

# Hard or Soft Classification? Large-margin Unified Machines

Yufeng Liu, Hao Helen Zhang, and Yichao Wu\*

## Abstract

Margin-based classifiers have been popular in both machine learning and statistics for classification problems. Among numerous classifiers, some are *hard* classifiers while some are *soft* ones. Soft classifiers explicitly estimate the class conditional probabilities and then perform classification based on estimated probabilities. In contrast, hard classifiers directly target on the classification decision boundary without producing the probability estimation. These two types of classifiers are based on different philosophies and each has its own merits. In this paper, we propose a novel family of large-margin classifiers, namely large-margin unified machines (LUMs), which covers a broad range of margin-based classifiers including both hard and soft ones. By offering a natural bridge from soft to hard classification, the LUM provides a unified algorithm to fit various classifiers and hence a convenient platform to compare hard and soft classification. Both theoretical consistency and numerical performance of LUMs are explored. Our numerical study sheds some light on the choice between hard and soft classifiers in various classification problems.

---

\*Yufeng Liu is Associate Professor, Department of Statistics and Operations Research, Carolina Center for Genome Sciences, University of North Carolina, Chapel Hill, NC 27599 (E-mail: yfliu@email.unc.edu). Hao Helen Zhang is Associate Professor, Yichao Wu is Assistant Professor, Department of Statistics, North Carolina State University (E-mail: hzhang2@stat.ncsu.edu; wu@stat.ncsu.edu). The authors are partially supported by NSF Grants DMS-0747575 (Liu), DMS-0645293 (Zhang) and DMS-0905561 (Wu), NIH Grants NIH/NCI R01 CA-149569 (Liu and Wu) , NIH/NCI P01 CA142538 (Liu and Zhang), and NIH/NCI R01 CA-085848 (Zhang).

**Keywords:** Class Probability Estimation, DWD, Fisher Consistency, Regularization, SVM.

## 1 Introduction

Classification is a very useful statistical tool for information extraction from data. As a supervised learning technique, the goal of classification is to construct a classification rule based on a training set where both covariates and class labels are given. Once obtained, the classification rule can then be used for class prediction of new objects whose covariates are available.

There is a large amount of literature on various classification methods, ranging from the very classical ones such as Fisher linear discriminant analysis (LDA) and logistic regression (Hastie et al., 2001), to the recent machine learning based ones such as the Support Vector Machine (SVM) (Boser et al., 1992; Cortes and Vapnik, 1995) and Boosting (Freund and Schapire, 1997; Friedman et al., 2000). Among various classification methods, there are two main groups of methods: *soft* and *hard* classification. Our notions of soft and hard classification are the same as those defined in Wahba (1998) and Wahba (2002). In particular, a soft classification rule generally estimates the class conditional probabilities explicitly and then makes the class prediction based on the largest estimated probability. In contrast, hard classification bypasses the requirement of class probability estimation and directly estimates the classification boundary. Typical soft classifiers include some traditional distribution-based likelihood approaches such as LDA and logistic regression. On the other hand, some margin-based approaches such as the SVM, generally distributional assumption-free, belong to the class of hard classification methods.

Given a particular classification task, one natural question to ask is that which type of classifiers should be used? Though a large number of classifiers are available, as it is typically the case, there is no single method working best for all problems. The choice

of classifiers really depends on the nature of the data set and the primary learning goal. Wahba (2002) provided some insights on soft versus hard classification. In particular, she demonstrated that the penalized logistic regression (PLR, Lin et al. (2000)) and the SVM can both be fit into optimization problems in Reproducing Kernel Hilbert Spaces (RKHS). However, the choice between the PLR and SVM for many practical problems is not clear. Recent rapid advances in high dimensional statistical data analysis also shed some light on this issue. With the large amount of high dimension low sample size (HDLSS) data available, effective statistical techniques for analyzing HDLSS data become more pressing. Traditional techniques such as LDA cannot even be computed directly when the dimension is larger than the sample size. Certain transformation or dimension reduction is needed in order to apply LDA. Margin-based methods such as the SVM provide a totally different view from the likelihood based approaches. For example, the SVM does not have any distributional assumption and focuses on the decision boundary only. It can be efficiently implemented for HDLSS data and has achieved great success in many applications (Vapnik, 1998; Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002). Recently, Marron et al. (2007) pointed out that the SVM has the “data piling” phenomena in HDLSS settings due to its non-differentiable hinge loss. In particular, when we project the training data onto the norm vector of separating hyperplane for the linear SVM in high dimensional problems, many projections are the same. They proposed a SVM variant, namely distance discriminant analysis (DWD), which does not have the data piling problem.

Between the two types of classifiers, soft classification provides more information than hard classification and consequently it is desirable in certain situations where the probability information is useful. However, if the class probability function is hard to estimate in some complicated problems, hard classification may yield more accurate classifiers by targeting on the classification boundary only (Wang et al., 2008). In practice, it is difficult to choose between hard and soft classifiers, and consequently, it will be ideal to connect them since each has its own strength. In this paper, we

propose a unified framework of large-margin classifiers which covers a broad range of methods from hard to soft classifiers. This new framework offers a unique transition from soft to hard classification. We call this family as Large-margin Unified Machines (LUMs). The proposed LUM family not only covers some well known large-margin classifiers such as the standard SVM and DWD, it also includes many new classifiers. One interesting example in the LUM family is the new hybrid of SVM and Boosting.

One important contribution of the LUM is that it sheds some light on the choice between hard and soft classifiers. Our intensive numerical study suggests that

- soft classifiers tend to work better when the underlying conditional class probability function is relatively smooth; or when the class signal level is relatively weak;
- hard classifiers tend to work better when the underlying conditional class probability function is relatively non-smooth; or when the two classes are close to be separable, i.e., the class signal level is relatively strong; or when the dimension is relatively large compared to the sample size.

Certainly, our observations may not be valid for all classification problems. Nevertheless, the LUM family provides a convenient platform for deep investigation of the nature of a classification problem. We propose an efficient tuning procedure for the LUMs, and the resulting tuned LUM is shown to give near-optimal classification performance compared with various classifiers in the LUM family. Thus we recommend it as a competitive classifier. Furthermore, we develop probability estimation techniques for the LUM family including hard classifiers such as the SVM. The probability estimation method using the refitted LUM provides very competitive probability estimation for both soft and hard classifiers in the LUM family.

Another major advantage of the LUM is its unified computational strategies for different classifiers in the family. In practice, different methods are often implemented with different method-specific algorithms. For example, the SVM is typically fitted

via solving a quadratic programming problem. The DWD was proposed to be computed using the second order cone programming (Marron et al., 2007). For Boosting in the classification setting, it can be implemented using the AdaBoost algorithm (Freund and Schapire, 1997). In this paper, a unified computational algorithm is developed for the LUMs to solve seemingly very different methods in the family such as the SVM and DWD. This greatly facilitates the implementation and comparison of various classifiers on a given problem. We further propose a simple gradient descent algorithm to handle high dimensional problems.

The remaining of the article is organized as follows. Section 2 introduces the proposed LUM methodology and discusses some interesting special cases. Section 3 tackles the issues of Fisher consistency and class probability estimation. Section 4 addresses the computational aspect of LUMs. Section 5 includes intensive simulated examples to demonstrate the performance of LUMs. Section 6 discusses two real data examples. Section 7 gives some final discussions. The appendix contains the technical proofs.

## 2 The LUM Methodology

In supervised learning, we are given a training sample  $\{(\mathbf{x}_i, y_i); i = 1, 2, \dots, n\}$ , distributed according to some unknown probability distribution  $P(\mathbf{x}, y)$ . Here,  $\mathbf{x}_i \in \mathcal{S} \subset \mathbb{R}^d$  and  $y_i$  denote the input vector and output variable respectively,  $n$  is the sample size, and  $d$  is the dimensionality of  $\mathcal{S}$ . Consider a two-class problem with  $y \in \{\pm 1\}$  and let  $p(\mathbf{x}) = P(Y = 1 | \mathbf{X} = \mathbf{x})$  be the conditional probability of class 1 given  $\mathbf{X} = \mathbf{x}$ . Let  $\phi(\mathbf{x}) : \mathbb{R}^d \rightarrow \{\pm 1\}$  be a classifier and  $C_{y\phi(\mathbf{x})}$  represent the cost of misclassifying input  $\mathbf{x}$  of class  $y$  into class  $\phi(\mathbf{x})$ . We set  $C_{y\phi(\mathbf{x})} = 0$  if  $\phi(\mathbf{x}) = y$  and  $C_{y\phi(\mathbf{x})} > 0$  otherwise. One important goal of classification is to obtain a classifier  $\phi(\mathbf{x})$  which can deliver the best classification accuracy. Equivalently, we aim to estimate the Bayes rule,  $\phi_B(\mathbf{x})$ , minimizing the generalization error (GE),  $\text{Err}(\phi) = E[C_{Y\phi(\mathbf{X})}]$ . When equal costs are used with  $C_{jl} = 1; \forall j \neq l$ , the Bayes rule can be reduced to  $\phi_B(\mathbf{x}) = \text{sign}(p(\mathbf{x}) - 1/2)$ .

The focus of this paper is on large-margin classifiers. Among various margin-based methods, the SVM is perhaps the most well known one. While the SVM was first invented in the machine learning community, it overlaps with classical statistics problems, such as nonparametric regression and classification. It is now known that the SVM can be fit in the regularization framework of *Loss + Penalty* using the hinge loss (Wahba, 1998). In the regularization framework, the loss function is used to keep the fidelity of the resulting model to the data. The penalty term in regularization helps to avoid overfitting of the resulting model. Specifically, margin-based classifiers try to calculate a function  $f(\mathbf{x})$ , a map from  $\mathcal{S}$  to  $\mathbb{R}$ , and use  $\text{sign}(f(\mathbf{x}))$  as the classification rule. By definition of the classification rule, it is clear that  $\text{sign}(yf(\mathbf{x}))$  reflects the classification result on the point  $(\mathbf{x}, y)$ . Correct classification occurs if and only if  $yf(\mathbf{x}) > 0$ . The quantity  $yf(\mathbf{x})$  is commonly referred as the *functional margin* and it plays a critical role in large-margin classification techniques. In short, the regularization formulation of binary large-margin classifiers can be summarized as the following optimization problem

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

where the minimization is taken over some function class  $\mathcal{F}$ ,  $J(f)$  is the regularization term,  $\lambda > 0$  is a tuning parameter, and  $V(\cdot)$  is a loss function. For example, the SVM uses the hinge loss function  $V(u) = [1 - u]_+$ , where  $(u)_+ = u$  if  $u \geq 0$  and 0 otherwise.

Besides the SVM, many other classification techniques can be fit into the regularization framework, for example, the PLR (Lin et al., 2000), the AdaBoost in Boosting (Freund and Schapire, 1997; Friedman et al., 2000), the import vector machine (IVM; Zhu and Hastie (2005)),  $\psi$ -learning (Shen et al., 2003; Liu and Shen, 2006), the robust SVM (RSVM, Wu and Liu (2007)), and the DWD (Marron et al., 2007).

Among various large margin classifiers, some are hard classifiers which only focus on estimating the decision boundary. The SVM is a typical example of hard classifiers (Wang et al., 2008). In contrast to hard classifiers, soft classifiers estimate the class conditional probability and the decision boundary simultaneously. For ex-

ample, the PLR and the AdaBoost can be classified as soft classifiers (Zhang, 2004; Tewari and Bartlett, 2005). One may wonder which one is more preferable, hard or soft classification? The answer truly depends on the problem given. In this paper, we propose a family of large-margin classifiers which covers a spectrum of classifiers from soft to hard ones. In particular, for given  $a > 0$  and  $c \geq 0$ , define the following family of new loss functions

$$V(u) = \begin{cases} 1 - u & \text{if } u < \frac{c}{1+c}, \\ \frac{1}{1+c} \left( \frac{a}{(1+c)u - c + a} \right)^a & \text{if } u \geq \frac{c}{1+c}. \end{cases} \quad (2)$$

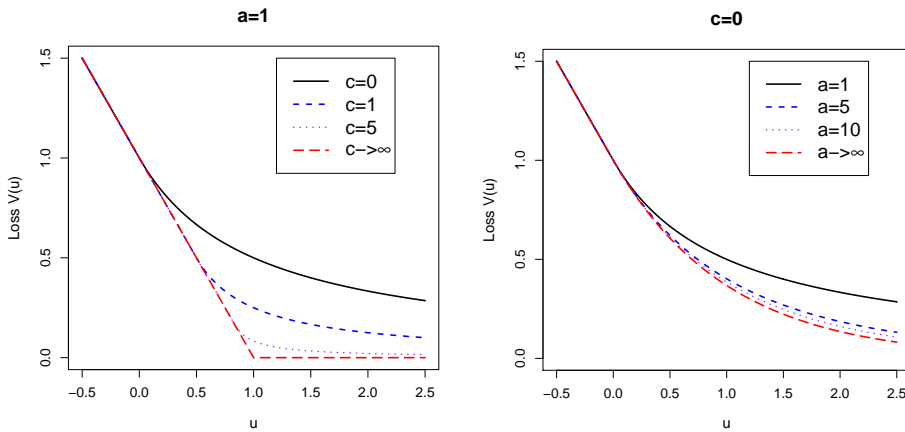


Figure 1: Plots of LUM loss functions. Left panel:  $a = 1$ ,  $c = 0, 1, 5, \infty$ ; Right panel:  $c = 0$ ,  $a = 1, 5, 10, \infty$ .

Note that  $V(u)$  in (2) has two different pieces, one for  $u < \frac{c}{1+c}$  and one for  $u \geq \frac{c}{1+c}$ , which are smoothly joined together at  $\frac{c}{1+c}$ . When  $u < c/(1+c)$ ,  $V(u)$  is same as the hinge loss,  $[1 - u]_+$ . Note that the hinge loss is not differentiable at  $u = 1$ , while  $V(u)$  is differentiable for  $\forall u$ . Figure 1 displays several LUM loss functions for different values of  $a$  and  $c$ . As we can see from Figure 1,  $a$  and  $c$  play different roles in the loss function:  $c$  controls the connecting point between the two pieces of the loss as well as the shape of the right piece, and  $a$  determines the decaying speed for the right piece.

We next show that the LUM is a very rich family, and in particular, it includes several well-known classifiers as special cases.

## 2.1 $a > 0$ and $c \rightarrow \infty$ : The Standard SVM

As pointed out before, the parameter  $c$  specifies the location of the connection point of the two pieces of  $V(u)$  in (2). The connection point  $u = c/(1+c)$  is between 0 and 1. As  $c$  increases, the connection point approaches 1 and the value  $V(u)$  for  $u > c/(1+c)$  becomes 0 when  $c \rightarrow \infty$ . Thus, the hinge loss of the SVM becomes a limiting case in the LUM family. This is also reflected in Figure 1.

## 2.2 $a = 1$ and $c = 1$ : DWD

Marron et al. (2007) pointed out that for HDLSS data, the SVM may suffer from “data piling” at the margin which may reduce generalizability. To solve the problem, they proposed a different classifier, namely DWD. The idea was motivated from the maximum separation idea of the SVM. In particular, the SVM tries to separate the two classes as much as possible without taking into account of correctly classified points far away from the boundary. Marron et al. (2007) argued that this property of the SVM can result in the “data piling” problem in HDLSS data. The method of DWD modifies the SVM by allowing all points to influence the solution. It gives high significance to those points that are close to the hyperplane, with little impact from correct points that are further away. Recently, Qiao et al. (2010) proposed weighted DWD, as an extension of the standard DWD.

Despite its interesting connection with the SVM, the original DWD proposal is not in the typical form  $Loss + Penalty$  of regularization. In this section, we show that DWD is a special case of the LUM family which corresponds to the LUM loss with  $a = 1$  and  $c = 1$ . This result can help us to further understand DWD in the regularization prospective.

For simplicity, consider  $f(\mathbf{x}) = \mathbf{w}'\mathbf{x} + b$ . The original DWD proposed by Marron et al. (2007) solves the following optimization problem

$$\begin{aligned} & \min_{\boldsymbol{\gamma}, \mathbf{w}, b, \boldsymbol{\xi}} \sum_i (1/\gamma_i) + C \sum_{i=1}^n \xi_i & (3) \\ \text{subject to} & \quad \gamma_i = y_i f(\mathbf{x}_i) + \xi_i, \mathbf{w}'\mathbf{w} \leq 1, \gamma_i \geq 0, \xi_i \geq 0, \forall i. \end{aligned}$$

Theorem 1 shows that the DWD is a special case of LUMs with  $a = 1$  and  $c = 1$ . Consequently, it provides more insight on the property of the DWD.

**Theorem 1.** *With a one-to-one correspondence between  $C$  and  $\lambda$ , the DWD optimization problem in (3) is equivalent to*

$$\min_{\mathbf{w}, b} V(y_i f(\mathbf{x}_i)) + \frac{\lambda}{2} \mathbf{w}'\mathbf{w}, \quad (4)$$

where  $V$  is the LUM loss function with  $a = 1$  and  $c = 1$ .

### 2.3 $a \rightarrow \infty$ and fixed $c$ : Hybrid of SVM and AdaBoost

Boosting is one of the most important statistical learning ideas in the past decade. It was originally proposed for classification problems by Freund and Schapire (1997) and has been extended to many other areas in statistics. The idea of the original boosting algorithm AdaBoost is to apply the weak classification algorithm sequentially to weighted versions of the data. Then weak classifiers are combined to produce a strong one. Friedman et al. (2000) showed that the AdaBoost algorithm is equivalent to forward stagewise additive modeling using the exponential loss function  $V(u) = \exp(-u)$ .

It turns out that the special case of the LUM family with  $a \rightarrow \infty$  and any fixed  $c \geq 0$  has a very interesting connection to the exponential loss. For any given  $c \geq 0$ , let  $a \rightarrow \infty$ , then  $V(u)$  becomes

$$V(u) = \begin{cases} 1 - u & \text{if } u < \frac{c}{1+c} \\ \frac{1}{(1+c)} \exp\{-[(1+c)u - c]\} & \text{if } u \geq \frac{c}{1+c}. \end{cases} \quad (5)$$

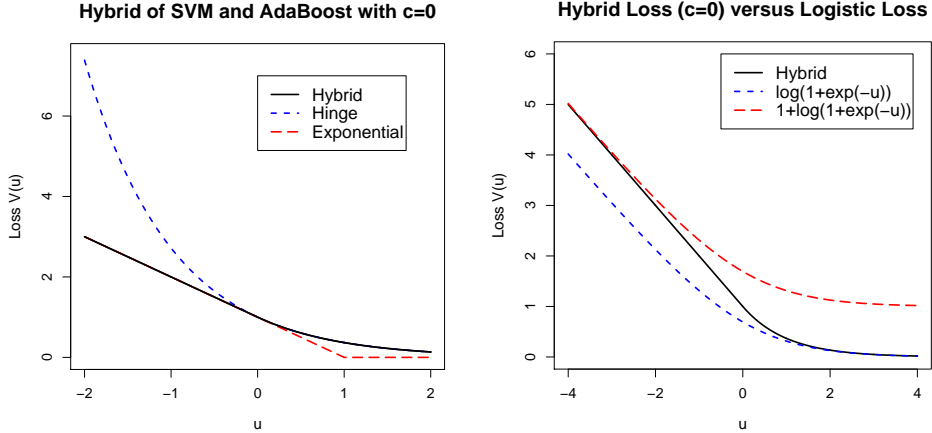


Figure 2: Left Panel: Plot of a hybrid loss of the hinge loss and the exponential loss with  $c = 0$ ; Right Panel: Plot of the hybrid loss with  $c = 0$  versus the logistic loss.

In particular, when  $c = 0$ , then  $V(u) = 1 - u$  for  $u < 0$  and  $e^{-u}$  otherwise, leading to the combination of the hinge loss and exponential loss. As a result, this limiting case of the LUM loss function can be viewed as a hybrid of the standard SVM and the AdaBoost.

The left panel of Figure 2 displays the hinge loss, the exponential loss, and the LUM loss with  $a \rightarrow \infty$  and  $c = 0$  as a hybrid loss of the two well known loss functions. Interestingly, the hybrid loss makes use of the exponential loss for the right side, consequently, all correctly classified points in the training data can influence the decision boundary. The hinge loss, in contrast, does not make use of the data information once its functional margin is larger than 1. For the left side of the hybrid loss, it uses the hinge loss which is much smaller and thus much closer to the 0-1 loss function. This can yield robustness of the resulting classifiers when there are outliers in the data.

The hybrid loss with  $c = 0$  also has an interesting connection with the logistic loss  $V(u) = \log(1 + \exp(-u))$ . Note that when  $u$  gets large, the logistic loss and exponential loss are very close with  $\lim_{u \rightarrow \infty} (\log(1 + \exp(-u))) / (\exp(-u)) = 1$ . When  $u$  becomes negative and gets sufficiently small,  $\log(1 + \exp(-u))$  is approximately  $-u$  and differs from the hinge loss by 1. Thus, the hybrid loss with  $c = 0$  lies between the logistic loss

$\log(1 + \exp(-u))$  and the logistic loss  $+1$ , i.e.,  $1 + \log(1 + \exp(-u))$ . This relationship can be visualized from the right panel of Figure 2. As shown in the plot, the right side of this hybrid loss behaves like the logistic loss when  $u$  gets sufficiently large. At the same time, the left side of the hybrid loss is approximately the same as the logistic loss  $+1$ . The main difference comes from the region of  $u$  around 0, where the hybrid loss penalizes the wrong classified points around the boundary more severely than the logistic loss. This difference may help the hybrid loss deliver better performance than the logistic loss in certain situations as shown in our simulation study in Section 5.

In Section 3, we will further study the aspect of class probability estimation of various loss functions in the LUM family, which can provide more insight on the behaviors of different loss functions.

### 3 Fisher Consistency and Probability Estimation

To gain further theoretical insight about LUMs, we study the statistical properties of the population minimizers of the LUM loss functions in this section. Since LUMs include both hard and soft classification rules as special cases, we will study its properties from two perspectives: the first one is its classification consistency, and the second one is to investigate whether and when LUMs can estimate the class conditional probability. As we will show, the unique framework of LUMs enables us to see how soft classifiers gradually become hard classifiers as the loss function changes.

Fisher consistency for a binary classification procedure based on a loss function can be defined to be that the population minimizer of the loss function has the same sign as  $p(\mathbf{x}) - 1/2$  (Lin, 2004). In our context, Fisher consistency requires that the minimizer of  $E[V(Yf(\mathbf{X}))|\mathbf{X} = \mathbf{x}]$  has the same sign as  $p(\mathbf{x}) - 1/2$ . Fisher consistency is also known as classification-calibrated (Bartlett et al., 2006) and is a desirable property for a loss function. The following proposition implies Fisher consistency of LUMs.

**Proposition 1.** *Denote  $p(\mathbf{x}) = P(Y = 1|\mathbf{X} = \mathbf{x})$ . For a given  $\mathbf{X} = \mathbf{x}$ , the theoretical*

minimizer  $f^*(\mathbf{x})$  of  $E[V(Yf(\mathbf{X}))]$  is as follows:

$$f^*(\mathbf{x}) = \begin{cases} -\frac{1}{1+c}(R(\mathbf{x})^{-1}a - a + c) & \text{if } 0 \leq p(\mathbf{x}) < 1/2 \\ [-\frac{c}{1+c}, \frac{c}{1+c}] & \text{if } p(\mathbf{x}) = 1/2 \\ \frac{1}{1+c}(R(\mathbf{x})a - a + c) & \text{if } 1/2 < p(\mathbf{x}) \leq 1, \end{cases} \quad (6)$$

where  $R(\mathbf{x}) = [p(\mathbf{x})/(1 - p(\mathbf{x}))]^{1/(a+1)}$ .

The proof follows from the first derivative of the loss function and the details are included in the Appendix. One immediate conclusion we can draw from Proposition 1 is that the class of LUM loss functions is Fisher consistent.

As a remark, for  $c \rightarrow \infty$ , the LUM loss  $V(yf(\mathbf{x}))$  reduces to the hinge loss for the SVM with the corresponding minimizer as follows:

$$f^*(\mathbf{x}) = \begin{cases} -1 & \text{if } 0 \leq p(\mathbf{x}) < 1/2 \\ (-1, 1) & \text{if } p(\mathbf{x}) = 1/2 \\ 1 & \text{if } 1/2 < p(\mathbf{x}) \leq 1. \end{cases}$$

As a result, the hinge loss is Fisher consistent for binary classification as previously noted by Lin (2002). Moreover, we emphasize that the SVM only estimates the classification boundary  $\{\mathbf{x} : p(\mathbf{x}) = 1/2\}$  without estimating  $p(\mathbf{x})$  itself (Wang et al., 2008). Thus, the LUM family becomes hard classifiers as  $c \rightarrow \infty$ .

For  $a \rightarrow \infty$ , the loss  $V(yf(\mathbf{x}))$  reduces to the hybrid loss of the hinge loss and the exponential loss, as shown in Section 2.3. Its corresponding minimizer is as follows:

$$f^*(\mathbf{x}) = \begin{cases} \frac{1}{1+c}[\ln(p(\mathbf{x})/(1 - p(\mathbf{x}))) - c] & \text{if } 0 \leq p(\mathbf{x}) < 1/2 \\ [-\frac{c}{1+c}, \frac{c}{1+c}] & \text{if } p(\mathbf{x}) = 1/2 \\ \frac{1}{1+c}[\ln(p(\mathbf{x})/(1 - p(\mathbf{x}))) + c] & \text{if } 1/2 < p(\mathbf{x}) \leq 1. \end{cases} \quad (7)$$

From the minimizer  $f^*(\mathbf{x})$  of  $V(yf(\mathbf{x}))$  in (6) with any finite  $c \geq 0$ , we can convert the corresponding probabilities  $p(\mathbf{x})$  as follows:

$$p(\mathbf{x}) = \begin{cases} [1 + [\frac{1}{a}\{-(1+c)f^*(\mathbf{x}) + a - c\}]^{a+1}]^{-1} & \text{if } f^*(\mathbf{x}) < -\frac{c}{1+c} \\ \frac{1}{2} & \text{if } f^*(\mathbf{x}) \in [-\frac{c}{1+c}, \frac{c}{1+c}] \\ 1 - [1 + [\frac{1}{a}\{(1+c)f^*(\mathbf{x}) + a - c\}]^{a+1}]^{-1} & \text{if } f^*(\mathbf{x}) > \frac{c}{1+c}. \end{cases} \quad (8)$$

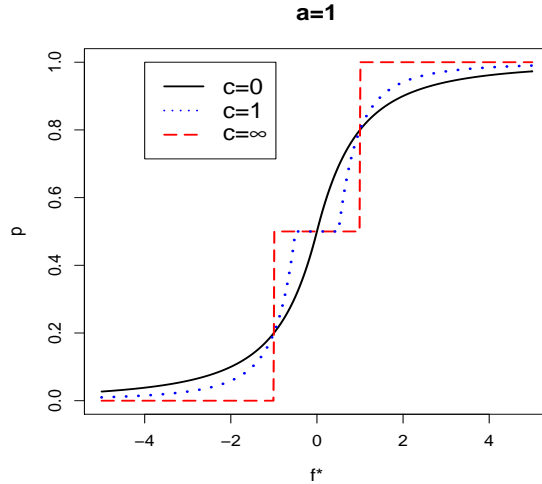


Figure 3: Plot of the correspondence between  $f^*(\mathbf{x})$  and  $p(\mathbf{x})$  as given in (8) with  $a = 1$  and  $c = 0, 1$ , and  $\infty$ .

In order to estimate  $p(\mathbf{x})$ ,  $c$  needs to be finite. Interestingly, when  $p(\mathbf{x}) = 1/2$ ,  $p(\mathbf{x})$  and  $f^*(\mathbf{x})$  do not have a one-to-one correspondence for any  $c > 0$ . In fact, 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  $f^*(\mathbf{x})$  and  $p(\mathbf{x})$  with  $a = 1$  and  $c = 0, 1$ , and  $\infty$ . When  $c > 0$ , the flat region of  $p(\mathbf{x})$  versus  $f^*(\mathbf{x})$  as shown in Figure 3 makes the estimation of  $p(\mathbf{x})$  more difficult. Thus, in terms of class conditional probability estimation, LUM loss functions with  $c = 0$  will be the best case since we have a one-to-one correspondence between  $p(\mathbf{x})$  and  $f^*(\mathbf{x})$  for  $\forall p(\mathbf{x}) \in [0, 1]$ . Another important point we want to make is that although as  $c$  increases, the class probability estimation may become more difficult, LUMs can still provide class probability estimation, even for very large values of  $c$ . The only exceptional case is  $c \rightarrow \infty$ , then the method reduces to the standard SVM whose minimizer cannot estimate  $p(\mathbf{x})$  as mentioned before. Therefore, unlike the SVM, LUMs can provide class probability estimation for any finite  $c$ , although the estimation deteriorates as  $c$  increases.

As a remark, we note that the relationship between  $f^*(\mathbf{x})$  and  $p(\mathbf{x})$  given in (6) and (8) can be cast as a link function used in generalized linear models (GLM)

(McCullagh and Nelder, 1989), although LUMs and GLM were originated from very different views. For example, the logistic regression uses the logit link function  $\log(p(\mathbf{x})/(1-p(\mathbf{x}))) = f(\mathbf{x})$  with  $p(\mathbf{x}) = \exp(f(\mathbf{x})) / (1 + \exp(f(\mathbf{x})))$ . Interestingly, this link function matches the form of the minimizer of our hybrid loss given in (7) with  $c = 0$ . This observation further confirms the soft classification behavior of the LUM family when  $c = 0$ .

Although the relationship between  $f^*(\mathbf{x})$  and  $p(\mathbf{x})$  can help us to obtain estimation of  $p(\mathbf{x})$ , calculation of  $f(\mathbf{x})$  for an accurate classification boundary and estimation of  $p(\mathbf{x})$  have different goals. Even if  $f(\mathbf{x})$  can yield an accurate classification boundary,  $f(\mathbf{x})$  itself may not be accurate enough for the estimation of  $p(\mathbf{x})$  since the classification boundary only depends on  $\text{sign}(f(\mathbf{x}))$ . This problem can be more severe if there is a lot of shrinkage on  $f(\mathbf{x})$  by the regularization term. To improve the probability estimation accuracy, we consider a refitted LUM by calculating a refined estimate associated with two parameters  $(\gamma_0, \gamma_1)$  as  $f^{(2)}(\mathbf{x}) = \gamma_0 + \gamma_1 f^{(1)}(\mathbf{x})$ , where  $f^{(1)}(\mathbf{x})$  is the solution by the original LUM regularization problem (1). In particular, with  $f^{(1)}$  given, the refitted LUM solves an unregularized problem, with the same  $a$  as for  $f^{(1)}(\mathbf{x})$  and  $c = 0$  in the LUM loss, as follows

$$\min_{\gamma_0, \gamma_1} \sum_{i=1}^n V(y_i(\gamma_0 + \gamma_1 f^{(1)}(\mathbf{x}_i))). \quad (9)$$

The idea is that  $f^{(2)}(\mathbf{x})$  serves as a scaled-corrected version of  $f^{(1)}(\mathbf{x})$  for probability estimation. This can be very effective if  $f^{(1)}(\mathbf{x})$  gives accurate classification boundary. Once  $f^{(2)}(\mathbf{x})$  is obtained, we can use it to estimate  $p(\mathbf{x})$  via (8), with the same  $a$  as for  $f^{(1)}(\mathbf{x})$  and  $c = 0$ . Here we use  $c = 0$  in the refitting step since the corresponding formula has a one-to-one correspondence between  $p(\mathbf{x})$  and  $f^*(\mathbf{x})$ . As shown in our simulated examples in Section 5, probability estimation using the refitted LUM often leads to great improvement over that of the original LUM solution.

## 4 Computational Algorithm

The LUM loss  $V(u)$  is convex and first-order differentiable, but it is not second-order differentiable. One can use various convex optimization tools to fit the LUMs, such as nonlinear optimization functions in R and Matlab and specialized licensed optimization solvers such as MOSEK and MINOS. All of these work well when the dimensionality of data is not too large. Next, we propose a simple alternative algorithm for the LUMs, the coordinate gradient descent algorithm, which works efficiently for both low and high dimensional data.

Now we present the coordinate descent algorithm to minimize the objective function of the LUMs

$$\min_f \lambda J(f) + \frac{1}{n} \sum_{i=1}^n V(y_i(f(\mathbf{x}_i))), \quad (10)$$

where  $J(f)$  is the regularization function of the function  $f(\cdot)$ . We focus on linear learning with  $f(\mathbf{x}) = \mathbf{x}'\mathbf{w} + b$  and  $J(f) = \frac{1}{2}\mathbf{w}'\mathbf{w}$ . The extension to kernel learning is straightforward. Denote the solution at the  $m$ -th step by  $\hat{\mathbf{w}}^{(m)} = (\hat{w}_1^{(m)}, \hat{w}_2^{(m)}, \dots, \hat{w}_d^{(m)})^T$  and  $\hat{b}^{(m)}$ . At the  $(m+1)$ -th step, for  $j = 1, 2, \dots, d$ , define  $\tilde{f}_{ij}^{(m+1)} = \sum_{k < j} x_{ik} \hat{w}_k^{(m+1)} + \sum_{k' > j} x_{ik'} \hat{w}_{k'}^{(m)} + \hat{b}^{(m)}$  and set

$$\hat{w}_j^{(m+1)} = \operatorname{argmin}_{w_j} \frac{\lambda}{2} w_j^2 + \frac{1}{n} \sum_{i=1}^n V(y_i(\tilde{f}_{ij}^{(m+1)} + x_{ij} w_j)). \quad (11)$$

Problem (11) involves a one-dimensional optimization and can be solved using many routine optimization techniques such as the Newton-Raphson method. Once finishing updating  $\hat{w}_j^{(m+1)}$  for  $j = 1, 2, \dots, d$ , we update the intercept by setting

$$\hat{b}^{(m+1)} = \operatorname{argmin}_b \sum_{i=1}^n V(y_i(b + \sum_{j=1}^d x_{ij} \hat{w}_j^{(m+1)})). \quad (12)$$

We keep iterating until convergence. The convergence criterion we used is to require  $\|\hat{\mathbf{w}}^{(m)} - \hat{\mathbf{w}}^{(m+1)}\|^2 + (b^{(m)} - b^{(m+1)})^2$  to be sufficiently small.

## 5 Simulation

In this section, we use simulated examples to demonstrate the numerical performance of LUMs in various scenarios. One main advantage of the LUM is that it provides a convenient platform to fit a spectrum of large-margin classifiers, from soft to hard, using one common algorithm. This has greatly facilitated a systematic exploration and comparison of different types of classifiers. Moreover, the LUM produces the class probabilities as a by-product for each combination of  $a$  and  $c$ .

We consider five examples. The first example illustrates a situation where soft classifiers outperform hard classifiers. The second example shows a case that hard classifiers outperform soft classifiers. The third example demonstrates the performance transition of hard and soft classifiers with varying signal strength. The fourth and fifth examples demonstrate how the performance of LUMs changes when the data dimensionality varies. We fit the LUM classifiers with various choices of  $a$  and  $c$ . For each method, we generate a training set of size 100 to fit the classifier, a validation set of size 100 to tune the regularization parameters. For each pair  $(a, c)$ , the corresponding tuning parameter  $\lambda$  is selected by minimizing the classification error over the validation set using a grid search.

For the purpose of classification, we fit the LUMs with different combinations of  $(a, c)$  and report the probability estimation and classification results associated with each pair  $(a, c)$ . In practice, a systematic tuning procedure is often desired to select the optimal  $(a, c)$ . Since two-dimensional tuning can be time-consuming, we suggest a fast and effective tuning procedure for the LUMs. Our empirical experience with the LUMs (shown later in five examples) as well as the loss shapes as shown in Figure 1 suggest that the classification performance of the LUMs is not quite sensitive to the choice of  $a$ , while  $c$  plays a more important role. The magnitude of  $c$  determines whether the classifier is soft or hard. For example,  $c = 0$  yields a soft classifier and  $c = 10,000$  gives a hard classifier. In light of this, we suggest to choose  $c$  from two extreme values  $\{0; 10,000\}$  while fixing  $a$  at a constant value such as 1,000. We call

the resulting classifier the “tuned LUM”, which is computationally more efficient than the exhaustive search over  $(a, c)$ -space. In some sense, the “tuned LUM” tries to make a greedy choice between a pair of hard and soft classifiers, the two ends of the LUM family. Our numerical results show that the tuned LUM in general performs nearly as well as the LUM with the best  $(a, c)$ . For the purpose of probability estimation, we compare the probability estimation errors of the LUMs and the refitted LUMs described in Section 3.

For each classifier, we evaluate its performance in both classification accuracy and probability estimation, respectively using the classification error of the estimated classifier and the mean absolute error (MAE)  $E_{\mathbf{X}}|p(\mathbf{X}) - \hat{p}(\mathbf{X})|$  of the estimated conditional probability on the test set of size 100,000. For further comparison, we also report the results of the PLR. We conduct 1,000 replications for each method in all examples, and report the average test errors, MAEs, and the associated Monte Carlo standard errors (SEs).

## 5.1 Nonlinear Example via Basis Expansion: Soft Better

In this example, Class +1 is generated from a mixture of two bivariate Normal distributions, and Class -1 is from one bivariate Normal distribution as follows

$$\begin{aligned}\mathbf{X}|Y = 1 &\sim 0.5N_2((0, 1)', 0.5I_2) + 0.5N_2((4, 1)', 0.5I_2), \\ \mathbf{X}|Y = -1 &\sim N_2((2, 0)', I_2).\end{aligned}$$

For this example, the corresponding Bayes error is 0.1349. The Bayes rule is nonlinear and is illustrated on the left panel of Figure 4. To capture the nonlinear boundary, we use a polynomial basis to fit the LUM. In particular, we map the input  $\{x_1, x_2\}$  to  $\{x_1, x_2, x_1^2, x_2^2, x_1x_2\}$  and fit the linear LUM with the expanded basis.

There are two parameters  $a$  and  $c$  in the LUM function, and each combination of them will result in a different classifier. As we pointed out before,  $a$  and  $c$  serve different roles in the LUM family of loss functions:  $c$  determines whether the corresponding

classifier is soft or hard, whereas  $a$  controls the decaying speed of the loss function. We explore the effect of  $a$  and  $c$ , via calculating the classifiers corresponding to various values of these parameters with  $a = 1, 5, 10, 100, 1000$  and  $c = 0, 1, 5, 10, 100, 1000, 10000$ . As  $c \rightarrow \infty$ , the LUM classifier approaches to the SVM. Since the solutions barely change after  $c \geq 1000$ , the LUM classifier with  $c \geq 1000$  can be viewed as a good approximation of the SVM classifier. Our limited numerical experience confirms that the LUM solution with large  $c$  and the SVM solution are almost identical.

The right panel of Figure 4 summarizes the average test classification errors over 1000 repetitions for different values of  $a$  and  $c$ . It is seen that LUMs with  $c = 0$  and 1 give the smallest test errors in all settings, suggesting that soft classification outperforms hard classification within the LUM family for this example. The effects of  $a$  appear to be very small. On the right panel of Figure 4, we also report the performance of the tuned LUM, denoted by “Tuned”, for each  $a$ . More explicitly the “tuned” LUM for each  $a$  is defined as follows. For each repetition and a fixed  $a$ , we apply another layer of tuning by comparing the best tuning errors over the validation set for  $c = 0$  and  $c = 10,000$  and the corresponding test error is given by the test error for the LUM with  $c = 0$  or  $c = 10,000$  depending on which one gives a smaller tuning error. The performance of the tuned LUM method is close to the best among all LUMs with various  $a$  and  $c$ . For comparison, we also calculated the classification accuracy of the PLR. The corresponding test error (SE) are 0.1563 (0.0005).

On the left panel of Figure 4, we depict the classification boundaries given by the Bayes rule and various LUM classifiers with  $c = 0, 1, 10, 1000$ , for one simulated dataset of size 200. As we can see from the plot, all LUMs are reasonably close to the Bayes boundary, and the one corresponding to  $c = 0$  is the closest one to the Bayes rule.

Table 1 summarizes the MAEs for the estimated class probabilities over the 1000 repetitions. The LUM with  $c = 0$  gives the smallest MAE among all the LUM estimates, which is consistent to their theoretical properties studied previously. The differences in MAEs between soft and hard LUMs are quite significant, in view of the SEs.

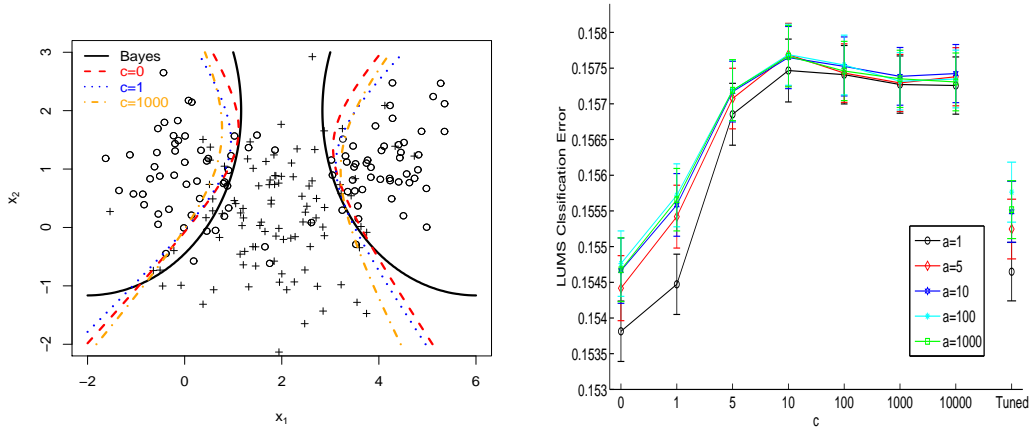


Figure 4: Left panel: Plot of LUM classification boundaries with  $a = 1$  and  $c = 0, 1, 1000$  for Example 5.1. Right panel: Classification errors  $\pm 1$  SE of LUMs.

Table 1: Average MAEs for probability estimation in Example 5.1. The corresponding SEs for the average LUM MAEs and average refitted LUM MAEs range from 0.0008 to 0.0014 and from 0.0006 to 0.0007, respectively.

c	a				
	1	5	10	100	1000
	LUM MAEs				
0	0.0948	0.0955	0.0971	0.0983	0.0984
1	0.0982	0.1027	0.1035	0.1045	0.1044
5	0.1117	0.1145	0.1155	0.1164	0.1165
10	0.1185	0.1216	0.1219	0.1227	0.1226
100	0.1276	0.1283	0.1286	0.1288	0.1287
1000	0.1289	0.1290	0.1292	0.1290	0.1291
10000	0.1291	0.1292	0.1293	0.1290	0.1294
	Refitted LUM MAEs				
0	0.0791	0.0782	0.0788	0.0795	0.0795
1	0.0780	0.0778	0.0783	0.0792	0.0792
5	0.0803	0.0796	0.0801	0.0807	0.0808
10	0.0811	0.0805	0.0808	0.0815	0.0816
100	0.0807	0.0800	0.0806	0.0813	0.0813
1000	0.0807	0.0799	0.0804	0.0810	0.0812
10000	0.0807	0.0801	0.0804	0.0811	0.0812

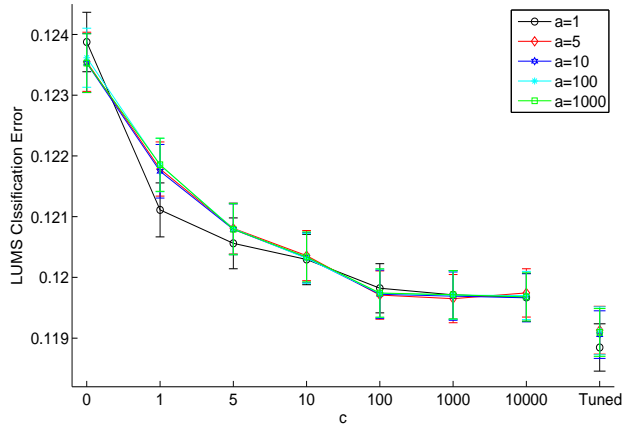


Figure 5: Classification results of LUMs for Example 5.2 with various  $a$  and  $c$ .

Interestingly, the refitted LUMs largely improve probability estimation. Furthermore, the difference between refitted soft and hard LUMs in terms of MAEs becomes much smaller than that of the original LUMs. The corresponding MAE (SE) for the PLR are 0.0957 (0.0012). Overall, we can conclude that LUMs yield competitive performance in terms of both classification accuracy and probability estimation.

## 5.2 Random Flipping Example: Hard better

In this 2-dimensional example, covariates  $X_1$  and  $X_2$  are independently generated from  $\text{Uniform}[-1, 1]$ . Conditional on  $X_1$  and  $X_2$ , the class output is generated as follows:  $Y = \text{sign}(X_1 + X_2)$  when  $|X_1 + X_2| \geq 0.5$ ; when  $|X_1 + X_2| < 0.5$ ,  $Y = \text{sign}(X_1 + X_2)$  with probability 0.8 and  $-\text{sign}(X_1 + X_2)$  with probability 0.2.

The underlying conditional class probability function of this example is a step function. Thus, in this case, class probabilities are difficult to estimate and the classification accuracy of soft classifiers may be sacrificed by probability estimation. We expect that hard classifiers, bypassing probability estimation, may work better than soft classifiers. The classification accuracy results are shown in Figure 5.

Indeed, hard classifiers work much better in this example in terms of classification

Table 2: Probability MAEs of Example 5.2 given by LUMs. The corresponding SEs for the average LUM MAEs and average refitted LUM MAEs range from 0.0027 to 0.0033 and from 0.0003 to 0.0008, respectively.

c	a				
	1	5	10	100	1000
	LUMS MAE				
0	0.2008	0.2030	0.2037	0.2055	0.2049
1	0.2068	0.2137	0.2142	0.2150	0.2153
5	0.2170	0.2200	0.2202	0.2212	0.2215
10	0.2190	0.2221	0.2218	0.2230	0.2230
100	0.2176	0.2177	0.2178	0.2183	0.2183
1000	0.2172	0.2170	0.2173	0.2178	0.2178
10000	0.2173	0.2174	0.2171	0.2175	0.2170
Best	0.2061	0.2104	0.2107	0.2115	0.2114
	Refitted LUMS MAE				
0	0.0814	0.0760	0.0753	0.0745	0.0747
1	0.0782	0.0746	0.0740	0.0736	0.0736
5	0.0782	0.0746	0.0741	0.0737	0.0736
10	0.0782	0.0746	0.0741	0.0736	0.0736
100	0.0783	0.0746	0.0740	0.0736	0.0736
1000	0.0782	0.0746	0.0741	0.0736	0.0736
10000	0.0782	0.0746	0.0740	0.0736	0.0736

accuracy. The tuned LUM works extremely well here. The test error (SE) of the PLR are 0.1242 (0.0005). Its performance is similar to that of the LUM with small  $c$ . Similar to Example 5.1, the classification performance is not very sensitive to the choice of  $a$ .

The probability estimation results of LUMs are summarized in terms of MAEs in Table 2. Since soft classifiers do not work well for this example, we do not expect them to produce accurate probability estimation. Indeed, as shown in Table 2, probability estimation of the original LUMs is not satisfactory. The refitted LUMs produce dramatic improvement over the original LUMs. Interestingly, the refitted LUMs with large  $c$ 's work even slightly better than those with smaller  $c$ 's. This is consistent with the observation by Wang et al. (2008) that better classification can be translated into better probability estimation. In this case, the original LUMs provide accurate classification boundaries and then the refitted LUMs correct the scales of the classification function by the original LUMs for better probability estimation. In this case, the MAE (SE) of the PLR are 0.1823 (0.003), much worse than our refitted LUMs.

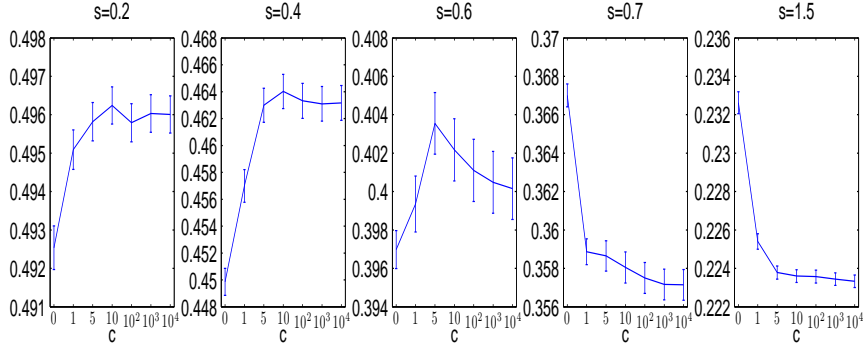


Figure 6: The classification performance of LUMs for Example 5.3 with varying signal levels.

### 5.3 Varying Signal Example: Transition between Hard and Soft Classifiers

In this two dimensional example, predictors  $X_1$  and  $X_2$  are uniformly distributed over  $\{(x_1, x_2) : |x_1| + |x_2| \leq 2\}$ . Conditional on  $X_1 = x_1$  and  $X_2 = x_2$ ,  $Y$  takes 1 with probability  $e^{s(x_1+x_2)}/(1+e^{s(x_1+x_2)})$  and  $-1$  with probability  $1/(1+e^{s(x_1+x_2)})$  for some  $s > 0$ . This is essentially a Binomial example with  $s$  controlling the signal strength level. As the previous examples have demonstrated that the effect of  $a$  on the classification error is very minimal, we fix  $a = 1,000$  in this example. The goal of the current example is to explore the performance pattern for different  $c$  as the signal level varies. Figure 6 plots the classification errors for different  $c$  and different signal levels.

As shown in Figure 6, LUMs with small  $c$ 's, i.e. soft classifiers, work better for the case of weak signals with  $s = 0.2$  and  $0.6$ . As we increase the signal levels, all classifiers give more accurate classification results. However, hard classifiers become more and more competitive. When  $s = 0.6$ , LUMs with small and large  $c$ 's outperform those with middle values of  $c$ . Once the signal level is relatively high with  $s = 0.7$  and  $1.5$ , hard classifiers, i.e. LUMs with large  $c$ 's, outperform soft classifiers. This example implies that which type of classifiers performs better is related to the signal strength contained in data. In this particular setup, soft classifiers tend to work better when

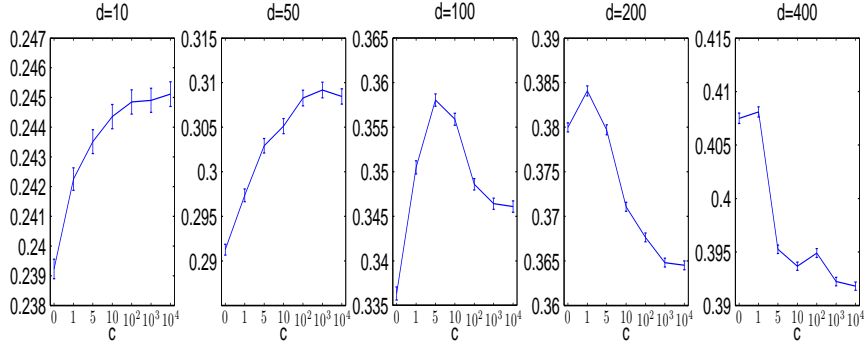


Figure 7: The classification performance of LUMs for Example 5.4 with increasing dimensions.

the signal is weak, and hard classifiers seem to work better in the case of strong signals.

## 5.4 Increasing Dimension Example with Nonsparse Signal: From Soft Better to Hard Better

For this example, the data are generated from a Gaussian mixture distribution: positive samples are from  $N(\boldsymbol{\mu}_d, I_d)$  and negative samples are from  $N(-\boldsymbol{\mu}_d, I_d)$  with  $\boldsymbol{\mu}_d = (\mu, \mu, \dots, \mu)^T$ . The dimension  $d$  changes from 10, 50, 100, 200, to 400. Here  $\mu$  controls the signal level. We choose  $\mu$  as a function of  $d$  such that the Bayes error is fixed at 22% for different dimensions.

The previous examples have demonstrated that the effect of  $a$  on the classification error is very minimal, so we fix  $a = 1,000$  in this example. We plot the classification errors in Figure 7 for different dimensionality  $d$ . As we can see from Figure 7, when the dimension  $d = 10$  and 50, LUMs with smaller  $c$ 's, i.e. soft classifiers, yield the best performance. As the dimension increases to  $d = 100$ , LUMs with large  $c$ 's become relatively more competitive. When the dimension equals to 400, LUMs with large  $c$ 's work the best. This indicates that hard classifiers tend to work better in high dimensional classification examples.

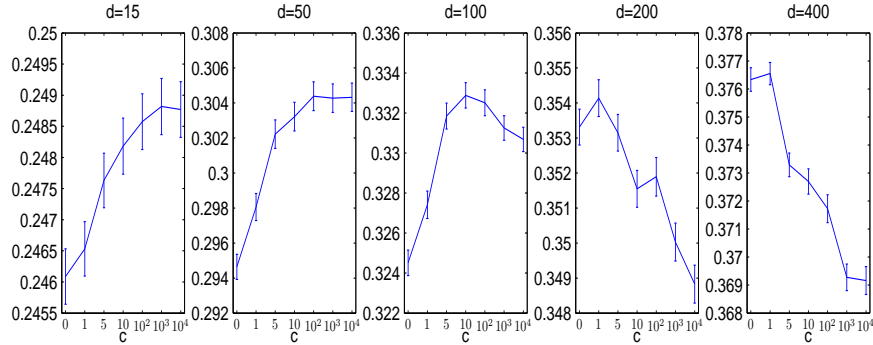


Figure 8: The classification performance of LUMs for Example 5.5 with increasing dimensions.

## 5.5 Increasing Dimension Example with Sparse Signal: From Soft Better to Hard Better

We generate the data as a mixture of two Gaussians in a 15-dimensional space. In particular, 40% of data are from  $N(\boldsymbol{\mu}_1, I_d)$  with  $Y = 1$  and 60% from  $N(\boldsymbol{\mu}_{-1}, I_d)$  with  $Y = -1$ , where  $I_d$  is the identity matrix of size  $d$ , the elements of  $\boldsymbol{\mu}_1$  and  $\boldsymbol{\mu}_{-1}$  are zeros except the first 10 nonzero elements being  $(0.7, 0.2, -0.7, 0.6, 0.3, 0.5, 0.5, -0.6, -0.6, 0.3)$  and  $(0.2, 0.7, -0.7, 0.1, 0.8, -0.2, 1.3, -0.2, -0.8, 0.6)$  respectively. The Bayes decision boundary is linear for this classification problem and the Bayes error is 0.210. Note that since only the first ten dimensions are useful for classification, the problem becomes more challenging as  $d$  increases as more noise variables are included in the classification task. We consider the cases of  $d = 15, 50, 100, 200, 400$ .

Figure 8 summarizes the average test classification errors over 1000 repetitions for different values  $c$  with  $a = 1000$ . The behavior of LUMs with varying  $c$  is similar to Example 5.4. For relatively low dimension cases with  $d = 15, 50$ , LUMs with smaller  $c$ 's are significantly better than those with large  $c$ 's. Thus, in that case, soft classification outperforms hard classification. As the dimension increases such as  $d = 100$ , LUMs with large  $c$ 's start to be more competitive. As  $d$  increases to 200 and 400, LUMs with large  $c$ 's are significantly better than LUMs with small  $c$ 's. Thus, hard classification

outperforms soft classification then. Overall, for this example, similar to Example 5.4, we can conclude that hard classification works better for high dimensional cases.

## 5.6 Summary of Simulated Examples

The above five simulated examples give us some insight on the performance of LUMs as well as the issue of soft versus hard classifiers. There are two parameters,  $a$  and  $c$ , for the LUM family. Based on our numerical experience, LUMs are not sensitive to the choice of  $a$ . Thus, we recommend to fix  $a$ , such as  $a = 1,000$ . The choice of  $c$  is more critical. Smaller  $c$ 's correspond to soft classifiers and large  $c$ 's correspond to hard classifiers.

Example 5.1 has relatively smooth underlying conditional probability functions. In this case, it is advantageous to perform classification and probability estimation simultaneously. In fact, the probability information can help to build a more accurate classifier. As a result, LUMs with small  $c$ 's such as  $c = 0$  work the best for these two examples.

The underlying conditional probability function for Example 5.2 is a step function and thus not a smooth function. In this case, probability estimation can be a more challenging task than classification itself. Hard classifiers tend to focus on the classification boundary itself and avoid the task of probability estimation. Indeed, LUMs with large  $c$ 's yield more accurate classifiers.

Example 5.3 illustrates the behaviors of LUMs with varying signal levels. In particular, it shows that LUMs with smaller (larger)  $c$ 's work better for cases of relatively weaker (stronger) signals than those of larger (smaller)  $c$ 's. This indicates that hard classifiers tend to work better for problems with stronger signals. Intuitively, this makes sense since hard classifiers such as the SVM tend to use data points close to the boundary to produce the classification boundary. When the signal is strong and the problem is close to separable, hard classifiers are more likely to work better than soft classifiers. In problems with relatively weak signals, soft classifiers may work better

since they make use of all data points. Points far from the boundary can still provide useful information.

Examples 5.4 and 5.5 show how dimensionality of the classification problem affects the choice between soft and hard classifiers. The common belief in statistical learning is that hard classifiers can be a good candidate for high dimensional problems. One potential reason is that data tend to be more separable in high dimensional spaces, so hard classifiers may yield classifiers with better generalization ability. Indeed, we can learn from Examples 5.4 and 5.5 that LUMs with large  $c$ 's become more competitive as the dimension increases.

Our numerical results suggest that our tuned LUMs between large and small  $c$ 's with a fixed  $a$  can yield near optimal classification accuracy. For problems that we are uncertain about the choice between soft and hard classifiers, we recommend to use the tuned LUM.

In terms of probability estimation, LUMs with smaller  $c$ 's yield much more accurate probability estimation than those of larger  $c$ 's. The more interesting lesson is that the step of refitting corrects the scale of the classification function and significantly improves the performance of probability estimation. It can be used as an attractive method of probability estimation for both hard and soft classifiers.

## 6 Real Example

We apply the proposed LUMs to two real data examples. The first example is the liver disorder dataset from the UCI benchmark repository. The dataset can be downloaded from <http://www.ics.uci.edu/~mlern/MLRepository.html>. This dataset contains 345 observations and 7 input features. The first 5 features are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. The other two features are related to the measure of alcoholic consumption per day. The goal of the problem is to use the blood test and alcoholic consumption

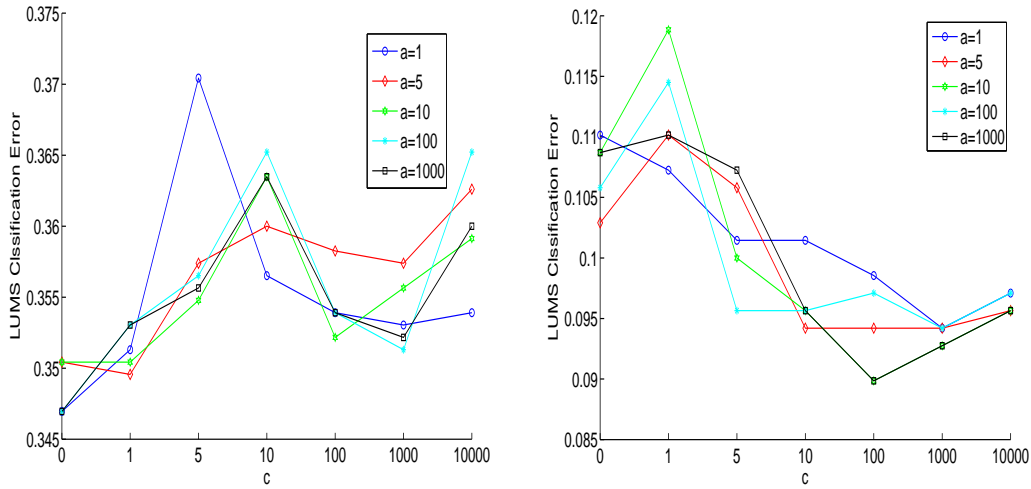


Figure 9: Plots of classification errors by LUMs with various  $a$  and  $c$ . The liver disorder example and the lung cancer example are displayed on the left and right panels respectively.

information to classify the status of liver disorder. The second example is the lung cancer dataset, described in Liu et al. (2008). The dataset contains 205 subjects including 128 adenocarcinoma, 20 carcinoid, 21 squamous, 17 normal tissues, 13 colon cancer metastasis, and 6 small cell carcinoma samples. Since the adenocarcinoma class is the most heterogeneous class, we consider the binary classification task of adenocarcinoma versus others. We select 200 genes with the most variation for classification.

Since there are no separate validation and testing data available for these data sets, we randomly divide each data set into three parts: one part used as the training set, one part for tuning, and the third part used as the testing set. We repeat this process ten times and report the average test errors for linear LUMs in Figure 9.

The liver disorder example is shown to be a relatively difficult classification problem with test errors over 30%. In view of the lesson we learn from Example 5.3, soft classification may work better in this case. The left panel of Figure 9 shows the test errors for various  $a$  and  $c$ . Again the effect of  $a$  is small, but it appears that smaller  $c$  works better, suggesting soft classification is more preferable here. We also calculated

the tuned LUMs. For  $a = 1,000$ , the corresponding test error is 0.3470, which is near optimal compared to other errors in the plot.

The lung cancer example is a high dimensional example and the right panel of Figure 9 suggests that hard classification works better here. This is consistent with the findings in Examples 5.4 and 5.5. In this case, the test error for the tuned LUM with  $a = 1,000$  is 0.0957. Again this is near optimal in view of the test errors corresponding to various  $(a, c)$  reported in right panel of Figure 9.

## 7 Discussion

Large-margin classifiers play an important role in classification problems. In this paper, we propose a rich family called the LUM which covers a wide range of classifiers from soft to hard ones. The new family provides a natural bridge between soft and hard classification. Properties of this new family helps to provide further insight on large-margin classifiers. Furthermore, the LUM family includes some existing classifiers such as the SVM and DWD as well as many new ones including the hybrid of the SVM and Boosting.

The LUM family is indexed by two parameters  $a$  and  $c$ . Our numerical study shows that the performance of LUMs is not very sensitive to the choice of  $a$ . Thus, we recommend to fix  $a$  at one value such as 1,000. The role of  $c$  is more critical for the resulting classifier since it connects soft and hard classifiers. Our numerical examples shed some light on the choice between soft and hard classifiers. Furthermore, our tuned LUM, with fixed  $a$  and tuned  $c$  in  $\{0; 10,000\}$ , appears to give near-optimal performance.

Our study on probability estimation shows that the step of refitting is critical. Moreover, our probability estimation scheme works well for both hard and soft classifiers in the LUM family. As a result, it can also be used as a probability estimation technique for hard classifiers such as the SVM.

# Appendix

## Proof of Theorem 1:

To simplify the formulation (3), we remove  $\gamma$  and get the following equivalent formulation of (3)

$$\begin{aligned} & \min_{\mathbf{w}, b, \boldsymbol{\xi}} \sum_i 1/(y_i f(\mathbf{x}_i) + \xi_i) + C \sum_{i=1}^n \xi_i & (13) \\ \text{subject to} & \quad \mathbf{w}' \mathbf{w} \leq 1, \xi_i \geq 0, \forall i. \end{aligned}$$

To get the minimizer of (13), for fixed  $y_i f(\mathbf{x}_i)$ , we first study the minimizer of

$$A(\xi_i) \equiv \frac{1}{y_i f(\mathbf{x}_i) + \xi_i} + C \xi_i$$

with  $\xi_i \geq 0$ . Note that

$$A'(\xi_i) = -\frac{1}{(y_i f(\mathbf{x}_i) + \xi_i)^2} + C.$$

- When  $y_i f(\mathbf{x}_i) \leq 1/\sqrt{C}$ ,  $A'(\xi_i) < 0$  if  $\xi_i < 1/\sqrt{C} - y_i f(\mathbf{x}_i)$ ,  $A'(\xi_i) = 0$  if  $\xi_i = 1/\sqrt{C} - y_i f(\mathbf{x}_i)$ , and  $A'(\xi_i) > 0$  if  $\xi_i > 1/\sqrt{C} - y_i f(\mathbf{x}_i)$ . Then the minimizer  $\xi_i^* = 1/\sqrt{C} - y_i f(\mathbf{x}_i)$ .
- When  $y_i f(\mathbf{x}_i) > 1/\sqrt{C}$ ,  $A'(\xi_i) > 0$  for  $\forall \xi_i \geq 0$ . Then  $\xi_i^* = 0$ .

Then one can show that the optimization problem (13) is equivalent to the following problem

$$\min_{\mathbf{w}, b} V_{DWD}^C(y_i f(\mathbf{x}_i)) \text{ subject to } \mathbf{w}' \mathbf{w} \leq 1, \quad (14)$$

where  $V_{DWD}^C(u) = 2\sqrt{C} - Cu$  if  $u \leq 1/\sqrt{C}$  and  $1/u$  otherwise.

Let  $\tilde{f} = f\sqrt{C}$ . Then we have  $V_{DWD}^C(y_i f(\mathbf{x}_i)) = \sqrt{C} V_{DWD}^1(y_i \tilde{f}(\mathbf{x}_i))$ , where  $V_{DWD}^1(u) = 2 - u$  if  $u \leq 1$  and  $1/u$  otherwise. Then the optimization problem (14) is equivalent to

$$\min_{\tilde{\mathbf{w}}, \tilde{b}} V_{DWD}^1(y_i \tilde{f}(\mathbf{x}_i)) \text{ subject to } \tilde{\mathbf{w}}' \tilde{\mathbf{w}} \leq C, \quad (15)$$

Using Lagrange multiplier  $\lambda > 0$ , (15) can be written in the following equivalent formulation

$$\min_{\tilde{\mathbf{w}}, \tilde{b}} V_{DWD}^1(y_i \tilde{f}(\mathbf{x}_i)) + \frac{\lambda}{2} \tilde{\mathbf{w}}' \tilde{\mathbf{w}}. \quad (16)$$

From the convex optimization theory, we know that there is an one-to-one correspondence between  $C$  in (15) and  $\lambda$  in (16). Denote  $(\hat{\mathbf{w}}, \hat{b})$  with given  $\lambda$  as the solution of (16). Then we have  $\hat{\mathbf{w}}' \hat{\mathbf{w}} = C$ .

Consider  $a = 1$  and  $c = 1$ , then our loss  $V(u) = 1 - u$  if  $u \leq 1/2$  and  $1/(4u)$  otherwise. Let  $u_1 = 2u$ . Then  $V(u) = V_{DWD}^1(2u)/2$  and  $V_{DWD}^1(u_1) = 2V(u_1/2)$ . Plugging the relationship into (16), we get

$$\min_{\mathbf{w}^*, b^*} V(y_i f^*(\mathbf{x}_i)) + \frac{\lambda^*}{2} \mathbf{w}^{*'} \mathbf{w}^*, \quad (17)$$

where  $f^* = \tilde{f}/2$  and  $\lambda^* = 2\lambda$ . Denote  $(\hat{\mathbf{w}}^*, \hat{b}^*)$  with given  $\lambda^*$  as the solution of (17). Then we have  $\hat{\mathbf{w}}^{*'} \hat{\mathbf{w}}^* = \hat{\mathbf{w}}' \hat{\mathbf{w}}/4 = C/4$ . Consequently, we can conclude that DWD is equivalent to LUM with  $a = 1$  and  $c = 1$ .

**Proof of Proposition 1:**

Notice  $E[V(Yf(\mathbf{X}))] = E[E(V(Yf(\mathbf{X}))|\mathbf{X} = \mathbf{x})]$ . We can minimize  $E[V(Yf(\mathbf{X}))]$  by minimizing  $E(V(Yf(\mathbf{X}))|\mathbf{X} = \mathbf{x})$  for every  $\mathbf{x}$ .

For any fixed  $\mathbf{x}$ ,  $E(V(Yf(\mathbf{X}))|\mathbf{X} = \mathbf{x})$  can be written as  $V(f(\mathbf{x}))p(\mathbf{x}) + V(-f(\mathbf{x}))(1 - p(\mathbf{x}))$ . For simplicity of notation, we drop  $\mathbf{x}$  and need to minimize  $g(f) = V(f)p + V(-f)(1 - p)$  with respect to  $f$ . To this end, we take the first order derivative of  $g(f)$  since it is differentiable. Then we have

$$g'(f) = pV'(f) - (1 - p)V'(-f), \quad (18)$$

$$V'(u) = \begin{cases} -1 & \text{if } u < \frac{c}{1+c}, \\ -\left(\frac{a}{(1+c)u - c + a}\right)^{a+1} & \text{if } u \geq \frac{c}{1+c}. \end{cases} \quad (19)$$

Plugging (19) into (18), we get

$$g'(f) = \begin{cases} -p + (1 - p)\left(\frac{a}{(1+c)(-f) - c + a}\right)^{a+1} & \text{if } f < -\frac{c}{1+c}, \\ 1 - 2p & \text{if } -\frac{c}{1+c} \leq f \leq \frac{c}{1+c}, \\ -p\left(\frac{a}{(1+c)f - c + a}\right)^{a+1} + (1 - p) & \text{if } f > \frac{c}{1+c}. \end{cases} \quad (20)$$

To obtain the minimizer of  $g(f)$ , we need to examine the signs of its derivative  $g'(f)$ . When  $p < 1/2$ ,  $g'(f) > 0$  for  $f \geq -\frac{c}{1+c}$  and the minimizer  $f^*$  can be obtained

by setting  $g'(f) = 0$  for  $f < -\frac{c}{1+c}$ . Similarly, when  $p > 1/2$ ,  $g'(f) < 0$  for  $f \leq \frac{c}{1+c}$  and the minimizer  $f^*$  can be obtained by setting  $g'(f) = 0$  for  $f > \frac{c}{1+c}$ . When  $p = 1/2$ ,  $g'(f) < 0$  for  $f < -\frac{c}{1+c}$ ,  $g'(f) > 0$  for  $f > \frac{c}{1+c}$ , and  $g'(f) = 0$  for  $f \in [-c/(1+c), c/(1+c)]$ . The desired result then follows.

## References

- Bartlett, P., Jordan, M., and McAuliffe, J. (2006), “Convexity, classification, and risk bounds,” *Journal of the American Statistical Association*, 101, 138–156.
- Boser, B., Guyon, I., and Vapnik, V. N. (1992), “A training algorithm for optimal margin classifiers,” *The Fifth Annual Conference on Computational Learning Theory, Pittsburgh ACM*, 142–152.
- Cortes, C. and Vapnik, V. N. (1995), “Support-Vector Networks,” *Machine Learning*, 20, 273–279.
- Cristianini, N. and Shawe-Taylor, J. (2000), *An introduction to support vector machines and other kernel-based learning methods*, Cambridge University Press.
- Freund, Y. and Schapire, R. (1997), “A decision theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, 55, 119–139.
- Friedman, J. H., Hastie, T., and Tibshirani, R. (2000), “Additive logistic regression: a statistical view of boosting,” *Annals of Statistics*, 28, 337–407.
- Hastie, T., Tibshirani, R., and Friedman, J. (2001), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer-Verlag: New York.
- Lin, X., Wahba, G., Xiang, D., Gao, F., Klein, R., and Klein, B. (2000), “Smoothing Spline ANOVA Models for Large Data Sets With Bernoulli Observations and the Randomized GACV,” *The Annals of Statistics*, 28, 1570–1600.

- Lin, Y. (2002), “Support Vector Machines and the Bayes Rule in classification,” *Data Mining and Knowledge Discovery*, 6, 259–275.
- (2004), “A note on margin-based loss functions in classification,” *Statistics and Probability Letters*, 68, 73–82.
- Liu, Y., Hayes, D. N., Nobel, A., and Marron, J. S. (2008), “Statistical significance of clustering for high dimension low sample size data,” *Journal of the American Statistical Association*, 103, 1281–1293.
- Liu, Y. and Shen, X. (2006), “Multicategory  $\psi$ -learning,” *Journal of the American Statistical Association*, 101, 500–509.
- Marron, J. S., Todd, M., and Ahn, J. (2007), “Distance weighted discrimination,” *Journal of the American Statistical Association*, 102, 1267–1271.
- McCullagh, P. and Nelder, J. (1989), *Generalized Linear Models*, Chapman & Hall/CRC.
- Qiao, X., Zhang, H. H., Liu, Y., Todd, M. J., and Marron, J. S. (2010), “Weighted distance weighted discrimination and its asymptotic properties,” *Journal of the American Statistical Association*, 105, 401–414.
- Schölkopf, B. and Smola, A. J. (2002), *Learning with kernels*, MIT Press.
- Shen, X., Tseng, G. C., Zhang, X., and Wong, W. H. (2003), “On  $\psi$ -learning,” *Journal of the American Statistical Association*, 98, 724–734.
- Tewari, A. and Bartlett, P. (2005), “On the consistency of multiclass classification methods,” in *Proceedings of the 18th Annual Conference on Learning Theory*, springer, vol. 3559, pp. 143–157.
- Vapnik, V. (1998), *Statistical learning theory*, Wiley.

- Wahba, G. (1998), “Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV,” *In: B. Schölkopf, C. J. C. Burges and A. J. Smola (eds), Advances in Kernel Methods: Support Vector Learning, MIT Press*, 125–143.
- (2002), “Soft and Hard Classification by Reproducing Kernel Hilbert Space Methods,” *In Proceedings of the National Academy of Sciences*, 16524–16530.
- Wang, J., Shen, X., and Liu, Y. (2008), “Probability estimation for large-margin classifiers,” *Biometrika*, 95, 149–167.
- Wu, Y. and Liu, Y. (2007), “Robust truncated-hinge-loss support vector machines,” *Journal of the American Statistical Association*, 102, 974–983.
- Zhang, T. (2004), “Statistical analysis of some multi-category large margin classification methods,” *Journal of Machine Learning Research*, 5, 1225–1251.
- Zhu, J. and Hastie, T. (2005), “Kernel Logistic Regression and the Import Vector Machine,” *Journal of Computational and Graphical Statistics*, 14, 185–205.