Statistics and Its Interface Volume 3 (2010) 465–475

Utility-based weighted multicategory robust support vector machines

Yufeng Liu*, Yichao Wu and Qinying He†

The Support Vector Machine (SVM) has been an important classification technique in both machine learning and statistics communities. The robust SVM is an improved version of the SVM so that the resulting classifier can be less sensitive to outliers. In many practical problems, it may be advantageous to use different weights for different types of misclassification. However, the existing RSVM treats different kinds of misclassification equally. In this paper, we propose the weighted RSVM, as an extension of the standard SVM. We show that surprisingly, the cost-based weights do not work well for weighted extensions of the RSVM. To solve this problem, we propose a novel utility-based weighting scheme for the weighted RSVM. Both theoretical and numerical studies are presented to investigate the performance of the proposed weighted multicategory RSVM.

Keywords and phrases: Multicategory classification, Robustness, SVM, Utility, Weighted learning.

1. INTRODUCTION

In supervised learning, one important goal is to build predictive models using a training dataset for future prediction. Among various learning tasks, classification plays an important role, both theoretically and practically. It has been widely applied in a wide range of disciplines such as medicine, engineering, and bioinformatics.

There are numerous classification techniques proposed in the literature. In particular, several machine learning approaches become popular and have been increasingly studied in both machine learning and statistics communities. Important examples include the Support Vector Machine (SVM, Boser et al. (1992); Cortes and Vapnik (1995)), Boosting (Freund and Schapire, 1997; Friedman et al., 2000), and others. See Hastie et al. (2009) for a comprehensive survey of different learning techniques. Many proposals on further improvements of these methods have been made in recent years. For example, Lee et al. (2004); Zhu et al. (2009) introduced multicategory versions of SVM and AdaBoost respectively. A more recent learning method, ψ-learning, was first proposed by Shen et al. (2003) as a competitive binary large margin classifier. Liu and Shen (2006) developed a multicategory extension of ψ-learning. Wu and Liu (2007) further generalized ψ-learning to robust SVMs. Other examples of machine learning classification methods include the Import Vector Machine (IVM, Zhu and Hastie (2005)) and Distance Weighted Discrimination (DWD, Marron et al. (2007); Qiao et al. (2010)).

To measure the performance of a classifier, one can quantify its prediction accuracy. One commonly used measure is the probability of misclassification. This measure treats all types of misclassification equally. Specifically, the cost of misclassifying a subject of class one into class two is the same as the cost of misclassifying a subject of class two into class one. This treatment may not be appropriate for many applications. One common example is the application of tumor classification. Clearly, misclassifying a cancer patient as normal is much more severe than the other type of misclassification. A misclassification on a normal patient can be corrected using further diagnosis. However, a wrong diagnosis on a cancer patient will delay the necessary treatment and can be life threatening. Another example is learning with samples having minority classes. In that case, the resulting classifier tends to sacrifice the minority classes and tries to classify the training points in majority classes correctly. Sometimes the classifier may misclassify all points of a minority class but still give a high overall classification accuracy (Qiao and Liu, 2009). Therefore, unequal cost assignments on different types of misclassification are needed.

To handle the problem of unequal cost assignments, Lin et al. (2004) generalized the original SVM to nonstandard situations. The nonstandard SVM allows unequal cost assignments on the two types of misclassification. Lee et al. (2004) extended this idea further for multicategory problems. For ψ-learning and robust SVM, the available methods can only deal with standard binary and multicategory classification. In this paper, we develop a general robust SVM technique which allows unequal cost assignments on different types of misclassification. The proposed technique covers the binary ψ-learning in Shen et al. (2003), multicategory ψ-learning in Liu and Shen (2006), and robust SVM (RSVM) in Wu and Liu (2007) as special cases.

The optimization problem of the standard RSVM involves nonconvex minimization. Wu and Liu (2007) proposed to decompose the nonconvex objective function into

*Corresponding author.

†The authors would like to thank the reviewers for their constructive comments and suggestions. The authors are partially supported by NSF Grants DMS-0747575 (Liu) and DMS-0905561 (Wu), NIH/NCI Grants R01 CA-149569 (Liu and Wu) and P01 CA142538 (Liu).

the difference of two convex functions and then use difference convex (DC) algorithm through iterative convex minimization to obtain a local solution. Since the existing weighted SVMs in the literature are based on the use of unequal costs of different misclassification, it is natural for us to use the same idea for the RSVM. Surprisingly, the resulting weighted RSVM using the unequal cost approach cannot be solved using the DC algorithm. To solve this difficulty, we propose the novel utility-based weighted RSVM which can be implemented via the DC algorithm. We show the equivalence of cost and utility under the 0–1 loss. Numerical examples are provided to illustrate the effectiveness of the new methodology.

The rest of this paper is organized as follows. In Section 2, we review the standard ψ-learning technique. In Section 3, we propose the weighted multicategory RSVM methodology. A computational algorithm using the DC algorithm is provided in Section 4, followed by numerical examples in Section 5. Some discussions are provided in Section 6. The Appendix contains the technical proof of the theorem.

2. STANDARD ψ-LEARNING AND ROBUST SVM

2.1 General framework

Consider a k-class classification problem. Let $\{(\boldsymbol{x}_i, y_i);\ i = 1, \ldots, n\}$ denote a training dataset. The n pairs of observations $(\boldsymbol{x}_i, y_i)$'s are assumed to be independent realizations of a random pair $(\boldsymbol{X}, Y)$, which has an unknown probability distribution $P(\boldsymbol{x}, y)$. Here $\boldsymbol{x} \in S \subset \mathbb{R}^d$ denotes an input vector and $y \in \{1, \ldots, k\}$ represents an output (class) variable. Throughout the paper, we use $\boldsymbol{X}$ and Y to denote random variables and $\boldsymbol{x}$ and y to represent corresponding observations.

Define $\mathbf{f} = (f_1, \ldots, f_k)$, each f_j being a mapping from S to $\mathbb{R}$, as a decision function vector. These k functions represent k different classes with f_j corresponding to class j; $j = 1, \ldots, k$. Once $\mathbf{f}$ is obtained from the training dataset, a classifier $\operatorname{argmax}_{j=1,\ldots,k} f_j(\boldsymbol{x})$ is employed to predict the class of any input vector $\boldsymbol{x} \in S$. In other words, $f_{\hat{y}}(\boldsymbol{x})$ is the maximum among k values of $\mathbf{f}(\boldsymbol{x})$. One important goal of multicategory classification is to find a classifier which minimizes the probability of misclassifying a new input vector $\boldsymbol{X}$, namely the generalization error (GE), $\mathrm{Err}(\mathbf{f}) = P[Y \neq \operatorname{argmax}_j f_j(\boldsymbol{X})]$. Denote the multiple comparison vector of class y versus the rest as $\mathbf{g}(f(\boldsymbol{x}), y) = (f_y(\boldsymbol{x}) - f_1(\boldsymbol{x}), \ldots, f_y(\boldsymbol{x}) - f_{y-1}(\boldsymbol{x}), f_y(\boldsymbol{x}) - f_{y+1}(\boldsymbol{x}), \ldots, f_y(\boldsymbol{x}) - f_k(\boldsymbol{x}))$. Then $\mathbf{f}$ produces correct classification for $(\boldsymbol{x}, y)$ if $\min(\mathbf{g}(f(\boldsymbol{x}), y)) > 0$. Using the notation of generalized functional margin $\min(\mathbf{g}(f(\boldsymbol{x}), y))$, we can rewrite the classification error rate on the training dataset as $(1/n)\sum_{i=1}^n I(\min(\mathbf{g}(f(\boldsymbol{x}_i), y_i)) \leq 0)$, where $I(\cdot)$ is an indicator function.

2.2 Standard ψ-learning

After replacing the indictor function, also known as the 0–1 loss, by a ψ-loss function, the standard multicategory ψ-learning proposed by Liu and Shen (2006) solves the following minimization problem:

$$\min_{\mathbf{f}} \left(\lambda \sum_{j=1}^k J(f_j) + \frac{1}{n} \sum_{i=1}^n \psi(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), y_i))) \right) \quad (1)$$

$$\text{subject to} \quad \sum_{j=1}^k f_j(\boldsymbol{x}) = 0 \ \ \forall \boldsymbol{x} \in S,$$

where $\psi(u) \in (0, 1]$ if $u \in (0, \tau)$ and $\psi(u) = I(u \leq 0)$ otherwise. The first term $\sum_{j=1}^k J(f_j)$ in the objective function in (1) can be viewed as a roughness penalty of $\mathbf{f}$. For example, in linear learning where each f_j is a linear function, a common choice of $J(f_j)$ is the squared L_2 norm of the corresponding linear coefficients for $\boldsymbol{x}$. The penalty term also helps to enforce maximum separation of the data in the separable case (Cristianini and Shawe-Taylor, 2000; Liu and Shen, 2006). The second term $\frac{1}{n}\sum_{i=1}^n \psi(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), y_i)))$ in the objective function is a measure of goodness of fit of $\mathbf{f}$ on the training dataset. The reason to use the ψ-loss instead of the 0–1 loss is that problem (1) would be ill-posed if we replace $\psi(\cdot)$ by $I(\cdot)$. In fact, the solution $\operatorname{argmin}_{\mathbf{f}}(\lambda\sum_{j=1}^k J(f_j) + \frac{1}{n}\sum_{i=1}^n I(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), y_i)) \leq 0))$ is essentially $\mathbf{0}$ since for any $c \in (0, 1)$, $c\mathbf{f}$ yields the same training error as $\mathbf{f}$, but a smaller penalty than $\mathbf{f}$. The positive values of $\psi(u)$ when $u \in (0, 1]$ aim to eliminate the scaling problem of $I(\cdot)$ and make (1) a well-defined optimization problem. One particular ψ-loss suggested by Liu and Shen (2006) is piecewise linear with $\psi(u) = 1$ if $u \leq 0$, $1-u$ if $u \in (0, 1]$, and 0 otherwise. The tuning parameter λ balances the penalty term and the data fit term. The sum-to-zero constraint $\sum_{j=1}^k f_j(\boldsymbol{x}) = 0$ in (1) helps to solve the potential identifiability problem of $\mathbf{f}$ and reduce the dimension of the optimization problem.

Shen et al. (2003); Liu and Shen (2006) showed that ψ-learning is robust to outliers and deliver high classification accuracy by using the ψ-loss, a loss that resembles the 0–1 loss closely. However, the methods they proposed only allow penalizing different types of misclassification equally.

2.3 Standard robust SVM

Wu and Liu (2007) proposed the robust SVM (RSVM) via truncating the unbounded hinge loss of the SVM. Notice that the hinge loss $H_1(u) = [1-u]_+$ grows linearly when u decreases with $u \leq 1$. This implies that a point with large $1 - \min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}), y)$ results in large H_1 and, as a consequence, greatly influences the final solution. Such points are typically far away from their own classes and tend to deteriorate the SVM performance. The RSVM utilizes the truncated hinge loss function to reduce the influence of outliers. In particular, the truncated hinge loss function can be expressed as $T_s(u) = H_1(u) - H_s(u)$, where $H_s(u) = [s - u]_+$.

The value of s of the truncated loss $T_s(u)$ specifies the location of truncation. We set $s \le 0$ since a truncated loss with $s > 0$ is constant for $u \in [-s, s]$ and cannot distinguish those correctly classified points with $\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}), y) \in (0, s]$ from those wrongly classified points with $\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}), y) \in [-s, 0]$. When $s = -\infty$, no truncation has been performed and $T_s(u) = H_1(u)$. When $s = 0$, the truncated loss $T_0(u)$ becomes the ψ loss. As shown in Wu and Liu (2007), the choice of s is important and affects the performance of the RSVM. Interestingly, the numerical examples in Wu and Liu (2007) suggest that $s = 0$ for ψ-learning is not the optimal choice. The best value of s appears to be $-1/(k-1)$. The corresponding truncated loss enjoys Fisher consistency and also delivers most accurate classification results compared to the truncated loss functions with other values of s.

In Section 3, we develop a weighted RSVM methodology which permits flexible treatments on different types of misclassification. Since ψ-learning is a special case of the RSVM, our method offers weighted ψ-learning as a byproduct.

3. WEIGHTED RSVM

To extend the standard RSVM, one can use weights on different types of misclassification. A common technique is to use misclassification costs as weights. For example, Lin et al. (2004); Lee et al. (2004) used costs for extension of the standard SVM to nonstandard situations. In Section 3.1, we explore the use of costs for possible extension of the multicategory SVM based on the generalized functional margin $\min(\mathbf{g}(f(\boldsymbol{x}), y))$. In view of the difficulty on the implementation of the corresponding method, in Section 3.2, we propose a new way of weighting using utilities, instead of costs.

3.1 Challenge with the cost formulation

In the standard learning case, we do not distinguish different types of misclassification. The 0–1 loss can be represented as $I(\min(\mathbf{g}(f(\boldsymbol{x}_i), y_i)) \le 0)$. A natural way to extend binary margin loss functions into a multicategory case is to use the generalized functional margin $\min(\mathbf{g}(f(\boldsymbol{x}_i), y_i))$ as its argument. For example, the multicategory SVM proposed by Liu and Shen (2006) uses the loss $[1 - \min(\mathbf{g}(f(\boldsymbol{x}_i), y_i))]_+$ and solves the following optimization problem

$$\min_{\mathbf{f}} \left(\lambda \sum_{j=1}^{k} J(f_j) + \frac{1}{n} \sum_{i=1}^{n} [1 - \min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), y_i))]_+ \right) \quad \text{subject to} \quad \sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0. \tag{2}$$

To extend standard learning via weighting, we need to consider different types of misclassification. Let $\phi(\boldsymbol{x}) : \mathbb{R}^d \to \{1, \ldots, k\}$ be a classifier and $C_{y\phi(\boldsymbol{x})}$ represent the cost of misclassifying input $\boldsymbol{x}$ with class y into class $\phi(\boldsymbol{x})$. We set $C_{y\phi(\boldsymbol{x})} = 0$ if $\phi(\boldsymbol{x}) = y$ and $C_{y\phi(\boldsymbol{x})} > 0$ otherwise. This implies that no penalty shall be given for correct classification and a positive cost can be used when an error occurs. Under the same framework as in Section 2, our goal is to obtain $\mathbf{f}$ which minimizes the GE $\mathrm{Err}(\mathbf{f}) = E[C_{Y\phi(\boldsymbol{X})}]$.

Notice that $E[C_{Y\phi(\boldsymbol{X})}] = E[\sum_{j=1}^{k} C_{Yj} I(\phi(\boldsymbol{X}) = j)] = E[\sum_{j=1}^{k} C_{Yj} I(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{X}), j)) > 0)]$. Then an empirical version of $\mathrm{Err}(\mathbf{f})$ based on the training dataset can be written as

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} C_{y_i j} I(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j)) > 0). \tag{3}$$

Because of the scaling problem of $I(\cdot)$ as discussed in Section 2, (3) can not be used for learning directly. However, one can use a convex approximation to replace the indicator function. For example, a natural approximation is to replace $I(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j)) > 0)$ by $[1 + \min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j))]_+$. Then we can get the following empirical minimization problem

$$\min_{\mathbf{f}} \left(\lambda \sum_{j=1}^{k} J(f_j) + \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} C_{y_i j} [1 + \min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j))]_+ \right) \quad \text{subject to} \quad \sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0. \tag{4}$$

Through the use of costs, (4) can be viewed as a weighted extension of the multicategory SVM formulation in (2). Surprisingly, unlike (2), (4) is not a convex minimization problem anymore. To see that, we can first introduce slack variables ξ_{ij}, as commonly done in the SVM optimization, to replace $[1 + \min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j))]_+$ with constraints $\xi_{ij} \ge 0$ and $\xi_{ij} \ge 1 + \min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j))$. Note that the region satisfying the constraints $\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j)) \le \xi_{ij} - 1$ is not convex. As a result, (4) cannot be directly implemented using convex optimization.

3.2 The new utility formulation

In contrast to the cost formulation discussed in Section 3.1, we explore the use of utility in this section. In particular, we assign the utility amount $U_{y\phi(\boldsymbol{x})}$ for classifying input $\boldsymbol{x}$ with class y into class $\phi(\boldsymbol{x})$. Naturally, one should set U_{jj} a bigger number comparing to other U_{jl} for $l \ne j$. As a remark, both our costs and utilities are nonnegative.

To generalize the 0–1 loss, instead of minimizing the total cost, one should maximize the utility $E[U_{Y\phi(\boldsymbol{X})}] = E[\sum_{j=1}^{k} U_{Yj} I(\phi(\boldsymbol{X}) = j)] = E[\sum_{j=1}^{k} U_{Yj} I(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{X}), j)) > 0)]$. Then an empirical version of the total utility is as follows

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} U_{y_i j} I(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j)) > 0). \tag{5}$$

Notice that maximizing $E[U_{Y\phi(\boldsymbol{X})}]$ is equivalent to minimizing $E[-U_{Y\phi(\boldsymbol{X})}]$. Then it is straightforward to see that

maximizing the total utility (5) is equivalent to minimizing the following quantity

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} U_{y_i j} I(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j)) \le 0). \tag{6}$$

Therefore, the utility-based multicategory SVM extension can be written as follows

$$\min_{\mathbf{f}} \left(\lambda \sum_{j=1}^{k} J(f_j) + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} U_{y_i j}[1 - \min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j))]_+ \right) \quad \text{subject to} \quad \sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0. \tag{7}$$

It is easy to see that problem (7) is a natural generalization of the standard SVM problem (2) with weights $U_{y_i j}$. Similarly, our proposed weighted RSVM solves the following problem

$$\min_{\mathbf{f}} \left(\lambda \sum_{j=1}^{k} J(f_j) + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} U_{y_i j} T_s(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j))) \right) \quad \text{subject to} \quad \sum_{j=1}^{k} f_j(\boldsymbol{x}) = 0. \tag{8}$$

Our weighted RSVM uses the multicategory weighted truncated hinge loss function $\sum_{j=1}^{k} U_{y_i j} T_s(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}_i), j)))$. To further explore the proposed loss, we study its consistency. In particular, we first define weighted Fisher consistency and then study Fisher consistency of general truncated loss functions. Note that the Bayes rule $\phi^*(\boldsymbol{X})$ that maximizes the expected utility $E[U_{Y\phi(\boldsymbol{X})}]$ is $\phi^*(\boldsymbol{x}) = \operatorname{argmax}_j \sum_{l=1}^{k} U_{lj} p_l(\boldsymbol{x})$, where $p_l(\boldsymbol{x}) = P(Y = l | \boldsymbol{X} = \boldsymbol{x})$. Assume that the function $\ell(\cdot)$ is non-increasing and $\ell'(0) < 0$ exists. Then the weighted Fisher consistency is defined as follows.

Definition 3.1 (Weighted Fisher Consistency)**.** Denote $\mathbf{f}^*(\boldsymbol{x}) = \operatorname{argmin}_{\mathbf{f}} E[\sum_{j=1}^{k} U_{Yj} \ell(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{X}), j)))|\boldsymbol{X} = \boldsymbol{x}]$. Then the corresponding weighted loss function is weighted Fisher consistent if $\operatorname{argmax} \mathbf{f}^*(\boldsymbol{x}) = \operatorname{argmax}_j \sum_{l=1}^{k} U_{lj} p_l(\boldsymbol{x})$.

As a remark, we note that our weighted Fisher consistency definition and results are more general than that of Wu et al. (2010). Essentially, the weights imposed by Wu et al. (2010) can be viewed as a special case of our utility-based learning with a diagonal utility matrix.

Let $\ell_{T_s}(\cdot) = \min(\ell(\cdot), \ell(s))$ with $s \le 0$. The following theorem, as an extension of the results in Wu and Liu (2007), states the weighted Fisher consistent results of the weighted truncated loss $\sum_{j=1}^{k} U_{Yj} \ell_{T_s}(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}), j)))$.

Theorem 3.1. *Assume that the function $\ell(\cdot)$ is non-increasing and $\ell'(0) < 0$ exists. Then a sufficient condition for the loss $\sum_{j=1}^{k} U_{yj} \ell_{T_s}(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{x}), j)))$ with $k > 2$ to be weighted Fisher consistent is that the truncation location s satisfies that $\sup_{\{u: u \ge -s \ge 0\}}(\ell(0) - \ell(u))/(\ell(s) - \ell(0)) \ge (k-1)$. This condition is also necessary if $\ell(\cdot)$ is convex.*

Similar to the standard learning, the truncation value s given in Theorem 3.1 depends on the class number k. For $\ell(u) = H_1(u), e^{-u}$, and $\log(1 + e^{-u})$, weighted Fisher consistency for $\ell_{T_s}(\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}), y))$ can be guaranteed for $s \in [-\frac{1}{k-1}, 0], [\log(1 - \frac{1}{k}), 0]$, and $[-\log(2^{\frac{k}{k-1}} - 1), 0]$, respectively. For the implementation of our weighted RSVM, we recommend to choose $s = -\frac{1}{k-1}$, same as that for the standard RSVM.

3.3 Construction of the utility matrix

As we mentioned earlier, using costs to extend standard to nonstandard learning is very common. Weighted loss functions using costs extend the standard 0–1 loss by allowing unequal costs on different types of misclassification. Typically, one sets costs for correct classification to be zero and sets various costs for different types of misclassification depending on the problem and context. In this section, we describe one way of constructing a utility matrix based on a predefined cost matrix and then show their equivalence.

With a prespecified cost matrix $\{C_{jl}; j, l = 1, \ldots, k\}$, we can construct the utility matrix with

$$\begin{pmatrix} U_{11} & \cdots & U_{1k} \\ \cdots & \cdots & \cdots \\ U_{k1} & \cdots & U_{kk} \end{pmatrix} = \max_{jl}\{C_{jl}\} \mathbf{1}_k \mathbf{1}_k^T - \begin{pmatrix} C_{11} & \cdots & C_{1k} \\ \cdots & \cdots & \cdots \\ C_{k1} & \cdots & C_{kk} \end{pmatrix},$$

where $\mathbf{1}$ is a vector of 1 with length k.

Next we demonstrate that this choice of utility is reasonable by showing that

$$\begin{aligned} &\operatorname{argmin}_{\mathbf{f}} E\left[\sum_{j=1}^{k} C_{Yj} I(\min \mathbf{g}(\mathbf{f}, j) > 0)\right] \\ &\quad = \operatorname{argmax}_{\mathbf{f}} E\left[\sum_{j=1}^{k} U_{Yj} I(\min \mathbf{g}(\mathbf{f}, j) > 0)\right]. \end{aligned} \tag{9}$$

The equality (9) implies that using our choice of utility matrix, the solution $\mathbf{f}$, which minimizes the expected cost, also maximizes the expected utility. This justifies the usage of our utility matrix.

To show (9), we define $P_j = P(Y = j | \boldsymbol{X} = \boldsymbol{x})$. Then we have

$$\begin{aligned} &\operatorname{argmin}_{\mathbf{f}} E\left[\sum_{j=1}^{k} C_{Yj} I(\min \mathbf{g}(\mathbf{f}, j) > 0)\right] \\ &\quad = \operatorname{argmin}_{\mathbf{f}} \sum_{l=1}^{k} P_l \sum_{j=1}^{k} [C_{lj} I(\min \mathbf{g}(\mathbf{f}, j) > 0)] \end{aligned}$$

$$= \text{argmax}_{\mathbf{f}} \sum_{l=1}^{k} P_l \sum_{j=1}^{k} [(\max_{j^* l^*} C_{j^* l^*} - C_{lj}) I(\min \mathbf{g}(\mathbf{f}, j) > 0)]$$

$$= \text{argmax}_{\mathbf{f}} E\left[\sum_{j=1}^{k} U_{Yj} I(\min \mathbf{g}(\mathbf{f}, j) > 0)\right].$$

Thus, (9) is proved. As a result, one can construct the utility matrix directly once the cost matrix is given for the proposed weighted RSVM.

4. NONCONVEX MINIMIZATION VIA DIFFERENCE CONVEX ALGORITHM

In this section, we develop a difference convex algorithm to solve (8). Note that the loss function in (8) is piecewise linear. Consequently it can also be solved by developing a mixed integer programming (MIP) algorithm as in Liu and Wu (2006). However due to the computational intensity of the MIP, we use the DC algorithm. The DC algorithm solves the nonconvex minimization problem via minimizing a sequence of convex subproblems (An and Tao, 1997; Liu et al., 2005). In particular, to apply the DC algorithm, we first rewrite the nonconvex objective function as a difference of two convex functions. Then we solve the original nonconvex optimization problem via iterative convex minimization problems. Each convex minimization subproblem serves as an approximation of the original problem.

For simplicity, we only focus on linear learning. The DC algorithm for nonlinear learning can be derived using kernel formulation similarly to the idea of iterative approximation in linear learning. More details on the implementation of nonlinear learning can be founded in Wu and Liu (2007). For linear learning, we set $f_j(\boldsymbol{x}) = \boldsymbol{w}_j^T \boldsymbol{x} + b_j$; $\boldsymbol{w}_j \in \Re^d$, $b_j \in \Re$, and $\boldsymbol{b} = (b_1, b_2, \ldots, b_k)^T \in \Re^k$, where $\boldsymbol{w}_j = (w_{1j}, w_{2j}, \ldots, w_{dj})^T$, and $\boldsymbol{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k)$. With the two-norm penalty $J(f_j) = \frac{1}{2}\|\boldsymbol{w}_j\|_2^2$, (8) simplifies to

(10)

$$\min_{\boldsymbol{W}, \boldsymbol{b}} \frac{1}{2} \sum_{j=1}^{k} \|\boldsymbol{w}_j\|_2^2 + C \sum_{i=1}^{n} \sum_{j=1}^{k} U_{y_i j} T_s(\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i), j))$$
$$\text{subject to} \quad \sum_{j=1}^{k} w_{jm} = 0, m = 1, 2, \ldots, d; \sum_{j=1}^{k} b_j = 0,$$

where $C = \frac{1}{n\lambda}$ and the constraints are adopted to avoid a non-identifiability issue of the solution.

We denote parameters $(\boldsymbol{W}, \boldsymbol{b})$ by Θ. By noting the fact that $T_s = H_1 - H_s$, the objective function in (10) can be decomposed as

$$Q(\Theta) = \frac{1}{2} \sum_{j=1}^{k} \|\boldsymbol{w}_j\|_2^2 + C \sum_{i=1}^{n} \sum_{j=1}^{k} U_{y_i j} H_1(\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i), j))$$
$$- C \sum_{i=1}^{n} \sum_{j=1}^{k} U_{y_i j} H_s(\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i), j))$$
$$= Q_{vex}(\Theta) + Q_{cav}(\Theta),$$

where

$$Q_{vex}(\Theta) = \frac{1}{2} \sum_{j=1}^{k} \|\boldsymbol{w}_j\|_2^2 + C \sum_{i=1}^{n} \sum_{j=1}^{k} U_{y_i j} H_1(\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i), j))$$

and

$$Q_{cav}(\Theta) = -C \sum_{i=1}^{n} \sum_{j=1}^{k} U_{y_i j} H_s(\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i), j))$$

denote the convex and concave parts, respectively.

Note that $\frac{\partial}{\partial \boldsymbol{w}_j} Q_{cav}(\Theta)$ and $\frac{\partial}{\partial b_j} Q_{cav}(\Theta)$ can be written respectively as follows

$$- C \sum_{i=1}^{n} \Big[U_{y_i j} (-I_{\{\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i), j) < s\}})$$
$$+ \sum_{j' \neq j} U_{y_i j'} (I_{\{j = \text{argmax}(f_m(\boldsymbol{x}_i):\, m \neq j'), f_{j'}(\boldsymbol{x}_i) - f_j(\boldsymbol{x}_i) < s\}}) \Big] \boldsymbol{x}_i^T$$

and

$$- C \sum_{i=1}^{n} \Big[U_{y_i j} (-I_{\{\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i), j) < s\}})$$
$$+ \sum_{j' \neq j} U_{y_i j'} (I_{\{j = \text{argmax}(f_m(\boldsymbol{x}_i):\, m \neq j'), f_{j'}(\boldsymbol{x}_i) - f_j(\boldsymbol{x}_i) < s\}}) \Big],$$

where $I_{\{A\}} = 1$ if event A is true, and 0 otherwise.

Define, for $j' \neq j$, $\beta_{ijj'} = C$ if $f_j^t - f_{j'}^t < s$ with $j' = \text{argmax}(f_m^t : m \neq j)$ and 0 otherwise. With the help of β_{ij}, we have

$$\frac{\partial}{\partial \boldsymbol{w}_j} Q_{cav}(\Theta) = \sum_{i=1}^{n} \left(U_{y_i j} \sum_{j' \neq j} \beta_{ijj'} - \sum_{j' \neq j} U_{y_i j'} \beta_{ij'j} \right) \boldsymbol{x}_i^T,$$

and

$$\frac{\partial}{\partial b_j} Q_{cav}(\Theta) = \sum_{i=1}^{n} \left(U_{y_i j} \sum_{j' \neq j} \beta_{ijj'} - \sum_{j' \neq j} U_{y_i j'} \beta_{ij'j} \right).$$

We now describe the iterative procedure of the DC algorithm for minimizing the nonconvex objective function in (10). For initialization, we can use the solution of (10) when $T_s(u) = H_1(u)$, i.e., when no truncation is performed on the loss function with $s = -\infty$. In that case, problem (10) becomes a convex minimization problem. Denote Θ_t to be the solution at the end of step t. At the step $(t+1)$, we apply

the linear approximation to the concave part and then the objective function becomes

$$Q(\Theta) = \frac{1}{2}\sum_{j=1}^{k}\|\boldsymbol{w}_j\|_2^2 + C\sum_{i=1}^{n}\sum_{j=1}^{k}U_{y_ij}H_1(\min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i), j)) + \sum_{j=1}^{k}\left\langle \frac{\partial}{\partial \boldsymbol{w}_j}Q_{cav}(\Theta_t), \boldsymbol{w}_j \right\rangle + \sum_{j=1}^{k} b_j \frac{\partial}{\partial b_j}Q_{cav}(\Theta_t).$$

Using slack variable ξ_{ij}'s for the hinge loss function, the optimization problem at step $(t+1)$ becomes

$$\begin{aligned}\min_{\boldsymbol{W},\boldsymbol{b},\boldsymbol{\xi}} \ & \frac{1}{2}\sum_{j=1}^{k}\|\boldsymbol{w}_j\|_2^2 + C\sum_{i=1}^{n}\sum_{j=1}^{k}U_{y_ij}\xi_{ij} \\ & + \sum_{j=1}^{k}\left\langle \frac{\partial}{\partial \boldsymbol{w}_j}Q_{cav}(\Theta_t), \boldsymbol{w}_j \right\rangle + \sum_{j=1}^{k} b_j \frac{\partial}{\partial b_j}Q_{cav}(\Theta_t) \\ \text{subject to} \ & \xi_{ij} \geq 0 \ \ i = 1, 2, \ldots, n; j = 1, 2, \ldots, k \\ & \xi_{ij} \geq 1 - \left[\boldsymbol{x}_i^T\boldsymbol{w}_j + b_j\right] + \left[\boldsymbol{x}_i^T\boldsymbol{w}_{j'} + b_{j'}\right], \\ & \quad i = 1, 2, \ldots, n; \ \ j' \neq j.\end{aligned}$$

The corresponding Lagrangian is

(11)

$$\begin{aligned}& L(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\xi}) \\ & = \frac{1}{2}\sum_{j=1}^{k}\|\boldsymbol{w}_j\|_2^2 + C\sum_{i=1}^{n}\sum_{j=1}^{k}U_{y_ij}\xi_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{k}u_{ij}\xi_{ij} \\ & \quad - \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{j'\neq j}\alpha_{ijj'}(\boldsymbol{x}_i^T\boldsymbol{w}_j + b_j - \boldsymbol{x}_i^T\boldsymbol{w}_{j'} - b_{j'} + \xi_{ij} - 1) \\ & \quad + \sum_{j=1}^{k}\left\langle \frac{\partial}{\partial \boldsymbol{w}_j}Q_{cav}(\Theta_t), \boldsymbol{w}_j \right\rangle + \sum_{j=1}^{k} b_j \frac{\partial}{\partial b_j}Q_{cav}(\Theta_t),\end{aligned}$$

subject to

$$\frac{\partial}{\partial \boldsymbol{w}_j}L = \boldsymbol{w}_j^T - \left[\sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ijj'} - U_{y_ij}\beta_{ijj'})\boldsymbol{x}_i^T - \sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ij'j} - U_{y_ij'}\beta_{ij'j})\boldsymbol{x}_i^T\right] = 0 \tag{12}$$

$$\frac{\partial}{\partial b_j}L = -\left[\sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ijj'} - U_{y_ij}\beta_{ijj'}) - \sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ij'j} - U_{y_ij'}\beta_{ij'j})\right] = 0 \tag{13}$$

$$\frac{\partial}{\partial \xi_{ij}}L = CU_{y_ij} - u_{ij} - \sum_{j'\neq j}\alpha_{ijj'} = 0, \tag{14}$$

where the Lagrangian multipliers are $u_{ij} \geq 0$ and $\alpha_{ijj'} \geq 0$ for any $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, k$, $j' \neq j$. Substituting (12)–(14) into (11) yields the corresponding dual problem

$$\begin{aligned}\min_{\boldsymbol{\alpha}} \ & \frac{1}{2}\sum_{j=1}^{k}\left\|\sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ijj'} - U_{y_ij}\beta_{ijj'})\boldsymbol{x}_i^T \right. \\ & \left. - \sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ij'j} - U_{y_ij'}\beta_{ij'j})\boldsymbol{x}_i^T\right\|_2^2 - \sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{j'\neq j}\alpha_{ijj'} \\ \text{subject to} \ & \sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ijj'} - U_{y_ij}\beta_{ijj'}) - \sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ij'j} \\ & \quad - U_{y_ij'}\beta_{ij'j}) = 0, \ j = 1, 2, \ldots, k \\ & 0 \leq \sum_{j'\neq j}\alpha_{ijj'} \leq CU_{y_ij}, \\ & \quad i = 1, 2, \ldots, n; \ j = 1, 2, \ldots, k \\ & \alpha_{ijj'} \geq 0, \ i = 1, 2, \ldots, n; \ j = 1, 2, \ldots, k; \ j' \neq j;\end{aligned}$$

where $\beta_{ijj'}$ is defined as above. Note that the above dual problem is a quadratic programming (QP) problem similar to that of the standard SVM. Much optimization software can be used to solve the above dual problem. Once the solution is obtained, the coefficients $\boldsymbol{w}_j$'s can be recovered as follows,

$$\boldsymbol{w}_j = \sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ijj'} - U_{y_ij}\beta_{ijj'})\boldsymbol{x}_i^T - \sum_{i=1}^{n}\sum_{j'\neq j}(\alpha_{ij'j} - U_{y_ij'}\beta_{ij'j})\boldsymbol{x}_i^T, \tag{15}$$

which satisfies the sum-to-zero constraint $\sum_{j=1}^{k} w_{mj} = 0$ for each $1 \leq m \leq d$ automatically.

After the solution of $\boldsymbol{W}$ is derived, $\boldsymbol{b}$ can be obtained via solving either a sequence of KKT conditions as used in the standard SVM or a linear programming (LP) problem. Denote $\tilde{f}_j(\boldsymbol{x}_i) = \boldsymbol{x}_i^T\boldsymbol{w}_j$. Then $\boldsymbol{b}$ can be obtained through the following LP problem:

$$\begin{aligned}\min_{\boldsymbol{\eta},\boldsymbol{b}} \ & C\sum_{i=1}^{n}\sum_{j=1}^{k}U_{y_ij}\eta_{ij} \\ & + \sum_{j=1}^{k}\left[\sum_{i=1}^{n}(U_{y_ij}\sum_{j'\neq j}\beta_{ijj'} - \sum_{j'\neq j}U_{y_ij'}\beta_{ij'j})\right]b_j \\ \text{subject to} \ & \eta_{ij} \geq 0, \ i = 1, 2, \ldots, n \\ & \eta_{ij} \geq 1 - (\tilde{f}_j(\boldsymbol{x}_i) + b_j) + \tilde{f}_{j'}(\boldsymbol{x}_i) + b_{j'}, \\ & \quad i = 1, 2, \ldots, n; \ j' \neq j \\ & \sum_{j=1}^{k} b_j = 0.\end{aligned}$$

We continue iterating the above convex optimization steps until convergence. Its convergence is guaranteed due to the fact that the objective function value decreases at each iteration.

5. NUMERICAL EXAMPLES

We investigate the performance of our weighted RSVM through simulated examples in Section 5.1. We use three simulated examples to show the effect of the utility matrix and the improvement of weighted RSVM over the weighted SVM. A handwritten digit recognition example is presented in Section 5.2 to further demonstrate the use of our proposed weighted RSVM. For all examples discussed in this section, the tuning parameter λ is selected over a grid using an independent tuning data set.

5.1 Simulation

We consider three simulated examples. For Examples 1 and 2, the underlying Bayes decision boundary is piecewise linear. We perform both linear and kernel nonlinear learning in Example 1 to demonstrate the change of the decision boundary with the utility matrix. Example 2 is used to show the advantage of the weighted RSVM over the weighted SVM when there are outliers in the data. For Example 3, the underlying Bayes decision boundary is nonlinear and we study the performance of our nonlinear weighted RSVM.

Example 1. This example is used to illustrate how boundary moves when we change the utility matrix. The data of this piecewise linear example are generated as follows: the class response Y has equal probabilities taking $1, 2$, or 3; conditional on $Y = y$, $\boldsymbol{X} \sim N(\boldsymbol{\mu}_y, 0.7^2 I_2)$ where I_2 is a 2×2 identity matrix, $\boldsymbol{\mu}_1 = (1, 0)^T$, $\boldsymbol{\mu}_2 = (-1/2, \sqrt{3}/2)^T$, and $\boldsymbol{\mu}_3 = (-1/2, -\sqrt{3}/2)^T$.

To study the effect of the utility matrix, we consider four different configurations of the utility matrix as follows:

$$U_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad U_{1,a} = \begin{pmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{16}$$

$$U_{2,a} = \begin{pmatrix} 1 & a & a \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad U_{3,a} = U_{2,a}^T,$$

where $0 \le a < 1$.

Since the true decision boundary is piecewise linear, we apply both linear learning and nonlinear kernel learning. For linear learning, the sample size is set to be $n = 1600$. Figure 1 shows how the decision boundaries change as we change the utility matrix. For the utility matrix $U_{1,a}$, the utility of classifying points in class 1 into class 2 increases as a increases. As a result, the region for class 1 becomes smaller while the region for class 2 becomes larger as a increases, as shown on the left panel of Figure 1. For the utility matrix $U_{2,a}$, the utilities of classifying points in class 1 into class 2 or 3 increase as a increases. Thus, the regions for classes 2 and 3 become larger and the region for class 1 becomes smaller as a increases, as shown on the middle panel of Figure 1. In contrast to $U_{2,a}$, for $U_{3,a}$, the utilities of classifying points in class 2 or 3 into class 1 increase as a increases. Consequently, the right panel of Figure 1 shows that the regions for classes 2 and 3 become smaller and the region for class 1 becomes larger as a increases. Overall, the results match our expectation when we change the utility matrix.

Next we apply nonlinear learning via Gaussian kernel and see how the boundaries move with the change of the utility matrix. In this case, we set the sample size $n = 800$. We show the boundaries for one typical realization in Figure 2. The first, second and third rows of Figure 2 correspond to the changes of decision boundaries as we change a in (16). The pattern changes are similar to that of linear learning. In particular, for the first row, the region for class 2 gets larger while that for class 1 gets smaller. For the second row, the regions for both classes 2 and 3 get larger while that for class 1 gets smaller. Interestingly, as a gets larger, the region for class 1 is mixed with the regions for class 2 and 3. This is possibly due to the same value for U_{12} and U_{13}. As a gets larger, the chances of classifying a point in class 1 into one of the three classes are approximately the same and thus the decision is close to being random. For the third row, the region for class 1 gets larger at the expense of smaller regions for classes 2 and 3.

Example 2. This example is used to demonstrate the performance of weighted SVM and weighted RSVM when there are outliers in the data. For simplicity, we use a similar data generation scheme as in Example 1. We first generate $(X, \tilde{Y})$ as in Example 1. Then we perform a data contamination step to generate outliers. Specifically, conditional on $\tilde{Y} = \tilde{y}$, random flip $\tilde{Y}$ to get the final response Y by setting $Y = j$ with probability $P(j, \tilde{y})$, where $j = 1, 2, 3$. We set $P(j, \tilde{y}) = 0.85$ when $j = \tilde{y}$ and 0.075 otherwise. With outliers existing in the data, we want to examine the effect of truncation of the hinge loss for the weighted RSVM.

For this example, the sample size is set to be $n = 400$. We generate an independent test set of size $n_{test} = 100n$. For a utility matrix $U = (u_{ij})$, we report the average utility on the test set as defined by $\sum_{i=1}^{n_{test}} u_{y_i \hat{y}_i} / n_{test}$ for every $(\boldsymbol{x}_i, y_i)$ in the test set with corresponding prediction $\hat{y}_i$. Means and standard deviations of the average utilities of over 100 repetition are reported for different methods in Table 1. We can see that using truncation can help to produce classifiers with higher utilities than those without truncation in most cases.

Example 3. Examples 1 and 2 have piecewise linear Bayes decision boundaries. For Example 3, we consider an example with nonlinear Bayes decision boundary. In particular, predictors $\boldsymbol{X} = (X_1, X_2)^T$ are generated with $X_1 \sim \mathit{Uniform}[-3, 3]$ and $X_2 \sim \mathit{Uniform}[-6, 6]$. Conditional on $\boldsymbol{X} = \boldsymbol{x}$, the initial reponse $\tilde{Y}$ takes value j with

Table 1. Utility comparison for the weighted SVM (WSVM) and RSVM (WRSVM) for Example 2

	Utility	$a = 0$	$a = 0.2$	$a = 0.4$	$a = 0.6$	$a = 0.8$
WRSVM	$U_{1,a}$	70.67 (0.24)	71.58 (0.36)	72.60 (0.31)	73.90 (0.42)	74.79 (0.94)
	$U_{2,a}$		72.53 (0.34)	74.57 (0.45)	76.42 (0.99)	79.99 (0.29)
	$U_{3,a}$		72.82 (0.28)	75.44 (0.31)	77.39 (1.84)	86.59 (0.09)
WSVM	$U_{1,a}$	70.62 (0.28)	71.32 (0.47)	72.13 (0.49)	72.88 (0.93)	73.96 (0.71)
	$U_{2,a}$		71.98 (0.70)	71.88 (1.04)	74.46 (0.38)	80.04 (0.26)
	$U_{3,a}$		72.60 (0.43)	74.35 (1.08)	75.90 (1.58)	86.61 (0.08)

Note: All table entries are multiplied by 100.

Table 2. Utility comparison for the weighted SVM (WSVM) and RSVM (WRSVM) for Example 3

	$U_{1,0.4}$	$U_{2,0.4}$	$U_{3,0.4}$
WRSVM	43.47 (1.88)	47.09 (1.11)	57.62 (3.10)
WSVM	43.10 (1.76)	47.28 (1.09)	55.13 (2.80)

Note: All table entries are multiplied by 100.

probability $\exp(f_j(\boldsymbol{x}_i))/\sum_{m=1}^{3}\exp(f_m(\boldsymbol{x}_i))$ with $f_1(\boldsymbol{x}) = -20x_1 + 2x_2^2 - x_2^2 + 2$, $f_2(\boldsymbol{x}) = -4x_1^2 + 2x_2^2 - 4$, and $f_3(\boldsymbol{x}) = 20x_1 + 2x_1^2 - x_2^2 + 2$. Conditional on $\tilde{Y} = \tilde{y}$, randomly flip $\tilde{Y}$ to get the final reponse Y by setting $Y = j$ with probability $P(j, \tilde{y})$, where $j = 1, 2, 3$. We set $P(j, \tilde{y}) = 0.85$ when $j = \tilde{y}$ and 0.075 otherwise.

We choose $n = 200$, $n_{test} = 10n$, and three specific utility matrices $U_{1,0.4}$, $U_{2,0.4}$, and $U_{3,0.4}$. Average utility results are given in Table 2. In general, loss truncation for the weighted RSVM helps to deliver classifiers with larger utilities except for the case of $U_{2,0.4}$.

5.2 Handwritten digit recognition

In this session, we use one real data example to further illustrate our proposed method. The real dataset we use is the "Pen-Based Recognition of Handwritten Digits" available online at the UCI Machine Learning Repository. See Alimoglu and Alpaydin (1996) for more information related to this data set.

For this digit dataset, the response variable is multicategory with class codes being $0, 1, 2, \ldots, 9$, representing corresponding digits. To simplify the task, we focus on three classes with class codes of 3, 6, and 9, which are labeled as class 1, 2, and 3, respectively, in our weighted RSVM. After combining both the training data and the testing data, we have 3,166 observations with class codes 3, 6, and 9. There are 16 predictors available, which are first standardized to have mean zero and standard deviation one. For each replication, we randomly select 50 observations from each class to be used as the training data, another 50 observations for each class from the remaining to be used as the tuning data, and the rest as the test data. We repeat this random splitting 20 times.

Table 3. Classification results for the hand written digit recognition data

		Percentage		
		$\hat{y} = 1$	$\hat{y} = 2$	$\hat{y} = 3$
$U_{2,0}$	$y = 1$	0.9929	0.0005	0.0066
	$y = 2$	0.0001	0.9989	0.0010
	$y = 3$	0.0130	0.0002	0.9868
$U_{2,0.2}$	$y = 1$	0.9886	0.0034	0.0080
	$y = 2$	0.0001	0.9990	0.0009
	$y = 3$	0.0109	0.0015	0.9875
$U_{2,0.6}$	$y = 1$	0.4545	0.1573	0.3882
	$y = 2$	0	0.9990	0.0010
	$y = 3$	0	0.0002	0.9998

We report the results based on these 20 splitting in Table 3. For each utility matrix and each random splitting, we calculate the average percentage of predicting observations of each class to different classes. For the purpose of illustration, we only report the results for the utility type $U_{2,a}$ in Table 3. For the case of standard learning with utility $U_{2,0}$, we can see that the correct classification rates for the three classes are all around 0.99. When we increase a, more and more data points in class 1 are classified into classes 2 and 3 as the design of this utility matrix encourages such kinds of misclassification.

6. DISCUSSION

Multicategory classification is an important statistical problem in practice. In this paper, we propose a weighted extension of the multicategory RSVM. A novel utility-based weighting method is proposed. The resulting weighted RSVM can be implemented using the DC algorithm. The connections between the cost matrix and the utility matrix are explored. Our numerical examples demonstrate the effectiveness of our weighted RSVM.

Our method requires a predefined utility matrix in order to apply the weighted learning. Although we show that the utility matrix can be constructed using the cost matrix, the method requires the users to specify the cost or utility matrix. How to best assign sensible costs or utilities for multicategory classification is an important topic. Further investigation is necessary.

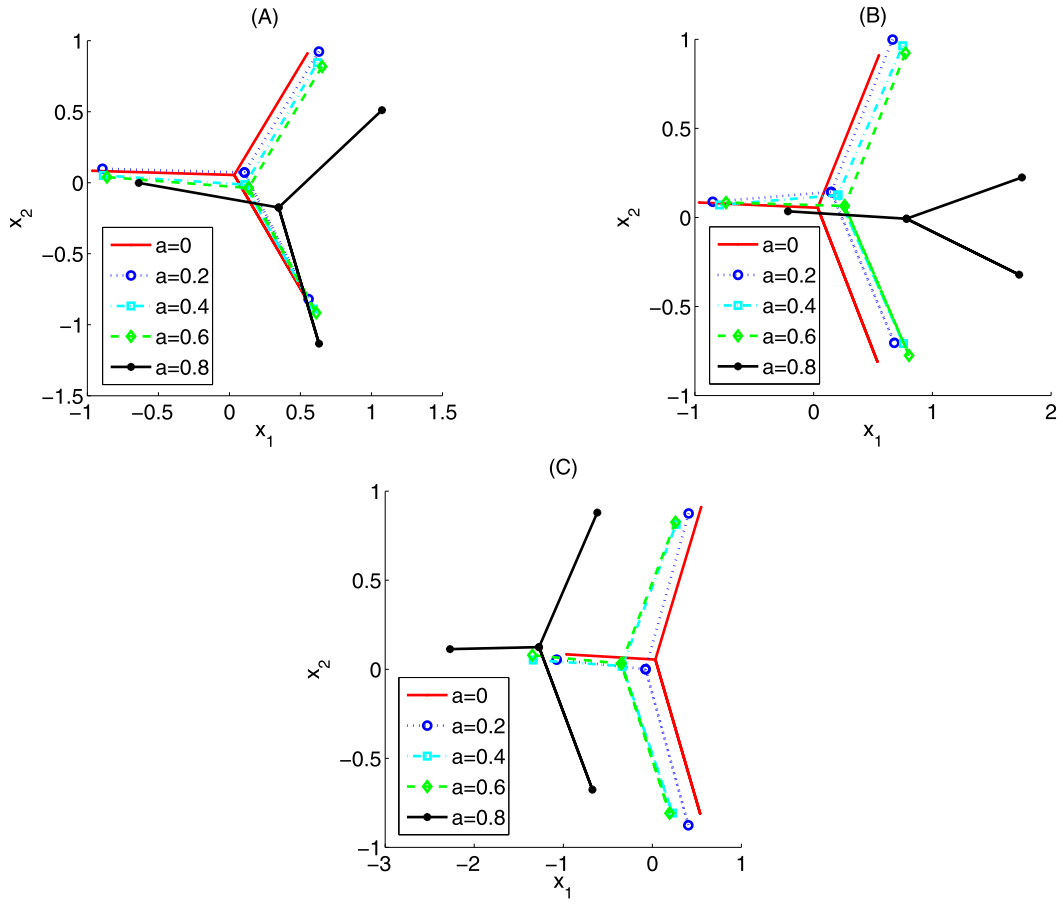

Figure 1. Boundaries for three configurations of the utility matrix with different a's in (16) for Example 1. The left, middle, and right panels correspond to utility matrices $U_{1,a}, U_{2,a}, U_{3,a}$ respectively.

APPENDIX

Proof of Theorem 1. Note that

$$E\Big[\sum_{j=1}^{k} U_{Yj}\ell_{T_s}(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{X}),j)))\Big] = E\Big[\sum_{l=1}^{k} U_{lj}\ell_{T_s}(\min(\mathbf{g}(\mathbf{f}(\boldsymbol{X}),j)))p_l(\boldsymbol{X})\Big].$$

For any given $\boldsymbol{x}$, we need to minimize $\sum_{l=1}^{k} U_{lj}\ell_{T_s}(g_j)p_l(\boldsymbol{x})$ where $g_j = \min \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}), j)$. By definition and the fact that $\sum_{j=1}^{k} f_j = 0$, we can conclude that $\max_j g_j \geq 0$ and at most one of g_j's is positive. Assume $j_p = \operatorname{argmax}_j \sum_{l=1}^{k} U_{lj}p_l(\boldsymbol{x})$ is unique. Then using the non-increasing property of ℓ_{T_s} and $\ell'(0) < 0$, the minimizer $\boldsymbol{f}^*$ satisfies that $g^*_{j_p} \geq 0$.

We are now left to show $g^*_{j_p} \neq 0$, equivalently that $\mathbf{0}$ cannot be a minimizer. For simplicity, denote $A = \sum_{l=1}^{k} u_{lj_p}p_l(\boldsymbol{x})$, $B = \sum_{j\neq j_p}\sum_{l=1}^{k} U_{lj}p_l(\boldsymbol{x})$, and $C = A + B$. Note $A > C/k$ due to the uniqueness of j_p. Then it is sufficient to show that there exists a solution with $g_{j_p} > 0$. By assumption, there exists $u_1 > 0$ such that $u_1 \geq -s$ and $(\ell(0) - \ell(u_1))/(\ell(s) - \ell(0)) \geq k - 1$. Consider a solution $\boldsymbol{f}^0$ with $f^0_{j_p} = u_1(k-1)/k$ and $f^0_j = -u_1/k$ for $j \neq j_p$. We want to show that $\boldsymbol{f}^0$ yields a smaller expected loss than $\mathbf{0}$, i.e., $A\ell_{T_s}(u_1) + B\ell_{T_s}(-u_1) < \ell_{T_s}(0)C$. Equivalently, $(\ell(0) - \ell(u_1))/(\ell(s) - \ell(0)) > B/A$, which holds due to the fact that $B/A < (k-1)$. This implies sufficiency of the condition.

To prove necessity of the condition, it is sufficient to show that if $(\ell(0)-\ell(u))/(\ell(s)-\ell(0)) < (k-1)$ for all u with $-u \leq s \leq 0$, $\mathbf{0}$ is a minimizer of $\sum_{l=1}^{k} U_{lj}\ell_{T_s}(g_j)p_l(\boldsymbol{x})$. Equivalently, we need to show that there exists $(p_1, \ldots, p_k)$ such that $\sum_{l=1}^{k} U_{lj}\ell_{T_s}(g_j)p_l(\boldsymbol{x}) \geq \ell_{T_s}(0)C$ for all $\boldsymbol{f}$. Without loss of generality, assume that $j_p = k$ and $f_1 \leq f_2 \leq \cdots \leq f_k$. Then $\sum_{l=1}^{k}\sum_{j=1}^{k} U_{lj}\ell_{T_s}(g_j)p_l(\boldsymbol{x}) = \sum_{j=1}^{k-1}\sum_{l=1}^{k} U_{lj}\ell_{T_s}(f_j - f_k)p_l(\boldsymbol{x}) + \sum_{l=1}^{k} U_{lk}\ell_{T_s}(f_k - f_{k-1})p_l(\boldsymbol{x}) \geq \ell_{T_s}(f_{k-1} - f_k)\sum_{j=1}^{k-1}\sum_{l=1}^{k} U_{lj}p_l(\boldsymbol{x}) + \ell_{T_s}(f_k - f_{k-1})\sum_{l=1}^{k} U_{lk}p_l(\boldsymbol{x}) = \ell_{T_s}(f_{k-1}-f_k)B + \ell_{T_s}(f_k - f_{k-1})A$ since ℓ_{T_s} is non-increasing. Thus it is sufficient to show $\ell_{T_s}(f_k - f_{k-1})A + \ell_{T_s}(f_{k-1} - f_k)B > \ell_{T_s}(0)C$, that is, $B(\ell_{T_s}(-u) - \ell(0)) > A(\ell(0) - \ell(u))$ for all $u > 0$. Since $\ell(\cdot)$ is convex, $B(\ell(-u) - \ell(0)) > A(\ell(0) - \ell(u))$ holds for all $0 < u \leq -s$ with

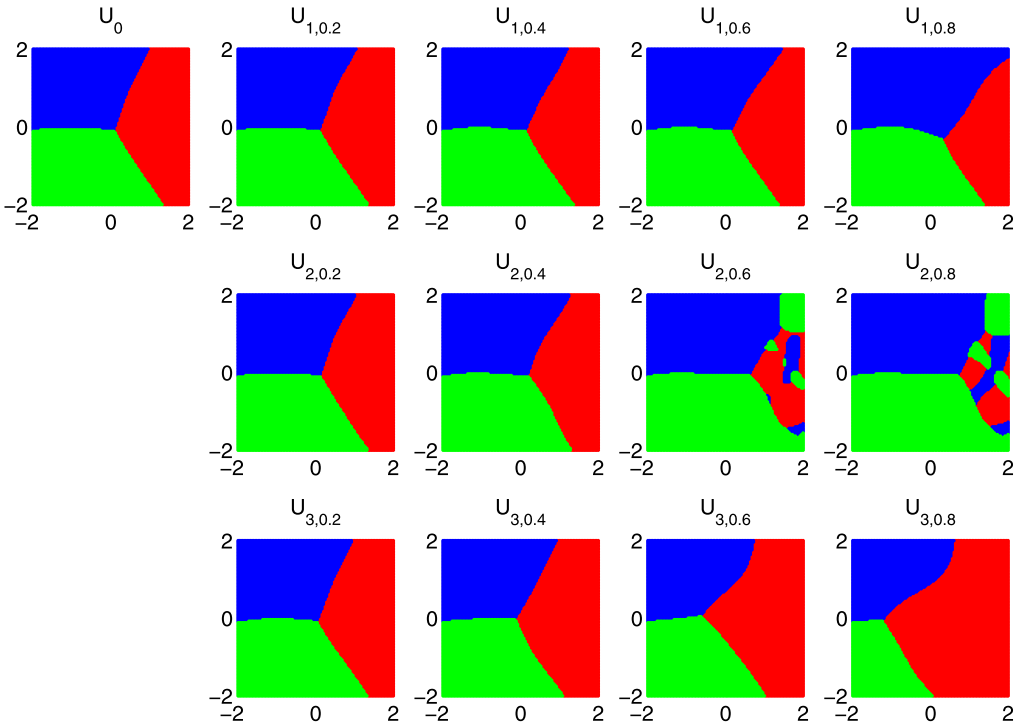

Figure 2. Boundaries for three utility matrix configuration with different a.

$\ell_{T_s}(-u) = \ell(-u)$. For $u \geq -s$, it is equivalent to show $B(\ell(s) - \ell(0)) > A(\ell(0) - \ell(u))$. By assumption, we can set $(\ell(s) - \ell(0)) = (\ell(0) - \ell(u))/(k-1) + a$ for some $a > 0$. Denote $(\ell(0) - \ell(u)) = W$. Then we need to have $B(W/(k-1)+a) > AW$. Let $A = C/k+\epsilon$. Then it becomes $((k-1)/kC-\epsilon)(W/(k-1)+a) > (C/k+\epsilon)W$, equivalently,

$$aC\frac{k-1}{k\epsilon} > \frac{k}{k-1}W + a. \tag{17}$$

For any given $a > 0$, $C \geq A > 0$ and $W > 0$, we can always find a small $\epsilon > 0$ to have (17) satisfied. The desired result then follows. □

Received 26 September 2010

REFERENCES

Alimoglu, F. and Alpaydin, E. (1996). Methods of Combining Multiple Classifiers Based on Different Representations for Pen-based Handwriting Recognition, *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96). Istanbul, Turkey.*

An, L. T. H. and Tao, P. D. (1997). Solving a class of linearly constrained indefinite quadratic problems by d.c. algorithms, *Journal of Global Optimization*, **11**, 253–285. MR1469128

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. MR1473055

Friedman, J. H., Hastie, T. and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting, *Annals of Statistics*, **28**, 337–407. MR1790002

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Second Edition, Springer-Verlag: New York. MR1851606

Lee, Y., Lin, Y. and Wahba, G. (2004). Multicategory Support Vector Machines, theory and application to the classification of microarray data and satellite radiance data, *Journal of the American Statistical Association*, 99, 67–81. MR2054287

Lin, Y., Lee, Y. and Wahba, G. (2004). Support vector machines for classification in nonstandard situations, *Machine Learning*, **46**, 191–202.

Liu, Y. and Shen, X. (2006). Multicategory ψ-learning, *Journal of the American Statistical Association*, **101**, 500–509. MR2256170

Liu, Y., Shen, X. and Doss, H. (2005). Multicategory ψ-learning and support vector machines: computational tools, *Journal of Computational and Graphical Statistics*, **14**, 219–236. MR2137899

Liu, Y. and Wu, Y. (2006). Optimizing ψ-learning via mixed integer programming, *Statistica Sinica*, **16**, 441–457. MR2267244

Marron, J. S., Todd, M. and Ahn, J. (2007). Distance weighted discrimination, *Journal of the American Statistical Association*, **102**, 1267–1271. MR2412548

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

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. MR2656058

Shen, X., Tseng, G. C., Zhang, X. and Wong, W. H. (2003). On ψ-learning, *Journal of the American Statistical Association*, **98**, 724–734. MR2011686

Wu, Y. and Liu, Y. (2007). Robust truncated-hinge-loss support vector machines, *Journal of the American Statistical Association*, **102**, 974–983. MR2411659

Wu, Y., Zhang, H. H. and Liu, Y. (2010). Robust Model-free Multiclass Probability Estimation, *Journal of the American Statistical Association*, **105**, 424–436. MR2656060

Zhu, J. and Hastie, T. (2005). Kernel Logistic Regression and the Import Vector Machine, *Journal of Computational and Graphical Statistics*, **14**, 185–205. MR2137897

Zhu, J., Zou, H., Rosset, S. and Hastie, T. (2009). Multi-class adaboost, *Statistics and Its Interface*, **2**, 349–360. MR2540092

Yufeng Liu
Department of Statistics and Operations Research
Carolina Center for Genome Sciences
University of North Carolina
Chapel Hill, NC 27599
USA
E-mail address: yfliu@email.unc.edu

Yichao Wu
Department of Statistics
North Carolina State University
Raleigh, NC 27695
USA
E-mail address: wu@stat.ncsu.edu

Qinying He
Research Institute of Economics and Management
Southwestern University of Finance and Economics
Chengdu, Sichuan 610074
China
E-mail address: he@swufe.edu.cn