FLEXIBLE CLASSIFICATION TECHNIQUES WITH BIOMEDICAL APPLICATIONS

Chong Zhang

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Statistics and Operations Research.

Chapel Hill
2014

ABSTRACT

Chong Zhang: FLEXIBLE CLASSIFICATION TECHNIQUES WITH BIOMEDICAL APPLICATIONS
(Under the direction of Yufeng Liu)

Classification problems are prevalent in many scientific disciplines, especially in biomedical research. Recently, margin based classifiers have become increasingly popular, partly due to their ability in handling large scale problems with fast computational speed and desirable theoretical properties. Despite the success of margin based classifiers, many challenges remain. For example, in practical problems, it can be desirable to estimate the class conditional probability accurately. For high dimensional classification data, penalized margin based classifiers are commonly used. However, when estimating the class conditional probability, the shrinkage effect from the penalty term in the corresponding optimization is often ignored. This effect can lead to large bias in estimation of the class conditional probability. Another important issue on classification is the comparison between soft and hard classifiers for multicategory problems. Moreover, regular multicategory margin based classifiers can suffer from inefficiency by using too many classification functions. In this dissertation, we propose several new classification techniques to overcome the challenges mentioned above. Comprehensive numerical and theoretical studies are presented to demonstrate the usefulness of our new proposed methodologies.

ACKNOWLEDGMENTS

I would like to thank my advisor Professor Yufeng Liu for his help and encouragement on my research. Working with him has brought so much fun and valuable experience. I have learned much through his supportive and encouraging guidance. It is a great pleasure to have him as my advisor.

I am also grateful to the Department of Statistics and Operations Research for providing a supportive and stimulating research environment. I would like to thank all the committee members, who have provided valuable suggestions and help in my dissertation. I thank all my friends at UNC for their friendship and help.

Lastly but not the least, I thank my fiancee Jie and my parents for their love and support.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Some Background on Classification

Classification is a typical supervised learning task and it is commonly seen in practice. Given a training dataset that has both the predictors and labels available, a classifier builds a model and predicts the label for new instances with only the predictors known. In this section, we briefly review some fundamental classification techniques and related concepts. In Section 1.1.1, some basic background of binary classification techniques in the statistical machine learning literature is introduced, and we focus on the large margin classifiers. We discuss the idea of extending large margin classification from binary to multicategory learning in Section 1.1.2. In Section 1.1.3, we consider the estimation problem of class conditional probabilities, which is another important aspect of classification besides the label prediction.

1.1.1 Binary Classification and Large Margin Classifiers

One important goal of classification is to minimize the prediction error rate. In particular, let the training data be $\{(\boldsymbol{x}_i, y_i); i = 1, \ldots, n\}$, *i.i.d.* from a fixed but unknown distribution $\mathbb{P}(\boldsymbol{X}, Y)$, where $\boldsymbol{X} \in \mathbb{R}^p$ is a p-dimensional vector of predictors and Y is the corresponding label. A classification technique builds a model on the training dataset and yields a prediction rule $\hat{Y}(\boldsymbol{X})$, where $\hat{Y}(\boldsymbol{X})$ denotes the predicted label of $\boldsymbol{X}$. Then, for a new instance with only $\boldsymbol{x}$ observed, we use $\hat{Y}(\boldsymbol{x})$ as its prediction. The corresponding prediction error rate can be expressed as $E_{\mathbb{P}(\boldsymbol{X},Y)}\big(I(\hat{Y}(\boldsymbol{X}) \neq Y)\big)$, where $I(\cdot)$ is the indicator function.

There are many classification techniques available in the literature. Well known classification methods include nearest-neighbors, linear discriminant analysis, logistic regression, tree based method, and many others. For the nearest-neighbors method, or sometimes referred to as the k-nearest-neighbors, the class label of $\boldsymbol{x}$ is determined by the majority of y's of k data points that are the closest to $\boldsymbol{x}$. Linear discriminant analysis assumes that the marginal distribution of each $\mathbb{P}(\boldsymbol{X}|Y)$ is normal, and estimates the parameters of the density by the observed data. Then for a new $\boldsymbol{x}$, we calculate its likelihood of being in each class, and the label is assigned according to which likelihood is the maximum. Logistic regression method models the log-odds of a data point as a function of $\boldsymbol{x}$, and find the corresponding maximum likelihood estimator. With the estimated parameters at hand, the prediction rule is to assign the label with the corresponding maximum probabilities. For tree based methods, one splits the space of $\boldsymbol{x}$ into a set of rectangles. Within each rectangle, the predicted label is the same as that of the majority of observations in that rectangle. See Hastie, Tibshirani and Friedman [2009] for a comprehensive review of various methods.

The classifiers aforementioned often work well in the traditional setting where we only have a relatively small number of predictors. With the advance of technology, the dimensionality of modern data in various scientific fields keeps increasing. Such high dimensional and complex data pose challenges for the development of suitable statistical techniques. There are various extensions of those methods mentioned above. In this dissertation, we focus on a set of machine learning based classification techniques, namely the large margin classifiers, that can handle high dimensional data well.

In the literature of large margin classification, where the terminology of large margin is to be defined below, one assumes that $Y \in \{+1, -1\}$. To classify the instances, typically one maps $\boldsymbol{X}$ onto the real line $\mathbb{R}$ through a classification function $f(\boldsymbol{X})$, and the corresponding prediction rule is $\hat{Y} = +1$ if $f(\boldsymbol{X}) \geq 0$, and $\hat{Y} = -1$ if $f(\boldsymbol{X}) < 0$. We assume the classification function $f(\cdot)$ belongs to a certain functional class $\mathcal{F}$. For example, one can consider $\mathcal{F}$ as a class of linear functions $f(\boldsymbol{X}) = \boldsymbol{X}^T\boldsymbol{\beta} + \beta_0$, where $\boldsymbol{\beta}$ is a parameter vector of length p and β_0 is the intercept.

To minimize the expected prediction error rate $E_{\mathbb{P}(\boldsymbol{X},Y)}\big(I(\hat{Y}(\boldsymbol{X}) \neq Y)\big)$, we may consider its empirical approximation

$$\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} I(\text{sign}(f(\boldsymbol{x}_i)) \neq y_i),$$

or equivalently,

$$\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} I(y_i f(\boldsymbol{x}_i) \leq 0). \tag{1.1}$$

Note that in (1.1), $I(\cdot)$ depends on y and $f(\boldsymbol{x})$ only through their product $yf(\boldsymbol{x})$. This product $yf(\boldsymbol{x})$, called the functional margin, plays an important role in a large margin classification technique. In particular, for a data point $(\boldsymbol{x}, y)$, the classification is correct if and only if $yf(\boldsymbol{x}) > 0$. Classification methods that are based on the functional margin $yf(\boldsymbol{x})$ are referred to as large margin classifiers.

The optimization problem (1.1) is typically NP-hard, hence it is common to apply a surrogate loss function $\ell(\cdot)$ in place of the indicator function $I(\cdot)$, plus a penalty term on f. The optimization problem can thus be written in a *loss + penalty* form

$$\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i f(\boldsymbol{x}_i)) + \lambda J(f). \tag{1.2}$$

Here $J(f)$ is the regularization term, or sometimes referred to as the penalty term on f, which prevents it from overfitting, and λ serves as a balance between the loss and the penalty term. A proper choice of λ is crucial in practice.

Different choices of the loss function ℓ lead to different binary large margin classifiers. To list a few, AdaBoost in Boosting [Freund and Schapire, 1997; Friedman, Hastie and Tibshirani, 2000] is shown to be approximately using the exponential loss $\ell(u) = \exp(-u)$, Penalized Logistic Regression [Lin et al., 2000] uses the deviance loss $\ell(u) = \log(1 + \exp(-u))$, Support Vector Machines (SVMs, Boser, Guyon and Vapnik [1992]) are equivalent to using the hinge loss $\ell(u) = (1-u)_+$, and Proximal SVMs [Fung and Mangasarian, 2001; Suykens and Vandewalle, 1999; Tang and Zhang, 2005] use $\ell(u) = (1-u)^2$. See Hastie, Tibshirani and Friedman [2009] for more discussions. Many of these large margin classifiers are shown to be very useful in practice, and the theoretical properties of these methods are well studied (see, among others, Steinwart and Scovel [2007], Blanchard, Bousquet and Massart [2008], Zhang [2004*b*], Bartlett, Jordan and McAuliffe [2006] and Shawe-Taylor and Cristianini [2004]).

In the next section, we introduce some extensions of binary classifiers for multicategory problems. The techniques for a multicategory classifier can be generalized from a binary one in many different ways, and we give a brief review of some existing multicategory classification methods.

1.1.2 Generalization to Multicategory Classification

In Sections 1.1.1, the focus has been on binary classification methods, especially the large margin classifiers. We first introduce how to extend binary large margin classification techniques to the multicategory case with the response variable consisting of more than two classes.

In a multicategory classification problem, instead of using $+1$ and -1 as the labels, we let $Y \in$

$\{1,\ldots,k\}$, where k is the number of classes. Similar as the binary case, we need to minimize the prediction error rate $E_{\mathbb{P}(\boldsymbol{X},Y)}\big(I(\hat{Y}(\boldsymbol{X}) \neq Y)\big)$. A natural idea for generalization is to employ a sequence of binary classifiers for the multicategory problem. There are two major approaches in this framework, namely, the one-versus-one and one-versus-rest methods.

The one-versus-one approach builds binary classifiers between classes i and j, for all different pairs of $i, j \in \{1,\ldots,k\}$. With $k(k-1)/2$ classifiers at hand, the prediction rule is based on a majority vote. In particular, a new instance $\boldsymbol{x}$ is labeled as class 1, if it is predicted as class 1 more often than the other classes among the $k(k-1)/2$ prediction results. When there is a tie, one may need to use random guessing to make a decision. The one-versus-rest approach takes class i as the positive class, and all other classes as the negative one, and trains k different classifiers. When a new instance is available, the prediction rule is to run k classifiers and choose the one that has the largest classification function value. The one-versus-rest approach may suffer from inconsistency when there is no dominating class for certain large margin classifiers such as the SVM [Liu and Yuan, 2011]. Hence it is desirable to study a classifier that takes all classes into consideration simultaneously.

For simultaneous multicategory large margin classification, one often aims to calculate a k-dimensional classification function vector $\boldsymbol{f}(\boldsymbol{X}) = (f_1(\boldsymbol{X}),\ldots,f_k(\boldsymbol{X}))^T$, where $\boldsymbol{f}$ belongs to some certain functional class $\mathcal{F}$ of interest. The prediction rule is $\hat{Y}(\boldsymbol{X}) = \text{argmax}_{j\in\{1,\ldots,k\}} f_j(\boldsymbol{X})$. Because of the argmax rule, if we add a constant to each element in $\boldsymbol{f}$, the prediction does not change. To overcome this difficulty and to reduce the dimension of the problem, it is common to impose a sum-to-zero constraint on $\boldsymbol{f}$. Namely, we restrict $\boldsymbol{f}$ such that $\sum_{j=1}^{k} f_j(\boldsymbol{X}) = 0$ for all $\boldsymbol{X}$. Many existing simultaneous classifiers follow this framework. See for example, Vapnik [1998], Crammer et al. [2001], Lee, Lin and Wahba [2004], Zhu and Hastie [2005], Liu and Shen [2006], Tang and Zhang [2006], Zhu et al. [2009], Liu and Yuan [2011] and Zhang and Liu [2013]. However, although the sum-to-zero constraint helps to ensure identifiability of the solution, it makes the computational problem more complex. In Chapter 4, we propose a new simplex based multicategory classification structure that is free of the sum-to-zero constraint, and hence it enjoys a faster computational speed.

In Section 1.1.3, we will discuss the issue of *class conditional probability* estimation.

1.1.3 Class Conditional Probability Estimation

In this section, we introduce the definition of class conditional probability for a binary case. The generalization to the multicategory case will be discussed later. In practice, besides accurate classification, an appropriate estimation of the class conditional probability can be important as well.

Recall from the previous section that we assume the data are *i.i.d.* from $\mathbb{P}(\boldsymbol{X}, Y)$. For a given $\boldsymbol{X} = \boldsymbol{x}$, suppose that, without loss of generality, $\mathbb{P}(\boldsymbol{x}, Y = +1) + \mathbb{P}(\boldsymbol{x}, Y = -1) > 0$. We define the probability $P(Y = +1|\boldsymbol{X} = \boldsymbol{x}) = \frac{\mathbb{P}(\boldsymbol{x},Y=+1)}{\mathbb{P}(\boldsymbol{x},Y=+1)+\mathbb{P}(\boldsymbol{x},Y=-1)}$ to be the class conditional probability of the positive class. The class conditional probability of the negative class can be defined in a similar manner. Intuitively, this class conditional probability indicates the chance of Y being in class $+1$ for a given predictor $\boldsymbol{x}$. In certain practical problems, an accurate estimation of the class conditional probability may help to provide more information on how confident the classification is.

We will explore the effect of the penalty term $J(f)$ on the estimation of class conditional probabilities in Chapter 2. In particular, we show that the penalty term tends to shrink the probability estimation towards 1/2 in binary cases, and we propose a simple refit procedure that helps to correct the bias.

In the literature of large margin classification, there exists two major groups, called the *soft* and *hard* classifiers Wahba [1999, 2002]. The essential difference between soft and hard classifiers is whether one needs to estimate the class conditional probability for the classification task or not. In particular, soft classifiers predict the label based on the obtained class conditional probabilities, while hard classifiers bypass the estimation of probabilities and focus on the decision boundary. In practice, for the goal of accurate classification, it is unclear which one to use in a given situation.

Recently, Liu, Zhang and Wu [2011] proposed a family of binary loss functions, namely the the Large-margin Unified Machines (LUM), that embraces many existing loss functions as special cases. In particular, the LUM loss can be expressed as

$$\ell(u) = \begin{cases} 1 - u & \text{if } u < \frac{c}{1+c} \\ \frac{1}{1+c}\left(\frac{a}{(1+c)u-c+a}\right)^a & \text{if } u \geq \frac{c}{1+c}, \end{cases}$$

with $c \geq 0$ and $a > 0$ being parameters of the LUM family. Note that the LUM family includes the SVM hinge loss with $c \to \infty$, and the Distance Weighted Discrimination (DWD, Marron, Todd and Ahn [2007]) with $c = 1$ and $a = 1$, as special cases. When $c = 0$, the classifier is a typical soft classifier which provides complete class conditional probability information. As c increases, the probability information becomes more vague. As $c \to \infty$, it becomes the SVM which only estimates the classification boundary $\{x : P(Y = +1|\boldsymbol{X} = \boldsymbol{x}) = 0.5\}$. The LUM loss function can be used to study the transition behavior from soft to hard classifiers. Liu, Zhang and Wu [2011] showed that for binary problems, an accurate estimation of class conditional probability can sometimes help to build a more accurate classifier, while in some other cases focusing on the classification boundary only may help to establish a more accurate classifier. We will extend this idea to multicategory problems in Chapter 3.

1.2 New Contributions and Outline

In this dissertation, we will investigate various aspects of large margin classifiers. We investigate the role of class conditional probability as well as its estimation. Furthermore, we will propose some new large margin classification techniques. The main outline of the dissertation is as follows:

- In Chapter 2, we investigate the problem of probability estimation for binary large margin classifiers and illustrate the resulting bias on class probability estimation caused by shrinkage. We show that such bias can be large for finite sample problems. As a result, modifications are needed. We propose a simple refit method [Zhang, Liu and Wu, 2013] that helps to correct the scale problem introduced by shrinkage and yields accurate class probability estimation.

- In Chapter 3, we tackle the problem of hard and soft classifiers in multicategory classification problems. The LUM family, proposed in Liu, Zhang and Wu [2011], enables one to study the behavior change from soft to hard binary classifiers. For multicategory cases, however, the concept of soft and hard classification becomes less clear. In that case, class probability estimation becomes more involved as it requires estimation of a probability vector. We generalize the idea in Liu, Zhang and Wu [2011] and propose a new Multicategory LUM [MLUM, Zhang and Liu, 2013] framework to investigate the behavior of soft versus hard classification under multicategory settings.

- In Chapter 4, we propose a new group of simultaneous multicategory large margin classification techniques. Among existing simultaneous multicategory extensions, a common one is to learn k different functions for a k class problem together with a sum-to-zero constraint on the functions. Such a formulation can be inefficient for multicategory learning. In this chapter, we propose a new Multicategory Angle-based large margin Classification [MAC, Zhang and Liu, 2014] technique. The proposed MAC associates different classes with vertices of a standard simplex. The prediction rule is to assign the class label that corresponds to the vertex whose angle with respect to the classification function is the smallest. All binary large margin classifiers can be naturally generalized to simultaneous multicategory classifiers through the MAC structure. MAC is free of the commonly used sum-to-zero constraint, and consequently enjoys more efficient computation.

- In Chapter 5, we propose a new multicategory support vector machine under the angle based framework. The new machine overcomes the difficulty that the regular angle based support vector machine is Fisher inconsistent. In particular, we propose to use the idea of convex combination of loss functions as in Liu and Yuan [2011]. We show that with appropriately chosen penalty, the corresponding optimization can be solved using the coordinate descent algorithm, which is extremely fast.

CHAPTER 2: SHRINKAGE ON PROBABILITY ESTIMATION

2.1 Introduction

In classification problems, the classification accuracy is one of the most important measures of classification performance. An accurate classifier can produce good prediction of class membership for new subjects. Another important issue is class probability estimation. As there are random errors involved in the class prediction, class probability estimation gives users information on how strong the evidence of classifying one subject into a particular class is. The problem of class probability estimation can be even more important than classification accuracy. For example, in disease diagnosis, it is vital for doctors and patients to know the chance of a certain disease instead of just a prediction. For two patients classified into the disease positive class, if their respective class probabilities are 0.501 and 0.999, the probability information is undoubtedly critical.

Our focus is on margin-based, sometimes called large margin, classification techniques. Such techniques can typically be written as a regularization problem of minimizing *Loss* + *Penalty*. Here, the loss term is used to ensure goodness of fit of the resulting model on the training data. The penalty term, also known as the regularization term, prevents overfitting through shrinkage so that the resulting model can produce accurate predictions. Many important classification techniques fit into the regularization framework, for example, Penalized Logistic Regression [PLR, Lin et al., 2000], AdaBoost in Boosting [Freund and Schapire, 1997; Friedman, Hastie and Tibshirani, 2000], Import Vector Machine [Zhu and Hastie, 2005], Proximal SVM [PSVM, Fung and Mangasarian, 2001; Suykens and Vandewalle, 1999; Tang and Zhang, 2005], ψ-learning [Shen et al., 2003], and more recently, Large-margin Unified Machines [Liu, Zhang and Wu, 2011]. The regularization term is especially important for high dimensional data analysis.

One can use the loss function, or sometimes the related likelihood function, to derive a formula for probability estimation using the classification function. For example, in PLR, one can use the logit transformation, i.e., the link function between the classification function and the class conditional probability, to estimate the probability. Lin [2000] showed that under certain conditions, the PLR probability estimator converges to the true probability as the training sample size gets large. It is common in practice to use such an estimator of probability, without taking the shrinkage effect into account. In this chapter, we demonstrate that when the sample size n is relatively small compared to the dimension p, the shrinkage effect can be very large on probability estimation. In practice, such probability estimators can have sizeable biases and consequently may give misleading results. Our goal is to explore the shrinkage effect on classification and more importantly on probability estimation. Through theoretical studies, we demonstrate how shrinkage affects probability estimation in binary classification problems. In particular, we show that shrinkage tends to force the resulting naive probability estimator towards 1/2 in standard learning, where both classes are treated equally. Inspired by this phenomenon, we explore new methods to achieve better probability estimation. Our proposed refit method is shown to give consistent probability estimation, and works remarkably well in the numerical examples. For illustration, we focus on PLR and PSVM in this chapter, however, our idea is applicable to general margin-based classifiers as well.

In Section 2.2, we review large margin classification techniques and explore some theoretical properties of shrinkage for several methods. Our results shed some light on poor probability estimation without adjusting the shrinkage effect. Both standard and weighted learning settings are considered. In Section 3, we propose a new two-stage refit method for better probability estimation. Given the classification function of the penalized method from the first step, we refit the classifier as a one-dimensional problem without penalization to correct the shrinkage bias. We show that the refit method often has a large gain in probability estimation, while keeping similar classification performance to the first step. Some asymptotic consistency results of the refit procedure are provided as well. In Section 2.4, we use simulated examples to examine the performance of the refit approach. In Section 2.5, we evaluate the methods on two real data examples. Some discussion is provided in Section 2.6. All technical proofs are collected in Section 2.7.

2.2 Large margin classifiers and the shrinkage effect

2.2.1 Framework of large margin classifiers

In supervised learning, we have a training data set $\{(\boldsymbol{x}_i, y_i); i = 1, \ldots, n\}$ which contains n observations, where, with a little bit abuse of notation, let $\boldsymbol{x}_i \in \mathbb{R}^p$ be a $p-$dimensional covariate vector and

y_i the response variable. The dimensionality of the problem p and the class conditional probability $p(\boldsymbol{x})$ defined later should not be confused. When y is a continuous variable, we have the well-known regression problem. In that case, it is common to assume that the data are *i.i.d.* observations according to an unknown probability distribution $\mathbb{P}(\boldsymbol{x}, y)$ and the goal is to estimate $E(y|\boldsymbol{x})$. When y is categorical, we then have a classification problem. Our focus in this chapter is on binary classification with $y \in \{\pm 1\}$.

For classification problems, it is common to have independent samples for each class, obtained from $\mathbb{P}(\boldsymbol{x}|y)$. The sample class proportions, however, can be different from population class proportions. Assume that π and $1-\pi$ are the proportions of positive and negative classes in the population, respectively. Similarly, π_s and $1-\pi_s$ are the class proportions of the sample. Then the joint distribution for the sample is $P_s(\boldsymbol{x}, y) = \mathbb{P}(\boldsymbol{x}|y=1)\pi_s + \mathbb{P}(\boldsymbol{x}|y=-1)(1-\pi_s)$ and the population joint distribution $\mathbb{P}(\boldsymbol{x}, y) = \mathbb{P}(\boldsymbol{x}|y=1)\pi + \mathbb{P}(\boldsymbol{x}|y=-1)(1-\pi)$. When there is sampling bias with $\pi \neq \pi_s$, one needs to make adjustments after obtaining estimation for $P_s(y=1|\boldsymbol{x})$. In particular, we can show that the population odds $P(y=1|\boldsymbol{x})/(1-P(y=1|\boldsymbol{x}))$ and the sample odds $P_s(y=1|\boldsymbol{x})/(1-P_s(y=1|\boldsymbol{x}))$ satisfy that

$$\frac{P(y=1|\boldsymbol{x})}{1-P(y=1|\boldsymbol{x})} = \frac{P_s(y=1|\boldsymbol{x})}{1-P_s(y=1|\boldsymbol{x})} \frac{(1-\pi_s)\pi}{\pi_s(1-\pi)}.$$

For simplicity, we first assume both the sample and population are from the same distribution $\mathbb{P}(\boldsymbol{x}, y)$. When there is sampling bias, we can use weighted learning in Section 2.2.3 to adjust the sampling bias.

To classify a new input vector $\boldsymbol{x}$, a classification discrimination function f is estimated from the training data set, and $\text{sign}[f(\boldsymbol{x})]$ is used as the predicted label. In the regularization framework, we solve the following optimization problem

$$\min_{f \in \mathcal{F}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell[y_i f(\boldsymbol{x}_i)] + \lambda J(f) \right\}, \tag{2.1}$$

where $\ell(\cdot)$ is a loss function that uses the functional margin to ensure goodness of fit of the model on the training data, $\mathcal{F}$ is the functional space of interest and $J(f)$ is a regularization term on f to avoid overfitting. The tuning parameter λ balances the two terms in (2.1) to ensure good generalization abilities of the resulting classifier for future prediction. A proper choice of λ is very important.

The loss function $\ell(\cdot)$ is typically pre-specified, and differs among various methods. For example, PLR uses the deviance loss $\ell(u) = \log(1 + e^{-u})$, AdaBoost is shown to be approximately equivalent to using the exponential loss $\ell(u) = e^{-u}$ [Friedman, Hastie and Tibshirani, 2000], SVM use the hinge loss $\ell(u) = (1-u)_+$, and PSVM use the squared error loss $\ell(u) = (1-u)^2$ [Fung and Mangasarian, 2001], which is essentially equivalent to least squares linear regression with binary response $Y \in \{\pm 1\}$. Square error loss has a close connection with linear discriminant analysis. In particular, if the class label Y is coded in a certain way, then minimizing the empirical square errors without regularization leads to Fisher's linear discriminant function. (See Chapter 4 of Hastie, Tibshirani and Friedman [2009]).

As different loss functions yield various methods, it is essential to study the properties of these loss functions. One important concept for a loss function is Fisher consistency [Lin, 2004; Bartlett, Jordan and McAuliffe, 2006], defined as follows. For a standard binary classification problem, the corresponding loss function $\ell(\cdot)$ is Fisher consistent if and only if $\text{sign}[f^*(\boldsymbol{x})] = \text{sign}[p(\boldsymbol{x}) - \frac{1}{2}]$, where $f^*(\boldsymbol{x}) = \text{argmin}_f E\{\ell[Yf(\boldsymbol{X})]|\boldsymbol{X} = \boldsymbol{x}\}$ and $p(\boldsymbol{x}) = P(Y = +1|\boldsymbol{X} = \boldsymbol{x})$. As we can see, Fisher consistency essentially ensures that a classifier is consistent in the classification sense. In practice, the resulting classification boundary asymptotically approaches the theoretically optimal boundary, i.e., the Bayes decision boundary $\{\boldsymbol{x} : p(\boldsymbol{x}) = 1/2\}$. Note that the terminology of Bayes decision boundary is a commonly used term to refer to the best theoretical classification boundary in the literature and the corresponding smallest error is known as the Bayes error [Hastie, Tibshirani and Friedman, 2009]. Fisher consistency is a weak requirement on the loss function of a classifier. Lin [2004] showed that a loss function $\ell(\cdot)$ is Fisher consistent if it satisfies

A.1. $\ell(u) < \ell(-u), \forall u > 0$.

A.2. $\ell'(0)$ exists.

Here $\ell'(u)$ is the derivative of $\ell(u)$. All the aforementioned loss functions satisfy **A.1** and **A.2** and thus are all Fisher consistent.

Ideally, we would like to transform the classification function $f(\boldsymbol{x})$ to estimate the class conditional probability $p(\boldsymbol{x})$. For PLR, we use the inverse logit transformation on $f(\boldsymbol{x})$ to estimate $p(\boldsymbol{x})$. Thus, once $\hat{f}(\boldsymbol{x})$ is obtained, we estimate $p(\boldsymbol{x})$ accordingly. Our goal is to explore a general Fisher consistent loss function $\ell(\cdot)$ and investigate conditions for us to estimate $p(\boldsymbol{x})$ through some transformation of $f(\boldsymbol{x})$.

In order to estimate $p(\boldsymbol{x})$ from $f(\boldsymbol{x})$, a natural condition on $\ell(\cdot)$ is to have a one-to-one mapping between $f^*(\boldsymbol{x})$ and $p(\boldsymbol{x})$. Theorem 2.2.1 below provides conditions on $\ell(\cdot)$ so that such a one-to-one correspondence exists. Note that a similar theorem with different assumptions was developed in Zou, Zhu and Hastie [2008*a*]. We denote by $\ell''(u)$ the second derivative of $\ell(u)$.

Theorem 2.2.1. *The following conditions are sufficient for the minimizer*

$$f^*(\boldsymbol{x}) = \underset{f}{\operatorname{argmin}}\, E\{\ell[Yf(\boldsymbol{X})]|\boldsymbol{X} = \boldsymbol{x}\}$$

and $p(\boldsymbol{x})$ to have a one-to-one correspondence:

B.1. *$\ell(u)$ is twice differentiable with respect to u.*

B.2. *$\ell'(u) + \ell'(-u) < 0, \forall u$.*

B.3. *$\ell'(u)\ell''(-u) + \ell'(-u)\ell''(u) < 0, \forall u$.*

Under these conditions, the mapping between $f^*(\boldsymbol{x})$ and $p(\boldsymbol{x})$ is $p(\boldsymbol{x}) = \frac{\ell'[-f^*(\boldsymbol{x})]}{\ell'[f^*(\boldsymbol{x})]+\ell'[-f^*(\boldsymbol{x})]}$.

Most of the loss functions mentioned earlier, for example the deviance loss, the exponential loss, and the squared error loss, satisfy the conditions in Theorem 2.2.1. Consequently, the corresponding class probability $p(\boldsymbol{x})$ can be estimated using the relationship between $p(\boldsymbol{x})$ and $f^*(\boldsymbol{x})$ provided in Theorem 2.2.1. For the hinge loss of SVM, $f^*(\boldsymbol{x}) = \text{sign}[p(\boldsymbol{x}) - 0.5]$, and one cannot estimate $p(\boldsymbol{x})$ directly.

When the conditions of Theorem 2.2.1 hold, we call the solution

$$p_0(f) = \frac{\ell'(-f)}{\ell'(f) + \ell'(-f)}, \tag{2.2}$$

the original method. Once $\hat{f}(\boldsymbol{x})$ is obtained, the estimator of $p(\boldsymbol{x})$ using the original method becomes $p_0[\hat{f}(\boldsymbol{x})]$. Sometimes $p_0[\hat{f}(\boldsymbol{x})]$ may be outside of $[0, 1]$. When $p_0[\hat{f}(\boldsymbol{x})] < 0$, or > 1, it is typically set to be 0, or 1, respectively. Notice that such an approach pays attention only to the first term in (1) for class probability estimation. Asymptotically the original probability estimator is consistent under various conditions [Lin, 2000]. However in practice, when the sample size is moderate or small, the shrinkage effect from the regularization term in (2.1) can be large. Consequently, the original method for probability estimation can be severely biased. We will demonstrate the shrinkage effect both theoretically and numerically. In Sections 2.2.2 and 2.2.3, we will explore the theoretical impact of shrinkage on class probability estimation for both standard and weighted learning settings.

2.2.2 Theoretical property of shrinkage

As we pointed out earlier, ignoring the effect of the regularization term $J(f)$ in (2.1) may create bias in class conditional probability estimation. Next we explore how this regularization term creates shrinkage on the estimation of the classification function $f(\boldsymbol{x})$, which leads to a large gap between the true class conditional probability and its original estimation. For simplicity, we consider linear learning with $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta}$. When the linear functional space spanned by $\boldsymbol{x}$ is insufficient, one may consider a higher, possibly infinite, dimensional space spanned by $\phi(\boldsymbol{x})$, where $\phi(\cdot)$ is the mapping of $\boldsymbol{x}$ from the linear space to a higher dimensional space. One may specify $\phi(\boldsymbol{x})$ explicitly, and perform learning with $f(\boldsymbol{x}) = \phi(\boldsymbol{x})^T\boldsymbol{\beta}$. Such an approach can be difficult to implement when an infinite dimensional space is needed. Another approach is to perform the mapping implicitly using the so-called *kernel* trick, with $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2)\rangle$, where $K(\cdot,\cdot)$ is a *kernel* function. With a given kernel function, one can perform kernel learning without explicitly specifying $\phi(\cdot)$. More details about the kernel learning and kernel trick can be found in Cristianini and Shawe-Taylor [2000], Schölkopf and Smola [2002] and Wahba [1999]. The Gaussian kernel is a commonly used non-linear kernel with $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\frac{(\boldsymbol{x}_1-\boldsymbol{x}_2)^T(\boldsymbol{x}_1-\boldsymbol{x}_2)}{\sigma^2})$, where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are two covariates in the original space and σ is a fixed constant. Our idea and method can be directly extended to the kernel framework, and we do not include the details here.

In the linear learning setup, we assume that the first coordinate of $\boldsymbol{x}$ corresponds to the constant term and as a result, the first element of $\boldsymbol{\beta}$ represents the intercept term β_0 of the linear function. In our theoretical exploration, for simplicity, we let $J(f) = \|\boldsymbol{\beta}\|^2 = \boldsymbol{\beta}^T\boldsymbol{\beta}$ be the regularization term. Note that here $J(f)$ includes β_0 as well. In practice the intercept is often not penalized.

To explore the effect of shrinkage, ideally, we should study $\operatorname{argmin}_f E\{\ell[Yf(\boldsymbol{X})] + J(f)\}$. However, we cannot derive the solution directly since it depends on the underlying distribution $\mathbb{P}(\boldsymbol{x}, y)$. Instead, we consider the conditional minimizer

$$f^{**}(\boldsymbol{x}) = \operatorname*{argmin}_f E\{\ell[Yf(\boldsymbol{X})] + J(f)|\boldsymbol{X} = \boldsymbol{x}\}.$$

Notice that this definition of $f^{**}(\boldsymbol{x})$ is pointwise in terms of $\boldsymbol{x}$, just as $f^*(\boldsymbol{x})$. In linear learning with

$f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta}$ and $J(f) = \|\boldsymbol{\beta}\|^2 = \boldsymbol{\beta}^T\boldsymbol{\beta}$, this is equivalent to finding

$$\boldsymbol{\beta}^{**}(\boldsymbol{x}) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, E\{\ell[Y\boldsymbol{X}^T\boldsymbol{\beta}] + \lambda\|\boldsymbol{\beta}\|^2|\boldsymbol{X} = \boldsymbol{x}\}. \tag{2.3}$$

Note that the solution $\boldsymbol{\beta}^{**}(\boldsymbol{x})$ in (2.3) depends on $\boldsymbol{x}$. Although our derivation is conditioned on $\boldsymbol{x}$, it can help to reveal the effect of shrinkage on probability estimation.

To calculate (2.3), define $S[\boldsymbol{\beta}(\boldsymbol{x})] = E[\ell(Y\boldsymbol{X}^T\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|^2|\boldsymbol{X} = \boldsymbol{x}] = p(\boldsymbol{x})\ell[\boldsymbol{x}^T\boldsymbol{\beta}(\boldsymbol{x})] + [1 - p(\boldsymbol{x})]\ell[-\boldsymbol{x}^T\boldsymbol{\beta}(\boldsymbol{x})] + \lambda\|\boldsymbol{\beta}(\boldsymbol{x})\|^2$. In order to minimize $S[\boldsymbol{\beta}(\boldsymbol{x})]$, we solve $\frac{\partial S[\boldsymbol{\beta}(\boldsymbol{x})]}{\partial \boldsymbol{\beta}(\boldsymbol{x})}|_{\boldsymbol{\beta}(\boldsymbol{x})=\boldsymbol{\beta}^{**}(\boldsymbol{x})} = 0$. Thus we have

$$p(\boldsymbol{x})\ell'[\boldsymbol{x}^T\boldsymbol{\beta}^{**}(\boldsymbol{x})]\boldsymbol{x} - [1 - p(\boldsymbol{x})]\ell'[-\boldsymbol{x}^T\boldsymbol{\beta}^{**}(\boldsymbol{x})]\boldsymbol{x} + 2\lambda\boldsymbol{\beta}^{**}(\boldsymbol{x}) = 0,$$

which is equivalent to

$$p(\boldsymbol{x})\boldsymbol{x} = \frac{\ell'[-f^{**}(\boldsymbol{x})]}{\ell'[-f^{**}(\boldsymbol{x})] + \ell'[f^{**}(\boldsymbol{x})]}\boldsymbol{x} - \frac{2\lambda}{\ell'[-f^{**}(\boldsymbol{x})] + \ell'[f^{**}(\boldsymbol{x})]}\boldsymbol{\beta}^{**}(\boldsymbol{x}). \tag{2.4}$$

Let $A[f^{**}(\boldsymbol{x})] = -\frac{2}{\ell'[-f^{**}(\boldsymbol{x})]+\ell'[f^{**}(\boldsymbol{x})]}$. Using the definition of $A[f^{**}(\boldsymbol{x})]$ and (2.2), (2.4) becomes

$$p(\boldsymbol{x})\boldsymbol{x} = p_0[f^{**}(\boldsymbol{x})]\boldsymbol{x} + \lambda A[f^{**}(\boldsymbol{x})]\boldsymbol{\beta}^{**}(\boldsymbol{x}). \tag{2.5}$$

Note that both $p_0[f^{**}(\boldsymbol{x})]$ and $A[f^{**}(\boldsymbol{x})]$ are scalars. Since $p(\boldsymbol{x})$ is fixed for a given $\boldsymbol{x}$, in order to have (2.5) hold, $\boldsymbol{\beta}^{**}(\boldsymbol{x})$ satisfies $\boldsymbol{\beta}^{**}(\boldsymbol{x}) = c(\boldsymbol{x})\boldsymbol{x}$, where $c(\boldsymbol{x}) = \frac{p(\boldsymbol{x})-p_0[f^{**}(\boldsymbol{x})]}{\lambda A[f^{**}(\boldsymbol{x})]}$ is a scalar that depends on $\boldsymbol{x}$. This implies that $\boldsymbol{\beta}^{**}(\boldsymbol{x})$ is a function of $\boldsymbol{x}$ and it varies according to different $\boldsymbol{x}$ because we derive such a relationship for a fixed $\boldsymbol{x}$. However, in practice, we calculate a common $\boldsymbol{\beta}$ for all $\boldsymbol{x}$'s. Nevertheless, our derivation on each fixed $\boldsymbol{x}$ using conditional expectation helps to shed some light on the effect of shrinkage.

To further simplify (2.5), for any p-dimensional vector $\boldsymbol{z}$ with $\boldsymbol{z}^T\boldsymbol{x} \neq 0$, we have

$$p(\boldsymbol{x})\boldsymbol{z}^T\boldsymbol{x} = p_0[f^{**}(\boldsymbol{x})]\boldsymbol{z}^T\boldsymbol{x} + \lambda A[f^{**}(\boldsymbol{x})]\boldsymbol{z}^T\boldsymbol{\beta}^{**}(\boldsymbol{x}),$$

and consequently we obtain the expression of $p(\boldsymbol{x})$ as

$$p(\boldsymbol{x}) = p_0[f^{**}(\boldsymbol{x})] + \lambda A[f^{**}(\boldsymbol{x})]\frac{\boldsymbol{z}^T\boldsymbol{\beta}^{**}(\boldsymbol{x})}{\boldsymbol{z}^T\boldsymbol{x}}. \tag{2.6}$$

If we set $\boldsymbol{z} = \boldsymbol{x}$, with $\boldsymbol{\beta}^{**}(\boldsymbol{x}) = c(\boldsymbol{x})\boldsymbol{x}$, we have $c(\boldsymbol{x}) = \frac{\boldsymbol{\beta}^{**}(\boldsymbol{x})^T\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} = \frac{f^{**}(\boldsymbol{x})}{\|\boldsymbol{x}\|^2}$. Thus $\text{sign}[c(\boldsymbol{x})] = \text{sign}[f^{**}(\boldsymbol{x})] = \text{sign}\{p_0[f^{**}(\boldsymbol{x})] - 0.5\}$, where the last equality follows the fact that the function $p_0(\cdot)$ in (2.2) is strictly increasing and $p_0(0) = 0.5$. Thus (2.6) can be expressed as

$$p(\boldsymbol{x}) = p_0[f^{**}(\boldsymbol{x})] + \lambda A[f^{**}(\boldsymbol{x})] \cdot |c(\boldsymbol{x})| \cdot \text{sign}\{p_0[f^{**}(\boldsymbol{x})] - 0.5\}, \tag{2.7}$$

where $A[f^{**}(\boldsymbol{x})] = -\frac{2}{\ell'[-f^{**}(\boldsymbol{x})]+\ell'[f^{**}(\boldsymbol{x})]} > 0$. Comparing to the formula of $p(\boldsymbol{x}) = p_0[f^{**}(\boldsymbol{x})]$ in Theorem 2.2.1, we have an extra term $t(\lambda) = \lambda A[f^{**}(\boldsymbol{x})] \cdot |c(\boldsymbol{x})| \cdot \text{sign}\{p_0[f^{**}(\boldsymbol{x})] - 0.5\}$, which comes from the regularization term $J(f)$. Interestingly, $t(\lambda)$ has the same sign as $p_0[f^{**}(\boldsymbol{x})] - 0.5$. When $p_0[f^{**}(\boldsymbol{x})] > 0.5$, $p(\boldsymbol{x}) = p_0[f^{**}(\boldsymbol{x})] + t(\lambda) > p_0[f^{**}(\boldsymbol{x})]$. This implies that using $p_0[f^{**}(\boldsymbol{x})]$ underestimates $p(\boldsymbol{x})$. Similarly, $p_0[f^{**}(\boldsymbol{x})]$ overestimates $p(\boldsymbol{x})$ when $p_0[f^{**}(\boldsymbol{x})] < 0.5$. As a result, we can conclude that shrinkage will push the original probability estimation towards 0.5 for binary classifiers. In Section 2.4, we confirm this finding via simulation and show the large biases of original probability estimation.

One important issue we would like to point out is that the formula (2.7) is derived using conditional expression for $\boldsymbol{X} = \boldsymbol{x}$. Thus strictly speaking, (2.7) is a correct way to estimate $p(\boldsymbol{x})$ if we have a solution of $\boldsymbol{\beta}^{**}(\boldsymbol{x})$ specific for each $\boldsymbol{x}$. This is certainly not feasible. For practical problems as given in (2.1), we need to solve for a common estimate of $\boldsymbol{\beta}$ using n observations. Thus (2.6) is not an applicable formula for the estimation of $p(\boldsymbol{x})$. In Section 2.3.1, we will introduce a simple refit method that works remarkably well. Next, we discuss the shrinkage effect on weighted learning.

2.2.3 Extension to weighted learning

So far, our focus has been on standard learning and we treat two classes equally. In this section we study the extension of shrinkage effect to weighted learning. Weighted learning can be useful in many situations. Here we briefly describe three scenarios: unequal costs, biased sampling, and unbalanced classification. Lin, Lee and Wahba [2002] previously discussed nonstandard situations such as unequal costs and biased sampling for the SVM.

Unequal costs are needed for many practical problems. For example, wrongly classifying a patient with a fatal disease to the healthy group may be viewed as substantially more costly than claiming the presence of the disease while it is not. In that case, unequal costs should be used to reflect the differences of these two types of misclassification.

Another important use of weighted learning is to adjust biased sampling. In many practical classification problems, the class proportions in the sample may be very different from those in the target population due to sampling bias. For example, if the two classes have very different proportions in the population, the smaller class may be oversampled, while the larger class may be undersampled so that the resulting sample can be more balanced. However, since we build the classifier based on the sample and predict classes of data from the population, this sampling bias can create problems. Weighted learning can be used to adjust such discrepancy.

Unbalanced classification is another case that weighted learning can be very effective. In standard learning it is common to evaluate the performance of a classifier by its overall prediction error rate. In real data applications, unbalanced classification problems can be challenging even when there is no sampling bias. For instance, in classifying patients into cancer versus non-cancer groups, we could have 99% healthy patients and 1% cancer patients in the sample. In that case, we may have a naive classifier that predicts all patients into the healthy group with a 99% overall classification accuracy. This classifier is certainly not desirable despite its good accuracy. To overcome this difficulty, one can use various weighted learning procedures [Qiao and Liu, 2009].

Denote by w_+ and w_- the weights for positive and negative classes, respectively. Then instead of (2.1), we solve the following optimization problem

$$\min_{f \in \mathcal{F}} \left\{ \frac{1}{n} \sum_{i=1}^{n} W(y_i) \ell[y_i f(\boldsymbol{x}_i)] + \lambda J(f) \right\},$$

where $W(y_i) = w_+$ if $y_i = +1$ and $W(y_i) = w_-$ otherwise. Here w_+ and w_- represent the weights for these two classes.

Denote $c(+1|-1)$ for the false-positive cost for points in the class -1 misclassified into the $+1$ class and similarly $c(-1|+1)$ for the false-negative cost for points in the class $+1$ misclassified into the -1 class. Then if the overall misclassification cost is used as the classification criterion, the optimal choice of w_+ and w_- is that $w_+ = \frac{c(-1|+1)\pi}{\pi_s}$ and $w_- = \frac{c(+1|-1)(1-\pi)}{1-\pi_s}$ [Qiao et al., 2010]. Note that both costs and class proportions are used in the construction of weights. Furthermore, estimators can be used if the true proportions are not available. More details on the justification of these weights as well as different classification criteria can be found in Qiao et al. [2010].

Next we show that our developments in Section 2.2.2 can be directly extended to weighted learning. The following proposition illustrates the Bayes boundary for weighted learning.

Proposition 2.1 [Wang, Shen and Liu, 2008]. Assume **A.1** and **A.2** hold. Then the minimizer $f^*(\boldsymbol{x}) = \operatorname{argmin}_f E\ \{W(Y)\ell[Yf(\boldsymbol{X})]|\boldsymbol{X} = \boldsymbol{x}\}$ satisfies $\operatorname{sign}[f^*(\boldsymbol{x})] = \operatorname{sign}[p(\boldsymbol{x}) - \frac{w_-}{w_+ + w_-}]$.

From Proposition 2.1, we can see that the new Bayes boundary for the population of interest incorporating the costs becomes $\{\boldsymbol{x} : p(\boldsymbol{x}) = \frac{w_-}{w_+ + w_-}\}$ for weighted learning. In Section 2.2.2, we show that with equal weights, the regularization term shrinks the probability estimation towards 1/2. In the weighted learning case, (2.4) becomes

$$p(\boldsymbol{x})\boldsymbol{x} = \frac{w_-\ell'[-f^{**}(\boldsymbol{x})]}{w_-\ell'[-f^{**}(\boldsymbol{x})] + w_+\ell'[f^{**}(\boldsymbol{x})]}\boldsymbol{x} - \frac{2\lambda}{w_-\ell'[-f^{**}(\boldsymbol{x})] + w_+\ell'[f^{**}(\boldsymbol{x})]}\boldsymbol{\beta}^{**}(\boldsymbol{x}).$$

If we define $A[f^{**}(\boldsymbol{x})] = -\frac{2}{w_-\ell'[-f^{**}(\boldsymbol{x})]+w_+\ell'[f^{**}(\boldsymbol{x})]}$ accordingly, using similar derivations as in Section 2.2.2, one can verify that (2.7) becomes

$$p(\boldsymbol{x}) = p_0[f^{**}(\boldsymbol{x})] + \lambda A[f^{**}(\boldsymbol{x})] \cdot |c(\boldsymbol{x})| \cdot \operatorname{sign}\{p_0[f^{**}(\boldsymbol{x})] - \frac{w_-}{w_+ + w_-}\}.$$

Thus, we can conclude that the regularization term now shrinks the probability estimation towards $\frac{w_-}{w_+ + w_-}$.

2.3 A new refit method for probability estimation

2.3.1 The refit procedure

Sections 2.2.2 and 2.2.3 demonstrate that although the shrinkage term in regularization helps to deliver accurate classification boundaries for large margin classifiers, it can adversely affect the accuracy of the probability estimation. Furthermore, it is difficult to derive an explicit correction term for the shrinkage effect on probability estimation. In this section, we propose a simple alternative to correct the biases introduced by shrinkage.

The idea of our refit method is as follows. In the linear case, we aim to estimate $\boldsymbol{\beta}$ so that $f = \boldsymbol{x}^T\boldsymbol{\beta}$

can yield class prediction based on whether $\text{sign}(f) > 0$ or not. From Section 2.2.2, we learned that although shrinkage affects the size of f, it does not change the sign of f. This implies that we can get a good classification direction through $\hat{f}$, although the scale may be too small for probability estimation due to shrinkage. Our idea is to make use of the solution $\hat{f}$ from the large margin classifier and project the data on this direction. As long as the classification error is good, as it is typically the case for large margin classifiers, the corresponding projection direction should be reasonable as well. Based on these considerations, we propose to refit the data on the projected one-dimensional space without penalty.

We would like to point out that the idea of refit is not entirely new. It has been used in the regression setting to improve regression parameter estimation. In particular, Meinshausen [2007] suggested a two-step fitting algorithm, the relaxed Lasso, to alleviate the problem of bias in the regression parameter estimation. He proposed to first apply the regular Lasso method [Tibshirani, 1996] to select a set of covariates as an "active set of variables", and then fit the Lasso again using the selected set of variables. The main idea of the Relaxed Lasso is to eliminate some unimportant variables in the first step. Then, the amount of shrinkage needed in the second step will be much smaller and consequently the resulting estimation bias can be alleviated compared to the original Lasso estimation. In contrast to regression, classification techniques aim to build accurate classification boundaries. Once we get a good classification direction vector using the decision boundary, we can project the data on this one-dimensional space to correct bias using refitting. Unlike the relaxed Lasso which requires regularization on the second step as well, we refit the data without shrinkage.

As discussed earlier, a refit step without shrinkage has no risk of overfitting since the projected space is only one-dimensional. However, this refit step can correct the scale bias caused by shrinkage on the original fit. After the refit step, we then use the original method to estimate $p(\boldsymbol{x})$, based on the obtained $\hat{f}$ from the refit step.

Next we use the standard PLR to illustrate our refit method, although the idea is the same for many other methods as well.

Our proposed procedure for the refit PLR is summarized as follows:

Step 1: (Original fit) Fit PLR on the training data to obtain $\hat{f} = \boldsymbol{x}^T\hat{\boldsymbol{\beta}}$. Proper tuning on λ is needed.

Step 2: (Projection) Create a new data set on the projected space. The new data set contains

$\{(\hat{\eta}_i, y_i); i = 1, \ldots, n\}$, where $\hat{\eta}_i = \boldsymbol{x}_i^T \hat{\boldsymbol{\beta}}$. Note that the new covariate space is only one-dimensional.

Step 3: (Refit) Fit the logistic regression without penalty on the new training data from Step 2 to get a new function $\hat{\hat{f}}(\hat{\eta}) = \hat{\gamma}_0 + \hat{\gamma}_1 \hat{\eta}$.

Step 4: (Probability estimation) Our final probability estimation formula becomes $\hat{p}(\boldsymbol{x}) = \frac{e^{\hat{\hat{f}}(\boldsymbol{x})}}{e^{\hat{\hat{f}}(\boldsymbol{x})}+1}$, where $\hat{\hat{f}}(\boldsymbol{x}) = \hat{\gamma}_0 + \hat{\gamma}_1 \boldsymbol{x}^T \hat{\boldsymbol{\beta}}$.

As we can see from the procedure, the refit method only adds small additional computational cost to the original PLR. The refit step is only a one-dimensional fit without penalty and can be done quickly. Furthermore, we suggest to refit the same model without regularization on the one dimensional projection space. However, one can use a different method here. For instance, in Step 1, one uses a binary classifier that can provide class conditional probability estimation, then the same classifier is also applied in Step 3, and the final probability estimation in Step 4 should be modified according to Theorem 2.2.1.

The new parameters in the refit step serve as a correction on the scale bias created by shrinkage on the first step. As we will show in Section 2.3.2 and in simulation, the refit method gives almost identical classification errors as the original method, and at the same time it offers remarkable improvement on probability estimation.

For weighted learning, the refit steps are almost the same except some slight modifications needed for Steps 3 and 4. Once the projection in Step 2 is done, we refit the one-dimensional model with weighted learning using the original w_+ and w_- as the weights, and obtain the corresponding probability estimator in Step 4. Using PLR as an example, the probability formula in Step 4 for weighted learning becomes $\hat{p}(\boldsymbol{x}) = \frac{w_- e^{\hat{\hat{f}}(\boldsymbol{x})}}{w_- e^{\hat{\hat{f}}(\boldsymbol{x})}+w_+}$.

2.3.2 Theoretical properties

In this section, we derive some asymptotic results for our refit method. In particular, we prove that under certain conditions, the refit procedure provides consistent probability estimation when the regularized method produces shrinkage on parameter estimation asymptotically. For simplicity, we first focus on linear learning with equal weights and then discuss the results for weighted learning.

First we introduce some assumptions.

Assumption C.1. The loss function $\ell(\cdot)$ is convex and differentiable.

Assumption C.2. The distribution $\mathbb{P}(\boldsymbol{x}, y)$ satisfies

$$P(Y = +1|\boldsymbol{X} = \boldsymbol{x}) = \frac{\ell'(-\boldsymbol{x}^T\boldsymbol{\beta}^*)}{\ell'(-\boldsymbol{x}^T\boldsymbol{\beta}^*) + \ell'(\boldsymbol{x}^T\boldsymbol{\beta}^*)} = p_0(\boldsymbol{x}^T\boldsymbol{\beta}^*),$$

where $\boldsymbol{\beta}^*$ is the global minimizer of $E[\ell(Y\boldsymbol{X}^T\boldsymbol{\beta})]$ that does not depend on $\boldsymbol{X}$.

Assumption C.3. The estimated $\hat{\boldsymbol{\beta}} = \operatorname{argmin}_{\boldsymbol{\beta}}[\frac{1}{n}\sum_{i=1}^n \ell(y_i\boldsymbol{x}_i^T\boldsymbol{\beta}) + \lambda J(\boldsymbol{\beta})]$ satisfies $\hat{\boldsymbol{\beta}} \to \theta\boldsymbol{\beta}^*$ in probability as $n \to \infty$, where $\theta \in (0, 1)$.

Next we discuss the use of these assumptions. For Assumption **C.1**, the convexity and differentiability are satisfied by many loss functions. Assumption **C.2** ensures that the class conditional probability $P(Y = +1|\boldsymbol{X} = \boldsymbol{x})$ depends on $\boldsymbol{x}$ only through the link function $\frac{\ell'(-\boldsymbol{x}^T\boldsymbol{\beta}^*)}{\ell'(-\boldsymbol{x}^T\boldsymbol{\beta}^*)+\ell'(\boldsymbol{x}^T\boldsymbol{\beta}^*)}$. For example, a similar assumption is used in logistic regression. Assumption **C.3** deals with the asymptotic behavior of $\hat{\boldsymbol{\beta}}$. For many large margin classifiers, the direction of $\hat{\boldsymbol{\beta}}$ is usually close to that of $\boldsymbol{\beta}^*$, yielding a good classification boundary. However, as discussed in Section 2.2.2, the regularization term creates bias in the estimation of $\boldsymbol{\beta}^*$.

The next theorem justifies that the refit procedure helps to correct the scale bias introduced by the regularization term, while keeping the classification boundary almost the same.

Theorem 2.3.1. *For linear learning, suppose that Assumptions* **A.1** *and* **C.1**-**C.3** *are satisfied. Then* $\hat{\gamma}_0 \to 0$ *and* $\hat{\gamma}_1 \to \frac{1}{\theta}$ *in probability, as* $n \to \infty$.

From Theorem 2.3.1, we can see that the refit method asymptotically corrects the scale from shrinkage, thus provides consistent probability estimation.

For weighted learning, we have a similar result. In that case, we modify Assumptions C.2 and C.3 as follows, and an immediate corollary follows from Theorem 2.3.1.

Assumption C.2'. The distribution $\mathbb{P}(\boldsymbol{x}, y)$ satisfies that

$$P(Y = +1|\boldsymbol{X} = \boldsymbol{x}) = \frac{w_-\ell'(-\boldsymbol{x}^T\boldsymbol{\beta}^*)}{w_-\ell'(-\boldsymbol{x}^T\boldsymbol{\beta}^*) + w_+\ell'(\boldsymbol{x}^T\boldsymbol{\beta}^*)} = p_0(\boldsymbol{x}^T\boldsymbol{\beta}^*),$$

where $\boldsymbol{\beta}^*$ is the global minimizer of $E[W_Y\ell(Y\boldsymbol{X}^T\boldsymbol{\beta})]$ that does not depend on $\boldsymbol{X}$. Here $W_Y = w_+$ if $Y = 1$ and w_- otherwise.

Assumption C.3'. The estimated $\hat{\boldsymbol{\beta}} = \operatorname{argmin}_{\boldsymbol{\beta}}[\frac{1}{n}\sum_{i=1}^n W(y_i)\ell(y_i\boldsymbol{x}_i^T\boldsymbol{\beta}) + \lambda J(\boldsymbol{\beta})]$ satisfies $\hat{\boldsymbol{\beta}} \to \theta\boldsymbol{\beta}^*$

in probability as $n \to \infty$, where $\theta \in (0,1)$.

Corollary 2.3.1. *For linear learning, suppose that Assumptions* **A.1**, **C.1**, **C.2'** *and* **C.3'** *are satisfied. Then $\hat{\gamma}_0 \to 0$ and $\hat{\gamma}_1 \to \frac{1}{\theta}$ in probability, as $n \to \infty$.*

2.4 Simulation

In this section, we use two simulated examples to illustrate the performance of PLR and PSVM. We compare the original and the refit methods for probability estimation. In both examples, we set the dimensions of covariates to be 5, 50, 100, 250, and 500.

Example 1: In this example, we generate the data as follows. For the positive class, the 1st and 2nd coordinates follow $N[(2,0)^T, I_2]$. For the negative class, the corresponding distribution is $N[(0,2)^T, I_2]$. The remaining $p-2$ covariates follow *i.i.d.* $N(0,1)$, where p is the dimensionality of $\boldsymbol{x}$.

The training data have 100 observations. The tuning parameter λ is chosen using a grid search method. Specifically, for each candidate λ value in $\{2^{-10}, 2^{-9}, \cdots, 2^{39}, 2^{40}\}$, we fit the model with PLR, and the misclassification error rate on a tuning data, whose observations are independent and identically distributed with respect to the training observations, is calculated. The tuning data have 100 observations. We choose the λ value that corresponds to the minimal error rate. Next a test set of size 10^4 is used to evaluate the performance of both classification accuracy and probability estimation. We repeat the whole procedure 1000 times to calculate the average misclassification rate, $P(Y \neq \hat{Y})$, and probability estimation error, $\frac{1}{\#(\text{test set})} \sum_{i \in \text{test set}} (|\hat{p}_i - p_i^{true}|)$. We report average probability estimation errors and average misclassification rates in Table 2.1. The corresponding standard errors are reported in parentheses. As a comparison, the regular logistic regression is also used here.

We can see from Table 2.1 that the absolute difference between p_{true} and $\hat{p}$ for the refit method is much smaller than the original estimator. This demonstrates the effectiveness of our refit method. In terms of classification errors, the refit method is almost identical to the original fit. As an illustration, we plot the classification boundaries before and after the refit step, along with the Bayes boundary, on the left panel of Figure 2.1, for one typical simulation. We can see that the refit step does not change the boundary much.

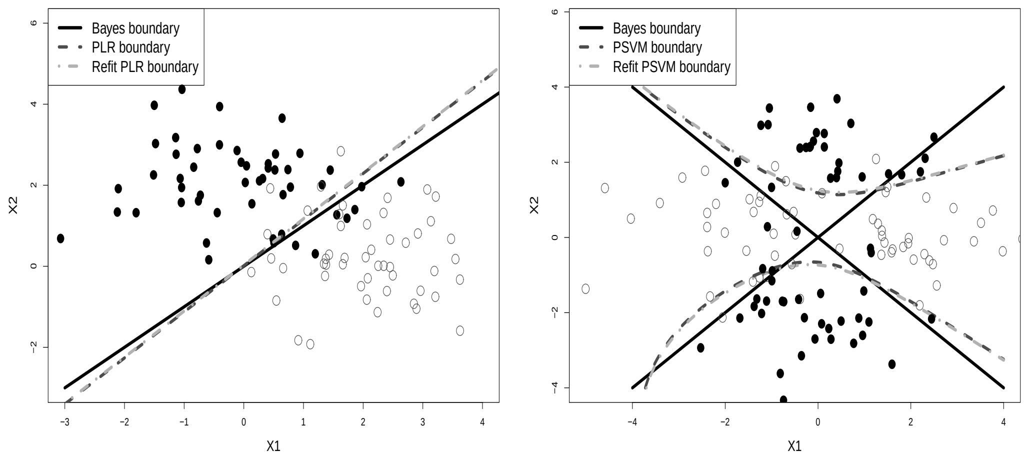
Example 2: This is a nonlinear example. Similar to Example 1, only the first two covariates are relevant for classification. For the positive class, the data points are generated as a mixture of normal distributions as $\frac{1}{2}N[(2,0)^T, I_2] + \frac{1}{2}N[(-2,0)^T, I_2]$, and the negative class is from a different mixture of normal distributions as $\frac{1}{2}N[(0,2)^T, I_2] + \frac{1}{2}N[(0,-2)^T, I_2]$. We use the PSVM as the loss function. For efficiency in computation, we add the second and third order of the 1^{st} and 2^{nd} coordinates into the original data, and then treat it as a linear learning. The training sample size is set to be 120. We generate 120 tuning observations in a similar manner as in Example 1. The tuning parameter λ is chosen in the same way as in the previous example, and the number of replications is also 1000. We report the probability estimation errors and the misclassification rates in Table 2.2. The Bayes boundary, the classification boundaries before and after the refit step are reported on the right panel of Figure 2.1. In this example, a similar conclusion as in Example 1 can be drawn from the results in Table 2.2 and the right panel of Figure 2.1.

	Dimension	5	50	100	250	500
mean($\|p^{true} - \hat{p}\|$)	Original PLR	0.2302 (0.00077)	0.2698 (0.00057)	0.3112 (0.00023)	0.4092 (0.00019)	0.4236 (0.00076)
	Refit PLR	0.0452 (0.00132)	0.1026 (0.00188)	0.1479 (0.00255)	0.1717 (0.00198)	0.2064 (0.00219)
	LR	0.0912 (0.00257)	NA	NA	NA	NA
Misclassication error	Original PLR	0.0847 (0.00067)	0.1259 (0.00126)	0.1621 (0.00190)	0.2321 (0.00200)	0.2874 (0.00171)
	Refit PLR	0.0851 (0.00059)	0.1258 (0.00152)	0.1603 (0.00237)	0.2325 (0.00290)	0.2895 (0.00177)
	LR	0.1089 (0.00223)	NA	NA	NA	NA

Table 2.1: The average classification and probability estimation errors using PLR for Example 1 with $n = 100$. The corresponding standard errors are reported in parentheses. The LR cannot be calculated for $p \geq 50$, and we use NA for those entries.

	Dimension	5	50	100	250	500
mean($\|p^{true} - \hat{p}\|$)	Original PSVM	0.1892 (0.00030)	0.2011 (0.00019)	0.2473 (0.00024)	0.3019 (0.00054)	0.3698 (0.00033)
	Refit PSVM	0.0801 (0.00377)	0.1195 (0.00560)	0.1278 (0.00573)	0.1503 (0.00399)	0.1610 (0.00298)
Misclassication error	Original PSVM	0.1621 (0.00199)	0.1939 (0.00356)	0.1944 (0.00511)	0.1957 (0.00299)	0.2016 (0.00318)
	Refit PSVM	0.1622 (0.00342)	0.1940 (0.00406)	0.1945 (0.00531)	0.1952 (0.00290)	0.2012 (0.00382)

Table 2.2: The average classification and probability estimation errors using PSVM for Example 2 with $n = 120$. The corresponding standard errors are reported in parentheses.



(a) Example 1, classification boundaries (b) Example 2, classification boundaries

Figure 2.1: The left panel shows the classification boundaries of the original and the refit methods for PLR in Example 1. The right panel shows the classification boundaries of the original and the refit methods for PSVM in Example 2. Clearly the classification boundaries of the original and the refit methods are almost identical.

2.5 Real data

In this section we investigate the performance of our proposed refit method on two real data sets, namely Liver and Ionosphere. The Liver dataset contains 345 patients with the class label being the liver disorder status. There are 6 input variables related to blood tests and they are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. For the Ionosphere data, the goal is to clarify good versus bad radar returns using 34 input attributes. Good radar returns are those showing evidence of certain structure in the ionosphere, and bad returns are those that do not. There are 351 samples in total. More information about these datasets can be found on the UCI machine learning repository database website, `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

In each example we randomly choose 70 observations for training, 75 for tuning, and the remaining for testing. In each random split, λ is chosen by a grid search as in the simulated examples, and we compare the probability estimation of the original methods with that of the refit methods. We apply PLR for both datasets. Since the underlying probability distribution is unknown, we evaluate the closeness of $\hat{p}$ to p in terms of the Cross Entropy Error (CRE, Wang, Shen and Liu [2008]), where

$\text{CRE}(\hat{p}) = -\frac{1}{\#(\text{test set})}\sum_{\text{test set}}\left\{\frac{1}{2}(1+y_i)\log\left[\hat{p}(x_i)\right] + \frac{1}{2}(1-y_i)\log\left[1-\hat{p}(x_i)\right]\right\}$. For each data set, we standardize the input covariates before the analysis. For the Liver data, the Gaussian kernel is applied, and the kernel parameter σ is set to be the first quartile pairwise distance between the positive and negative classes. Through 1000 times of random splitting, the CRE's of the original estimator and of the refit method are 1.959(0.001) and 1.908(0.004), respectively. For the Ionosphere data, linear learning is used. The CRE's of the original estimator and of the refit method are 3.211(0.062) and 0.525(0.006), respectively. This suggests that the refit method improves probability estimation for both real data examples. Interestingly, the improvement for the Ionosphere data appears to be larger than the Liver data, possibly due to its higher dimension of covariates.

2.6 Possible Future Work

In this chapter we investigate the problem of probability estimation for large margin classifiers and illustrate the bias problem on class probability estimation created by shrinkage. We show that such bias can be large for finite sample problems. As a result, alternative procedures are needed. Our simple refit method helps to correct the scale problem introduced by shrinkage and yields accurate class probability estimation.

As a remark, we would like to mention that the work of Zhu and Hastie [2003] provides a promising path for further improvement of probability estimation. In particular, they proposed an interesting idea of feature selection through density estimation. For our case, suitable feature selection via density estimation may help to yield a more flexible and robust projection space for the refit step.

Our focus in this chapter is on binary classification. For multicategory problems, we believe similar phenomena exist and corrections are necessary as well. Since there will be multiple classification functions, the projection step is more complicated. Further investigation will be pursued.

2.7 Proofs

Proof of Theorem 2.2.1. Notice that $\min E\{\ell[Yf(\boldsymbol{X})]\} = \min E\{E\big[\ell[Yf(\boldsymbol{X})]|\boldsymbol{X}=\boldsymbol{x}\big]\}$. Letting $S = E\{\ell[Yf(\boldsymbol{X})]|\boldsymbol{X}=\boldsymbol{x}\} = p(\boldsymbol{x})\ell[f(\boldsymbol{x})]+[1-p(\boldsymbol{x})]\ell[-f(\boldsymbol{x})]$, we have $\frac{\partial S}{\partial f}|_{f=f^*} = \ell'(f^*)p-\ell'(-f^*)(1-p) = 0$. As $\ell'(u)+\ell'(-u)<0, \forall u$, $p(\boldsymbol{x}) = \frac{\ell'[-f^*(\boldsymbol{x})]}{\ell'[f^*(\boldsymbol{x})]+\ell'[-f^*(\boldsymbol{x})]}$. Now taking the derivative of $p(\boldsymbol{x})$ with respect to $f^*(\boldsymbol{x})$ yields $\frac{dp}{df^*} = -\frac{\ell'(f^*)\ell''(-f^*)+\ell'(-f^*)\ell''(f^*)}{[\ell'(f^*)+\ell'(-f^*)]^2}$. Thus by Condition **B.3**, $p(\boldsymbol{x})$ is a strictly increasing function of $f^*(\boldsymbol{x})$, which guarantees a one-to-one correspondence between them. □

Proof of Theorem 2.3.1. Let $x_i^* = \boldsymbol{x}_i^T \boldsymbol{\beta}^*$. Recall that $\hat{x}_i^* = \boldsymbol{x}_i^T \hat{\boldsymbol{\beta}}$. The empirical loss function we minimize in the refit step is

$$\min_{\gamma_0,\gamma_1} \frac{1}{n} \sum_{i=1}^{n} \ell[y_i(\hat{x}_i^* \gamma_1 + \gamma_0)],$$

and we define

$$L(\hat{\gamma}_0, \hat{\gamma}_1) := \frac{1}{n} \sum_{i=1}^{n} \ell[y_i(\hat{x}_i^* \hat{\gamma}_1 + \hat{\gamma}_0)].$$

Assume that $\hat{\gamma}_0$ does not converge to 0 in probability, as $n \to \infty$. We then have a subsequence of $\hat{\gamma}_0$'s that converges to another real number $z \neq 0$ in probability. For simplicity, assume that the entire sequence $\hat{\gamma}_0$ converges to z. Note that as $n \to \infty$, $L(0, \frac{1}{\theta})$ converges to $E[\ell(yx^*)]$ by Assumption **C.3**. Because $L(\hat{\gamma}_0, \hat{\gamma}_1)$ does not converge to $E[\ell(yx^*)]$ for any choice of $\hat{\gamma}_1$ if $\hat{\gamma}_0 \to z$, we can conclude that for large enough n, $L(\hat{\gamma}_0, \hat{\gamma}_1) > L(0, \frac{1}{\theta})$, by Assumption **C.2**. This contradicts the fact that $(\hat{\gamma}_0, \hat{\gamma}_1)$ is the minimizer of L. Thus $\hat{\gamma}_0$ converges to 0 in probability, as $n \to \infty$. A similar argument can show that $\hat{\gamma}_1$ converges to $\frac{1}{\theta}$ in probability, as $n \to \infty$. This completes the proof. □

Proof of Corollary 2.3.1. In weighted learning, the empirical loss function we minimize in the refit step is

$$\min_{\gamma_0,\gamma_1} \frac{1}{n} \sum_{i=1}^{n} W(y_i) \ell[y_i(\hat{x}_i^* \gamma_1 + \gamma_0)],$$

and now the definition of L becomes

$$L(\hat{\gamma}_0, \hat{\gamma}_1) := \frac{1}{n} \sum_{i=1}^{n} W(y_i) \ell[y_i(\hat{x}_i^* \hat{\gamma}_1 + \hat{\gamma}_0)].$$

The rest of the proof follows the same line as that of Theorem 2.3.1, except that $L(0, \frac{1}{\theta})$ converges to $E[W_y \ell(yx^*)]$ instead of $E[\ell(yx^*)]$. □

CHAPTER 3: MULTICATEGORY LUM

3.1 Introduction

Classification problems are commonly seen in practice. When one faces a classification task, there are many possible techniques to choose from. To list a few, logistic regression and Fisher linear discriminant analysis (LDA) are very classical classification methods. The Support Vector Machine [SVM, Cortes and Vapnik, 1995; Wahba, 1999] and Boosting [Freund and Schapire, 1997] are more recent machine learning based large margin classification tools. Despite some known properties of these methods, a practitioner often needs to face one natural question: which method should one choose to solve the classification problem in hand?

Wahba [2002] discussed the concept of soft versus hard classification. Recall the difference between soft and hard classifiers from Chapter 1. Typical examples of soft classifiers include logistic regression and LDA. On the other hand, the SVM is a typical example of hard classifiers, which is without any strong distributional assumption. Another example of hard classifiers is the $\psi-$learning [Shen, Tseng and Zhang, 2003]. When class probability estimation is necessary, one can perform multiple weighted learning for probability estimation [Wang, Shen and Liu, 2008]. For a given problem, the choice between hard and soft classifiers can be difficult. Recently, Liu, Zhang and Wu [2011] proposed the LUM family, which is a rich group of classifiers in the sense that it connects hard and soft classifiers in one spectrum. The LUM family provides a natural platform for comparisons between soft and hard classifiers. More importantly, it enables us to observe the performance transition from soft to hard classification.

The existing development on the LUM is limited to the binary case. For multicategory problems, further development is necessary. In particular, probability estimation becomes more challenging as one needs to estimate a probability vector. Furthermore, multicategory consistency is much more involved, especially for hard classifiers. For instance, there are a lot of developments on multicategory SVMs in the literature [Vapnik, 1998; Weston and Watkins, 1999; Crammer et al., 2001; Lee, Lin and Wahba, 2004; Wang and Shen, 2007; Liu and Yuan, 2011] and many of them are not consistent when there is no dominating class, i.e, the maximum class probability is less than 0.5. So far, the only group of consistent multicategory piecewise linear hinge loss functions available is the family of reinforced hinge loss functions by Liu and Yuan [2011], which covers the loss by Lee, Lin and Wahba [2004] as a special case. For probability estimation, there are several existing multicategory soft classifiers, such as Adaboost in Boosting [Freund and Schapire, 1997; Zou, Zhu and Hastie, 2008*b*; Zhu et al., 2009], logistic regression [Lin et al., 2000], proximal SVMs [Tang and Zhang, 2006], and multicategory composite least squares classifiers [Park et al., 2010].

In this chapter, we propose new Multicategory Large-margin Unified Machines (MLUMs). Similar to the binary case, the MLUM is a broad family that embraces many of the aforementioned classifiers as special cases. It helps to shed some light on the choice between multicategory soft and hard classifiers, and provides some insights on the behavior change from soft to hard classification methods. Our theoretical studies show that the MLUM is always Fisher consistent, and is able to provide class conditional probability estimation. Moreover, we extend the excess risk concept discussed in Bartlett, Jordan and McAuliffe [2006] to the multicategory case and study its convergence rate. We also propose an efficient tuning procedure for the MLUM family. Our numerical results show that the behaviors of different classifiers vary from setting to setting. In particular, we have the following observations.

- Soft classifiers tend to give more accurate classification results by estimating the conditional class probability when the true probability functions are relatively smooth.
- Hard classifiers bypass the probability estimation and may work better when estimation of the underlying probability functions is challenging, such as the step function.
- When the data are noisy with outliers, soft classifiers tend to be very sensitive and unstable. Some MLUM member in-between hard and soft tends to work the best. This was not observed in the binary case [Liu, Zhang and Wu, 2011].

Although our observations may not hold for all classification problems, it can help us to understand the classification behaviors better. Furthermore, our numerical results also suggest that the performance of the proposed tuned MLUM is very competitive.

The rest of this chapter is organized as follows. In Section 3.2, we give some motivation and introduce the MLUM family. Section 3.3 explores some statistical properties of the MLUM family. Section 3.4 addresses the computational aspect of the MLUM. In Section 3.5, we demonstrate the numerical performance of MLUM via several simulated examples. Section 3.6 discusses some benchmark examples and one gene expression dataset. Some discussion is provided in Section 3.7. The technical proofs are collected in Section 3.8.

3.2 Methodology

In this section, we first briefly review the background of binary classification, and the different ways of generalization to multicategory problems. The notion of soft and hard classification is first reviewed in the binary classification context. Then we propose a MLUM framework which helps us to understand soft versus hard classification in the multicategory setting.

3.2.1 Background on Binary Classification

With a training dataset given, one main goal of classification is to build a classifier for us to predict the class label y using the input vector $\boldsymbol{x}$. Here we assume that the training data are *i.i.d.* samples from an unknown underlying distribution $\mathbb{P}(\boldsymbol{x}, y)$. Using the functional margin introduced above, the theoretical $0-1$ loss can be directly written as $L(yf(\boldsymbol{x})) = I(yf(\boldsymbol{x}) \leq 0)$. Our goal is to find a classification function f such that the expected loss of f, denoted by

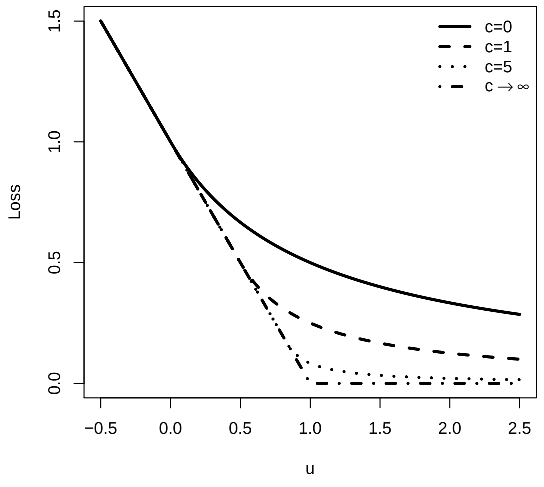
$$R(f(\cdot)) = E_{\boldsymbol{X},Y} L(Yf(\boldsymbol{X})), \tag{3.1}$$

is as small as possible. The infimum of $R(f(\cdot))$, denoted by R^*, is called the Bayes error. Because the optimization problem (3.1) is NP-hard, people propose different surrogate loss functions which lead to different classification techniques. These methods can be roughly grouped into two categories, namely, soft and hard classifiers. In practice, it is unclear which one to use for a particular problem. To answer this question, Liu, Zhang and Wu [2011] proposed to use the LUM loss function $\ell(\cdot)$, where

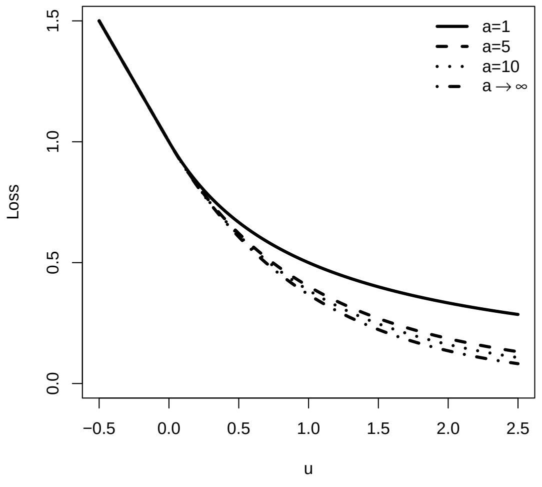
$$\ell(u) = \begin{cases} 1-u & \text{if } u < \frac{c}{1+c} \\ \frac{1}{1+c}\left(\frac{a}{(1+c)u-c+a}\right)^a & \text{if } u \geq \frac{c}{1+c}, \end{cases} \tag{3.2}$$

with $c \geq 0$ and $a > 0$ being parameters of the LUM family. See Figure 3.1 for the shape of $\ell(u)$ with a few values of a and c [Liu, Zhang and Wu, 2011]. Note that the LUM family includes the SVM hinge loss with $c \to \infty$, and the Distance Weighted Discrimination [DWD, Marron, Todd and

Ahn, 2007] with $c = 1$ and $a = 1$, as special cases. The parameter c is an index of soft versus hard classifiers. In particular, $c = 0$ corresponds to a typical soft classifier, and $c \to \infty$ corresponds to the SVM, a typical hard classifier. Consequently, the LUM connects soft and hard classifiers as a family, and enables one to deeply investigate the spectrum of the transition behavior.



(a) a=1



(b) c=0

Figure 3.1: Plots of several LUM loss functions. On the left panel, we have $a = 1$ and $c = 0, 1, 5, \infty$, and on the right panel we have $c = 0$ and $a = 1, 5, 10, \infty$.

So far, our focus has been on binary methods. In the next section, we briefly review some existing methods for multicategory classification problems.

3.2.2 Existing Multicategory Classification Methods

To solve a multicategory problem, a natural and direct way is to implement multiple binary classifiers. As discussed in Chapter 1, these methods have their own drawbacks. Hence, it is desirable to have a simultaneous multicategory classifier that considers k classes altogether.

With simultaneous multicategory classifiers, one solves the following optimization problem

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} V(\boldsymbol{f}(\boldsymbol{x}_i), y_i) + \lambda J(\boldsymbol{f}), \tag{3.3}$$

with V being a loss function for a multicategory problem, and $J(\boldsymbol{f})$ is a regularization term defined for the multicategory problems. Note that because of the argmax rule, adding a constant to all the elements of $\boldsymbol{f}(\boldsymbol{x})$ does not change the prediction rule. To ensure identifiability of our solution and to reduce the dimension of the problem, a sum-to-zero constraint, $\sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0$, is commonly used. As a result, this formulation is equivalent to the binary problem with $k = 2$. With the argmax prediction rule, a sensible loss function V should encourage f_y to be the maximum among $\{f_j; j = 1, \ldots, k\}$.

For soft classification in multiclass problems, Zhu and Hastie [2005] used the generalized logistic loss $V = -f_y(\boldsymbol{x}) + \log(e^{f_1(\boldsymbol{x})} + \cdots + e^{f_k(\boldsymbol{x})})$. Tang and Zhang [2006] employed the square loss $V = (\boldsymbol{z} - \boldsymbol{f})^T(\boldsymbol{z} - \boldsymbol{f})$, where $\boldsymbol{z} = -\frac{1}{k-1}(1, 1, \ldots, 1)^T + \frac{k}{k-1}\boldsymbol{e}_y$, and $\boldsymbol{e}_j$ is the vector with 1 at the j^{th} element and 0 elsewhere. Zhu et al. [2009] extended the Adaboost to a multicategory learning method with the exponential loss $V = \exp(-\frac{1}{k}\boldsymbol{z}^T\boldsymbol{f})$. In the literature of hard classifiers, there are several ways to extend the binary hinge loss of the SVM to the simultaneous multicategory case. Here we list several commonly used versions with the sum-to-zero constraint:

Loss 1 (Naive hinge loss) $[1 - f_y(\boldsymbol{x})]_+$;

Loss 2 [Vapnik, 1998] $\sum_{j \neq y}[1 - (f_y(\boldsymbol{x}) - f_j(\boldsymbol{x}))]_+$;

Loss 3 [Crammer et al., 2001; Liu, Shen and Doss, 2005] $\sum_{j \neq y}[1 - \min_j(f_y(\boldsymbol{x}) - f_j(\boldsymbol{x}))]_+$;

Loss 4 [Lee, Lin and Wahba, 2004] $\sum_{j \neq y}[1 + f_j(\boldsymbol{x})]_+$.

Losses 1, 2 and 3 are known to be inconsistent [Lee, Lin and Wahba, 2004; Liu, 2007; Tewari and Bartlett, 2007]. In contrast, Loss 4 is Fisher consistent. Recently, Liu and Yuan [2011] proposed the Reinforced Multicategory Support Vector Machine (RMSVM), which employs a convex combination of the naive hinge loss and the Loss 4 by Lee, Lin and Wahba [2004] as follows,

$$V(\boldsymbol{f}(\boldsymbol{x}), y) = \gamma[(k-1) - f_y(\boldsymbol{x})]_+ + (1-\gamma)\sum_{j \neq y}[1 + f_j(\boldsymbol{x})]_+, \tag{3.4}$$

subject to $\sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0$. Interestingly, the reinforced hinge loss function is Fisher consistent when $0 \leq \gamma \leq 1/2$, and includes the Loss 4 in Lee, Lin and Wahba [2004] as a special case with $\gamma = 0$. Liu and Yuan [2011] showed that the loss function (3.3) with $\gamma = 1/2$ yields the best overall classification performance, among $\gamma \in [0, 0.5]$. Inspired by the RMSVM loss formulation, we propose to extend the LUM family to the MLUM in an analogous way, as discussed in the next section.

Next we examine the sum-to-zero constraint $\sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0$ for different multicategory losses. In linear learning, we assume that $f_j(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta}_j + b_j; j = 1, \ldots, k$, and the L_2 penalty is $J(\boldsymbol{f}) = \frac{1}{2}\sum_{j=1}^{k} \|\boldsymbol{\beta}_j\|^2$. In kernel learning, we have $f_j = g_{j,\mathcal{H}} + b_j$ and $g_{j,\mathcal{H}}; j = 1, \ldots, k$ belong to some Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$, while b_j's are constants. The L_2 penalty for kernel learning is $J(\boldsymbol{f}) = \frac{1}{2}\sum_{j=1}^{k} \|g_{j,\mathcal{H}}\|_{\mathcal{H}}^2$, where $\|\cdot\|_{\mathcal{H}}$ is the norm in $\mathcal{H}$ introduced by its corresponding kernel (see, for example, Aronszajn [1950] and Wahba [1999] for more details about RKHS). Notice that in both cases, the intercepts $b_j; j = 1, \ldots, k$ are not penalized. Thus, if we add a constant to all $b_j; f = 1, \ldots, k$, the prediction on any new instance does not change. Thus, the sum-to-zero constraint helps to obtain unique solutions. The next proposition shows that, if the loss function depends on $\boldsymbol{f}$ only through its element-wise difference $f_i - f_j; i \neq j$ as in Losses 2 and 3, then without the sum-to-zero constrain, the solutions $\{\hat{\boldsymbol{\beta}}_j\}$ in linear learning or $\{\hat{g}_{j,\mathcal{H}}\}$ in kernel learning automatically sum to zero, under the L_2 penalty. Note that this phenomenon for the SVM was previously noted by Wu and Liu [2007].

Proposition 3.1: Suppose the loss function $V(\boldsymbol{f}, y)$ depends on $\boldsymbol{f}$ only through $f_i - f_j; i \neq j$, then the solution to (3.3), using the L_2 penalty without the sum-to-zero constraint, satisfies that $\sum_{j=1}^{k} \hat{\boldsymbol{\beta}}_j = \mathbf{0}$ for linear learning, and $\sum_{j=1}^{k} \hat{g}_{j,\mathcal{H}} = 0$ for RKHS learning.

Notice that the condition in Proposition 3.1 is satisfied by MSVM Losses 2 and 3 mentioned above. For other loss functions such as Losses 1 and 4, the result does not hold. However, for those loss functions, the sum-to-zero constraint is more essential for theoretical properties such as Fisher consistency. This constraint was also used in many other simultaneous multicategory classification papers, for example, Tang and Zhang [2006], Wang and Shen [2007], and Zhu et al. [2009].

3.2.3 MLUM Family

Soft and hard classifiers have been both studied in the literature of simultaneous multicategory classification. Which one to use in practice remains to be a challenging question. The LUM family is a broad family which embraces both soft and hard classifiers in binary cases, yet no such convenient platform is available in the multicategory framework. In this chapter, we propose a new family of MLUMs to study multicategory problems. In particular, we make use of the idea of reinforced multicategory hinge loss by Liu and Yuan [2011], and propose the following MLUM loss family,

$$V(\boldsymbol{f}, y) = \gamma \ell(f_y(\boldsymbol{x})) + (1-\gamma) \sum_{j \neq y} \ell(-f_j(\boldsymbol{x})), \tag{3.5}$$

under the constraint $\sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0$, where $\ell(u) : \mathbb{R} \to \mathbb{R}$ is the LUM loss function, and $\gamma \in [0, 1]$. When $k = 2$, the MLUM reduces to the binary LUM loss.

The main motivation to use the MLUM loss function (3.5) is based on the argmax rule for multicategory classification. For a given data point $(\boldsymbol{x}, y)$, in order to make the corresponding classification correct, we need to have the corresponding $f_y(\boldsymbol{x})$ to be the maximum among $\boldsymbol{f}(\boldsymbol{x})$. To that end, the first term in (3.5) encourages $f_y(\boldsymbol{x})$ to be large, and the second term encourages $f_j(\boldsymbol{x}), j \neq y$, to be small. Consequently, both terms in (3.5) try to make f_y big.

To further comprehend the MLUM family, we rewrite (3.5) using the multiple comparison vector representation proposed by Liu and Shen [2006]. In particular, we define the comparison vector $g(\boldsymbol{f}(\boldsymbol{x}), y) = \big(f_y(\boldsymbol{x}) - f_1(\boldsymbol{x}), \ldots, f_y(\boldsymbol{x}) - f_{y-1}(\boldsymbol{x}), f_y(\boldsymbol{x}) - f_{y+1}(\boldsymbol{x}), \ldots, f_y(\boldsymbol{x}) - f_k(\boldsymbol{x})\big)$. Then by the argmax classification rule, a data point $(\boldsymbol{x}, y)$ is misclassified if and only if $\min g(\boldsymbol{f}(\boldsymbol{x}), y) \leq 0$. Let $u = g(\boldsymbol{f}(\boldsymbol{x}), y)$, then the $0-1$ loss can be written as $I\{\min_j u_j \leq 0\}$, and the MLUM loss can be expressed as

$$V(\boldsymbol{f}, y) = \gamma \ell(\sum_{j=1}^{k-1} u_j / k) + (1-\gamma) \sum_{j=1}^{k-1} \ell(-\sum_{i=1}^{k-1} u_i / k + u_j). \tag{3.6}$$

Figure 3.2 shows the plot of (3.6) with $\gamma = 0, 0.5, 1$ and $c = 0, a = 1$ (soft classifier), and the $0-1$ loss with $k = 3$, for comparison. We can see that as γ changes, the shape of the MLUM loss functions varies a lot, although they are all convex upper envelopes of the $0-1$ loss. Moreover, as γ increases, the value of the loss function increases when u_1 and u_2 are both negative, and decreases when just one of them is negative.

In the next section, we explore some statistical properties of the MLUM family, which can help us understand the MLUM better with respect to the parameters involved in (3.5).

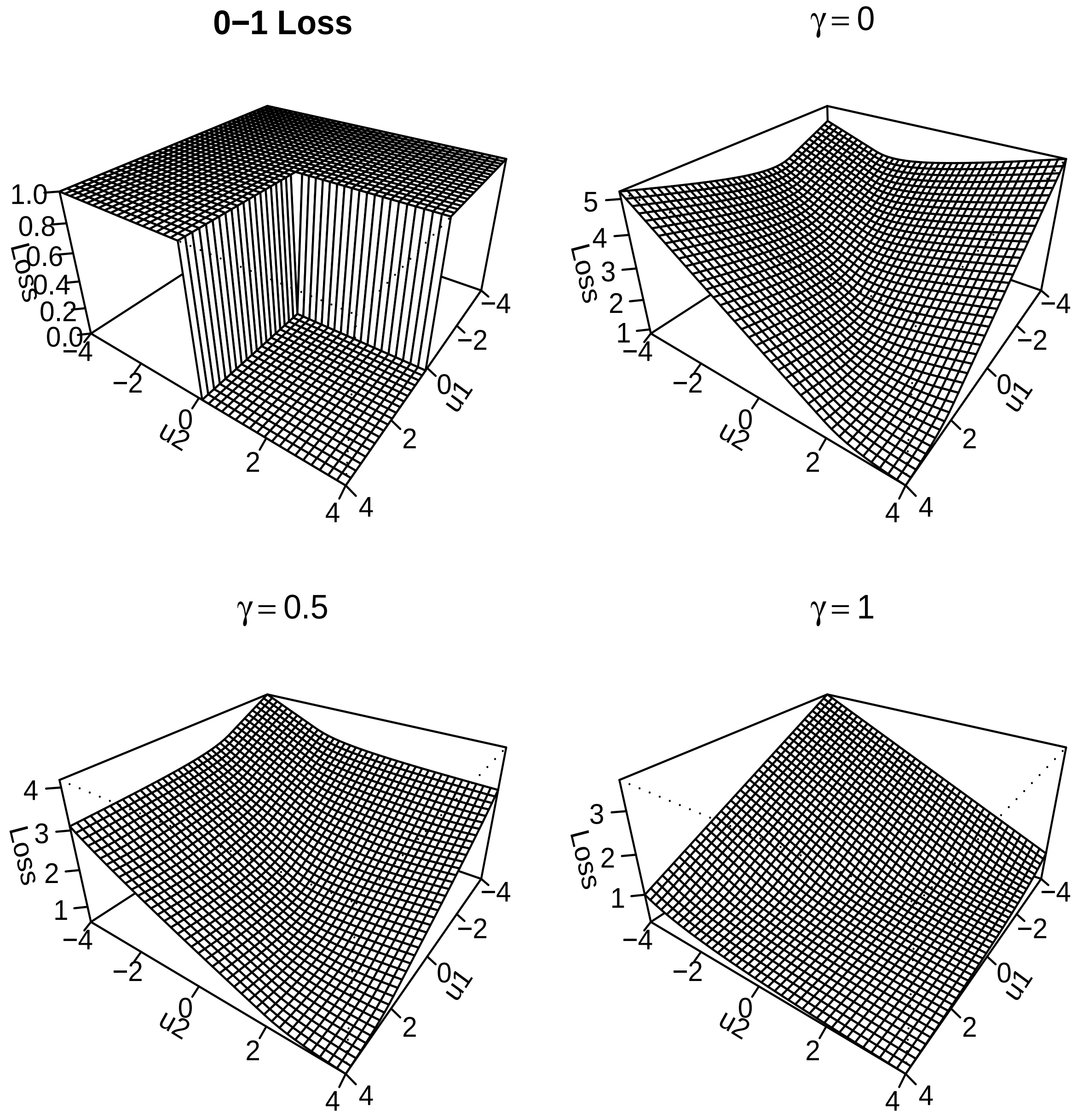


Figure 3.2: Plots of the $0-1$ loss function (top left panel) and MLUM loss functions with $\gamma = 0$ (top right), $\gamma = 0.5$ (bottom left), $\gamma = 1$ (bottom right), for $c = 0$, $a = 1$ and $k = 3$.

3.3 Statistical Properties

In this section, we study the statistical properties of the MLUM family. We first study its consistency in Section 3.3.1, then derive the formula for class probability estimation in Section 3.3.2. In Section 3.3.3, we extend the notion of the excess V-risk, as defined in Bartlett, Jordan and McAuliffe

[2006] and Zhang [2004*b*], from the binary case to the multicategory one. We show that the convergence rate of the excess V-risk depends on the size of the functional space, as well as the convergence rate of the estimated classification function to its theoretical minimizer within the functional space.

3.3.1 Fisher Consistency

As different loss functions yield different methods, it is essential to study the properties of these loss functions. One important concept is the Fisher consistency [Zhang, 2004*a*; Bartlett, Jordan and McAuliffe, 2006], defined as follows. For a binary classification problem with the corresponding classification function f, a loss function $V(\cdot)$ is Fisher consistent if and only if $\text{sign}(f^*(\boldsymbol{x})) = \text{sign}(p(\boldsymbol{x}) - \frac{1}{2})$, where $f^*(\boldsymbol{x}) = \text{arginf}_f\, E[V(Yf(\boldsymbol{X}))|\boldsymbol{X} = \boldsymbol{x}]$ and $p(\boldsymbol{x}) = p(Y = 1|\boldsymbol{X} = \boldsymbol{x})$. As we can see, Fisher consistency essentially ensures the corresponding decision boundary induced by f^* is identical to the Bayes boundary $\{\boldsymbol{x} : p(\boldsymbol{x}) = 1/2\}$.

In the multicategory classification literature, to tackle the Fisher consistency problem, we need the following definitions.

Definition 1 (Expected $V-$loss): Define the expected $V-$loss as

$$E_{\boldsymbol{X},Y} V(\boldsymbol{f}, y) = E_{\boldsymbol{X}}\big[\sum_{j=1}^{k} V(\boldsymbol{f}(\boldsymbol{X}), j) P_j(\boldsymbol{X}) | \boldsymbol{X} = \boldsymbol{x}\big],$$

where $\boldsymbol{P}(\boldsymbol{x}) = (P_1(\boldsymbol{x}), \dots, P_k(\boldsymbol{x}))$ is the class conditional probability.

Definition 2 (Conditional $V-$loss): For any $\boldsymbol{x}$, define the conditional $V-$loss as

$$S(\boldsymbol{f}, \boldsymbol{x}) = \sum_{j=1}^{k} V(\boldsymbol{f}(\boldsymbol{x}), j) P_j(\boldsymbol{x}). \tag{3.7}$$

Because of the argmax rule, Fisher consistency means that for any given $\boldsymbol{P}(\boldsymbol{x})$, the minimizer $\boldsymbol{f}^*(\boldsymbol{x}) = (f_1^*(\boldsymbol{x}), \dots, f_k^*(\boldsymbol{x}))$ of $S(\boldsymbol{f}, \boldsymbol{x})$ is such that $\text{argmax}_j\, P_j(\boldsymbol{x}) = \text{argmax}_j\, f_j^*(\boldsymbol{x})$. Furthermore, if $\text{argmax}_j\, P_j(\boldsymbol{x})$ is unique, then so is $\text{argmax}_j\, f_j^*(\boldsymbol{x})$. Next we show that the MLUM loss function is always Fisher consistent with a finite c and for any $\gamma \in [0, 1]$, $a > 0$. To that end, we need to show that, if P_1 is the unique maximum among $\{P_1, \dots, P_k\}$, then $f_1^* > \max\{f_2^*, \dots, f_k^*\}$. The next lemma assures that in the MLUM family, f_1^* is the maximum among $\{f_1^*, \dots, f_k^*\}$ (not necessarily unique), even with $c = \infty$.

Lemma 3.3.1. *In the MLUM family with $c \in [0, \infty]$, $a > 0$ and $\gamma \in [0, 1]$, suppose for $i, j \in \{1, \ldots, k\}$, we have $P_i > P_j$, then $f_i^* \geq f_j^*$.*

Clearly the above lemma is not sufficient for Fisher consistency, because the uniqueness of f_1^* is not guaranteed. Liu and Yuan [2011] showed that if we replace $\ell(\cdot)$ in (3.5) with the hinge loss with some minor modifications as in (3.4), the Fisher consistency will fail when $\gamma > 1/2$. The deficiency of the hinge loss is that it is not differentiable at the point 1, which then assures $f_j^* \leq 1; j = 1, \ldots, k$. Because the LUM loss ℓ is always differentiable with finite c, Fisher consistency of the MLUM is guaranteed, as in the following theorem.

Theorem 3.3.1. *The MLUM loss function (3.5) with any $a > 0$, $\gamma \in [0, 1]$ and $c \in [0, \infty)$, is Fisher consistent.*

As a remark, we note that the proof of the preceding theorem can shed some light on why the RMSVM is not Fisher consistent when $\gamma > 1/2$. Since the hinge loss is not differentiable at 1, the maximal possible value of $f_j^*; j = 1, \ldots, k$ in the RMSVM is 1. When $P_1 > P_2 \geq P_3 \cdots \geq P_k$, we may have $f_1^* = f_2^* = 1$. Thus the loss can be inconsistent. See Remark 3.3 in the appendix for more discussions.

3.3.2 Probability Estimation

Class conditional probability estimation is very important in many applications. It is common to use the relationship between $\boldsymbol{f}^*$ and the class probability $\boldsymbol{P}$ for estimation of the latter using $\hat{\boldsymbol{f}}^n$, where $\hat{\boldsymbol{f}}^n$ is the empirical solution to (3.3) with V being the proposed MLUM loss. In this section we convert the minimizer $\boldsymbol{f}^*$ into the class conditional probability estimation. When an estimated $\hat{\boldsymbol{f}}^n$ is obtained, one can use the formula to obtain the estimated probability $\hat{\boldsymbol{P}}$. The following theorem gives the probability estimation formula for the MLUM family with any finite c.

Theorem 3.3.2. *Let $\hat{E}(j) = [\gamma\ell'(\hat{f}_j) + (1-\gamma)\ell'(-\hat{f}_j)]$ and $\hat{F}(j) = (1-\gamma)\ell'(-\hat{f}_j); j = 1, \ldots, k$, where $\hat{f}_j$ is the j^{th} element of $\hat{\boldsymbol{f}}^n$. Then the probability estimation of $\hat{P}_j$ for the MLUM can be expressed as*

$$\hat{P}_j = \frac{1}{\hat{E}(j)}\{\hat{F}(j) + \frac{1}{\sum_{i=1}^k \frac{1}{\hat{E}(i)}}[1 - \sum_{i=1}^k \frac{\hat{F}(i)}{\hat{E}(i)}]\}; j = 1, \ldots, k. \tag{3.8}$$

Note that the class probability estimation requires that $\hat{P}_j \in [0,1]$ for $j = 1, \dots, k$ and $\sum_{j=1}^k \hat{P}_j = 1$. One can check that the requirement of the estimated probabilities summing to one is satisfied in (3.8). For $\gamma = 1$, $F(j) = 0$, and one can directly verify the estimated $\hat{P}_i$ using (3.8) is proper. However, with $\gamma \neq 1$, $\hat{P}_j$ in (3.8) may be outside of $[0,1]$. To ensure a proper estimation in the sense of each $0 \leq \hat{P}_j \leq 1$ and $\sum_{j=1}^k \hat{P}_j = 1$, we apply a scaled probability estimates using the following formula

$$\hat{P}_j^{scaled} = \frac{\hat{P}_j - \min_{i=1,\dots,k} \hat{P}_i}{\sum_{m=1}^k (\hat{P}_m - \min_{i=1,\dots,k} \hat{P}_i)}.$$

A similar strategy was previously used in Park et al. [2010]. Note that there may be other scaling strategies.

In the binary case, the LUM provides class conditional probability estimation with any finite c. In particular, with the classification function f, $p(\boldsymbol{x})$ does not have a one-to-one relationship with $f^*(\boldsymbol{x})$ when $p(\boldsymbol{x}) = 1/2$ and $c > 0$. In that case, all values of $f^*(\boldsymbol{x}) \in [-\frac{c}{1+c}, \frac{c}{1+c}]$ correspond to $p(\boldsymbol{x}) = 1/2$. Figure 3.3 displays the relationship between $p(\boldsymbol{x})$ and $f^*(\boldsymbol{x})$ with $a = 1$ and $c \in \{0, 1, \infty\}$. As is shown in Figure 3.3, when $c > 0$, the flat region of $p(\boldsymbol{x})$ makes the estimation of class conditional probability more difficult [Liu, Zhang and Wu, 2011]. However, the LUM is still able to provide probability estimation for any finite c , although as c increases, the probability information becomes less complete. When $c \to \infty$, the LUM reduces to the standard SVM, which cannot provide any detailed information about probability, due to its minimizer $f^*(\boldsymbol{x}) = \text{sign}(p(\boldsymbol{x}) - 1/2)$.

A similar pattern exists in the MLUM case. In multicategory problems, the conditional probability becomes a vector, and this makes the transition behavior of probability estimation from soft to hard classification more complex. In particular, we need to generalize the flat region in Figure 3.3 to the multicategory setting. From (3.8), we can see that for any given $\hat{\boldsymbol{f}}^n = (\hat{f}_1, \dots, \hat{f}_k)$, the estimated probability depends entirely on $\hat{E}(j)$ and $\hat{F}(j)$; $j = 1, \dots, k$. Note that the LUM loss has the derivative

$$\ell'(u) = \begin{cases} -1 & \text{if } u < \frac{c}{1+c} \\ -\left(\frac{a}{(1+c)u - c + a}\right)^{(a+1)} & \text{if } u \geq \frac{c}{1+c}. \end{cases}$$

When $\hat{f}_i \in [-\frac{c}{1+c}, \frac{c}{1+c}]$, $\hat{f}_j \in [-\frac{c}{1+c}, \frac{c}{1+c}]$, one can see that $\hat{E}(i) = \hat{E}(j)$, $\hat{F}(i) = \hat{F}(j)$, and consequently we have $\hat{P}_i = \hat{P}_j$. This implies that for any obtained $\hat{\boldsymbol{f}}^n$, if the classification signal is weak such that

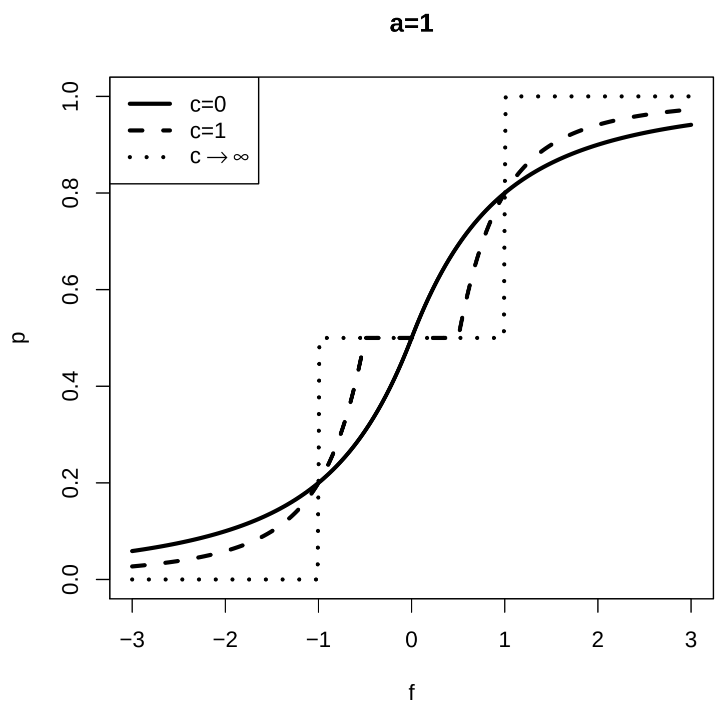


Figure 3.3: Plots of the correspondence between $f^*(\boldsymbol{x})$ and $p(\boldsymbol{x})$ with $a = 1$, $c \in \{0, 1, \infty\}$.

both $\hat{f}_i$ and $\hat{f}_j$ fall into $[-\frac{c}{1+c}, \frac{c}{1+c}]$, then we are not able to tell the difference between the two classes in terms of conditional probabilities. Moreover, if $\hat{f}_j \in [-\frac{c}{1+c}, \frac{c}{1+c}]$ for all $j \in \{1, \ldots, k\}$, then $\hat{P}_j = 1/k$ for all j using (3.8). In that case, the estimated probability cannot help us to identify the max probability class.

To further illustrate the relationship between $\hat{\boldsymbol{f}}^n$ and $\hat{\boldsymbol{P}}$, we use Figure 3.4 to display the relationship between $(\hat{f}_1, \hat{f}_2)$ and $\hat{P}_1$ for $k = 3$, $\gamma = 1$, $a = 1$ and $c = 0, 2, 5$ and ∞. When $c > 0$, the flat region of $\hat{\boldsymbol{P}}$ makes the estimation of the conditional probability more difficult. As c increases, $[-\frac{c}{1+c}, \frac{c}{1+c}]$ becomes wider, and the function becomes closer to a step function. Eventually, when $c \to \infty$, the method reduces to the RMSVM whose classification function $\boldsymbol{f}$ can only produce classification boundary without further probability information. Therefore, similar to the binary case, the MLUM can provide class probability estimation for any finite c, although the estimation deteriorates as c increases.

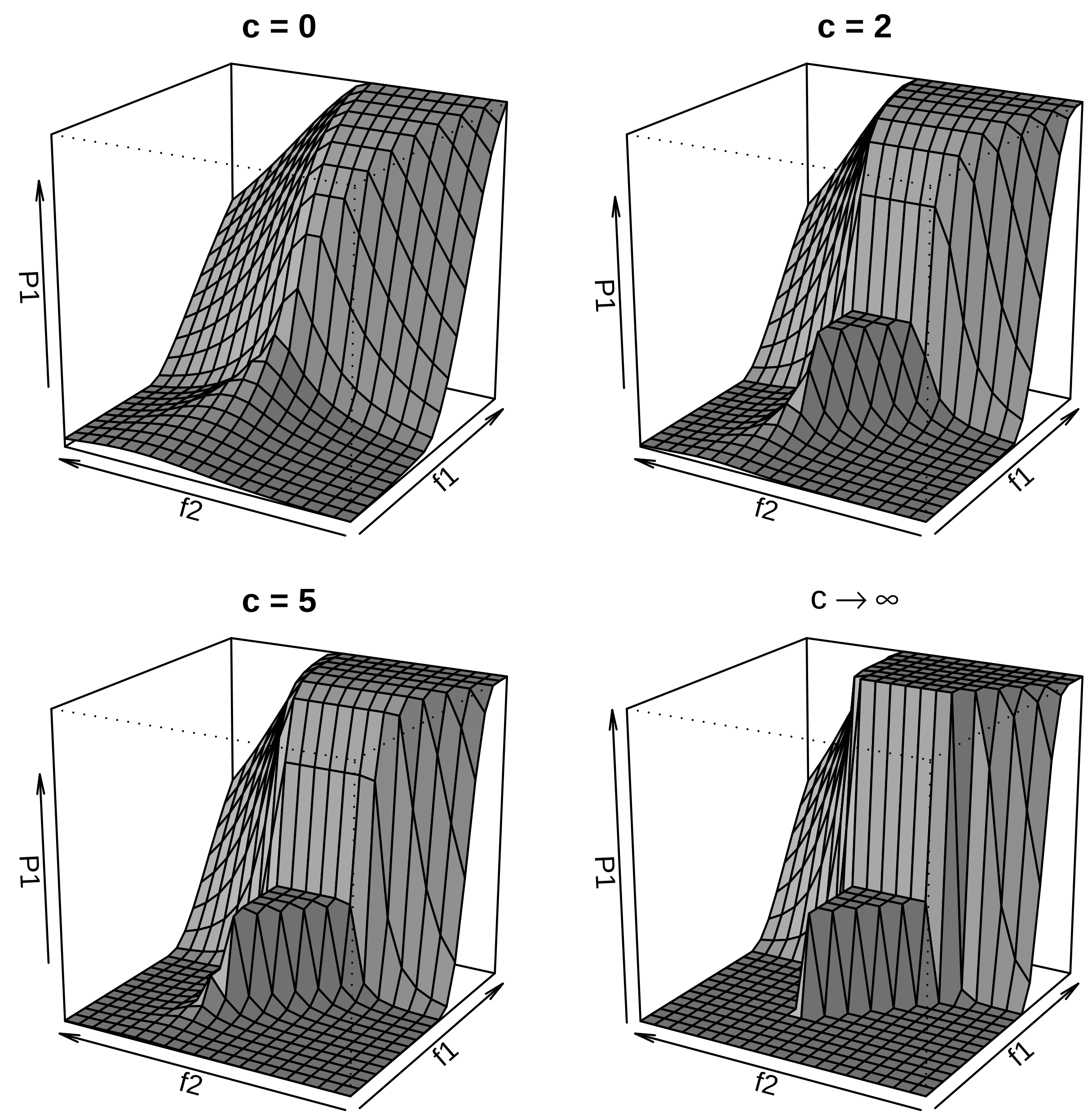


Figure 3.4: Visualization of the relationship between f_1^*, f_2^* and P_1, with $\gamma = 1$, $a = 1$, and $c \in \{0, 2, 5, \infty\}$. The plots show the transition of the MLUM from soft classification (top left panel) to hard classification (bottom right panel).

3.3.3 Asymptotic Properties

For binary problems, Zhang [2004*b*] and Bartlett, Jordan and McAuliffe [2006] investigated the effect of employing the surrogate loss in place of the $0-1$ loss, in terms of classification performance. They considered the excess risk and the excess $V-$risk, defined as follows. The excess risk is the difference between the expected $0-1$ loss for f and its theoretical infimum. It can be written as $R(f(\cdot)) - R^*$, where $R(f(\cdot))$ and R^* are defined in Section 3.2.1. Similarly, the excess $V-$risk can be written as $Q(f(\cdot)) - Q^*$, where $Q(f(\cdot)) = E_{\boldsymbol{X},Y} V(f(\boldsymbol{X}), Y)$ is the expected loss using V as the loss function, and $Q^* = \min Q(f(\cdot))$.

Typically we are interested in the convergence of the excess risk. Steinwart and Scovel [2007] studied the convergence rate of the Gaussian kernel binary SVM problem. For problems with general loss functions in the binary case, Zhang [2004*b*] and Bartlett, Jordan and McAuliffe [2006] showed that if the excess V-risk converges to 0, so does the excess risk, under some mild assumptions. Zhang [2004*a*] further studied the relationship between the two excess risk functionals in multicategory problems. In particular, the convergence rates of the two excess risks are well studied in Wang and Shen [2007], in the setting of L_1 penalized multicategory SVM with linear learning. In this section, we employ the generalization of the excess V-risk from binary to multicategory cases as in Zhang [2004*a*], and explore the explicit form for MLUM. Moreover, we study the relationship between the convergence rate of the classification function $\hat{\boldsymbol{f}}^n$ and that of the excess V-risk, as well as the size of the functional space.

Recall that we can rewrite $Q(\boldsymbol{f}(\cdot))$ as $E_{\boldsymbol{X}} S(\boldsymbol{f}, \boldsymbol{x})$. Consider the MLUM case with $S(\boldsymbol{f}, \boldsymbol{x}) = \sum_{j=1}^k \left[\gamma \ell(f_j(\boldsymbol{x})) + (1-\gamma) \sum_{i \neq j} \ell(-f_i(\boldsymbol{x}))\right] P_j(\boldsymbol{x})$. It can be verified that

$$S(\boldsymbol{f}, \boldsymbol{x}) = \sum_{j=1}^k P_j(\boldsymbol{x})[\gamma \ell(f_j(\boldsymbol{x})) - (1-\gamma)\ell(-f_j(\boldsymbol{x}))] + (1-\gamma)\sum_{j=1}^k \ell(-f_j(\boldsymbol{x})).$$

For brevity, let $Q(\boldsymbol{P}, \boldsymbol{f}) = \sum_{j=1}^k P_j[\gamma\ell(f_j) - (1-\gamma)\ell(-f_j)] + (1-\gamma)\sum_{j=1}^k \ell(-f_j)$. Note that $Q(\boldsymbol{P}, \boldsymbol{f})$ does not involve $\boldsymbol{x}$. For any given $\boldsymbol{P}$, let $\boldsymbol{f}_\ell^*(\boldsymbol{P}) = \operatorname{arginf}_{\boldsymbol{f} \in \mathbf{R}^k} Q(\boldsymbol{P}, \boldsymbol{f})$, where $\mathbf{R}^k$ is the $k-$dimensional real space. Define $Q^*(\boldsymbol{P}) = Q(\boldsymbol{P}, \boldsymbol{f}_\ell^*(\boldsymbol{P}))$ to be the optimum value for any given $\boldsymbol{P}$, and $\Delta Q(\boldsymbol{P}, \boldsymbol{f}) = Q(\boldsymbol{P}, \boldsymbol{f}) - Q^*(\boldsymbol{P})$. The excess $V-$risk is equivalent to $\Delta Q(\boldsymbol{f}(\cdot)) = E_{\boldsymbol{X}} \Delta Q(\boldsymbol{P}(\boldsymbol{X}), \boldsymbol{f}(\boldsymbol{X}))$. In Zhang [2004*b*] and Bartlett, Jordan and McAuliffe [2006], the authors showed that in binary problems, we can bound the excess risk by some transformation of $\Delta Q(\boldsymbol{f}(\cdot))$. A similar result for multicategory problem is obtained in Zhang [2004*a*]. To study the functional form of $\Delta Q(\boldsymbol{f}(\cdot))$ in binary problems, Zhang [2004*b*] proposed to use the Bregman divergence. To our knowledge, not much has been done for multicategory classification. Here we extend the results in Zhang [2004*b*] to the multicategory version.

The Bregman divergence of a convex function $g(\cdot)$ is defined as

$$d_g(f_1, f_2) = g(f_2) - g(f_1) - g'(f_1)(f_2 - f_1).$$

Here $g'(\cdot)$ is a subgradient of the convex function $g(\cdot)$. In Theorem 2.2 of Zhang [2004*b*], it was shown that in the binary case, if g is differentiable, $\Delta Q(P, f) = P d_g(f_g^*(P), f) + (1 - P) d_g(-f_g^*(P), -f)$. For the MLUM family, we have the following result, which is a generalization of the binary formula.

Theorem 3.3.3. $\Delta Q(\boldsymbol{P}, \boldsymbol{f}) = \sum_{j=1}^{k} \left[P_j \gamma d_\ell(f_j^*, f_j) + (1 - P_j)(1 - \gamma) d_\ell(-f_j^*, -f_j) \right].$

Theorem 3.3.3 can be used to establish the connection of the convergence rate of the classification function and that of the excess $V-$risk. Suppose the classification function $\boldsymbol{f}$ varies in a certain space $\mathcal{F}$. We can decompose the excess $V-$risk as $\Delta Q(\hat{\boldsymbol{f}}^n) = [\Delta Q(\hat{\boldsymbol{f}}^n) - \Delta Q(\boldsymbol{f}^{\mathcal{F}})] + [\Delta Q(\boldsymbol{f}^{\mathcal{F}})]$, where $\boldsymbol{f}^{\mathcal{F}}$ is the function that achieves the minimal of $\Delta Q(\boldsymbol{f}) : \boldsymbol{f} \in \mathcal{F}$ pointwisely. We may refer to the first term in the previous display as the $V-$estimation error and the second term as the $V-$approximation error. When $\mathcal{F}$ is rich enough, so that

$$\underset{\boldsymbol{f} \in \mathcal{F}}{\operatorname{arginf}} \, E_{\boldsymbol{X}, Y}(V(\boldsymbol{f}(\boldsymbol{X}), Y)) = \underset{\boldsymbol{f} \in \mathbb{R}^k}{\operatorname{arginf}} \, E_{\boldsymbol{X}, Y}(V(\boldsymbol{f}(\boldsymbol{X}), Y)) = \boldsymbol{f}^*, \tag{3.9}$$

the estimator $\hat{\boldsymbol{f}}^n$ will converge to $\boldsymbol{f}^*$ as the sample size n grows, under some mild conditions. In this case the $V-$approximation error is zero. We can explore how the convergence rate of $\hat{\boldsymbol{f}}^n$ affects the convergence rate of the excess $V-$risk, which is essentially the $V-$estimation error. First we introduce some definitions and assumptions. Recall that a, c and γ are parameters in the MLUM loss function (3.5).

Let $\mu(\cdot)$ be the regular Lebesgue measure. For any fixed a, c and γ, the distribution $\mathbb{P}(\boldsymbol{X}, Y)$ naturally defines k probability measures on the real line: $\tau_j(B) = P(f_j^* \in B); j = 1, \ldots, k$, where B is any Borel measurable set.

Assumption A: For any $c \geq 0$, $a > 0$ and $\gamma \in [0, 1]$, $\tau_j \ll \mu; j = 1, \ldots, k$. Namely, every measure τ_j is absolutely continuous with respect to the Lebesgue measure μ.

Assumption B: For any $c \geq 0$, $a > 0$ and $\gamma \in [0, 1]$, $n^q(\hat{\boldsymbol{f}}^n(\boldsymbol{x}, y) - \boldsymbol{f}^*(\boldsymbol{x}, y)) \to \boldsymbol{T}(c, a, \gamma, \boldsymbol{x}, y)$ in distribution, where $\boldsymbol{T}(c, a, \gamma, \boldsymbol{x}, y) = (T_1, \ldots, T_k)^T$ is a multivariate random variable, whose distribution depends on c, a, γ, and varies among different $(\boldsymbol{x}, y)$; $q > 0$ is a constant. Furthermore, suppose that for fixed c, a, and γ, $\int_{\boldsymbol{X},Y} |\sup_{1\leq j\leq k} T_j|^2 d\mathbb{P}(\boldsymbol{X},Y) < \infty$.

Now we are in the position of introducing the following theorem.

Theorem 3.3.4. *In the MLUM family with the underlying distribution $\mathbb{P}$, suppose Assumptions A and B are satisfied, and (3.9) holds. Then for any fixed c, a, and γ,*

$$\Delta Q(\hat{\boldsymbol{f}}^n) = O(n^{-2q}).$$

Note that Assumption A can be verified if there is no probability mass point in the distribution D. Under Assumption A, the expected loss $Q(\boldsymbol{f}(\cdot))$ has bounded second order derivative for a fixed c almost surely. Hence under some conditions, $q = 1/2$ for regular finite dimensional problems, and $\hat{\boldsymbol{f}}^n$ is root $n-$consistent. For example, in the literature of sieve estimation with linear learning, a finite dimensional problem enjoys the root $n-$consistency of $\hat{\boldsymbol{\beta}}$ [Shen and Wong, 1994], with a proper choice of regularization term. In this case, the integrability in Assumption B can be satisfied if $\boldsymbol{X}$ is bounded. From Theorem 3.3.4, we can conclude the excess $V-$risk is $n-$consistent for any a, c and γ, when the function space is large enough.

Remark 3.1. Assumption A ensures that there is no probability mass point where the LUM loss function $\ell(\cdot)$ is not twice differentiable. Without Assumption A we are not even guaranteed to have convergence for any suitable transformation of $\hat{\boldsymbol{f}}^n - \boldsymbol{f}^*$. The square integrable requirement of Assumption B is essential in the sense that it prevents the distribution of $\boldsymbol{T}$ from diverging with large probability when $(\boldsymbol{X}, Y)$ varies.

Remark 3.2. The potential problem that $\boldsymbol{f}^*$ is not unique does not affect the result of Theorem 3.3.4. Because S in (3.7) is convex, any partial derivative of S with respect to f_j is non-decreasing. Suppose there is a flat region $[h_1, h_2]$ of value 0 in the derivative function. Then f_j^* must be within $[h_1, h_2]$ and the second order partial derivative with respect to f_j is 0 in (h_1, h_2). Because the MLUM loss function is continuously differentiable, either the first order derivative is not differentiable on the boundary of (h_1, h_2), which we assume probability 0, or it is differentiable with derivative 0. Note that if the first order derivative of a convex function $\psi(\cdot)$ is 0 within (h_1, h_2), then for any $t_1, t_2 \in (h_1, h_2)$, and any other t_3, $d_\psi(t_1, t_3) = d_\psi(t_2, t_3)$. This means that the choice of f_j^* is not essential, as long as the empirical minimizer $\hat{f}_j$ approaches $[h_1, h_2]$.

The situation when the $V-$approximation error vanishes is rare. When it is nonzero, in other words,

$$\inf_{\boldsymbol{f}\in\mathcal{F}} E_{\boldsymbol{X},Y}(V(\boldsymbol{f}(\boldsymbol{X}),Y)) > \inf_{\boldsymbol{f}\in\mathbb{R}^k} E_{\boldsymbol{X},Y}(V(\boldsymbol{f}(\boldsymbol{X}),Y)), \tag{3.10}$$

Theorem 3.3.4 is not applicable, because the excess $V-$risk does not converge to 0 any more. Then we are interested in the convergence rate of the $V-$estimation error, $\Delta Q(\hat{\boldsymbol{f}}^n) - \Delta Q(\boldsymbol{f}^{\mathcal{F}})$. First, we need to modify Assumption B as follows.

Assumption B': For any $c \geq 0$, $a > 0$ and $\gamma \in [0,1]$, $n^q(\hat{\boldsymbol{f}}^n(\boldsymbol{x},y) - \boldsymbol{f}^*(\boldsymbol{x},y)) \to \boldsymbol{T}(c,a,\gamma,\boldsymbol{x},y)$ in distribution, where $\boldsymbol{T}(c,a,\gamma,\boldsymbol{x},y) = (T_1,\ldots,T_k)^T$ is a multivariate random variable, whose distribution depends on c, a, γ, and varies among different $(\boldsymbol{x},y)$; $q > 0$ is a constant. Furthermore, suppose that for fixed c, a, and γ, $\int_{\boldsymbol{X},Y} |\sup_{1\leq i\leq k} T_i| d\mathbb{P}(\boldsymbol{X},Y) < \infty$, and $\int_{\boldsymbol{X},Y} |\sup_{1\leq i\leq k} f_i^*| + |\sup_{1\leq i\leq k} f_i^H| d\mathbb{P}(\boldsymbol{X},Y) < \infty$.

Theorem 3.3.5. *In the MLUM family with the underlying distribution $\mathbb{P}$, suppose Assumptions A and B' are satisfied, and (3.10) holds. Then for any fixed c, a, and γ,*

$$\Delta Q(\hat{\boldsymbol{f}}^n) - \Delta Q(\boldsymbol{f}^{\mathcal{F}}) = O(n^{-q}).$$

From Theorem 3.3.5, we can see that when the theoretical minimizer does not belong to the function class $\mathcal{F}$, the convergence rate of the excess V-risk is the same as $\hat{\boldsymbol{f}}^n$. When $q = 1/2$ under some mild conditions, the excess V-risk is also root n-consistent.

3.4 Computational Algorithm

In this section, we discuss how to implement (3.3) with the MLUM family. The MLUM loss (3.5) is convex and first-order differentiable, and one can apply standard tools to solve the optimization problem, for example, the OPTIM function in R. Here we propose to use the well-known cyclic coordinate descent algorithm [Tseng, 2001; Friedman, Hastie and Tibshirani, 2010*a*].

Next we present the coordinate descent algorithm to solve the following MLUM optimization problem

$$\min_{\boldsymbol{f}} \frac{1}{n}\sum_{i=1}^{n}\left[\gamma\ell(f_y(\boldsymbol{x}_i)) + (1-\gamma)\sum_{j\neq y}\ell(-f_j(\boldsymbol{x}_i))\right] + \lambda J(\boldsymbol{f}),$$
$$\text{subject to } \sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0.$$

We choose the first $(k-1)^{th}$ elements of $\boldsymbol{f}$ as free parameters, and let $\sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$. For simplicity we focus on the linear learning with $f_j(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta}_j + b_j, j = 1,\ldots,k$, and the L_2 penalty with $J(\boldsymbol{f}) = \frac{1}{2}\sum_{j=1}^{k}\|\boldsymbol{\beta}_j\|^2$. Extensions to kernel learning and other convex and separable types of regularization term are relatively straightforward and are not included here.

We now describe the coordinate descent algorithm in detail. Denote the solution at the m^{th} step by $\Xi^{(m)} = (\boldsymbol{\beta}_1^{(m)},\ldots,\boldsymbol{\beta}_k^{(m)})^T$ and $\boldsymbol{b} = (b_1^{(m)},\ldots,b_k^{(m)})^T$. At the $(m+1)^{th}$ step, define $\tilde{\boldsymbol{f}}_{i,j,-p} = (\boldsymbol{x}_i^T\boldsymbol{\beta}_1^{(m+1)} + b_1^{(m+1)},\ldots,\boldsymbol{x}_i^T\boldsymbol{\beta}_{j-1}^{(m+1)} + b_{j-1}^{(m+1)}, W_{i,1},\ldots,\boldsymbol{x}_i^T\boldsymbol{\beta}_{k-1}^{(m)} + b_{k-1}^{(m)}, W_{i,2})^T$ for $i = 1,\ldots,n$, $j = 1,\ldots,k-1$ and $p = 1,\ldots,d$, where d is the dimension of the problem. Here $W_{i,1} = \sum_{q<p} x_{iq}\Xi_{j,q}^{(m+1)} + \sum_{q>p} x_{iq}\beta_{j,q}^{(m)} + b_j^{(m)}$, and $W_{i,2}$ is determined by other components such that the component sum of $\tilde{\boldsymbol{f}}_{i,j,-p}$ is 0. Set

$$\Xi_{j,p}^{(m+1)} = \operatorname*{argmin}_z \frac{\lambda}{2}\Big(z^2 + [\sum_{s=1}^{p-1}\Xi_{j,s}^{(m+1)} + z + \sum_{s=p+1}^{d}\Xi_{j,s}^{(m)}]^2\Big) + \frac{1}{n}\sum_{i=1}^{n} V(\tilde{\boldsymbol{f}}_{i,j,-p} + x_{i,j}z, y_i).$$

The optimization above involves a one-dimensional update and can be solved efficiently. Once we finish updating the entire j^{th} row of $\Xi^{(m+1)}$, we update the intercept b_j using a similar idea as the above optimization without the regularization term. We continue the iteration until convergence.

The above coordinate descent method can be summarized as the following pseudo-code.

```
Step 1 Start with $\boldsymbol{\beta}_j = (0,\ldots,0)^T$ and $b_j = 0$, for all $j = 1,\ldots,k$. We keep $\boldsymbol{\beta}_k = -\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2 - \cdots - \boldsymbol{\beta}_{k-1}$
       and $b_k = -b_1 - b_2 - \cdots - b_{k-1}$.

Step 2 At the beginning of the $m^{th}$ loop, suppose we have $(\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_k)$ and $(b_1,\ldots,b_k)$ as the inter-
       mediate results.

    2.1 Update, in orders, the elements of $\boldsymbol{\beta}_1$.

    2.2 After the entire vector $\boldsymbol{\beta}_1$ has been updated, update the intercept $b_1$.
```

2.3 Update the vectors $\boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_{k-1}$ and the intercepts $b_2, \ldots, b_{k-1}$, in the same manner as above.

Step 3 Repeat Step 2 until convergence.

3.5 Simulated Examples

In this section, we use six simulated examples to demonstrate the numerical performance of the MLUM family. Our unified algorithm, discussed in Section 3.4, greatly facilitates a systematic exploration and comparison of different types of classifiers. The MLUM also provides class conditional probability estimation as a by-product. The simulated examples we consider here are different scenarios in which the behavior of different classifiers varies from one to another. Since we know the underlying data generation schemes for simulated examples, the results can shed some light on the behavior of MLUM and offer some insights on the choice of methods in different situations.

We divide this section into four subsections. Section 5.5.1 contains two examples in which the hard classifier works the best. Section 5.5.2 consists of two examples that favor the soft classifier. Section 5.5.3 includes two examples where the MLUM with $c = 1$, corresponding to a new multicategory DWD, can outperform both hard and soft classifiers, which was not observed in the binary LUM case in Liu, Zhang and Wu [2011]. Section 5.5.4 provides some summary on the effect of hard versus soft classifiers and helps to shed light on the choice of classification methods. For each simulation, we apply the linear learning with the L_2 penalty, and the corresponding tuning parameter λ is selected by minimizing the classification error over a separate tuning set using a grid search.

In practice, a systematic tuning procedure for selection of the optimal (a, c, γ) is needed. The three-dimensional tuning can be very time consuming, and we suggest a fast and effective tuning procedure for the MLUM family. Similar to the binary case, the choice of a in the MLUM loss has little impact on the performance of the classifiers, so we fix a at 1. Interestingly, we find that the behavior of MLUM with $c = 1$ can outperform the hard ($c \to \infty$) and soft ($c = 0$) classifiers in certain cases. Based on our numerical experience, we suggest to choose γ from $\{0, 0.25, 0.5, 0.75, 1\}$. In particular, we propose to tune the MLUM with $c \in \{0, 1, 1000\}$ and $\gamma \in \{0, 0.25, 0.5, 0.75, 1\}$, holding a fixed at 1. We call this procedure the "tuned MLUM".

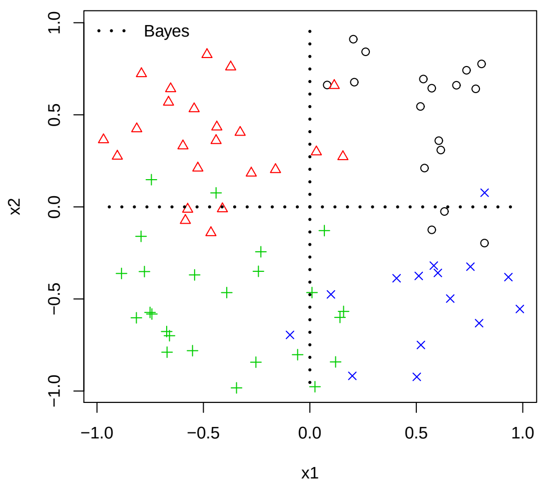
For each example, we evaluate both classification performance and probability estimation. We use the test error to measure classification accuracy, on a test set with size 10^6. To explore the effect of different c independently, we let c vary in $\{0, 1, 10, 100, 1000\}$, and let the tuning procedure automatically choose the best γ from $\{0, 0.25, 0.5, 0.75, 1\}$. We call this procedure "tuned γ, fixed c". Similarly, with γ varying in $\{0, 0.1, 0.2, \dots, 0.9, 1\}$, we select the best c from $\{0, 1, 1000\}$, and call the resulting classifier "tuned c, fixed γ", to observe the effect of γ. For comparison, we also include the results of Losses 1-4 mentioned in Section 3.2.2, RMSVM, Multicategory Penalized Logistic Regression (MPLR), along with the tuned MLUM, in Tables 3.7. Here for the MPLR, we replace the loss function ℓ by the logistic loss $V(f, y) = \log(1+e^{-yf})$, while keeping the convex combination so that the classifier is tuned with different γ values. We note that this MPLR is new and we can call it the "tuned MPLR", analogous to the tuned MLUM. Note that in most examples, the tuned MLUM performs better than the other methods, and is recommended.

We conduct 1000 replications for each simulation, and report the average test errors. For probability estimation, we use the criterion of the mean absolute error (MAE), $E_{\boldsymbol{X}}|p(\boldsymbol{X}) - \hat{p}(\boldsymbol{X})|$. The MLUMs with $c \in \{0, 1, 1000\}$ and $\gamma \in \{0, 0.25, 0.5, 0.75, 1\}$ are fit to explore the pattern of probability estimation accuracy. Through 1000 replicates, MAEs with their standard errors are reported. As the number of covariates for Examples 1, 2, 4 and 5 is 2, we plot the corresponding marginal distributions of $\boldsymbol{x}$ for typical training samples in Figure 3.5.

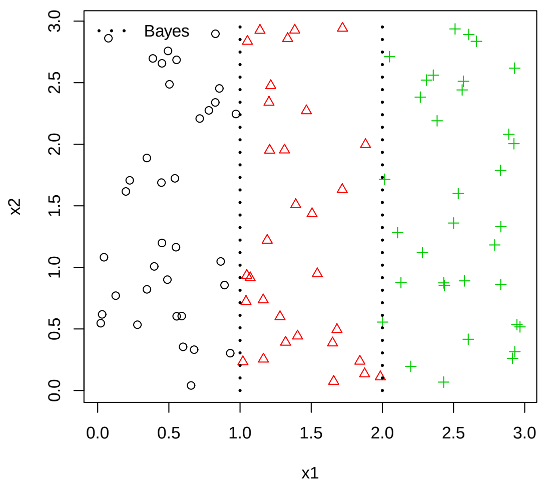
3.5.1 When Hard Classification is Better

In this section, we generate the data such that the underlying probability functions are step functions. In this case, estimation of the conditional probability function is challenging, and we expect the hard classifier to perform better because it bypasses probability estimation.

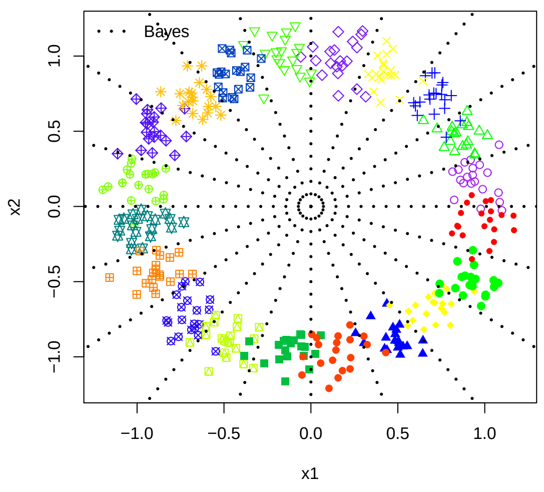
Example 1: We generate two dimensional data uniformly on $[-1, 1]^2$, with four classes. One sample of X is given in Figure 3.5(a). Conditional on X_1 and X_2, the class output is generated as follows. Let $Y = 1$ if $X_1 > 0.2$ and $X_2 > 0.2$, $Y = 2$ if $X_1 < -0.2$ and $X_2 > 0.2$, $Y = 3$ if $X_1 < -0.2$ and $X_2 < -0.2$ and $Y = 4$ if $X_1 > 0.2$ and $X_2 < -0.2$. When $X_1 > 0.2$ and $X_2 \in [-0.2, 0.2]$, $P(Y = 1) = P(Y = 4) = 1/2$. Similarly, when $X_1 < -0.2$ and $X_2 \in [-0.2, 0.2]$, $P(Y = 2) = P(Y = 3) = 1/2$, and when $X_2 > 0.2$ and $X_1 \in [-0.2, 0.2]$, $P(Y = 1) = P(Y = 2) = 1/2$, and when $X_2 < -0.2$ and $X_1 \in [-0.2, 0.2]$, $P(Y = 2) = P(Y = 4) = 1/2$. When the data points fall in $[-0.2, 0.2]^2$, the probabilities of being in four classes are equal. In this example we use 80 data points for training and



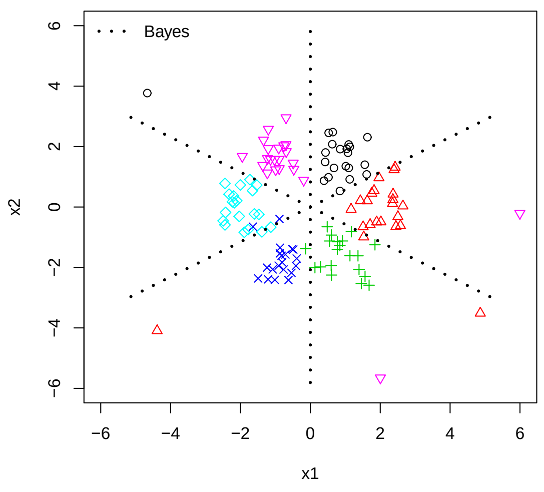
(a)



(b)



(c)



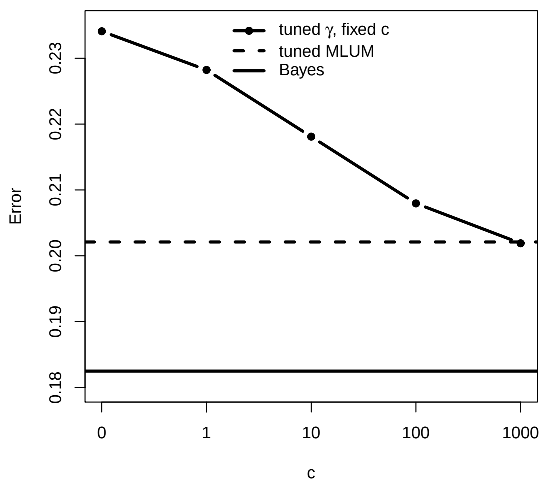
(d)

Figure 3.5: Plots of a typical sample for Examples 1(a), 2(b), 4(c) and 5(d).

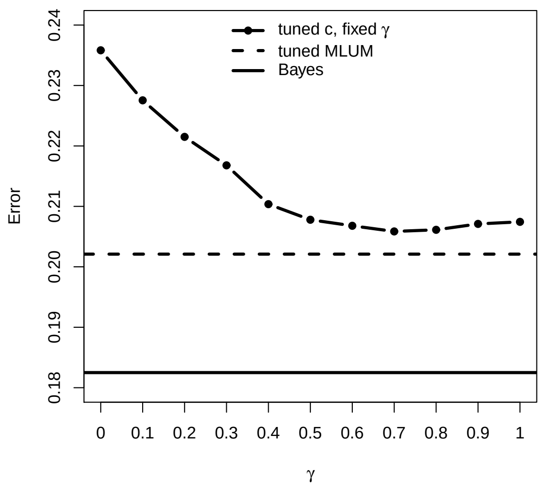
another 80 for tuning.

This is a multicategory generalization of Example 2 in Liu, Zhang and Wu [2011]. The underlying conditional class probability function of this example is a step function. Thus class probabilities are difficult to estimate in this case, and the classification accuracy of soft classifiers may be sacrificed by probability estimation. We expect hard classifiers to perform better than the soft ones, because the former bypasses probability estimation.

The test errors of this example are reported in Figure 3.6. As we expect, hard classifiers outperform soft ones. Interestingly, the hard classifier is not at its optimal with $\gamma = 0.5$, but at $\gamma = 1$, although the difference between them is relatively small. This seems to be different from the examples in Liu and Yuan [2011]. We would like to point out that, with finite c, the classification performance of MLUM differs from RMSVM, because the MLUM with finite c is always Fisher consistent, while the RMSVM does not guarantee consistency for $\gamma > 0.5$. Also, the tuned MLUM is more flexible and achieves the best classification accuracy. The MAEs are reported in Table 3.1, and the soft classifier gives the best probability estimation.



(a) The test error of Example 1 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.



(b) The test error of Example 1 with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

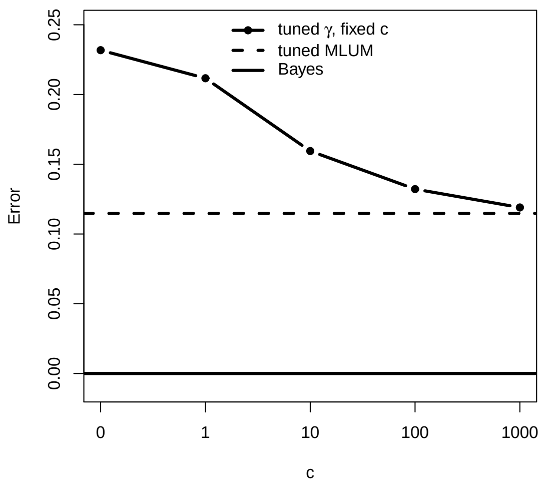
Figure 3.6: Classification error rates in Example 1. The left panel shows that the hard classifier performs better than the other ones in this example. The right panel shows that the test error is minimized with γ around 0.7.

MAE	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$c = 0$	0.1226	0.1208	0.1165	0.1050	0.0491
	(0.005844)	(0.001845)	(0.002189)	(0.003622)	(0.009019)
$c = 1$	0.1422	0.1368	0.1328	0.1215	0.0935
	(0.005240)	(0.003535)	(0.005132)	(0.003987)	(0.004582)
$c = 1000$	0.1424	0.1387	0.1372	0.1321	0.1267
	(0.002661)	(0.005278)	(0.001347)	(0.004159)	(0.008138)

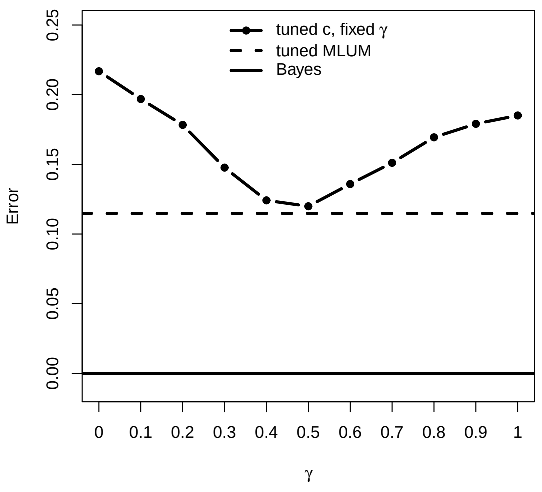
Table 3.1: The MAEs for different c and γ in Example 1.

Example 2: We generate a three-class problem in this example, and $\boldsymbol{X}$ is two dimensional. One sample of X is illustrated in Figure 3.5(b). The first class is uniformly distributed in the square $x_1 \in [0,1], x_2 \in [0,3]$, the second class is uniform from $x_1 \in [1,2], x_2 \in [0,3]$, and the third class is uniform from $x_1 \in [2,3], x_2 \in [0,3]$. There are 60 training data points and 60 tuning ones, respectively.

We report the classification errors in Figure 3.7, and the MAEs in Table 3.2. Based on the results, both test errors and MAEs suggest that the hard classifier performs significantly better than the other ones. Interestingly, the soft classifier gives worse probability estimation than hard classifiers in this example. Here the parameter $\gamma = 0.5$ works better than other values, which is consistent with the findings in Liu and Yuan [2011].



(a) The test error of Example 2 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.



(b) The test error of Example 2 with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.7: Classification error rates in Example 2. The hard classifier works the best, as is suggested by the left panel. The right panel shows that $\gamma = 0.5$ is optimal.

MAE	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$c = 0$	0.2113	0.1969	0.1933	0.1887	0.1892
	(0.006877)	(0.006228)	(0.004803)	(0.001011)	(0.009567)
$c = 1$	0.1790	0.1734	0.1662	0.1693	0.1674
	(0.001738)	(0.005951)	(0.003742)	(0.006418)	(0.003838)
$c = 1000$	0.1551	0.1523	0.1469	0.1430	0.1422
	(0.002738)	(0.007453)	(0.007681)	(0.001394)	(0.008392)

Table 3.2: The MAEs for different c and γ in Example 2.

3.5.2 When Soft Classification is Better

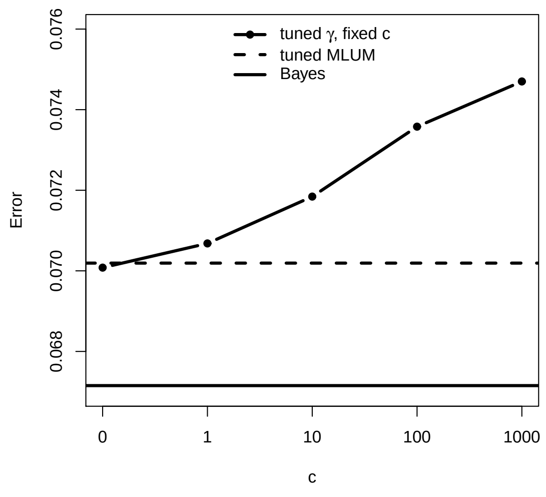
In this section we generate the data so that the conditional probability function is smooth and relatively easy to estimate. In such cases the accurate probability estimation may help the soft classifier to build more accurate classification boundaries.

Example 3: This example involves a three-dimensional feature space with eight classes. In particular,

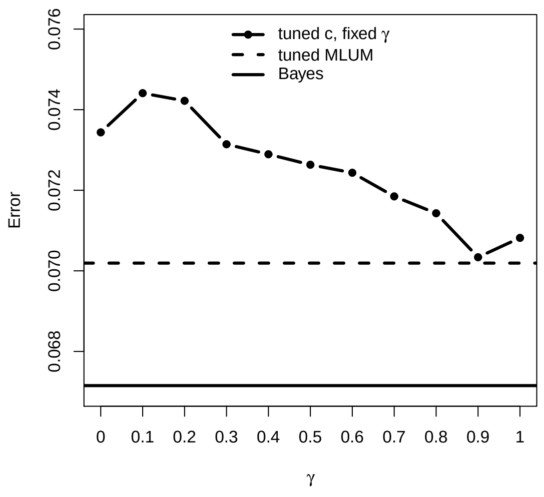
$$P(Y = j) = 1/8, P(\boldsymbol{X}|Y = j) \sim N(\mu_j, 0.5^2 I_2); j = 1, \ldots, 8,$$

where $\mu_j = (1,1,1)^T$, $(1,1,-1)^T$, $(1,-1,1)^T$, $(1,-1,-1)^T$, $(-1,1,1)^T$, $(-1,1,-1)^T$, $(-1,-1,1)^T$, $(-1,-1,-1)^T$, for $j = 1, \ldots, 8$, respectively. Because the number of classes is large, we generate 160 observations for training, and another 160 for tuning.

The classification performance is reported in Figure 3.8, and the MAEs are reported in Table 3.3. This is a case in which the soft classifier works the best both in terms of classification accuracy as well as probability estimation.



(a) The test error of Example 3 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.



(b) The test error of Example 3 with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.8: Classification error rates in Example 3. The left panel shows that this is an example in which soft classifier works the best. The right panel illustrates that the test error is minimized with $\gamma = 0.9$.

MAE	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$c = 0$	0.2362	0.2183	0.1956	0.1617	0.0887
	(0.005048)	(0.008325)	(0.007328)	(0.004206)	(0.000884)
$c = 1$	0.2629	0.2438	0.2230	0.1785	0.0867
	(0.003554)	(0.003873)	(0.006405)	(0.001282)	(0.000671)
$c = 1000$	0.2867	0.2540	0.2432	0.1990	0.1073
	(0.003513)	(0.001959)	(0.007493)	(0.002192)	(0.000560)

Table 3.3: The MAEs for different c and γ in Example 3.

Example 4: In this example we have 2 covariates, and the number of classes is 20. A sample of X is drawn in Figure 3.5(c). Each marginal distribution of $\boldsymbol{X}|Y = j; j = 1, \ldots, 20$ follows a $N(\mu_j, \sigma^2 I_2)$ distribution. Here we choose μ_j such that $\mu_j; j = 1, \ldots, 20$ are evenly spaced on the unit circle. We choose σ such that the Bayes error is 0.1, and we use 400 training data points and another 400 for tuning.

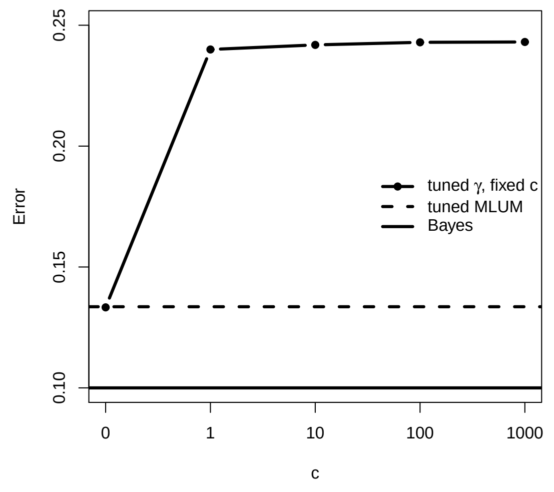
The test errors are reported in Figure 3.9, and the MAEs in Table 3.4. In this example, the soft classifier significantly dominates the others in terms of prediction accuracy. The tuned MLUM method always chooses $c = 0$ through out the 1000 simulations. The MLUM family yields the best accuracy with γ approximately 0.7. The MAEs for different c and γ do not differ much, possibly due to the large number of classes.

MAE	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$c = 0$	0.0801	0.0799	0.0812	0.0813	0.0806
	(0.001627)	(0.009215)	(0.001278)	(0.006627)	(0.007840)
$c = 1$	0.0832	0.0836	0.0829	0.0834	0.0822
	(0.007646)	(0.006726)	(0.001157)	(0.008513)	(0.006180)
$c = 1000$	0.0829	0.0831	0.0821	0.0819	0.0816
	(0.008364)	(0.001699)	(0.002103)	(0.003235)	(0.002697)

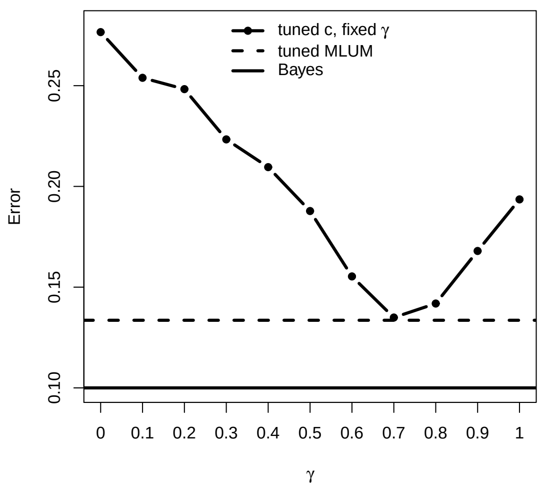
Table 3.4: The MAEs for different c and γ in Example 4.

3.5.3 When MLUM with c=1 is Better

Liu, Zhang and Wu [2011] showed that among many examples, the classifier that works the best in the LUM family appears to be either soft or hard classifiers. In the MLUM family, however, we observe that the MLUM with $c = 1$ (a new multicategory DWD) can sometimes yield the best performance



(a) The test error of Example 4 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.



(b) The test error of Example 4 with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.9: Classification error rates in Example 4. The left panel shows the soft classifier works reasonably well in this example. When other choices of c fails, tuned MLUM automatically selects $c = 0$ and thus keeps a good performance. In the right panel, when $\gamma = 0.7$, the classifier has a minimum error rate.

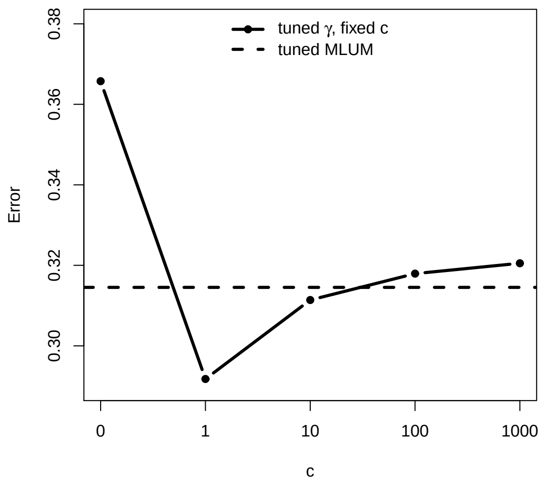
in terms of classification accuracy. In particular, we explore the effect of outliers in Examples 5 and 6. In these two examples, we add a small percentage of noisy data into the originally clean datasets, and observe that soft classifiers can be very sensitive to outliers in terms of classification accuracy, while MLUMs with $c \geq 1$ appear to be quite robust.

Example 5: We plot a sample of X in Figure 3.5(d). In this example, there are 95% data points from a six-class distribution with

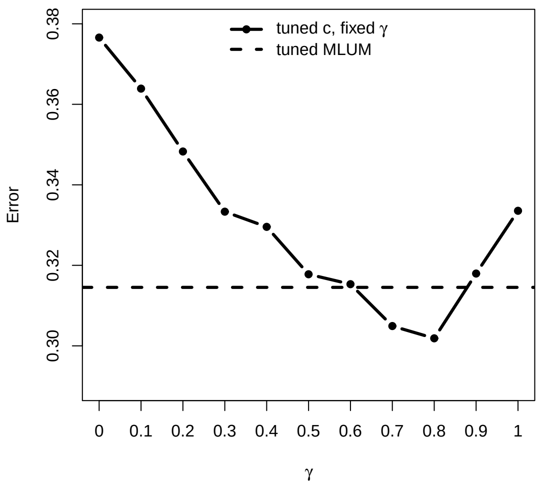
$$P(Y = j) = 1/6, P(\boldsymbol{X}|Y = j) \sim N(\mu_j, 0.5^2 I_2); j = 1, \ldots, 6,$$

where $\mu_j = (1, \sqrt{3})^T$, $(2, 0)^T$, $(1, -\sqrt{3})^T$, $(-1, -\sqrt{3})^T$, $(-2, 0)^T$, $(-1, \sqrt{3})^T$, for $j = 1, \ldots, 6$, respectively. The rest 5% instances are outliers, which are uniformly distributed on the circle $x_1^2 + x_2^2 = 36$ with their labels randomly chosen. We generate independent training and tuning sets, both of size 100.

From the test errors in Figure 3.10, we see that MLUM with $c = 0$ is less stable than the other c values, and the best classifier is $c = 1$. From the MAEs in Table 3.5, we see that the probability estimation with $c = 1$ is more accurate than the others.



(a) The test error of Example 5 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.



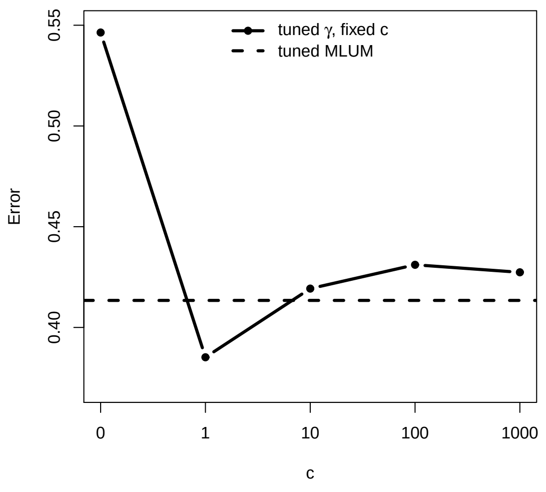
(b) The test error of Example 5 with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.10: Classification error rates in Example 5. As in Example 5, the prediction accuracy with $c = 1$ is the highest. The MLUM with $\gamma = 0.8$ works better than the others.

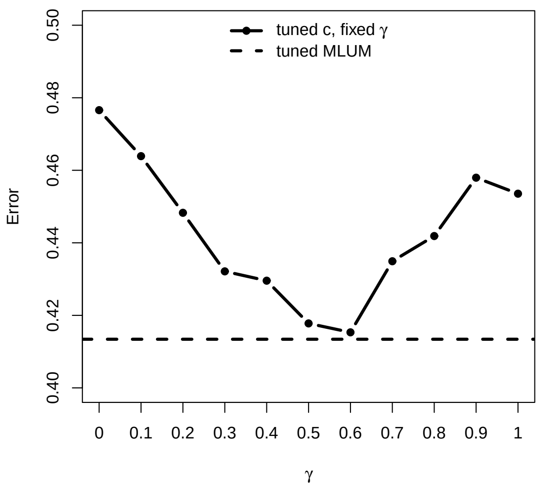
MAE	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$c = 0$	0.2328	0.2321	0.2300	0.2292	0.2303
	(0.003625)	(0.006968)	(0.009631)	(0.005956)	(0.014856)
$c = 1$	0.2312	0.2307	0.2266	0.2241	0.2257
	(0.010180)	(0.004729)	(0.007306)	(0.004518)	(0.001417)
$c = 1000$	0.2432	0.2416	0.2397	0.2397	0.2399
	(0.005403)	(0.004657)	(0.011136)	(0.014401)	(0.006237)

Table 3.5: The MAEs for different c and γ in Example 5.

Example 6: In this example, we add noisy observations to the data in Example 4. As the classification boundary is very sensitive to the outliers, we only have 1% noisy instances, whose distribution is the same as in Example 5. Compared to Figure 3.9, Figure 3.11 shows that $c = 0$ is more sensitive to noise than the other c values, while $c = 1$ is the most robust one. Table 3.6 shows that the probability estimation with $c = 1$ is the best, as in Example 5.



(a) The test error of Example 6 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.



(b) The test error of Example 6 with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.11: Classification error rates in Example 6. In contrast to Example 4, with noisy points in the dataset, $c = 0$ performs worse than the other c values. The right panel suggests that the MLUM works well with γ around 0.6.

MAE	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
$c = 0$	0.1294	0.1267	0.1245	0.1258	0.1267
	(0.007833)	(0.004012)	(0.009447)	(0.005301)	(0.006370)
$c = 1$	0.1198	0.1135	0.1131	0.1142	0.1155
	(0.004456)	(0.010214)	(0.010776)	(0.007559)	(0.003054)
$c = 1000$	0.1326	0.1299	0.1276	0.1268	0.1270
	(0.003148)	(0.002562)	(0.004809)	(0.004215)	(0.003149)

Table 3.6: The MAEs for different c and γ in Example 6.

3.5.4 Summary of Simulation Results

Our simulated examples provide some insights on hard versus soft classification methods, and suggest that there is no single classifier that works universally the best in all cases. Varying from soft to hard, the patterns of classification performance can differ significantly, for different settings. Our simulation studies showed different cases in which hard ($c = \infty$), soft ($c = 0$) and in-between ($c = 1$) classifiers work the best, respectively.

When the underlying conditional probability function is a step function, probability estimation can be a difficult problem for soft classifiers, and the prediction accuracy may be compromised. In Examples 1 and 2, we see that the hard classification method performs better than the others, because it directly focuses on the boundary estimation while bypassing the probability estimation. This finding is consistent with the binary case in Liu, Zhang and Wu [2011].

In contrast to Examples 1 and 2, for Examples 3 and 4, the underlying conditional probability functions are relatively smooth. Soft classifiers with $c = 0$ can build more accurate classification boundaries through estimation of the probability functions. This is also observed in the binary case in Liu, Zhang and Wu [2011].

One new observation is that the MLUM with $c = 1$ may work the best in some situations. This was not reported in the binary LUM cases [Liu, Zhang and Wu, 2011]. Interestingly, the soft classifier is quite vulnerable to potential outliers in the data. In Example 5, we add 5% outliers to the clean samples, and the soft classifier performs the worst among the others. In Example 4 we see that soft classifier outperforms the other methods by around 10% in terms of classification accuracy. However in Example 6 when we add only 1% outliers to the data, the classification accuracy of the soft classifier is reduced by over 40% while that of the others are just around $15 - 20\%$. This indicates that trying to assign a conditional probability estimation to an outlier may severely reduce the accuracy of the classification boundary estimation.

Another observation is that in the simulated datasets, the optimal γ is always in or close to the recommended set $\{0, 0.25, 0.5, 0.75, 1\}$. Therefore the tuned MLUM procedure is sufficient enough to avoid tuning on the entire interval $\gamma \in [0, 1]$. When we are uncertain about which classifier to use, we propose to apply our tuned MLUM method. As is shown in the simulation studies, the tuned MLUM automatically selects the near optimal classifier from a rich family, and has advantages in terms of classification accuracy. Lastly, Table 3.7 reports comparisons of five MSVM methods, as discussed in Section 3.2.2, MPLR, with our proposed MLUM. The results show that MLUM delivers the most accurate classifier in most cases.

	MSVM 1	MSVM 2	MSVM 3	MSVM 4	RMSVM	MPLR	MLUM
Ex 1	0.2062	0.2077	**0.2019**	0.2123	0.2057	0.2113	0.2021
Ex 2	0.1223	0.1475	0.1361	0.1246	0.1190	0.2377	**0.1147**
Ex 3	0.0716	0.0716	0.0719	0.0717	0.0715	0.0706	**0.0701**
Ex 4	0.2411	0.2439	0.2338	0.2410	0.2411	0.1419	**0.1335**
Ex 5	0.3519	0.3568	0.3638	0.3697	0.3329	0.3719	**0.3145**
Ex 6	0.4528	0.4237	0.4497	0.4611	0.4323	0.5321	**0.4134**

Table 3.7: The classification error rates for all simulated examples, for MSVMs with Losses 1-4 defined in Section 3.2.2, the RMSVM, the tuned MPLR and the tuned MLUM.

3.6 Real Data Examples

This section empirically tests the performance of the MLUM family, on two types of datasets. The first type of data examples are several benchmark datasets available on the UCI machine learning website. The second one is a recent microarray gene expression dataset in cancer research. For all the real datasets, we apply the MLUM family in the same way as in the simulation part, and we repeat the procedures 1000 times to evaluate the performance.

3.6.1 Benchmark Data

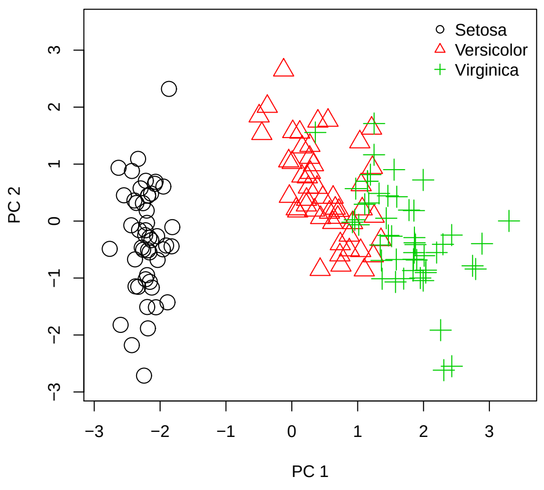
We consider seven benchmark datasets in this section: Breast Tissue (Breast), Dermatology, Image Segmentation (Image), Iris, Vehicle Silhouettes (Vehicle), Vertebral Column (Vertebral) and Wine. In each case, we perform a four-fold cross validation on roughly 4/5 of the dataset, and test the performance on the remaining 1/5. For Iris and Image data, we apply linear learning. For Vehicle, we apply the second order polynomial kernel learning. The Gaussian kernel is used for the Breast, Dermatology, Wine and Vertebral datasets, with the σ parameter value being the median of all pairwise distances between one category and the other. For all datasets, we standardize the attributes before further analysis.

For illustration, we summarize the datasets in Table 3.8. As we observed in the simulation studies that potential outliers may decrease the accuracy of soft classifiers, we perform Principal Component Analysis (PCA) on each dataset and examine if there are any obvious potential outliers on the PCA projected plots. From the results of the real examples, we observe that soft classifiers tend to have relatively worse performance on datasets with potential outliers. For demonstration, we report the PCA plots for the Iris and Vehicle data in Figure 3.12. In the Iris data, there seems to be no significant outliers and the soft classifier works very well. In the Vehicle data, there appears to be several potential outliers, and the soft classifier turns out to be the worst in terms of classification accuracy.

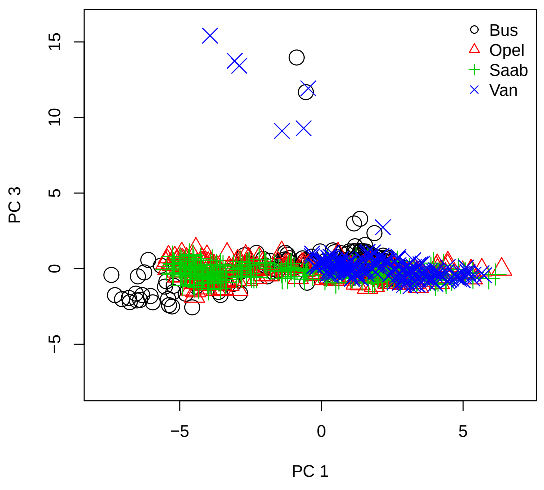
Breast: The dataset contains 10 variables and 22, 21, 14, 15, 16 and 18 instances for the following

Name	n	d	k	Kernel	Outlier	Best c
Breast	106	10	6	Gaussian	Yes	1000
Dermatology	366	34	6	Gaussian	No	1000
Image	210	19	7	Linear	Yes	1
Iris	150	4	4	Linear	No	100
Vehicle	946	18	4	2^{nd} poly.	Yes	1000
Vertebral	310	6	3	Gaussian	Yes	1
Wine	178	13	3	Gaussian	No	0

Table 3.8: Summary of the benchmark datasets in Section 3.6.1.



(a) PCA plot for the Iris data, PC1 vs PC2.

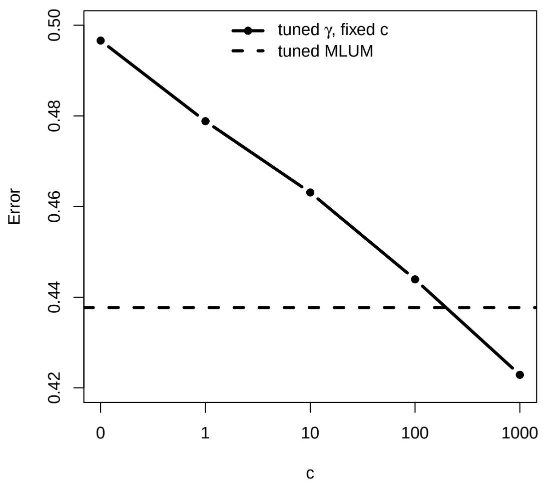


(b) PCA plot for the Vehicle data, PC1 vs PC3.

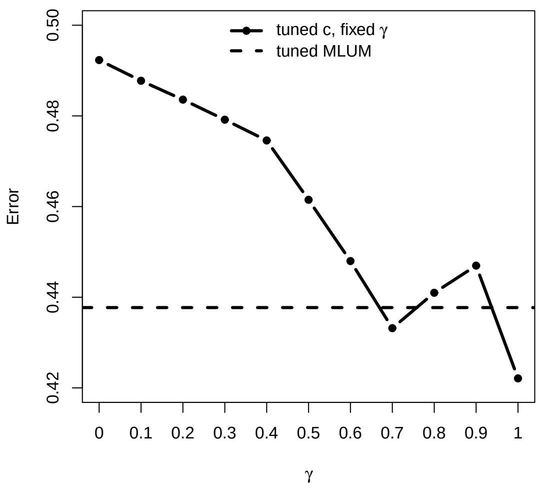
Figure 3.12: PCA projection plots for Iris (left panel) and Vehicle (right panel) data. The plots indicate the Iris data appear to be quite clean, while there may exist some outliers for the Vehicle data.

6 breast diseases respectively: carcinoma, fibro-adenoma, mastopathy, glandular, connective, and adipose. The classification errors in Figure 3.13 suggest that hard classification works better, and $\gamma = 1$ would be the best choice when c is tuned. The tuned MLUM performs slightly worse than the RMSVM, but the difference is not large.

Dermatology: There are 6 classes in the Dermatology dataset, namely, psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis and pityriasis rubra pilaris, with 112, 61, 72, 49, 52 and 20 instances respectively. There are 34 covariates measured for each observation. The classification results are reported in Figure 3.14. With $c \to \infty$ the classification accuracy is the best. In addition, $\gamma = 0.2$ performs better than other values.

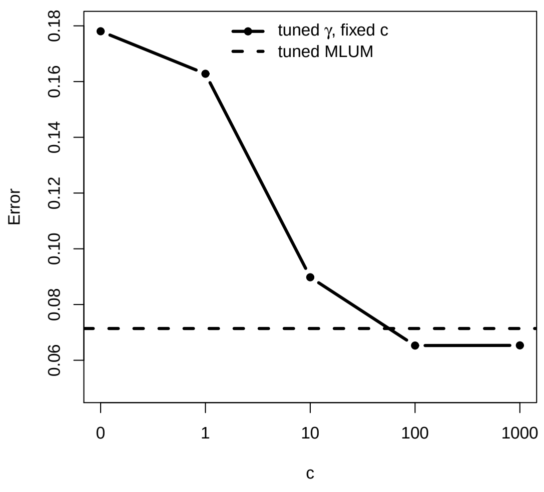


(a) The test error of Breast data with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

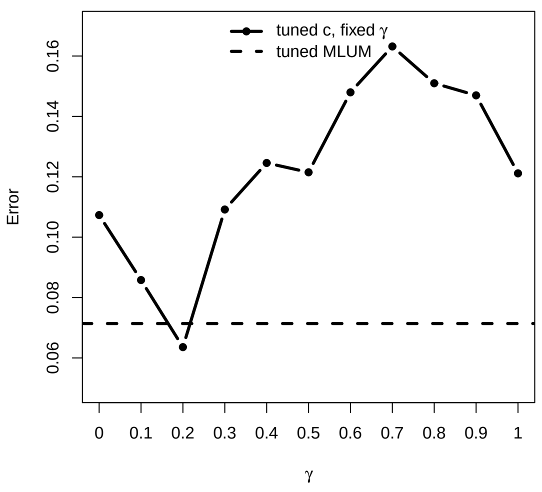


(b) The test error of Breast data with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.13: Classification error rates in the Breast data. The hard classification method dominates the others in terms of classification accuracy, as shown in the left panel. The right panel suggests that the classification error is the best when c is tuned with $\gamma = 1$.



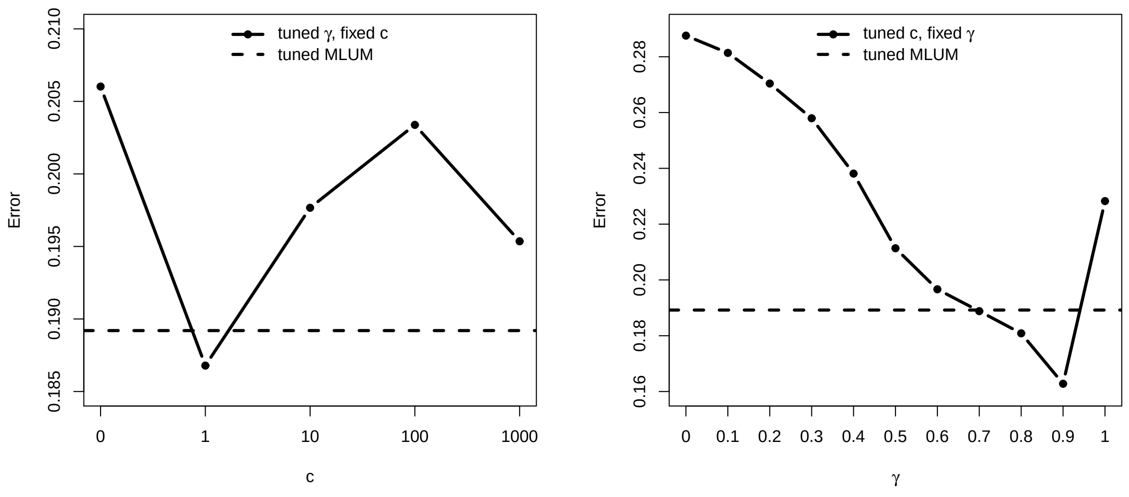
(a) The test error of Dermatology data with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.



(b) The test error of Dermatology data with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.14: Classification error rates in the Dermatology data. The hard classifier works the best, as is shown in the left panel. The right panel shows that when c is tuned, $\gamma = 0.2$ works better than the others.

Image: In the Image dataset, there are 7 classes available with 30 observations in each group. The class names are brickface, cement, foliage, grass, path, sky and window. There are 19 covariates in the dataset. We report the test errors in Figure 3.15. In this case $c = 1$ performs the best, and $\gamma = 0.9$ is the optimal choice when c is tuned.



(a) The test error of Image data with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

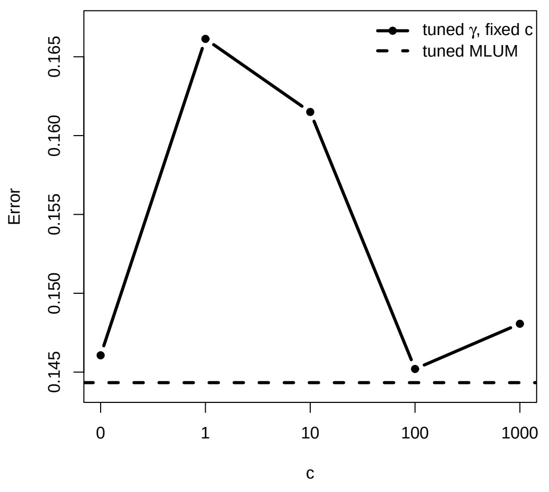
(b) The test error of Image data with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.15: Classification error rates in the Image data. Left panel shows that $c = 1$ performs the best on the Image data. The right panel shows $\gamma = 0.9$ dominates other values.

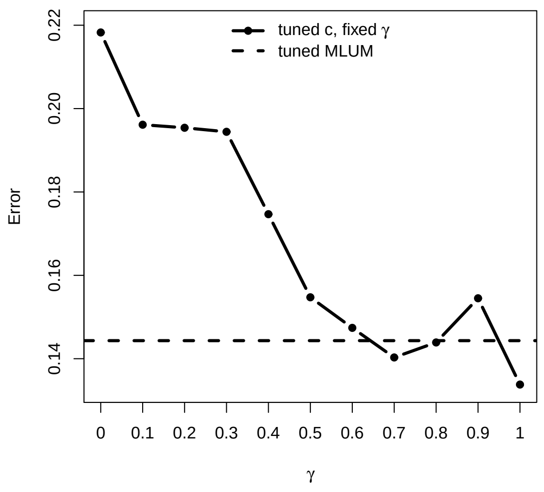
Iris: There are three classes in the Iris dataset, namely, Iris-setosa, Iris-versicolor and Iris-virginica. Within each class the number of observations is 50. There are four covariates. The test errors are reported in Figure 3.16. In this example, the DWD ($c = 1$) performs the worst among all classifiers, in contrast to the simulated Example 1 where the DWD dominates. Interestingly, hard and soft classifiers perform similarly in terms of classification accuracy. For the effect of γ, the MLUM family with $\gamma = 1$ appears to work the best.

Vehicle: The Vehicle dataset contains 4 classes of observations, namely, bus, opel, saab and van. There are $218, 212, 217, 199$ instances in the four groups, respectively. There are 18 variables associated with the data. The test errors in Figure 3.17 show that the hard classifier performs better than the other methods, and when c is automatically chosen, $\gamma = 0.8$ dominates other values.

Vertebral: The three classes involved in the Vertebral dataset are normal, disk hernia and spondilolysthesis. The numbers of instances are 100, 60 and 150, respectively. Six covariates are

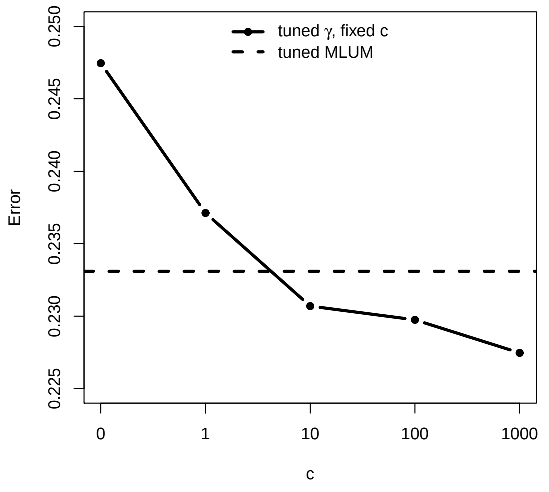


(a) The test error of Iris data with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

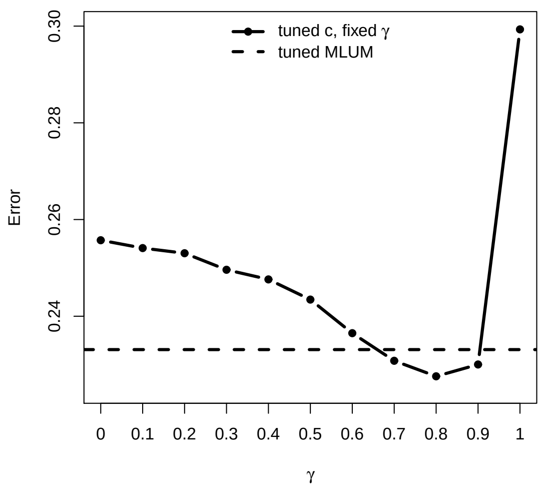


(b) The test error of Iris data with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.16: Classification error rates in the Iris data. The left panel shows the test error of soft and hard classifiers are roughly comparable, while $c = 1$ (DWD) is the worst. The right panel indicates $\gamma = 1$ is the optimal choice, if c is tuned.



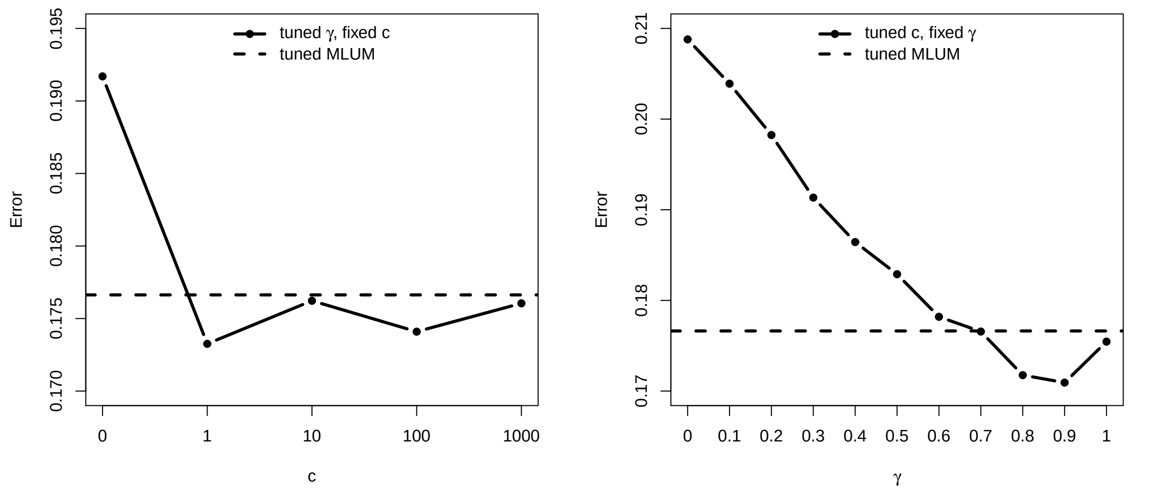
(a) The test error of Vehicle data with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.



(b) The test error of Vehicle data with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.17: Classification error rates in the Vehicle data. The hard classifier works better than the other ones, as is shown in the left panel. The right panel suggests the best γ is 0.8. Note that the test error significantly increases when γ moves from 0.9 to 1.

present in the dataset. Figure 3.18 illustrates the MLUM performs quite consistently for $c \geq 1$, with the best classifier being the MLUM with $c = 1$. In terms of the behavior of γ, the optimal choice is $\gamma = 0.9$.



(a) The test error of Vertebral data with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

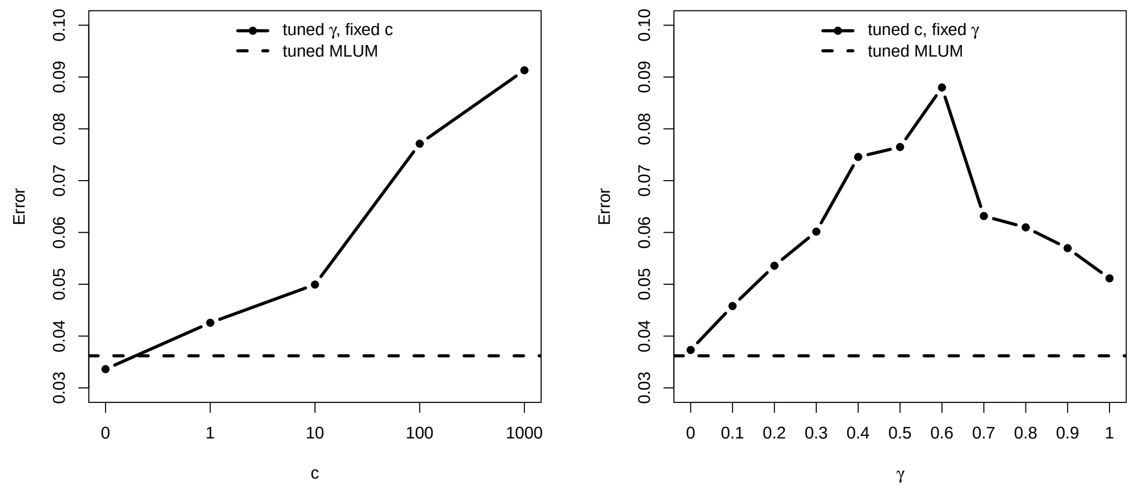
(b) The test error of Vertebral data with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.18: Classification error rates in the Vertebral data. In the left panel, we can see the MLUM classifiers work roughly the same for $c \geq 1$, with $c = 1$ being the optimal. The best γ is 0.9, as is suggested by the right panel.

Wine: The wine dataset measures 3 classes (59, 71 and 48 each) of wine with 13 attributes. In Figure 3.19, we see that soft classifier has the most accurate prediction result, and $\gamma = 0$ is the best choice when c is tuned.

3.6.2 Glioblastoma Multiforme Cancer Data

In this section we apply the MLUM family to a cancer dataset and explore its performance. The data was used by Verhaak et al. [2010]. They investigate Glioblastoma Multiforme (GBM) cancer on patients from The Cancer Genome Atlas Research Network. There are 356 observations in the dataset, which consist four classes, namely, Proneural, Neural, Classical and Mesenchymal. The numbers of patients in each class are 97, 56, 92 and 111, respectively. There are 23285 genes in the dataset. In each replication, the numbers of observations in the train, tune and test data are chosen to be roughly the same. To reduce computational cost, for each replication we pick 500 genes which have the largest



(a) The test error of Wine data with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

(b) The test error of Wine data with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.19: Classification error rates in the Wine data. The left panel suggests that soft classification method performs better in terms of classification accuracy than the others. With $\gamma = 0$, the MLUM method works the best than the other γ values, which is shown in the right panel.

median absolute deviation values in the training dataset.

To visualize the data, we plot the first two projected principal component directions of the data in Figure 3.20. From Figure 3.20, we can see the classes Proneural and Mesenchymal are relatively far from each other, and Classical and Neural are in between. Biologically, Neural and Proneural are more similar to each other than the other classes. We report the MLUM results in Figure 3.21. From the plots, we can see that the MLUM with $\gamma = 0.75$ performs uniformly the best in terms of classification accuracy, and hard classifiers dominate the rest.

Lastly, we report the comparison of the tuned MLUM versus other methods in Table 3.9. Overall, the tuned MLUM performs the best, as in the simulated examples.

3.7 Summary and Possible Future Work

Multicategory classification is commonly seen in practice. In this chapter, we generalize the binary LUM family to the simultaneous multicategory classifiers, namely, the MLUM family. The MLUM is very general and it includes many popular multicategory classification methods as special cases. In

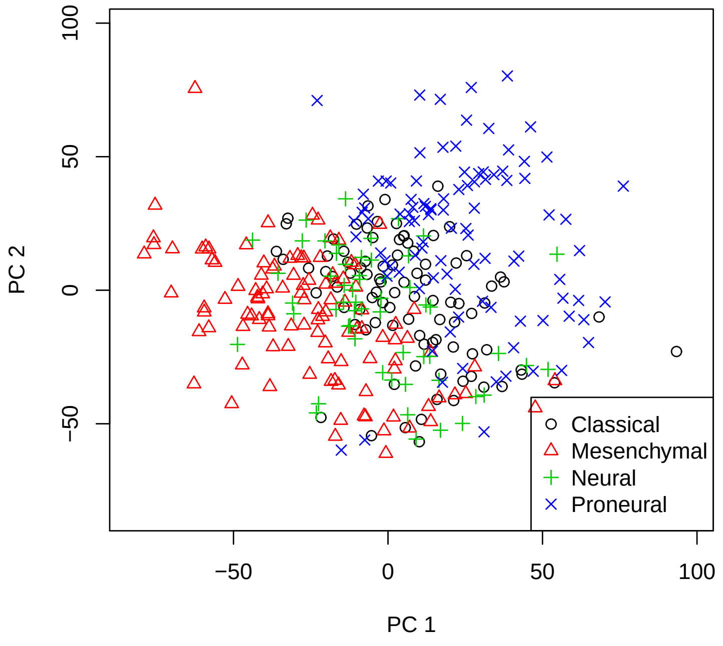
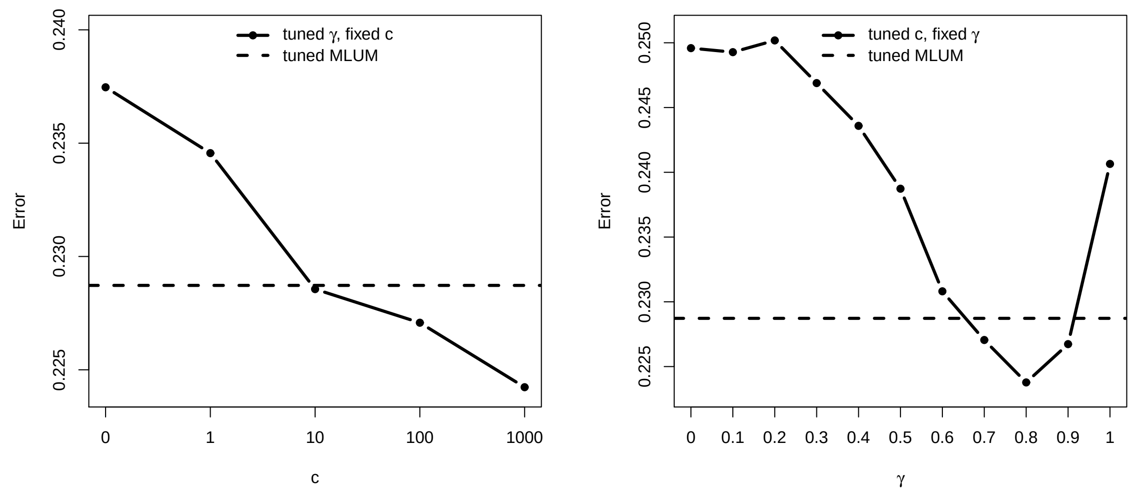


Figure 3.20: PCA plot of the GBM data, PC 1 vs PC 2.



(a) The test error of GBM data with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

(b) The test error of GBM data with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 3.21: For the GBM data, hard classifier works the best, as is shown in the left panel. The right panel illustrates that the optimal choice of γ is 0.8.

	MSVM 1	MSVM 2	MSVM 3	MSVM 4	RMSVM	MPLR	MLUM
Breast	0.4497	0.4531	0.4391	0.4623	**0.4331**	0.4951	0.4377
Dermatology	0.0907	0.1496	0.1053	0.1398	0.1048	0.1688	**0.0713**
Image	0.1985	0.2083	0.1999	0.2255	0.2100	0.1978	**0.1892**
Iris	0.1557	0.1647	0.1487	0.1480	0.1605	0.1498	**0.1443**
Vehicle	0.2519	0.2535	0.2489	0.2598	0.2419	0.2680	**0.2331**
Vertebral	**0.1741**	0.1931	0.1860	0.1951	0.1856	0.3359	0.1766
Wine	0.0437	0.0743	0.0507	0.0670	0.0891	0.0348	**0.0361**
GBM	0.2291	0.2478	0.2351	0.2478	0.2394	0.2340	**0.2287**

Table 3.9: The classification error rates for all real data examples, for MSVMs with Losses 1-4 defined in Section 3.2.2, the RMSVM, the tuned MPLR and the tuned MLUM.

particular, the MLUM family includes both soft and hard classifiers, and provides a platform to explore the transition behavior from soft to hard classification problems. In theoretical studies, we show that the MLUM is always Fisher consistent for any finite c, and can provide class conditional probability estimation. We explore some asymptotic behavior of the MLUM family, and demonstrate that the convergence rate of the excess $V-$risk is closely related to the convergence rate of the classification function $\hat{\boldsymbol{f}}$, as well as the size of the function class space. The numerical examples show hard and soft classifiers behave quite differently in various settings, and they help to shed some light on the choice between the two. In particular, the numerical results suggest that the underlying probability function and potential outliers may have a significant effect on the performance of soft, hard and in-between classifiers. Furthermore, for practical applications, we propose an automatically tuning procedure for the MLUM. We numerically demonstrate that the tuned MLUM outperforms several other multicategory techniques and should be a competitive addition to the existing classification toolbox.

We have looked at the effect of shrinkage on the estimation of class conditional probability, in the binary case. In the multicategory case, we believe a similar pattern exists. An open problem is how to correct the bias in probability estimation when we perform multicategory classifications. Future exploration will be pursued.

3.8 Proofs

Proof of Proposition 3.1: For brevity we only prove the kernel learning case, and the proof for the linear learning is analogous. By representer theorem, we have that $g_{j,\mathcal{H}}$ can be written as $g_{j,\mathcal{H}}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_{i,j} K(\boldsymbol{x}_i, \boldsymbol{x})$, where $K(\cdot,\cdot)$ is the associated reproducing kernel, and $\boldsymbol{x}_i; i = 1, \dots, n$

are the training data points. With the kernel learning, the penalty becomes $\frac{1}{2}\sum_{j=1}^{k}\boldsymbol{\alpha}_j^T K\boldsymbol{\alpha}_j$, where $\boldsymbol{\alpha}_j = (\alpha_{1,j}, \ldots, \alpha_{n,j})^T; j = 1, \ldots, k$, and with a little abuse of notation, K is the gram matrix with the $(i,j)^t h$ element $K_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. One can verify that the sum-to-zero constraint here is equivalent to that $\sum_{j=1}^{k}\alpha_{i,j} = 0$, for all i.

Because the loss V depends on $\boldsymbol{f}$ only through $f_i - f_j$, it suffices to prove that for a given solution $\hat{\boldsymbol{\alpha}}_j = (\hat{\alpha}_{1,j}, \ldots, \hat{\alpha}_{n,j}); j = 1, \ldots, k$ such that $\sum_{j=1}^{k}\hat{\alpha}_{i,j} = 0$ for all i, $\sum_{j=1}^{k}\hat{\boldsymbol{\alpha}}_j^T K\hat{\boldsymbol{\alpha}}_j < \sum_{j=1}^{k}(\hat{\boldsymbol{\alpha}}_j - \boldsymbol{z})^T K(\hat{\boldsymbol{\alpha}}_j - \boldsymbol{z})$. Here $\boldsymbol{z}$ is any fixed vector of length n that is not $\mathbf{0}$. To this end, we see that $\sum_{j=1}^{k}\hat{\boldsymbol{\alpha}}_j^T K\hat{\boldsymbol{\alpha}}_j - \sum_{j=1}^{k}(\hat{\boldsymbol{\alpha}}_j - \boldsymbol{z})^T K(\hat{\boldsymbol{\alpha}}_j - \boldsymbol{z})$ can be simplified as $2(\hat{\boldsymbol{\alpha}}_1^T + \cdots + \hat{\boldsymbol{\alpha}}_k^T)K\boldsymbol{z} - k\boldsymbol{z}^T K\boldsymbol{z}$. Note that $\hat{\boldsymbol{\alpha}}_1^T + \cdots + \hat{\boldsymbol{\alpha}}_k^T = \mathbf{0}$, and K is positive definite, so that $2(\hat{\boldsymbol{\alpha}}_1^T + \cdots + \hat{\boldsymbol{\alpha}}_k^T)K\boldsymbol{z} - k\boldsymbol{z}^T K\boldsymbol{z} < 0$, and this completes the proof. $\square$

Proof of Lemma 3.3.1: We prove by contradiction. Without loss of generality, assume that $P_1 > P_2$, and $f_1^* < f_2^*$. Now let $\boldsymbol{f} = (f_2^*, f_1^*, f_3^*, \ldots, f_k^*)$, and we want to show that $S(\boldsymbol{f}, \boldsymbol{x}) < S(\boldsymbol{f}^*, \boldsymbol{x})$. One can verify that $S(\boldsymbol{f}, \boldsymbol{x}) - S(\boldsymbol{f}^*, \boldsymbol{x}) = (P_2 - P_1)\big(\gamma\delta^+ + (1-\gamma)\delta^-\big)$, where $\delta^+ = \ell(f_1^*) - \ell(f_2^*)$ and $\delta^- = \ell(-f_2^*) - \ell(-f_1^*)$. Since the ℓ function is monotonely decreasing, and for $u < 0$ the monotonicity is strict, both δ^+ and δ^- are non-negative, with at least one of them being non-zero. Thus $S(\boldsymbol{f}, \boldsymbol{x}) < S(\boldsymbol{f}^*, \boldsymbol{x})$, and $\boldsymbol{f}^*$ is not the minimizer of S. $\square$

Proof of Theorem 3.3.1: Assume, without loss of generality, that $P_1 > P_2 > \cdots > P_k$, and we need to show that $f_1^* > f_2^* \geq \cdots \geq f_k^*$. We prove by contradiction. Because of Lemma 3.3.1, $f_1^* \geq f_2^*$. Suppose $f_1^* = f_2^*$, and let $\boldsymbol{f} = (f_1^* + \epsilon, f_2^* - \epsilon, f_3^*, \ldots, f_k^*)$, where $\epsilon > 0$ is sufficiently small. We show that $S(\boldsymbol{f}, \boldsymbol{x}) < S(\boldsymbol{f}^*, \boldsymbol{x})$. For simplicity in notation, let $f = f_1^* = f_2^*$. After some calculation we have

$$\begin{aligned} S(\boldsymbol{f}, \boldsymbol{x}) - S(\boldsymbol{f}^*, \boldsymbol{x}) = (1-\gamma)[\delta_1 - \delta_2] + P_1\big[\gamma(-\delta_4) + (1-\gamma)(-\delta_1)\big] \\ +P_2\big[\gamma(\delta_3) + (1-\gamma)(\delta_2)\big] \end{aligned} \tag{3.11}$$

where

$$\begin{cases} \delta_1 = \ell(-f-\epsilon) - \ell(-f) \\ \delta_2 = \ell(-f) - \ell(-f+\epsilon) \\ \delta_3 = \ell(f-\epsilon) - \ell(f) \\ \delta_4 = \ell(f) - \ell(f+\epsilon). \end{cases}$$

Since $\ell(\cdot)$ is differentiable, we may rewrite the above equations as:

$$\begin{cases} \delta_1 = -\epsilon\ell'(-f) + o(\epsilon) \\ \delta_2 = -\epsilon\ell'(-f) + o(\epsilon) \\ \delta_3 = -\epsilon\ell'(f) + o(\epsilon) \\ \delta_4 = -\epsilon\ell'(f) + o(\epsilon). \end{cases}$$

and so (3.11) becomes $\epsilon(P_1 - P_2)[\gamma\ell'(f) + (1-\gamma)\ell'(-f)] + o(\epsilon)$. Because $\ell(\cdot)$ is convex and strictly monotonely decreasing, $\ell'(\cdot) < 0$, thus $S(\boldsymbol{f}, \boldsymbol{x}) - S(\boldsymbol{f}^*, \boldsymbol{x}) < 0$. $\square$

Remark 3.3. The reason why the hinge loss is not Fisher consistent when $\gamma > 1/2$ becomes more clear from the proof of Theorem 3.3.1. In the literature of the hinge loss, we should replace the derivatives with the corresponding left/right derivatives in the previous argument. When $f = 1$, $\delta_4 = 0$, $|P_1(1-\gamma)\delta_1|$ does not necessarily dominate $P_2\big[\gamma(\delta_3) + (1-\gamma)(\delta_2)\big]$ when $\gamma > 1/2$. By allowing differentiability in the loss function, we overcome the disadvantage of non-Fisher consistency.

Proof of Theorem 3.3.2: Due to the constraint $\sum_{j=1}^k f_j = 0$, the degrees of freedom for S in (3.7) is only $k-1$. Rewrite S as

$$S = (1-\gamma)\sum_{j=1}^k \ell(-f_j) + \sum_{j=1}^k P_j\big[\gamma\ell(f_j) - (1-\gamma)\ell(-f_j)\big].$$

Let the first $k-1$ components of $\boldsymbol{f}$ be the free parameters. Take partial derivative of S with respect to $f_1, \ldots, f_{k-1}$, and we have

$$0 = \frac{\partial S}{\partial f_j}|_{\boldsymbol{f}=\boldsymbol{f}^*} = -F^*(j) + F^*(k) + P_jE^*(j) - P_kE^*(k); j = 1, \ldots, k-1,$$

where E^* and F^* are defined in an obvious manner. Let $\boldsymbol{P}_{k-1} = (P_1, \ldots, P_{k-1})^T$, we can rewrite the above equations as $M\boldsymbol{P}_{k-1} - \boldsymbol{b} = \boldsymbol{0}_{k-1}$, or $M\boldsymbol{P}_{k-1} = \boldsymbol{b}$ where $\boldsymbol{b} = \big([(1-\gamma)\ell'(-f_1^*) + \gamma\ell'(f_k^*)], \ldots, [(1-\gamma)\ell'(-f_{k-1}^*) + \gamma\ell'(f_k^*)]\big)^T$ and $M = diag(E^*(1), \ldots, E^*(k-1)) + E^*(k)J_{k-1}$. Here J_{k-1} is the $k-1$ by $k-1$ matrix with every element being 1. Since $E(\cdot) < 0$, $1 + E^*(k)\sum_{j=1}^{k-1} E^*(j) > 0$, an application of the Sherman-Morrison formula (which guarantees the invertibility of M) gives $\boldsymbol{P}_{k-1} = M^{-1}\boldsymbol{b}$. Let $P_k = 1 - \sum_{j=1}^{k-1} P_j$. The equations (3.8) follow after some simplification and substituting $\hat{\boldsymbol{f}}$ for $\boldsymbol{f}^*$. Note that from the form of (3.8), we conclude the choice of the free parameters in $\boldsymbol{f}$ is not essential.

□

Proof of Theorem 3.3.3: By definition,

$$\Delta Q(\boldsymbol{P}, \boldsymbol{f}) = Q(\boldsymbol{P}, \boldsymbol{f}) - Q^*(\boldsymbol{P}).$$

We can rewrite the RHS of the display as

$$\begin{aligned}
&(1-\gamma)\sum_{j=1}^{k}\ell(-f_j) + \sum_{j=1}^{k} P_j[\gamma\ell(f_j) - (1-\gamma)\ell(-f_j)] \\
&-(1-\gamma)\sum_{j=1}^{k}\ell(-f_j^*) - \sum_{j=1}^{k} P_j[\gamma\ell(f_j^*) - (1-\gamma)\ell(-f_j^*)].
\end{aligned}$$

With adding and subtracting, rearrange to obtain that the above display is equivalent to

$$\begin{aligned}
(1-\gamma)\sum_{j=1}^{k}[\ell(-f_j) - \ell(-f_j^*) - \ell'(-f_j^*)(-f_j + f_j^*)] + (1-\gamma)\sum_{j=1}^{k}\ell'(-f_j^*)(-f_j + f_j^*) \\
+\sum_{j=1}^{k} P_j[\gamma\{\ell(f_j) - \ell(f_j^*) - \ell'(f_j^*)(f_j - f_j^*)\} \\
-(1-\gamma)\{\ell(-f_j) - \ell(-f_j^*) - \ell'(-f_j^*)(-f_j + f_j^*)\}] \\
+\sum_{j=1}^{k} P_j[\gamma\ell'(f_j^*)(f_j - f_j^*) - (1-\gamma)\ell(-f_j^*)(-f_j + f_j^*)].
\end{aligned}$$

In combination with the Bregman's divergence, observe that the above is essentially RHS of Theorem 3.3.3 plus $(1-\gamma)\sum_{j=1}^{k}\ell'(-f_j^*)(-f_j + f_j^*) + \sum_{j=1}^{k} P_j[\gamma\ell'(f_j^*)(f_j - f_j^*) - (1-\gamma)\ell(-f_j^*)(-f_j + f_j^*)]$, so it suffices to show the latter, denoted by W, equals 0.

Using the same notation as in Theorem3.3.2, we have

$$\begin{aligned}
W &= \sum_{j=1}^{k}[P_j\gamma\ell'(f_j^*)(f_j - f_j^*) + (1-P_j)(1-\gamma)\ell'(-f_j^*)(-f_j + f_j^*)] \\
&= \sum_{j=1}^{k}[-F^*(j) + P_jE^*(j)](f_j - f_j^*)
\end{aligned}$$

Because of the sum-to-zero constraint, the above display is equivalent to

$$W = \sum_{j=1}^{k-1} [-F^*(j) + F^*(k) + P_j E^*(j) - P_k E^*(k)](f_j - f_j^*)$$

Note that $-F^*(j) + F^*(k) + P_j E^*(j) - P_k E^*(k)$ is just the j^{th} element of $\bigtriangledown Q^*(\boldsymbol{P})|_{\boldsymbol{f}=\boldsymbol{f}^*}$, choosing the first $k-1$ elements in $\boldsymbol{f}$ as free parameters and taking partial derivatives. Thus $W = 0$, and the desired result follows. $\square$

Proof of Theorem 3.3.4: Expand the Bregman divergence in Theorem 3.3.3 into the Taylor series form:

$$\Delta Q(\boldsymbol{P}, \boldsymbol{f}) = \sum_{i=1}^{k} \big[P_i \gamma [\frac{\ell^{(2)}(f_i^*)}{2} (\hat{f}_i - f_i^*)^2 + o((\hat{f}_i - f_i^*)^2)]$$
$$+(1 - P_i)(1 - \gamma)[\frac{\ell^{(2)}(-f_i^*)}{2} (\hat{f}_i - f_i^*)^2 + o((\hat{f}_i - f_i^*)^2)]\big].$$

Note that the second order derivative of ℓ is bounded for every a and finite c. Because $\tau_i \ll \mu, i = 1, \ldots, k$, we can multiply both sides by n^{2q}, and take expectation to obtain

$$n^{2q} \Delta Q(\hat{\boldsymbol{f}}^n) = O(\int_{\boldsymbol{X},Y} | \sup_{1 \leq j \leq k} T_j|^2 d\mathbb{P}(\boldsymbol{X}, Y)).$$

Because of Assumption B, the RHS is bounded, and the desired result follows. $\square$

Proof of Theorem 3.3.5: Note that $(\hat{f}_i - f_i^*)^2 - (f_i^H - f_i^*)^2 = (\hat{f}_i - f_i^H)(\hat{f}_i + f_i^H - 2f_i^*)$. $(\hat{f}_i - f_i^H)$ is n^q consistent, and $|\hat{f}_i + f_i^H - 2f_i^*| \leq |\hat{f}_i - f_i^H| + 2(|f_i^H| + |f_i^*|) \to 2(|f_i^H| + |f_i^*|)$. The rest of the proof is analogous to that of Theorem 3.3.4. $\square$

CHAPTER 4: MULTICLASS ANGLE-BASED CLASSIFIER

4.1 Introduction

The existing simultaneous multicategory classifiers with the sum-to-zero constraint have some drawbacks. For instance, in the literature of SVMs, the geometric interpretation of maximum separation in the binary case is well understood, while the extension to multicategory SVMs does not give such a clear perspective. Furthermore, the sum-to-zero constraint makes the associated optimization problem more complex. In this chapter, we propose a new Multicategory Angle-based large margin Classification (MAC) technique that not only provides a natural generalization of binary large margin methods, but also overcomes the disadvantages of the existing simultaneous methods mentioned above. In particular, we consider a standard simplex structure in an Euclidean space, whose dimension is one less than the number of categories [Hill and Doucet, 2007; Lange and Wu, 2008]. The details of a standard simplex are introduced in Section 4.2. Note that the number of vertices of such a simplex is equal to the number of categories. We then associate each class to one vertex, and the classification function vector maps the covariates of a given instance to the Euclidean space that the simplex lies in. Note that unlike the regular simultaneous multicategory classification techniques, here the length of the classification function vector is the same as the dimensionality of the Euclidean space. The prediction rule is then to assign the given instance to the category whose corresponding vertex has the smallest angle with respect to the classification function vector.

Compared to the regular simultaneous multicategory classification methods, the angle-based method has several attractive properties. First, the geometric interpretation of the lease angle prediction rule is very easy to understand through our newly defined classification regions. See Fig. 4.1. The corresponding functional margin can be directly generalized from the binary to the multicategory case. The minimizers of the new multicategory large-margin unified machines using the angle-based structure help us to better understand the transition behaviors from soft to hard classifiers [Wahba, 2002; Liu, Zhang and Wu, 2011]. Second, our angle-based method enjoys a lower computational cost than the regular procedure with the sum-to-zero constraint, as we implicitly transfer the constraint onto our newly defined functional margins. Furthermore, we show that the angle-based method has many desirable theoretical properties. In particular, we provide some theoretical insights on the advantages of our angle-based method in Section 4.3.5. Despite the sum-to-zero constraint, using k classification functions instead of $k-1$ can introduce extra variability in the estimated classifier. This can deteriorate the classification performance of the regular simultaneous classifiers. In linear learning problems with high dimensional predictors, this extra variability can be large. Consequently, the angle-based methods can significantly outperform the regular simultaneous methods. We confirm this insight numerically by comparing our angle-based methods with some existing simultaneous multicategory classifiers.

The rest of the chapter is organized as follows. In Section 4.2, we introduce our proposed MAC method. In Section 4.3, we derive some statistical properties for MAC. We discuss how to implement the optimization problem associated with the MAC technique, as well as the tuning procedure, in Section 4.4. We perform simulation studies in Section 4.5, and some real data analysis in Section 4.6. Some discussions are provided in Section 4.7. All technical proofs are collected in the appendix.

4.2 Methodology

Despite the importance of binary classifiers, multicategory problems are prevalent in practice. In that case, it is desirable to extend binary large margin classifiers to obtain the multicategory versions.

For multicategory problems, where $Y \in \{1, \ldots, k\}$ and $k > 2$ is the number of classes, the regular simulteneous procedure is to map $\boldsymbol{x}$ into $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^k$, and the corresponding prediction rule is $\hat{y} = \operatorname{argmax}_j \hat{f}_j(\boldsymbol{x})$. As mentioned above, the geometric interpretation of such a prediction rule is vague. Furthermore, if we have no constraint on $\boldsymbol{f}$ other than $J(\boldsymbol{f})$, then adding a constant to each element of $\boldsymbol{f}$ does not change the prediction result. Hence, the classification function vector $\boldsymbol{f}$ can be shifted by any constant without a change in the prediction, and it may result in non-unique solutions. If we assume the elements in $\boldsymbol{f}$ sum to zero, the uniqueness is guaranteed, and many statistical properties such as the Fisher consistency can be derived. This constraint is used in many papers, for example, Lee, Lin and Wahba [2004], Tang and Zhang [2006], Wang and Shen [2007], Zhu et al. [2009] and Liu and Yuan [2011]. However, this extra constraint makes the corresponding optimization problem more challenging.

The proposed MAC method we introduce next overcomes the difficulties mentioned above. We give some clear geometric explanations on how MAC works for multicategory classification problems. Since MAC does not involve the sum-to-zero constraint explicitly, the corresponding computational speed of MAC can be significantly faster than the regular methods.

To begin with, we define the standard simplex in a $k-1$ dimensional space, as proposed in Lange and Wu [2008]. In our notation, a standard simplex means that the distance between any vertex and the central point of the simplex is 1. For example, when $k=3$, a simplex is an equilateral triangle in $\mathbb{R}^2$, and when $k=4$, it is a regular tetrahedron in $\mathbb{R}^3$. For multicategory classification, let $\mathcal{Y}$ be the collection of k vectors in $\mathbb{R}^{k-1}$ with each element

$$\mathcal{Y}_j = \begin{cases} (k-1)^{-1/2}\mathbf{1}, & \text{if } j=1, \\ -\frac{1+\sqrt{k}}{(k-1)^{3/2}}\mathbf{1} + \sqrt{\frac{k}{k-1}}e_{j-1}, & \text{if } 2 \le j \le k, \end{cases}$$

where e_j is a vector in $\mathbb{R}^{k-1}$ such that its every element is 0, except the j-th one is 1. One can check that $\mathcal{Y} = \{\mathcal{Y}_j;\ j=1,\ldots,k\}$ forms a standard simplex with k vertices in the $k-1$ dimension space. The center of $\mathcal{Y}$ is at the origin, and each $\mathcal{Y}_j$ has Euclidean norm 1.

The set $\{\mathcal{Y}_j;\ j=1,\ldots,k\}$ defines k directions in $\mathbb{R}^{k-1}$, and the angles between any two directions are equal. Every vector in $\mathbb{R}^{k-1}$ generates k different angles with respect to $\{\mathcal{Y}_j;\ j=1,\ldots,k\}$. Here we assume the angles are in $[0,\pi]$. Without loss of generality, we let $\mathcal{Y}_j$ represent class j. Our method is to map $\boldsymbol{x}$ to $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^{k-1}$, and make a prediction to the class whose corresponding angle is the smallest. In particular, we let $\hat{y} = \text{argmin}_j \angle(\mathcal{Y}_j, \hat{\boldsymbol{f}})$, where $\angle(\cdot,\cdot)$ denotes the angle between two vectors. Based on the prediction rule, we can split the space $\mathbb{R}^{k-1}$ into k disjoint "classification regions", whose formal definition is given below, together with a boundary set.

Definition: A Classification Region (CR) with respect to class j, CR_j; $j \in \{1,\ldots,k\}$, is the subset of $\mathbb{R}^{k-1}$, such that if $\hat{\boldsymbol{f}} \in \text{CR}_j$ then $\hat{y} = j$.

The CRs are closely connected to the prediction rule. In Figure 4.1 we plot the CRs for MAC with $k=3$. Note that for a given vector $\boldsymbol{f} \in \mathbb{R}^{k-1}$, $\angle(\mathcal{Y}_j, \boldsymbol{f})$ is the smallest if and only if the projection of $\boldsymbol{f}$ on $\mathcal{Y}_j$ is the largest for $j=1,\ldots,k$. In other words, $\angle(\mathcal{Y}_j, \boldsymbol{f})$ being the smallest is equivalent to $\langle \boldsymbol{f}, \mathcal{Y}_j \rangle > \langle \boldsymbol{f}, \mathcal{Y}_l \rangle$, for $l \neq j$, where $\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle = \boldsymbol{x}_1^T \boldsymbol{x}_2$ is the inner product of vectors $\boldsymbol{x}_1, \boldsymbol{x}_2$, and $\boldsymbol{x}^T$ is the transpose of $\boldsymbol{x}$. Moreover, note that the smaller $\angle(\mathcal{Y}_j, \boldsymbol{f})$ is, the larger $\langle \boldsymbol{f}, \mathcal{Y}_j \rangle$ is. Therefore, for any binary large margin loss function $\ell(\cdot)$, we propose the MAC classifier

$$\min_{\boldsymbol{f} \in F} \frac{1}{n} \sum_{i=1}^{n} \ell(\langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i} \rangle) + \lambda J(\boldsymbol{f}). \tag{4.1}$$

By duality, (4.1) is equivalent to the following optimization problem

$$\min_{\boldsymbol{f} \in F} \frac{1}{n} \sum_{i=1}^{n} \ell(\langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i} \rangle), \text{ subject to } J(\boldsymbol{f}) \leq s. \tag{4.2}$$

Our prediction rule is now equivalent to $\hat{y} = \text{argmax}_j \langle \hat{\boldsymbol{f}}(\boldsymbol{x}), \mathcal{Y}_j \rangle$. The value $\langle \boldsymbol{f}(\boldsymbol{x}), \mathcal{Y}_y \rangle$, which is equivalent to the projected length of $\boldsymbol{f}(\boldsymbol{x})$ on $\mathcal{Y}_y$, can be viewed as a generalized functional margin of $(\boldsymbol{x}, y)$, and $\ell(\langle \boldsymbol{f}(\boldsymbol{x}), \mathcal{Y}_y \rangle)$ measures the loss of assigning the label y to $\boldsymbol{x}$, in terms of the inner product between $\boldsymbol{f}(\boldsymbol{x})$ and $\mathcal{Y}_y$. From this perspective, the loss function in (4.1) and (4.2) encourages $\angle(\mathcal{Y}_{y_i}, \boldsymbol{f}(\boldsymbol{x}_i))$ to be as small as possible, or equivalently, it encourages the projection of $\boldsymbol{f}(\boldsymbol{x}_i)$ on $\mathcal{Y}_{y_i}$ to be as large as possible. Note that $\sum_{j=1}^{k} \langle \mathcal{Y}_j, \boldsymbol{f} \rangle = 0$, which means we implicitly transfer the sum-to-zero constraint on the classification function $\boldsymbol{f}$ in the regular simultaneous multicategory classification, to our new functional margins $\langle \mathcal{Y}_j, \boldsymbol{f} \rangle$. Hence, without an explicit constraint on the classification function $\boldsymbol{f}$, we reduce the computational burden of the optimization problem. As we will see in Sections 4.5 and 4.6, it is much more efficient to compute the MAC solution than the regular multicategory model with k functions.

We would like to point out that representation of the multicategory class label using the simplex structure was used in the literature previously. Some special cases with certain loss functions have been studied in the literature using the simplex class coding, such as the ϵ-insensitive loss Lange and Wu [2008]; Wu and Lange [2010]; Wu and Wu [2012], Boosting [Saberian and Vasconcelos, 2011] and SVMs [Hill and Doucet, 2007]. Our method is more general and covers general large margin classifiers. The methods proposed by Saberian and Vasconcelos [2011] and Hill and Doucet [2007] can be viewed as special cases of the MAC structure.

In the next section, to better understand the behavior of the MAC structure, we study some statistical properties of our proposed MAC method.

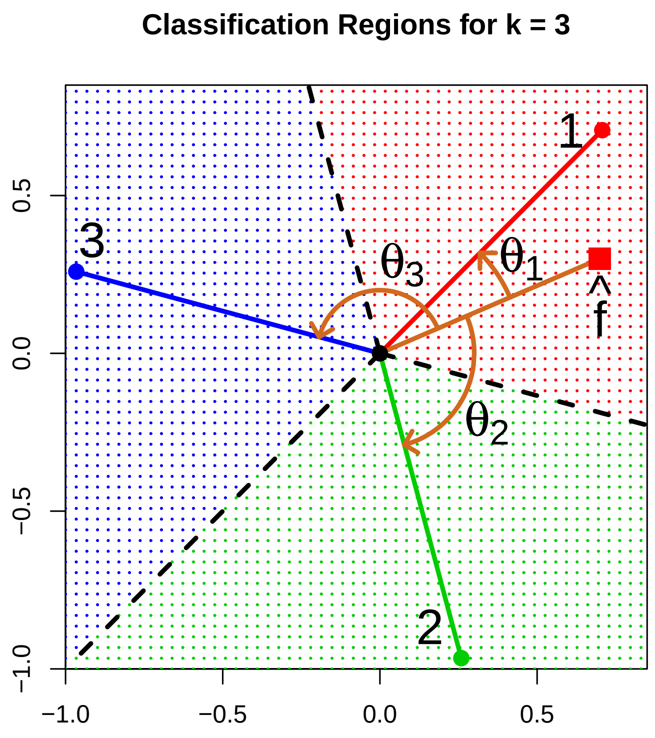


Figure 4.1: Classification regions for $k = 3$. The mapped observation $\hat{\boldsymbol{f}}$ is predicted as class 1 (red), because $\theta_1 < \theta_2 < \theta_3$. Dashed lines are the classification boundaries.

4.3 Statistical Properties

In this section, we explore several statistical properties of MAC. First, we study its Fisher consistency and theoretical minimizer in Section 4.3.1. Second, the class conditional probability estimation formula is derived in Section 4.3.2. Then we derive some asymptotic results for linear learning MAC with the Elastic Net penalty [Zou and Trevor, 2005] in Section 4.3.3. Lastly, we give a finite sample error bound in Section 4.3.4.

4.3.1 Fisher Consistency and Theoretical Minimizer

Fisher consistency is also known as classification calibration [Bartlett, Jordan and McAuliffe, 2006] or infinite sample consistency [Zhang, 2004a]. Intuitively, when the functional space $\mathcal{F}$ is large enough, and we have infinitely many samples, the resulting prediction rule of a Fisher consistent large margin classifier can yield the minimum misclassification rate. It is a fundamental requirement of a classification loss function.

For brevity, define the class conditional probabilities $P_j = P(Y = j|\boldsymbol{X} = \boldsymbol{x});\ j = 1, \ldots, k$. Under our angle-based prediction rule, Fisher consistency requires that for a given $\boldsymbol{x}$ such that $P_y > P_j;\ j \neq y$ for some $y \in \{1, \ldots, k\}$, the vector $\boldsymbol{f}^*(\boldsymbol{x})$ that minimizes $E[\ell(\langle f(\boldsymbol{X}), \mathcal{Y}_Y\rangle)|\boldsymbol{X} = \boldsymbol{x}]$ satisfies that $y = \operatorname{argmax}_j \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j\rangle$ and such an argmax is unique. Here we call $\boldsymbol{f}^*$ the theoretical minimizer.

Next we show that MAC is Fisher consistent if the loss ℓ has a negative derivative function.

Theorem 4.3.1. *The MAC loss ℓ in (4.1) and (4.2) is Fisher consistent if the derivative ℓ' exists and is negative.*

Note that among many others, the logistic loss, the exponential loss, and the LUM family with $c < \infty$ meet the condition of Theorem 4.3.1, and thus are all Fisher consistent.

For the SVM, the hinge loss is not differentiable and hence does not satisfy the condition of Theorem 4.3.1. As the LUM family includes the hinge loss as a special case ($c \to \infty$), we show that the multicategory angle-based SVM is not Fisher consistent, by studying the theoretical minimizer $\boldsymbol{f}^*$ of the entire LUM family. The next theorem gives the explicit expression of $\boldsymbol{f}^*$ in terms of $P_j;\ j = 1, \ldots, k$.

Theorem 4.3.2. *Consider the MAC with ℓ in the LUM family. Assume that $P_1 > P_2 > \cdots > P_{k-1} > P_k > 0$. The theoretical minimizer $\boldsymbol{f}^*$ is such that, for $a \in (0, +\infty)$ and $c \in [0, +\infty]$,*

$$\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle = \begin{cases} \frac{1}{1+c}[(\frac{P_j}{P_k})^{\frac{1}{a+1}}a - a + c] & \text{if } j \neq k, \\ -\sum_{u=1}^{k-1} \frac{1}{1+c}[(\frac{P_u}{P_k})^{\frac{1}{a+1}}a - a + c] & \text{if } j = k. \end{cases}$$

Hence $\boldsymbol{f}^* = \sum_{j=1}^{k-1} \frac{1}{1+c}\frac{k-1}{k}[(\frac{P_j}{P_k})^{\frac{1}{a+1}}a - a + c](\mathcal{Y}_j - \mathcal{Y}_k)$. For $a = \infty$ and $c \in [0, +\infty]$, we have that

$$\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle = \begin{cases} \frac{1}{1+c}[\log(\frac{P_j}{P_k}) + c] & \text{if } j \neq k, \\ -\sum_{u=1}^{k-1} \frac{1}{1+c}[\log(\frac{P_u}{P_k}) + c] & \text{if } j = k. \end{cases}$$

In this case, $\boldsymbol{f}^* = \sum_{j=1}^{k-1} \frac{1}{1+c}\frac{k-1}{k}[\log(\frac{P_j}{P_k}) + c](\mathcal{Y}_j - \mathcal{Y}_k)$.

When $c \to \infty$ with ℓ being the hinge loss, we see from Theorem 4.3.2 that the theoretical minimizer $\boldsymbol{f}^*$ satisfies $\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle = 1;\ j = 1, \ldots, k-1$ and $\langle \mathcal{Y}_k, \boldsymbol{f}^*\rangle = -k+1$. Therefore, the multicategory angle-based SVM is not Fisher consistent. Note that we assume $P_1 > P_2 > \cdots > P_{k-1} > P_k > 0$ in

Theorem 4.3.2 only for the sake of simplicity in expression. When P_i and P_{i+1} are the same, the theoretical minimizer is not unique. When $P_k = 0$, $\boldsymbol{f}^*$ is unbounded. We discuss more about this issue in Remark 2 of the appendix.

4.3.2 Class Conditional Probability Estimation

Recall the definition of class conditional probability $P_j;\ j = 1, \ldots, k$ in Section 4.3.1. In practice, after we build the classification model, it is very important and helpful to estimate P_j accordingly [Wang, Shen and Liu, 2008; Wu, Zhang and Liu, 2010]. In this section, we convert the theoretical minimizer $\boldsymbol{f}^*$ to estimate the probability P_j. In particular, after obtaining the solution to (4.1) or (4.2), $\hat{\boldsymbol{f}}$, one can apply the probability estimation formula to estimate $\hat{P}_j$ as in the next theorem.

Theorem 4.3.3. *In the MAC structure, suppose the loss function ℓ is differentiable. Given $\hat{\boldsymbol{f}}$, the estimated probabilities are $\hat{P}_j = \frac{\ell'(\langle \mathcal{Y}_j, \hat{\boldsymbol{f}} \rangle)^{-1}}{\sum_{i=1}^{k} \ell'(\langle \mathcal{Y}_i, \hat{\boldsymbol{f}} \rangle)^{-1}};\ j = 1, \ldots, k.$*

Note that if ℓ has a negative derivative ℓ', as in Theorem 4.3.1, one can verify that the estimated class conditional probability is proper, in the sense that $0 \leq \hat{P}_j \leq 1;\ j = 1, \ldots, k$ and $\sum_{j=1}^{k} \hat{P}_j = 1$.

In binary classification problems, it is known that SVM does not provide class conditional probability estimation [Lin, 2002], because the hinge loss is not differentiable at 1. Liu, Zhang and Wu [2011] showed that when $c \to \infty$, $\hat{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x})$, as a function of f^*, is a step function, and thus cannot provide any class conditional probability information.

In the multicategory case, the probability estimation becomes more involved because we have a classification function vector instead of a one dimensional classification function. Observe that the probability estimation formula in Theorem 4.3.3 depends on $\hat{\boldsymbol{f}}$ only through the derivative function ℓ'. Note that for $i \neq j$, $\hat{P}_i = \hat{P}_j$ if $\ell'(\langle \mathcal{Y}_i, \hat{\boldsymbol{f}} \rangle) = \ell'(\langle \mathcal{Y}_j, \hat{\boldsymbol{f}} \rangle)$. For the LUM loss, $\ell'(u) = -1$ for $u \leq \frac{c}{c+1}$. Thus, if $\hat{\boldsymbol{f}}$ satisfies that $\langle \mathcal{Y}_i, \hat{\boldsymbol{f}} \rangle \leq \frac{c}{c+1}$ and $\langle \mathcal{Y}_j, \hat{\boldsymbol{f}} \rangle \leq \frac{c}{c+1}$, the estimated conditional class probability for classes i and j are the same. As $c \to \infty$, $\frac{c}{c+1} \to 1$, and the estimated probabilities are all $1/k$ for $\hat{\boldsymbol{f}}$ with a norm small enough. When $\langle \mathcal{Y}_j, \hat{\boldsymbol{f}} \rangle$ is large and $\langle \mathcal{Y}_i, \hat{\boldsymbol{f}} \rangle$ is small for $i \neq j$, $\hat{P}_j = 1$ and $\hat{P}_i = 0$. We show this phenomenon in Figure 4.2 with $k = 3$. From Figure 4.2, we see that in the multicategory angle-based LUM case, as c increases, the probability function becomes closer to a step function and thus the accuracy of probability estimation deteriorates. In the limit, the multicategory SVM provides very limited probability information.

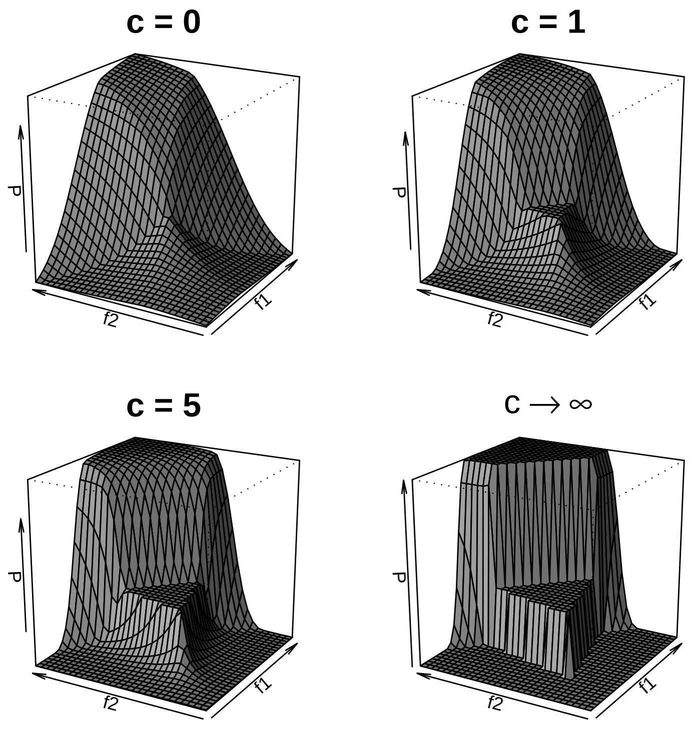


Figure 4.2: Visualization of the relationship between f_1^*, f_2^* and P_1, with $k = 3$, $a = 1$, and $c \in \{0, 1, 5\}$ and $c \to \infty$. Note that as c increases, the function becomes closer to a step function, with more probability information lost.

4.3.3 Asymptotic Results

In this section, we explore the asymptotic properties of our proposed MAC. In particular, we extend the notations of excess ℓ-risk and excess risk, which were used by Bartlett, Jordan and McAuliffe [2006], from the binary case to the multicategory counterpart, and study their convergence rates as the sample size $n \to \infty$. In this chapter, we focus on linear learning with a diverging number of covariates p, under the form (4.2). The penalty $J(\boldsymbol{f})$ we consider is the linear combination of the L_1 and L_2 penalties, similar to the Elastic Net penalty [Zou and Trevor, 2005]. For simplicity, we assume that the intercepts are included in $J(\boldsymbol{f})$ by artificially adding a variable to $\boldsymbol{x}$ whose value is always 1. In this case, we can write $J(\boldsymbol{f}) = \alpha \sum_{j=1}^{k-1} \|\boldsymbol{\beta}_j\|_1 + (1-\alpha) \sum_{j=1}^{k-1} \|\boldsymbol{\beta}_j\|_2^2$, where $\boldsymbol{\beta}_j;\ j = 1, \ldots, k-1$ are vectors of length $p \geq 2$ that include the intercepts. Here $\|\cdot\|_1$ and $\|\cdot\|_2$ are the regular L_1 norm and L_2 norm of a vector, and $\alpha \in [0, 1]$ is the parameter that controls the relative proportion of the L_1 and L_2 penalties.

The L_1 penalty is well known for its variable selection property, and we show that the convergence rates of the excess ℓ-risk for $\alpha > 0$ and $\alpha = 0$ are different. In particular, we show in Section 4.3.3.1 that when $\alpha > 0$, we have convergence as long as $p = o(e^n)$, and when $\alpha = 0$, the convergence requires that $p = o(n)$. Furthermore, we study the convergence rate of $\hat{\boldsymbol{\beta}}_j$ to its best value, under the uniform metric, in Section 4.3.3.2. In Section 4.3.3.3, we study the convergence rate of the excess risk using the results obtained in Section 4.3.3.1 and a conversion condition. Because we make some assumptions for the theory, in Section 4.3.3.4, we give an illustrative example, where the assumptions can be verified and the convergence rates can be computed.

For the remaining discussion of Section 4.3, we assume that the number of covariates (including the intercept) is $p = p_n$, and the penalty is $J(\boldsymbol{f}) \le s = s_n$, where both p_n and s_n may depend on n. We assume that each coordinate $x^{(l)} \in [0,1]$; $l = 1, \dots, p_n$, and the loss function ℓ is Lipschitz with the constant 1. Our theory can be analogously extended to other bounded domains and Lipschitz functions with different constants. Because p_n may become unbounded as $n \to \infty$, we assume that the underlying distribution $\mathbb{P}(\boldsymbol{X}, Y)$ is defined on $([0,1]^\infty \times \{1, \dots, k\}, \sigma^\infty([0,1]^\infty) \times 2^{\{1,\dots,k\}})$ with $\sigma^\infty([0,1]^\infty)$ being the σ-field generated by the open balls introduced by the uniform metric $d(\boldsymbol{x}, \boldsymbol{x}') = \sup_j |x_j - x'_j|$.

4.3.3.1 Convergence Rate of the Excess ℓ-Risk

Before stating our theory, we first introduce some notations and definitions. In the literature of linear learning, we have that $\boldsymbol{f} = (f_1, \dots, f_{k-1}) : f_j(\boldsymbol{x}) = \boldsymbol{\beta}_j^T \boldsymbol{x}$; $j = 1, \dots, k-1$. Note that the intercepts are included in $\boldsymbol{\beta}_j$'s, as discussed above. Let $\mathcal{F}(p_n, s_n) = \{\boldsymbol{f} = (f_1, \dots, f_{k-1}) : f_j(\boldsymbol{x}) = \boldsymbol{\beta}_j^T \boldsymbol{x};\ j = 1, \dots, k-1, \alpha \sum_{j=1}^{k-1} \|\boldsymbol{\beta}_j\|_1 + (1-\alpha) \sum_{j=1}^{k-1} \|\boldsymbol{\beta}_j\|_2^2 \le s_n\}$, and let $\mathcal{F}(p_n) = \bigcup_{0 \le s_n < \infty} \mathcal{F}(p_n, s_n)$ be the full p_n-dimensional model. Suppose

$$\hat{\boldsymbol{f}} = \operatorname*{argmin}_{\boldsymbol{f} \in \mathcal{F}(p_n, s_n)} \frac{1}{n} \sum_{i=1}^{n} \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}(\boldsymbol{x}_i) \rangle)$$

is the empirical minimizer of the optimization problem (4.2), and $\boldsymbol{f}^{(p_n)} = \operatorname{arginf}_{\boldsymbol{f} \in \mathcal{F}(p_n)} E\ell(\langle \mathcal{Y}_Y, \boldsymbol{f}(\boldsymbol{x}) \rangle)$ represents the best model under the full p_n-dimensional model. Note that $\boldsymbol{f}^{(p_n)}$ may not belong to $\mathcal{F}(p_n)$.

Let $e_\ell(\boldsymbol{f}, \boldsymbol{f}^{(p_n)}) = E\ell(\langle \mathcal{Y}_Y, \boldsymbol{f} \rangle) - E\ell(\langle \mathcal{Y}_Y, \boldsymbol{f}^{(p_n)} \rangle)$. We call $e_\ell(\boldsymbol{f}, \boldsymbol{f}^{(p_n)})$ the excess ℓ-risk, and name $e(\boldsymbol{f}, \boldsymbol{f}^{(p_n)}) = E(L(Y, \boldsymbol{f})) - E(L(Y, \boldsymbol{f}^{(p_n)}))$ the excess risk, where L is the $0-1$ loss. Hence $E(L(Y, \boldsymbol{f}))$

is the generalization error of $\boldsymbol{f}$. Note that these definitions are natural generalizations from the binary case discussed in Bartlett, Jordan and McAuliffe [2006].

We are ready to present our main results in this section.

Theorem 4.3.4. *Assume $\tau_n := (\frac{\log(p_n)}{n})^{1/2} \to 0$ as $n \to \infty$. If $\alpha > 0$,*

$$e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)}) = O(\max(s_n \tau_n \log(\tau_n^{-1}), d_n)),$$

a.s. under $\mathbb{P}$; and if $\alpha = 0$,

$$e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)}) = O(\max(\sqrt{p_n s_n} \tau_n \log(\tau_n^{-1}), d_n)),$$

a.s. under $\mathbb{P}$. Here $d_n = \inf_{\boldsymbol{f} \in \mathcal{F}(p_n, s_n)} e_\ell(\boldsymbol{f}, \boldsymbol{f}^{(p_n)})$ is the approximation error between $\mathcal{F}(p_n, s_n)$ and $\mathcal{F}(p_n)$.

In Theorem 4.3.4, the balance between the approximation error and estimation error is represented by d_n and s_n. In particular, as s_n increases, the function class $\mathcal{F}(p_n, s_n)$ becomes larger and the approximation error d_n decreases. At the same time, the estimation error becomes larger. In this view, the best trade-off takes place when $s_n \tau_n \log(\tau_n^{-1}) \sim d_n$ for $\alpha > 0$, and $\sqrt{p_n s_n} \tau_n \log(\tau_n^{-1}) \sim d_n$ for $\alpha = 0$. If the model depends on finitely many predictors, we have a simpler version of Theorem 4.3.4.
Assumption A: There exists a finite s^*, such that $\boldsymbol{f}^{(p_n)} \in \mathcal{F}(p_n, s^*)$ for all p_n.

Under Assumption A, d_n is strictly 0 for n and s_n large enough, which leads to the following corollary.

Corollary 4.3.1. *When Assumption A is met, we have $s_n = s^*$ for all large n. Consequently, if $\alpha > 0$, $e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)}) = O(\tau_n \log(\tau_n^{-1}))$, a.s. under $\mathbb{P}$; and if $\alpha = 0$, $e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)}) = O(\sqrt{p_n} \tau_n \log(\tau_n^{-1}))$, a.s. under $\mathbb{P}$.*

Note that there are important differences between the cases of $\alpha > 0$ and $\alpha = 0$. For the former case, the convergence of $\tau_n \log(\tau_n^{-1})$ requires that $\tau_n = (\frac{\log(p_n)}{n})^{1/2}$ converges, or in other words, p_n grows no faster than the exponential of n. In the pure L_2 penalty case where $\alpha = 0$, the convergence of $\sqrt{p_n} \tau_n \log(\tau_n^{-1})$ requires that $\frac{p_n}{n} = o(1)$, which is a much stronger constraint on the diverging speed of p_n. Theorem 4.3.4 and Corollary 4.3.1 help to shed some light on the effectiveness of the L_1 penalty when the true model contains many noise covariates.

4.3.3.2 Convergence Rate of the Parameters

We have established the convergence rate for the excess ℓ-risk in the previous section. As we focus on linear learning, the convergence rates for the estimated parameters $\hat{\boldsymbol{\beta}}_j;\ j = 1, \ldots, k-1$ are of interest as well. In this section, we explore the convergence rate for $\hat{\mathcal{B}} - \mathcal{B}^{(p_n)}$, where $\mathcal{B} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{k-1})$ is a p_n by $k-1$ matrix whose columns are the parameters, $\hat{\mathcal{B}} = (\hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_{k-1})$ is the estimated parameter matrix, and $\mathcal{B}^{(p_n)} = (\boldsymbol{\beta}_1^{(p_n)}, \ldots, \boldsymbol{\beta}_{k-1}^{(p_n)})$ is the matrix whose columns represents the best parameters under the full p_n-dimensional model. Note that in Section 4.3.3 we assume that p_n depends on n and can go to infinity, hence we study the convergence rate with respect to the uniform metric, $d(\mathcal{B}_1, \mathcal{B}_2) = \sup_{i,j}((\mathcal{B}_1)_{i,j} - (\mathcal{B}_2)_{i,j})$, where $\mathcal{B}_{i,j}$ is the (i,j)-th element of the matrix $\mathcal{B}$. First we introduce an assumption that is valid for many applications.

Assumption B: The marginal distribution of $\boldsymbol{X}$ is absolutely continuous with respect to the Lebesgue measure on $[0,1]^\infty$.

Under Assumption B, we study the convergence rate of $d(\hat{\mathcal{B}}, \mathcal{B}^{(p_n)})$ in the next theorem.

Theorem 4.3.5. *Suppose the loss function ℓ is differentiable and convex, and the theoretical minimizer $\boldsymbol{f}^* \in \mathcal{F}(p_n)$ for some p_n. Then under Assumption B, $d(\hat{\mathcal{B}}, \mathcal{B}^{(p_n)}) = O(\tau_n \log(\tau_n^{-1}))^{1/2}$ for $\alpha > 0$ and $d(\hat{\mathcal{B}}, \mathcal{B}^{(p_n)}) = O(\sqrt{p_n}\tau_n \log(\tau_n^{-1}))^{1/2}$ for $\alpha = 0$, a.s. under P.*

On the other hand, suppose the loss function ℓ is differentiable and convex, and the theoretical minimizer $\boldsymbol{f}^* \notin \mathcal{F}(\infty)$. Then under Assumptions A and B, $d(\hat{\mathcal{B}}, \mathcal{B}^{(p_n)}) = O(\tau_n \log(\tau_n^{-1}))$ for $\alpha > 0$ and $d(\hat{\mathcal{B}}, \mathcal{B}^{(p_n)}) = O(\sqrt{p_n}\tau_n \log(\tau_n^{-1}))$ for $\alpha = 0$, a.s. under P.

Theorem 4.3.5 indicates that the convergence rate of $\hat{\mathcal{B}}$ to $\mathcal{B}^{(p_n)}$ depends on whether the function class $\mathcal{F}$ is large enough. In practice, the situation that $\boldsymbol{f}^* \in \mathcal{F}(p_n)$ for some p_n rarely happens, and the convergence rate of $\hat{\mathcal{B}}$ to $\mathcal{B}^{(p_n)}$ is the same as the excess ℓ-risk for most of the cases, if the true model is indeed sparse. We give an illustrative example in Section 4.3.3.4, where the theoretical minimizer $\boldsymbol{f}^* \in \mathcal{F}(2)$, and the convergence rate of $\hat{\mathcal{B}}$ to $\mathcal{B}^{(p_n)}$ is slower than that of the excess ℓ-risk.

4.3.3.3 Convergence Rate of the Excess Risk

In Section 4.3.3.1, we studied the convergence rate of the excess ℓ-risk. In practice, the convergence rate is of interest if the classification accuracy of a given model converges to the best possible accuracy. In other words, we are interested in the consistency and the convergence rate of the excess risk. In this section, we establish a relationship between $e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)})$ and $|e(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)})|$. Note that $e(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)})$ is the difference between the misclassification rate of $\hat{\boldsymbol{f}}$ and $\boldsymbol{f}^{(p_n)}$. The technique of studying the excess risk using excess ℓ-risk in the binary case was previously studied in Zhang [2004*b*] and Bartlett, Jordan and McAuliffe [2006].

Define the L_2 metric on $\mathcal{F}$ as $\delta(\boldsymbol{f}, \boldsymbol{f}') = (\sum_{j=1}^{k-1} E(f_j(\boldsymbol{X}) - f'_j(\boldsymbol{X}))^2)^{1/2}$. The following conversion assumption controls the behaviors of the excess ℓ-risk and the excess risk in a small neighborhood of $\boldsymbol{f}^{(p_n)}$, introduced by the L_2 metric. Note that the same assumption was also used in Wang and Shen [2007] and Shen and Wang [2007].

Assumption C: There exist constants $\gamma_1 \in [1, \infty)$ and $\gamma_2 \in (0, \infty)$ such that for all small $\epsilon > 0$,

$$\inf_{\{\boldsymbol{f} \in \mathcal{F}(p_n): \delta(\boldsymbol{f}, \boldsymbol{f}^{(p_n)}) \geq \epsilon\}} e_\ell(\boldsymbol{f}, \boldsymbol{f}^{(p_n)}) \geq \xi_1(p_n)\epsilon^{\gamma_1}, \tag{4.3}$$

$$\sup_{\{\boldsymbol{f} \in \mathcal{F}(p_n): \delta(\boldsymbol{f}, \boldsymbol{f}^{(p_n)}) \leq \epsilon\}} |e(\boldsymbol{f}, \boldsymbol{f}^{(p_n)})| \leq \xi_2(p_n)\epsilon^{\gamma_2}, \tag{4.4}$$

where $\xi_1(p_n)$ and $\xi_2(p_n)$ may depend on p_n.

Corollary 4.3.2. *Under Assumption C, assume that $\tau_n \to 0$ as $n \to \infty$. Then $|e(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)})| = O(\max(s_n\tau_n \log(\tau_n^{-1}), d_n))^{\frac{\gamma_2}{\gamma_1}}$ a.s. under $\mathbb{P}$ if $\alpha > 0$, and $|e(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)})| = O(\max(\sqrt{p_n s_n}\tau_n \log(\tau_n^{-1}), d_n))^{\frac{\gamma_2}{\gamma_1}}$ a.s. under $\mathbb{P}$ if $\alpha = 0$. Moreover, suppose Assumption A is also met, then if $\alpha > 0$, $|e(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)})| = O(\tau_n \log(\tau_n^{-1}))^{\frac{\gamma_2}{\gamma_1}}$, a.s. under $\mathbb{P}$; and if $\alpha = 0$, $|e(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)})| = O(\sqrt{p_n}\tau_n \log(\tau_n^{-1}))^{\frac{\gamma_2}{\gamma_1}}$, a.s. under $\mathbb{P}$.*

In the next section, we give an simple yet illustrative example in which Assumptions A, B and C are satisfied, and γ_1, γ_2 can be calculated explicitly.

4.3.3.4 An Illustrative Example

In this section, we consider an example, where Assumptions A, B and C can be verified. In view of Corollary 4.3.2, the constants γ_1, γ_2 can be calculated explicitly.

For simplicity, let $X^{(1)} \in [-1, 1]$ and the rest $\boldsymbol{X}$'s be independent random noises which are continuously distributed. Recall that we include the intercepts in $\boldsymbol{\beta}_j$; $j = 1, \ldots, k-1$. Without loss of generality, we assume that the last element of $\boldsymbol{X}$ is fixed at 1, and the last elements of $\boldsymbol{\beta}_j$'s correspond to the intercept. Hence the true model in this example depends on $X^{(1)}$ and $X^{(p_n)}$ only. Consider a binary classification problem, where the prior proportions of the two classes are the same, and f is one dimensional. As an example, we use the LUM loss with $c = 0$ and $a = 1$. Let the marginal distribution of $X^{(1)}|Y = 1$ and $X^{(1)}|Y = -1$ be proportional to $-\ell'(-x)$ and $-\ell'(x)$. That is, for each fixed $x \in [-1, 1]$, $\frac{P(x|Y=1)}{P(x|Y=1)+P(x|Y=-1)} = \frac{\ell'(-x)}{\ell'(-x)+\ell'(x)}$, where $P(x|Y = 1)$ is the marginal probability density function of $X^{(1)}$ given $Y = 1$. In particular, $P(x|Y = 1) = -\frac{2}{3}\ell'(-x)$ and $P(x|Y = -1) = -\frac{2}{3}\ell'(x)$, $x \in [-1, 1]$. One can verify in this case the Bayes parameter $\boldsymbol{\beta}^*$ satisfies that $\boldsymbol{\beta}^* = (1, 0, \ldots, 0)^T$. Furthermore, one can verify that $\boldsymbol{x}^T\boldsymbol{\beta}^* = x^{(1)} = f^*(\boldsymbol{x})$. This indicates that $\boldsymbol{f}^* \in \mathcal{F}^{(p_n)}$ is satisfied for any $p_n \geq 2$.

From Theorem 4.3.5, we have $e_\ell(f, f^*) \geq C_1 d(\mathcal{B}, \mathcal{B}^*)^2$ for some constants C_1. Next, one can verify that there exists a constant C_2 such that $d(\mathcal{B}, \mathcal{B}^*)^2 \geq C_2\delta(f, f^*)^2$ because $x^{(1)}$ is bounded. Hence, we have that for $C_0 = C_1C_2$, $e_\ell(f, f^*) \geq C_0\delta(f, f^*)^2$. Thus we may choose $\gamma_1 = 2$, and $\xi_1(p_n) = C_0$, in Assumption C. Note that C_0 is independent of p_n.

Next we calculate γ_2 and $\xi_2(p_n)$. Without loss of generality, we restrict our discussion to $\boldsymbol{f} \in \mathcal{F}(2) := \{f : f(\boldsymbol{x}) = \beta_1 x^{(1)} + \beta_{p_n}\}$. This is because the other dimensions are irrelevant to the classification problem, and the excess risk is monotonely increasing with respect to the L_2 metric δ. Thus the Bayes parameter is $\beta_1^* = 1$ and $\beta_{p_n}^* = 0$.

Define $W(\beta_1, \beta_{p_n}) = E_{f=\beta_1 x^{(1)}+\beta_{p_n}}(L(f, Y))$. For any z_1, z_0, one can verify that the new classification boundary is $x^{(1)} = \frac{-z_0}{1+z_1}$. Thus, $W(\beta_1^*+z_1, \beta_{p_n}^*+z_0)-W(\beta_1^*, \beta_0^*) = |\int_0^t -\frac{2}{3}\ell'(-x)dt - \int_0^t -\frac{2}{3}\ell'(x)dt|$, where $t := \frac{-z_0}{1+z_1}$. After simplification, we have that $W(\beta_1^*+z_1, \beta_{p_n}^*+z_0)-W(\beta_1^*, \beta_{p_n}^*) = \frac{4}{3}|z_0|+o(z_0) \leq 2|z_0|$ for z_1 and z_0 small enough. Notice that $\delta(f, f^*) = E((f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2)^{1/2} \geq \frac{1}{\sqrt{2}}E(|f(\boldsymbol{x}) - f^*(\boldsymbol{x})|) = \frac{1}{\sqrt{2}}|z_0|$, where $f(\boldsymbol{x}) = (\beta_1^* + z_1)x^{(1)} + (\beta_{p_n}^* + z_0) = (1 + z_1)x^{(1)} + z_0$ and $f^*(\boldsymbol{x}) = x^{(1)}$. As a result, we can choose $\gamma_2 = 1$, and $\xi_2(p_n) = 2\sqrt{2}$, which is independent of p_n, in Assumption C. Note that this is a binary classification problem, and the methods in Zhang [2004*b*] and Bartlett, Jordan and McAuliffe [2006] yield similar results in the power $\frac{\gamma_2}{\gamma_1} = \frac{1}{2}$.

4.3.4 Finite Sample Error Bound

We use the surrogate loss ℓ as an upper bound of the $0-1$ loss, because it makes the optimization problem trackable, and the theoretical analysis of the former is easier to deal with. In Sections 4.3.3.1 and 4.3.3.3, we see that Assumption C of conversion builds a relationship between the excess ℓ-risk and the excess risk, and furthermore one can establish the convergence rate of the excess risk by that of the excess ℓ-risk. In this section, we study the finite sample bound on the expectation of the ℓ-risk given $\hat{\boldsymbol{f}}$, $E\ell(\langle \mathcal{Y}_Y, \hat{\boldsymbol{f}}\rangle)$, which can be regarded as an upper bound on the misclassification rate of $\hat{\boldsymbol{f}}$.

Theorem 4.3.6. *The solution $\hat{\boldsymbol{f}}$ to (4.2) satisfies that, with probability at least $1-\delta$,*

$$E\ell(\langle \mathcal{Y}_Y, \hat{\boldsymbol{f}}\rangle) \le \begin{cases} \frac{1}{n}\sum_{i=1}^{n}\ell(\langle \mathcal{Y}_{y_i}, \hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle) + \frac{6s_n}{\alpha}\sqrt{\frac{\log(2/\delta)}{n}} & \\ +\big(\frac{4(k-1)s_n}{\alpha n^{1/4}} + \frac{4s_n}{\alpha n^{1/4}}\sqrt{\log\frac{e+e(2p_n(k-1))}{\sqrt{n}}}\big) & \text{if } \alpha > 0, \\ \frac{1}{n}\sum_{i=1}^{n}\ell(\langle \mathcal{Y}_{y_i}, \hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle) + 6\sqrt{\frac{(k-1)p_n s_n\log(2/\delta)}{n}} & \\ +\big(\frac{4(k-1)\sqrt{p_n s_n}}{n^{1/4}} + \frac{4\sqrt{p_n s_n}}{n^{1/4}}\sqrt{\log\frac{e+e(2p_n(k-1))}{\sqrt{n}}}\big) & \text{if } \alpha = 0. \end{cases}$$

Theorem 4.3.6 gives an upper bound on $E\ell(\langle \mathcal{Y}_Y, \hat{\boldsymbol{f}}\rangle)$ that depends only on n, s_n, p_n, α and the training sample, hence it is directly computable from the data and the model we choose.

Remark 1: When solving the optimization (4.1), to calculate the result in Theorem 4.3.6, one may substitute $\frac{s_n}{\alpha}$ by $\sum_{j=1}^{k-1}\|\hat{\boldsymbol{\beta}}_j\|_1$ if $\alpha > 0$ and $p_n s_n$ by $\sum_{j=1}^{k-1}\|\hat{\boldsymbol{\beta}}_j\|_2^2$ if $\alpha = 0$.

4.3.5 Comparison to Existing Methods

In this section, we provide some theoretical insight on the comparison between our angle-based method and the regular multicategory large-margin classifiers using k classification functions with the sum-to-zero constraint. We show that if the true signal in linear learning is sparse, then our angle-based method can enjoy a smaller estimation error, with the approximation error fixed at 0. Consequently, our method can have more accurate prediction.

We focus on comparing the complexity of functional classes of the corresponding optimization problems. To illustrate the idea, we use the method in Lee, Lin and Wahba [2004] as an example for the regular simultaneous methods. The proofs and conclusions are analogous for many other simultaneous classifiers. To begin with, we introduce some notations. Let $t(p_n, s_n) = s_n/\alpha$ if $\alpha > 0$ and $(p_n s_n)^{1/2}$ if $\alpha = 0$. For our angle-based method, let $\boldsymbol{f}^{(p_n,s_n)} = \operatorname{argmin}_{\boldsymbol{f}\in\mathcal{F}(p_n,s_n)} E\{\ell(\langle \boldsymbol{f}, \mathcal{Y}_Y\rangle)\}$. Define $h_{\boldsymbol{f}}(\cdot) = \{2t(p_n,s_n)\}^{-1}\ \{\ell(\langle \boldsymbol{f}, \mathcal{Y}_\cdot\rangle) - \ell(\langle \boldsymbol{f}^{(p_n,s_n)}, \mathcal{Y}_\cdot\rangle)\}$, and let $\mathcal{H} = \{h_{\boldsymbol{f}} : \boldsymbol{f} \in \mathcal{F}(p_n,s_n)\}$. For the multicategory support vector machine by Lee, Lin and Wahba [2004], define $\mathcal{G}(p_n,s_n) = \{\boldsymbol{f} = (f_1,\ldots,f_k) : f_j(\boldsymbol{x}) = \boldsymbol{\beta}_j^T\boldsymbol{x}\ (j = 1,\ldots,k),\ \alpha\sum_{j=1}^{k}\|\boldsymbol{\beta}_j\|_1 + (1-\alpha)\sum_{j=1}^{k}\|\boldsymbol{\beta}_j\|_2^2 \le s_n\}$ with the sum-to-zero constraint. Let $\boldsymbol{f}_L^{(p_n,s_n)} = \operatorname{argmin}_{\boldsymbol{f}\in\mathcal{G}(p_n,s_n)} E\{\mathcal{L}(\boldsymbol{f}, Y)\}$, where $\mathcal{L}$ is the multicategory support vector machine loss used by Lee, Lin and Wahba [2004]. Define analogously $h_{L,\boldsymbol{f}}(\cdot) = \{2t(p_n, s_n)\}^{-1}$ $\{\mathcal{L}(\boldsymbol{f}, \cdot) - \mathcal{L}(\boldsymbol{f}_L^{(p_n,s_n)}, \cdot)\}$, and $\mathcal{H}_L = \{h_{L,\boldsymbol{f}} : \boldsymbol{f} \in \mathcal{G}(p_n, s_n)\}$. The next proposition provides the comparison between $\mathcal{H}$ and $\mathcal{H}_L$ in terms of their uniform covering numbers $N(\epsilon, \cdot)$. More details about uniform covering number are provided in the Supplementary Material.

Proposition 4.1. For positive ϵ small enough, $\log\{N(\epsilon, \mathcal{H})\}$ is bounded by $(2/\epsilon^2)\log[e + e\{2p_n(k-1)\}\epsilon^2] + \log(k-1)$, and $\log\{N(\epsilon, \mathcal{H}_L)\}$ is bounded by $(2/\epsilon^2)\log\{e + e(2p_n k)\epsilon^2\} + \log(k)$.

From Proposition 4.1 and its proof, we can conclude that for fixed α, the upper bound of $N(\epsilon, \mathcal{H})$ of the angle-based methods is smaller than that of $N(\epsilon, \mathcal{H}_L)$ with k functions, because our angle-based classifiers use only $k-1$ classification functions. In the Supplementary Material, we give an example where the upper bound of $N(\epsilon, \mathcal{H}_L)$ is almost tight. Furthermore, assume the true classification signal depends only on a finite number of predictors. In order to have the approximation error $d_n = 0$, one can verify that our angle-based method requires a smaller s_n compared to the multicategory support vector machines in Lee, Lin and Wahba [2004]. As a result, for a classification problem with sparse signal, our angle-based method can have a functional class smaller than that of Lee, Lin and Wahba [2004]. Consequently, the angle-based method can have a smaller estimation error, which can lead to a better classification performance. See the appendix for more details on how the covering number effects the estimation error. Intuitively, the regular simultaneous classifiers with k functions can introduce extra variability in the estimated classifier, which can deteriorate the corresponding classification performance. Our angle-based method with $k-1$ functions circumvents this difficulty and is more efficient.

As a remark, we point out that from Proposition 4.1, the difference of the uniform covering numbers becomes larger as the dimensionality p_n increases. This suggests that the difference in classification performance between our angle-based methods and the regular multicategory large-margin classifiers can be large for high dimensional problems. We confirm this finding in Section 4.5. In particular, we observe that for a classification problem with fixed and sparse signal, the difference in classification performance increases when the dimensionality increases.

4.4 Computational Algorithms and Tuning Procedures

In this section, we first discuss how to solve the optimization problem (4.1) or (4.2). Then we discuss how to select the best tuning parameters, which is very important in practice.

For a convex surrogate loss ℓ and a convex penalty term $J(\boldsymbol{f})$, (4.1) and (4.2) are convex problems and can be solved by many standard optimization tools [Boyd and Vandenberghe, 2004], depending on the choice of loss functions and the penalty. For instance, with the logistic deviance loss and the Elastic Net penalty, one can use the cyclic coordinate descent method in (4.1) [Friedman, Hastie and Tibshirani, 2010*a*]. If one applies the Proximal SVM loss [Fung and Mangasarian, 2001; Suykens and Vandewalle, 1999; Tang and Zhang, 2005] with $\ell(u) = (1-u)^2$ and the L_2 penalty, then there exists explicit solutions in (4.1), or one can obtain the equivalent solution by solving the KKT conditions generated from (4.2). In this section, we demonstrate how to implement the MAC problem (4.2) with the standard hinge loss and the L_1 penalty, under linear learning.

Both the hinge loss and the L_1 penalty are piecewise linear, and one can solve (4.2) via linear programming. For simplicity, we let the intercept be included in the penalty term, as in Section 4.3.3. The optimization problem (4.2) can be written as

$$\begin{aligned} &\min \frac{1}{n}\sum_{i=1}^{n}(1-\langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i}\rangle)_+, \\ \text{subject to } &\sum_{j=1}^{k-1}\|\boldsymbol{\beta}_j\|_1 \le s. \end{aligned}$$

Introducing $\xi_i;\ i=1,\ldots,n$ as slack variables, we have the above problem equivalent to

$$\begin{aligned} &\min \frac{1}{n}\sum_{i=1}^{n}\xi_i, \\ \text{subject to } &\sum_{j=1}^{k-1}\|\boldsymbol{\beta}_j\|_1 \le s, \\ &\xi_i \ge 0;\ i=1,\ldots,n, \\ &\xi_i - 1 + \langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i}\rangle \ge 0;\ i=1,\ldots,n, \\ &\boldsymbol{f}(\boldsymbol{x}_i) = (\boldsymbol{x}_i^T\beta_1, \ldots, \boldsymbol{x}_i^T\beta_{k-1})^T;\ i=1,\ldots,n. \end{aligned}$$

This optimization problem can be solved efficiently using linear programming.

In practice, a proper choice of the tuning parameter λ or s is crucial for the accuracy of the resulting classifier. Different tuning parameters correspond to different prediction models. There are various tuning techniques in the literature. Here, we briefly discuss the Cross Validation (CV) procedure which is commonly used in practice. For the CV approach, one partitions the data into q subsets of observations whose sizes are roughly the same. Each time a single subset of observations are used as the tuning data, and the remaining $q-1$ subsets are used as the training data. One repeats the process q times when all observations have been used in the tuning data, and the q prediction results are combined to select the best tuning parameter.

4.5 Simulation Examples

In this section, we conduct three simulated examples to explore the performance of MAC. For each example, we apply the logistic loss, SVM hinge loss and tuned LUM loss which corresponds to the tuned LUM classifier [Liu, Zhang and Wu, 2011] for ℓ in (4.1). As a comparison, we implement the existing methods in Vapnik [1998] (SVM1), Crammer et al. [2001] (SVM2), Lee, Lin and Wahba [2004] (SVM3) and Zhu and Hastie [2005] (Logistic). We show that MAC has more accurate classification results, and the computational cost for the MAC structure is significantly smaller than these alternative methods.

In each simulated example, we generate the dataset whose signal depends only on a few covariates, and we add extra noise covariates. For the classification performance, we compare both the test error rates (error), and probability estimation using the mean absolute error (MAE), $E|p-\hat{p}|$, on a test set of size 10^5. Here for the MAC SVM methods, we apply the results in Theorem 4.3.3 using the LUM loss with $c \to \infty$ to estimate the probability. Within one replication, the model is built on a train dataset, and the optimal tuning parameter is chosen by minimizing the classification error rate over an independent tune set with a grid search. We choose the commonly used L_2 penalty as the regularization term $J(\boldsymbol{f})$. Through 1000 replicates, we report the average time of training a model, tuning it on 30 different values of the tuning parameters, and making prediction on the test set. All simulations are done using R, on a 2.80 GHz Intel processor.

Example 1: In this example, we generate an eighteen-class dataset, where $P(Y=j)=1/18;\ j=1,\ldots,18$, and $P(\boldsymbol{X}|Y=j) \sim N(\mu_j, \sigma^2 I_2);\ j=1,\ldots,18$, where μ_j's are equally distributed on the unit circle, and σ is chosen such that the Bayes error is 0.1. We add 198 noisy covariates, which are *i.i.d.* from $N(0, 0.01)$. The train and tune datasets both have sample sizes 400.

Example 2: This is a four-class example with equal probabilities in $P(Y=j);\ j=1,\ldots,4$. The first class is uniformly distributed in the rectangular $[2,4]\times[-1,1]$, the second class is in $[0,2]\times[-1,1]$, the third class is in $[-2,0] \times [-1,1]$ and the last class is in $[-4,-2] \times [-1,1]$. Then we add 498 *i.i.d.* noisy covariates which are uniformly distributed in $[-1,1]$ to the dataset. We choose 100 train data points and another 100 for tuning.

Example 3: We let this ten-class example be in $\mathbb{R}^9$, and the marginal distributions of $\boldsymbol{X}|Y=j;\ j=1,\ldots,k$ are normal, while the ten mean vectors of different classes form a standard simplex in $\mathbb{R}^9$. The variance parameters are chosen such that the Bayes error is at 0.05. We then add 491 noisy covariates, each following *i.i.d.* $N(0,0.01)$ distribution, into the dataset. There are 200 data points for both the train and tune datasets.

		Ex 1			Ex 2			Ex 3		
		Error	MAE	Time	Error	MAE	Time	Error	MAE	Time
Existing Methods	SVM1	0.617	NA	34.69	0.644	NA	10.71	0.609	NA	49.46
	SVM2	0.638	NA	37.75	0.629	NA	11.62	0.597	NA	51.78
	SVM3	0.667	NA	39.11	0.633	NA	13.38	0.674	NA	53.46
	Logistic	0.469	0.143	26.46	0.657	0.158	9.116	0.388	0.247	34.77
MAC	Logistic	0.409	0.107	10.38	0.495	0.136	4.892	0.130	0.113	17.53
	SVM	0.551	0.162	14.51	0.433	0.141	5.771	0.156	0.133	26.97
	Tuned LUM	0.380	0.089	20.76	0.417	0.135	8.230	0.117	0.107	35.92

Table 4.1: Summary of the classification results for the simulated examples.

Table 4.1 reports the behaviors of the MACs and other alternative methods. We can see that MAC methods have smaller error rates overall, while the tuned MAC-LUM works the best. The computational times (in seconds) of MAC are significantly smaller than that of the other alternative methods. In terms of probability estimation, the approaches of Vapnik [1998], Crammer et al. [2001] and Lee, Lin and Wahba [2004] do not provide class conditional probability information, while all our MAC methods can estimate class conditional probabilities, and our methods are more accurate than the logistic classifier used in Zhu and Hastie [2005].

As a remark, we note that the existing SVMs we considered work poorly in Example 3, while the logistic method is moderately better than the existing SVMs. There are two possible reasons. Firstly, note that the centers of the ten classes in Example 3 form a standard simplex. This simplex matches the simplex structure of MAC, and consequently MAC may work much better here. The standard simultaneous methods do not have this property, and hence may be suboptimal. Secondly, the underlying class conditional probability functions are smooth, and hence soft classifiers like logistic regression (see Wahba [1999], Wahba [2002] and Liu, Zhang and Wu [2011] for definitions of soft and hard classifiers) tend to work better than hard classifiers such as the SVMs. Note that the performance difference of hard versus soft classifiers in the binary case was previously discussed in Liu, Zhang and Wu [2011].

4.6 Real Data Examples

In this section, we demonstrate the performance of our MAC via several real data analysis. In particular, we choose two datasets, CNAE-9 and Semeion Handwritten Digit (Semeion), from the UCI machine learning website, and another recent Glioblastoma Multiforme Cancer (GBM) gene expression dataset [Verhaak et al., 2010]. We summarize some attributes of these datasets in Table 4.2. We let the number of data points in the train, tune and test datasets be roughly the same, and perform a similar analysis on these datasets as in the simulation part. To reduce the computational time in the GBM dataset, we choose 1000 genes which have the largest median absolute deviation values based on the training sample within each replication.

Name	n	p	k
CNAE-9	1080	256	9
Semeion	1593	256	10
GBM	356	23285	4

Table 4.2: Summary of the real datasets in Section 4.6.

The results for the real datasets are reported in Table 4.3. We can see that the classification accuracy of our MAC methods are better than that of the existing methods, and MAC with the tuned LUM loss works the best overall. Furthermore, the computational time for MAC is considerably shorter. This is consistent with the simulation results.

4.7 Discussion and Future Direction

In this chapter we propose a novel Multicategory Angle-based large margin Classification (MAC) structure. Compared with the standard simultaneous multicategory large margin classifiers, MAC

		CNAE-9		Semeion		GBM	
		Error	Time	Error	Time	Error	Time
Existing Methods	SVM1	0.143	40.76	0.143	19.48	0.202	159.7
	SVM2	0.141	42.62	0.144	18.81	0.197	140.3
	SVM3	0.139	42.74	0.147	20.65	0.199	145.7
	Logistic	0.149	30.39	0.136	16.48	0.203	101.9
MAC	Logistic	0.128	15.71	0.131	8.911	0.179	67.16
	SVM	0.126	22.32	0.132	11.62	0.175	89.33
	Tuned LUM	0.126	32.96	0.125	16.17	0.176	116.8

Table 4.3: Summary of the classification results for the real datasets.

provides a natural geometric interpretation of multicategory classification. Through the association between classes and vertices of a standard simplex, we divide the classification function space into disjoint classification regions. Using the minimum angle prediction rule, the proposed MAC provides a simple multicategory large margin formulation. We show that MAC enjoys many desirable theoretical properties. Because MAC is free of the commonly used sum-to-zero constraint, the computational speed of MAC is significantly faster than that of many other existing methods. Our simulation examples and real data analyzes show that the classification accuracy of MAC is very competitive.

We would like to point out one possible future research direction, which is motivated by the Fisher inconsistency of the MAC SVM. For the regular simultaneous multicategory method, a naive generalization of the binary SVM is not Fisher consistent [Liu, 2007], just like the MAC SVM. Lee, Lin and Wahba [2004] proposed a Multicategory SVM (MSVM) method that is Fisher consistent even though the hinge loss is not differentiable. Liu and Yuan [2011] proposed a Reinforced MSVM (RMSVM) framework with the method in Lee, Lin and Wahba [2004] as a special case. The RMSVM technique uses a loss function that is a convex combination of the multicategory loss in Lee, Lin and Wahba [2004] and a naive hinge loss. Liu and Yuan [2011] studied the Fisher consistency of the classifiers with all possible convex combinations, and proposed a Fisher consistent MSVM method that outperforms the naive MSVM, the MSVM in Lee, Lin and Wahba [2004] and the MSVM in Weston and Watkins [1999]. We believe that a similar reinforce approach can be taken under the MAC framework to make the classifier Fisher consistent and to improve the classification accuracy. Further investigation will be pursued.

4.8 Appendix

Before the proof of Theorem 4.3.1, we introduce the following lemma.

Lemma 4.8.1. *Suppose we have an arbitrary* $\boldsymbol{f} \in \mathbb{R}^{k-1}$*. For any* $u, v \in \{1, \ldots, k\}$ *such that* $u \neq v$*, define* $\boldsymbol{T}_{u,v} = \mathcal{Y}_u - \mathcal{Y}_v$*. For any scalar* $z \in \mathbb{R}$*,* $\langle(\boldsymbol{f} + z\boldsymbol{T}_{u,v}), \mathcal{Y}_w\rangle = \langle\boldsymbol{f}, \mathcal{Y}_w\rangle$*, where* $w \in \{1, \ldots, k\}$ *and* $w \neq u, v$*. Furthermore, we have that* $\langle(\boldsymbol{f} + z\boldsymbol{T}_{u,v}), \mathcal{Y}_u\rangle - \langle\boldsymbol{f}, \mathcal{Y}_u\rangle = -\langle(\boldsymbol{f} + z\boldsymbol{T}_{v,u}), \mathcal{Y}_v\rangle + \langle\boldsymbol{f}, \mathcal{Y}_v\rangle$*.*

Proof of Lemma 4.8.1: Because $\mathcal{Y}$ is a standard simplex, $\boldsymbol{T}_{u,v} \perp \mathcal{Y}_w$ for any $w \neq u \neq v$, and this proves the first part of Lemma 4.8.1. The second part follows after some calculation. □

The above lemma is useful in the sense that for any given $\boldsymbol{f}$, we may start from $\boldsymbol{f}$, and move along the direction defined by $\boldsymbol{T}_{u,v}$, without changing the inner product between $\boldsymbol{f}$ and $\mathcal{Y}_w$, where $w \neq u, v$. Moreover, if we move along $\boldsymbol{T}_{u,v}$, the sum of the inner products $\langle\boldsymbol{f}, \mathcal{Y}_u\rangle + \langle\boldsymbol{f}, \mathcal{Y}_v\rangle$ remains unchanged. This observation leads to the proof of Theorem 4.3.1.

Proof of Theorem 4.3.1: Recall the definition of $\boldsymbol{f}^*$ is that $\boldsymbol{f}^* = \operatorname{argmin}_{\boldsymbol{f}} E[\ell(\langle f(\boldsymbol{X}), \mathcal{Y}_Y\rangle)|\boldsymbol{X} = \boldsymbol{x}] = \operatorname{argmin}_{\boldsymbol{f}} \sum_{j=1}^k P_j \ell(\langle\mathcal{Y}_j, \boldsymbol{f}\rangle)$.

We need to show that when $P_1 > P_2$, $\langle\mathcal{Y}_1, \boldsymbol{f}^*\rangle > \langle\mathcal{Y}_2, \boldsymbol{f}^*\rangle$. We do this by contradiction. If $\langle\mathcal{Y}_1, \boldsymbol{f}^*\rangle = \langle\mathcal{Y}_2, \boldsymbol{f}^*\rangle$, let $\boldsymbol{f}^{**}$ be such that $\langle\mathcal{Y}_j, \boldsymbol{f}^{**}\rangle = \langle\mathcal{Y}_j, \boldsymbol{f}^*\rangle$ for $j \geq 3$ and $\langle\mathcal{Y}_1, \boldsymbol{f}^{**}\rangle = \langle\mathcal{Y}_1, \boldsymbol{f}^*\rangle + \epsilon$, $\langle\mathcal{Y}_2, \boldsymbol{f}^{**}\rangle = \langle\mathcal{Y}_2, \boldsymbol{f}^*\rangle - \epsilon$, where $\epsilon > 0$ is a small number. Note that such a vector $\boldsymbol{f}^{**}$ always exists because of Lemma 4.8.1 and the fact that inner product is continuous. Now $\sum_{j=1}^k P_j \ell(\langle\mathcal{Y}_j, \boldsymbol{f}^{**}\rangle) - \sum_{j=1}^k P_j \ell(\langle\mathcal{Y}_j, \boldsymbol{f}^*\rangle)$ is equivalent to $\epsilon(P_1 - P_2)\ell'(\langle\mathcal{Y}_1, \boldsymbol{f}^*\rangle) + o(\epsilon)$ because we assume ℓ is differentiable. Since $P_1 - P_2 > 0$ and $\ell' < 0$, we have $\sum_{j=1}^k P_j \ell(\langle\mathcal{Y}_j, \boldsymbol{f}^{**}\rangle) < \sum_{j=1}^k P_j \ell(\langle\mathcal{Y}_j, \boldsymbol{f}^*\rangle)$, and it is a contradiction. This completes the proof. □

Proof of Theorem 4.3.2: By proof of Theorem 4.3.1, we have that when $P_1 > P_2$, $\langle\mathcal{Y}_1, \boldsymbol{f}^*\rangle > \langle\mathcal{Y}_2, \boldsymbol{f}^*\rangle$. Note that because the LUM loss $\ell(u)$ is linear when $u < 0$. From Lemma 4.8.1, we can conclude that the theoretical minimizer must be such that $\langle\mathcal{Y}_j, \boldsymbol{f}^*\rangle$ is the corresponding theoretical minimizer of a binary LUM problem with the corresponding class conditional probability $P(Y = 1|\boldsymbol{X}) = \frac{P_j}{P_j + P_k}$, for $j \neq k$. The rest of the proof simply follows the argument in Proposition 3.1. For $\langle\mathcal{Y}_k, \boldsymbol{f}^*\rangle$, note that $\sum_{j=1}^k \langle\mathcal{Y}_j, \boldsymbol{f}\rangle = 0$. This completes the proof. □

Remark 2: As mentioned in Section 4.3.1, the assumption $P_1 > P_2 > \cdots > P_{k-1} > P_k > 0$ greatly simplifies the displays of $\boldsymbol{f}^*$ in Theorem 4.3.2. Note that from the proof of Theorem 4.3.2, when $P_i = P_{i+1}$, we can move $\boldsymbol{f}^*$ on the same direction as $\boldsymbol{T}_{i,i+1}$ without changing the value $\sum_{j=1}^k P_j \ell(\langle\mathcal{Y}_j, \boldsymbol{f}^*\rangle)$. Therefore, the theoretical minimizer $\boldsymbol{f}^*$ is non-unique in this case. When $P_k = 0$, one can verify that $\sum_{j=1}^k P_j \ell(\langle\mathcal{Y}_j, \boldsymbol{f}^*\rangle)$ keeps decreasing as $\langle\mathcal{Y}_k, \boldsymbol{f}^*\rangle$ decreases and $\langle\mathcal{Y}_1, \boldsymbol{f}^*\rangle$ increases. Hence $\boldsymbol{f}^*$ becomes unbounded in this case. □

Proof of Theorem 4.3.3: Take partial derivative of $\sum_{j=1}^{k} P_j \ell(\langle \mathcal{Y}_j, \boldsymbol{f}^* \rangle)$ with respect to the m^{th} element of $\boldsymbol{f}^*$, f_m^*; $m = 1, \ldots, k-1$, and we have

$$0 = \sum_{j=1}^{k} P_j \ell'(\langle \mathcal{Y}_j, \boldsymbol{f}^* \rangle) \mathcal{Y}_{j,m}; \ m = 1, \ldots, k-1,$$

where $\mathcal{Y}_{j,m}$ is the m^{th} element of $\mathcal{Y}_j$. Observe that the above display can be written as

$$\sum_{j=1}^{k} P_j \ell'(\langle \mathcal{Y}_j, \boldsymbol{f}^* \rangle) \mathcal{Y}_j = \mathbf{0}_{k-1},$$

a weighted linear combination of $\mathcal{Y}_j$'s with the corresponding weight $P_j \ell'(\langle \mathcal{Y}_j, \boldsymbol{f}^* \rangle)$ on $\mathcal{Y}_j$. Because we have $\sum_{j=1}^{k} \mathcal{Y}_j = 0$, and also note that $\sum_{j=1}^{k} P_j = 1$, we may conclude that $P_j = \frac{\ell'(\langle \mathcal{Y}_j, \boldsymbol{f}^* \rangle)^{-1}}{\sum_{i=1}^{k} \ell'(\langle \mathcal{Y}_i, \boldsymbol{f}^* \rangle)^{-1}}$. Replace the theoretical minimizer by the estimated classification function and this completes the proof. □

Before proving Theorem 4.3.4, we first introduce some further notations.

Define $t(p_n, s_n) = s_n/\alpha$ if $\alpha > 0$ and $\sqrt{p_n s_n}$ if $\alpha = 0$. Hence for $\alpha > 0$, we have that $\hat{f}_j(\boldsymbol{x}) = \hat{\boldsymbol{\beta}}_j \boldsymbol{x} \leq \|\hat{\boldsymbol{\beta}}_j\|_1 \leq t(p_n, s_n)$, and for $\alpha = 0$, $\hat{f}_j(\boldsymbol{x}) = \hat{\boldsymbol{\beta}}_j \boldsymbol{x} \leq \sqrt{p_n} \|\hat{\boldsymbol{\beta}}_j\|_2 \leq t(p_n, s_n)$, for $j = 1, \ldots, k-1$. Thus, it suffices to consider $\tilde{\mathcal{F}}(p_n, s_n) = \mathcal{F}(p_n, s_n) \cap \{\boldsymbol{f} : \|\boldsymbol{f}\|_1 \leq (k-1) t(p_n, s_n)\}$.

Define $\boldsymbol{f}^{(p_n, s_n)} = \text{argmin}_{\boldsymbol{f} \in \mathcal{F}(p_n, s_n)} E\ell(\langle \boldsymbol{f}, \mathcal{Y}_Y \rangle)$, and $h_{\boldsymbol{f}}(\cdot) = (2(k-1)t(p_n, s_n))^{-1} (\ell(\langle \boldsymbol{f}, \mathcal{Y}_\cdot \rangle) - \ell(\langle \boldsymbol{f}^{(p_n, s_n)}, \mathcal{Y}_\cdot \rangle))$, and $\mathcal{H} = \{h_{\boldsymbol{f}} : \boldsymbol{f} \in \tilde{\mathcal{F}}(p_n, s_n)\}$. Because ℓ is Lipschitz with constant 1, $\ell(\langle \boldsymbol{f}, \cdot \rangle) - \ell(\langle \boldsymbol{f}', \cdot \rangle) \leq \langle \boldsymbol{f} - \boldsymbol{f}', \cdot \rangle$. Note that each coordinate of $\mathcal{Y}$ is bounded by 1, hence $\ell(\langle \boldsymbol{f}, \cdot \rangle) - \ell(\langle \boldsymbol{f}', \cdot \rangle) \leq \|\boldsymbol{f}\|_1 + \|\boldsymbol{f}'\|_1 \leq 2(k-1)t(p_n, s_n)$. Thus the $L_2(Q)$ diameter of $\ell(\langle \boldsymbol{f}, \cdot \rangle) - \ell(\langle \boldsymbol{f}^{(p_n, s_n)}, \cdot \rangle)$ is bounded by $(2(k-1)t(p_n, s_n))$, and the $L_2(Q)$ diameter of $\mathcal{H}$ is bounded by 1.

The proof of Theorem 4.3.4 is based on the next theorem.

Theorem 4.8.1. *For given n, p_n and s_n, assume that there exists $M > 0$ that satisfies*

$$(\log_2 \frac{16\epsilon_0\sqrt{6}}{M} + 1)^2 (\frac{256 \log(e + e(2p_n(k-1))\epsilon_0^2)}{n}) \leq \frac{M^2}{256}, \tag{4.5}$$

where $\epsilon_0 > 0$ is such that

$$\frac{2\log(e + e(2p_n(k-1))\epsilon_0^2)}{\epsilon_0^2} = \frac{1}{4} n M^2. \tag{4.6}$$

Then for $d_n = \inf_{\boldsymbol{f} \in \mathcal{F}(p_n,s_n)} e_\ell(\boldsymbol{f}, \boldsymbol{f}^{(p_n)})$, *we have*

$$P(e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)}) \geq 8(k-1)t(p_n,s_n)M + d_n) \leq 6(1 - \frac{1}{16nM^2})^{-1}\exp(-nM^2). \tag{4.7}$$

Proof of Theorem 4.8.1: Define the empirical process $h \to P_n h - Ph$, where $h \in \mathcal{H}$, $Ph = \int h dP$ and $P_n h = \frac{1}{n}\sum_{i=1}^n h(y_i)$. We have, by definition of d_n,

$$P(e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)}) > 8(k-1)t(p_n,s_n)M + d_n) \leq P(e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n,s_n)})(2(k-1)t(p_n,s_n))^{-1} > 4M).$$

Because $\hat{\boldsymbol{f}}$ satisfies that $e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n,s_n)})(2(k-1)t(p_n,s_n))^{-1} > 4M$, and $\hat{\boldsymbol{f}}$ minimizes the empirical loss $\frac{1}{n}\sum_{i=1}^n \ell(\langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i}\rangle)$, one can verify that $\frac{1}{n}\sum_{i=1}^n(\ell(\langle \boldsymbol{f}^{(p_n,s_n)}, \mathcal{Y}_{y_i}\rangle) - \ell(\langle \hat{\boldsymbol{f}}, \mathcal{Y}_{y_i}\rangle)) \geq 0$. Hence, we have

$$\begin{aligned}
&P(e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)}) > 8(k-1)t(p_n,s_n)M + d_n)\\
\leq\ & P^{outer}(\sup_{\boldsymbol{f} \in \tilde{\mathcal{F}}(p_n,s_n): e_\ell(\boldsymbol{f}, \boldsymbol{f}^{(p_n,s_n)})(2(k-1)t(p_n,s_n))^{-1} > 4M} \frac{1}{n}\sum_{i=1}^n(\ell(\langle \boldsymbol{f}^{(p_n,s_n)}, \mathcal{Y}_{y_i}\rangle) - \ell(\langle \boldsymbol{f}, \mathcal{Y}_{y_i}\rangle)) > 0)\\
\leq\ & P^{outer}(\sup_{\boldsymbol{f} \in \tilde{\mathcal{F}}(p_n,s_n): e_\ell(\boldsymbol{f}, \boldsymbol{f}^{(p_n,s_n)})(2(k-1)t(p_n,s_n))^{-1} > 4M} -\frac{1}{n}\sum_{i=1}^n[h_{\boldsymbol{f}}(y_i) - Eh_{\boldsymbol{f}}(Y)]\\
&\quad > (2(k-1)t(p_n,s_n))^{-1}E(\ell(\langle \boldsymbol{f}, \mathcal{Y}_Y\rangle) - \ell(\langle \boldsymbol{f}^{(p_n,s_n)}, \mathcal{Y}_Y\rangle))).
\end{aligned}$$

Here P^{outer} is the outer probability. Note that in the region $\boldsymbol{f} \in \tilde{\mathcal{F}}(p_n,s_n) : e_\ell(\boldsymbol{f}, \boldsymbol{f}^{(p_n,s_n)})$ $(2(k-1)t(p_n,s_n))^{-1} > 4M$, $(2(k-1)t(p_n,s_n))^{-1}E(\ell(\langle \boldsymbol{f}, \mathcal{Y}_Y\rangle) - \ell(\boldsymbol{f}^{(p_n,s_n)}, \mathcal{Y}_Y))$ is always larger than $4M$. Hence we have

$$P(e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)}) > 8(k-1)t(p_n,s_n)M + d_n) \leq P^{outer}(\sup_{h \in \mathcal{H}} |P_n h - Ph| > 4M).$$

The proof is then complete from Lemma 4.8.3. $\square$

Now we can prove Theorem 4.3.4.

Proof of Theorem 4.3.4: Let $M = 5\tau_n \log(\tau_n^{-1})$. Now we need to verify that (4.5) is valid for the choice of M and ϵ_0 determined by (4.6). Note that ϵ_0 goes to 0. Because if ϵ_0 is bounded away from 0 and ∞, the left hand side of (4.6) is of order $O(\log p_n)$, and the right hand side of (4.6) is of order

$O(\log p_n \log^2(\frac{1}{\tau_n}))$, which is a contradiction; if $\epsilon_0 \to \infty$, the left hand side of (4.6) is of order $o(\log p_n)$, which is still a contradiction. Now (4.5) is equivalent to

$$(\log_2 \frac{16\epsilon\sqrt{6}}{M} + 1)^2 \leq \frac{nM^2}{2^{16}\log(e + e(2p_n(k-1))\epsilon_0^2)}. \tag{4.8}$$

We have $\log_2 \frac{16\epsilon\sqrt{6}}{M} + 1 \propto \log_2 \frac{\epsilon_0}{M} \preceq \log_2 \frac{1}{M} = \log_2 \frac{1}{\tau_n}\frac{1}{\log\frac{1}{\tau_n}} \preceq \log\frac{1}{\tau_n}$, where $\propto$ is up to a constant equivalent, and $\preceq$ is up to a constant less than or equal to. Hence the left hand side of (4.8) has an order no greater than $O(\log^2(\tau_n^{-1}))$. For the right hand side of (4.8), we have $\frac{nM^2}{2^{16}\log(e+e(2p_n(k-1))\epsilon_0^2)} \propto \frac{1}{\epsilon_0^2}$. Note that the left hand side of (4.6) has order $O(\log p_n \log^2(\frac{1}{\tau_n}))$. If the order of $\frac{1}{\epsilon_0}$ is less than that of $\log\frac{1}{\tau_n}$, we have the order of the right hand side of (4.6) smaller than $O(\log p_n \log^2(\frac{1}{\tau_n}))$, because ϵ_0 goes to 0. Thus, (4.5) is valid, because the order of left hand side of (4.8) is less than that of the right hand side.

Lastly, note that $nM^2 = 25\log p_n \log^2(\frac{1}{\tau_n}) > \frac{5}{2}\log p_n \log(\frac{1}{\tau_n}) \geq \frac{5}{2}\log n > 2\log n$, and we have $\exp(-nM^2) \leq \exp(-2\log n) = \frac{1}{n^2}$. The results then follows from Borel-Cantelli Lemma. □

Now we need to provide the inner steps. There are two major steps, summarized as the following two lemmas. The first lemma controls the complexity of $\mathcal{H}$ in terms of the $L_2(Q)$ entropy.

For any $\epsilon > 0$, define $\mathcal{G}$ to be a ϵ-net of a class of function $\mathcal{F}$ if, for any $f \in \mathcal{F}$, there exists $g \in \mathcal{G}$ such that $\|g - f\|_{Q,2} \leq \epsilon$. Now let the $L_2(Q)$ covering number $N(\epsilon, \mathcal{F}, L_2(Q))$ be the minimal size of all possible ϵ-nets, and denote by $H(\epsilon, \mathcal{F}, L_2(Q))$ the logarithm of $N(\epsilon, \mathcal{F}, L_2(Q))$, which is referred to as the $L_2(Q)$ entropy. Define the uniform $L_2(Q)$ covering number, $N(\epsilon, \mathcal{F})$, to be $\sup_Q N(\epsilon, \mathcal{F}, L_2(Q))$, and define the uniform $L_2(Q)$ entropy $H(\epsilon, \mathcal{F})$ in a similar way. Lemma 4.8.2 gives an upper bound on $H(\epsilon, \mathcal{F})$.

Lemma 4.8.2. *For any $\epsilon > 0$, $H(\epsilon, \mathcal{H}) \leq \frac{2}{\epsilon^2}\log(e + e(2p_n(k-1))\epsilon^2)$.*

Proof of Lemma 4.8.2: The proof follows the same line as that of Theorem A.1 in Wang and Shen [2007]. To bound the $L_2(Q)$ entropy of $\mathcal{H}$, we can bound the $L_2(Q)$ entropy of $\mathcal{G} := \{\ell(\langle \boldsymbol{f}, \mathcal{Y}_\cdot\rangle) : \sum_{j=1}^{k-1}\|\boldsymbol{\beta}_j\|_1 \leq t(p_n, s_n)\}$. Next we find a ϵ-net on $\mathcal{G}$. Let $g = \ell(\langle \boldsymbol{f}, \mathcal{Y}_y\rangle), g' = \ell(\langle \boldsymbol{f}', \mathcal{Y}_y\rangle) \in \mathcal{G}$.

Note that

$$\|g - g'\|_{Q,2}^2 = E(\ell(\langle Y, \boldsymbol{f}(\boldsymbol{X})\rangle) - \ell(\langle Y, \boldsymbol{f}'(\boldsymbol{X})\rangle))^2 \leq E(\langle Y, \boldsymbol{f}(\boldsymbol{X}) - \boldsymbol{f}'(\boldsymbol{X})\rangle)^2 \leq E(\sum_{j=1}^{k-1}|f_j(\boldsymbol{X}) - f_j'(\boldsymbol{X})|)^2.$$

Now using Cauchy-Schwartz inequality, we have $E(\sum_{j=1}^{k-1}|f_j(\boldsymbol{X}) - f_j'(\boldsymbol{X})|)^2 \leq (k-1)\sum_{j=1}^{k-1}\|f_j - f_j'\|_{Q,2}^2$. Next, define $\vec{\boldsymbol{x}} = (\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_{k-1}^T)^T$ with each $\boldsymbol{x}_j$ a p_n dimensional vector. Let $\vec{f}(\vec{\boldsymbol{x}}) = \sum_{j=1}^{k-1}\boldsymbol{\beta}_j^T\boldsymbol{x}_j$. Also let $\vec{Q}$ be the distribution of $\vec{\boldsymbol{X}} = (\delta_1\boldsymbol{X}_1, \ldots, \delta_{k-1}\boldsymbol{X}_{k-1})$, where $\boldsymbol{X}_j$'s are *i.i.d.* randomly distributed with any arbitrary distribution Q, and $(\delta_1, \ldots, \delta_{k-1})$ has a joint distribution $P((\delta_1, \ldots, \delta_{k-1})^T = e_j) = (k-1)^{-1}$. Thus we may conclude that $\sum_{j=1}^{k-1}\|f_j - f_j'\|_{Q,2}^2 = (k-1)E_{\vec{Q}}(\vec{f} - \vec{f}')^2$, and so $\|g - g'\|_{Q,2}^2 \leq (k-1)^2\|\vec{f} - \vec{f}'\|_{Q,2}^2$. As a result, if we can bound $L_2(Q)$ entropy of the function class $\vec{\mathcal{F}} = \{\vec{f} : \vec{f}(\vec{\boldsymbol{x}}) = \sum_{j=1}^{k-1}\sum_{l=1}^{p_n}\beta_{j,l}x_{j,l};\ \sum_{j=1}^{k-1}\|\boldsymbol{\beta}_j\|_1 \leq t(p_n, s_n)\}$, we can bound $\mathcal{H}$.

To bound $\vec{\mathcal{F}}$, define $w_{j,l}(\vec{\boldsymbol{x}}) = t(p_n, s_n)x_{j,l}$. Then $\mathcal{J} = \{\pm w_{j,l}\}$ forms a basis for $\vec{\mathcal{F}}$. That is, each $\vec{f} = \sum_{j=1}^{k-1}\sum_{l=1}^{p_n}\beta_{j,l}x_{j,l} = \sum_{j=1}^{k-1}\sum_{l=1}^{p_n}\frac{|\beta_{j,l}|}{t(p_n,s_n)}(\text{sign}(\beta_{j,l})w_{j,l}(\vec{\boldsymbol{x}}))$ is a convex combination of $w_{j,l}$. Thus, $\vec{\mathcal{F}}$ is the convex hull of $\mathcal{J}$. By Lemma 2.6.11 in Van der Vaart and Wellner [2000], $N(\epsilon\text{diam}\mathcal{J}, \vec{\mathcal{F}}, L_2(\vec{Q})) \leq (e + e(2p_n(k-1))\epsilon^2)^{2/\epsilon^2}$, where $\text{diam}\mathcal{J} = \sup_{J_1,J_2\in\mathcal{J}}\|J_1 - J_2\|_{\vec{Q},2} \leq 2t(p_n, s_n)$. Thus, we conclude that

$$N(\epsilon, \mathcal{H}, L_2(Q)) = N(2(k-1)t(p_n, s_n)\epsilon, \mathcal{G}, L_2(Q))$$
$$\leq N(2t(p_n, s_n)\epsilon, \mathcal{J}, L_2(\vec{Q})) \leq (e + e(2p_n(k-1))\epsilon^2)^{2/\epsilon^2}.$$

Since the final bound is independent of Q, we have that the bound is uniform for any Q. □

Lemma 4.8.2 yields a bound on the entropy of $\mathcal{H}$ in the form of $A\epsilon^{-2}$ where A is a constant. Theorem A.2 in Wang and Shen [2007] obtains a probability tail bound on $\sup_{h\in\mathcal{H}}|P_nh - Ph|$. We include it as Lemma 4.8.3 for completeness.

Lemma 4.8.3. (Theorem A.2, Wang and Shen [2007]). *Assume n, p_n, M and ϵ_0 satisfy (4.5) and (4.6). We have* $P^{outer}(\sup_{h\in\mathcal{H}}|P_nh - Ph| > 4M) \leq 6(1 - \frac{1}{16nM^2})^{-1}\exp(-nM^2)$.

To prove Theorem 4.3.5, we need to first introduce the Bregman divergence and a preceding lemma. Note that Zhang [2004*b*] used the same Bregman divergence notation on binary classification problems.

Define the Bregman divergence of a convex function $g(\cdot)$ as

$$d_g(f_1, f_2) = g(f_2) - g(f_1) - g'(f_1)(f_2 - f_1),$$

where $g'(\cdot)$ in general is a subgradient of the convex function $g(\cdot)$. Define $e_{\ell|\boldsymbol{X}}(\boldsymbol{f}, \boldsymbol{f}') = [\ell(\langle \mathcal{Y}_Y, \boldsymbol{f}\rangle) - \ell(\langle \mathcal{Y}_Y, \boldsymbol{f}'\rangle)]|\boldsymbol{X} = \sum_{j=1}^k P_j[\ell(\langle \mathcal{Y}_j, \boldsymbol{f}\rangle) - \ell(\langle \mathcal{Y}_j, \boldsymbol{f}'\rangle)]$. We see, from the following lemma, that $e_{\ell|\boldsymbol{X}}(\boldsymbol{f}, \boldsymbol{f}')$ can be expressed in terms of the weighted sum of the Bregman divergence.

Lemma 4.8.4. *Suppose ℓ is differentiable. For any $\boldsymbol{f}$, $e_{\ell|\boldsymbol{X}}(\boldsymbol{f}, \boldsymbol{f}^*) = \sum_{j=1}^k P_j d_\ell(\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle, \langle \mathcal{Y}_j, \boldsymbol{f}\rangle)$.*

Proof of Lemma 4.8.4: One can verify that

$$e_{\ell|\boldsymbol{X}}(\boldsymbol{f}, \boldsymbol{f}^*) = \sum_{j=1}^k P_j d_\ell(\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle, \langle \mathcal{Y}_j, \boldsymbol{f}\rangle) + \sum_{j=1}^k P_j \ell'(\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle)(\langle \mathcal{Y}_j, \boldsymbol{f}\rangle - \langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle),$$

by definition of the Bregman divergence. Thus it suffices to show $\sum_{j=1}^k P_j \ell'(\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle)(\langle \mathcal{Y}_j, \boldsymbol{f}\rangle - \langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle) = 0$. Note that the partial derivative of $\sum_{j=1}^k P_j \ell(\langle \mathcal{Y}_j, \boldsymbol{f}\rangle)$ with respect to $\boldsymbol{f}$, denoted by $\bigtriangledown$, is $\mathbf{0}_{k-1}$, a zero vector of length $k-1$, at $\boldsymbol{f}^*$ because we assume that ℓ is differentiable, and $\boldsymbol{f}^*$ achieves the minimum of $\sum_{j=1}^k P_j \ell(\langle \mathcal{Y}_j, \boldsymbol{f}\rangle)$. Moreover, note that the m^{th} element of $\bigtriangledown$, $\bigtriangledown_m$, is $\sum_{j=1}^k P_j \ell'(\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle)\mathcal{Y}_j^{(m)}$, where $\mathcal{Y}_j^{(m)}$ is the m^{th} element of $\mathcal{Y}_j$. After some calculation, we have that $\sum_{j=1}^k P_j \ell'(\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle)(\langle \mathcal{Y}_j, \boldsymbol{f}\rangle - \langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle) = \langle \bigtriangledown, (\boldsymbol{f} - \boldsymbol{f}^*)\rangle = 0$. This completes the proof. $\square$

Proof of Theorem 4.3.5: Note that under the condition of Theorem 4.3.5, the set on which ℓ is not second order differentiable or the second order derivative is unbounded has measure 0 with respect to the Lebesgue measure, which we assume with probability 0 under linear learning. Hence following Lemma 4.8.4, we can expand the $e_{\ell|\boldsymbol{X}}(\hat{\boldsymbol{f}}, \boldsymbol{f}^*)$ into the Taylor series form

$$e_{\ell|\boldsymbol{X}}(\hat{\boldsymbol{f}}, \boldsymbol{f}^*) = \sum_{j=1}^k P_j[\frac{\ell''(\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle)}{2}(\langle \mathcal{Y}_j, \hat{\boldsymbol{f}}\rangle - \langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle)^2 + o((\langle \mathcal{Y}_j, \hat{\boldsymbol{f}}\rangle - \langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle)^2)], \tag{4.9}$$

with probability 1. Suppose the theoretical minimizer $\boldsymbol{f}^* \in \mathcal{F}(p_n)$ for some p_n. As n grows, note that (4.9) implies that $e_{\ell|\boldsymbol{X}}(\hat{\boldsymbol{f}}, \boldsymbol{f}^*) \propto (d(\hat{\mathcal{B}}_n, \mathcal{B}^{(p_n)}))^2$ for all $\boldsymbol{x}$ because ℓ'' is finite almost everywhere. Thus, we can conclude that $d(\hat{\mathcal{B}}_n, \mathcal{B}^{(p_n)}) \propto (e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^*))^{1/2}$ after taking expectation with respect to the marginal distribution of $\boldsymbol{X}$.

For the second part, note that

$$\begin{aligned} e_{\ell|\boldsymbol{X}}(\hat{\boldsymbol{f}}, \boldsymbol{f}^*) - e_{\ell|\boldsymbol{X}}(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)}) = &\sum_{j=1}^k P_j[\frac{\ell''(\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle)}{2}(\langle \mathcal{Y}_j, \hat{\boldsymbol{f}}\rangle - \langle \mathcal{Y}_j, \boldsymbol{f}^{(p_n)}\rangle) \\ &(\langle \mathcal{Y}_j, \hat{\boldsymbol{f}}\rangle + \langle \mathcal{Y}_j, \boldsymbol{f}^{(p_n)}\rangle - 2\langle \mathcal{Y}_j, \boldsymbol{f}^*\rangle) + o((\langle \mathcal{Y}_j, \hat{\boldsymbol{f}}\rangle - \langle \mathcal{Y}_j, \boldsymbol{f}^{(p_n)}\rangle))]. \end{aligned}$$

Now because $\langle \mathcal{Y}_j, \hat{\boldsymbol{f}} \rangle + \langle \mathcal{Y}_j, \boldsymbol{f}^{(p_n)} \rangle - 2\langle \mathcal{Y}_j, \boldsymbol{f}^* \rangle$ is bounded above by Assumption B, and bounded away from 0 because $\boldsymbol{f}^* \notin \mathcal{F}(\infty)$, analogous argument as in the previous paragraph proves the claim, and this completes the proof. $\Box$

Proof of Corollary 4.3.2: The proof uses the following lemma from Wang and Shen [2007].

Lemma 4.8.5. *(Lemma 2, Wang and Shen [2007]) Under Assumption C, there exists a constant $\xi(p_n)$ that may depend on p_n such that for all $\boldsymbol{f} \in \mathcal{F}(p_n)$, $|e(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)})| \leq \xi(p_n) e_\ell(\hat{\boldsymbol{f}}, \boldsymbol{f}^{(p_n)})^{\frac{\gamma_2}{\gamma_1}}$.*

The rest of the proof is immediate from Theorem 4.3.4 and Corollary 4.3.1. $\Box$

To prove Theorem 4.3.6, we apply a recent technique, the Rademacher complexities, whose error bound can be computed directly from the data. For more details about the Rademacher complexities, we refer the readers to Bartlett and Mendelson [2002], Koltchinskii and Panchenko [2002], Shawe-Taylor and Cristianini [2004], Bartlett, Bousquet and Mendelson [2005], Koltchinskii [2006] and the references therein.

We begin by defining the Rademacher complexity. Recall the definition of $\mathcal{F}(p_n, s_n)$ in Section 4.3.3.1. Let $\sigma := (\sigma_i;\ i = 1, \ldots, n)$ be *i.i.d.* random variables that take 1 and -1 with equal probability. Denote by S a sample of $(\boldsymbol{x}_i, y_i)$; $i = 1, \ldots, n$, *i.i.d.* from $P(\boldsymbol{X}, Y)$. Define the empirical Rademacher complexity of the function class $\mathcal{F}(p_n, s_n)$, with S given, to be

$$\hat{R}_n(\mathcal{F}(p_n, s_n)) = E_\sigma[\sup_{\boldsymbol{f} \in \mathcal{F}(p_n, s_n)} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}(\boldsymbol{x}_i) \rangle)].$$

Here E_σ means taking expectation with respect to the joint distribution of σ. Similarly, define the Rademacher complexity of $\mathcal{F}(p_n, s_n)$ to be

$$R_n(\mathcal{F}(p_n, s_n)) = E_{\sigma, S}[\sup_{\boldsymbol{f} \in \mathcal{F}(p_n, s_n)} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}(\boldsymbol{x}_i) \rangle)].$$

We divide the proof of Theorem 4.3.6 into two inner steps.

The first step is a theorem that shows with probability at least $1-\delta$, where $0 < \delta < 1$, $E\ell(\langle \mathcal{Y}_Y, \hat{\boldsymbol{f}} \rangle)$ is bounded by the sum of its empirical measurement, the Rademacher complexity of the function class $\mathcal{F}(p_n, s_n)$, and an penalty term on δ.

Theorem 4.8.2. *Let $R_n(\mathcal{F}(p_n, s_n))$ and $\hat{R}_n(\mathcal{F}(p_n, s_n))$ be defined as above. Then with probability at* *least* $1-\delta$,

$$E\ell(\langle \mathcal{Y}_Y, \hat{\boldsymbol{f}} \rangle) \le \frac{1}{n}\sum_{i=1}^{n} \ell(\langle \mathcal{Y}_{y_i}, \hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle) + 2R_n(\mathcal{F}(p_n, s_n)) + T_n(\delta), \tag{4.10}$$

where $T_n(\delta) = \frac{2s_n}{\alpha}\sqrt{\frac{\log(1/\delta)}{n}}$ *if* $\alpha > 0$, *and* $T_n(\delta) = 2\sqrt{\frac{(k-1)p_n s_n \log(1/\delta)}{n}}$ *if* $\alpha = 0$. *Moreover, with probability at least* $1-\delta$,

$$E\ell(\langle \mathcal{Y}_Y, \hat{\boldsymbol{f}} \rangle) \le \frac{1}{n}\sum_{i=1}^{n} \ell(\langle \mathcal{Y}_{y_i}, \hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle) + 2\hat{R}_n(\mathcal{F}(p_n, s_n)) + 3T_n(\delta/2). \tag{4.11}$$

The second step is a theorem that gives an upper bound on the empirical Rademacher complexity which is completely computable from the data and the model parameters k, n, p_n, s_n and α.

Theorem 4.8.3. *When* $\alpha > 0$, *the empirical Rademacher complexity*

$$\hat{R}_n(\mathcal{F}(p_n, s_n)) \le \frac{2(k-1)s_n}{\alpha n^{1/4}} + \frac{2s_n}{\alpha n^{1/4}}\sqrt{\log\frac{e + e(2p_n(k-1))}{\sqrt{n}}}.$$

When $\alpha = 0$,

$$\hat{R}_n(\mathcal{F}(p_n, s_n)) \le \frac{2(k-1)\sqrt{p_n s_n}}{n^{1/4}} + \frac{2\sqrt{p_n s_n}}{n^{1/4}}\sqrt{\log\frac{e + e(2p_n(k-1))}{\sqrt{n}}}.$$

Proof of Theorem 4.8.2: The proof consists of three parts. The first part uses the McDiarmid inequality [McDiarmid, 1989] to bound the left hand side of (4.10) in terms of its empirical estimation plus the expectation of their supremum difference, $E\phi$, where ϕ is defined below. The second part shows that $E\phi$ is bounded by the Rademacher complexity using symmetrization inequalities. The third part bounds the Rademacher complexity by the empirical Rademacher complexity.

For a given sample S, define

$$\phi(S) = \sup_{\boldsymbol{f} \in \mathcal{F}(p_n, s_n)} [E\ell(\langle \mathcal{Y}_Y, \boldsymbol{f}(\boldsymbol{X})\rangle) - \frac{1}{n}\sum_{i=1}^{n} \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}(\boldsymbol{x}_i)\rangle)].$$

Let $S^{(i,\boldsymbol{x})} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}'_i, y_i), \ldots, (\boldsymbol{x}_n, y_n)\}$ be a sample of $(\boldsymbol{X}, Y)$, and note that the difference between S and $S^{(i,\boldsymbol{x})}$ is only on the $\boldsymbol{x}$ value of their i^{th} pair. Similarly, define $S^{(i,y)} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_i, y'_i), \ldots, (\boldsymbol{x}_n, y_n)\}$, where the difference is on the y value. By definition, we have

$$\begin{aligned}|\phi(S) - \phi(S^{(i,\boldsymbol{x})})| = |&\sup_{\boldsymbol{f}\in\mathcal{F}(p_n,s_n)} [E\ell(\langle \mathcal{Y}_Y, \boldsymbol{f}(\boldsymbol{X})\rangle) - \frac{1}{n}\sum_S \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}(\boldsymbol{x}_i)\rangle)] \\ &- \sup_{\boldsymbol{f}\in\mathcal{F}(p_n,s_n)} [E\ell(\langle \mathcal{Y}_Y, \boldsymbol{f}(\boldsymbol{X})\rangle) - \frac{1}{n}\sum_{S^{(i,\boldsymbol{x})}} \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}(\boldsymbol{x}_i)\rangle)]| \end{aligned} \tag{4.12}$$

For brevity let $\boldsymbol{f}^S$ be the function that achieves the supremum of $\phi(S)$. The case that no function reaches the supremum can be treated analogously, with only additional discussion on the arbitrarily small difference between $\phi(\boldsymbol{f})$ and its supremum, and we omit the details here. Now, $|\phi(S) - \phi(S^{(i,\boldsymbol{x})})| \le |E\ell(\langle \mathcal{Y}_Y, \boldsymbol{f}^S(\boldsymbol{X})\rangle) - \frac{1}{n}\sum_S \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}^S(\boldsymbol{x}_i)\rangle) - E\ell(\langle \mathcal{Y}_Y, \boldsymbol{f}^S(\boldsymbol{X})\rangle) + \frac{1}{n}\sum_S \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}^S(\boldsymbol{x}_i)\rangle)|$. After simplification, we have that

$$\begin{aligned}|\phi(S) - \phi(S^{(i,\boldsymbol{x})})| \le& \frac{1}{n}|\ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}^S(\boldsymbol{x}_i)\rangle) - \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}^S(\boldsymbol{x}'_i)\rangle)| \\ \le& \frac{1}{n}|\langle \mathcal{Y}_{y_i}, \boldsymbol{f}^S(\boldsymbol{x}_i) - \boldsymbol{f}^S(\boldsymbol{x}'_i)\rangle| \\ \le& T_n^*,\end{aligned}$$

where $T_n^* = \frac{2s_n}{n\alpha}$ if $\alpha > 0$ and $T_n^* = \frac{2}{n}\sqrt{(k-1)p_n s_n}$ otherwise. Similarly, one can show that $|\phi(S) - \phi(S^{(i,y)})| \le T_n^*$. Hence, by the McDiarmid inequality, for any $t > 0$, $P(\phi(S) - E\phi(S) \ge t) \le \exp(-\frac{2t^2}{n(T_n^*)^2})$, or equivalently, with probability at least $1-\delta$, $\phi(S) - E\phi(S) \le T_n(\delta)$. Rearranging the orders and replacing the supremum by $\hat{\boldsymbol{f}}$, we have that with probability at least $1-\delta$, $E\ell(\langle \mathcal{Y}_Y, \hat{\boldsymbol{f}}\rangle) \le \frac{1}{n}\sum_{i=1}^n \ell(\langle \mathcal{Y}_{y_i}, \hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle) + E\phi(S) + T_n$. This is the first part of the proof.

Next, we bound $E\phi(S)$ by the Rademacher complexity.

Define $S' = \{(\boldsymbol{x}'_i, y'_i);\ i = 1, \ldots, n\}$ as a duplicate sample of size n with identical distribution as S. Denote by E_S the action of taking expectation with respect to the distribution of S. Note that $E_{S'}[\frac{1}{n}\sum_{S'} \ell(\langle \mathcal{Y}_{y'_i}, \hat{\boldsymbol{f}}(\boldsymbol{x}'_i)\rangle)|S] = E\ell(\langle \mathcal{Y}_Y, \hat{\boldsymbol{f}}(\boldsymbol{x})\rangle)$, and

$$E_{S'}[\frac{1}{n}\sum_S \ell(\langle \mathcal{Y}_{y_i}, \hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle)|S] = \frac{1}{n}\sum_S \ell(\langle \mathcal{Y}_{y_i}, \hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle).$$

We have, by Jensen's inequality and the property of σ,

$$\begin{aligned}
E\phi(S) &= E_S\big[\sup_{\boldsymbol{f}\in\mathcal{F}(p_n,s_n)} E_{S'}[\frac{1}{n}\sum_{S'}\ell(\langle\mathcal{Y}_{y_i'},\hat{\boldsymbol{f}}(\boldsymbol{x}_i')\rangle) - \frac{1}{n}\sum_{S}\ell(\langle\mathcal{Y}_{y_i},\hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle)]|S\big] \\
&\leq E_{S,S'}[\sup_{\boldsymbol{f}\in\mathcal{F}(p_n,s_n)} \frac{1}{n}\sum_{S'}\ell(\langle\mathcal{Y}_{y_i'},\hat{\boldsymbol{f}}(\boldsymbol{x}_i')\rangle) - \frac{1}{n}\sum_{S}\ell(\langle\mathcal{Y}_{y_i},\hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle)] \\
&= E_{S,S',\sigma}[\sup_{\boldsymbol{f}\in\mathcal{F}(p_n,s_n)} \frac{1}{n}\sum_{S'}\sigma_i\ell(\langle\mathcal{Y}_{y_i'},\hat{\boldsymbol{f}}(\boldsymbol{x}_i')\rangle) - \frac{1}{n}\sum_{S}\sigma_i\ell(\langle\mathcal{Y}_{y_i},\hat{\boldsymbol{f}}(\boldsymbol{x}_i)\rangle)] \\
&\leq 2R_n\mathcal{F}(p_n,s_n).
\end{aligned}$$

Hence the second part is proved.

Finally, we need to bound $R_n\mathcal{F}(p_n,s_n)$ in terms of $\hat{R}_n\mathcal{F}(p_n,s_n)$. This can be done in an analogous way as in the first part. In particular, apply the McDiarmid inequality on $\hat{R}_n\mathcal{F}(p_n,s_n)$ and its expectation $R_n\mathcal{F}(p_n,s_n)$. Similarly, we can show that with probability at least $1-\delta$, $R_n\mathcal{F}(p_n,s_n) \leq \hat{R}_n\mathcal{F}(p_n,s_n) + 2T_n(\delta)$. This completes the proof of the third part. The final results can be obtained by choosing the confidence $1-\delta/2$ in the first and third steps, and combining the inequalities of the three steps. $\square$

To prove Theorem 4.8.3, we need the Hoeffding's inequality.

Theorem 4.8.4. *(Hoeffding's Inequality). Let X be a random variable with mean 0 and range in $[a,b]$. Then for any fixed $z>0$, $E(e^{zX}) \leq e^{z^2(b-a)^2/8}$.*

Proof of Theorem 4.8.3: Define $R = \frac{\hat{R}_n\mathcal{F}(p_n,s_n)}{2(k-1)t(p_n,s_n)}$. Let

$$h_{\boldsymbol{f}}(\cdot) = (2(k-1)t(p_n,s_n))^{-1}\ell(\langle\boldsymbol{f},\mathcal{Y}_\cdot\rangle),$$

and $\mathcal{H} = \{h_{\boldsymbol{f}} : \boldsymbol{f} \in \tilde{\mathcal{F}}(p_n,s_n)\}$. Note that the entropy number of $\mathcal{H}$ can still be bounded by Lemma 4.8.2. Now construct an ϵ-net $\mathcal{G}$ of $\mathcal{H}$, such that for all $\boldsymbol{f}\in\mathcal{F}(p_n,s_n)$, there exists $\boldsymbol{g}\in\mathcal{F}(p_n)$, such that $h_{\boldsymbol{g}}\in\mathcal{G}$ and the $L_2(Q)$ distance between $h_{\boldsymbol{f}}$ and $h_{\boldsymbol{g}}$ is less than ϵ, for any arbitrary measure Q. Here without loss of generality we may assume $\boldsymbol{g}\in\mathcal{F}(p_n,s_n)$ because $\mathcal{F}(p_n,s_n)$ is convex. Hence, we have

$$\begin{aligned}
R \leq & \frac{1}{2(k-1)t(p_n,s_n)} E_\sigma[\sup_{\boldsymbol{g}} \frac{1}{n}\sum_{i=1}^n \sigma_i \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{g}(\boldsymbol{x}_i)\rangle)] \\
& + \frac{1}{2(k-1)t(p_n,s_n)} E_\sigma[\sup_{\boldsymbol{f},\boldsymbol{g} \text{ close}} \frac{1}{n}\sum_{i=1}^n \sigma_i \big(\ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{f}(\boldsymbol{x}_i)\rangle) - \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{g}(\boldsymbol{x}_i)\rangle)\big)] \\
:= & R_1 + R_2,
\end{aligned}$$

where by "$\boldsymbol{f},\boldsymbol{g}$ close" we mean that $h_{\boldsymbol{f}}$ is in the neighborhood of $h_{\boldsymbol{g}}$ with radius ϵ. Hence, we have that R_2 is less than ϵ, the L_2 distance between $h_{\boldsymbol{f}}$ and $h_{\boldsymbol{g}}$, because of the property of σ and the Hölder's inequality.

To bound R_1, let z be any fixed positive constant. By Jensen's inequality,

$$\begin{aligned}
\exp(2(k-1)t(p_n,s_n)znR_1) \leq & E_\sigma \exp[z \sup_{\boldsymbol{g}} \sum_{i=1}^n \sigma_i \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{g}(\boldsymbol{x}_i)\rangle)] \\
\leq & \sum_{\boldsymbol{g}} E_\sigma[\exp(z\sum_{i=1}^n \sigma_i \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{g}(\boldsymbol{x}_i)\rangle))] \\
\leq & \sum_{\boldsymbol{g}} \prod_{i=1}^n E_{\sigma_i}[\exp(z\sigma_i \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{g}(\boldsymbol{x}_i)\rangle))].
\end{aligned}$$

Note that $E_{\sigma_i}[\sigma_i \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{g}(\boldsymbol{x}_i)\rangle)] = 0$, and

$$\ell(0) - t(p_n,s_n) \leq \ell(\langle \mathcal{Y}_{y_i}, \boldsymbol{g}(\boldsymbol{x}_i)\rangle) \leq \ell(0) + t(p_n,s_n).$$

By Hoeffding's inequality, we have

$$\begin{aligned}
\exp(2(k-1)t(p_n,s_n)znR_1) \leq & \sum_{\boldsymbol{g}} \prod_{i=1}^n \exp(\frac{z^2(2t(p_n,s_n))^2}{8}) \\
\leq & |\mathcal{G}| \exp(\frac{nz^2 t(p_n,s_n)^2}{2}).
\end{aligned}$$

Hence, $2(k-1)t(p_n,s_n)nR_1 \leq \frac{\log|\mathcal{G}|}{z} + \frac{nzt(p_n,s_n)^2}{2}$. Let $z = \frac{\sqrt{2\log|\mathcal{G}|}}{t(p_n,s_n)\sqrt{n}}$, and we have

$$2(k-1)t(p_n,s_n)nR_1 \leq t(p_n,s_n)\sqrt{2n\log|\mathcal{G}|}.$$

Rewrite the above display,

$$R_1 \le \frac{\sqrt{2\log|\mathcal{G}|}}{2(k-1)\sqrt{n}} \le \frac{1}{(k-1)\epsilon}\sqrt{\frac{\log(e+e(2p_n(k-1))\epsilon^2)}{n}}.$$

Combining the upper bounds of R_1 and R_2, we have

$$\hat{R}_n\mathcal{F}(p_n,s_n) \le 2(k-1)t(p_n,s_n)[\epsilon + \frac{1}{(k-1)\epsilon}\sqrt{\frac{\log(e+e(2p_n(k-1))\epsilon^2)}{n}}].$$

Choose $\epsilon = n^{-1/4}$ and this completes the proof. $\square$

CHAPTER 5: REINFORCED ANGLE-BASED SVM

5.1 Introduction

Although the angle-based classification in Chapter 4 does not require the sum-to-zero constraint, the direct generalization of SVM in the angle-based classification structure is not Fisher consistent. Therefore, it is desirable to develop an MSVM classifier in the angle-based classification framework with Fisher consistency. To this end, we propose the Reinforced Angle-based Multicategory Support Vector Machine (RAMSVM) in this chapter, which uses convex combinations of loss functions as in Liu and Yuan [2011]. In particular, we first explain how to modify existing MSVM losses in the angle-based classification framework, then introduce our RAMSVM loss family as a convex combination. We show that with a proper choice of the convex combination parameter, the new RAMSVM classifier enjoys Fisher consistency. Furthermore, since RAMSVM is free of the sum-to-zero constraint, the corresponding optimization is shown to be equivalent to an objective function in a quadratic polynomial form with box constraints. This can be solved using the very efficient coordinate descent method [Tseng, 2001; Fan et al., 2008; Friedman, Hastie and Tibshirani, 2010*b*]. In contrast, most other existing MSVMs employ the sum-to-zero constraint [Zhang and Liu, 2013], and use Quadratic Programming (QP) for the corresponding optimization [Lee, Lin and Wahba, 2004; Liu and Yuan, 2011], which typically relies on existing QP solvers. Compared to these MSVM methods, our new RAMSVM can enjoy a much faster computational speed. We confirm this finding in Section 5.5. Furthermore, we show that our suggested RAMSVM can often outperform several other competitors in terms of classification accuracy.

The rest of this chapter is organized as follows. In Section 5.2, we introduce our RAMSVM classifier. In Section 5.3, we study Fisher consistency of the RAMSVM family. We develop the coordinate descent algorithm for solving the corresponding optimization in Section 5.4. Numerical studies with simulated and real data sets are performed in Section 5.5. Some discussions are provided in Section 5.6. All proofs are collected afterwards.

5.2 Methodology

First we remind the notations and existing MSVM methods for completeness. For a multicategory classification problem with k classes, let $(\boldsymbol{x}_i, y_i);\ i = 1, \ldots, n$ be the observed training data points. Here $\boldsymbol{x}$ denotes the p-dimensional predictor vector, and $y \in \{1, \ldots, k\}$ is the corresponding label. The regular simultaneous multicategory large-margin classifiers use a k-dimensional classification function $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))^T$, and the prediction rule is $\hat{y}(\boldsymbol{x}) = \operatorname{argmax}_{j \in \{1,\ldots,k\}} f_j(\boldsymbol{x})$. The corresponding optimization can typically be written as

$$\min_{\boldsymbol{f} \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} V(\boldsymbol{f}(\boldsymbol{x}_i), y_i) + \frac{1}{2} \lambda J(\boldsymbol{f}), \tag{5.1}$$

where $\mathcal{F}$ is the function class we are interested in, $J(\cdot)$ is a penalty that controls the complexity of $\mathcal{F}$ to prevent overfitting, and λ is the tuning parameter that controls the balance between the loss and the penalty terms. Here $V(\boldsymbol{f}(\boldsymbol{x}), y)$ measures the loss of using $\boldsymbol{f}(\boldsymbol{x})$ as the classification function for $(\boldsymbol{x}, y)$. Different loss functions correspond to different classifiers. In the simultaneous MSVM literature, the following loss functions are commonly used as extensions from binary SVMs to MSVMs:

MSVM 1 (Naive hinge loss) $V(\boldsymbol{f}(\boldsymbol{x}), y) = [1 - f_y(\boldsymbol{x})]_+$;

MSVM 2 [Vapnik, 1998] $V(\boldsymbol{f}(\boldsymbol{x}), y) = \sum_{j \neq y} [1 - (f_y(\boldsymbol{x}) - f_j(\boldsymbol{x}))]_+$;

MSVM 3 [Crammer et al., 2001; Liu, Shen and Doss, 2005] $V(\boldsymbol{f}(\boldsymbol{x}), y) = \sum_{j \neq y} [1 - \min_j (f_y(\boldsymbol{x}) - f_j(\boldsymbol{x}))]_+$;

MSVM 4 [Lee, Lin and Wahba, 2004] $V(\boldsymbol{f}(\boldsymbol{x}), y) = \sum_{j \neq y} [1 + f_j(\boldsymbol{x})]_+$;

MSVM 5 [Liu and Yuan, 2011] $V(\boldsymbol{f}(\boldsymbol{x}), y) = \gamma[(k-1) - f_y(\boldsymbol{x})]_+ + (1-\gamma) \sum_{j \neq y} [1 + f_j(\boldsymbol{x})]_+$;

where $[u]_+ = \max(u, 0)$ and $\gamma \in [0, 1]$ in MSVM 5 is the convex combination parameter.

Although binary SVM is known to be Fisher consistent, the MSVMs 1-3 are not. To overcome this challenge, Lee, Lin and Wahba [2004] proposed the MSVM method 4 which was shown to be Fisher consistent. Furthermore, Liu and Yuan [2011] proposed the Reinforced MSVM (RMSVM, MSVM 5) method, which uses a convex combination of the naive hinge loss (MSVM 1) and the loss for MSVM 4 as a new class of loss functions. RMSVM was shown to be Fisher consistent when $\gamma \in [0, 0.5]$.

As discussed in Chapter 4, it is inefficient to train a multicategory classifier with k classification functions and the sum-to-zero constraint. To overcome this difficulty, we proposed the multicategory angle-based large-margin classification technique. However, when using the regular hinge loss $\ell(u) = [1-u]_+$, the corresponding angle-based SVM is not Fisher consistent. Therefore, it is desirable to have a generalization of the binary SVM classifier in the angle-based classification framework which enjoys Fisher consistency. Note that the RMSVM method achieves Fisher consistency by using a convex combination of loss functions. In particular, one part of the loss function encourages $f_y(\boldsymbol{x})$ to be as large as possible, and the other encourages $f_j(\boldsymbol{x});\ j \neq y$ to be as small as possible. Motivated by this convex combination idea, we propose the following Reinforced Angle-based Multicategory SVM (RAMSVM) classifier

$$\min_{\boldsymbol{f}\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\Big\{(1-\gamma)\sum_{j\neq y_i}[1+\langle \boldsymbol{f}(\boldsymbol{x}_i),\mathcal{Y}_j\rangle]_+ + \gamma[(k-1)-\langle \boldsymbol{f}(\boldsymbol{x}_i),\mathcal{Y}_{y_i}\rangle]_+\Big\} + \frac{\lambda}{2}J(\boldsymbol{f}), \tag{5.2}$$

where $\gamma \in [0,1]$ is the convex combination parameter. Note that the first part in the loss term of (5.2) can be regarded as a modified MSVM loss of Lee, Lin and Wahba [2004] in the angle-based classification framework. More generally, the inner products $\langle \boldsymbol{f}(\boldsymbol{x}),\mathcal{Y}_j\rangle;\ j=1,\ldots,k$ can be regarded as new functional margins of $(\boldsymbol{x},y)$. Many other existing MSVMs can be generalized into the angle-based classification framework in a similar manner.

The motivation for using such a combination of loss functions in (5.2) is based on the prediction rule of the angle-based classification. Since the least angle prediction rule

$$\hat{y}(\boldsymbol{x}) = \operatorname*{argmin}_{j\in\{1,\ldots,k\}} \angle(\boldsymbol{f}(\boldsymbol{x}),\mathcal{Y}_j)$$

is equivalent to $\hat{y}(\boldsymbol{x}) = \mathrm{argmax}_{j\in\{1,\ldots,k\}}(\langle\mathcal{Y}_j,\boldsymbol{f}(\boldsymbol{x})\rangle)$, we need to have $\langle \boldsymbol{f}(\boldsymbol{x}),\mathcal{Y}_y\rangle$ to be the maximum among all the k inner products $\langle \boldsymbol{f}(\boldsymbol{x}),\mathcal{Y}_j\rangle;\ j=1,\ldots,k$. To that end, the second part in the loss term of (5.2) encourages $\langle \boldsymbol{f}(\boldsymbol{x}),\mathcal{Y}_y\rangle$ to be large. On the other hand, observe that $\sum_{j=1}^{k}\langle \boldsymbol{f}(\boldsymbol{x}),\mathcal{Y}_j\rangle = 0$ for all $\boldsymbol{x}$, which means that the angle-based classification framework transfers the explicit sum-to-zero constraint in the previous formulation onto the inner products. Hence, as the first part in the loss term of (5.2) encourages $\langle \boldsymbol{f}(\boldsymbol{x}),\mathcal{Y}_j\rangle;\ j\neq y$ to be small, it implicitly encourages $\langle \boldsymbol{f}(\boldsymbol{x}),\mathcal{Y}_y\rangle$ to be large.

Since most existing MSVMs can be cast into QP problems, they are typically implemented using existing QP solvers. For RAMSVM, we show in Section 5.4 that it can be solved using the coordinate descend method, which can be very fast. Compared to other MSVM implementations using QP solvers, our RAMSVM can enjoy a much faster computational speed. We demonstrate the computational advantage of RAMSVM using numerical examples in Section 5.5.

In the next section, we introduce the details of Fisher consistency, and show that RAMSVM is Fisher consistent when $\gamma \in [0, 1/2]$.

5.3 Fisher Consistency

Recall from Chapter 4 that for angle-based classifiers, Fisher consistency requires that

$$\operatorname*{argmax}_{j \in \{1,\ldots,k\}} \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle = \operatorname*{argmax}_{j \in \{1,\ldots,k\}} P_j(\boldsymbol{x}),$$

where $\boldsymbol{f}^*(\boldsymbol{x})$ is the minimizer of the conditional loss $E[V(\boldsymbol{f}(\boldsymbol{X}), Y)|\boldsymbol{X} = \boldsymbol{x}]$, and when

$$\operatorname*{argmax}_{j \in \{1,\ldots,k\}} P_j(\boldsymbol{x})$$

is unique, so is $\operatorname{argmax}_{j \in \{1,\ldots,k\}} \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle$. For the RAMSVM classifier (5.2), we study its Fisher consistency in the following theorem.

Theorem 5.3.1. *For multicategory classification problems with $k > 2$, the RAMSVM loss function (5.2) is Fisher consistent when $\gamma \in [0, 0.5]$, and is not Fisher consistent when $\gamma \in (0.5, 1]$.*

From Theorem 5.3.1, we can conclude that the proposed RAMSVM provides a large family of consistent MSVM classifiers with $\gamma \in [0, 1/2]$. For $\gamma > 1/2$, the Fisher consistency cannot be guaranteed. In Section 5.5, we study the effect of different γ on the classification accuracy. Interestingly, we observed that the Fisher consistent RAMSVM with $\gamma = 1/2$ can provide a robust classifier with competitive classification accuracy. In particular, the numerical results show that RAMSVM with $\gamma = 0$ or $\gamma = 1$ can be suboptimal in certain cases, while $\gamma = 1/2$ gives a robust classifier whose performance is close to be optimal in many situations. This demonstrates the advantage of the convex combination of SVM loss functions in the angle-based classification structure.

In the next section, we develop an efficient algorithm to solve (5.2). In particular, we show that RAMSVM can be solved using the coordinate descend method, and within each coordinate-wise optimization procedure the update value can be calculated explicitly. This greatly boosts the computational speed.

5.4 Algorithm

In this section, we show how to solve the optimization problem (5.2). We demonstrate that with the intercepts penalized in linear learning or kernel learning as in Fan et al. [2008], RAMSVM can be solved using the coordinate descend method [Tseng, 2001; Friedman, Hastie and Tibshirani, 2010*b*] and consequently enjoys a fast computational speed. First, we focus on linear learning with the commonly used L_2 penalty. Then we develop the algorithm for kernel learning. The discussion on how to solve weighted learning problems is provided afterwards.

5.4.1 Linear Learning

For linear learning, we assume $f_q(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta}_q;\ q = 1, \ldots, k-1$, where $\boldsymbol{\beta}_q;\ q = 1, \ldots, k-1$ are the parameters to estimate. The penalty term $J(\boldsymbol{f})$ can be written as $J = \sum_{q=1}^{k-1} \boldsymbol{\beta}_q^T\boldsymbol{\beta}_q$. Note that we include the intercepts in $\boldsymbol{x}$ to simplify notations.

We solve (5.2) in its dual form. Using new slack variables $\xi_{i,j};\ i = 1, \ldots, n, j = 1, \ldots, k$, (5.2) with linear learning can be written as

$$
\begin{aligned}
\min_{\boldsymbol{\beta}_q, \xi_{i,j}} \ & \frac{n\lambda}{2} \sum_{q=1}^{k-1} \boldsymbol{\beta}_q^T\boldsymbol{\beta}_q + \sum_{i=1}^{n} [\gamma \xi_{i,y_i} + (1-\gamma) \sum_{j \neq y_i} \xi_{i,j}], \\
\text{s.t. } & \xi_{i,j} \geq 0;\ i = 1, \ldots, n,\ j = 1, \ldots, k, \\
& \xi_{i,y_i} + \langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i} \rangle - (k-1) \geq 0;\ i = 1, \ldots, n, \\
& \xi_{i,j} - \langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_j \rangle - 1 \geq 0;\ i = 1, \ldots, n,\ j \neq y_i.
\end{aligned}
$$

Define the corresponding Lagrangian function L as

$$L = \frac{n\lambda}{2}\sum_{q=1}^{k-1}\boldsymbol{\beta}_q^T\boldsymbol{\beta}_q - \sum_{i=1}^{n}\sum_{j=1}^{k}\tau_{i,j}\xi_{i,j} + \sum_{i=1}^{n}[\gamma\xi_{i,y_i} + (1-\gamma)\sum_{j\neq y_i}\xi_{i,j}]$$
$$-\sum_{i=1}^{n}\alpha_{i,y_i}[\xi_{i,y_i} + \langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i}\rangle - (k-1)] - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}[\xi_{i,j} - \langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_j\rangle - 1].$$

Here $\alpha_{i,j}$ and $\tau_{i,j};\ i = 1,\ldots,n, j = 1,\ldots,k$ are the Lagrangian multipliers. One can verify that L can be rewritten as

$$L = \frac{n\lambda}{2}\sum_{q=1}^{k-1}\boldsymbol{\beta}_q^T\boldsymbol{\beta}_q - \sum_{i=1}^{n}\alpha_{i,y_i}\langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i}\rangle + \sum_{i=1}^{n}\alpha_{i,y_i}(k-1) + \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}$$
$$+\sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_j\rangle + \sum_{i=1}^{n}\sum_{j=1}^{k}[A_{i,j} - \tau_{i,j} - \alpha_{i,j}]\xi_{i,j}, \tag{5.3}$$

where $A_{i,j} = [\gamma I(j = y_i) + (1-\gamma)I(j \neq y_i)]$.

After taking partial derivative of L with respect to $\boldsymbol{\beta}_q;\ q = 1,\ldots,k-1$ and $\xi_{i,j};\ i = 1,\ldots,n, j = 1,\ldots,k$, we have

$$\frac{\partial L}{\partial \xi_{i,j}} = A_{i,j} - \tau_{i,j} - \alpha_{i,j} = 0;\ i = 1,\ldots,n, j = 1,\ldots,k,$$
$$\frac{\partial L}{\partial \boldsymbol{\beta}_q} = n\lambda\boldsymbol{\beta}_q - \sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q}\boldsymbol{x}_i + \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}\boldsymbol{x}_i = 0;\ q = 1,\ldots,k-1,$$

where $\mathcal{Y}_{j,q}$ represents the q^{th} element of $\mathcal{Y}_j$. Hence

$$\boldsymbol{\beta}_q = \frac{1}{n\lambda}[\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q}\boldsymbol{x}_i - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}\boldsymbol{x}_i]. \tag{5.4}$$

Plug (5.4) in L, and one can obtain that, after simplification,

$$L = -\frac{n\lambda}{2}\sum_{q=1}^{k-1}\boldsymbol{\beta}_q^T\boldsymbol{\beta}_q + \sum_{i=1}^{n}\alpha_{i,y_i}(k-1) + \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j},$$

where $\boldsymbol{\beta}_q$ is given in (5.4). Now take the negative of L and the dual form of (5.2) can be expressed as

$$\min_{\alpha_{i,j}} \; -\sum_{i=1}^{n}\alpha_{i,y_i}(k-1)-\sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}+\frac{1}{2n\lambda}\sum_{q=1}^{k-1}[\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q}\boldsymbol{x}_i-\sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}\boldsymbol{x}_i]^T$$
$$[\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q}\boldsymbol{x}_i-\sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}\boldsymbol{x}_i]$$
$$\text{s.t. } 0\leq\alpha_{i,j}\leq A_{i,j};\; i=1,\ldots,n,j=1,\ldots,k. \tag{5.5}$$

Note that $\boldsymbol{\beta}_q$ is a linear vector of $\alpha_{i,j}$'s, and hence L is a quadratic term of $\{\alpha_{i,j};\; i=1,\ldots,n,j=1,\ldots,k\}$. One can verify that $\partial^2(-L)/\partial\alpha_{i,j}^2 = \boldsymbol{x}_i^T\boldsymbol{x}_i/(n\lambda) > 0$, for all i and j. Moreover, the constraints in (5.5) are box constraints. Therefore, the dual optimization (5.5) is a strictly convex problem and can be solved by the well known coordinate descend method. Furthermore, because the objective function is quadratic, within each step of coordinate-wise optimization the next update value can be calculated explicitly. This greatly boosts the computational speed. Compared to the regular MSVMs with the sum-to-zero constraint that use QP [Lee, Lin and Wahba, 2004; Liu and Yuan, 2011], our proposed RAMSVM enjoys a much faster computational speed. In Section 5.5, we show that RAMSVM can outperform the MSVMs by Lee, Lin and Wahba [2004] and Liu and Yuan [2011] in terms of computational speed.

5.4.2 Kernel Learning

Next, we briefly discuss the case with kernel learning. We show that with the regular squared norm regularization and the intercepts penalized, the optimization can also be solved using the coordinate descend method. To begin with, denote by K the corresponding kernel function, and by $\boldsymbol{K} = \big(K(\boldsymbol{x}_i,\boldsymbol{x}_{i'})\big)_{i=1,\ldots,n,i'=1,\ldots,n}$ the gram matrix. Define $A_{i,j}$ as in the linear case. If the penalty we choose is the squared norm penalty in the corresponding kernel space [see, for example, Shawe-Taylor and Cristianini, 2004, for details], then by the Representer Theorem [Kimeldorf and Wahba, 1971], we have $f_q(\boldsymbol{x}) = \theta_{q,0}+\sum_{i=1}^{n}\theta_{q,i}K(\boldsymbol{x}_i,\boldsymbol{x})$ and $J(\boldsymbol{f}) = \sum_{q=1}^{k-1}\boldsymbol{\theta}_q^T\boldsymbol{K}\boldsymbol{\theta}_q$. Here $\boldsymbol{\theta}_q = (\theta_{q,1},\ldots,\theta_{q,n})^T$ is the kernel product coefficient vector, for $q=1,\ldots,k-1$. Introduce the slack variables $\xi_{i,j}$ as in the linear case.

If we penalize on the intercepts $\theta_{q,0};\; q=1,\ldots,k-1$ as well, we have that (5.2) is equivalent to

$$
\begin{aligned}
\min_{\boldsymbol{\theta}_q,\theta_{q,0},\xi_{i,j}} \; & \frac{n\lambda}{2}\sum_{q=1}^{k-1}\boldsymbol{\theta}_q^T K\boldsymbol{\theta}_q + \frac{n\lambda}{2}\sum_{q=1}^{k-1}\theta_{q,0}^2 + \sum_{i=1}^{n}[\gamma\xi_{i,y_i} + (1-\gamma)\sum_{j\neq y_i}\xi_{i,j}],\\
\text{s.t. } & \xi_{i,j} \geq 0;\; i=1,\dots,n,\; j=1,\dots,k,\\
& \xi_{i,y_i} + \langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i}\rangle - (k-1) \geq 0;\; i=1,\dots,n,\\
& \xi_{i,j} - \langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_j\rangle - 1 \geq 0;\; i=1,\dots,n,\; j\neq y_i. \qquad (5.6)
\end{aligned}
$$

Next, we introduce the Lagrangian multipliers $\tau_{i,j}$ and $\alpha_{i,j}$, take partial derivative with respect to $\boldsymbol{\theta}_q$, $\theta_{q,0}$ and $\xi_{i,j}$ and set to zero. Without loss of generality, assume that the gram matrix $\boldsymbol{K}$ is invertible. We have that

$$
\boldsymbol{\theta}_q = \frac{1}{n\lambda}\boldsymbol{K}^{-1}[\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q}\boldsymbol{K}_i - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}\boldsymbol{K}_i], \qquad (5.7)
$$

$$
\theta_{q,0} = \frac{1}{n\lambda}[\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q} - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}], \qquad (5.8)
$$

where $\boldsymbol{K}_i$ is the i^{th} column of $\boldsymbol{K}$. After plugging (5.7) and (5.8) in (5.6), (5.2) can be shown to be equivalent to

$$
\begin{aligned}
\min_{\alpha_{i,j}} \; & \frac{1}{2n\lambda}\sum_{q=1}^{k-1}[\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q}\boldsymbol{K}_i - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}\boldsymbol{K}_i]^T\boldsymbol{K}^{-1}\\
& [\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q}\boldsymbol{K}_i - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}\boldsymbol{K}_i]\\
& + \frac{1}{2n\lambda}\sum_{q=1}^{k-1}[\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q} - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}]^2 - \sum_{i=1}^{n}\alpha_{i,y_i}(k-1) - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j},\\
\text{s.t. } & 0 \leq \alpha_{i,j} \leq A_{i,j};\; i=1,\dots,n, j=1,\dots,k. \qquad (5.9)
\end{aligned}
$$

Note that $\boldsymbol{K}_i^T\boldsymbol{K}^{-1}\boldsymbol{K}_j = K(\boldsymbol{x}_i,\boldsymbol{x}_j)$. One can verify that (5.9) can be solved in an analogous manner as (5.5).

5.4.3 Weighted Learning

So far we have focused on the optimization problem with equal weights of loss on different classes. In practice, it is prevalent to have one class whose size is significantly larger than that of another class. For example, in cancer research, the sample size of patients with a rarely seen cancer can be very limited, while the number of healthy samples is large. In this case, standard classification without adjusting for the unbalanced sample sizes can lead to a suboptimal result. To overcome this difficulty, it has been proposed to use different weights of loss for different classes [Qiao and Liu, 2009]. This weighted learning technique can also be applied to practical problems with possible biased sampling. Therefore, it is desirable to study the extension from standard classification to weighted learning. Next, we use linear learning as an example and demonstrate how to solve the corresponding optimization problems with given weights on different observations.

Let the weight of loss for the ith observation be w_i. Assume that $w_i > 0;\ i = 1, \ldots, n$. In weighted learning, the optimization (5.2) becomes

$$\min_{\boldsymbol{f}\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} w_i\Big\{(1-\gamma)\sum_{j\neq y_i}[1+\langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_j\rangle]_+ + \gamma[(k-1) - \langle \boldsymbol{f}(\boldsymbol{x}_i), \mathcal{Y}_{y_i}\rangle]_+\Big\} + \frac{\lambda}{2}J(\boldsymbol{f}),$$

which, after some calculation, can be rewritten as

$$\begin{aligned}\min_{\alpha_{i,j}}\ & -\sum_{i=1}^{n}\alpha_{i,y_i}(k-1) - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j} + \frac{1}{2n\lambda}\sum_{q=1}^{k-1}[\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q}\boldsymbol{x}_i - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}\boldsymbol{x}_i]^T \\ & [\sum_{i=1}^{n}\alpha_{i,y_i}\mathcal{Y}_{y_i,q}\boldsymbol{x}_i - \sum_{i=1}^{n}\sum_{j\neq y_i}\alpha_{i,j}\mathcal{Y}_{j,q}\boldsymbol{x}_i] \\ & \text{s.t. } 0 \leq \alpha_{i,j} \leq \bar{A}_{i,j};\ i = 1,\ldots,n, j = 1,\ldots,k, \end{aligned} \tag{5.10}$$

where $\bar{A}_{i,j} = w_i[\gamma I(j = y_i) + (1-\gamma)I(j \neq y_i)]$. Note that the difference between (5.5) and (5.10) is in the definition of $A_{i,j}$ and $\bar{A}_{i,j}$. Because $\bar{A}_{i,j} \geq 0$, the optimization (5.10) can be solved in a similar manner as (5.5).

In the next section, we demonstrate the performance of RAMSVM via several simulated examples and a real data set. As we will see, the algorithm developed in this section provides a very efficient way to solve the corresponding optimization problem.

5.5 Numerical Results

In this section, we explore the numerical performance of RAMSVM. In particular, in Section 5.5.1, we compare the performance of RAMSVM with RMSVM (MSVM 5) and the MSVM by Lee, Lin and Wahba [2004] (MSVM 4) via two simulated examples, and in Section 5.5.2, we study a real world data set. For RMSVM, we follow the suggestion of Liu and Yuan [2011] and use $\gamma = 0.5$. For RAMSVM, we demonstrate the effect of γ on the classification accuracy. We show that RAMSVM enjoys a faster computational speed. Furthermore, we show that the Fisher consistent RAMSVM with $\gamma = 0.5$ yields a robust MSVM method, which is highly competitive.

5.5.1 Simulated Examples

We consider two simulated examples in this section. The first example is constructed such that linear learning suffices, and we apply linear learning as well as the Gaussian kernel learning. For the second example, we design the marginal distribution of $\boldsymbol{X}$ such that linear classification boundary cannot separate the classes well. Therefore, we use the second order polynomial kernel and the Gaussian kernel for this example. For Gaussian kernel learning, the kernel parameter σ is chosen to be the median of all the pairwise distances between one category and others.

We let the convex combination parameter γ vary in $\{0, 0.1, 0.2, \ldots, 0.9, 1\}$ and study the effect of γ on the classification accuracy for RAMSVM. To select the best tuning parameter λ, we use a grid search. In particular, we train the classifiers on a training data set. The classifier that has the smallest prediction error rate on an independent tuning data set is then selected, and we record the prediction error rate on a separate testing data set with size 10^6. We repeat this procedure 1000 times and report the average prediction error rates on the testing data set. To compare the computational speed among different methods, in each replication we record the total time of training the classifiers for 50 different tuning parameters, selecting the best parameter λ and predicting on the testing data set. We report the boxplots of time used for the 1000 replications. For RAMSVM, we only report the time for $\gamma = 1/2$, as the differences in terms of computational time for various γ values are small. All simulations are done with R [R Core Team, 2013] on a 2.80 GHz Intel processor.

Example 1: This is an eight class example with equal probabilities $Pr(Y = j)$; $j = 1, \ldots, 8$. The marginal distribution of $\boldsymbol{X}$ for each class is normal, and the mean vectors for different classes form a simplex in $\mathbb{R}^7$. We choose the covariance matrices of the normal distributions so that the corresponding Bayes classification error is 5%. We use 150 observations for training and another 150 for tuning.

Example 2: We generate four classes with $Pr(Y = j) = 1/4;\ j = 1, \ldots, 4$ on $\mathbb{R}^2$. For each class, the predictor vector $\boldsymbol{X}|Y = j$ follows a mixed normal distribution. In particular, for class j, $\boldsymbol{X}$ follows $0.5N((\cos(j\pi/4), \sin j(\pi/4))^T, \Sigma) + 0.5N((\cos(j\pi/4 + \pi), \sin j(\pi/4 + \pi))^T, \Sigma)$. Here Σ is chosen such that the Bayes error is 5%. Both the training and tuning data are of size 150.

The simulation results are reported in Figures 5.1, 5.2, 5.3 and 5.4. From Figures 5.3 and 5.4, we can see that RAMSVM with the coordinate descend method enjoys a very fast computational speed, compared to the MSVM 4 and RMSVM with the sum-to-zero constraint using QP. In Figure 5.1, one can see that for linear learning, the classification error rate decreases as γ increases from 0 to 1/2. When γ ranges in $[1/2, 1]$, the change in classification accuracy is small. However, for kernel learning, the pattern is reversed, as shown in Figures 5.1 and 5.2. In particular, the classification error rates increase as γ increases from 1/2 to 1, and when γ is in $[0, 1/2]$, the classification accuracy does not change much. From Figures 5.1 and 5.2, one can conclude that the classification accuracy of RAMSVM with $\gamma = 1/2$ is close to the optimum for all situations, and hence choosing $\gamma = 1/2$ provides a Fisher consistent and robust classifier. We recommend using $\gamma = 1/2$ for all classification problems. Note that the classification accuracy of RAMSVM with $\gamma = 1/2$ is highly competitive compared to the existing MSVMs.

5.5.2 Real Data Analysis

In this section, we test the performance of RAMSVM using the Vertebral Column data set, which can be found on the UCI machine learning website [Bache and Lichman, 2013]. There are three classes of patients, namely, normal, disk hernia and spondilolysthesis, in the data, with corresponding sizes 100, 60 and 150, respectively. Because the three classes are of different sizes, the weighted learning technique is applied here. We choose the second order polynomial kernel and the Gaussian kernel for learning. The Gaussian kernel parameter is chosen in an analogous way as in Section 5.5.1. For each replication, we use approximately 5/6 of the data as the training data, and the rest as the testing data set. The best tuning parameter is selected via a 5-fold cross validation within the training data.

The classification error rates are reported in Figure 5.5. The error rate for $\gamma = 0.8$ appears to be the smallest. However, RAMSVM with $\gamma = 1/2$ provides a Fisher consistent classifier whose performance is robust and close to the optimum, and hence is recommended. The comparison of computational speed shows a similar result as in the simulation section, and is omitted.

5.6 Discussion and Future Direction

In this chapter, we propose RAMSVM as a new angle-based MSVM method. We show that RAMSVM has two advantages. First, it is free of the sum-to-zero constraint, and hence the corresponding optimization procedure is more efficient. In particular, we develop a new algorithm to train the classifier using coordinate descend method, which is free of QP solvers, hence very fast. Second, RAMSVM overcomes the difficulty of inconsistency, compared to the angle-based SVM method. Numerical comparison between RAMSVM and some existing MSVMs demonstrate the usefulness of our method. We recommend using RAMSVM with $\gamma = 1/2$, which yields a Fisher consistent classifier whose performance is robust and often close to the optimum.

One open direction is to derive further theoretical properties for RAMSVM such as finite sample error bound on the classification error rates [Zhang, 2004*b*; Bartlett, Jordan and McAuliffe, 2006]. It will be interesting to explore the effect of γ on these results.

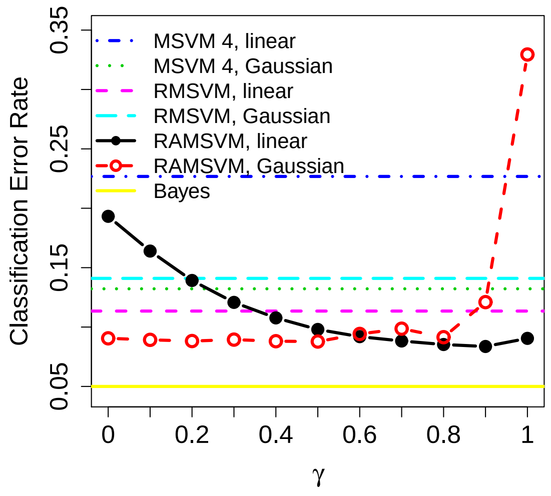


Figure 5.1: The effect of different γ values on the classification performance of RAMSVM for simulated Example 1, along with the comparison to some existing MSVM methods. The standard errors of the classification error rates in Example 1 range from 0.001 to 0.012.

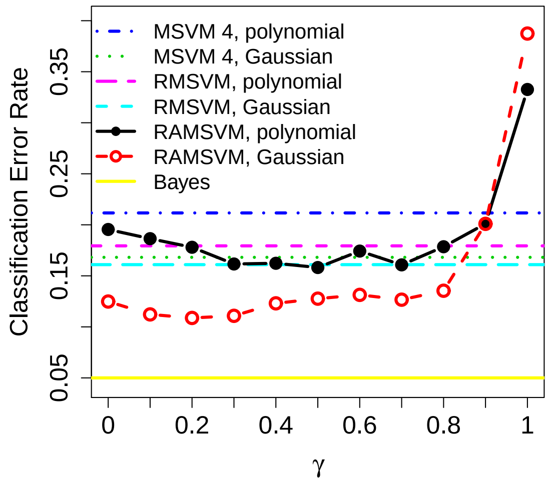


Figure 5.2: The effect of different γ values on the classification performance of RAMSVM for simulated Example 2, along with the comparison to some existing MSVM methods. The standard errors of the classification error rates in Example 2 range from 0.001 to 0.010.

5.7 Proofs

We prove Theorem 5.3.1 in this section. The proof consists of two parts. First we show that with $\gamma \leq 1/2$ the RAMSVM is Fisher consistent. Then we show that when $\gamma > 0$, the Fisher consistency cannot be guaranteed.

In this proof we assume $P_1 > P_2 \geq \cdots \geq P_k$. We need to show that $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle > \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle$ for $j \neq 1$. First, we show that $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle \geq \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle$ for any j. Note that if this is not true, then by Lemma 4.8.1, there exists $\boldsymbol{f}'(\boldsymbol{x}) \in \mathbb{R}^{k-1}$ such that $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle = \langle \boldsymbol{f}'(\boldsymbol{x}), \mathcal{Y}_j \rangle$ and $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle = \langle \boldsymbol{f}'(\boldsymbol{x}), \mathcal{Y}_1 \rangle$. One can verify that $E[V(\boldsymbol{f}^*(\boldsymbol{X}), Y)|\boldsymbol{X} = \boldsymbol{x}] > E[V(\boldsymbol{f}'(\boldsymbol{X}), Y)|\boldsymbol{X} = \boldsymbol{x}]$, which contradicts to the definition of $\boldsymbol{f}^*$.

Next, we show that $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle \leq k-1$. Note that we have $\sum_{j=1}^{k} \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle = 0$. If $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle > k-1$, then there exists q such that $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_q \rangle < -1$. By Lemma 4.8.1, there exists $\boldsymbol{f}'(\boldsymbol{x})$ such that $\langle \boldsymbol{f}'(\boldsymbol{x}), \mathcal{Y}_j \rangle = \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle$ for $j \neq 1, q$, $\langle \boldsymbol{f}'(\boldsymbol{x}), \mathcal{Y}_1 \rangle = \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle - \epsilon$, and $\langle \boldsymbol{f}'(\boldsymbol{x}), \mathcal{Y}_q \rangle = \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_q \rangle + \epsilon$, where ϵ is a small positive number. Now $E[V(\boldsymbol{f}^*(\boldsymbol{X}), Y)|\boldsymbol{X} = \boldsymbol{x}] - E[V(\boldsymbol{f}'(\boldsymbol{X}), Y)|\boldsymbol{X} = \boldsymbol{x}] = \{(1-P_1)(1-\gamma) + P_k\gamma\}\epsilon > 0$, which is a contradiction.

Next, we show that if $\gamma \leq 1/2$ then $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle \geq -1$. Suppose this is not true and $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle < -1$ for fixed $j \neq 1$. Because the inner product $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle \leq k-1$ is the maximum, we have

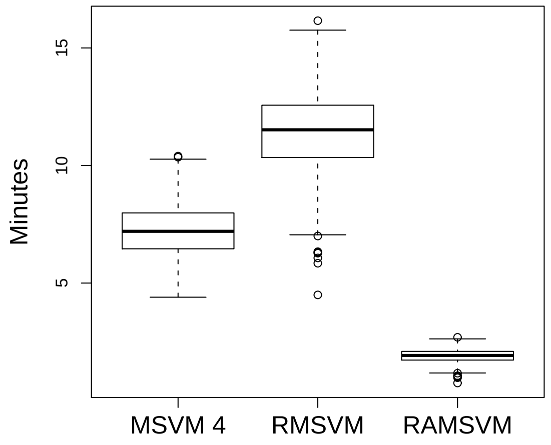


Figure 5.3: The boxplots of computational time in one replication (in minutes) for MSVM 4, RMSVM, and our new RAMSVM in Example 1 with linear learning.

that $-1 < \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_q \rangle \leq k-1$ for some q. Define $\boldsymbol{f}'(\boldsymbol{x})$ such that $\langle \boldsymbol{f}'(\boldsymbol{x}), \mathcal{Y}_i \rangle = \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_i \rangle$ for $i \neq j, q$, $\langle \boldsymbol{f}'(\boldsymbol{x}), \mathcal{Y}_q \rangle = \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_q \rangle - \epsilon$ and $\langle \boldsymbol{f}'(\boldsymbol{x}), \mathcal{Y}_j \rangle = \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle + \epsilon$. One can verify that $E[V(\boldsymbol{f}'(\boldsymbol{X}), Y)|\boldsymbol{X} = \boldsymbol{x}] - E[V(\boldsymbol{f}^*(\boldsymbol{X}), Y)|\boldsymbol{X} = \boldsymbol{x}] = [P_q - 1 + (1-P_j)\gamma]\epsilon$. As $\gamma \leq 1/2$, $[P_q - 1 + (1-P_j)\gamma] < 0$, hence this is a contradiction. Thus we have $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle \geq -1$.

Lastly, using the above results and an argument similar to Lemma A.2 in Liu and Yuan [2011], we have that $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle = k-1$ and $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_j \rangle = -1;\ j \neq 1$. This completes the first part of the proof.

For the second part, we show that if $\gamma > 1/2$, then the RAMSVM can be inconsistent. We do so by giving a counter example. Let $k = 3$ and $P_3 = 0$. Then $E[V(\boldsymbol{f}^*(\boldsymbol{X}), Y)|\boldsymbol{X} = \boldsymbol{x}] = \gamma P_1[2 - \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle]_+ + P_2(1-\gamma)[1 + \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle]_+ + \gamma P_2[2 - \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_2 \rangle]_+ + P_1(1-\gamma)[1 + \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_2 \rangle]_+ + (1-\gamma)[1 + \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_3 \rangle]_+$. If $1/2 < P_1 < \gamma$, then one can verify that the minimizer $\boldsymbol{f}^*$ is such that $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_1 \rangle = \langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_2 \rangle = 2$ and $\langle \boldsymbol{f}^*(\boldsymbol{x}), \mathcal{Y}_3 \rangle = -4$. This is not Fisher consistent. □

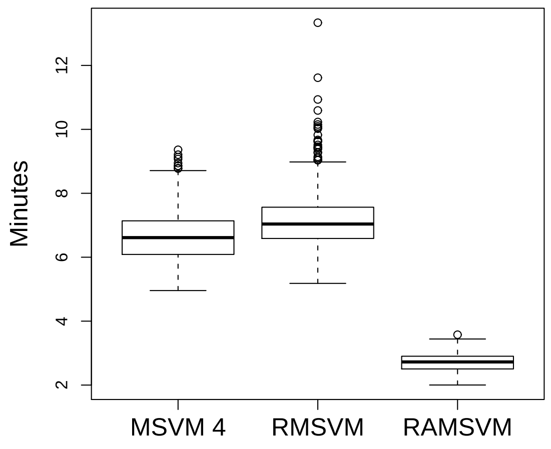


Figure 5.4: The boxplots of computational time in one replication (in minutes) for MSVM 4, RMSVM, and our new RAMSVM in Example 1 with the Gaussian kernel learning.

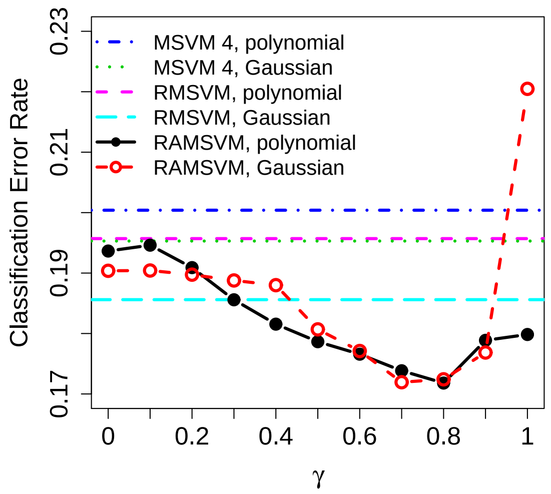


Figure 5.5: The classification accuracy of RAMSVM for different γ values with Gaussian kernel and second order polynomial kernel learning, along with the results from MSVM 4 and RMSVM. The standard errors of the classification error rates range from 0.003 to 0.017.

BIBLIOGRAPHY

Aronszajn, N. 1950. "Theory of Reproducing Kernels." *Transactions of the American Mathematical Society* 68(3):337–404.

Bache, K. and M. Lichman. 2013. "UCI Machine Learning Repository.". University of California, Irvine, School of Information and Computer Sciences.
URL: *http://archive.ics.uci.edu/ml*

Bartlett, P. L. and S. Mendelson. 2002. "Rademacher and Gaussian complexities: Risk bounds and structural results." *Journal of Machine Learning Research* 3:463–482.

Bartlett, Peter L, Michael I Jordan and Jon D McAuliffe. 2006. "Convexity, Classification, and Risk Bounds." *Journal of the American Statistical Association* 101(473):138–156.

Bartlett, Peter L., Olivier Bousquet and Shahar Mendelson. 2005. "Local Rademacher complexities." *Annals of Statistics* 33(4):1497–1537.

Blanchard, Gilles, Oliver Bousquet and Pascal Massart. 2008. "STATISTICAL PERFORMANCE OF SUPPORT VECTOR MACHINES." *Annals of Statistics* 36(2):489–531.

Boser, Bernhard E., Isabelle M. Guyon and Vladimir N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*. COLT '92 New York, NY, USA: ACM pp. 144–152.
URL: *http://doi.acm.org/10.1145/130385.130401*

Boyd, Stephen and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Cortes, C. and V. Vapnik. 1995. "Support Vector networks." *Machine Learning* 20:273–297.

Crammer, Koby, Yoram Singer, Nello Cristianini, John Shawe-taylor and Bob Williamson. 2001. "On the algorithmic implementation of multiclass kernel-based vector machines." *Journal of Machine Learning Research* 2:265–292.

Cristianini, N. and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press.

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin. 2008. "LIBLINEAR: A library for large linear classification." *Journal of Machine Learning Research* 9:1871–1874.

Freund, Yoav and Robert E. Schapire. 1997. "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of Computer and System Sciences* 55(1):119–139.

Friedman, Jerome H., Trevor Hastie and Rob Tibshirani. 2010*a*. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33(1):1–22.

Friedman, Jerome, Trevor Hastie and Robert Tibshirani. 2000. "Additive logistic regression: a statistical view of boosting." *The Annals of Statistics* 28(2):337–407.

Friedman, Jerome, Trevor Hastie and Robert Tibshirani. 2010*b*. "Regularized Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33(1).

Fung, G. and O. L. Mangasarian. 2001. Proximal support vector machine classifiers. In *Proceedings KDD-2001: Knowledge Discovery and Data Mining*. pp. 77–86.

Hastie, T. J., R. J. Tibshirani and J. Friedman. 2009. *The Elements of Statistical Learning.* Springer.

Hill, Simon I. and Arnaud Doucet. 2007. “A framework for kernel-based multi-category classification.” *J. Artif. Int. Res.* 30(1):525–564.

Kimeldorf, G. and G. Wahba. 1971. “Some results on Tchebycheffian spline functions.” *Journal of Mathematical Analysis and Applications* 33:82–95.

Koltchinskii, V. 2006. “Local Rademacher complexities and oracle inequalities in risk minimization.” *Annals of Statistics* 34(6):2593–2656.

Koltchinskii, V. and D. Panchenko. 2002. “Empirical Margin Distributions and Bounding the Generalization Error of Combined Classifiers.” *Annals of Statistics* 30(1):1–50.

Lange, K. and T.T. Wu. 2008. “An MM Algorithm for Multicategory Vertex Discriminant Analysis.” *Journal of Computational and Graphical Statistics* 17(3):527–544.

Lee, Y., Y. Lin and G. Wahba. 2004. “Multicategory Support Vector Machines, theory, and application to the classification of microarray data and satellite radiance data.” *Journal of the American Statistical Association* 99:67–81.

Lin, X., G. Wahba, D. Xiang, F. Gao, R. Klein and B. Klein. 2000. “Smoothing spline ANOVA models for large data sets with Bernoulli observations and the randomized GACV.” *The Annals of Statistics* 28(6):1570–1600.

Lin, Y. 2000. “Tensor product spave anova models.” *The Annals of Statistics* 28:734–755.

Lin, Y. 2004. “A note on margin-based loss functions in classification.” *Stat. and Prob. Letters* 68:73–82.

Lin, Y., Y. Lee and G. Wahba. 2002. “Support Vector Machine for Classification in Nonstandard Situation.” *Machine Learning* 46:191–202.

Lin, Yi. 2002. “Some Asymptotic Properties of the Support Vector Machine.”.

Liu, Y., H. H. Zhang and Y. Wu. 2011. “Soft or hard classification? Large margin unified machines.” *Journal of the American Statistical Association* 106:166–177.

Liu, Y. and M. Yuan. 2011. “Reinforced multicategory support vector machines.” *Journal of Computational and Graphical Statistics* 20(4):901–919.

Liu, Yufeng. 2007. Fisher consistency of multicategory support vector machines. In *Eleventh International Conference on Artificial Intelligence and Statistics.* pp. 289–296.

Liu, Yufeng and Xiaotong Shen. 2006. “Multicategory Ψ-learning.” *Journal of the American Statistical Association* 101:500–509.

Liu, Yufeng, Xiaotong Shen and H. Doss. 2005. “Multicategory Ψ-learning and support vector machine: computational tools.” *Journal of Computational and Graphical Statistics* 14(1):219–236.

Marron, J. S., M. Todd and J. Ahn. 2007. “Distance Weighted Discrimination.” *Journal of the American Statistical Association* 102:1267–1271.

McDiarmid, C. 1989. On the method of bounded dierences. In *In Surveys in Combinatorics.* Cambridge University Press pp. 148–188.

Meinshausen, Nicolai. 2007. "Relaxed lasso." *Computational Statistics and Data Analysis* 52:374–393.

Park, Seo Young, Yufeng Liu, Dacheng Liu and Paul Scholl. 2010. "Multicategory Composite Least Squares Classifiers." *Statistical Analysis and Data Mining* 3(4):272–286.

Qiao, X., H. H. Zhang, Y. Liu, M. J. Todd and J. S. Marron. 2010. "Weighted distance weighted discrimination and its asymptotic properties." *Journal of the American Statistical Association* 105(489):401–414.

Qiao, X. and Y. Liu. 2009. "Adaptive weighted learning for unbalanced multicategory classification." *Biometrics* 65:159–168.

R Core Team. 2013. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing.
URL: *http://www.R-project.org*

Saberian, M. and N. Vasconcelos. 2011. Multiclass boosting: Theory and algorithms. In *NIPS.*

Schölkopf, B. and A. J. Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning).* The MIT Press.

Shawe-Taylor, John and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis.* Cambridge University Press.

Shen, X., G. C. Tseng and W. H. Zhang, X. andWong. 2003. "On Ψ-learning." *Journal of the American Statistical Association* 98:724–734.

Shen, X., G.C. Tseng, X. Zhang and W.H. Wong. 2003. "On ψ-learning." *Journal of the American Statistical Association* 98:724–734.

Shen, X. and L. Wang. 2007. "Generalization error for multi-class margin classification." *Electrical Journal of Statistics* 1:307–330.

Shen, Xiaotong and Wing Hung Wong. 1994. "CONVERGENCE RATE OF SIEVE ESTIMATES." *The Annals of Statistics* 22(2):580–615.

Steinwart, Ingo and Clint Scovel. 2007. "FAST RATES FOR SUPPORT VECTOR MACHINES USING GAUSSIAN KERNELS1." *The Annals of Statitics* 35(2):575–607.

Suykens, J. A. K. and J. Vandewalle. 1999. "Least Squares Support Vector Machine Classifiers." *Neural Processing Letters* 9:293–300.

Tang, Yongqiang and Hao Helen Zhang. 2005. "Multiclass Proximal Support Vector Machines." *Journal of Computational and Graphical Statistis* 15(2):339–355.

Tang, Yongqiang and Hao Helen Zhang. 2006. "Multiclass Proximal Support Vector Machines." *Journal of Computational and Graphical Statistics* 15(2):339–355.

Tewari, A. and P. L. Bartlett. 2007. "On the consistency of multiclass classification methods." *Journal of Machine Learning Research* 8:1007–1025.

Tibshirani, R. 1996. "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society, Series B* 58:267–288.

Tseng, P. 2001. "Convergence of a block coordinate descent method for nondifferentiable minimization." *Journal of Optimization Theory and Applications* 109:475–494.

Van der Vaart, A. W. and J. A. Wellner. 2000. *Weak convergence and Empirical processes with Application to Statistics.* Springer.

Vapnik, V. 1998. *Statistical Learning Theory.* Wiley.

Verhaak, Roel G., Katherine A. Hoadley, Elizabeth Purdom, Victoria Wang, Yuan Qi, Matthew D. Wilkerson, C. Ryan Miller, Li Ding, Todd Golub, Jill P. Mesirov, Gabriele Alexe, Michael Lawrence, Michael O'Kelly, Pablo Tamayo, Barbara A. Weir, Stacey Gabriel, Wendy Winckler, Supriya Gupta, Lakshmi Jakkula, Heidi S. Feiler, J. Graeme Hodgson, C. David James, Jann N. Sarkaria, Cameron Brennan, Ari Kahn, Paul T. Spellman, Richard K. Wilson, Terence P. Speed, Joe W. Gray, Matthew Meyerson, Gad Getz, Charles M. Perou, D. Neil Hayes and Cancer Genome Atlas Research Network. 2010. "Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1." *Cancer cell* 17(1):98–110.

Wahba, G. 1999. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In *Advances in Kernel Methods Support Vector Learning.* MIT Press pp. 69–88.

Wahba, G. 2002. Soft and Hard Classification by Reproducing Kernel Hilbert Space Methods. In *Proceedings of the National Academy of Sciences.* pp. 16524–16530.

Wang, J., X. Shen and Y. Liu. 2008. "Probability estimation for large margin classifiers." *Biometrika* 95(1):149–167.

Wang, L. and X. Shen. 2007. "On L_1-norm multi-class support vector machines: methodology and theory." *Journal of the American Statistical Association* 102:595–602.

Weston, Jason and Chris Watkins. 1999. *Support Vector Machines for Multi-Class Pattern Recognition.* Vol. 4 Citeseer pp. 219–224.
URL: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.144.1692*

Wu, T.T. and K. Lange. 2010. "Multicategory Vertex Discriminant Analysis for High-Dimensional Data." *The Annals of Applied Statistics* 4(4):1698–1721.

Wu, T.T. and Y. Wu. 2012. "Nonlinear Vertex Discriminant Analysis with Reproducing Kernels." *Statistical Analysis and Data Mining* 5:167–176.

Wu, Y. and Y. Liu. 2007. "Robust truncated-hinge-loss support vector machines." *Journal of the American Statistical Association* 102(479):974–983.

Wu, Yichao, Hao Helen Zhang and Yufeng Liu. 2010. "Robust Model-Free Multiclass Probability Estimation." *Journal of the American Statistical Association* 105(489):424–436.

Zhang, C. and Y. Liu. 2013. "Multicategory Large-margin Unified Machines." *Journal of Machine Learning Research* 14:1349–1386.

Zhang, C. and Y. Liu. 2014. "Multicategory Angle-based Large-margin Classification." *Biometrika* . In press.

Zhang, C., Y. Liu and Z. Wu. 2013. "On the effect and remedies of shrinkage on classification probability estimation." *The American Statistician* 67(3):134–142.

Zhang, Tong. 2004*a*. “Statistical Analysis of Some Multi-Category Large Margin Classification Methods.” *Journal of Machine Learning Research* 5:1225–1251.

Zhang, Tong. 2004*b*. “Statistical behavior and consistency of classification methods based on convex risk minimization.” *The Annals of Statitics* 32:56–85.

Zhu, J., H. Zou, S. Rosset and T. Hastie. 2009. “Multi-class Adaboost.” *Statistics and Its Interface* 2(3):349–360.

Zhu, Ji and Trevor Hastie. 2005. “Kernel Logistic Regression and the Import Vector Machine.” *Journal of Computational and Graphical Statistics* 14(1):185–205.

Zhu, Mu and Trevor J. Hastie. 2003. “Feature extraction for non-parametric discriminant analysis.” *Journal of Computational and Graphical Statistics* pp. 101–120.

Zou, H., J. Zhu and T. Hastie. 2008*a*. “New multicategory boosting algorithms based on multicategory Fisher-consistent losses.” *The Annals of Applied Statistics* 2:1290–1306.

Zou, H., J. Zhu and T. Hastie. 2008*b*. “New Multicategory Boosting Algorithms Based on Multicategory Fisher-Consistent Losses.” *Annals of Applied Statistics* 2(4):1290–1306.

Zou, H. and T. Trevor. 2005. “A framework for kernel-based multi-category classification.” *Journal of the Royal Statistical Society, Series B,* 67(2):301–320.