evaluation for Yeast Dataset 58
5.3 Reduced Evaluation for Abalone Dataset 59
5.4 Reduced Evaluation for Statlog Dataset 59
5.5 Reduced Evaluation for Letter Recognition Dataset 59

LIST OF FIGURES

1.1 Efficient frontier examples in 3-D. 8

2.1 Ideal Classification. 14
2.2 Classification when imbalance is not considered. 15

3.1 Efficient frontier examples in 3-D. 23
3.2 Efficient frontier for a bicriteria optimization problem. 24
3.3 Objective value vs. Total error. 27
3.4 *grid1* Search in Majority Epsilon vs. Minority Epsilon Space 31
3.5 Sample ROC Curve 39
3.6 The best five percent of g-means values for Soybean 41
3.7 The best five percent of AUC values for Soybean 42
3.8 The best g-means and the best AUC values for Soybean 43
3.9 The best five percent of g-means values for Yeast 44
3.10 The best five percent of AUC values for Yeast 45
3.11 The best g-means and the best AUC values for Yeast 46

4.1 Trade-off between the three objective 53
4.2 How the norm changes due to the error on minority class 55
4.3 How the norm changes due to the error on majority class 56

Chapter 1

INTRODUCTION AND BACKGROUND

1.1 Introduction

In recent years, due to improvements on data storage costs, massive amount of business and research data have been collected and stored [41]. Therefore extracting useful information from this data has arisen as an interesting and popular research area. The phrase *Knowledge Discovery in Databases (KDD)* is used to describe this process. Specifically, KDD is defined as "the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data" [26]. Similarly, data mining is defined as "a step in the KDD process consisting of enumeration of patterns or models over the data" [26]. Data mining combines methods of statistics, databases, machine learning and pattern recognition to extract-mine- concepts, concept interrelations and interesting patterns from large datasets [56]. The key characteristic of data mining is that it used a discover-based approach rather than verification-based approach. In a verification-based approach, the user hypothesizes some presumptions and tries to prove or disprove these presumptions. However, in the discovery-based approach existing data is analyzed in order to extract new or previously unknown or unrealized information [55]. Briefly, the most important property of the data mining is that data directs the user's method.

Optimization techniques are widely used for theoretical and applied problems, as Euler said "Nothing happens in the universe that does not have a maximum or a minimum" [47]. By mathematical programming, that is optimization subject to constraints, various problems of various fields can be formulated and effectively solved. Recently, mathematical programming techniques have been widely used for data mining problems [40], since they perform well while they provide

theoretical basis on the problems. Most of the mathematical models introduced are either Linear Programming or Convex Quadratic Programming [40] both of which are polynomial-time solvable [48]. Also fast linear and quadratic programming codes like CPLEX that is capable of solving very large problems help the mathematical models introduced to be applied for large problems as well as small ones [40]. Finally, considering a general idea of 'finding a rule that minimizes the errors made in prediction' some data mining problems like classification the use of mathematical programming is prospected.

Support Vector Machines (SVMs), introduced by Vapnik et al. [49], is one of the data mining tools that is based on mathematical programming and has a strong theoretical basis. We will be giving detailed background information about classification problem of data mining and support vector machines method proposed as a solution approach for this problem.

One of the most challenging problems of data mining is the concept classification on an imbalanced dataset since classifier performance dramatically falls in such a case [34], [13]. In this study, we aim to improve classification performance of support vector machines in imbalanced datasets. Our motivation leading to the choice of support vector machines lies on its theoretical basis, optimization based structure as well as the results stating that SVMs are shown to be least sensitive to class imbalance among decision trees and multi-layer Perceptrons(MLP) [34].

To summarize the contents of this thesis, Chapter 1 continues with relevant background information about data mining and Support Vector Machines that will form a basis for the rest of the study. Chapter 2 summarizes the literature of Classification on Imbalanced Datasets. In this chapter application areas with imbalanced datasets are mentioned as well as the various solution approaches proposed by the researchers. We group these solution approaches into two as Data Level and Algorithm Level approaches. Chapter 3 focuses on the multi objective nature of the Support Vector Machines and we propose and explain our solution approach. We introduce three variations based on our main solution approach. We also introduce the datasets and evaluation metrics used. Chapter 4 contains the information about the experiments conducted. Also the detailed results of the experiments are given. Chapter 5 describes a heuristic for reducing the computational time of the proposed methods in case of a extremely large and

imbalanced dataset. Also some results of the heuristic method are listed. Finally, Chapter 6 concludes the study summarizing and discussing the main results. Also possible directions for future research are discussed.

1.2 Fundamental Concepts in Data Mining

A definition of data mining is given as "finding interesting trends or patterns in large datasets, in order to guide decisions about future activities" in [43]. A more technical definition can be found in [44] as 'the process of employing one or more computer learning techniques to automatically analyze and extract knowledge from data contained within a database'. The knowledge extracted from the data will be a model or generalization that represents the characteristics of the data. All data mining methods involve a process forming general concept definitions by observing specific examples of concepts to be learned, also called induction-based learning [44]. There are various types of data mining algorithms that carry out different tasks. The main idea of these algorithms is fitting a model to the data or finding a generalization of the data. The aim is determining a model (or a generalization) which is closest to the characteristics of the data in terms of some criteria preferred. There are particular types of data mining problems that can be summarized in five groups:

*classification

*prediction

*estimation

*association mining

*clustering

The first three are all examples of supervised data mining, where the goal is to find the value of a particular target variable. Association mining and clustering are unsupervised tasks where the goal is to uncover structure in data without respect to a particular target variable. Now let us briefly define these problems. Classification, is the process of assigning instances to categories or classes that are predetermined. The aim is building models that are able to assign new instances to one set of predetermined classes or categories. Further on we will be mentioning the

classification problem in detail. Estimation, similar to classification, aims to determine a value for an unknown output attribute. The unlikely part is that in estimation the output attribute(s) are numerical rather than categorical [44]. Prediction uses data from the past to predict what is likely to happen in the future. Difference of prediction is that building a predictive model requires separation in time between the model inputs or predictors and the model output, the thing to be predicted, unlike classification or estimation. If this separation is not maintained, the model will not work [8]. Association mining/ Association rule mining finds interesting associations and/or correlation relationships among large set of data items. Association rules show attribute value conditions that occur frequently together in a given dataset. A typical and widely-used example of association rule mining is Market Basket Analysis. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature. Clustering is the task of segmenting a heterogeneous population into a number of more homogeneous subgroups or clusters [8]. In clustering, there does not exist any predefined classes. The model built with this method groups the data instances together based on a similarity scheme, a measure of cluster quality. The aims is discovering structures in the data and the meaning of the formed clusters is up to the user. There are many methods introduced to solve these data mining problems in the literature. We will be briefly explaining the most common data mining methods. The Nearest Neighbor Algorithm is one of the first algorithms used to determine a solution for data mining problems, particularly the classification problem. For a given point in space, the nearest neighbor algorithm selects the value of the attribute of the nearest point and does not consider the values of other points in the space. This is a simply implemented method which leads us to the k-Nearest Neighbor Algorithm. In this method an instance is classified by a majority of its neighbors, in other words it is assigned to the most common class amongst its k nearest neighbors [18]. A method introduced for all supervised data mining problems is called Decision Trees. A decision tree is a tree where the root and each internal node is labeled with a question. The arcs emanating from each node represent each possible answer to the associated question. Each leaf node represents a prediction of a solution to the problem under consideration [20]. Another method, Regression, is a supervised learning technique that generalizes a set of numeric data

by creating a mathematical equation relating one or more input attributes to a single numeric output attribute [44]. Regression is generally used to predict future values based on past value. Another statistics based method used for classification problem is called Naive Bayes. It assumes that the contribution by all attributes are independent and that each contributes equally to the problem. Classification is based on the conditional probability of each independent attribute determined by Bayes rule of conditional probability. Another method introduced for data mining problems is Artificial Neural Network. It is a set of interconnected nodes designed to imitate the function of the human brain [44]. It is applicable for supervised data mining problems as well as unsupervised ones. Briefly called Neural Network, it is an information processing system that consists of a graph representing the processing system as well as various algorithms that access that graph [20]. Association rules, introduced for the problem Association mining, are used to discover interesting associations between attributes. They provide information in the form of if-then statements. Genetic Algorithms, used for clustering, prediction and association mining, are examples of evolutionary computing methods. In this approach a starting model is assumed and through many iterations, models are combined to create new models. The best of these, as defined by a fitness function, are then put into the next iteration [20]. The latest introduced method for data mining problems is called Support Vector Machines method. It can be applied to both classification and clustering problems and it is based on the theoretical frame of optimization [49].

After brief definitions of data mining problems, let us focus on the classification problem. Classification is the most popular data mining problem that has various applications in real life problems, including image and pattern recognition, medical diagnosis, detection of frauds, etc. As mentioned above, classification consists of two phases which are called training phase and test phase, respectively. The dataset used in the first phase to create a learning model is called training set. Training set consists of the input data as well as the class assignment data (called class label) for each instance. In the second phase, test set which contains the new and unseen instances, is used. Test set consists of only the input data and the aim is determining class label for each instance in the test set. When the model is constructed in the first phase,

two main issues are considered. The first one is called the generalization ability of the model meaning the ability of a learning machine to classify new examples correctly that differ from those used for training. In other words, how much we are mislead in choosing the optimal function when generalizing from a given training example to a general prediction function determines the generalization error, which is the measure of the generalization ability of the model. The second issue is the empricial error, also called observed error, which is the frequency of errors made during the testing phase for a model. The main problem in the construction of the model is how to balance the trade-off between generalization error and the empirical error of the model since they have conflicting objectives. When too much importance is given to the empirical error, without considering the generalization error, overfitting occurs. Overfitting is the case where the learned model fits almost exactly to the training data, however in this case the learned model performs poorly on the test data since it just learned the training set instead of generalizing a pattern from it. On the other hand when generalization error is too much emphasized that may lead to underfitting. Underfitting is the case where the model is same whatever the input data is and the model has no attempt to fit the data. If the empirical error is too high and generalization error is too low in the training set this would lead a bad performance in the test set. Hence, it is important to balance the trade-off between the generalization error and the empirical error while constructing a learning model. All classification methods deal with this balancing problem in various ways. Our study focus on a particular classification method Support Vector Machines. Therefore, we will be giving relevant background information about its application for classification problem in the next section.

1.3 Support Vector Machine Classification

In SVM Classification, the problem is defined as classifying the points into two sets, referred, I^+ and I^-, which is also called binary classification. The training set consists of ℓ points in $\Re^n$, meaning n-dimensional real space, with a class assignment information. The training set can be

represented as

$$(x_1, y_1), ..., (x_\ell, y_\ell) \, , x_i \in \Re^n \, , y \in \{+1, -1\} \tag{1.1}$$

where x_i is the i^th instance in the training set and y_i is the class assignment label of this instance. Remark that the test set consists of k points in $\Re^n$ and goal of classification is to assign these points to either I^+ or I^- by using the model learned on the training set. First suppose the training data given as in (1.1) can be separated by a hyperplane of the form,

$$\mathbf{w}.\mathbf{x} + b = 0 \ , \ \mathbf{w} \in \Re^n \ , \ b \in \Re \tag{1.2}$$

If this separation is completed without error and if the distance of closest point to the hyperplane is maximal then this hyperplane, characterized by $\mathbf{w}$ and b, is called the Optimal Hyperplane [49]. The optimal hyperplane, also called the maximal margin hyperplane, is described with the inequalities:

$$\mathbf{w}.\mathbf{x}_i + b \geq 1 \ if \ y_i = +1 \tag{1.3}$$

$$\mathbf{w}.\mathbf{x}_i + b \leq 1 \ if \ y_i = -1 \tag{1.4}$$

and a brief notation is given as:

$$y_i[\mathbf{w}.\mathbf{x}_i + b] \geq 1, \ i = 1, \ldots, \ell, \tag{1.5}$$

The geometric representation of these equations is in the Figure 1.1,

where margin is represented. The filled instances of each class represent the support vectors which we would mention later. Margin is the width that the boundary could be increased before hitting a data point. The optimal hyperplane maximizes the distance between the hyperplane and 'the difficult points' which are points closest to the decision boundary. Since the distance

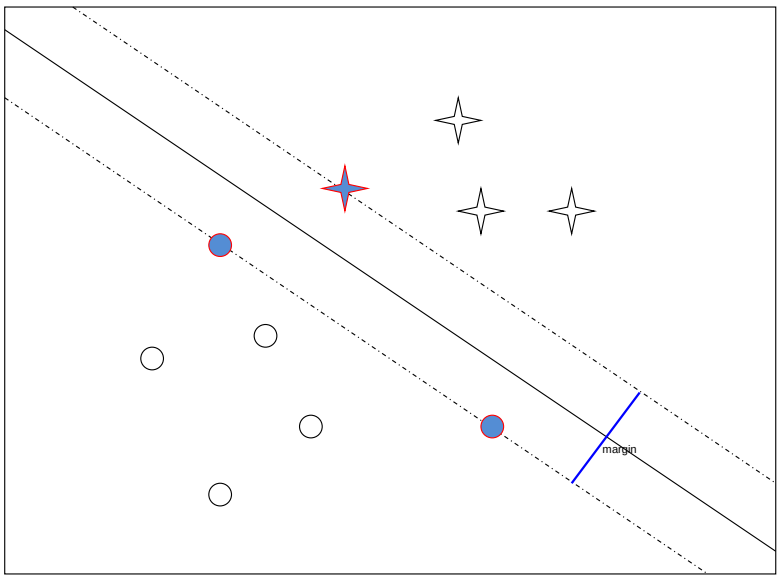

Figure 1.1: Efficient frontier examples in 3-D.

of an example from the hyperplane is given with the formula,

$$d_i = y_i.\frac{(\mathbf{w}.\mathbf{x}_i + b)}{\|w\|_2} \tag{1.6}$$

where $\|\|_2$ denotes the Euclidean norm. The geometric margin represented in Figure 1.1 is found as

$$\frac{2}{\|\mathbf{w}\|_2} \tag{1.7}$$

where $\mathbf{w}$ is the weight vector defining the hyperplane. The objective of this classification is to separate two classes as much as possible and in order to accomplish this the margin should be maximized. If the objective is formulated it will be maximizing Equation 1.7 and this leads to minimizing the inverse of geometric margin equation, in other words minimizing $\|\mathbf{w}\|_2$. Now let us summarize these findings.

Proposition. Given a linearly separable training set

$$X = (x_1, y_1), \ldots, (x_\ell, y_\ell) \tag{1.8}$$

the hyperplane characterized by $(\mathbf{w}, b)$ that solves the optimization problem

$$(HSVM) \qquad \min \frac{1}{2}\mathbf{w}.\mathbf{w} \tag{1.9}$$

$$s.t. \quad y_i((\mathbf{w}.\mathbf{x}_i) + b) \quad \geq \quad 1, \quad i = 1, \ldots, \ell, \tag{1.10}$$

$$\tag{1.11}$$

constructs the optimal hyperplane with geometric margin given as (1.7).
(HSVM), abbreviated for hard-margin SVM formulation, is a quadratic optimization problem in which separation is perfect. The solution to this optimization problem is given by Lagrangian Dual method by transforming the problem into its corresponding dual. The Lagrangian of the problem is constructed as,

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_{i=1}^{\ell} \alpha_i [y_i(\mathbf{w}.\mathbf{x}_i + b) - 1] \tag{1.12}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. The dual problem is maximization of the Lagrangian with respect to α_i's. As it is clear, minimization problem (HSVM) with respect to ($\mathbf{w}$,b) is equivalent to the maximization of Lagrangian with respect to α_i's by the Duality Theorem [6]. The dual solution is found by differentiating the Lagrangian with respect to $\mathbf{w}$ and b and then substituting the findings into the primal Lagrangian we end up with the following proposition [17]:

Proposition. Given a linearly separable training set

$$X = (x_1, y_1), \ldots, (x_\ell, y_\ell) \tag{1.13}$$

and supposing the parameters α^* solve the following quadratic optimization problem

$$(Dual - HSVM) \qquad \min \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (x_i.x_j) \tag{1.14}$$

$$s.t. \quad \sum_{i=1}^{\ell} y_i \alpha i = 1, \quad i = 1, \ldots, \ell, \tag{1.15}$$

$$\alpha_i \geq 0, \ i = 1, \ldots, \ell \tag{1.16}$$

Then the weight vector

$$w^* = \sum_{i=1}^{\ell} y_i \alpha_i^* x_i \tag{1.17}$$

constructs the optimal hyperplane with geometric margin given in (1.7) .
Representation of w is sparse since many of the *alpha* values end up being zero. Thus the decision boundary characterized by w will be determined by data points only with non-zero *alphas*. These data points that are closest to the boundary are called Support Vectors.

Remark that b does not appear in the dual formulation, hence b^* would be found by using primal constraints.

Even though (HSVM) is a sophisticated model with a strong theoretical background, it generally can not be used in real world datasets because of the noisy structures of these datasets. HSVM produces a perfectly consistent hypothesis and does not consider training errors [17]. In HSVM the bound on margin pursues the hyperplane to separate the data perfectly. However, in a real world data with noise, the primal problem would become infeasible and the dual problem would become unbounded. Therefore the problem can not be solved. For a more robust formulation, slack variables are introduced in order to allow the margin constraints to be violated within some bound

$$y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 - \xi_i, \quad , i = 1, \ldots, \ell \tag{1.18}$$

$$\xi_i \geq 0, \quad i = 1, \ldots, \ell \tag{1.19}$$

This formulation allows some instances to fall within the margin, however it penalizes misclassifications. The goal is still placing the hyperplane far from each class but also trying to minimize the training set error at the same time. Since minimizing the number of misclassifications is proven to be a NP-complete problem [39], the objective function chooses to minimize the sum of distances from the margin. Hence the optimization problem, soft-margin SVM, becomes,

$$(SSVM) \qquad \min \mathbf{w}.\mathbf{w} \quad + \quad C\sum_i \xi_i \tag{1.20}$$

$$s.t. \quad y_i(\mathbf{w}.\mathbf{x}_i + b) \quad \geq \quad 1 - \xi_i, \quad i = 1, \ldots, \ell, \tag{1.21}$$

$$\xi_i \quad \geq \quad 0, \quad i = 1, \ldots, \ell, \tag{1.22}$$

where C is the trade-off parameter between the non-compromising objectives maximizing margin-generalization error- and minimizing sum of ξ_i's-empirical error-.

1.3.1 L_1 Norm SVM Formulation

Throughout various formulations of SVMs (SSVM) is the most popular one [17],[49]. In SSVM L_2 norm is used in order to measure the generalization ability through the margin as also seen from the formulations we explained above. Meanwhile, let us remind some basic properties of the norm of a vector. Briefly norm is a function that assigns a strictly positive length to all vectors in a vector space. The most common norm used is L_2 Norm, also called Euclidean Norm, is given with the formula,$x \in R^n$

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} = \sqrt{\mathbf{x}.\mathbf{x}} \tag{1.23}$$

L_1 Norm, also called Manhattan Norm, is given as:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| \tag{1.24}$$

and finally the generalized form, Lp Norm, is given as,

$$\|\mathbf{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}} \tag{1.25}$$

where $p \geq 1$ is a real number. Even though L_2 Norm is the most generally used among the norm functions, alternative formulations for SVMs also exists using L_1 Norm instead of L_2 Norm. As discussed in [10], there are two principal reasons leading to this choice,

1. The problem is reduced to a linear program with important theoretical properties benefiting usage as a computational tool

2. L_1 Norm formulation is less sensitive to outliers and performs feature selection better than the L_2 Norm [2].

Bradley and Mangasarian [10] proposed the L_1 Norm SVM formulation as,

$$(L_1SVM) \qquad \begin{aligned} min & \sum_{i=1}^{n} |w_i| + C \sum_{i=1}^{n} \xi_i \\ s.t. & \quad (1.21), (1.22) \end{aligned} \tag{1.26}$$

Our choice of L_1 Norm depends mainly on the reduction of the mathematical programs to linear programs and also the findings in [2] that could be summarized as the classifier performance of L_1 Norm formulation is no worse than the classifier performance of L_2 Norm formulation.

Chapter 2

LITERATURE REVIEW

2.1 The Imbalanced Datasets

The class imbalance problem, also known as Imbalanced Dataset Problem is a remarkable challenge that is studied in the knowledge discovery and data mining framework as mentioned in [34] and [13]. The majority of the classification algorithms assume training sets are well balanced and misclassification costs are equal. The aim of these algorithms are usually maximizing the accuracy. However this balance in datasets usually does not exist in real-life datasets. Instead, one of the classes can be represented by only a few examples where the other class is represented by a large number of examples and problem in classification arises when class of interest is rare in the sample set. In imbalanced datasets, the class having more examples is called the majority class and the one having fewer examples is called the minority class. Without loss of generality, we assume the majority class consists of negative examples of binary classification problem and the minority class consists of positive examples of binary classification. Also, in this study classification problem is considered as binary classification since multi class problems are typically solved by converting into binary class problems.

Some of the application domains in which imbalance in data exists are, detection of fraud in telephone calls [25], detection of credit card fraud [11], response rate in direct marketing [38], rare medical diagnoses [53], detection of oil spills in satellite radar images [36] and spotting unreliable telecommunication customers [23]. The ratio of imbalance in these datasets may vary from 1:10 to more dramatic situations like 1:100000 [42].

For an application domain like credit card fraud detection, search is focused only on the minority class. In other words the aim of the classification is distinguishing the minority class. Furthermore, considering diagnosis of a cancer, classifying a cancer case as non-cancer is much

more serious than a false positive alert meaning diagnosis of a non-cancer case as cancer. However, when there is an imbalanced distribution in the dataset and misclassification of minority is more serious than misclassification of majority, the problem arises. Since the aim of a standard classifier is maximizing the overall accuracy, the classifier would be biased toward one class and the outcome of this training would be, 'low error rate for majority class' but 'unacceptable error rate for the minority class'. In other words the classifier would tend to classify all examples as majority leading to a high accuracy but in the meantime it would miss minority examples. Let us demonstrate the problem in a figure via a classifier line. Now assume we have the dataset in Figure 2.1, where black dots represents the majority class and white rectangle represents the minority class. The line represents the boundary that ideally separates the two classes with allowing a misclassification. However a non-biased classification algorithm would learn a boundary as shown in Figure 2.2 which misses the minority instance since it is cheaper to classify it as majority.

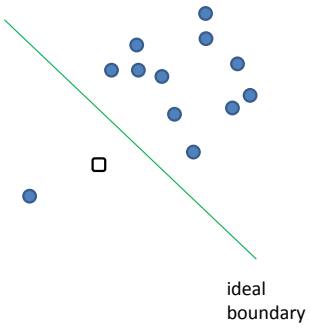

Figure 2.1: Ideal Classification.

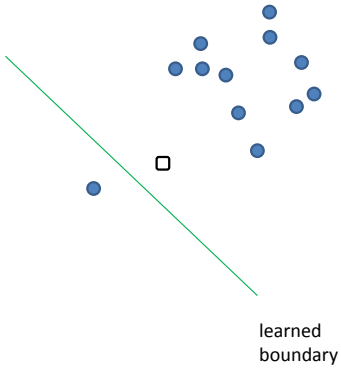

Figure 2.2: Classification when imbalance is not considered.

2.2 Handling the Imbalanced Datasets

While the Imbalanced Dataset Problem becomes popular, many strategies are proposed in order to handle this problem. Surveys [34], [51], [13], [35], [52], [28] are suggested for detailed information. Mainly, these proposed methods and approaches are covered in two parts, on data level and on algorithm level, and most important ones are discussed below.

2.2.1 Data Level Solution Approaches

The first idea that comes into mind in order to handle an imbalanced dataset, is to adjust the class distributions by resampling the data. Data level approaches are based on this idea and their objective is balancing the distribution of the data by resampling. These methods are in various forms like random undersampling, random oversampling and some improved sampling techniques.

Random undersampling aims to rebalance the dataset by eliminating instances of the majority

class until the class distributions are equal. The major drawback is undersampling may throw useful data which hurts the data mining process. Also, the randomness of the sample space is hurt in case of undersampling. Random oversampling replicates the instances of minority class until equivalent class distributions are achieved. Since most of the random oversampling methods copy the instances of minority class until balance is achieved, likelihood of overfitting occurrence increases [28]. Additionally, random oversampling may be computationally very expensive in case of a large dataset with a highly imbalanced distribution. There are also two other non-improved sampling techniques called focused undersampling and focused oversampling. In focused undersampling the majority class instances that are lying further away from the decision boundary are eliminated. In focused oversampling minority instances that occurred on the boundary between the majority and minority classes are eliminated. In [31] Japkowicz showed that sampling techniques are effective however the focused sampling techniques did no better than the random sampling techniques in the considered domain.

There are some other sampling methods that follow a heuristic in rebalancing the distributions, which we call generally improved sampling techniques. Kubat and Matvin [37] proposed One-Sided Selection method that undersamples the majority class by removing noisy, borderline and redundant instances and only safe instances are kept. Noisy instances are defined as mislabeled ones, redundant instances are the ones that are already represented by other instances in the training set and finally borderline instances are the ones that are too close to the decision boundary. In order to detect noisy and borderline instances Kubat and Matvin suggested to use Tomek Links [46]. Given two instances x and y from different classes and distance d(x,y), (x,y) is called a Tomek link if there is no instance z such that $d(x, z) < d(x, y)$ or $d(y, z) < d(y, x)$. If two instances form a Tomek link, then either one instance is noise or both instances are borderlines. Afterwards, they suggest to remove the redundant instances by CNN (Closest Nearest Neighbor) method. Later based on the method mentioned above Batista [5] proposed a more sophisticated sampling technique. However, the loss of information continues in case of One-Sided Selection and also this method is costly because of the calculation of Tomek links. Another approach to be examined in this part is the method proposed by Ling and Li [38] which combines oversampling

of minority class with undersampling of majority class. However this method shows no significant improvement on the other methods. Chawla et al. [12] proposed an over-sampling method SMOTE that oversamples the minority class by generating synthetic minority class instances instead of replicating the existing instances. For each instance from the minority class, its k nearest neighbors from the same class are determined and then due to some over-sampling rate some of these instances are randomly selected. Then, new synthetic instances are generated along the line between the minority instances and their selected nearest neighbors. This process avoids the over-fitting problem and is shown to result in better classification performance than other sampling techniques [12],[1].

2.2.2 Algorithm Level Solution Approaches

In this part, first we will discuss approaches that are based on cost-sensitivity and then some other approaches proposing algorithms or heuristics for the Imbalanced Dataset Problem. We will be focusing on the methods improving SVMs, however some methods regarding the other classifiers like k Nearest Neighborhood, Naive Bayes, Neural Networks and decision trees can be found in [4],[52], [32], [15], [57], [33] and in their references. Another method apart from changing the class distributions is defining fixed and unequal misclassification costs in the process of decision making. The aim of cost sensitive classification is minimizing the misclassification cost while seeking to minimize error rate. In the model, misclassification costs are defined by a cost matrix C, with entry $C(i, j)$ expressing the cost of predicting that an instance is in class i when it belongs to the class j. Also in the matrix diagonal elements are zero, meaning true classification has no cost. Assuming $P(j|x)$, the probability of each class j for a given instance x is known, the Bayesian optimal prediction class i for instance x minimizing the conditional risk is given with the formula below,

$$R(i|x) = \sum_{j} C(i, j)P(j|x) \tag{2.1}$$

In other words it is the expected cost of predicting that instance x belongs to the class i. Domingos [19] presented a general method, MetaCost, to make classifiers cost sensitive. The idea depends on finding a way to estimate class probabilities $P(j|x)$. Another work on cost sensitive classification can be found in [22].

The algorithm based approaches regarding SVMs are generally proposed for binary classification problems with the idea of biasing the algorithm so that the learned hyperplane discriminates the minority class better. When SVMs are applied to an imbalanced dataset the algorithm tries to classify all instances in minority class as majority class since it would be less costly because it finds the largest margin possible. For a detailed explanation and a comprehensive analysis the reader is referred to the work of Wu and Chang in [54]. In [1], a method that is a combination of SMOTE by Chawla et.al. [12] with Different Costs (DC) Algorithm by Veropoulos [50] is presented. They first start with mentioning the fact that undersampling results in information loss. However simple oversampling does not also perform well enough because it resamples the minority instances on top of each other and this does not improve the boundary definition. With SMOTE, positive instances are more dense resulting in a well defined boundary. Also by using DC in the SVM formulation, the boundary is pushed away from positive instances by penalizing errors of different classes differently. They evaluated their method and also SMOTE, DC, Regular SVM and Regular Undersampling on 10 UCI Datasets. They show that their method outperforms all others in terms of sensitivity, specificity and g-means metrics. In [14], they proposed a method based on SVM classification and backward pruning technique. The idea is to improve classification rate of minority class while sacrificing the rate for the other class at minimum. Training a SVM on the set of all SVs should produce the same decision boundary as by using all training examples. Also SVs are the representatives for classification of the class they belong to, so removing SVs from a particular class result in more misclassified samples in that class and more correctly classified samples in the other class. With these ideas, the method proposed in this paper first trains SVM on the whole dataset. After finding SVs, a new SVM training is done on the set that consists of SVs of majority class and whole minority class samples. Then the procedure continues with removing a SV each time until an optimal set is

achieved. While choosing the SV which will be removed, a criterion function J=(minority class accuracy)/(majority class error) is used to order SVs from worst to best. Finally the decision boundary will be distorted as minority classification rate is improved and majority error rate does not increase dramatically. The method is evaluated on two real-world datasets and it is shown that their method outperforms the cost weighting based methods. Another work on improving SVM performance in imbalanced datasets is proposed in [54]. The method proposed, The Class-Boundary-Alignment Algorithm, adjusts the boundary with an alignment in kernel in other words updates the kernel to develop bias. Other work on improving SVM performance can be found in [16], [21], [45]

Chapter 3

ALTERNATIVE METHOD HANDLING THE IMBALANCE

The theory of SVM is based on the idea of balancing the generalization ability of the classifier and the empirical error observed on the data. Generalization ability is characterized by margin whereas empirical error is characterized by the sum of slack variables in a soft margin SVM. Briefly, SVM has two objectives: maximizing the margin and minimizing the sum of slack variables. In the standard formulation these objectives are combined with weighted sum method via a trade-off parameter. Our formulation approach is based on splitting the empirical error into two parts, analyse as the empirical error of the majority class and the empirical error of the minority class. With this approach we will be able to emphasize the empirical error of the minority class as much as needed.

In this case, the formulation for the SVM problem will be a multicriteria (in fact tricriteria) optimization problem. The solution to a multicriteria optimization problem consists of the set of efficient solutions. Let us remind the definitions of efficiency.

Consider the multicriteria optimization problem of

$$(MCOP) \qquad \min(f_1(x), \ldots, f_Q(x)),$$
$$s.t. \quad x \in X$$

For $i = 1, \ldots, Q$, let $y_i^* = min_{x \in X} f_i(x)$ denote the optimal value for each individual objective function. Then the point $y^* = (y_1^*, y_2^*, \ldots, y_Q^*)$ is called the ideal point for this problem. If $y^* = f(x^*)$ for some $x^* \in X$ then x^* is a solution to this problem. However these objective functions generally conflict with each other. For instance, in the SVM case, objectives of minimizing the

norm and minimizing the sum of slack variables conflict with each other and a trade-off is needed for a solution.

Definition: $x^0 \in R^n$ is an efficient solution for the multicriteria optimization problem if $x^0 \in X$ and there exists no $x \in X$ such that $f(x) \leq f(x^0)$ with $f_j(x) \leq f_j(x^0)$ for at least one $j \in 1, \ldots, Q$.

In real life problems conflicting objectives are common and there generally exist a continuum of efficient solutions. The collection of all efficient solutions is called the efficient set.

One of the solution approaches proposed in the Operations Research literature to the multicriteria optimization problem is creating a non-negative weighted combination of all of the objective functions, which is indeed what the standard SVM formulation implements. The weighting factor acts as a trade-off parameter between the objectives. In other words, for a particular value of this parameter the solution to the problem is an efficient solution. However the efficient set may consist of a continuum of points, it is not an easy task to determine the appropriate value of the trade-off parameter leading to a preferred solution. In practice, when the standard SVM formulation is used, the trade-off parameter is chosen after a series of experimentations. Another well known alternative approach, handling the multicriteria optimization problem is the ϵ-constraint method. In this method, one of the original objectives is chosen to be minimized while others are converted into constraints. Considering the problem $(MCOP)$, the converted ϵ-constraint problem becomes,

$$\begin{aligned} \min \quad & f_k(x) \\ & s.t. x \in X \\ & f_j(x) \leq \epsilon_j for j = 1, \ldots, Q, j \neq k \end{aligned}$$

where $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_Q) \in R^Q$.

All the approaches that handle the imbalanced datasets in the literature, which we mentioned before, choose a type of evaluation metric and conduct their experiments with a limited -chosen-

set of parameters. However, we aim to compute a more systematic sample of the efficient frontier by using results in multicriteria optimization summarized above and report the results with more than one metric in order to allow the decision maker to choose the result that is the most convenient for him.

The fundamental idea of SVM classification is separating two classes as much as we can while limiting the misclassifications, in other words maximizing the margin while minimizing the sum of distances of misclassified instances to the margin. As mentioned our approach examines the sum of distances of misclassifications for each class separately in order to control them accurately. Therefore considering the three objectives the multi criteria formulation becomes,

$$
\begin{aligned}
(SVM-3C) \qquad \min\ & e^T(w^+ + w^-), \sum_i \xi_i^-, \sum_i \xi_i^+ \\
s.t.\ \ & y_i((w^+ - w^-)^T x_i + b) \geq 1 - \xi_i^-, \quad i = 1, \ldots, \ell_-, && (3.1)\\
& y_i((w^+ - w^-)^T x_i + b) \geq 1 - \xi_i^+, \quad i = \ell_- + 1, \ldots, \ell, && (3.2)\\
& \xi_i^- \geq 0, \quad i = 1, \ldots, \ell^-, && (3.3)\\
& \xi_i^+ \geq 0, \quad i = \ell^- + 1, \ldots, \ell, && (3.4)\\
& w^+, w^- \geq 0 && (3.5)
\end{aligned}
$$

where $\mathbf{e}$ is the vector of 1's, ℓ^- is the number of instances that belong to the majority class, ℓ^+ is the number of instances that belong to the minority class and $\ell = \ell_- + \ell+$ is the number of all instances in the dataset. First notice that we used L_1 Norm formulation to have LP formulation. Here objective is formulated by considering the fact that L_1 Norm formulation is no worse than L_2 Norm in certain conditions, as mentioned before, and computationally remarkable benefits of using L_1 Norm. The idea motivating this work is briefly to control the trade-off among all three objectives, $e^T(w^+ + w^-), \sum_i \xi_i^-, \sum_i \xi_i^+$ without assigning some cost parameters beforehand. Also another purpose is to let the decision maker prefer any solution from the efficient frontier. Applying these definitions to the formulation $(SVM - 3C)$, a solution $(\bar{w}^+, \bar{w}^-, \bar{b}, \bar{\xi^-}, \bar{\xi^+})$ is

efficient if for any feasible solution $(w^+, w^-, b, \xi^-, \xi^+)$, $e^T(w^+ + w^-) < e^T(\bar{w}^+ + \bar{w}^-)$ implies $\sum_i \xi_i^- > \sum_i \bar{\xi}_i^-$ or $\sum_i \xi_i^+ > \sum_i \bar{\xi}_i^+$. The set of all efficient outcomes $(e^T(\bar{w}^+ + \bar{w}^-), \sum_i \bar{\xi}_i^-, \sum_i \bar{\xi}_i^+$ generate the *efficient frontier*. However, for the three criteria case the efficient frontier would be a subset of the boundary of a three-dimensional convex polyhedron, briefly a piecewise surface such as the one demonstrated in Figure 3.1 In that case computing the whole efficient

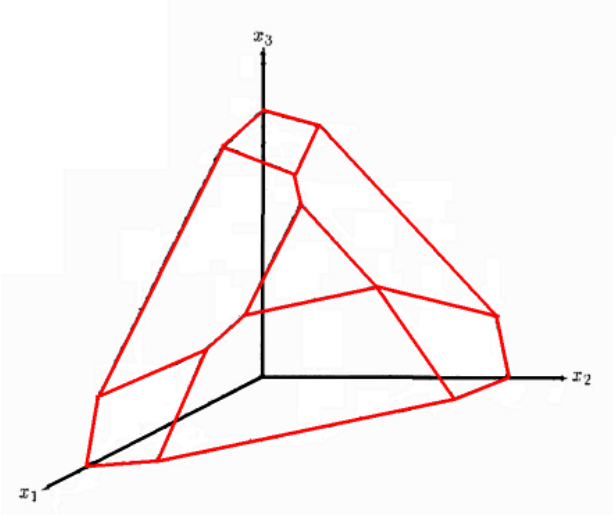

Figure 3.1: Efficient frontier examples in 3-D.

frontier would be an involved task and computationally expensive [27]. However for two criteria optimization case it is shown in [7] that a full characterization of the efficient frontier may be obtained. A representation of the two criteria case is shown in Figure 3.2.

Therefore we will propose some alternative two criteria reductions of $(SVM - 3C)$ in order to solve the problem.

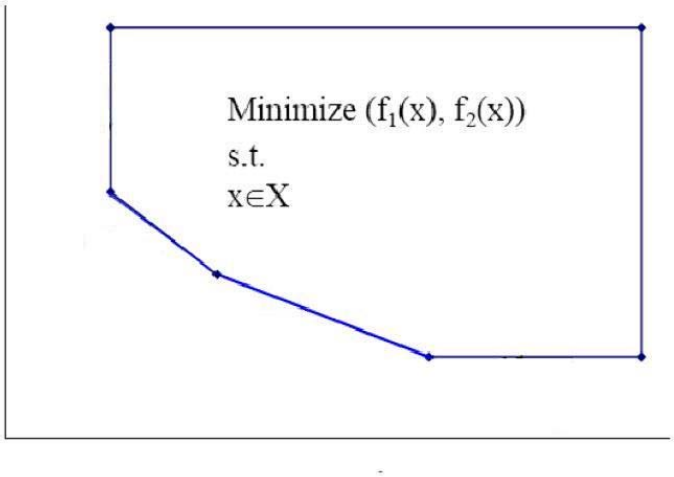

Figure 3.2: Efficient frontier for a bicriteria optimization problem.

On the other hand remark that the standard cost-sensitive SVM formulation is given as (CS-SVM) formulation in the literature([50]).

$$
\begin{aligned}
(CS-SVM) \qquad \min \quad & \mathbf{w.w} + C^- \sum_i \xi_i^- + C^+ \sum_i \xi_i^+ & \\
s.t. \quad & y_i((w^+ - w^-)^T x_i + b) \geq 1 - \xi_i^-, \quad i = 1, \ldots, \ell^-, & (3.1)\\
& y_i((w^+ - w^-)^T x_i + b) \geq 1 - \xi_i^+, \quad i = \ell^- + 1, \ldots, \ell, & (3.2)\\
& \xi_i^- \geq 0, \quad i = 1, \ldots, \ell^-, & (3.3)\\
& \xi_i^+ \geq 0, \quad i = \ell^- + 1, \ldots, \ell, & (3.4)\\
& w^+, w^- \geq 0 & (3.5)
\end{aligned}
$$

Converting this problem into LP by using L_1 Norm and the transformation $w = w^+ - w^-$ we

end up with the cost-sensitive SVM formulation with L_1 Norm;

$$(CSSVM - L_1) \qquad \min \quad e^T(w^+ + w^-) + C^- \sum_i \xi_i^- + C^+ \sum_i \xi_i^+$$
$$s.t. \qquad (3.1), (3.2), (3.3), (3.4), (3.5)$$

The above formulation $(CSSVM - L_1)$ is actually the $(SVM - 3C)$ problem in which the weighted combination method of multi criteria solution approach is applied. We will state that after solving $(SVM - 3C)$, obtaining the corresponding values of C^-, C^+ from the $(CSSVM - L_1)$ formulation is possible based on Theorem 1 after explaining our approach in detail in Section 3.2.

3.1 Obtaining Efficient Frontier in the Two Criteria Case

Recently, in [3] a systematic procedure to obtain the entire efficient frontier by adjusting ϵ is proposed for the L_1 Norm SVM. Considering the objective of maximizing the margin as minimizing the norm the multi objective SVM is formulated as

$$(2C1N - SVM) \qquad \min e^T(w^+ + w^-), \sum_i \xi_i$$
$$s.t. \quad y_i((w^+ - w^-)^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, \ell, \tag{3.6}$$
$$\xi_i \geq 0, \quad i = 1, \ldots, \ell, \tag{3.7}$$
$$w^+, w^- \geq 0 \tag{3.8}$$

If the ϵ-constraint method is applied to $(2C1N - SVM)$, the resulting problem is

$$P(\epsilon) \qquad z(\epsilon) = \min \ e^T(w^+ + w^-)$$
$$s.t. (3.6), (3.7), (3.8)$$
$$\sum_i \xi_i \leq \epsilon \tag{3.9}$$

In [3] they suggest to start with solving the linear program $P(\epsilon)$ for a large and proper value of ϵ that is chosen in a certain way. The solution characterizes a hyperplane that minimizes the norm with an upper bound of ϵ on the empirical error. The corresponding basis will remain optimal for a certain interval as ϵ decreases. Therefore, using the theory of sensitivity analysis in linear programming, the lower end of this interval can be determined. At this point the basis would change and the same procedure would be followed until ϵ becomes zero or the problem $P(\epsilon)$ becomes infeasible. In practice, the proposed algorithm first chooses a sufficiently large and proper value of ϵ, let us name ϵ_0, and solves the problem $P(\epsilon_0)$ and collects this optimal solution to a set in the initialization step. Then in the main step, in each iteration k, the minimum value of the right-hand-side of the constraint (3.9) for which the optimal basis remains optimal is computed and named ϵ_k and the new optimal solution $(\epsilon_k, z(\epsilon_k))$ is collected into the set. Iteration procedure continues until the stopping criteria mentioned above is met.

While we control the change in the total error, the norm also changes due to the trade-off we mentioned. The two objectives, maximizing the norm and minimizing the empirical error are conflicting and we can see this nature clearly in the Figure 3.3. Figure 3.3 displays the total empirical error computed in the formulation with respect to the norm of the SVM classifier, which is the objective function value of the respective subproblem $P(\epsilon)$. This figure is obtained from the results of an experiment conducted with this method, introduced in [3], on the dataset Yeast. The description of this data set is given in 3.3.2. The selection of ϵ_0 is based on a theorem in [7]. It says when ϵ_0 is chosen so as to guarantee that $\epsilon_0 = \min_{|w|=0} \sum_i \xi_i$, the entire efficient frontier is obtained as the piecewise linear curve of the elements by accumulating each optimal solution pair in each iteration. When $|w| = 0$, $\xi_i = \max\{0, 1-b\}$ for $+1$ instances and $\xi_i = \max\{0, 1+b\}$ for -1 instances, thus $\sum_i \xi_i = \sum_{i \in I^+} \max\{0, 1-b\} + \sum_{i \in I^-} \max\{0, 1+b\}$, where $I+$ denotes the set of $+1$ instances and I^- denotes the set of -1 instances. The optimal choice for b is -1 if $\ell^+ < \ell^-$, $+1$ if $\ell^+ > \ell^-$ and $-1 \leq b \leq 1$ if $\ell^+ = \ell^-$,leading to $\sum_i \xi_i = 2 \times \min\{\ell^+, \ell^-\}$ where ℓ^- is the number of instances of majority class and ℓ^+ is the number of instances of minority class. From now on, we will refer this method *2C* throughout the thesis. Motivating from this idea, we will be introducing some two criteria reductions and solution methods in the following section.

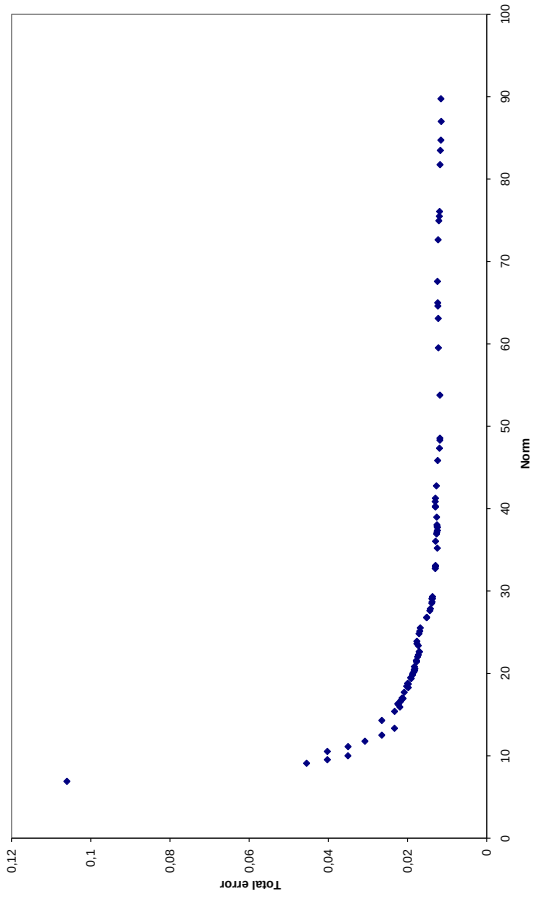

Figure 3.3: Objective value vs. Total error.

3.2 Grid Search Model Reducing Three Criteria to Two Criteria

One of the alternative reductions we propose is to convert the majority class empirical error objective to a constraint and set to a fixed value as a first step so the problem will be reduced to a bicriteria optimization problem. Next step is to convert the minority class empirical error objective to an ϵ-constraint with a sufficiently large and appropriate chosen ϵ. Then the algorithm proposed for the two criteria case can be adapted for this case [3]. Briefly, for a fixed value of majority class empirical error an exhaustive search would take place for the minority class empirical error value. The main idea completing this procedure is dividing the right-hand side value corresponding to the majority class, ϵ_1 into grids starting from a sufficiently large and proper value. Each grid line means fixing ϵ_1 to the number corresponding to this grid. After fixing the epsilon of majority, the procedure searches the entire efficient frontier by means of the right-hand-side value corresponding to the minority class, ϵ_2. First the linear program is solved for a sufficiently large and proper value of ϵ_2 (ϵ_1 is given with grid information). The solution would be a hyperplane minimizing the L_1 Norm with an upper bound of ϵ_2 on the empirical error for the given ϵ_1. The basis that corresponds to this solution will remain optimal for a certain interval as ϵ_2 decreases. At this point, a change of basis would take place, and the same argument may be repeated until ϵ_2 becomes zero or the problem becomes infeasible. When the procedure terminates for the fixed value of ϵ_1, one grid would be searched. Search of the grids would continue until ϵ_1 becomes zero or the problem becomes infeasible. We would be calling

this search method *grid1*. The Linear Program formulation can be represented as,

$$
\begin{aligned}
(P-3C) \qquad & \min\ e^T(w^+ + w^-) & & & \\
s.t. \quad & y_i((w^+ - w^-)^T x_i + b) & \geq & 1-\xi_i^-, \quad i=1,\ldots,\ell^-, & (3.10)\\
& y_i((w^+ - w^-)^T x_i + b) & \geq & 1-\xi_i^+, \quad i=\ell^-+1,\ldots,\ell, & (3.11)\\
& \xi_i^- & \geq & 0, \quad i=1,\ldots,\ell^-, & (3.12)\\
& \xi_i^+ & \geq & 0, \quad i=\ell^-+1,\ldots,\ell, & (3.13)\\
& w^+, w^- & \geq & 0 & (3.14)\\
& \sum_i \xi_i^- & \leq & \epsilon_1 & (3.15)\\
& \sum_i \xi_i^+ & \leq & \epsilon_2 & (3.16)
\end{aligned}
$$

Before continuing with the algorithm procedure, let us state a result which points out that the ϵ-constraint method and the cost-parameter approach are closely related, and obtaining the corresponding values of C^-, C^+ by applying our approach is possible.

Theorem 1 Let u_1^* and u_2^* denote the optimal value of the dual variable associated with constraints 3.15 and 3.16 respectively in $(P-3C)$. Then an optimal solution to $(P-3C)$ solves $(CSSVM-L_1)$ with $C^- = u_1^*$ and $C^+ = u_2^*$.

Proof. Consider the Lagrange relaxation function of $(P-3C)$

$$L(\epsilon_1,\epsilon_2,u_1,u_2) = \min_{s.t.(3.10),(3.11),(3.12),(3.13),(3.14)} e^T(w^+ + w^-) + u_1(\sum_i \xi_i^- - \epsilon_1) + u_2(\sum_i \xi_i^+ - \epsilon_2)$$

Let (u_1^*, u_2^*) denote an optimal solution for Lagrangian dual problem of $\max_{u\geq 0} L(\epsilon_1,\epsilon_2,u_1^*,u_2^*)$. Since $(P-3C)$ is a LP then the duality theorem implies that,

$$z(\epsilon_1,\epsilon_2) = L(\epsilon_1,\epsilon_2,u_1^*,u_2^*)$$

and for an optimal solution $(w^{-^*}, w^{+^*}, b^*, \xi_i^{-^*}, \xi_i^{+^*})$ to problem $(P-3C)$

$$e^T(w^{-^*} + w^{+^*}) + u_1^*(\sum_i \xi_i^{-^*} - \epsilon_1) + u_2^*(\sum_i \xi_i^{+^*} - \epsilon_2) = z(\epsilon_1, \epsilon_2)$$

Since

$$L(\epsilon_1, \epsilon_2, u_1^*, u_2^*) = \min_{s.t.(3.10),(3.11),(3.12),(3.13),(3.14)} e^T(w^+ + w^-) + u_1^*(\sum_i \xi_i^- - \epsilon_1) + u_2^*(\sum_i \xi_i^+ - \epsilon_2)$$

becomes

$$L(\epsilon_1, \epsilon_2, u_1^*, u_2^*) = \min_{s.t.(3.10),(3.11),(3.12),(3.13),(3.14)} e^T(w^+ + w^-) + u_1^*(\sum_i \xi_i^-) + u_2^*(\sum_i \xi_i^+) - u_1^*\epsilon_1 - u_2^*\epsilon_2$$

and $u_1^*\epsilon_1$, $u_2^*\epsilon_2$ are constant for given values of u_1^*, u_2^*, ϵ_1 and ϵ_2 evaluating $L(\epsilon_1, \epsilon_2, u_1^*, u_2^*)$ is equivalent to solving $(CSSVM - L_1)$ with $C_1 = u_1^*, C_2 = u_2^*$ that means $(w^{-^*}, w^{+^*}, b^*, \xi_i^{-^*}, \xi_i^{+^*})$ is an optimal solution for $(CSSVM - L_1)$.

From an algorithmic point of view, initialized from the starting values of ϵ_1 and ϵ_2, at each iteration for a fixed value of ϵ_1 an LP is solved in order to obtain the value of ϵ_2 for the next iteration. When termination occurs for this fixed value of ϵ_1, iteration continues with the next grid value of ϵ_1 and starting value of ϵ_2. The search procedure of the method *grid1* is represented in the majority epsilon vs. minority epsilon space in Figure 3.4. It is seen from the Figure 3.4, for a grid leg of ϵ_1 value ϵ_2 values are scanned the change of basis. The algorithm of the procedure is given below.

The Algorithm - *grid1*

Initialize. Set $\epsilon^0 = (\epsilon_1^0, \epsilon_2^0) = (\ell^-, \ell^+)$, where ℓ^- is the number of instances of class -1 and ℓ^+ is the number of instances of class $+1$. Solve P-3C with ϵ^0. Set $\mathcal{L} = \{\epsilon^0, z(\epsilon^0)\}$, set $k = 1$ and go to Step k.

Step k. Set $\epsilon_1^k = \epsilon_1^{k-1} - \frac{\epsilon_1^0}{gridsize} * (k-1)$

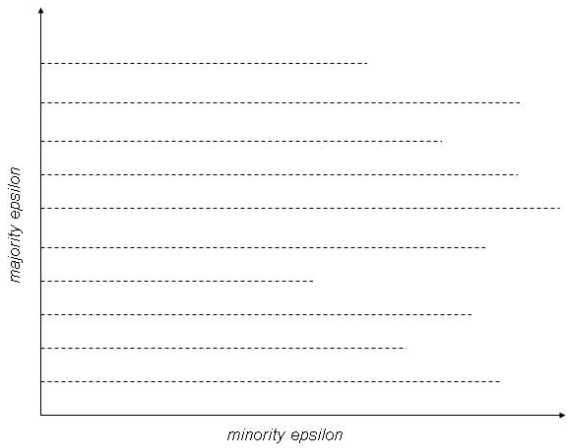

Figure 3.4: *grid1* Search in Majority Epsilon vs. Minority Epsilon Space

k.1 [**Initialize.**] Set $(\epsilon_2^k)^0 = \epsilon_2^0$, $l = 1$, go to Step l

k.2 [**Step l.**]

l.1. Solve $P-3C$ with $(\epsilon_1^k, (\epsilon_2^k)^l)$ and find u^{kl}, the minimum value of the right-hand-side of constraint (3.16) for which the optimal basis to P-3C remains optimal and set $\mathcal{L} = \mathcal{L} \cup \{(\epsilon^k)^l, z((\epsilon^k)^l)\}$.

l.2. If $u^{kl} = 0$, set $k = k + 1$.
If $k \leq gridsize$ go to Step k. Otherwise stop.

l.3. If $P(\epsilon_k)$ is infeasible, set $k = k + 1$.
If $k \leq gridsize$ go to Step k. Otherwise stop. Otherwise set $l = l + 1$ and go to l.1

l.4. If $u^{kl} \geq 0$, set $(\epsilon_2^k)^{(l+1)} = u^{kl} - \delta$, $l = l + 1$ and go to l.1

For implementation reasons, we introduce a tolerance parameter δ that controls the sensitivity of the basis change. After a P-3C problem is solved, new $(\epsilon_2^k)^l$ is set to minimum value of the right-hand-side of the related constraint in theory. However in practice, in order to get a new

basis we should set new $(\epsilon_2^k)^l$ to a value less than the lower bound found from the sensitivity analysis. Therefore, a tolerance parameter is used that assigns the new $(\epsilon_2^k)^l$ to the lower bound of the related constraint obtained minus the tolerance parameter value.

The choice of sufficiently large and proper starting values of ϵ_1 and ϵ_2 can be computed as, considering the constraints (3.10) and (3.11) when $|w| = 0$

$$\xi_i^- \geq 1 + b$$

$$\xi_i^+ \geq 1 - b$$

and since all $\xi_i \geq 0$

$$\xi_i^- = \max\{0, 1 + b\}$$

$$\xi_i^+ = \max\{0, 1 - b\}$$

are obtained. Thus,

$$\sum_i \xi_i^- = \sum_{i \in I^-} \max\{0, 1 + b\}$$

$$\sum_i \xi_i^+ = \sum_{i \in I^+} \max\{0, 1 - b\}$$

where I^- denotes the set of -1 instances and I^+ denotes the set of $+1$ instances. The optimal choice of $b = 0$ leads to sufficiently large and proper values for ϵ_1 and ϵ_2

$$\sum_i \xi_i^- = \left|I^-\right| = \epsilon_i^1$$

$$\sum_i \xi_i^+ = \left|I^+\right| = \epsilon_i^2$$

Also the same procedure is adapted for the case that the minority class empirical error is fixed to be searched in a grid and the majority class empirical error is searched exhaustively with the algorithm proposed. We would be calling this second method *grid2*.

In addition to the methods proposed above, another way to scan the majority epsilon vs. minority epsilon space is dividing both axes of the space into grids. In other words, instead of applying an exhaustive search on the minority epsilon axis we would also divide it into grids as majority epsilon axis. The algorithm of this method, called *gridS*, is given below:

The Algorithm - *gridS*

Initialize. Set $\epsilon^0 = (\epsilon_1^0, \epsilon_2^0) = (\ell^-, \ell^+)$, where n is the number of instances of class -1 and p is the number of instances of class $+1$. Solve P-3C with ϵ^0. Set $\mathcal{L} = \{\epsilon^0, z(\epsilon^0)\}$, set $k = 1$ and go to Step k.

Step k. Set $\epsilon_1^k = \epsilon_1^{k-1} - \frac{\epsilon_1^0}{gridsize} * k$

k.1 [**Initialize.**] Set $(\epsilon_2^k)^0 = \epsilon_2^0$, $l = 1$, go to Step l

k.2 [**Step l.**]

l.1. Solve $P - 3C$ with $(\epsilon_1^k, (\epsilon_2^k)^l)$ and set $\mathcal{L} = \mathcal{L} \cup \{(\epsilon^k)^l, z((\epsilon^k)^l)\}$.

l.2. If $(\epsilon_2^k)^l = 0$, set $k = k + 1$.

If $k \leq gridsize$ go to Step k. Otherwise stop.

l.3. If $P(\epsilon_k)$ is infeasible, set $k = k + 1$.

If $k \leq gridsize$ go to Step k. Otherwise stop.

Otherwise set $l = l + 1$ and go to l.1

l.4. If $(\epsilon_2^k)^l \geq 0$, set $(\epsilon_2^k)^(l+1) = (\epsilon_2^k)_1^l - \frac{(\epsilon_2^0)^1}{gridsize} * (l-1)$, $l = l + 1$ and go to l.1

3.3 Datasets

3.3.1 Soybean Dataset

Soybean is a UCI Dataset with 19 classes and 35 attributes. Training set consists of 307 examples while test set consists of 376 examples. For all datasets used in this study, the suffix after each dataset indicates the class chosen as the positive class, clearly all the other classes are chosen as the negative class. In Soybean12 dataset, there are some missing values. In order to handle this problem in the training set,we remove the examples with missing values and accept some loss of information. However for the test set, missing values are replaced with the average of each corresponding attribute. The properties of Soybean12 after these processes is listed below:

	Positive Examples	Negative Examples	Total
Soybean12	44	598	642
Soybean12 Training	20	246	266
Soybean12 Test	24	352	376

Table 3.1: Properties of Soybean Dataset

Soybean12 has an imbalance of 10:123(=0.0813) in the training set and 3:44(=0.0681) in the test set.

3.3.2 Yeast Dataset

Yeast is a UCI Dataset with 10 classes and 8 attributes. The original datasets consists of 1484 samples. In this experiment we choose two classes, as suggested in [14], ME3 for minority class with 163 examples and CYT for majority class with 463 examples. Then the data is divided into two sets, training and test. The properties of the data used is summarized in the table below: Yeast has an imbalance of 3:10(=0.3) in the training set and 88:213(=0.4131) in the test set. Yeast is clearly a more balanced dataset compared to Soybean12.

	Positive Examples	Negative Examples	Total
Yeast	163	463	626
Yeast Training	75	250	325
Yeast Test	88	213	301

Table 3.2: Properties of Yeast Dataset

3.3.3 Abalone Dataset

Abalone is a UCI Dataset with 29 classes and 8 attributes. The original datasets consists of 4177 samples and since there is no missing value in the data, all samples are used. In this experiment we choose class 19 as the positive class and all other classes as negative class, as suggested in [1]. Data is divided into two sets, training and test. Numerical properties of the dataset Abalone19 is summarized below:

	Positive Examples	Negative Examples	Total
Abalone19	32	4145	4177
Abalone19 Training	26	3454	3480
Abalone19 Test	6	691	697

Table 3.3: Properties of Abalone Dataset

Abalone19 has an imbalance of 13:1727(=0.007527) in the training set and 6:691(=0.00868) in the test set. Remark that Abalone19 is 100 times more imbalanced than Soybean12 and 1000 times more imbalanced than Yeast.

3.3.4 Statlog Dataset

Statlog (Landsat Satellite) is a UCI Dataset with 7 classes and 36 attributes. The original datasets consists of 6435 samples and since there is no missing value in the data, all samples are used. In this experiment we choose class 4 as the positive class and all other classes as negative class in order to create the most imbalanced setting from the dataset. Data is given as training

set, 4435 samples and test set 2000 samples. Numerical properties of the dataset Statlog4 is summarized below:

	Positive Examples	Negative Examples	Total
Statlog4	626	5809	6435
Statlog4 Training	415	4020	4435
Statlog4 Test	211	1789	2000

Table 3.4: Properties of Statlog Dataset

Statlog4 has an imbalance of 0.10323 in the training set and 0.11794 in the test set.

3.3.5 Letter Recognition Dataset

Letter Recognition (Landsat Satellite) is a UCI Dataset with 26 classes and 16 attributes. The original datasets consists of 20000 samples and since there is no missing value in the data, all samples are used. In this experiment we choose class H as the positive class and all other classes as negative class in order to create the most imbalanced setting from the dataset. Data is divided as training set, 14000 samples and test set, 6000 samples. Numerical properties of the dataset Letter RecognitionH is summarized below:

	Positive Examples	Negative Examples	Total
LetterRecogH	734	19266	20000
LetterRecogH Training	514	13486	14000
LetterRecogH Test	220	5780	6000

Table 3.5: Properties of Letter Recognition Dataset

LetterRecogH has an imbalance of 0.03811 in the training set and 0.03806 in the test set.

3.4 Evaluation Metrics

When evaluating classifiers on imbalanced datasets, instead of accuracy which is the proportion of correctly classified examples over whole dataset, a pair of statistical measures called sensitivity

and specificity may be preferred. This preference is based on the fact that accuracy alone will mislead the evaluation since a classifier that classifies the whole positive class incorrectly may still have a high accuracy. Hence considering the accuracy of positive examples-sensitivity- and the accuracy of the negative examples-specificity- separately will benefit the evaluation of the classifier. The sensitivity measures the proportion of correctly classified positives over whole positive examples and the specificity measures the proportion of correctly classified negatives over whole negative examples. Given the confusion matrix,

		True Value	
		Positive	Negative
Computed Value	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

sensitivity and specificity of a classifier are determined with the formulas,

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

In practice, there is usually a trade-off between these measures. Hence there is a need for some other metric that combines them for a more understandable and comparable representation. In order to meet this need, a metric called g-means is introduced([37]). The idea that is introduced in [37] suggests to maximize the accuracy on each of the classes while keeping these accuracies balanced. G-means is the geometric mean of the sensitivity and specificity given with the below formula

$$G - means = \sqrt{Sensitivitiy * Specificity}$$

Here sensitivity is also called True Positive Rate or Hit Rate in the literature. Another metric defined is False Positive Rate or also called False Alarm Rate is

$$FPRate = \frac{FP}{FP + TN} = 1 - Specificity$$

TP Rate and FP Rate are used when plotting ROC (Receiver Operating Characteristics) graphs which indicates the trade-off between hit rates and false alarm rates, in other words benefits and costs, respectively. Normally a binary classifier yields one confusion matrix and hence one point on the ROC curve plot. To plot a complete curve, in addition to binary classification information, probabilities that quantify the likelihood of belonging to a class are needed. Establishing a cut-off value for the probability of belonging to the positive class, a new confusion matrix can be derived for each such threshold value. The ROC curve would then depict the results for all possible threshold values [24]. In addition to being a generally functional method for performance evaluation of a classifier it is claimed in [37] that with some characteristics they are especially functional for imbalanced domains. An example ROC curve is demonstrated in Figure 3.5, In our study, we interpret the probability that quantify the likelihood of belonging to a class, as the distance value of each instance to the constructed hyperplane. The algorithm proposed by Fawcett in [24] is used to calculate a ROC Curve. However, in order to compare performance of different classifiers a reduced metric would be more effective than a two-dimensional representation. Likewise g-means, a single valued metric is commonly used for this purpose [9] which is the area under the ROC curve, abbreviated AUC. Since it is the portion of the area of a unit square it is between 0 and 1.

Remark that the functionality of g-means and AUC are discussed above, but considering the fact that the attention of classification is on the minority class, in case of imbalanced dataset sensitivity is also a functional metric. Since g-means and AUC focus on a trade-off between representational metrics of majority class and minority class, a user may choose sole sensitivity

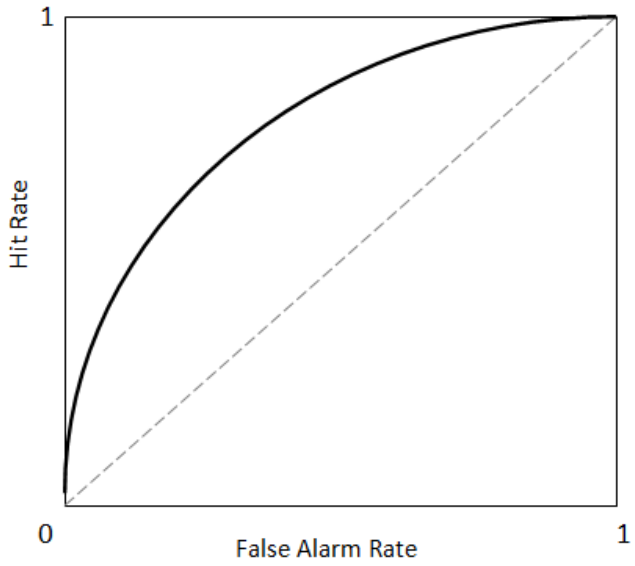

Figure 3.5: Sample ROC Curve

metric for analysis since it benefits and focuses on the classification of minority class correctly. In our approach, we solve many subproblems for different set of parameters. Hence we have different values of these metrics for each subproblem and the choice of the metric is left for the user. We do not have the opportunity to analyze all metrics for each dataset in the experiments we will be conducting. However, we would list a detailed analysis of these metrics for three datasets with the method *grid1* which may lead the choice of the user.

For Soybean12 dataset the best five percent of g-means values in terms of sensitivity and specificity can be examined in Figure 3.6. As it is seen from Figure 3.6, choice of best 5 percentage of g-means covers the best values of sensitivity and specificity. On the other hand, when we plot the best five percent of of AUC value in terms of sensitivity and specificity, it mainly covers the whole sensitivity and specificity values as seen in Figure 3.7. If we represent the overall best g-means value and best AUC value in Figure 3.8 we clearly see that best value of g-means and AUC compromise for the same point. To sum up, for Soybean12 dataset it is seen that AUC

covers g-means values whereas g-means represents the best five percent of better. Therefore, for Soybean dataset, choosing g-means as the evaluation metric is meaningful.

For Yeast dataset, the best five percent of g-means values in terms of sensitivity and specificity is represented in Figure(3.9).The best five percent of AUC values in terms of sensitivity and specificity can be seen in Figure(3.10) and finally the best values of g-means and AUC are represented in Figure(3.11) in terms of sensitivity and specificity. As in the Soybean case best five percent of g-means values covers the best values of sensitivity and specificity whereas the best five percent of AUC values covers the whole sensitivity and specificity values. When we plot the overall best g-means value and the overall best AUC value in Figure(3.11) we see that values do not compromise however the difference is omittable. Therefore, for Yeast dataset, it is seen that choosing either g-means or AUC as the evaluation metric is possible.

As mentioned before, we see that none of the metrics is superior or eligible among the others hence we will report the three metrics g-means, AUC and sensitivity and leave the choice for the decision maker.

Note that the origin in all graphs are $(0, 0)$, but for the sake of representation we limited the axes between 0.4 and 1.01.

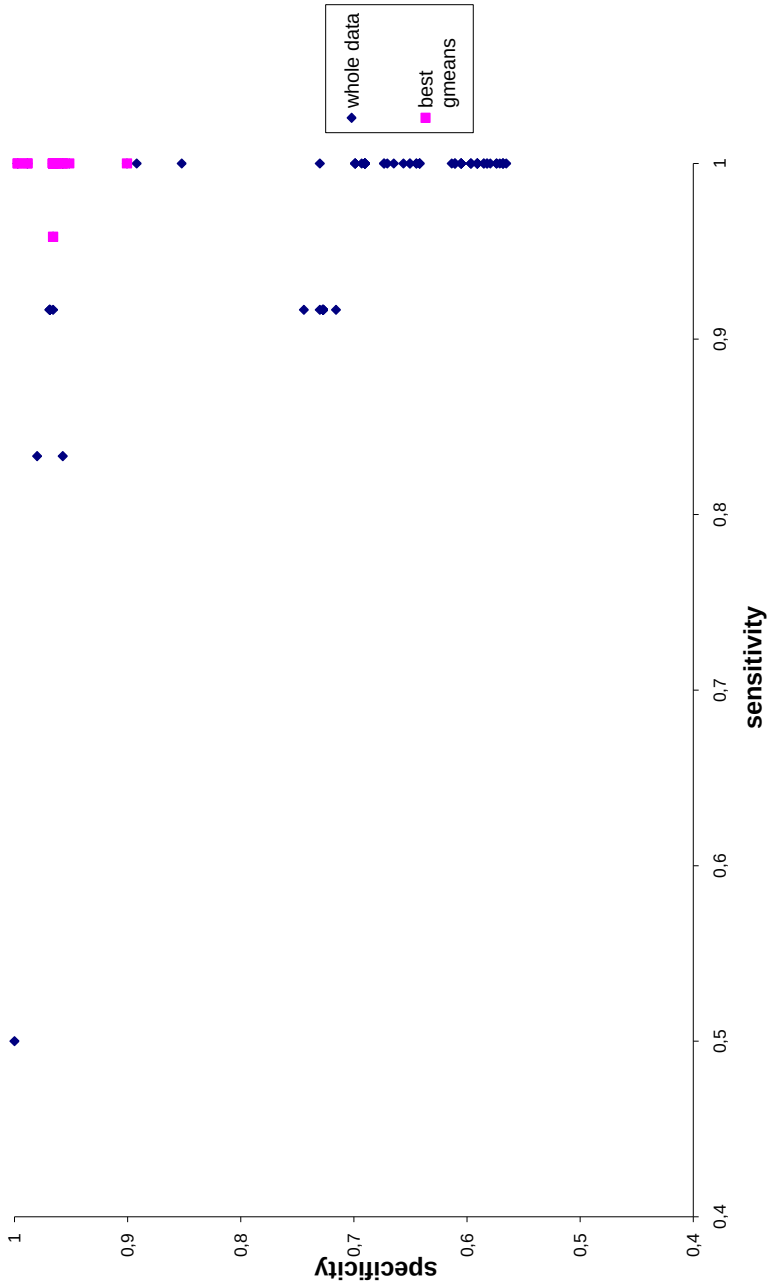

Figure 3.6: The best five percent of g-means values for Soybean

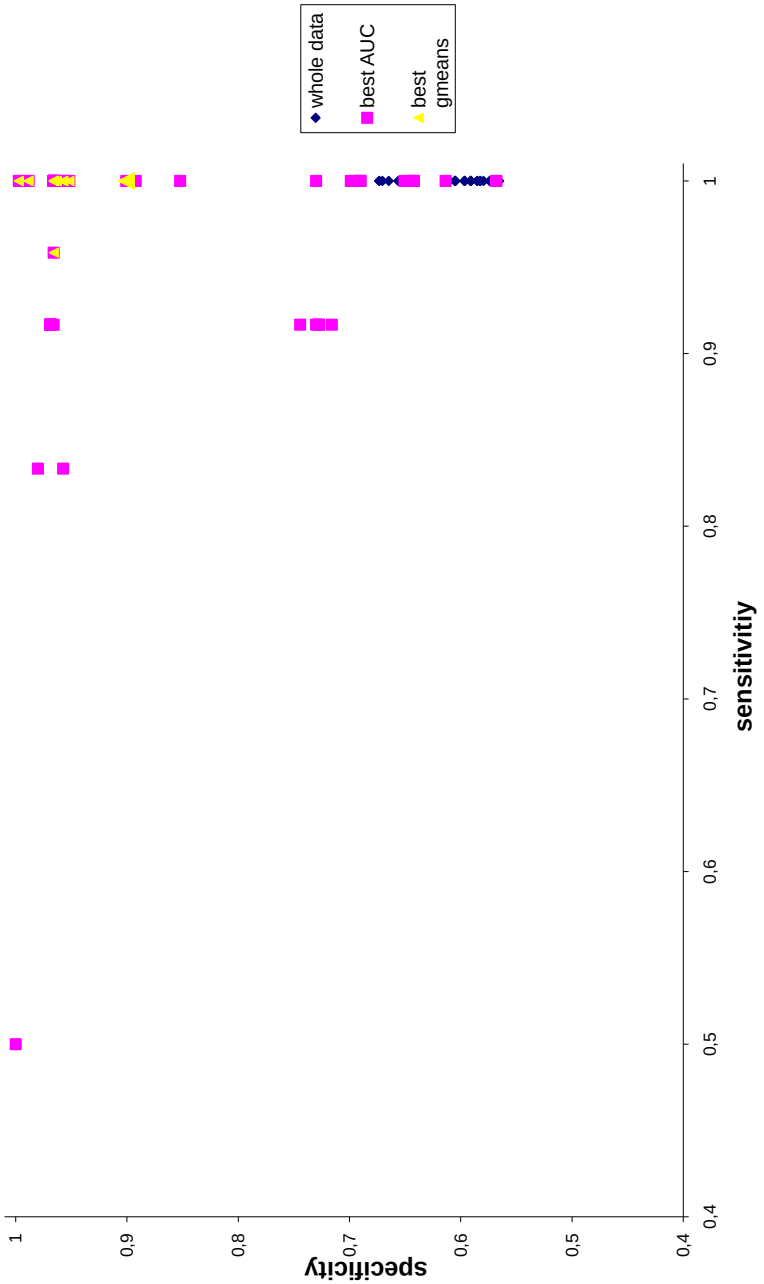

Figure 3.7: The best five percent of AUC values for Soybean

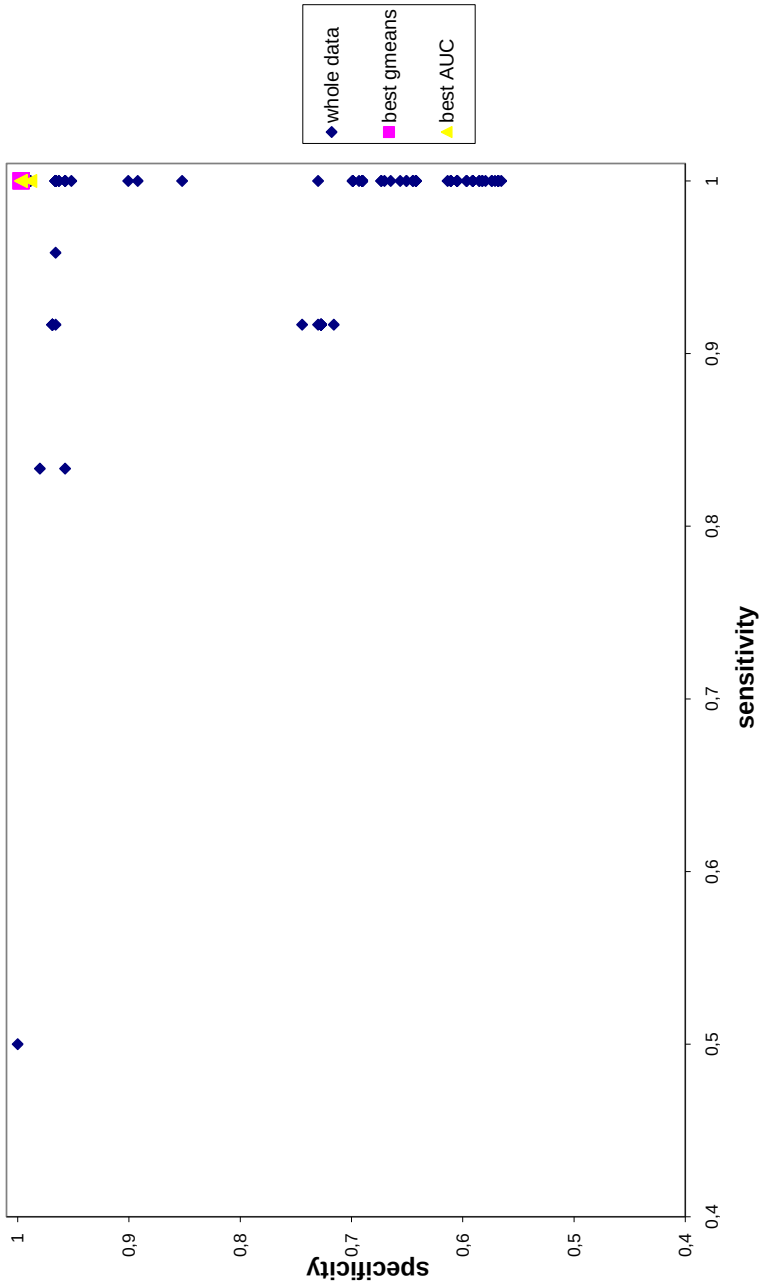

Figure 3.8: The best g-means and the best AUC values for Soybean

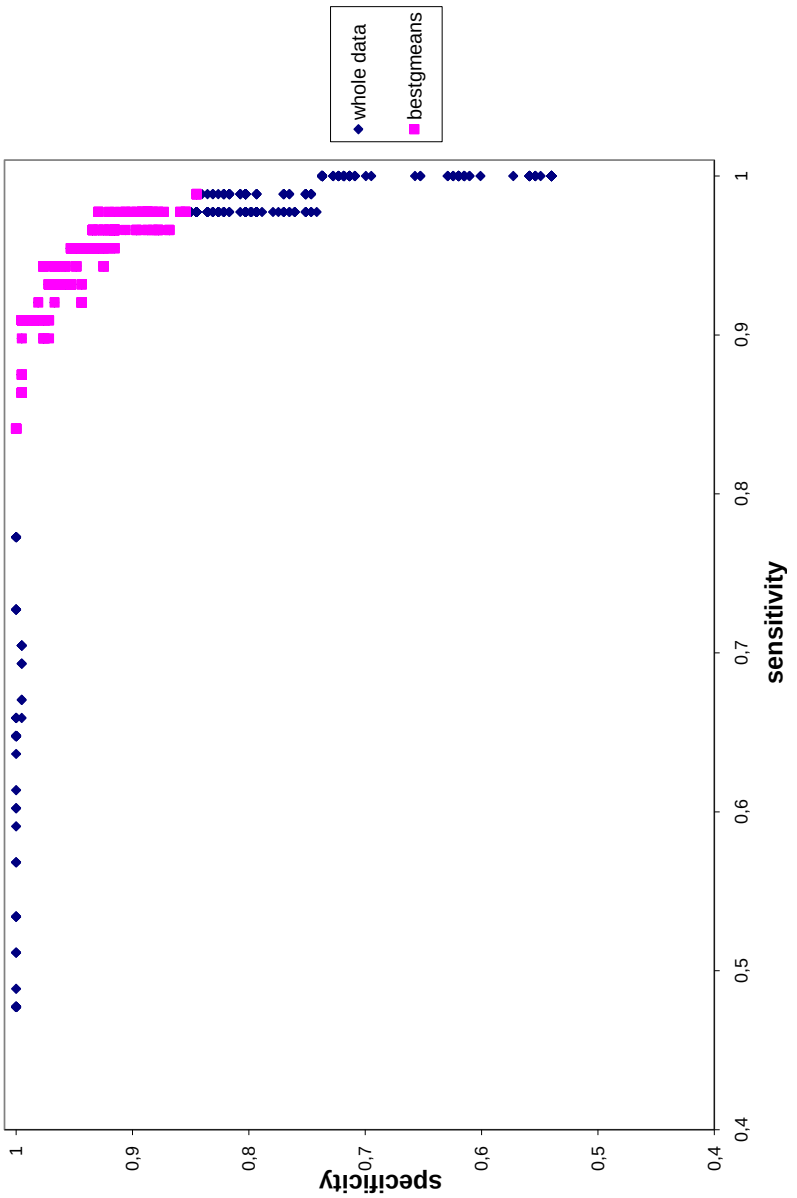

Figure 3.9: The best five percent of g-means values for Yeast

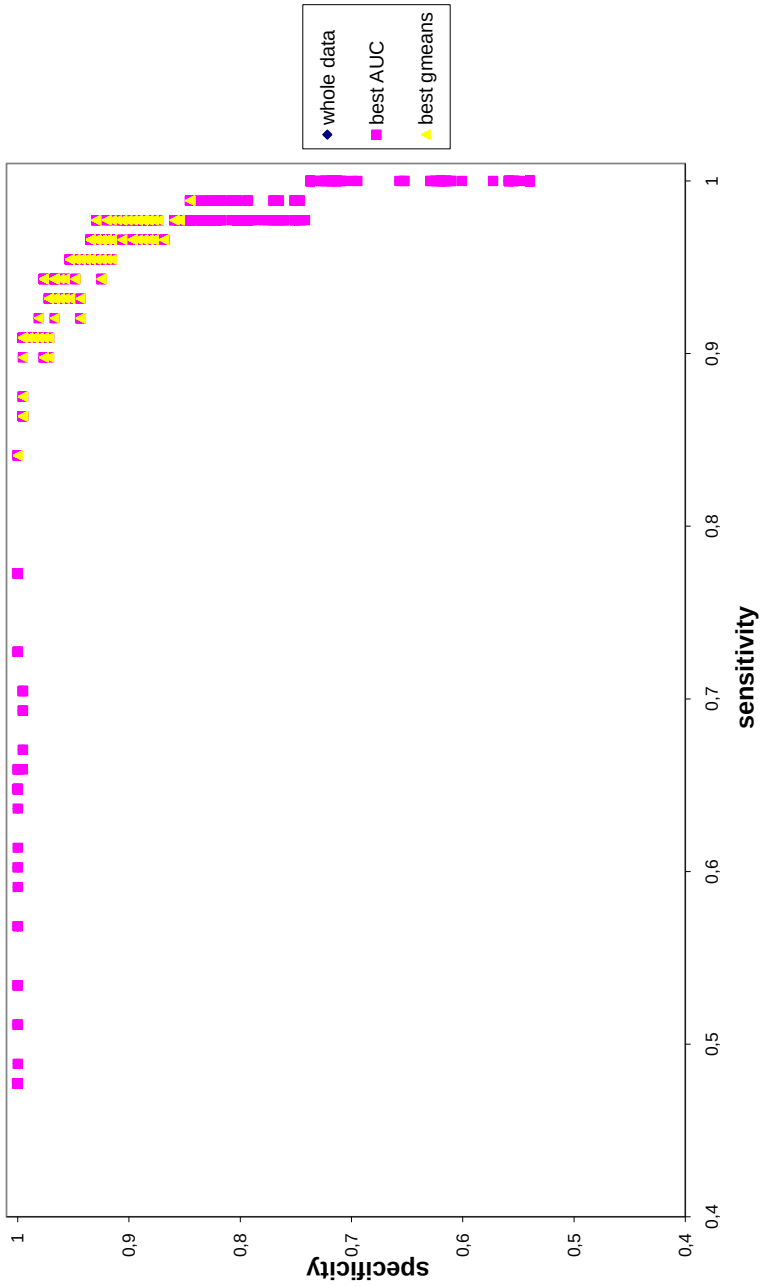

Figure 3.10: The best five percent of AUC values for Yeast

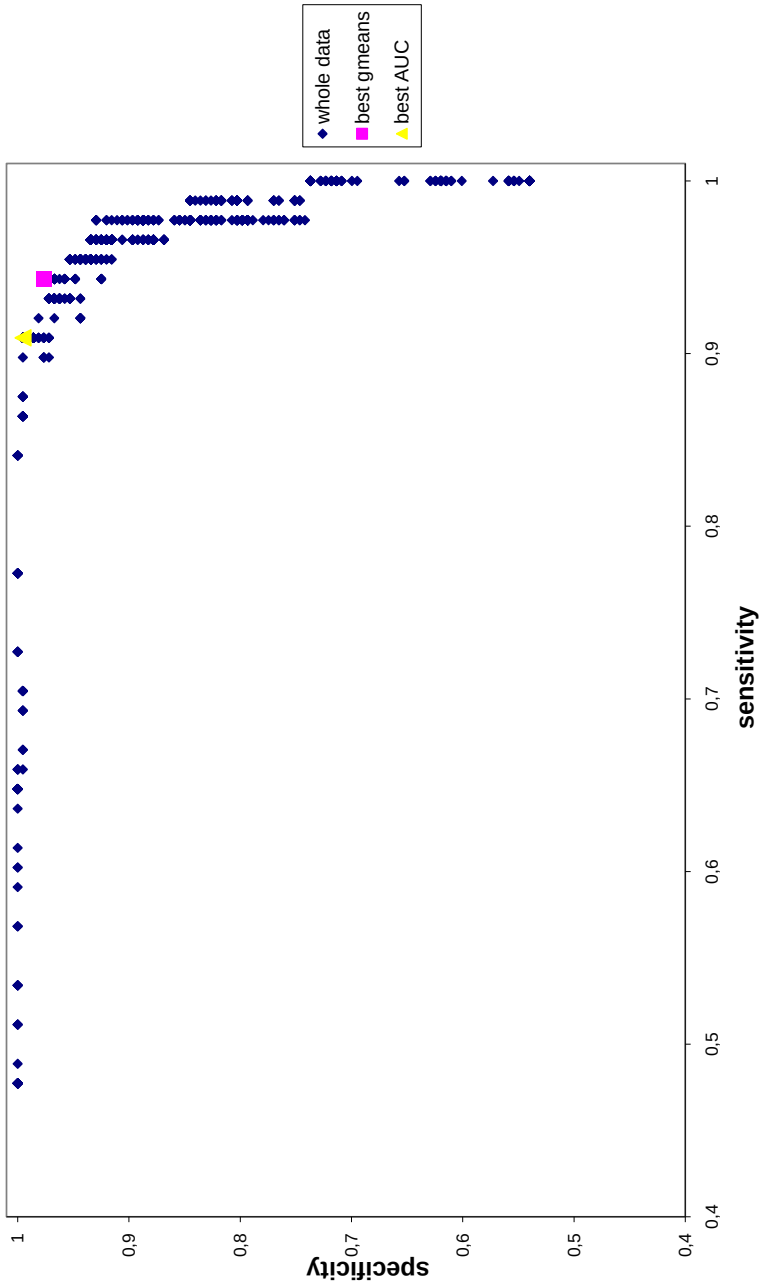

Figure 3.11: The best g-means and the best AUC values for Yeast

Chapter 4

EXPERIMENTS

4.1 Experiment Settings

The experiments are conducted on the five datasets introduced before with the comprehensive methods *grid1*, *grid2*, *gridS*, *2C* and four variants of cost-sensitive SVM classification. The variants differ in the choice of the trade-off parameter C in the cost-sensitive SVM formulation and referred as *C=1*, *C=10*, *C=100*, *C=1/ImbRatio* for C values 1,10,100, the ratio of number of positive instances over number of negative instances, respectively. The tolerance parameter is chosen as 0.01, a value obtained empirically. This value is large enough to keep the search reasonable and small enough to keep the search comprehensive.

We use ILOG Cplex [30] Linear Programming Solver through ILOG Concert Technology [29] which is an interface representing object-oriented models. ILOG Concert Technology is used with C++ language implementations in Microsoft Visual Studio 2005.

The experiments are conducted on two different machines; some are conducted on Intel Core2 CPU, 1.83Ghz, 1GB RAM, Microsoft XP Professional whereas some are conducted on Intel Pentium Dual Core 3.20 GHz, 2 GB RAM, Windows NT Server 2003 OS. Hence comparing CPU times of the experiments is not reasonable in this case. However we will be listing the number of iterations and the number of subproblems solved for each experiment, since these values are independent from the machine we will be able to compare the methods.

4.2 Results

The results of 8 methods are presented with metrics discussed in Section 3.4 in columns *g-means*, *AUC* and *sensitivity*. Also the number of total LP iterations and the number of subproblems solved are given in columns *LP iteration* and *breakpoint*. Remark that we report LP iterations since it is a good indicator of CPU time when the size of the formulation is known.

Soybean12	gmeans	AUC	sensitivity	LP iteration	breakpoint
grid1	0.998579	0.99858	1	4317	104
grid2	0.998579	0.99858	1	6825	387
grids	0.998579	0.99858	1	1033	100
2C	0.998579	0.99858	1	61	15
C=1	0.994302	0.99728	1	17	1
C=10	0.994302	0.99728	1	17	1
C=100	0.994302	0.99728	1	17	1
C=1/ImbRatio	0.994302	0.99728	1	17	1

Table 4.1: Evaluation for soybean dataset

The experiments conducted on Soybean12 dataset revealed that the comprehensive methods, named *grid1*, *grid2*, *grids*, *2C* are clearly better than the cost-sensitive SVM methods, named *C=1*, *C=10*, *C=100*, *C=1/ImbRatio* (Number of negative instances over number of positive instances in the dataset), in terms of g-means and AUC. Sensitivity is equal for all methods. It is also seen that *2C* has the best result since for equal *g-means*, *AUC* and *sensitivity* values it has the least number of iterations, meaning it is the fastest method. Additionally, for Soybean12 dataset maximum g-means determined by SDC method and SMOTE algorithm in [1] are equal and 1. Their corresponding sensitivity values are also 1.

The experiments conducted on Yeast dataset revealed that the cost-sensitive comprehensive methods, named *grid1*, *grid2*, *grids* are clearly better than the cost-sensitive SVM methods and *2C* method in terms of *g-means* and *sensitivity*. *AUC* metric does not have a pattern compatible with other two metrics, the best AUC value is determined for *grid2* method and the second

yeast	gmeans	AUC	sensitivity	LP iteration	breakpoint
grid1	0.959709	0.99344	1	4912	865
grid2	0.958485	0.9944	1	6840	1174
grids	0.955084	0.99312	1	2725	100
2C	0.933102	0.99402	0.909091	225	94
C=1	0.917011	0.99386	0.840909	92	1
C=10	0.941979	0.98271	0.954545	95	1
C=100	0.938574	0.9905	0.977273	121	1
C=1/ImbRatio	0.950436	0.9921	0.943182	103	1

Table 4.2: Evaluation for yeast dataset

best is determined for *2C* method. Among cost-sensitive SVM methods *C=1/ImbRatio* has the highest *g-means* metric value. We see that *grid1* has the best g-means value, however has a high number of iterations. Decision maker may choose *grids* method for a small loss in *g-means* value if time is considered to be important. For Yeast dataset, maximum g-means determined by SV-Pruning method in [14] is 0.9304 with sensitivity of 0.87, the benchmark they used, cost modifying with $(C_1/C_2) = (3/1)$, result in g-means value of 0.9408 with sensitivity 0.9195.

Abalone19	gmeans	AUC	sensitivity	LP iteration	breakpoint
grid1	0.913795	0.93994	1	43193	5517
grid2	0.918534	0.94501	1	56499	25390
grids	0.890536	0.93367	1	34890	54
2C	0	0	0	75	1
C=1	0	0	0	57	1
C=10	0	0	0	310	1
C=100	0.909032	0.88857	1	2070	1
C=1/ImbRatio	0.881552	0.88326	1	2207	1

Table 4.3: Evaluation for abalone dataset

The experiments conducted on Abalone19 dataset revealed that the methods *grid1* and *grid2* perform better than all other methods in terms of *g-means*, *AUC* and *sensitivity*. We see that *grid2* has the best g-means value and AUC value among all other metrics, however sensitivity

value is same with *grid1*, *grids*, *C=100* and *C=1/ImbRatio*. Since the number of iterations value of *grid2* does not differ dramatically from *grid1*'s a decision maker may choose *grid2* method in such a case. Additionally, for Abalone19 dataset, maximum g-means determined by SDC method and SMOTE algotihm in [1] are 0.7449 and 0.0, respectively. Their corresponding sensitivity values are 0.808 and 1, respectively.

Also, recall that the comprehensive method *grids* searches each 10 grid value of majority epsilon for 10 grid value of minority epsilon, in other words 10x10=100 combination is searched. Hence, in a *grids* search, the maximum number of breakpoints is 100, however if any of the stopping criteria is met, infeasibility or zero epsilon, algorithm ends before searching 100 breakpoints. Here, it is important to point out that the number of LP iterations of *grids* is close to the number of LP iterations of *grid1* for many of the datasets we have used. This depends on the fact that, *grids* resolves the problem for each majority epsilon-minority epsilon combination (10x10=100 times), however *grid1* rebuilts the problem and continues with sensitivity analysis for each combination. Although we were not able to locate any benchmark results reported in the literature for the following two datasets, we include them for the purpose of enriching our computational experiments.

Statlog4	gmeans	AUC	sensitivity	LP iteration	breakpoint
grid1	0.766798	0.78048	1	62135	10864
grid2	0.763787	0.78055	1	101567	22144
grids	0.761393	0.77904	1	50706	37
2C	0	0	0	730	1
C=1	0	0	0	1253	1
C=10	0.707106	0.82009	0.952607	3785	1
C=100	0.631306	0.81025	1	3675	1
C=1/ImbRatio	0.705823	0.7617	0.943128	3771	1

Table 4.4: Evaluation for statlog dataset

The experiments conducted on Statlog4 dataset revealed that *grid1*, *grid2* and *grids* perform better than *2C*, *C=1*, *C=10*, *C=100* and *C=1/ImbRatio* in terms of g-means and sensitivity. *C=10* method has the highest AUC value for Statlog4 dataset. It is seen that the highest

g-means value is achived by *grid1* method.

LetterRecogH	gmeans	AUC	sensitivity	LP iteration	breakpoint
grid1	0.791614	0.83361	1	49819	6248
grids	0.781872	0.83888	1	177120	43
2C	0	0	0	1027	1
C=1	0	0	0	1361	1
C=10	0.637274	0.82009	0.431818	7979	1
C=100	0.683426	0.81025	0.995455	12488	1
C=1/ImbRatio	0.782344	0.84201	0.831818	10626	1

Table 4.5: Evaluation for letter recognition dataset

The experiments conducted on LetterRecogH dataset revealed that *grid1*, *grids* and *C=1/ImbRatio* methods perform better than other methods in terms of *g-means*. The best *g-means* value is attained for *grid1* method, whereas the best AUC value is attained for *C=1/ImbRatio* method. Due to computational limitations and the remarkably large size of the dataset LetterRecogH *grid2* is not ran for this dataset.

To sum up, grid1 method has the best performance in two cases, also grid2 method has the best performance in two cases and in one case all comprehensive search methods result in the same performance in terms of g-means. Generally, the loss in performance is small enough to be tolerated where *grid2* beats *grid1* and since *grid2* requires more computational effort/time choosing *grid1* is reasonable. Also since conducting a more comprehensive and detailed search on the minority epsilon, since it belongs to the class we are interested, choosing *grid1* instead of *grid2* is reasonable. The performance value results in terms of *sensitivity* are consistent with the results in terms of *g-means* for all methods. However AUC achives the best performance with all comprehensive methods in one case, with grid2 method in two cases and with C adjusting cost-sensitive SVM methods in two cases. In the Statlog4 dataset case AUC value achived by *C=10* method is remarkably higher than all others that we can not generalize a best method considering AUC metric.

Considering the C adjusting cost-sensitive methods, even though they can not reach the performance values that comprehensive methods reach, they perform good according to the dataset and the distribution of the dataset. Therefore, for a better analysis a large set of parameters should be tried. However, since the parameters are adjusted by trial and error, when a large set of parameters are tried the expense of this will approximate to the expense of any comprehensive methods. We look at g-means and sensitivity as a combination most of the time while assessing the proposed methods. Due to the results of the experiments we see that AUC metric does not differentiate between the methods. It is a strong measure but as in LetterRecogH case, very different sensitivity levels may have similar AUC values. This is a drawback for imbalanced datasets and in this case we may suggest using g-means and sensitivity for evaluating the methods.

Overall, comprehensive methods, except *2C*, perform better than methods such as SMOTE and SDC which are the most well-known benchmarks in the literature. The difference is difficult to generalize and changes from dataset to dataset. However, one needs to note that comprehensive methods come at significant computational cost. For instance the run for *grid2* on dataset Statlog4 with 101567 LP iterations and 22144 breakpoints takes 40695.4 CPU seconds. But the general CPU behavior is an average and tolerable CPU time when datasets are not too large, for example, the run for *grid1* on Abalone19 dataset with 4177 instances takes 4509.86 seconds. Finally, it is important to highlight and demonstrate the multi objective nature of the problem (P-3C) that forms the method *grid1*. As mentioned the problem consists of three objective functions, maximizing the norm, minimizing the empirical error on majority class and minimizing the error on minority class. Let us demonstrate the trade-off between these objectives with the results of the experiment conducted on Yeast dataset with the method *grid1*:

In Figure 4.1 the multi objective nature of the problem can be clearly seen. Figure 4.1 displays the results of an experiment in the three dimensional objective space. The three objectives of the formulation (CS-SVM), norm, total positive error and total negative error appear as the coordinates. Each point in the figure symbolizes a classifier that possesses the corresponding norm, positive error and negative error. Since the multiobjective formulation seeks to minimize

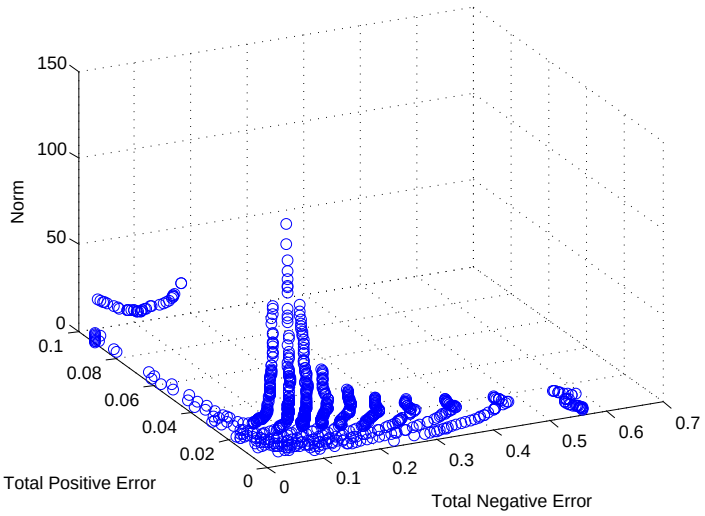

Figure 4.1: Trade-off between the three objective

all of these quantities, points closer to the origin are the most desirable ones. In Figure 4.1, it is also possible to observe that we are only sampling in the three dimensional objective space and we are not able to generate the entire efficient frontier in the three dimensional space.

In Figures 4.2 and 4.3, the projections of the three dimensional graph onto two dimensional ones can be observed. For instance in Figure 4.2, one can observe how total positive error materializes as a function of norm. Note that not all points on this two-dimensional graph are efficient in a two objective sense. That is because the points are computed by taking three objectives into consideration, and for all points in Figure 4.2, there is an associated total negative error value that is not being displayed. In general, these figures help demonstrate the fact that several alternative SVM classifiers can be obtained by varying the tolerated error values (ϵ_1 and ϵ_2) in the proposed formulation. We note that some of these classifiers display a more desirable performance in terms of the trade-off among the norm, the positive error and the negative error.

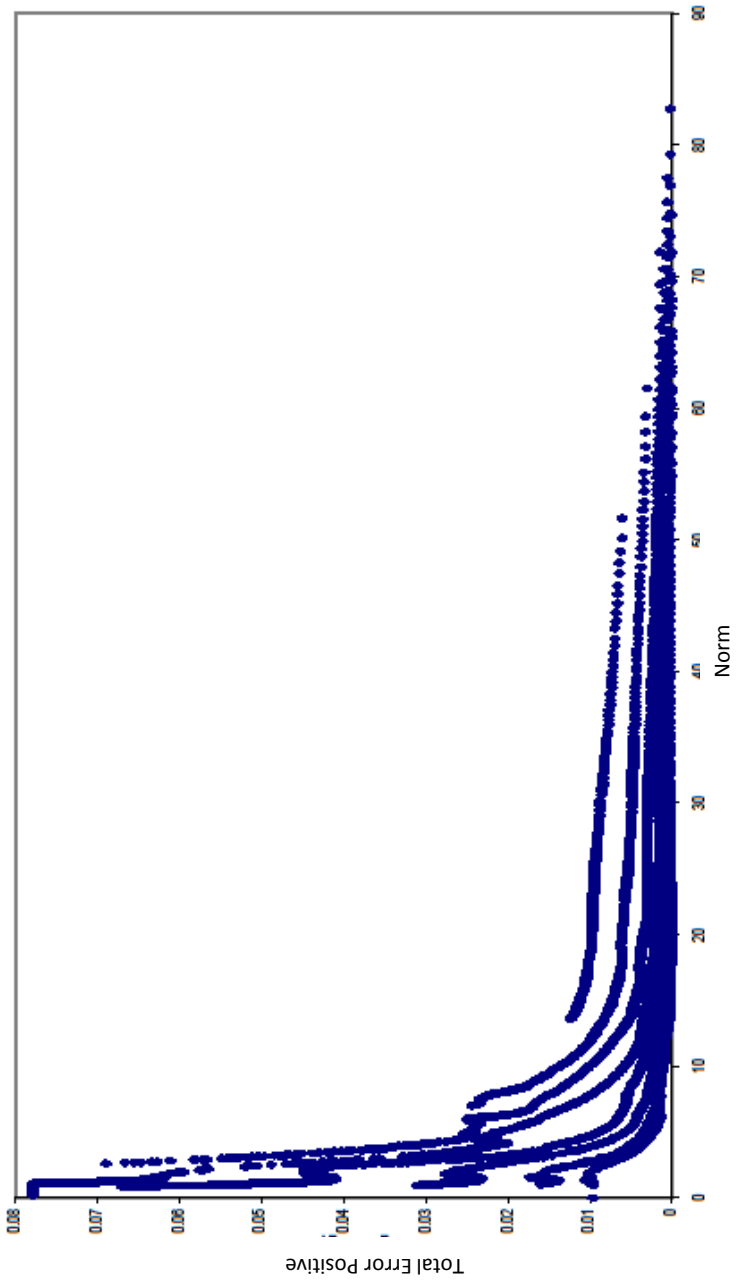

Figure 4.2: How the norm changes due to the error on minority class

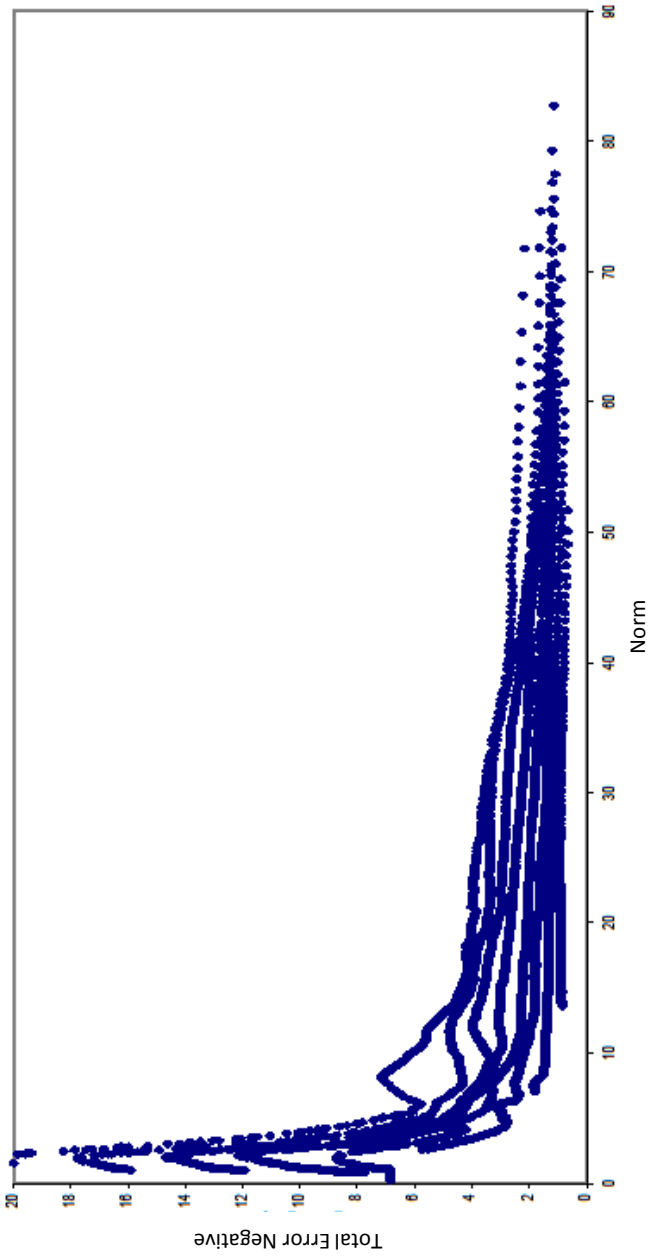

Figure 4.3: How the norm changes due to the error on majority class

Chapter 5

REDUCING COMPUTATIONAL TIME

With the proposed method *grid1* a big portion of the majority epsilon versus minority epsilon space is scanned as seen in the demonstration in Figure 3.4 It is seen from the figure, with this method, for a grid leg value of majority epsilon nearly all of the minority values are scanned. Remark that at each iteration the methods solves an Linear Programming problem for a specific majority epsilon value and minority epsilon value. For a small problem (in terms of number of instances and attributes) this comprehensive search is eligible since the computational effort is tolerable and it is likely to reach the best possible values for different metrics. However, for a very large problem, the computational effort may be to much to handle in a reasonable time therefore a heuristic should be proposed. These are also true in case of *grid2* method. We analyse the results for each 5 datasets mentioned above to come up with a solution for this problem. The search of the majority epsilon values do not reveal any common pattern among 5 datasets. The maximum values for g-means metric attained at the various grids of majority epsilon for each dataset hence we can not eliminate any portion of this axis. When the pattern search is conducted on the minority epsilon values, for Soybean12 dataset we see that reducing the starting value of minority epsilon from 20 to 8 does not hurt the best g-means value. The starting value of minority epsilon can be reduced from 75 to 58 for Yeast dataset , from 26 to 14 for Abalone19 dataset, from 415 to 255 for Statlog4 dataset, from 314 to 123 for LetterRecogH dataset without hurting the best g-means values for each of them. As it is seen from this analysis there is not a common pattern revealing a behavior for reducing the portion of minority epsilon scanned. Nevertheless we can conclude that reducing the starting value of the minority epsilon around some limits, that are determined with the properties of dataset, does not hurt the best g-means values. Therefore we introduce a parameter α that lies

in the interval $[0, 1]$ and used to reduce the starting value of the minority epsilon. α value is determined according to the importance given to the minority class by the decision maker. Also remark that in SVM formulation we can not control the number of misclassified instances, which is a NP-Hard problem, but we can control the sum of distances to the hyperplane. Greater α means we let more instances from the minority class to be misclassified whereas a smaller α means we let less instances from the minority class to be misclassified. In other words α value gets smaller as the decision maker pays more attention to the minority class. We conducted experiments on the 5 datasets mentioned, with $\alpha = 0.5$ and comparison in terms of g-means and number of iteration is listed below. In the tables *HalfMingrid1* row stands for the method proposed above in order to reduce computational time and *grid1* row stands for the results of the original method that are also listed in Section 4.2.

Soybean12	gmeans	AUC	sensitivity	LP iteration	breakpoint
HalfMingrid1	0.998579	0.99858	1	3310	97
grid1	0.998579	0.99858	1	4317	104

Table 5.1: Reduced Evaluation for Soybean Dataset

For Soybean12 dataset *HalfMingrid1* performed same as *grid1* in terms of all metrics that are reported. Also since the number of iterations decreased by 23 percent, choosing *HalfMingrid1* method is reasonable in such a case.

Yeast	gmeans	AUC	sensitivity	LP iteration	breakpoint
HalfMingrid1	0.959709	0.99344	1	4113	780
grid1	0.959709	0.99344	1	4912	865

Table 5.2: Reduced Evaluation for Yeast Dataset

For Yeast dataset, like in the Soybean dataset case, two methods performed same in terms of all metrics that are reported. As in the previous case choosing *HalfMingrid1* method is reasonable since the number of iterations decreased by 16 percent.

Abalone19	gmeans	AUC	sensitivity	LP iteration	breakpoint
HalfMingrid1	0.89297	0.90135	1	21955	2018
grid1	0.913795	0.93994	1	43193	5517

Table 5.3: Reduced Evaluation for Abalone Dataset

For Abalone19 dataset, *HalfMingrid1* method performed worse than *grid1* method in terms of *g-means* and AUC, as expected. However *sensitivity* values are same for both methods. The number of iterations decreased by 49 percent but the loss in *g-means* and AUC values are high, around 0.02. If the computation time is too important and that much loss is tolerable by the decision maker *HalfMingrid1* may be preferred.

Statlog4	gmeans	AUC	sensitivity	LP iteration	breakpoint
HalfMingrid1	0.745707	0.77507	0.985782	25051	4845
grid1	0.766798	0.78048	1	62135	10864

Table 5.4: Reduced Evaluation for Statlog Dataset

For Statlog4 dataset, *HalfMingrid1* again performed worse than *grid1* method in terms of *g-means*, AUC and *sensitivity*. The number of iterations needed decreased by 59 percent and the loss in *g-means* and AUC values are around 0.02. As in the previous case if the loss is tolerable and computational time is a constraint, decision maker may prefer *HalfMingrid1*.

LetterRecogH	gmeans	AUC	sensitivity	LP iteration	breakpoint
HalfMingrid1	0.791614	0.99858	0.968182	91350	12318
grid1	0.791614	0.83361	1	49819	6248

Table 5.5: Reduced Evaluation for Letter Recognition Dataset

For LetterRecogH dataset, the results did not meet the expectations. *HalfMingrid1* performed worse than *grid1* in terms of *gmeans* and *sensitivity*. Even though *HalfMingrid1* performed slightly better in terms of AUC remarkable increase in the number of iterations for *HalfMingrid1* makes this heuristic not prefable.

To conclude, we can say if the dataset is slightly imbalanced (Yeast) or linearly separable (Soybean12) *HalfMingrid1* would cover the best values for three metrics. However, when the dataset is extremely imbalanced or not separable halving the starting value of minority epsilon results in some loss in the three metrics. In this case, the decision considering the trade-off between the loss in the best performance for metrics and the computation time is left to decision maker. For very big problems in terms of both number of instances and number of attributes it is possible to choose *HalfMingrid1* in order to complete the computations in a reasonable time.

Chapter 6

CONCLUSION

In this study, we mainly aim to improve the classification performance of SVMs on imbalanced datasets. The Imbalanced Dataset problem is a remarkable challenge studied in knowledge discovery and data mining framework. This type of datasets can be found in many fields such as banking data (fraud detection), health (diagnosis of a disease), scientific research results, etc. Therefore there are many studies concerning different aspects of this problem. We mainly divide these studies into two parts, Data-Level Solution Approaches and Algorithm-Level Solution Approaches. The studies belonging to the first part focuses on rebalancing the number of instances in both classes, either eliminating some instances of the majority class or replicating some instances of the minority class until the balance condition is met. Also, some rules may be applied to the process of eliminating or replicating as we mentioned. On the other hand, the studies belonging to the second part, focuses on two methods. They either adjust misclassification costs or bias the algorithm in order to force the classifier for a desired separation. Our approach can be included in the Algorithm-Level Approach since the main model is constructed on using distinct misclassification allowances for majority and minority classes.

As mentioned, our approach is based on the model in which distinct misclassification costs for majority and minority classes introduced via constraints on misclassification errors. These allowance values are generally determined empirically, by trial and error in other words, in the studies we encountered. However due to computational efforts and absence of a systematic procedure that can scan all/most of the values that are suitable, some general values (C=1, C=10) or the imbalance ratio (which is the ratio of number of minority class instances over number of majority class instances) is used. Therefore the most valuable set of these misclassification cost parameters that leads to desired classification performance may be lost due to this limited

scanning abilities.

Our contribution pertains this absence of a systematic procedure that can scan all/most of the values that are suitable for misclassification allowances. In the model we mentioned, there exists distinct misclassification allowances for both classes. Therefore, there are three objectives in our model one of them controls the generalization error -maximizing the margin- and the two others control the empirical error -minimizing cost of misclassification of majority class and minority class. It is clear from this explanation that SVMs can be considered as Multi Objective Optimization problems. With this fundamental idea, we apply ϵ-constraint method, one of the solution approaches of Multi Objective Optimization problems to the SVM formulation. Since defining a piecewise surface that results from the ϵ-constraint parametrization is inefficient in this case, we propose two alternative reductions that transforms both objectives regarding the empirical error to ϵ-constraints. After reconstructing the formulation in this way, we use an iterative algorithm that allows us to scan different values of these ϵ-constraints. In this algorithm we based the iterative procedure on sensitivity analysis since the formulation using L_1-Norm results in a Linear Programming Formulation. We try various reductions into simpler subproblems. We finally propose a heuristic in order to reduce computational effort in case of large problem sizes. Our results on five datasets implied that using the comprehensive search methods, the methods proposed in this study *grid1*, *grid2*, *grid*, are considered to have better performances in terms of three metrics *gmeans*, *AUC*, *sensitivity*. However in some cases the performance values of cost-sensitive methods in terms of three metrics are slightly different from the comprehensive search methods but have remarkably less computational effort again compared to the proposed methods. In other words, adjusting parameters for a cost-sensitive SVM model by trial and error may result in good performance values but it can not attain the best values that the comprehensive methods generally find. Also considering trying a set of parameters C in order to reach reasonable performance values, the expense of trial and error approximates the expense of the comprehensive searches as the dataset gets more imbalanced.

Although the heuristic method we proposed did not perform consistently well, its overall performance is acceptable. Therefore, if there exists some limitations on the computational efforts,

the decision maker may choose this heuristic, with considering the trade-off between computational cost and better performance values. Another important result is that the comprehensive search method with three criteria formulation *grid1* performs better than the one with two criteria formulation *2C* in most of the experiments. This proves the statement that in addition to the systematic search procedure proposed, assigning distinct misclassification allowances for majority and minority classes improves the performance of SVMs significantly. Restating the findings, we recommend using *grid1* for better results in an imbalanced dataset. If the imbalanced dataset is large in terms of number of instances or number of attributes, it is reasonable to use the heuristic proposed for a more economic use of CPU time.

Due to the metrics analysis in Section 3.4 and the results summarized in Section 4.2, metrics do not consistently point out the best method and they may not discriminate among alternative methods. Even though it is not covered within the scope of this thesis, finding alternative metrics may still be an interesting topic as a future research area. Hence, we can conclude if the decision maker has no certain or crucial choice for metric, using any of them will not make a significant difference for reporting.

Even though we observed an improvement on the classification performance of SVMs on imbalanced datasets, our results are limited since we did not use Kernels in the formulation. Using the formulation with Kernels, proposed in [10], may improve the classification performance. Also the results will be analyzed better in that case since now the dependence on the separability of the dataset may also effect the performance of the classifier.

BIBLIOGRAPHY

[1] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. *In ECML*, pages 39–50, 2004.

[2] H. Aytug, L. He, and G. J. Koehler. Risk minimization and minimum description for linear discriminant functions by genetic algorithms. *INFORMS Journal on Computing*, 2008.

[3] H. Aytug and S. Sayin. Choosing the trade-off parameter for one-norm support vector machines. working paper.

[4] R. Barandela, J. S. Snchez, V. Garca, and E. Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849 – 851, 2003.

[5] G. Batista, A. Carvalho, and M. C. Monard. Applying one-sided selection to unbalanced datasets. pages 315–325. Springer-Verlag, 2000.

[6] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms*. Wiley-Interscience, 2006.

[7] H. Benson. Vector optimization with two objective functions. *Journal of Optimization Theory and Applications*, 28(2):253–257, June 1979.

[8] M.J.A. Berry and G.S Linoff. *Data mining techniques: for marketing, sales, and customer relationship management*. * Wiley Computer Publishing, 2004.

[9] 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.

[10] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Machine Learning Proceedings of the Fifteenth International Conference(ICML 98*, pages 82–90. Morgan Kaufmann, 1998.

[11] P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164–168. AAAI Press, 1998.

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, (16):321–357, 2002.

[13] N. V. Chawla and N. Japkowicz. Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explorations*, 6:2004, 2004.

[14] X. Chen, B. Gerlach, and D. Casasent. Pruning support vectors for imbalanced data classification. In *Proceedings of International Joint Conference on Neural Networks, Montreal, Canada*, 2005.

[15] W. W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[16] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*, pages 367–373. MIT Press, 2002.

[17] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, United Kingdom, 2000.

[18] B.V. Dasarathy. *Nearest neighbor (NN) norms: NN pattern classification techniques*. Los Alamitos, CA: IEEE Computer Society Press, 1991.

[19] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

[20] M.H. Dunham. *Data mining introductory and advanced topics*. Prentice Hall/Pearson Education, 2003.

[21] T. Eitrich and B. Lang. Parallel tuning of support vector machine learning parameters for large and unbalanced data sets. *CompLife 2005*, pages 253–264, 2005.

[22] C. Elkan. The foundations of cost-sensitive learning. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[23] K. J. Ezawa, M. Singh, and S. W. Norton. Learning goal oriented bayesian networks for telecommunications risk management. In *Proceedings of the 13th International Conference on Machine Learning*, pages 139–147. Morgan Kaufmann, 1996.

[24] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. *Machine Learning*, 31, 2004.

[25] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1:291–316, 1997.

[26] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. 1996.

[27] B. Fruhwirth and K. Mekelburg. On the efficient point set of tricriteria linear programs. *European Journal of Operations Research*, (72):192–199, 1994.

[28] Q. Gu, Z. Cai, L. Zhu, and B. Huang. Data mining on imbalanced data sets. pages 1020–1024, Dec. 2008.

[29] SA ILOG. CPLEX: ILOG CONCERT TECHNOLOGY 25 User's Manual and Reference Manual, 2007.

[30] SA ILOG. CPLEX: ILOG CPLEX 11.0 User's Manual and Reference Manual, 2007.

[31] N. Japkowicz. The class imbalance problem: Significance and strategies. In *In Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI*, pages 111–117, 2000.

[32] N. Japkowicz. Concept-learning in the presence of between-class and within-class imbalances. volume Volume 2056/2001, pages 67–77. Springer Berlin / Heidelberg, 2001.

[33] N. Japkowicz. Supervised versus unsupervised binary-learning by feedforward neural networks, 2004.

[34] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis, IOS Press*, 6:429–449, 2002.

[35] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Handling imbalanced datasets: A review, 2006.

[36] M. Kubat, R. C. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. In *Machine Learning*, pages 195–215, 1998.

[37] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. *Proceedings of the 14th International Conference on Machine Learning*, 1997.

[38] C. Ling and C. Li. Data mining for direct marketing problems and solutions. In *Proc. 4th International Conf. on Knowledge Discovery and Data Mining (KDD-98) New York*, pages 73–79. AAAI Press, 1998.

[39] O. L. Mangasarian. Misclassification minimization. *Journal of Global Optimization*, 5(4):309–323, 1994.

[40] O. L. Mangasarian. Mathematical programming in data mining. *Data mining and knowledge discovery*, 1(2):183–201, 1997.

[41] B. Maxon. Defining Data Mining. *DBMS Data Warehouse Supplement, Available at http://www.dbmsmag.com/9608d53.html*, August 1996.

[42] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42/3:203–231, 2001.

[43] R. Ramakrishnan and J. Gehrke. *Database management systems.* McGraw-Hill Science/Engineering/Math, 2003.

[44] R. J. Roiger and M. W. Geatz. *Data mining: a tutorial-based primer*. Addison Wesley Boston, 2003.

[45] Y. Tang, Y. Q. Zhang, N. V. Chawla, and S. Krasser. Svms modeling for highly imbalanced classification. *IEEE Transactions on systems, man and cybernetics*, 39(1):281–288, 2009.

[46] I. Tomek. Two modifications of cnn. *IEEE Transactions on Systems, Man and Cybernetics*, 6(11):769–772, 1976.

[47] Y. Z. Tsypkin. Foundations of the theory of learning systems. Translated by ZJ Nikolic. 1973.

[48] R. J. Vanderbei. Linear Programming: Foundations and Extensions, volume 37 of International Series in Operations Research and Management Science, 2001.

[49] V. N. Vapnik. *The Nature of Statistical Learning Theory, Second Edition*. Springer, New York, 2000.

[50] K. Veropoulos, C. Campbell, and N. Cristianini. Controlling the sensitivity of support vector machines. *Proceedings of the International Joint Conference on AI*, pages 55–60, 1999.

[51] S. Visa. Issues in mining imbalanced data sets - a review paper. In *in Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference, 2005*, pages 67–73, 2005.

[52] G. M. Weiss. Mining with rarity: A unifying framework. *SIGKDD Explorations*, 6(1):7–19, June.2004.

[53] I. Witten and E. Frank. *Data Mining:practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, San Mateo, CA, 2000.

[54] G. Wu and E. Y. Chang. Class-boundary alignment for imbalanced dataset learning. In *In ICML 2003 Workshop on Learning from Imbalanced Data Sets*, pages 49–56, 2003.

[55] G. W. Wynn and J. C. Crawford. Data Mining: A Concept of Customer Relationship Marketing.

[56] Y. Yoon. Discovering knowledge in corporate databases. *Information Systems Management*, 16(2):64–71, 1999.

[57] B. Zadrozy and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 204–213, New York, NY, USA, 2001. ACM.

VITA

AYŞEGÜL ÖZTÜRK was born in Bursa, Turkey, on May 25, 1984. She graduated from Balıkesir Science High School in 2002. She received her B.S. degree in Mathematics Engineering from Istanbul Technical University, Istanbul, in 2007. In September 2007, she joined the Industrial Engineering Department of Koç University as a teaching and research assistant.