Published in final edited form as:
J Mach Learn Res. 2013 May 1; 14: 1349–1386.

Multicategory Large-Margin Unified Machines

Chong Zhang and
Department of Statistics and Operations Research, The University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA

Yufeng Liu
Department of Statistics and Operations Research, Carolina Center for Genome Sciences, Department of Biostatistics, The University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA

Chong Zhang: CHONGZ@LIVE.UNC.EDU; Yufeng Liu: YFLIU@EMAIL.UNC.EDU

Abstract

Hard and soft classifiers are two important groups of techniques for classification problems. Logistic regression and Support Vector Machines are typical examples of soft and hard classifiers respectively. The essential difference between these two groups 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. To tackle this problem, the Large-margin Unified Machine (LUM) was recently proposed as a unified family to embrace both groups. The LUM family 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. In this paper, we propose a new Multicategory LUM (MLUM) framework to investigate the behavior of soft versus hard classification under multicategory settings. Our theoretical and numerical results help to shed some light on the nature of multicategory classification and its transition behavior from soft to hard classifiers. The numerical results suggest that the proposed tuned MLUM yields very competitive performance.

Keywords

hard classification; large-margin; soft classification; support vector machine

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 classical classification methods. The Support Vector Machine (SVM, Boser et al., 1992; 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. Soft classifiers estimate the class conditional probabilities and make the decision rule based on the obtained probabilities. Typical examples include logistic regression and LDA. Hard classifiers, on the

other hand, focus on estimating the decision boundary without probability estimation. One typical example of hard classifiers is the SVM, which is a well known hard classifier without strong distributional assumptions. Another example of hard classifiers is Ψ−learning (Shen et al., 2003). When class probability estimation is necessary, one can perform multiple weighted learning for probability estimation of hard classifiers (Wang et al., 2008). For a given problem, the choice between hard and soft classifiers can be difficult. Recently, Liu et al. (2011) proposed a family of large-margin classifiers, namely, the Large-margin Unified Machine (LUM). The LUM family is a rich group of classifiers in the sense that it connects hard and soft classifiers in one spectrum. It 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 et al., 2004;Wang and Shen, 2007; Liu and Yuan, 2011). Most of them are not consistent when there is no dominating class, that is, the maximum class probability is less than 0.5 (Tewari and Bartlett, 2007; Liu, 2007). Recently, Liu and Yuan (2011) proposed a group of consistent multicategory piecewise linear hinge loss functions, namely a family of reinforced hinge loss functions, which covers the loss by Lee et al. (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 et al., 2008; 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).

We propose a new group of Multicategory Large-margin Unified Machines (MLUMs) in this paper. 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 provide 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 et al. (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. A MLUM member, in-between hard and soft classifiers, tends to work the best. This was not observed in the binary case (Liu et al., 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 suggest that the performance of the proposed tuned MLUM is very competitive.

The rest of this paper is organized as follows. In Section 2, we give some motivation and introduce the MLUM family. Section 3 explores some statistical properties of the MLUM family. Section 4 addresses the computational aspect of the MLUM. In Section 5, we demonstrate the numerical performance of MLUM via several simulated examples. Section 6 discusses some benchmark examples and one gene expression data set. Some discussion is provided in Section 7. The technical proofs are collected in the appendix.

2. Methodology

In this section, we first introduce the background of binary classification, then discuss 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.

2.1 Background on Binary Classification

With a training data set given, one main goal of classification is to build a classifier for us to predict the class label y using the input vector $\mathbf{x}$. Here we assume that the training data are *i.i.d*. samples from an unknown underlying distribution $D(\mathbf{x}, y)$. In binary classification with $y \in \{\pm 1\}$, we want to estimate a function $f(\mathbf{x}) : \mathbf{R}^d \to \mathbf{R}$ and use sign $(f(\mathbf{x}))$ as the classification rule. Because of the sign rule and the class labels $\{\pm 1\}$, the quantity $yf(\mathbf{x})$ indicates whether the classification of a point $(\mathbf{x}, y)$, sign(f), is correct or not. In particular, we have correct classification if and only if $yf(\mathbf{x}) > 0$. This quantity $yf(\mathbf{x})$ is known as the functional margin in the large-margin classification literature.

Using the functional margin, the theoretical 0 – 1 loss can be directly written as $L(yf(\mathbf{x})) = I(yf(\mathbf{x}) \le 0)$. Our goal is to find a classification function f such that the expected loss of f, denoted by

$$R(f(\cdot)) = E_{\mathbf{X},Y} L(Yf(\mathbf{X})), \quad (1)$$

is as small as possible. The infimum of $R(f(\cdot))$, denoted by R^*, is called the Bayes error. In practice, given a training sample $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ independently drawn from D, we want to find a function f in a functional space H that minimizes the empirical loss

$$\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} L(y_i f(\mathbf{x}_i)), \quad (2)$$

which can be considered as an empirical approximation to the expected loss (1).

Due to the non-convexity and discontinuity of the 0 – 1 loss function L, the minimization of the empirical loss (2) is typically NP-hard and difficult to implement in practice. Many surrogate loss functions have been proposed to alleviate this problem. Furthermore, regularization is often used as well to avoid overfitting. In particular, one can replace the 0 – 1 loss L with a surrogate loss V, and solve the following optimization problem

$$\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} V(y_i f(\mathbf{x}_i)) + \lambda J(f), \quad (3)$$

where $J(f)$ is a regularization function of f that helps to prevent overfitting, and λ is the tuning parameter that balances the loss function term and the regularization term. Different loss functions correspond to different classification methods. To list a few, AdaBoost

employs the exponential loss $V = \exp(-yf)$ (Friedman et al., 2000), SVM (Boser et al., 1992) uses the hinge loss $V = [1 - yf]_+$, and logistic regression (Lin et al., 2000; Zhu and Hastie, 2005) uses the deviance loss $V = \log(1 + \exp(-yf))$. Different 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 et al. (2011) proposed to use the LUM loss function $\ell(\cdot)$ for V in (3), where

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

with $c \geq 0$ and $a > 0$ being parameters of the LUM family. See Figure 1 for the shape of $\ell(u)$ with a few values of a and c (Liu et al., 2011). Note that the LUM family includes the SVM hinge loss with $c \to \infty$, and the Distance Weighted Discrimination (DWD, Marron et al., 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 thoroughly investigate the transition behavior in this spectrum.

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

2.2 Existing Multicategory Classification Methods

To solve a multicategory problem, a natural and direct way is to implement multiple binary classifiers. For example, one can implement the one-versus-one or one-versus-rest methods. Consider a k–category classification problem with the label $y \in \{1, 2, \ldots, k\}$. The one-versus-one approach applies a given binary classifier to a problem of j_1 versus j_2 for all possible $j_1, j_2 \in \{1, 2, \ldots, k\}$. Overall, $\frac{k(k-1)}{2}$ binary classifiers are performed, followed by a majority vote step. In particular, one counts the number of votes for each class obtained from the binary classifiers and classifies the point into the class with the maximum number of votes. When there is a tie for the maximum votes among the classes, one can combine the binary probability information to assign labels, if the binary classification probability estimation is available (Wu et al., 2004). If a hard classifier is used for the one-versus-one approach, then the label is randomly chosen among the classes with equal maximum votes. This can be suboptimal. Furthermore, when k is large, the number of binary classifiers needed can be large as well. In addition, some class sizes can be very small. Another similar technique is the one-versus-rest approach. In particular, it relabels the data in the class j as the positive class and the rest as the negative class, for $j \in \{1, 2, \ldots, k\}$, and performs a sequence of k binary classification problems. The one-versus-rest approach may have inconsistency for some classifiers such as SVMs (Liu and Yuan, 2011). Hence, it is desirable to have a simultaneous multicategory classifier that considers k classes altogether.

The idea of simultaneous multicategory classifiers is as follows. Consider a k–category classification problem. Given an input vector $\mathbf{x}$, we would like to predict its corresponding label $y \in \{1, 2, \ldots, k\}$. Instead of using a one-dimensional classification function $f(\mathbf{x})$ as in the binary case, now we employ a k–dimensional function vector $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$, and predict the class label y of $\mathbf{x}$ using $\text{argmax}_{j=1, \ldots, k} f_j(\mathbf{x})$. Similar as in (3), we are interested in solving the following optimization problem

$$\min_{f\in H}\frac{1}{n}\sum_{i=1}^{n}V\left(\mathbf{f}(\mathbf{x}_i),y_i\right)+\lambda J(\mathbf{f}), \quad (4)$$

with V being a loss function for a multicategory problem, and $J(\mathbf{f})$ being a regularization term defined for the multicategory problems. To reduce the dimension of the problem and to obtain good theoretical properties, a sum-to-zero constraint, $\sum_{j=1}^{k}f_j(\mathbf{x})=0$, 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 $V = -f_y(\mathbf{x}) + \log(e^{f_1(\mathbf{x})} + \cdots + e^{f_k(\mathbf{x})})$. Tang and Zhang (2006) employed the squared loss $V = (\mathbf{z} - \mathbf{f})^T(\mathbf{z} - \mathbf{f})$, where $\mathbf{z}= -\frac{1}{k-1}(1,1,\ldots,1)^T+\frac{k}{k-1}\mathbf{e}_y$, and $\mathbf{e}_j$ 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 $V=\exp(-\frac{1}{k}\mathbf{z}^T\mathbf{f})$. 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) $[1 - f_y(\mathbf{x})]_+$;

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

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

Loss 4 (Lee et al., 2004) $\Sigma_{j\neq y}[1 + f_j(\mathbf{x})]_+$.

Losses 1, 2 and 3 are known to be inconsistent (Lee et al., 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 Loss 4 by Lee et al. (2004) as follows,

$$V(\mathbf{f}(\mathbf{x}),y)=\gamma[(k-1)-f_y(\mathbf{x})]_+ +(1-\gamma)\sum_{j\neq y}[1+f_j(\mathbf{x})]_+, \quad (5)$$

subject to $\sum_{j=1}^{k}f_j(\mathbf{x})=0$. Interestingly, the reinforced hinge loss function is Fisher consistent when $0 \le \gamma \le 1/2$, and includes Loss 4 in Lee et al. (2004) as a special case with $\gamma = 0$. Liu and Yuan (2011) showed that the loss function (4) 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 $\sum_{j=1}^{k}f_j(\mathbf{x})=0$ for different multicategory losses. In linear learning, we assume that $f_j(\mathbf{x}) = \mathbf{x}^T\beta_j + b_j$; $j = 1, \ldots, k$, and the L_2 penalty is $J(\mathbf{f})=\frac{1}{2}\sum_{j=1}^{k}\|\beta_j\|^2$. 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 $J(\mathbf{f})=\frac{1}{2}\sum_{j=1}^{k}\|g_{j,\mathcal{H}}\|^2_{\mathcal{H}}$, 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. Hence, the sum-to-zero constraint helps to obtain unique solutions. The next proposition shows that, if the loss function depends on **f** only through its element-wise differences $f_i - f_j$; $i \neq j$ as in Losses 2 and 3, then without the sum-to-zero constraint, the solutions $\{\hat{\beta}_j\}$ in linear learning or $\{\hat{g}_{j,\mathscr{H}}\}$ 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 1: Suppose the loss function $V(\mathbf{f}, y)$ depends on **f** only through $f_i - f_j$; $i \neq j$, then the solution to (4), using the L_2 penalty without the sum-to-zero constraint, satisfies that $\sum_{j=1}^{k} \hat{\beta}_j = \mathbf{0}$ for linear learning, and $\sum_{j=1}^{k} \hat{g}_{j,\mathscr{H}} = 0$ for RKHS learning.

Notice that the condition in Proposition 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).

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 a convenient platform is available in the multicategory framework. In this paper, 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,

$$V(\mathbf{f}, y) = \gamma \ell\left(f_y(\mathbf{x})\right) + (1-\gamma) \sum_{j \neq y} \ell\left(-f_j(\mathbf{x})\right), \quad (6)$$

under the constraint $\sum_{j=1}^{k} f_j(\mathbf{x}) = 0$, where $\ell(u) : \mathbf{R} \to \mathbf{R}$ 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 (6) is based on the argmax rule for multicategory classification. For a given data point $(\mathbf{x}, y)$, in order to obtain a correct classification decision rule, we need to have the corresponding $f_y(\mathbf{x})$ to be the maximum among $\mathbf{f}(\mathbf{x})$. To that end, the first term in (6) encourages $f_y(\mathbf{x})$ to be large, and the second term encourages $f_j(\mathbf{x})$, $j \neq y$, to be small. Consequently, both terms in (6) try to make f_y big.

To further comprehend the MLUM family, we rewrite (6) using the multiple comparison vector representation proposed by Liu and Shen (2006). In particular, we define the comparison vector $g(\mathbf{f}(\mathbf{x}), y) = (f_y(\mathbf{x}) - f_1(\mathbf{x}), \ldots, f_y(\mathbf{x}) - f_{y-1}(\mathbf{x}), f_y(\mathbf{x}) - f_{y+1}(\mathbf{x}), \ldots, f_y(\mathbf{x}) - f_k(\mathbf{x}))$. Then by the argmax classification rule, a data point $(\mathbf{x}, y)$ is misclassified if and only if $\min g(\mathbf{f}(\mathbf{x}), y) \leq 0$. Let $u = g(\mathbf{f}(\mathbf{x}), y)$, then the $0 - 1$ loss can be written as $I\{\min_j u_j \leq 0\}$, and the MLUM loss can be expressed as

$$V(\mathbf{f}, y) = \gamma \ell\left(\sum_{j=1}^{k-1} u_j / k\right) + (1-\gamma) \sum_{j=1}^{k-1} \ell\left(-\sum_{i=1}^{k-1} u_i / k + u_j\right). \quad (7)$$

Figure 2 shows the plot of (7) 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 (6).

3. Statistical Properties

In this section, we study statistical properties of the MLUM family. We first study its consistency in Section 3.1, and then derive the formula for class probability estimation in Section 3.2. In Section 3.3, we extend the notion of the excess *V*-risk, as defined in Bartlett et al. (2006) and Zhang (2004a), 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.

3.1 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, 2004b; Bartlett et al., 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 $\operatorname{sign}(f^*(\mathbf{x}))=\operatorname{sign}(p(\mathbf{x})-\frac{1}{2})$, where $f^*(\mathbf{x}) = \operatorname{arginf}_f E[V(Y f(\mathbf{X}))|\mathbf{X} = \mathbf{x}]$ and $p(\mathbf{x}) = p(Y = 1|\mathbf{X} = \mathbf{x})$. Based on the definition, Fisher consistency essentially ensures the corresponding decision boundary induced by f^* is identical to the Bayes boundary $\{\mathbf{x} : p(\mathbf{x}) = 1/2\}$.

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

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

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

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

$$S(\mathbf{f},\mathbf{x})=\sum_{j=1}^{k}V\left(\mathbf{f}(\mathbf{x}),j\right)P_j(\mathbf{x}). \quad (8)$$

Because of the argmax rule, Fisher consistency means that for any given $\mathbf{P}(\mathbf{x})$, the minimizer $\mathbf{f}^*(\mathbf{x})=(f_1^*(\mathbf{x}),\ldots,f_k^*(\mathbf{x}))$ of $S(\mathbf{f}, \mathbf{x})$ is such that $\operatorname{argmax}_j P_j(\mathbf{x})=\operatorname{argmax}_j f_j^*(\mathbf{x})$. Furthermore, if $\operatorname{argmax}_j P_j(\mathbf{x})$ is unique, then so is $\operatorname{argmax}_j f_j^*(\mathbf{x})$. 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 $\{P_1, \ldots, P_k\}$, then

$f_1^* > \max\{f_2^*, \ldots, f_k^*\}$. The next lemma assures that in the MLUM family, f_1^* is the maximum among $\{f_1^*, \ldots, f_k^*\}$ (not necessarily unique), even with $c = \infty$.

Lemma 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 (6) with the hinge loss with some minor modifications as in (5), the Fisher consistency will fail when $\gamma > 1/2$. The deficiency of the hinge loss is due to its non-differentiability 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 2 *The MLUM loss function* (6) *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 in the appendix for more discussions.

3.2 Probability Estimation

Class conditional probability estimation is very important in many applications. It is common to use the relationship between $\mathbf{f}^*$ and the class probability $\mathbf{P}$ for estimation of the latter using $\hat{\mathbf{f}}^n$, where $\hat{\mathbf{f}}^n$ is the empirical solution to (4) with V being the proposed MLUM loss. In this section we convert the minimizer $\mathbf{f}^*$ into the class conditional probability estimation. When an estimated $\hat{\mathbf{f}}^n$ is obtained, one can use the formula to obtain the estimated probability $\hat{\mathbf{P}}$. The following theorem gives the probability estimation formula for the MLUM family with any finite c.

Theorem 3 *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* $\hat{\mathbf{f}}^n$. *Then the probability estimation of* P_j *for the MLUM can be expressed as*

$$\hat{P}_j = \frac{1}{\hat{E}(j)} \left\{ \hat{F}(j) + \frac{1}{\sum_{i=1}^k \frac{1}{\hat{E}(i)}} \left(1 - \sum_{i=1}^k \frac{\hat{F}(i)}{\hat{E}(i)} \right) \right\}; j = 1, \ldots, k. \quad (9)$$

Note that the class probability estimation requires that $\hat{P}_j \in [0, 1]$ for $j = 1, \ldots, k$ 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 (9). For $\gamma = 1$, $F(j) = 0$, and one can directly verify the estimated $\hat{P}_i$ using (9) is proper. However, with $\gamma \neq 1$, $\hat{P}_j$ in (9) 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

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

A similar strategy was previously used in Park et al. (2010). Note that besides this one, other scaling strategies can be used as well.

In the binary case, the LUM provides class conditional probability estimation with any finite c. In particular, with the classification function f, $p(\mathbf{x})$ does not have a one-to-one relationship with $f^*(\mathbf{x})$ when $p(\mathbf{x}) = 1/2$ and $c > 0$. In that case, all values of $f^*(\mathbf{x}) \in [-\frac{c}{1+c}, \frac{c}{1+c}]$ correspond to $p(\mathbf{x}) = 1/2$. Figure 3 displays the relationship between $p(\mathbf{x})$ and $f^*(\mathbf{x})$ with $a = 1$ and $c \in \{0, 1, \infty\}$. As shown in Figure 3, when $c > 0$, the flat region of $p(\mathbf{x})$ makes the estimation of class conditional probability more difficult (Liu et al., 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 $f^*(\mathbf{x}) = \text{sign}(p(\mathbf{x}) - 1/2)$.

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 to the multicategory setting. From (9), we can see that for any given $\mathbf{f}^{\hat{n}} = (f_1, \ldots, f_k)$, the estimated probability depends entirely on $\hat{E}(j)$ and $F(j)$; $j = 1, \ldots, k$. 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 $\hat{E}(i) = \hat{E}(j)$, $F(\hat{i}) = F(\hat{j})$, and consequently we have $\hat{P}_i = \hat{P}_j$. This implies that for any obtained $\mathbf{f}^{\hat{n}}$, if the classification signal is weak such that 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 (9). In that case, the estimated probability cannot help us to identify the max probability class.

To further illustrate the relationship between $\mathbf{f}^{\hat{n}}$ and $\hat{\mathbf{P}}$, we use Figure 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 $\hat{\mathbf{P}}$ 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 $\mathbf{f}$ can only produce classification boundary without containing any 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.

3.3 Asymptotic Properties

For binary problems, Zhang (2004a) and Bartlett et al. (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 2.1.

Similarly, the excess V–risk can be written as $Q(f(\cdot)) - Q^*$, where $Q(f(\cdot)) = E_{\mathbf{X},Y}V(f(\mathbf{X}), Y)$ 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 (2004a) and Bartlett et al. (2006) showed that if the excess V-risk converges to 0, so does the excess risk, under some mild assumptions. Zhang (2004b) further studied the relationship between the two excess risks 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 (2004b), and explore the explicit form for MLUM. Moreover, we study the relationship between the convergence rate of the classification function $\hat{\mathbf{f}}^n$ and that of the excess V-risk, as well as the size of the functional space.

Recall that we can rewrite $Q(\mathbf{f}(\cdot))$ as $E_{\mathbf{X}}S(\mathbf{f}, \mathbf{x})$. Consider the MLUM case with $S(\mathbf{f},\mathbf{x})=\sum_{j=1}^{k}[\gamma\ell(f_j(\mathbf{x}))+(1-\gamma)\sum_{i\neq j}\ell(-f_i(\mathbf{x}))]P_j(\mathbf{x})$. It can be verified that $S(\mathbf{f},\mathbf{x})=\sum_{j=1}^{k}P_j(\mathbf{x})[\gamma\ell(f_j(\mathbf{x}))-(1-\gamma)\ell(-f_j(\mathbf{x}))]+(1-\gamma)\sum_{j=1}^{k}\ell(-f_j(\mathbf{x}))$. For brevity, let $Q(\mathbf{P},\mathbf{f})=\sum_{j=1}^{k}P_j[\gamma\ell(f_j)-(1-\gamma)\ell(-f_j)]+(1-\gamma)\sum_{j=1}^{k}\ell(-f_j)$. Note that $Q(\mathbf{P}, \mathbf{f})$ does not involve $\mathbf{x}$. For any given $\mathbf{P}$, let $\mathbf{f}_\ell^*(\mathbf{P})=\operatorname{arginf}_{\mathbf{f}\in\mathbf{R}^k}Q(\mathbf{P},\mathbf{f})$, where $\mathbf{R}^k$ is the k–dimensional real space. Define $Q^*(\mathbf{P})=Q(\mathbf{P},\mathbf{f}_\ell^*(\mathbf{P}))$ to be the optimum value for any given $\mathbf{P}$, and $\Delta Q(\mathbf{P}, \mathbf{f}) = Q(\mathbf{P}, \mathbf{f}) - Q^*(\mathbf{P})$. The excess V–risk is equivalent to $\Delta Q(\mathbf{f}(\cdot)) = E_{\mathbf{X}}\Delta Q(\mathbf{P}(\mathbf{X}), \mathbf{f}(\mathbf{X}))$. Zhang (2004a) and Bartlett et al. (2006) showed that in binary problems, one can bound the excess risk by some transformation of $\Delta Q(\mathbf{f}(\cdot))$. A similar result for multicategory problems is obtained in Zhang (2004b). To study the functional form of $\Delta Q(\mathbf{f}(\cdot))$ in binary problems, Zhang (2004a) proposed to use the Bregman divergence. Here we extend the results in Zhang (2004a) 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 (2004a), it was shown that in the binary case, if g is differentiable, $\Delta Q(P,f)=Pd_g\left(f_g^*(P),f\right)+(1-P)d_g\left(-f_g^*(P),-f\right)$. For the MLUM family, we have the following result, which is a generalization of the binary formula.

Theorem 4 $\Delta Q(\mathbf{P},\mathbf{f})=\sum_{j=1}^{k}[P_j\gamma d_\ell(f_j^*,f_j)+(1-P_j)(1-\gamma)d_\ell(-f_j^*,-f_j)]$.

Theorem 4 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 $\mathbf{f}$ varies in a certain space H. We can decompose the excess V–risk as $\Delta Q(\hat{\mathbf{f}}^n) = [\Delta Q(\hat{\mathbf{f}}^n) - \Delta Q(\mathbf{f}^H)] + [\Delta Q(\mathbf{f}^H]$, where $\mathbf{f}^H$ is the function that achieves the minimal of $\Delta Q(\mathbf{f}) : \mathbf{f} \in H$ 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 H is rich enough, so that

$$\arg\inf_{\mathbf{f}\in H} E_{\mathbf{X},Y}\left(V(\mathbf{f}(\mathbf{X}),Y)\right) = \arg\inf_{\mathbf{f}\in \mathbf{R}^k} E_{\mathbf{X},Y}\left(V(\mathbf{f}(\mathbf{X}),Y)\right) = \mathbf{f}^*, \quad (10)$$

the estimator $\hat{\mathbf{f}^n}$ will converge to **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 $\hat{\mathbf{f}^n}$ 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 (6).

Let μ(·) be the regular Lebesgue measure. For any fixed *a*, *c* and γ, the distribution *D*(**X**, *Y*) naturally defines *k* probability measures on the real line: $\tau_j(B) = P(f_j^* \in B)$; *j* = 1, …, *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, …, *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 $\gamma \in [0, 1]$, $n^q(\hat{\mathbf{f}^n}(\mathbf{x}, y) - \mathbf{f}^*(\mathbf{x}, y)) \to \mathbf{T}(c, a, \gamma, \mathbf{x}, y)$ in distribution, where $\mathbf{T}(c, a, \gamma, \mathbf{x}, y) = (T_1, \ldots, T_k)^T$ is a multivariate random variable, whose distribution depends on *c*, *a*, γ, and varies among different (**x**, *y*); $q > 0$ is a constant. Furthermore, suppose that for fixed *c*, *a*, and γ, $\int_{\mathbf{X},Y} |\sup_{1\leq j\leq k} T_j|^2 dD(\mathbf{X}, Y) < \infty$.

Now we are in the position to introduce the following theorem.

Theorem 5 *In the MLUM family with the underlying distribution D, suppose Assumptions A and B are satisfied, and* (10) *holds. Then for any fixed c, a, and* γ,

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

Note that Assumption A can be verified if there is no probability mass point in the distribution *D*. Under Assumption A, the expected loss $Q(\mathbf{f}(\cdot))$ has bounded second order derivative for a fixed *c* almost surely. Hence under some conditions, $q = 1/2$ for regular finite dimensional problems, and $\hat{\mathbf{f}^n}$ 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 $\hat{\beta}$ (Shen and Wong, 1994), with a proper choice of the regularization term. In this case, the integrability in Assumption B can be satisfied if **X** is bounded. From Theorem 5, we can conclude that the excess *V* –risk is *n*–consistent for any *a*, *c* and γ, when the function space is large enough.

Remark 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 $\hat{\mathbf{f}^n} - \mathbf{f}^*$. The square integrable requirement of Assumption B is essential in the sense that it prevents the distribution of **T** from diverging with large probability when (**X**, *Y*) varies.

Remark 2. The potential problem of non-unique **f*** does not affect the result of Theorem 5. Because *S* in (8) 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 the V–approximation error is nonzero, in other words,

$$\inf_{\mathbf{f}\in H} E_{\mathbf{X},Y}\left(V(\mathbf{f}(\mathbf{X}),Y)\right) > \inf_{\mathbf{f}\in \mathbf{R}^k} E_{\mathbf{X},Y}\left(V(\mathbf{f}(\mathbf{X}),Y)\right), \quad (11)$$

Theorem 5 is not applicable, because the excess V–risk does not converge to 0 anymore. Then we are interested in the convergence rate of the V–estimation error, $\Delta Q(\hat{\mathbf{f}}^n) - \Delta Q(\mathbf{f}^H)$. First, we need to modify Assumption B as follows.

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

Theorem 6 *In the MLUM family with the underlying distribution D, suppose Assumptions A and B' are satisfied, and* (11) *holds. Then for any fixed c, a, and* γ,

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

From Theorem 6, we can see that when the theoretical minimizer does not belong to the function class H, the convergence rate of the excess V-risk is the same as $\hat{\mathbf{f}}^n$. When $q = 1/2$ under some mild conditions, the excess V-risk is also root n-consistent.

4. Computational Algorithm

In this section, we discuss how to implement (4) with the MLUM family. The MLUM loss (6) 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 et al., 2010).

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

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

We choose the first $(k-1)^{\text{th}}$ elements of $\mathbf{f}$ as free parameters, and let $\sum_{j=1}^{k} f_j(\mathbf{x}) = 0$ for all $\mathbf{x}$. For simplicity we focus on the linear learning with $f_j(\mathbf{x}) = \mathbf{x}^T\beta_j + b_j$, $j = 1, \ldots, k$, and the L_2 penalty with $J(\mathbf{f}) = \frac{1}{2}\sum_{j=1}^{k} \|\beta_j\|^2$. 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 $\Xi^{(m)}=(\beta_1^{(m)},\ldots,\beta_k^{(m)})^T$ and $\mathbf{b}=(b_1^{(m)},\ldots,b_k^{(m)})^T$. At the $(m+1)^{th}$ step, define $\tilde{\mathbf{f}}_{i,j,-p}=(\mathbf{x}_i^T\beta_1^{(m+1)}+b_1^{(m+1)},\ldots,\mathbf{x}_i^T\beta_{j-1}^{(m+1)}+b_{j-1}^{(m+1)},W_{i,1},\ldots,\mathbf{x}_i^T\beta_{k-1}^{(m)}+b_{k-1}^{(m)},W_{i,2})^T$ 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 $\tilde{\mathbf{f}}_{i,j,-p}$ is 0. Set

$$\Xi_{j,p}^{(m+1)}=\underset{z}{\operatorname{argmin}}\frac{\lambda}{2}\left(z^2+[\sum_{s=1}^{p-1}\Xi_{j,s}^{(m+1)}+z+\sum_{s=p+1}^{d}\Xi_{j,s}^{(m)}]^2\right)+\frac{1}{n}\sum_{i=1}^{n}V(\hat{\mathbf{f}}_{i,j,-p}+x_{i,j}z,y_i). \quad (12)$$

The optimization of (12) 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 (12) 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 $\beta_j = (0, \ldots, 0)^T$ and $b_j = 0$, for all $j = 1, \ldots, k$. We keep $\beta_k = -\beta_1 - \beta_2 - \cdots - \beta_{k-1}$ and $b_k = -b_1 - b_2 - \cdots - b_{k-1}$.

Step 2 At the beginning of the m^{th} loop, suppose we have $(\beta_1, \ldots, \beta_k)$ and $(b_1, \ldots, b_k)$ as the intermediate results.

2.1 Update, in orders, the elements of β_1.

2.2 After the entire vector β_1 has been updated, update the intercept b_1.

2.3 Update the vectors $\beta_2, \ldots, \beta_{k-1}$ and the intercepts $b_2, \ldots, b_{k-1}$, in the same manner as above.

Step 3 Repeat Step 2 until convergence.

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 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 5.1 contains two examples in which the hard classifier works the best. Section 5.2 consists of two examples that favor the soft classifier. Section 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 et al. (2011). Note that this multicategory DWD is different from the version proposed in Huang et al. (2013). Section 5.4 provides some summary on the effect of hard versus soft classifiers and gives some insight 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, …, 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 2.2, RMSVM, Multicategory Penalized Logistic Regression (MPLR), along with the tuned MLUM, in Table 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 would like to point out that this MPLR is new and we refer it as 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), $E_{\mathbf{X}}|p(\mathbf{X}) - \hat{p}(\mathbf{X})|$. 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 $\mathbf{x}$ for typical training samples in Figure 5.

5.1 Hard Classification 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 hard classifiers to perform better than the soft ones, because the former bypasses probability estimation.

Example 1: We generate two dimensional data uniformly on $[-1, 1]^2$, with four classes. 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 another 80 for tuning.

This is a multicategory generalization of Example 2 in Liu et al. (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.

The test errors of this example are reported in Figure 6. As we expect, hard classifiers outperform soft ones. With tuned C, MLUM with $\gamma > 0.5$ works better than those with $\gamma \leq 0.5$. 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 1, and the soft classifier gives the best probability estimation.

Example 2: We generate a three-class problem in this example, and **X** is two dimensional. 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 7, and the MAEs in Table 2. Based on the results, both test errors and MAEs suggest that the hard classifier performs significantly better than the others. 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).

5.2 Soft Classification 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,

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

where the means $\mu_j = (1, 1, 1)^T$, $(1, 1, -1)^T$, $(1, -1, 1)^T$, $(1, -1, -1)^T$, $(-1, 1, 1)^T$, $(-1, 1, -1)^T$, $(-1, -1, 1)^T$, $(-1, -1, -1)^T$, 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 8, and the MAEs are reported in Table 3. This is a case in which the soft classifier works the best both in terms of classification accuracy as well as probability estimation.

Example 4: In this example we have 2 covariates, and the number of classes is 20. Each marginal distribution of $\mathbf{X}|Y = j$; $j = 1, \ldots, 20$ follows a $N(\mu_j, \sigma^2 I_2)$ distribution. Here we choose μ_j such that μ_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 9, and the MAEs in Table 4. In this example, the soft classifier significantly dominates the others in terms of prediction accuracy. The tuned MLUM method always chooses $c = 0$ throughout 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.

5.3 MLUM with $c = 1$ Better

Liu et al. (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 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 data sets, 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: In this example, there are 95% data points from a six-class distribution with

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

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 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 5, we see that the probability estimation with $c = 1$ is more accurate than the others.

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 that of Example 5. Compared to Figure 9, Figure 11 shows that $c = 0$ is more sensitive to noise than the other c values, while $c = 1$ is the most robust one. Table 6 shows that the probability estimation with $c = 1$ is the best, as in Example 5.

5.4 Summary of Simulation Results

Our simulated examples provide some insights on hard versus soft classification methods, and suggest that no single classifier 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 et al. (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 et al. (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 et al., 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 those of the

others are just around 15 – 20%. This indicates that trying to estimate class conditional probabilities when data contain outliers may severely reduce the accuracy of the classification boundary estimation.

Another observation is that in the simulated data sets, 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 it is uncertain about which classifier to use, we propose to apply our tuned MLUM method. As 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 7 reports comparisons of five MSVM methods, as discussed in Section 2.2, and MPLR, with our proposed tuned MLUM. The results show that the tuned MLUM delivers the most accurate classifier in most cases.

6. Real Data Examples

This section empirically tests the performance of the MLUM family, on two types of data sets. The first type of data examples includes several benchmark data sets available on the UCI machine learning website. The second one is a recent microarray gene expression data set in cancer research. For all the real data sets, 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.

6.1 Benchmark Data

We consider seven benchmark data sets 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 data set, and test the performance on the remaining 1/5. For Iris and Image data, we apply linear learning. For the Vehicle data set, we apply the second order polynomial kernel learning. The Gaussian kernel is used for the Breast, Dermatology, Wine and Vertebral data sets, with the σ parameter value being the median of all pairwise distances between one category and the other. For all data sets, we standardize the attributes before further analysis.

For illustration, we summarize the data sets in Table 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 data set 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 data sets with potential outliers. For demonstration, we report the PCA plots for the Iris and Vehicle data in Figure 12. The Iris data set doesn't appear to have any obvious outliers and the soft classifier works very well there. In the Vehicle data, there appears to have several potential outliers, and the soft classifier turns out to be the worst in terms of classification accuracy.

Breast: The data set contains 10 variables and 22, 21, 14, 15, 16 and 18 instances for the following 6 breast diseases respectively: carcinoma, fibro-adenoma, mastopathy, glandular, connective, and adipose. The classification errors in Figure 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 data set, 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 14. With $c \to \infty$, the classification accuracy is the best. In addition, $\gamma = 0.2$ performs better than other values.

Image: In the Image data set, 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 data set. We report the test errors in Figure 15. In this case, $c = 1$ performs the best, and $\gamma = 0.9$ is the optimal choice when c is tuned.

Iris: There are three classes in the Iris data set, 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 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 data set 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 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 data set are normal, disk hernia and spondilolysthesis. The numbers of instances are 100, 60 and 150, respectively. Six covariates are present in the data set. Figure 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 $\gamma = 0.9$.

Wine: The wine data set measures 3 classes (59, 71 and 48 each) of wine with 13 attributes. In Figure 19, we see that soft classifier has the most accurate prediction result, and $\gamma = 0$ is the best choice when c is tuned.

6.2 Glioblastoma Multiforme Cancer Data

In this section we apply the MLUM family to a cancer data set and explore its performance. The data set was previoulsy used by Verhaak et al. (2010). They investigated Glioblastoma Multiforme (GBM) cancer on the genetic data of patients from The Cancer Genome Atlas Research Network. There are 356 observations in the data set, which consists 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 data set. 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 median absolute deviation values in the training data set.

To visualize the data, we plot the first two projected principal component directions of the data in Figure 20. From Figure 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 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 9. Overall, the tuned MLUM performs the best, as in the simulated examples.

7. Discussion

Multicategory classification is commonly seen in practice. In this paper, 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 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 $\hat{\mathbf{f}}$, as well as the size of the function class space. The numerical examples show that 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 automatic 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.

Acknowledgments

The authors would like to thank the Action Editor Professor Saharon Rosset and three reviewers for their constructive comments and suggestions, and Derek Y. Chiang for helpful discussions. The authors are supported in part by NIH/NCI grant R01 CA-149569.

Appendix

Appendix A. Proof of Proposition 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 $g_{j,\mathcal{H}}(\mathbf{x})=\sum_{i=1}^{n}\alpha_{i,j}K(\mathbf{x}_i,\mathbf{x})$, where $K(\cdot,\cdot)$ is the associated reproducing kernel, and $\mathbf{x}_i$; $i = 1, \ldots, n$ are the training data points. With the kernel learning, the penalty becomes $\frac{1}{2}\sum_{j=1}^{k}\alpha_j^T K\alpha_j$, where $\alpha_j = (\alpha_{1,j}, \ldots, \alpha_{n,j})^T$; $j = 1, \ldots, k$, and with a little abuse of notation, K is the gram matrix with the $(i, j)^t h$ element $K_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$. 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 $\mathbf{f}$ only through $f_i - f_j$, it suffices to prove that for a given solution $\hat{\alpha_j} = (\hat{\alpha_{1,j}}, \ldots, \hat{\alpha_{n,j}})$; $j = 1, \ldots, k$ such that $\sum_{j=1}^{k}\hat{\alpha}_{i,j}=0$ for all i, $\sum_{j=1}^{k}\hat{\alpha}_j^T K\hat{\alpha}_j<\sum_{j=1}^{k}(\hat{\alpha}_j-\mathbf{z})^T K(\hat{\alpha}_j-\mathbf{z})$. Here $\mathbf{z}$ is any fixed vector of length n that is not $\mathbf{0}$. To this end, we see that $\sum_{j=1}^{k}\hat{\alpha}_j^T K\hat{\alpha}_j-\sum_{j=1}^{k}(\hat{\alpha}_j-\mathbf{z})^T K(\hat{\alpha}_j-\mathbf{z})$ can be simplified as $2(\hat{\alpha}_1^T+\cdots+\hat{\alpha}_k^T)K\mathbf{z}-k\mathbf{z}^T K\mathbf{z}$. Note that $\hat{\alpha}_1^T+\cdots+\hat{\alpha}_k^T=\mathbf{0}$, and K is positive definite, so that $2(\hat{\alpha}_1^T+\cdots+\hat{\alpha}_k^T)K\mathbf{z}-k\mathbf{z}^T K\mathbf{z}<0$, and this completes the proof.

Appendix B. Proof of Lemma 1

We prove by contradiction. Without loss of generality, assume that $P_1 > P_2$, and $f_1^* < f_2^*$. Now let $\mathbf{f}=(f_2^*, f_1^*, f_3^*, \ldots, f_k^*)$, and we want to show that $S(\mathbf{f}, \mathbf{x}) < S(\mathbf{f}^*, \mathbf{x})$. One can verify that $S(\mathbf{f}, \mathbf{x}) - S(\mathbf{f}^*, \mathbf{x}) = (P_2 - P_1)(\gamma\delta^+ + (1-\gamma)\delta^-)$, 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 $S(\mathbf{f}, \mathbf{x}) < S(\mathbf{f}^*, \mathbf{x})$, and $\mathbf{f}^*$ is not the minimizer of S.

Appendix C. Proof of Theorem 2

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 1, $f_1^* \geq f_2^*$. Suppose $f_1^* = f_2^*$, and let $\mathbf{f}=(f_1^*+\varepsilon, f_2^* - \varepsilon, f_3^*, \ldots, f_k^*)$, where $\varepsilon > 0$ is sufficiently small. We show that $S(\mathbf{f}, \mathbf{x}) < S(\mathbf{f}^*, \mathbf{x})$. For simplicity in notation, let $f = f_1^* = f_2^*$. After some calculation we have

$$S(\mathbf{f}, \mathbf{x}) - S(\mathbf{f}^*, \mathbf{x}) = (1-\gamma)[\delta_1 - \delta_2] + P_1[\gamma(-\delta_4) + (1-\gamma)(-\delta_1)] + P_2[\gamma(\delta_3) + (1-\gamma)(\delta_2)] \quad (13)$$

where

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

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

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

and so (13) becomes $\varepsilon(P_1 - P_2)[\gamma\ell'(f) + (1-\gamma)\ell'(-f)] + o(\varepsilon)$. Because $\ell(\cdot)$ is convex and strictly monotonely decreasing, $\ell'(\cdot) < 0$, thus $S(\mathbf{f}, \mathbf{x}) - S(\mathbf{f}^*, \mathbf{x}) < 0$.

Remark 3. The reason why the hinge loss is not Fisher consistent when $\gamma > 1/2$ becomes more clear from the proof of Theorem 2. For 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 $P_2[\gamma(\delta_3) + (1-\gamma)(\delta_2)]$ with $\gamma > 1/2$. By allowing differentiability in the loss function, we overcome the disadvantage of non-Fisher consistency.

Appendix D. Proof of Theorem 3

Due to the constraint $\sum_{j=1}^{k} f_j = 0$, the degree of freedom for S in (8) is only $k-1$. Rewrite S as

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

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

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

where E^* and F^* are defined in an obvious manner. Let $\mathbf{P}_{k-1}=(P_1,\ldots,P_{k-1})^T$, we can rewrite the above equations as $M\mathbf{P}_{k-1}-\mathbf{b}=0_{k-1}$, or $M\mathbf{P}_{k-1}=\mathbf{b}$ where $\mathbf{b}=([1-\gamma)\ell'(-f_1^*)+\gamma\ell'(f_k^*)],\ldots,[(1-\gamma)\ell'(-f_{k-1}^*)+\gamma\ell'(f_k^*)])^T$ and $M=\mathrm{diag}(E^*(1),\ldots,E^*(k-1))+E^*(k)J_{k-1}$. Here J_{k-1} is the $k-1$ by $k-1$ matrix with every element being 1. Since $E(\cdot)<0, 1+E^*(k)\sum_{j=1}^{k-1}E^*(j)>0$, and we apply the Sherman-Morrison formula (which guarantees the invertibility of M), to obtain $\mathbf{P}_{k-1}=M^{-1}\mathbf{b}$. Let $P_k=1-\sum_{j=1}^{k-1}P_j$. The equations (9) follow after some simplification and substituting $\hat{\mathbf{f}}$ for $\mathbf{f}^*$. Note that from the form of (9), we conclude the choice of the free parameters in $\mathbf{f}$ is not essential.

Appendix E. Proof of Theorem 4

By definition,

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

We can rewrite the RHS of the display as

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

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

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

In combination with the Bregman divergence, observe that the above is essentially RHS of Theorem 4 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, we have

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

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

$$W=\sum_{j=1}^{k-1}[-F^*(j)+F^*(k)+P_jE^*(j)-P_kE^*(k)](f_j-f_j^*).$$

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

Appendix F. Proof of Theorem 5

Expand the Bregman divergence in Theorem 4 into the Taylor series form:

$$\Delta Q(\mathbf{P},\mathbf{f})=\sum_{i=1}^{k}\left[P_i\gamma[\frac{\ell^{(2)}(f_i^*)}{2}(\widehat{f_i}-f_i^*)^2+o((\widehat{f_i}-f_i^*)^2)]+(1-P_i)(1-\gamma)[\frac{\ell^{(2)}(-f_i^*)}{2}(\widehat{f_i}-f_i^*)^2+o((\widehat{f_i}-f_i^*)^2)]\right].$$

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

$$n^{2q}\Delta Q(\widehat{\mathbf{f}}^n) = O\left(\int_{\mathbf{x},Y} |\sup_{1\le j\le k} T_j|^2 dD(\mathbf{X}, Y)\right).$$

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

Appendix G. Proof of Theorem 6

Note that $(\widehat{f}_i - f_i^*)^2 - (f_i^H - f_i^*)^2 = (\widehat{f}_i - f_i^H)(\widehat{f}_i + f_i^H - 2f_i^*)$. $(\widehat{f}_i - f_i^H)$ is n^q consistent, and $|\widehat{f}_i + f_i^H - 2f_i^*| \le |\widehat{f}_i - f_i^H| + 2(|f_i^H| + |f_i^*|) \to 2(|f_i^H| + |f_i^*|)$. The rest of the proof is analogous to that of Theorem 5.

References

Aronszajn N. Theory of reproducing kernels. Transactions of the American Mathematical Society. 1950; 68(3):337–404.

Bartlett PL, Jordan MI, McAuliffe JD. Convexity, classification, and risk bounds. Journal of the American Statistical Association. 2006; 101(473):138–156.

Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92. 1992:144–152. ISBN 0-89791-497-X.

Cortes C, Vapnik V. Support vector networks. Machine Learning. 1995; 20:273–297.

Crammer K, Singer Y, Cristianini N, Shawe-taylor J, Williamson B. On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research. 2001; 2:265–292.

Freund Y, Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences. 1997; 55(1):119–139.

Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting. Annals of Statistics. 2000; 28(2):337–407.

Friedman J, Hastie T, Tibshirani R. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software. 2010; 33(1):1–22. [PubMed: 20808728]

Huang H, Liu Y, Du Y, Perou C, Hayes DN, Toddand M, Marron JS. Multiclass distance weighted discrimination. Journal of Computational and Graphical Statistics. 2013 Forthcoming.

Lee Y, Lin Y, Wahba G. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. Journal of the American Statistical Association. 2004; 99(465):67–81.

Lin X, Wahba G, Xiang D, Gao F, Klein R, Klein B. Smoothing spline anova models for large data sets with bernoulli observations and the randomized gacv. Annals of Statistics. 2000; 28(6):1570–1600.

Liu Y. Fisher consistency of multicategory support vector machines. Eleventh International Conference on Artificial Intelligence and Statistics. 2007:289–296.

Liu Y, Shen X. Multicategory ψ-learning. Journal of the American Statistical Association. 2006; 101(474):500–509.

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

Liu Y, Shen X, Doss H. Multicategory ψ-learning and support vector machine: computational tools. Journal of Computational and Graphical Statistics. 2005; 14(1):219–236.

Liu Y, Zhang HH, Wu Y. Soft or hard classification? large margin unified machines. Journal of the American Statistical Association. 2011; 106(493):166–177. [PubMed: 22162896]

Marron JS, Todd M, Ahn J. Distance weighted discrimination. Journal of the American Statistical Association. 2007; 102(480):1267–1271.

Park SY, Liu Y, Liu D, Scholl P. Multicategory composite least squares classifiers. Statistical Analysis and Data Mining. 2010; 3(4):272–286. [PubMed: 21218128]

Shen X, Wong WH. Convergence rate of sieve estimates. Annals of Statistics. 1994; 22(2):580–615.

Shen X, Tseng GC, Zhang X, Wong WH. On ψ-learning. Journal of the American Statistical Association. 2003; 98(463):724–734.

Steinwart I, Scovel C. Fast rates for support vector machines using gaussian kernels. Annals of Statistics. 2007; 35(2):575–607.

Tang Y, Zhang HH. Multiclass proximal support vector machines. Journal of Computational and Graphical Statistics. 2006; 15(2):339–355.

Tewari A, Bartlett PL. On the consistency of multiclass classification methods. Journal of Machine Learning Research. 2007; 8:1007–1025.

Tseng P. Convergence of a block coordinate descent method for nondifferentiable minimization. Journal of Optimization Theory and Applications. 2001; 109(3):475–494.

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

Verhaak RG, Hoadley KA, Purdom E, Wang V, Qi Y, Wilkerson MD, Miller CR, Ding L, Golub T, Mesirov JP, Alexe G, Lawrence M, O'Kelly M, Tamayo P, Weir BA, Gabriel S, Winckler W, Gupta S, Jakkula L, Feiler HS, Hodgson JG, James CD, Sarkaria JN, Brennan C, Kahn A, Spellman PT, Wilson RK, Speed TP, Gray JW, Meyerson M, Getz G, Perou CM, Hayes DN. Cancer Genome Atlas Research Network. Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in pdgfra, idh1, egfr, and nf1. Cancer Cell. 2010; 17(1):98–110. [PubMed: 20129251]

Wahba, G. Advances in Kernel Methods Support Vector Learning. MIT Press; 1999. Support vector machines, reproducing kernel hilbert spaces and the randomized GACV; p. 69-87.

Wahba G. Soft and hard classification by reproducing kernel hilbert space methods. Proceedings of the National Academy of Sciences. 2002; volume 99:16524–16530.

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

Wang L, Shen X. On l_1-norm multi-class support vector machines: methodology and theory. Journal of the American Statistical Association. 2007; 102(478):595–602.

Weston J, Watkins C. Support vector machines for multi-class pattern recognition. Proceedings of the Seventh European Symposium on Artificial Neural Networks. 1999; 4:219–224.

Wu TF, Lin CJ, Weng RC. Probability estimates for multi-class classification by pairwise coupling. Journal of Machine Learning Research. 2004; 5:975–1005.

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

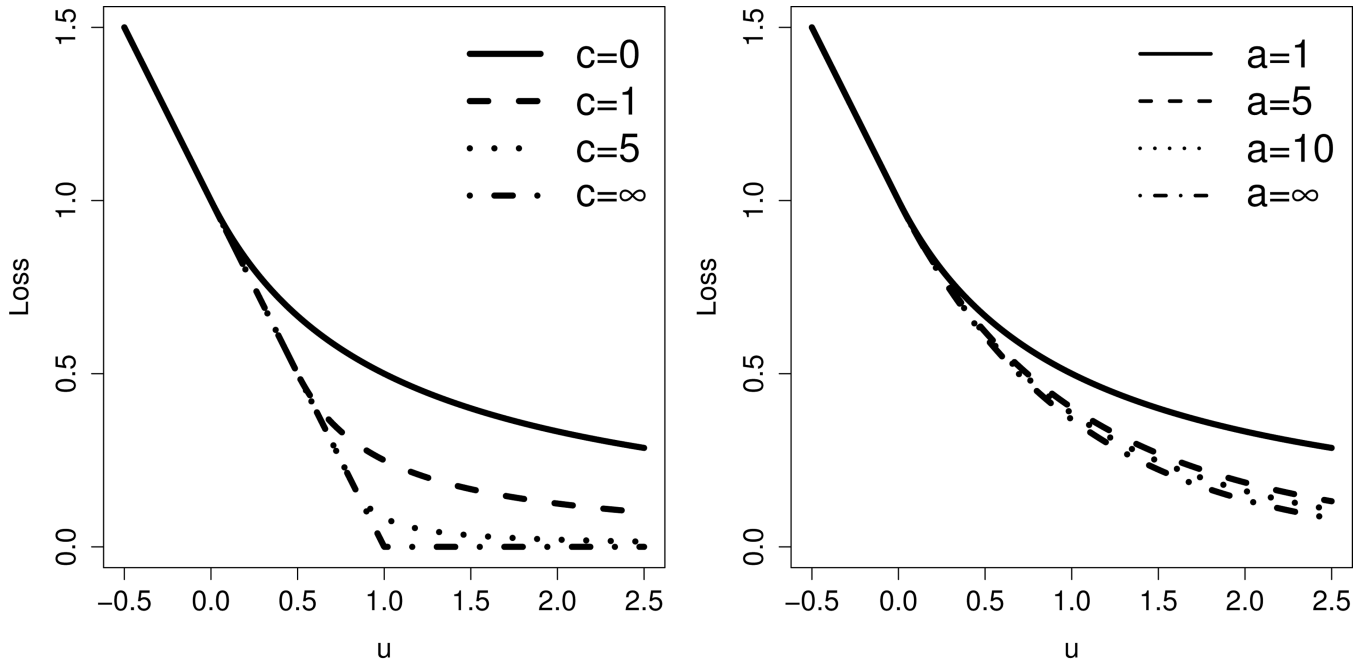
Zhang T. Statistical behavior and consistency of classification methods based on convex risk minimization. Annals of Statistics. 2004a; 32:56–85.

Zhang T. Statistical analysis of some multi-category large margin classification methods. Journal of Machine Learning Research. 2004b; 5:1225–1251.

Zhu J, Hastie T. Kernel logistic regression and the import vector machine. Journal of Computational and Graphical Statistics. 2005; 14(1):185–205.

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

Zou H, Zhu J, Hastie T. New multicategory boosting algorithms based on multicategory fisher-consistent losses. Annals of Applied Statistics. 2008; 2(4):1290–1306.



(a) a=1 (b) c=0

Figure 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 $a = 1,5,10,\infty$.

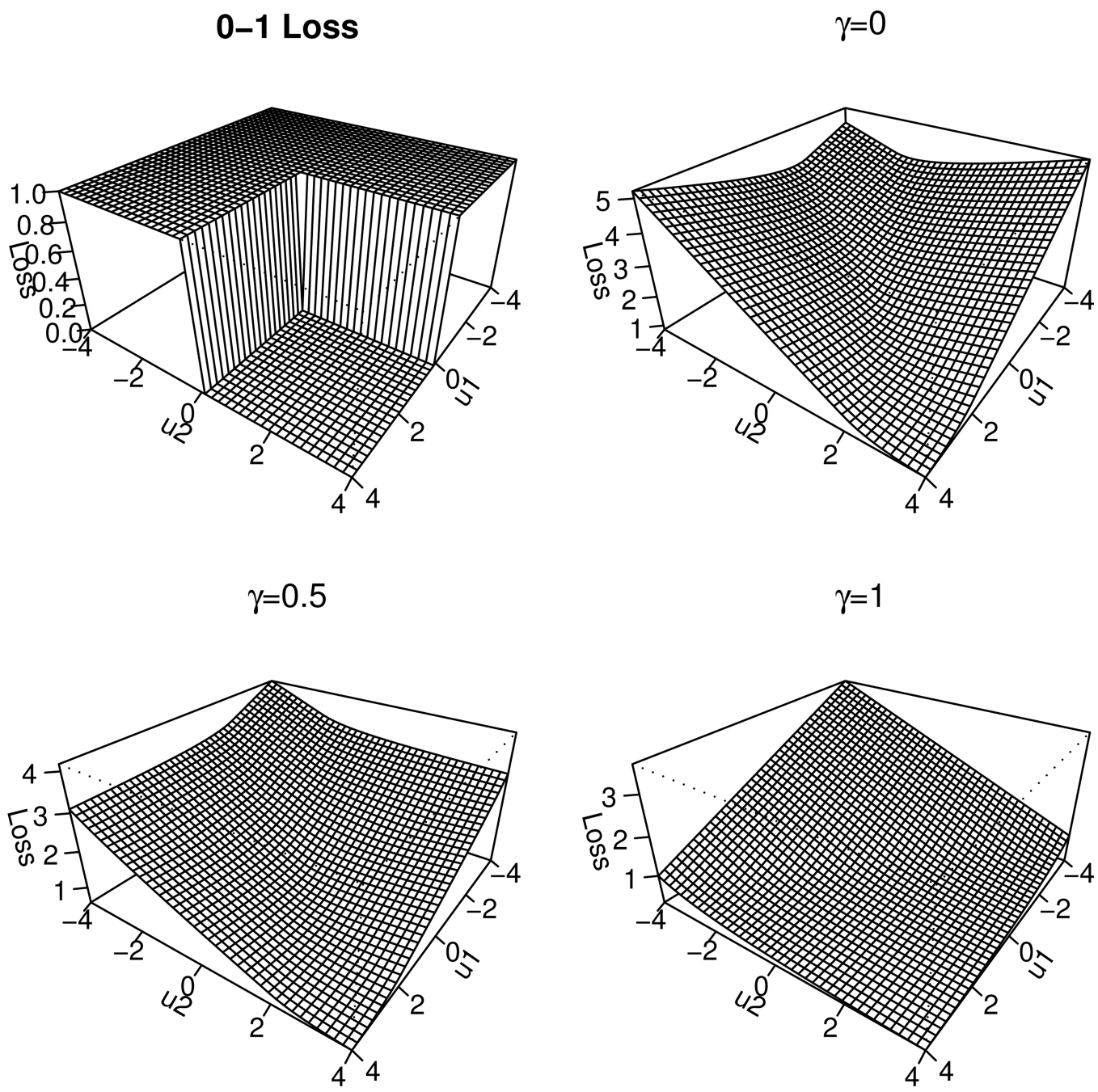


Figure 2.
Plots of the 0 – 1 loss function (top left panel) and MLUM loss functions with $\gamma = 0$ (top right), $\gamma = 0.5$ (bottom left), $\gamma = 1$ (bottom right), for $c = 0$, $a = 1$ and $k = 3$.

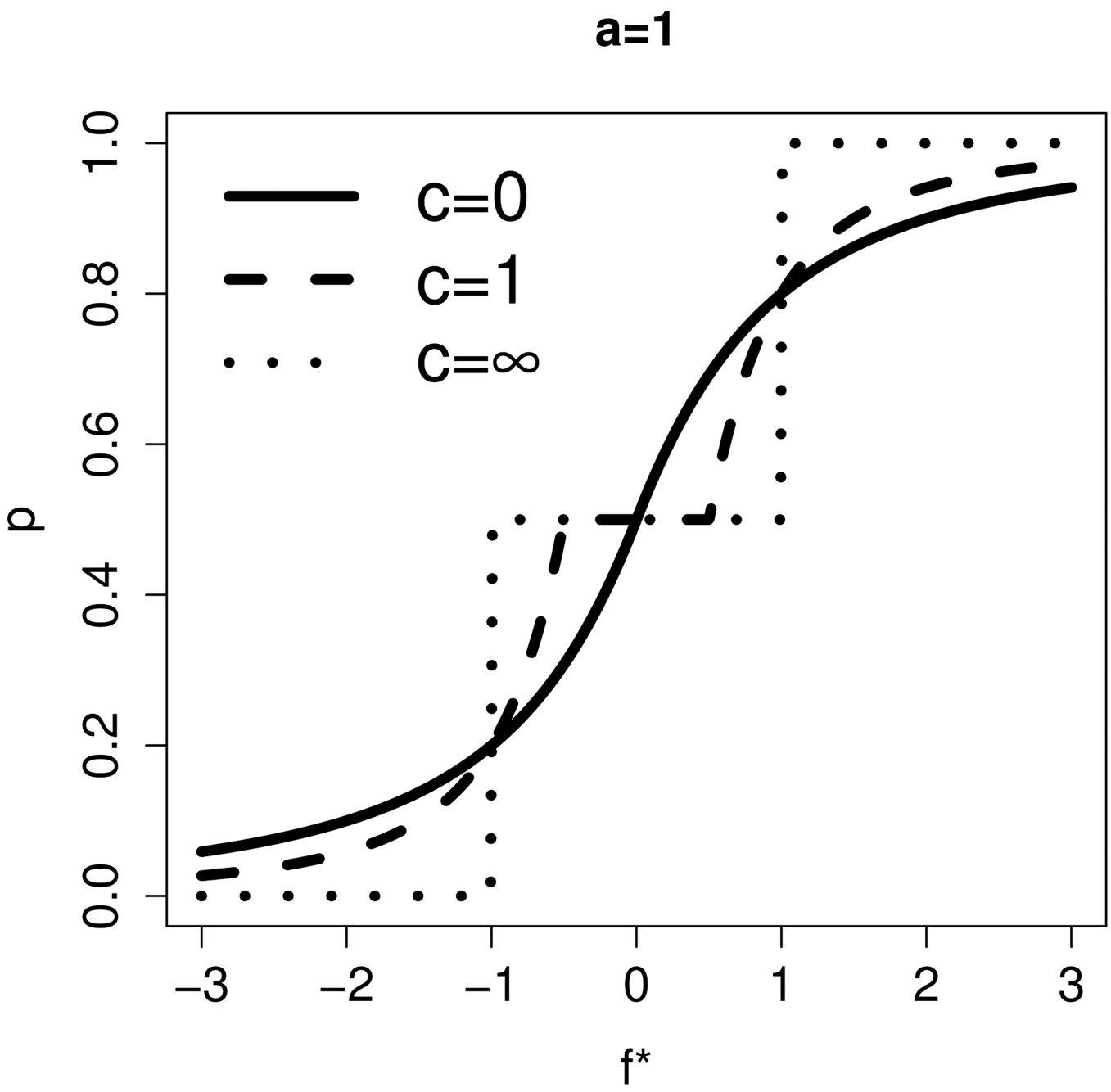


Figure 3.
Plot of the correspondence between $f^*(\mathbf{x})$ and $p(\mathbf{x})$ with $a = 1$, $c \in \{0,1,\infty\}$.

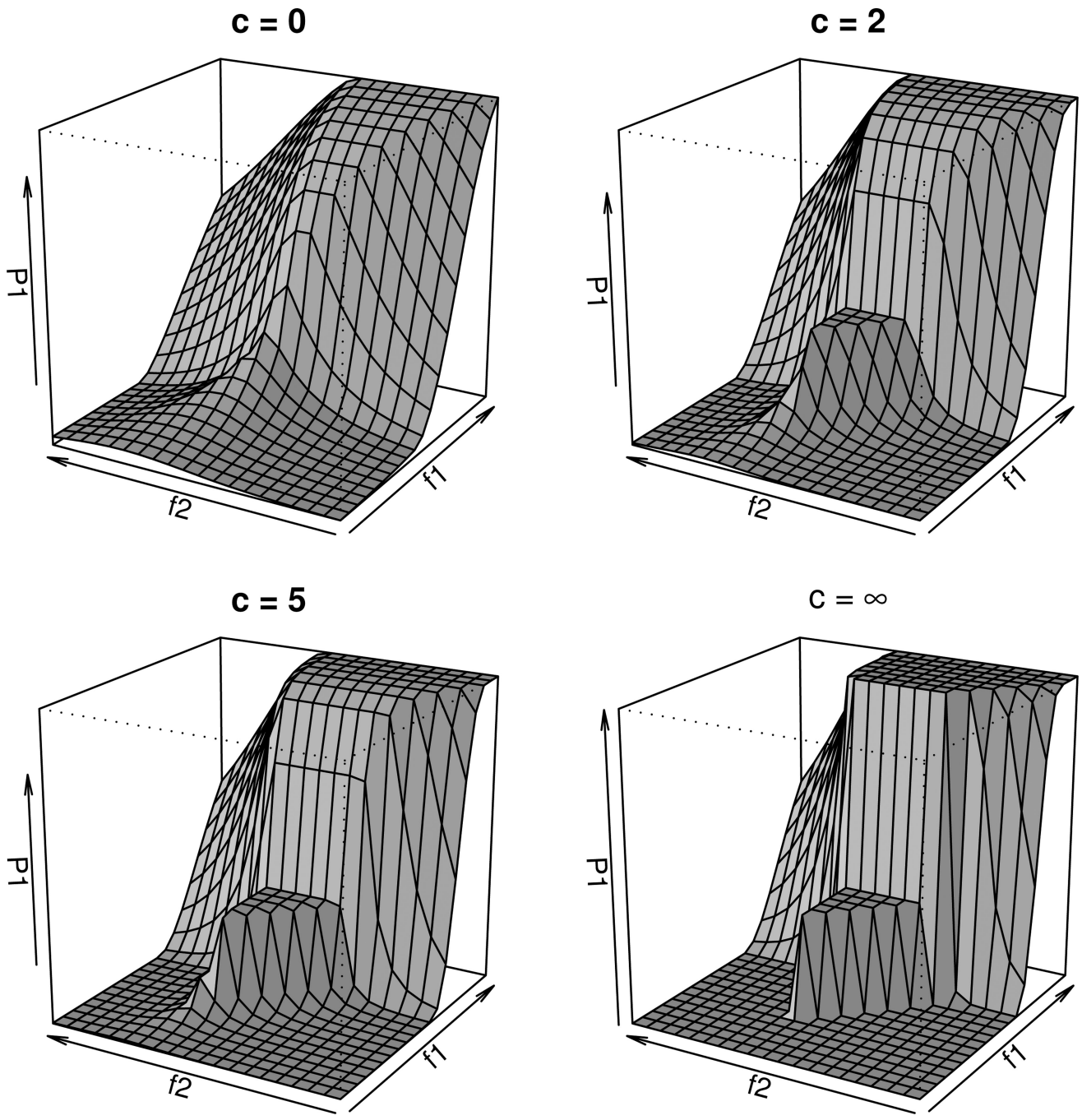


Figure 4.

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).

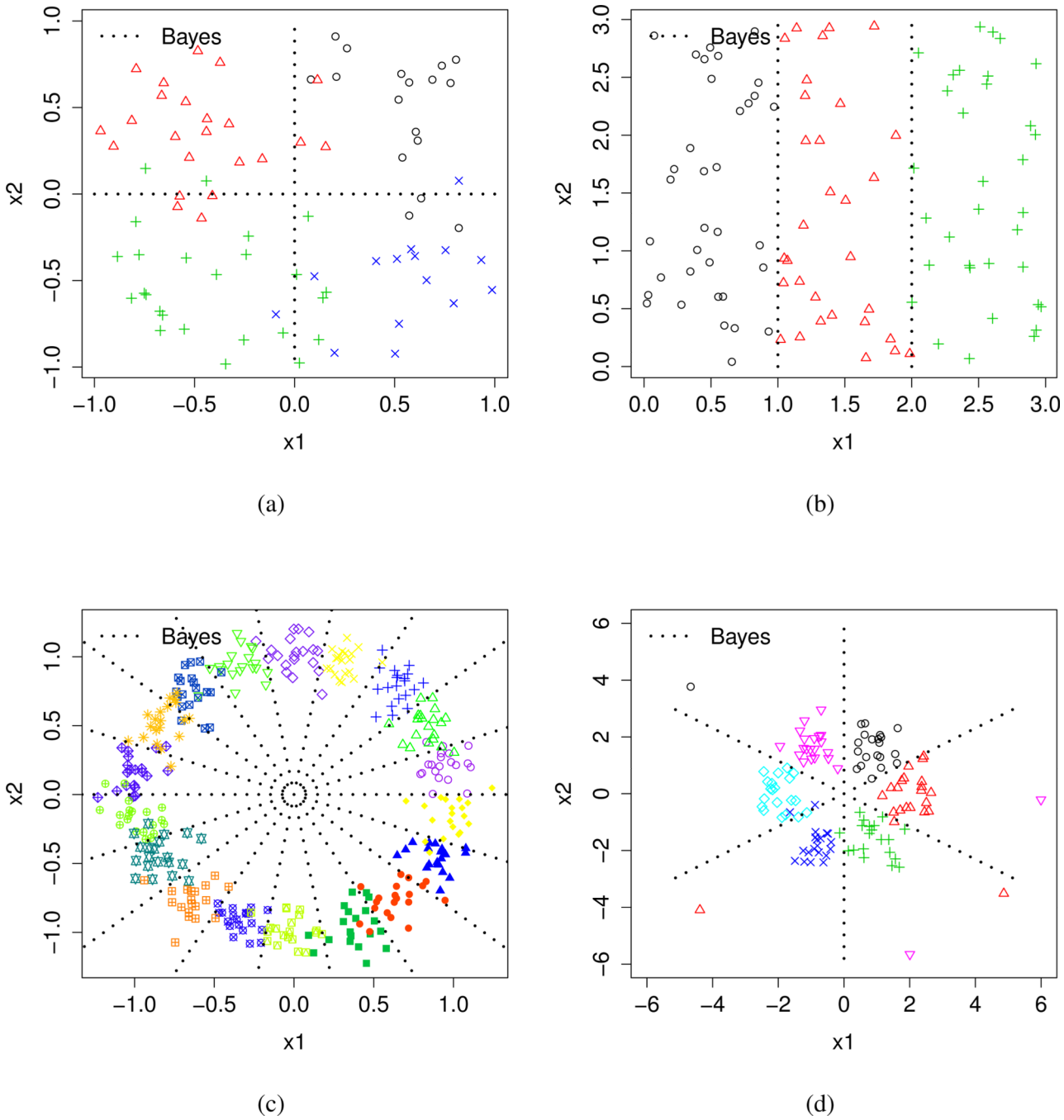
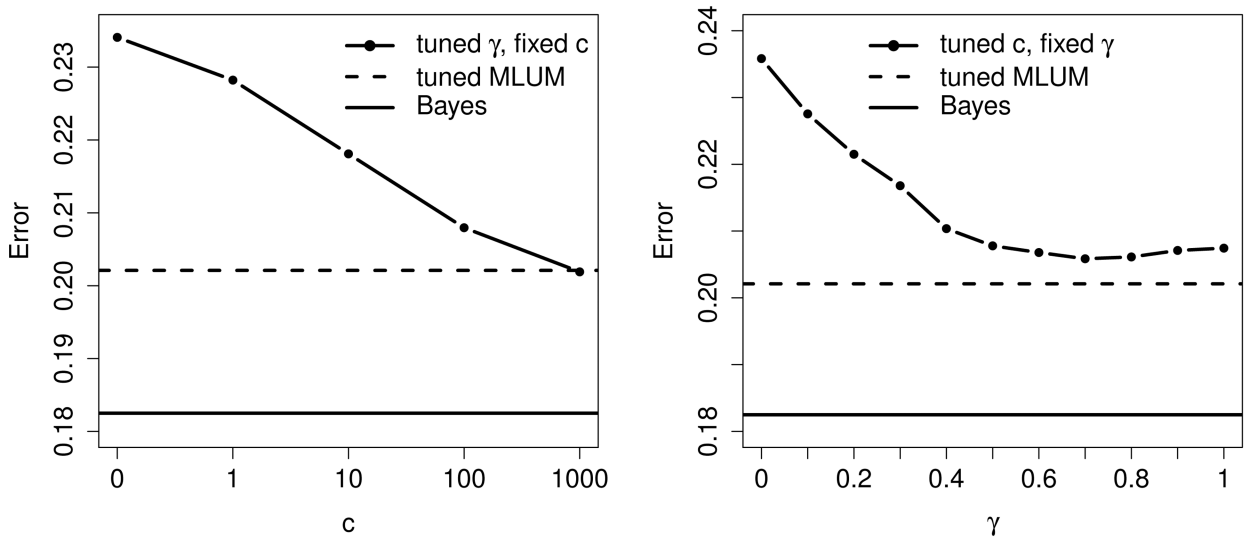


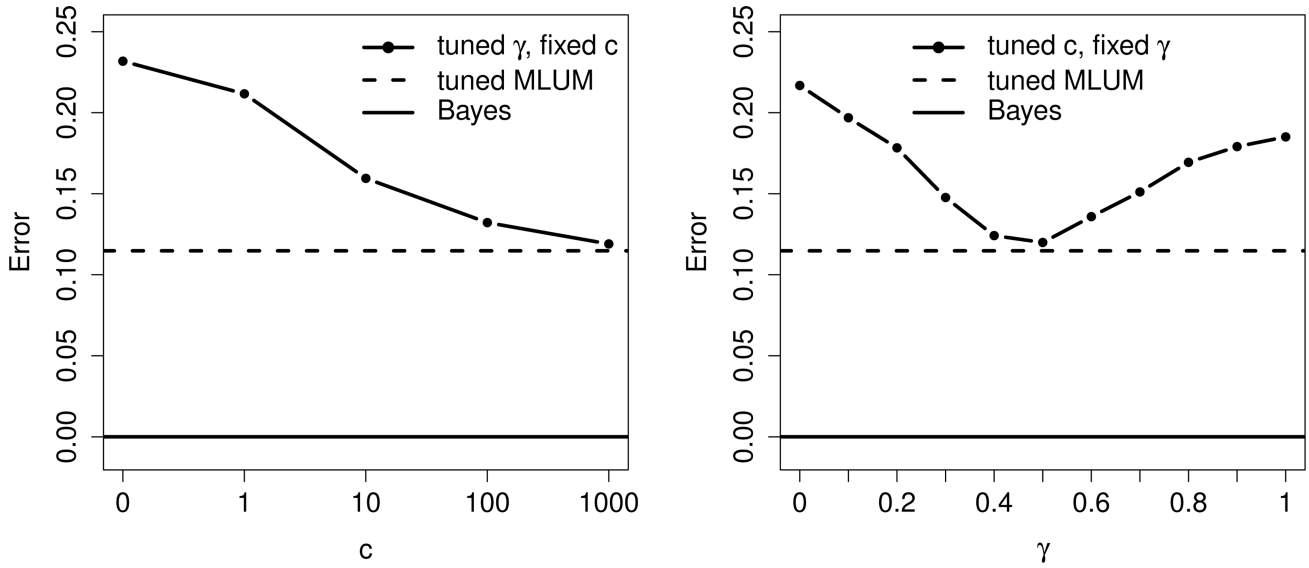
Figure 5.
Plots of typical samples for Examples 1(a), 2(b), 4(c) and 5(d) respectively.



(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.

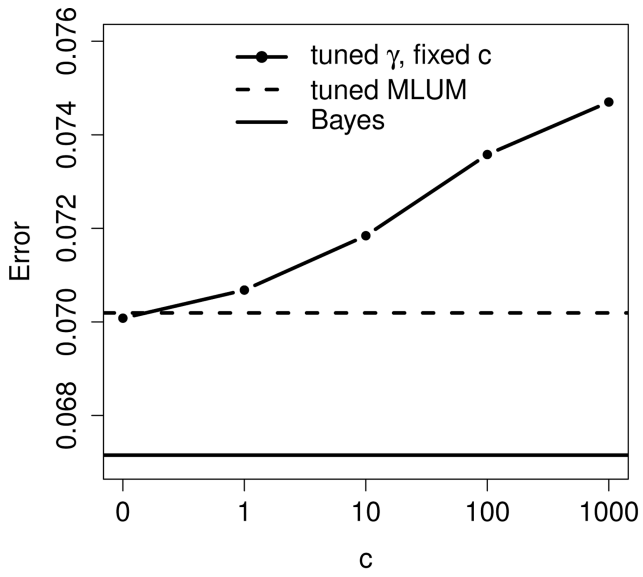
Figure 6.
Classification error rates in Example 1. The left panel shows that the hard classifier performs better than the other ones in this example. The right panel shows that the test error is minimized with γ around 0.7.



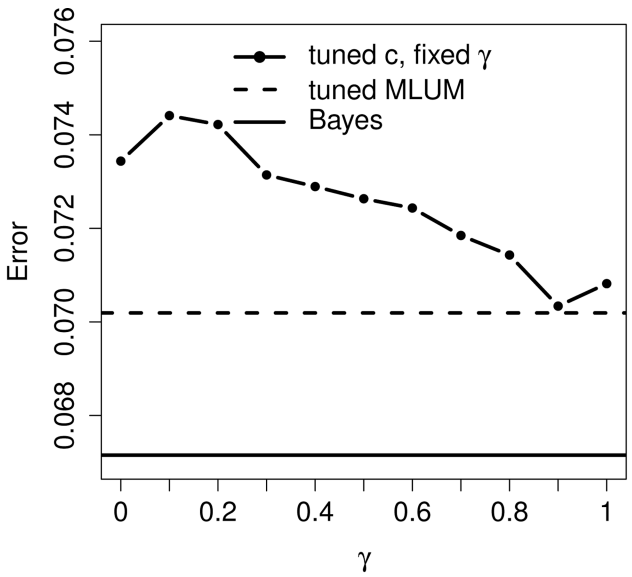
(a) The test error of Example 2 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

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

Figure 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.

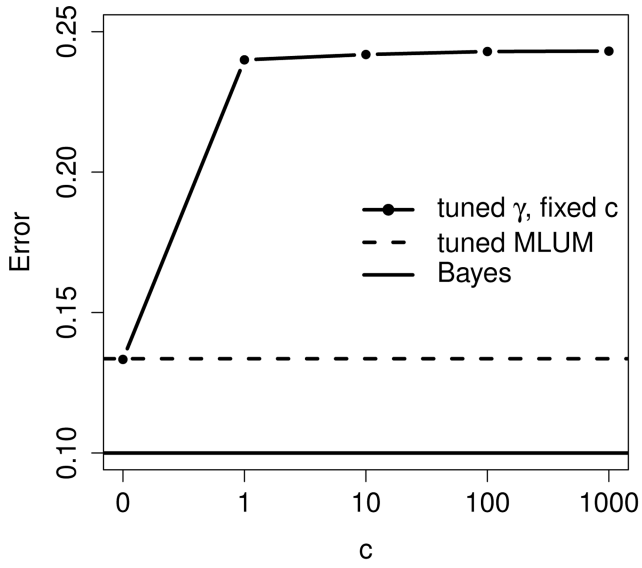


(a) The test error of Example 3 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

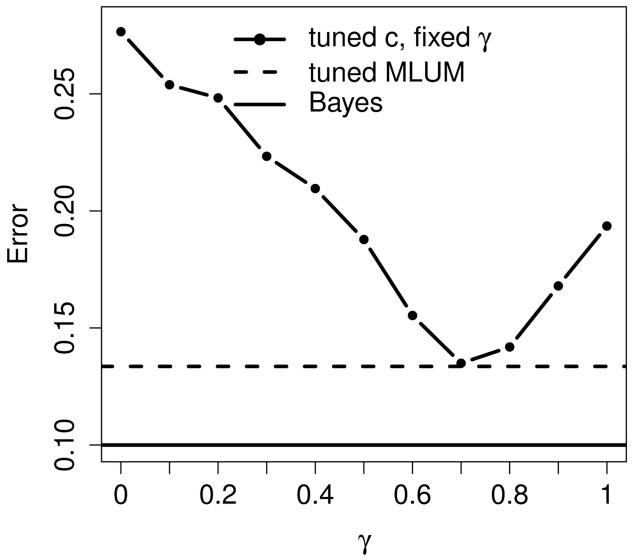


(b) The test error of Example 3 with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 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 $\gamma = 0.9$.

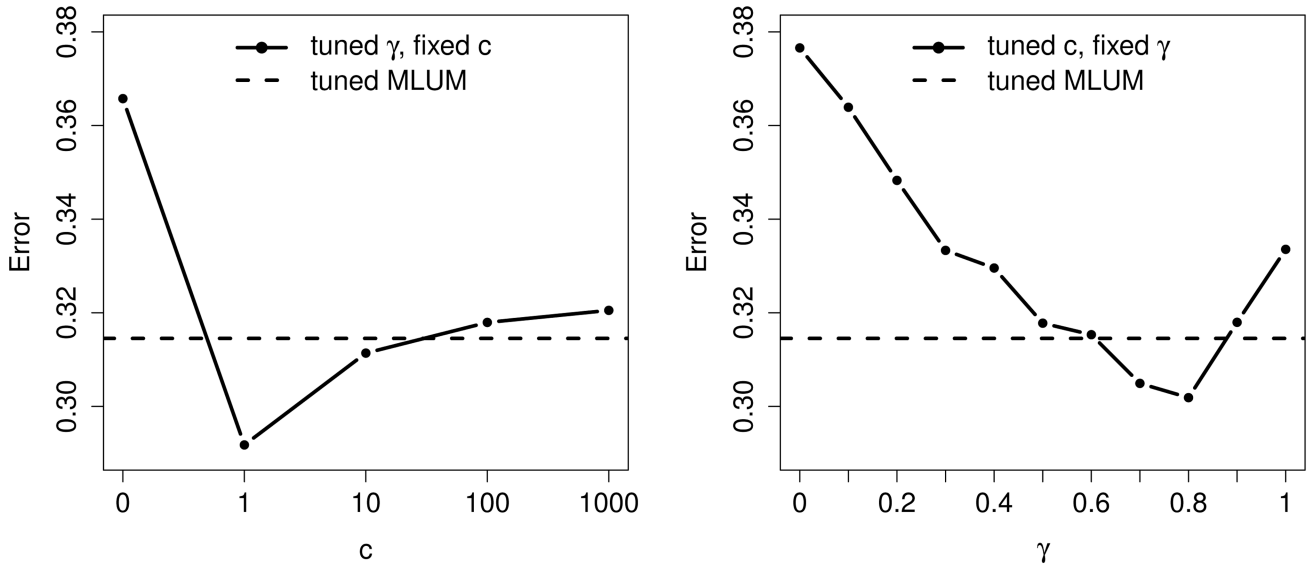


(a) The test error of Example 4 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.



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

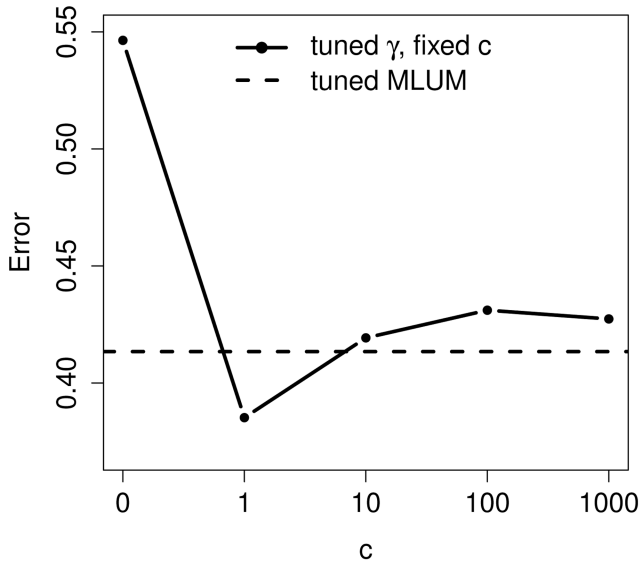
Figure 9.
Classification error rates in Example 4. The left panel shows the soft classifier works reasonably well in this example. When other choices of c fail, tuned MLUM automatically selects $c = 0$ and thus keeps a good performance. In the right panel, when $\gamma = 0.7$, the classifier has a minimum error rate.



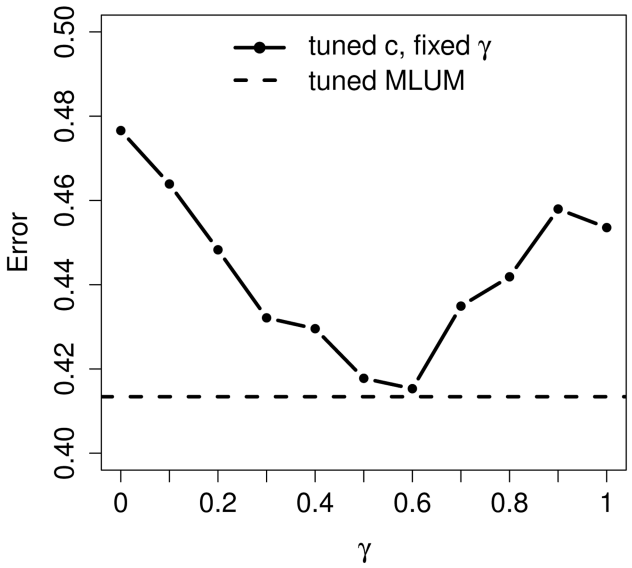
(a) The test error of Example 5 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

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

Figure 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.

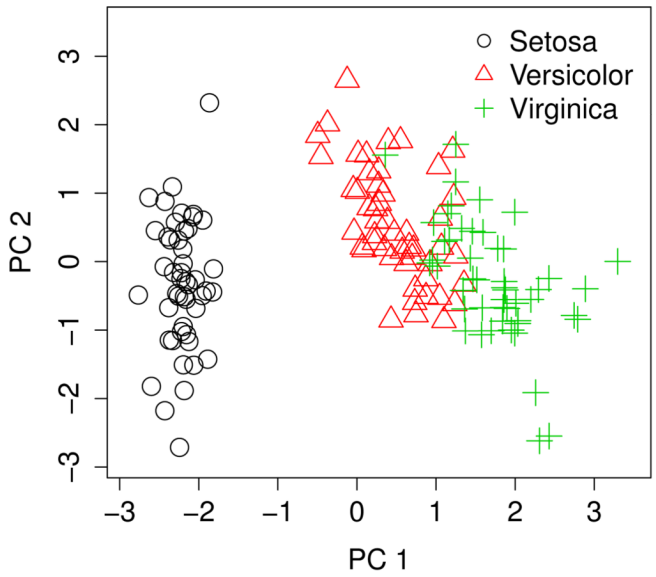


(a) The test error of Example 6 with c in $\{0, 1, 10, 100, 1000\}$ and γ tuned.

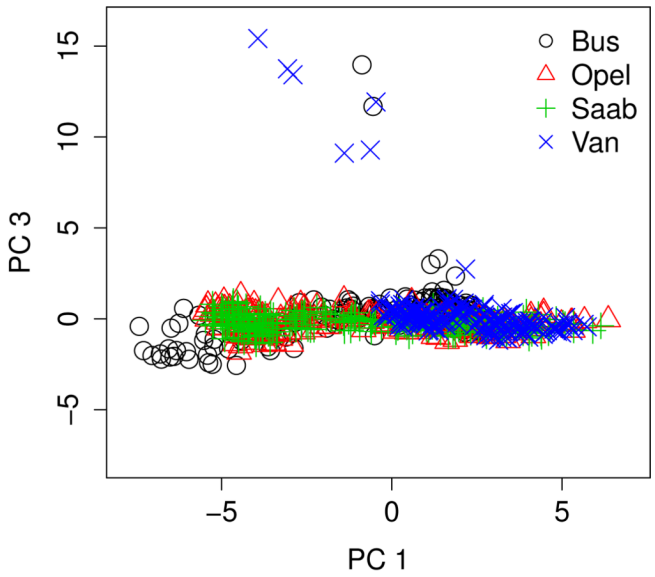


(b) The test error of Example 6 with γ in $\{0, 0.1, \ldots, 0.9, 1\}$ and c tuned.

Figure 11.
Classification error rates in Example 6. In contrast to Example 4, with noisy points in the data set, $c = 0$ performs worse than the other c values. The right panel suggests that the MLUM works well with γ around 0.6.

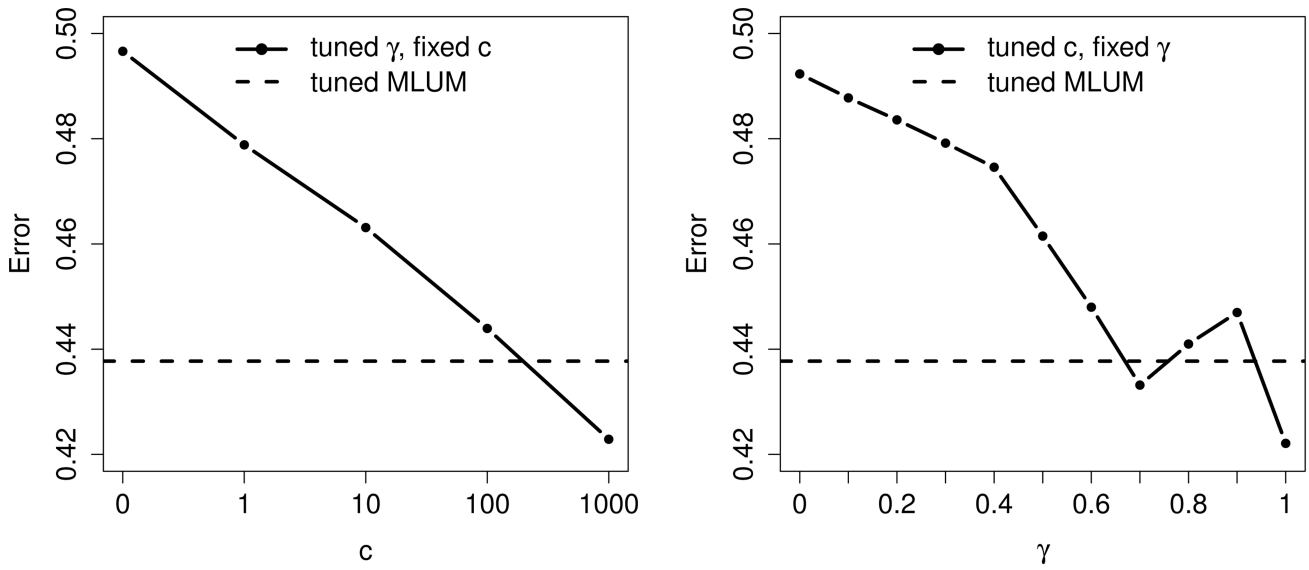


(a) PCA plot for the Iris data, PC1 vs PC2.



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

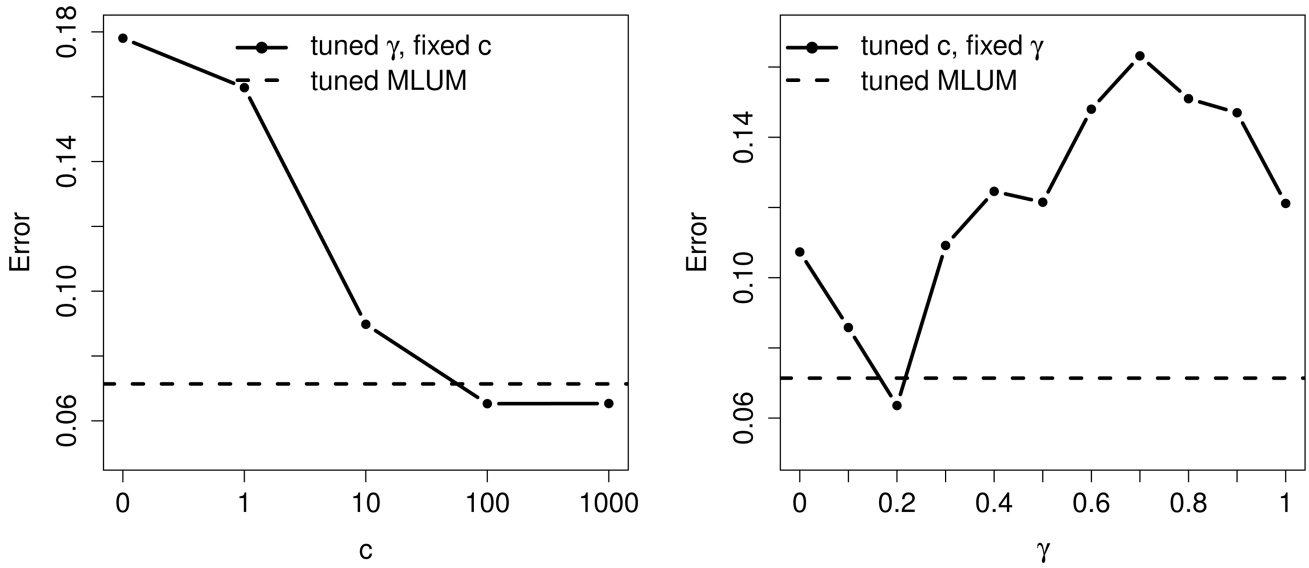
Figure 12.
PCA projection plots for Iris (left panel) and Vehicle (right panel) data. The plots indicate the Iris data appear to be quite clean, while there may exist some outliers for the Vehicle data.



(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.

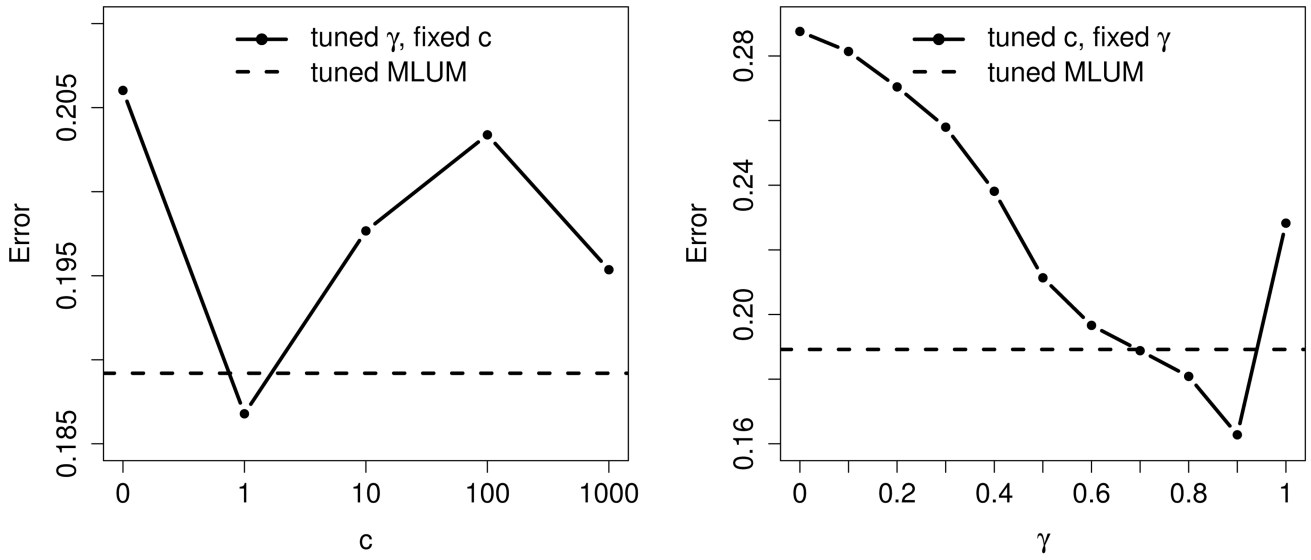
Figure 13.
Classification error rates in the Breast data. The hard classification method dominates the others in terms of classification accuracy, as shown in the left panel. The right panel suggests that the classification error is the best when c is tuned with $\gamma = 1$.



(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.

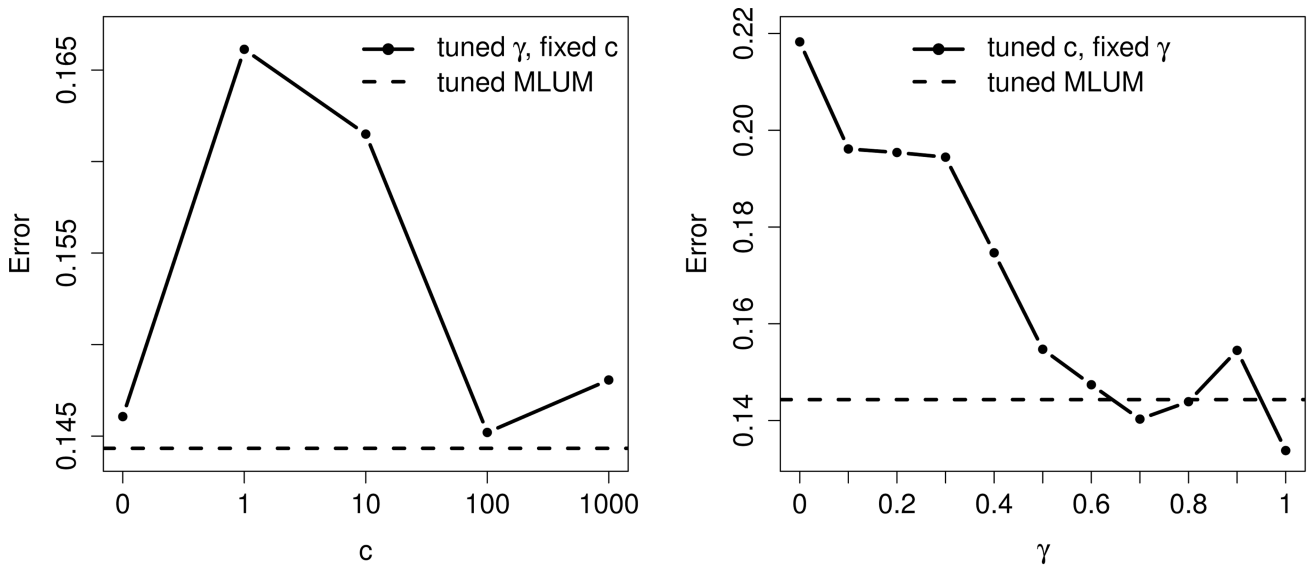
Figure 14.
Classification error rates in the Dermatology data. The hard classifier works the best, as shown in the left panel. The right panel shows that when c is tuned, $\gamma = 0.2$ works better than the others.



(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.

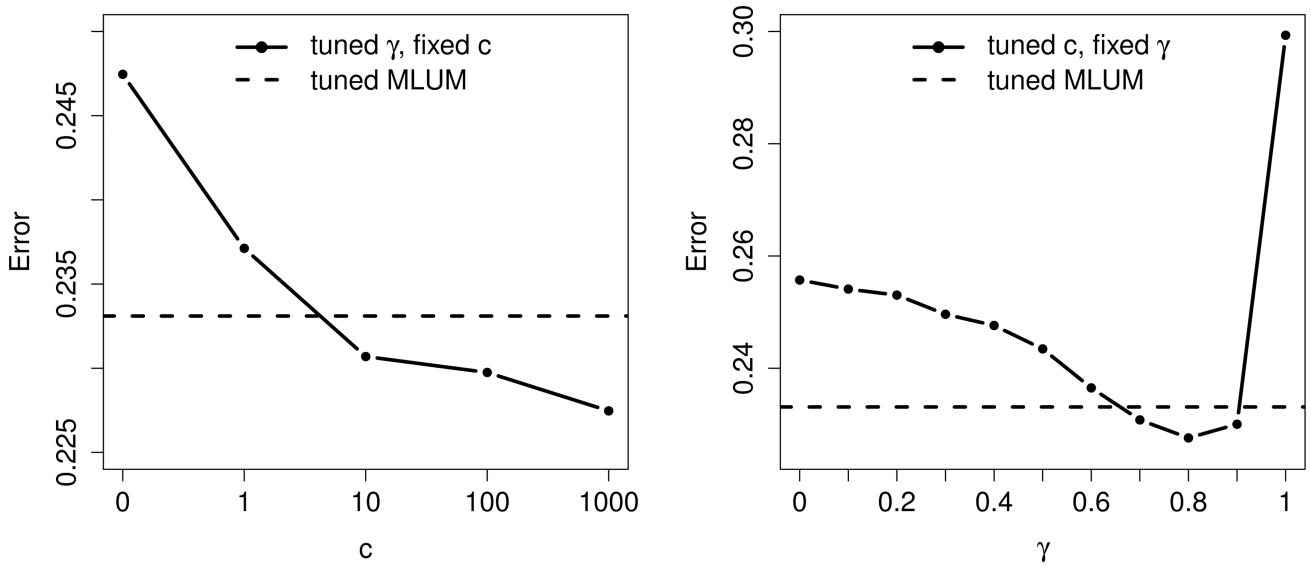
Figure 15.
Classification error rates in the Image data. Left panel shows that $c = 1$ performs the best on the Image data. The right panel shows $\gamma = 0.9$ dominates other values.



(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.

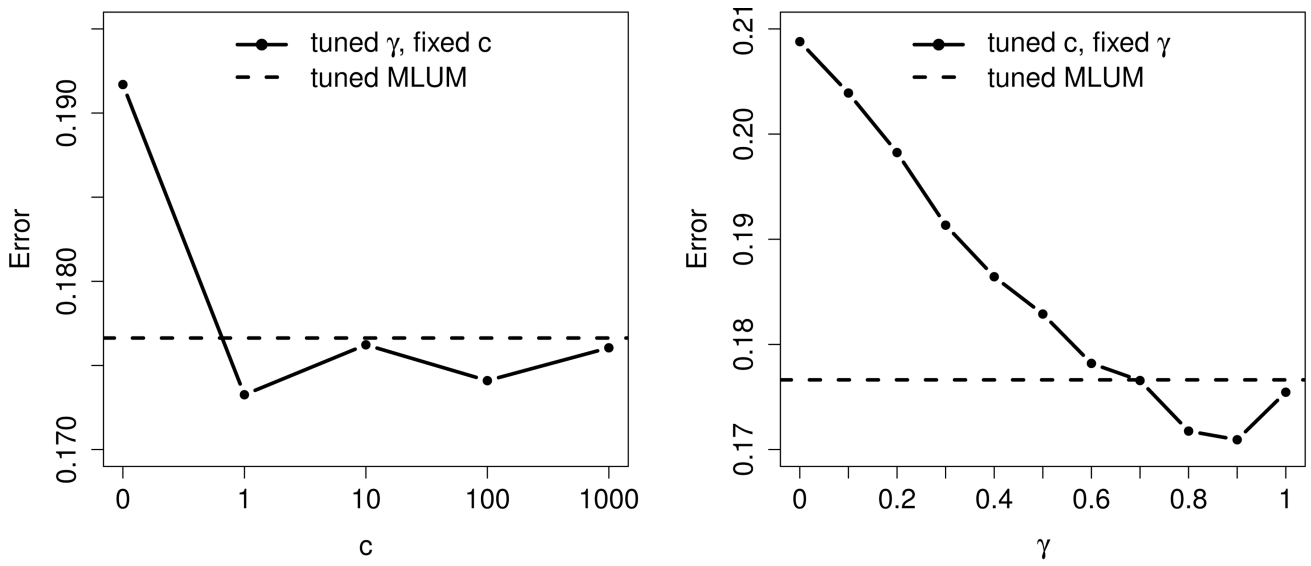
Figure 16.
Classification error rates in the Iris data. The left panel shows the test error of soft and hard classifiers are roughly comparable, while $c = 1$ (DWD) is the worst. The right panel indicates $\gamma = 1$ is the optimal choice, if c is tuned.



(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.

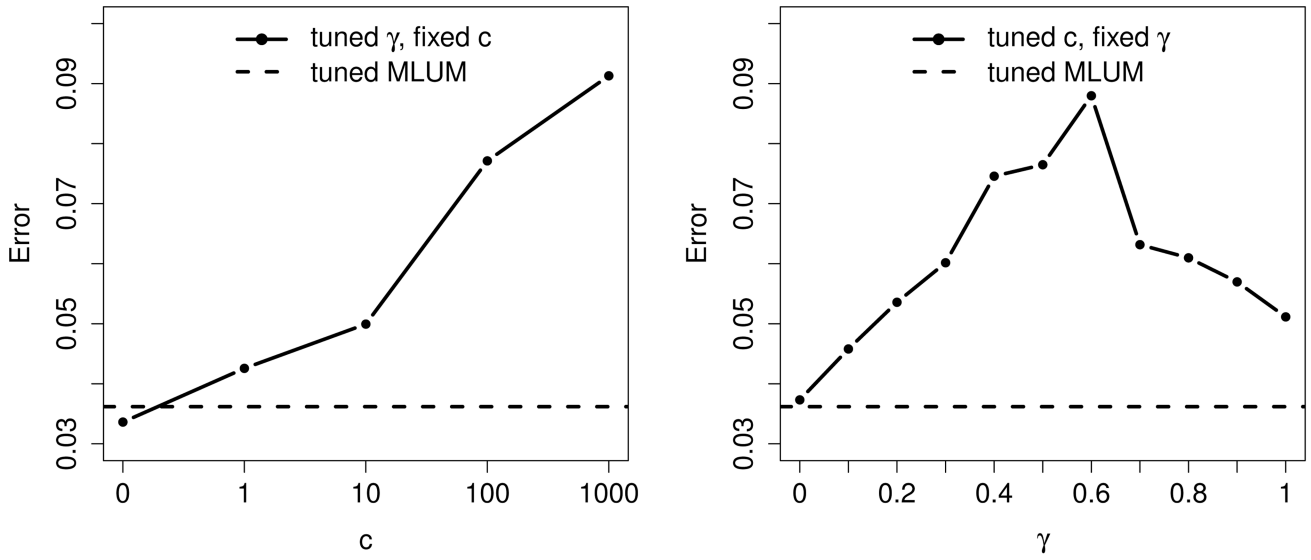
Figure 17.
Classification error rates in the Vehicle data. The hard classifier works better than the other ones, as shown in the left panel. The right panel suggests the best γ is 0.8. Note that the test error significantly increases when γ moves from 0.9 to 1.



(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 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.



(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 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.

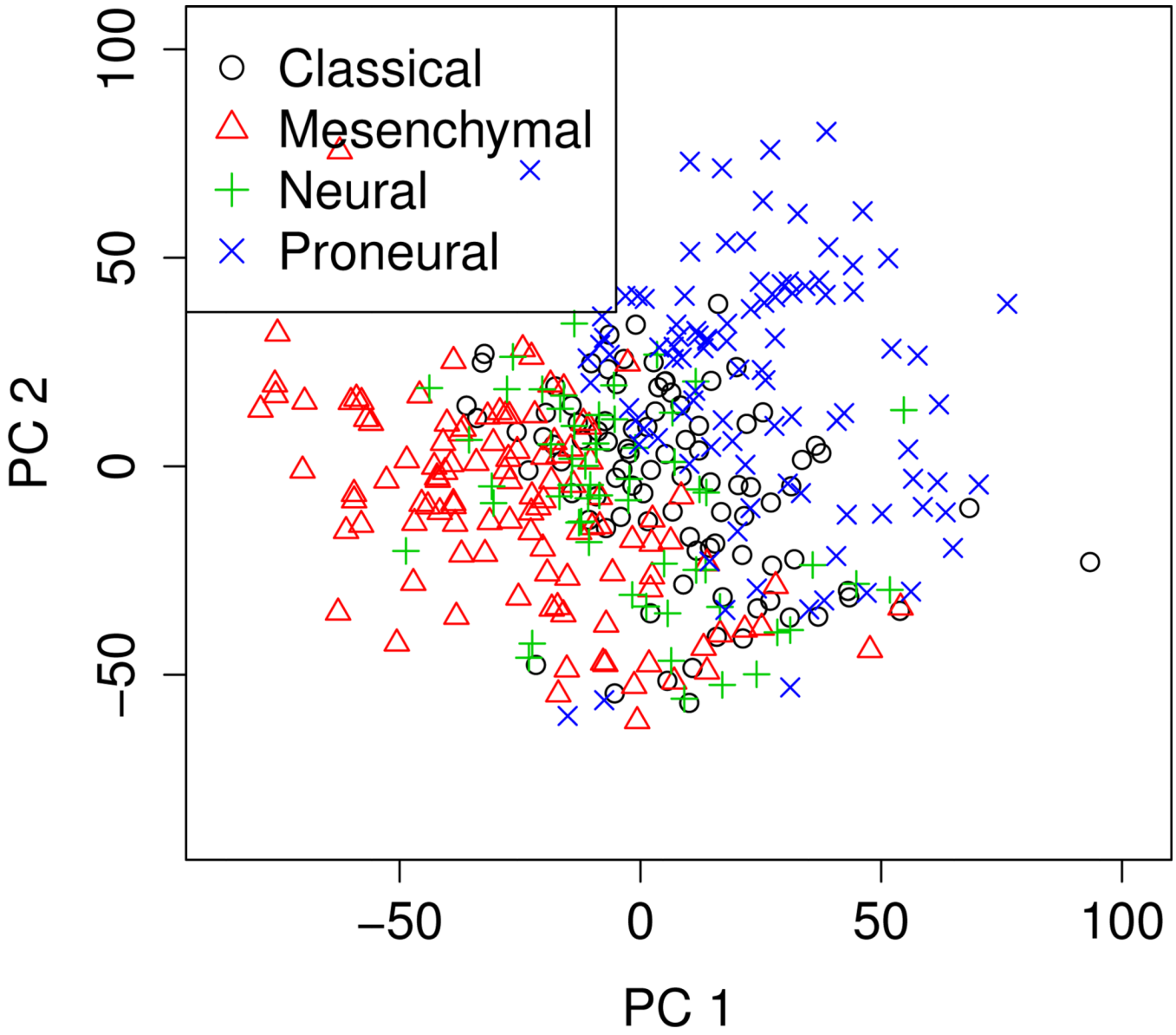
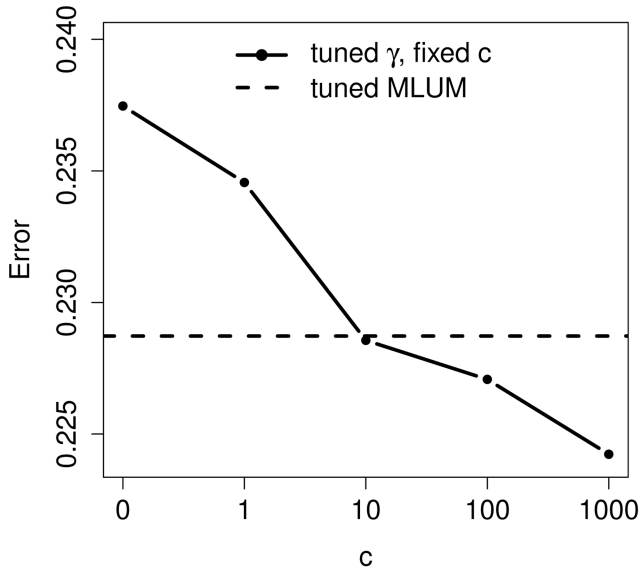
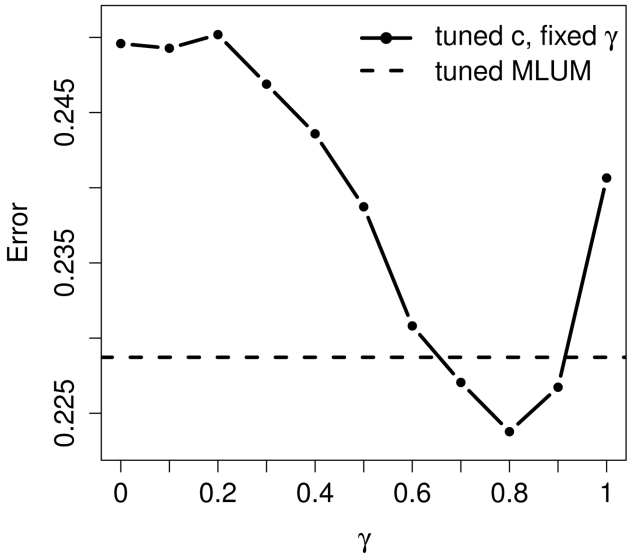


Figure 20.
PCA plot of the GBM data, PC 1 vs PC 2.



(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 21.
For the GBM data, hard classifier works the best, as shown in the left panel. The right panel illustrates that the optimal choice of γ is 0.8.

Table 1

The MAEs for different c and γ in Example 1.

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

Table 2

The MAEs for different c and γ in Example 2.

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

Table 3

The MAEs for different c and γ in Example 3.

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

Table 4

The MAEs for different c and γ in Example 4.

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

Table 5

The MAEs for different c and γ in Example 5.

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

Table 6

The MAEs for different c and γ in Example 6.

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

Table 7

The classification error rates for all simulated examples, for MSVMs with Losses 1–4 defined in Section 2.2, the RMSVM, the tuned MPLR, and the tuned MLUM.

	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 8

Summary of the benchmark data sets in Section 6.1.

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	3	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 9

The classification error rates for all real data examples, for MSVMs with Losses 1–4 defined in Section 2.2, the RMSVM, the tuned MPLR and the tuned MLUM.

	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**