Statistica Sinica **16**(2006), 617-633

MULTI-CATEGORY SUPPORT VECTOR MACHINES, FEATURE SELECTION AND SOLUTION PATH

Lifeng Wang and Xiaotong Shen

University of Minnesota

Abstract: Support Vector Machines (SVMs) have proven to deliver high performance. However, problems remain with respect to feature selection in multi-category classification. In this article, we propose an algorithm to compute an entire regularization solution path for adaptive feature selection via L_1-norm penalized multi-category MSVM (L1MSVM). The advantages of this algorithm are three-fold. First, it permits fast computation for fine tuning, which yields accurate prediction. Second, it greatly reduces the cost of memory. This is especially important in genome classification, where a linear program with tens of thousands of variables has to be solved. Third, it yields a selection order in which the features can be examined sequentially. The performance of the proposed algorithm is examined in simulations and with data.

Key words and phrases: Genome classification, hinge loss, L_1-norm, penalty with, regularization.

1. Introduction

Support Vector Machines (SVMs), as powerful classification tools, have been widely used and proven effective in binary classification. For multi-category classification, several versions of L_2-norm multi-category SVMs (MSVMs) have been introduced, including Vapnik (1998), Weston and Watkins (1998), Bredensteiner and Bennett (1999), Guermuer (2002), Lee, Lin and Wahba (2004) and Liu and Shen (2005). Although MSVMs have been successful in a number of applications, they may not perform well when the number of features is much higher than that of observations. Recently, Wang and Shen (2005) proposed an L_1-norm MSVM (L1MSVM), which performs classification and feature selection simultaneously, and proves to be more effective in high-dimensional problems, especially in presence of many irrelevant features.

The performance of L1MSVM depends highly on the choice of a tuning parameter, denoted by s. Selection of the optimal s is typically performed via a grid search, which requires solving L1MSVM repeatedly at each value of s. This is computationally intensive. In some applications with a large number of features, in genome classification via microarrays for example, computation of L1MSVM

at a single value of s requires solving a linear program with tens of thousands of variables which, due to memory constraints, is computationally infeasible for a standard package (such as "lpsolve" in R). This situation is especially severe for k-category classification ($k \geq 3$), because the size of a linear program increases in the order of k^2. In this article, we address the computational issue by developing an efficient algorithm to identify an entire regularization path for all possible values of s simultaneously, which facilitates adaptive selection.

Our algorithm is motivated by Zhu, Hastie, Rosset and Tibshirani (2003), Efron, Hastie, Johnstone and Tibshirani (2004) and Hastie, Rosset, Tibshirani and Zhu (2004). Most relevant here is that Zhu, Hastie, Rosset and Tibshirani (2003), which showed that the solution path of the L_1-norm binary SVM is piecewise linear in its tuning parameter s, based on which an algorithm was developed to identify the linear segments of the solution path sequentially. It is observed that, as s increases from 0, the features may enter or leave a fitted model along the path. Furthermore, at most one variable enters or leaves the fitted model, and it happens only if the path reaches a joint, at which the direction of the path changes. Consequently, the segment following the joint can be identified by solving no more than p systems of linear equations, where p is the number of features.

A generalization from the binary case to the multi-category case is highly non-trivial. In binary classification, the classification rule is determined by one decision function, while in the k-category classification ($k \geq 3$), k decision functions need to be trained subject to a zero-sum constraint. Therefore, each feature is associated with k coefficients in k decision functions. Consequently, at a joint of the solution path, more than one coefficient may be added to the model simultaneously, which requires examining up to $p!/[(p-k+1)!(k-1)!]$ possibilities. In this complex situation, linear programs need to be solved, which makes computation much more difficult than the binary case. We tackle this problem by adding small perturbations to samples, which implicitly arranges the coefficients of features in a sequence such that at most one coefficient enters or leaves the model at a joint, thus bypassing the difficulty.

This article is organized as follows. Section 2 briefly introduces the methodology. Section 3 proposes an efficient algorithm that yields the entire solution path. Simulations and applications to data are presented in Section 4, followed by a summary.

2. Methodology

In the context of classification, a random vector $Z = (X, Y)$ from an unknown probability distribution is given, where input $X \in \mathbb{R}^p$ is a vector of p variables,

and output Y is categorical, indicating the class label. In k-category classification, y is usually coded as $\{1,\ldots,k\}$. A decision function vector $f=(f_1,\ldots,f_k)$ is introduced, f_c representing class c; $c=1,\ldots,k$. The resulting classification rule is $\Phi_f(x)=\arg\max_c f_c(x)$ that assigns a new input vector x to class c having the highest value $f_c(x)$. To avoid redundancy, a zero-sum constraint $\sum_{c=1}^k f_c=0$ is enforced. The goal is to find the f that minimizes the generalization error $E(I[Y\neq\Phi_f(x)])$ based on a training sample $z_i=(x_i,y_i)$, $i=1,\ldots,n$.

For linear problems, decision functions $f_c(x)=w_c^Tx+b_c$, $c=1,\ldots,k$, are linear. Here the column vectors $w_c=(w_{c,1},\ldots,w_{c,p})^T\in\mathbb{R}^p$ and $b_c\in\mathbb{R}^1$ are subject to zero-sum constraints $\sum_{c=1}^k w_c=\vec{0}$ and $\sum_{c=1}^k b_c=0$, with $\vec{0}$ a p-dimensional column vector. For nonlinear problems, $f_c(x)=\sum_{j=1}^q w_{c,j}h_j(x)+b_c$ based on a basis $\{h_j(x)\}_{j=1}^q$. This is equivalent to the linear formulation, with $H=(h_j(x_i))_{n\times q}$ being a design matrix. For simplicity, we use linear representations. According to Wang and Shen (2005), L1MSVM solves the problem:

$$\min_{w_c,b_c;c=1,\ldots,k}\sum_{i=1}^n L(f,z_i), \tag{2.1}$$

$$\text{subject to } \sum_{c=1}^k\|w_c\|_1\le s \quad\text{and}\quad \sum_c w_c=\vec{0};\sum_c b_c=0, \tag{2.2}$$

where $\sum_{c=1}^k\|w_c\|_1=\sum_{c=1}^k\sum_{j=1}^p|w_{c,j}|$ is an L_1-norm penalty, s is a tuning parameter, and $L(f,z_i)$ is the generalized hinge loss

$$L(f,z_i)=\sum_{c\neq y_i}[f_c(x_i)+1]_+, \tag{2.3}$$

with $[x]_+=xI(x>0)$. It is worth mentioning that (2.1)-(2.2) can be straightforwardly extended to other existing formulations of MSVMs, using different forms of $L(f,z)$, such as in Vapnik (1998), Weston and Watkins (1998), Bredensteiner and Bennett (1999), Guermuer (2002), Liu and Shen (2005) and Zhang (2004).

For any given value of s, (2.1)$-$(2.2) becomes the linear program

$$\min_{w_{c,j}^+,w_{c,j}^-,b_c^+,b_c^-,\xi_{i,c}}\sum_{\substack{1\le i\le n,1\le c\le k\\ c\neq y_i}}\xi_{i,c} \tag{2.4}$$

$$\text{subject to}\quad \sum_j(w_{c,j}^+-w_{c,j}^-)x_{ij}+(b_c^+-b_c^-)+1\le\xi_{i,c}; \tag{2.5}$$

$$(i,c)\in\{(i,c):c\neq y_i\}, \tag{2.6}$$

$$\sum_{c,j}(w_{c,j}^++w_{c,j}^-)\le s, \tag{2.7}$$

$$\sum_c (w_{c,j}^+ - w_{c,j}^-) = 0; \text{and} \sum_c (b_c^+ - b_c^-) = 0. \tag{2.8}$$

$$w_{c,j}^+, w_{c,j}^-, b_c^+, b_c^-, \xi_{i,c} \geq 0, \tag{2.9}$$

which is solved via a standard package, "lpsolve" in R. The solution of (2.1)−(2.2), $\hat{w}_{c,j}(s)$ and $\hat{b}_c(s)$, can be obtained as $\hat{w}_{c,j}(s) = \hat{w}_{c,j}^+ - \hat{w}_{c,j}^-$ and $\hat{b}_c(s) = \hat{b}_c^+ - \hat{b}_c^-$, $c = 1, \ldots, k$, $j = 1, \ldots, p$, where $\hat{w}_{c,j}^+, \hat{w}_{c,j}^-, \hat{b}_c^+$, and $\hat{b}_c^-$ are the solutions of (2.4)−(2.9). This yields $\hat{f}(x) = (\hat{w}_1^T x + \hat{b}_1, \ldots, \hat{w}_k^T x + \hat{b}_k)$ and the corresponding $\Phi(x) = \arg\max_c (\hat{w}_c x + \hat{b}_c)$.

L1MSVM can be cast into the framework of regularization as follows.

$$\min_{w_c, b_c; c=1,\ldots,k} \sum_{i=1}^n L(f, z_i) + \lambda \sum_{c=1}^k \|w_c\|_1, \quad \text{s.t.} \quad \sum_c w_c = \vec{0}; \sum_c b_c = 0, \tag{2.10}$$

where λ is a nonnegative regularization parameter. When $\sum_{c=1}^k \|w_c\|_1$ is replaced by its L_2-norm counterpart $\sum_{c=1}^k \|w_c\|_2^2$, (2.10) is equivalent to MSVM in Lee, Lin and Wahba (2004). As discussed in Wang and Shen (2005), the L_1-penalty shrinks the estimated coefficients and coerces some small coefficients to be exactly zero. For sufficiently large λ, or sufficiently small s, many estimated coefficients $\hat{w}_{c,j}$ become exactly zero, which enables L1MSVM to perform feature selection within the framework of classification. In addition, the L_1-penalty yields consistency of L1MSVM in presence of many irrelevant features (Wang and Shen (2005)), which is particularly useful for problems where the number of features exceeds that of observations.

2.3. Computational issues

A key to the performance of L1MSVM is the choice of tuning parameter s, which controls the tradeoff between training and generalization, and determines the number of features used in classification. To obtain a classifier with good generalization performance, adaptive selection of s is necessary. In application, selection is usually performed based on cross-validation applied to different values of s to seek the s yielding best performance. In this process, (2.4)−(2.9) are solved repeatedly with respect to s. This is computationally intensive, particularly in high-dimensional problems.

The memory required for computation is also of concern. For any fixed s, L1MSVM solves (2.4)−(2.9), which is a linear program of dimension $2(p+1)k + n(k-1)$. When p exceeds thousands, a standard package such as "lpsolve" in R, fails to allocate memory required to store the constraint matrix. Therefore, a straightforward computation of L1MSVM is not feasible without an efficient algorithm.

Motivated by Zhu, Hastie, Rosset and Tibshirani (2003) and Hastie, Rosset, Tibshirani and Zhu (2004), we develop an efficient algorithm that constructs an entire path of solution $\hat{w}_{c,j}(s)$ of (2.1)−(2.2) as a function of s. This permits rapid computation of the adaptive selection of s, reducing the memory requirement and making computation of extremely high-dimensional linear program feasible, since L1MSVM selects no more than nk features and, consequently, at most nk variables need to be stored.

3. Solution Path

The idea for computing the whole solution path originated from parametric linear programming in operations research. A general parametric linear program can be written as

$$\begin{aligned} &\min_{x} \quad c^T x \\ &\text{subject to } \boldsymbol{A}x = b + \theta d \text{ and } x \geq 0. \end{aligned} \tag{3.1}$$

It can be shown that the optimal solution $\hat{x}(\theta)$ is piecewise linear in θ and can be determined by tracking the basis B, consisting of those indices of the nonzero variables in x, c.f., Bertsimas and Tsitsiklis (1997).

In the context of L1MSVM, we have a similar problem with a tuning parameter s in place of θ in (3.1). For simplicity, write w_c and b_c in the form of a $k \times (p+1)$ matrix w, where $w_{c,j}$, $j = 1, \ldots, p$, is the jth entry of $\vec{w}_c$ and $w_{c,0} = b_c$. The input vector x_i is now expanded to $(1, x_i)$, with 1 corresponding to $w_{c,0} = b_c$. A key technique in our algorithm is to add a small perturbation ϵ_i to the constant term x_{i0} such that $x_{i0} = 1 + \epsilon_i$, where ϵ_i depends on i. In implementations, we take $\epsilon_i = \sigma u_i$, $i = 1, \ldots, n$, with u_i generated independently from a uniform distribution over $[0, 1]$, and $\sigma = 10^{-5}$. When σ is sufficiently small, the perturbation has little effect on $\hat{w}(s)$, while making our algorithm efficient and numerically stable. The effect of this perturbation is twofold.

- The perturbation yields a unique solution at $s = 0$. When $s = 0$, (2.1)−(2.2) reduce to

$$\min_{\sum_c w_{c,0}=0} \sum_{c=1}^{k} \Big(\sum_{c \neq y_i} [x_{i0} w_{c,0}(0) + 1]_+ \Big), \tag{3.2}$$

  with $x_{i0} = 1 + \epsilon_i$. Without the perturbation, $x_{i0} = 1$, $i = 1, \ldots, n$, and (3.2) becomes

$$\min_{\sum_c w_{c,0}=0} \sum_{c=1}^{k} (n - n_c)[w_{c,0}(0) + 1]_+, \tag{3.3}$$

  where n_c is the number of samples in class c, $c = 1, \ldots, k$. This may not yield a unique solution. For instance, consider a situation where there exist

two classes j and j' such that $n_j = n_{j'} = \arg\max_c n_c$. It can be verified that any $w_{c,0}$, $c = 1, \ldots, k$, satisfying $w_{c,0} = -1$, $c \neq j, j'$, $w_{j,0}, w_{j',0} \geq -1$ and $w_{j,0} + w_{j',0} = k - 2$, minimize (3.3).

- The perturbation is a necessary condition for Theorems 1 and 2, which lead to efficient initialization of the solution path. In our algorithm, the initialization starts from $s = 0$ as in (3.2), which is a linear program in k-dimensional Euclidian space. In contrast, without this perturbation, the path in a small neighborhood of $s = 0$ seems intractable; see Remark A for details. In Zhu, Hastie, Rosset and Tibshirani (2003), this difficulty is bypassed by initializing the path at $s = \Delta s > 0$, which requires solving a linear program of order p. This becomes a more difficult problem in the high-dimensional case.

Under the perturbation, the solution of (2.1)−(2.2), $\hat{w}(s)$, is uniquely defined at each value of s. This results in a piecewise linear $\hat{w}_{c,j}(s)$ in s, as shown in Theorem 1.

Theorem 1. *Rewrite* $\sum_{i=1}^n L(f(x_i), y_i)$ *as* $l(w)$ *and define* t^* *as* $\inf\{\sum_{(c,j):j\neq 0} |w^*_{c,j}| :$ w^* *minimizes* $l(w)$*, subject to* $\sum_c w_{c,j} = 0$*;* $j = 0, \ldots, p\}$*. Then the solution of* (2.1) *is unique with probability* 1 *when* $s < t^*$*, provided the distribution of* $x \in \mathbb{R}^{p+1}$ *is absolutely continuous with respect to Lebesgue Measure. Furthermore,* $\hat{w}_{c,j}(s)$ *is piecewise linear in* s *for* $c = 1, \ldots, k$ *and* $j = 0, \ldots, p$*.*

The proof is given in the Appendix. For an illustration, we provide a simple example and display the solution paths in Figure 1.

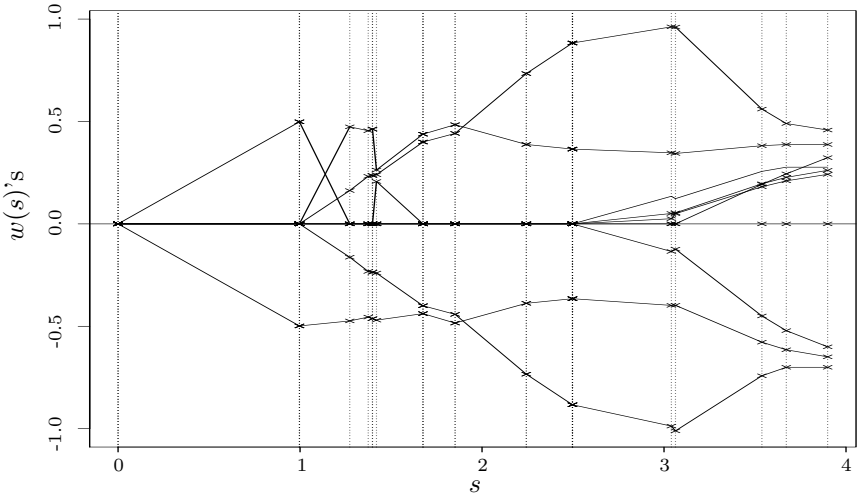

Figure 1. Solution paths of $\hat{w}_{c,j}(s)$ for a three-class classification problem with three-dimensional input, trained on 15 samples. Nine paths $\hat{w}_{c,j}(s)$, $c = 1, 2, 3$, $j = 1, 2, 3$ are displayed, with asterisks indicating the joints of $\hat{w}_{c,j}(s)$. Any segment between two adjacent joints is linear.

3.1. Algorithm

The basic idea of identifying the solution path of $\hat{w}(s)$ for all s is to locate joints of $\hat{w}_{c,j}(s)$, and to determine the corresponding right derivative of $\hat{w}_{c,j}(s)$, denoted as $d_{c,j}(s)$, at each joint. Suppose that $\hat{w}_{c,j}(s_l)$ and $d_{c,j}(s_l)$ are determined at the lth joint s_l. Then $\hat{w}_{c,j}(s)$ can be obtained as $\hat{w}_{c,j}(s) = \hat{w}_{c,j}(s_l) + d_{c,j}(s_l)(s - s_l)$ for all $s_l < s < s_{l+1}$, due to piecewise linearity. This greatly reduces computational cost, that is, there is no need to solve (2.1)−(2.2) at each s, as long as we determine $\hat{w}_{c,j}(s_l)$ and $d_{c,j}(s_l)$.

Before stating our algorithm, we first describe the basic ideas behind each step. Our algorithm keeps track of the following sets:

- $\mathcal{A}(\hat{w}(s)) = \{(c, j) : \hat{w}_{c,j}(s) \neq 0; c = 1, \ldots, k; j = 0, 1, \ldots, p\}$, which collects the non-zero coefficients;
- $\mathcal{J}(\hat{w}(s)) = \{j : \hat{w}_{c,j}(s) \neq 0 \text{ for some } c\}$, which consists of the indices j from $\{0, \ldots, p\}$ such that the zero-sum constraint $\sum_c \hat{w}_{c,j} = 0$ is active;
- $\mathcal{E}(\hat{w}(s)) = \{(i, c) : \sum_{j=0}^{p} \hat{w}_{c,j}(s) x_{ij} + 1 = 0, c \neq y_i\}$, so $(i, c) \in \mathcal{E}(\hat{w}(s))$ implies that x_i is on the margin $f_c + 1 = 0$.

Theorem 2. *For any $0 \leq s < t^*$, $|\mathcal{A}(\hat{w}(s))| = |\mathcal{E}(\hat{w}(s))| + |\mathcal{J}(\hat{w}(s))| + 2$ if $\hat{w}(s)$ is at a joint; $|\mathcal{A}(\hat{w}(s))| = |\mathcal{E}(\hat{w}(s))| + |\mathcal{J}(\hat{w}(s))| + 1$ otherwise, $|\mathcal{S}|$ the cardinality of set $\mathcal{S}$.*

As given in the proofs of Theorems 1 and 2, for s between two joints, $d_{c,j}(s)$ is 0 for $(c, j) \in \mathcal{A}(\hat{w}(s))^c$, and the non-zero right derivatives $d_{c,j}(s)$, $(c, j) \in \mathcal{A}(\hat{w}(s))$, are the unique solution of a system of equation.

$$\begin{cases} \sum\limits_{j:(c,j)\in\mathcal{A}(\hat{w}(s))} x_{ij} d_{c,j}(s) = 0, & (i, c) \in \mathcal{E}(\hat{w}(s)), \\ \sum\limits_{c:(c,j)\in\mathcal{A}(\hat{w}(s))} d_{c,j}(s) = 0, & j \in \mathcal{J}(\hat{w}(s)), \\ \sum\limits_{(c,j)\in\mathcal{A}(\hat{w}(s));j\neq 0} \operatorname{sign}(\hat{w}_{c,j}(s)) d_{c,j}(s) = 1, & \end{cases}$$

denoted as $M(s)d = r(s)$ for simplicity. Evidently, $M(s)$ and $r(s)$ are determined by $\mathcal{A}(\hat{w}(s))$, $\mathcal{J}(\hat{w}(s))$ and $\mathcal{E}(\hat{w}(s))$. The sets $\mathcal{A}(\hat{w}(s))$, $\mathcal{J}(\hat{w}(s))$ and $\mathcal{E}(\hat{w}(s))$, as well as $M(s)$, $r(s)$ and the solution $d_{c,j}(s)$, remain unchanged between two points, which can be used to determine the lth joint s_l and the following joint s_{l+1}.

As a first step in our algorithm, we start at the first joint $s = 0$ by solving (3.2) and increase s toward t^*. At the second step, as s increases from the lth joint s_l, $\mathcal{A}(\hat{w}(s_l))$, $\mathcal{J}(\hat{w}(s_l))$ and $\mathcal{E}(\hat{w}(s_l))$ need to be updated in view of the formula in Theorem 2. Two kinds of actions can take place: (a) adding one or two elements (c, j) to $\mathcal{A}(\hat{w}(s_l))$; (b) removing one element (i, c) from $\mathcal{E}(\hat{w}(s_l))$. Note that different actions result in different $\mathcal{A}(\hat{w}(s))$, $\mathcal{J}(\hat{w}(s))$ and $\mathcal{E}(\hat{w}(s))$, each of which is associated with a system of equations $M(s)d = r(s)$ that determines a

possible derivative $d_{c,j}(s)$. Then, we examine all possible derivatives and choose the one that leads to the steepest descent rate of the cost function $l(\hat{w}(s))$ with respect to s. At the third step, the derivatives $d_{c,j}(s)$; $c = 1, \ldots, k$, $j = 0, \ldots, p$, remain unchanged until one of $\mathcal{A}(\hat{w}(s))$, $\mathcal{J}(\hat{w}(s))$ and $\mathcal{E}(\hat{w}(s))$ changes. This implies that one of the following events occurs: (a) one or two elements (c, j) leave $\mathcal{A}(\hat{w}(s))$, i.e., $\hat{w}_{c,j}(s)$ becomes zero for (c, j); (b) one element (i, c) enters $\mathcal{E}(\hat{w}(s))$, i.e., $\{\sum_{j=0}^{p} w_{c,j}(s)x_{ij} + 1\}$ becomes zero for (i, c). The joint s_{l+1} is obtained by identifying a value of s, at which such an event first occurs. This whole progress is repeated until s reaches t^*.

Now we present the algorithm that computes the entire solution path.

Step 1. Initialize $\hat{w}(s)$ at $s = 0$.

Set $\hat{w}_{c,j}(0) = 0$; $j = 1, \ldots, p$. Obtain $\hat{w}_{c,0}(0)$ by solving

$$\min_{\sum_c w_{c,0}=0} \sum_{c=1}^{k} \Big(\sum_{c \neq y_i} [x_{i0} w_{c,0}(0) + 1]_+ \Big).$$

Then $\mathcal{A}(\hat{w}(0)) = \{(c, 0) : c = 1, \ldots, k\}$, $\mathcal{J}(\hat{w}(0)) = \{0\}$, and $\mathcal{E}(\hat{w}(0))$ can be obtained after we obtain $\hat{w}_{c,0}$.

Step 2. Given $\hat{w}(s_l)$ at the lth joint s_l, compute the right derivative $d_{c,j}(s)$ of $\hat{w}_{c,j}(s)$ for $s_l \leq s < s_{l+1}$.

Because $|\mathcal{A}(\hat{w}(s_l))| = |\mathcal{E}(\hat{w}(s_l))| + |\mathcal{J}(\hat{w}(s_l))| + 2$, and $|\mathcal{A}(\hat{w}(s))| = |\mathcal{E}(\hat{w}(s))| + |\mathcal{J}(\hat{w}(s))| + 1$ for $s_l < s < s_{l+1}$, one of the three events must occur as s increases from s_{l+1}: (1) $|\mathcal{A}(\hat{w}(s))|$ increases by 1; (2) $|\mathcal{A}(\hat{w}(s_l))|$ increases by 2 and $|\mathcal{J}(\hat{w}(s_l))|$ increases by 1; (3) $|\mathcal{E}(\hat{w}(s_l))|$ decreases by 1. We examine these cases.

(a) Add one element to $\mathcal{A}(\hat{w}(s_l))$.

For each $(c', j') \in \mathcal{A}(\hat{w}(s_l))^c$ such that $j' \in \mathcal{J}(\hat{w}(s_l))$, update $\mathcal{A}(\hat{w}(s)) = \mathcal{A}(\hat{w}(s_l)) \cup \{(c', j')\}$, and solve the following system of equations with respect to $\{d_{c,j} : (c, j) \in \mathcal{A}(\hat{w}(s_l))\}$,

$$\begin{cases} \displaystyle\sum_{j:(c,j)\in\mathcal{A}(\hat{w}(s))} x_{ij} d_{c,j} &= 0, \quad (i, c) \in \mathcal{E}(\hat{w}(s_l)) \\ \displaystyle\sum_{c:(c,j)\in\mathcal{A}(\hat{w}(s))} d_{c,j} &= 0, \quad j \in \mathcal{J}(\hat{w}(s_l)) \\ \displaystyle\sum_{(c,j)\in\mathcal{A}(\hat{w}(s_l)), j\neq 0} \operatorname{sign}(\hat{w}_{c,j}(s_l)) d_{c,j} + |d_{c'j'}| &= 1. \end{cases}$$

Then, compute

$$\frac{\Delta l(\hat{w}(s))}{\Delta s} = \sum_{(i,c)\in\mathcal{E}_+(\hat{w}(s_l))} \Big(\sum_{j:(c,j)\in\mathcal{A}(\hat{w}(s))} x_{ij} d_{c,j} \Big),$$

where $\mathcal{E}_+(\hat{w}(s_l)) = \{(i, c) : \sum_{j=0}^{p} \hat{w}_{c,j}(s_l)x_{ij} + 1 > 0, c \neq y_i\}$.

(b) Add two elements to $\mathcal{A}(\hat{w}(s_l))$.
For each pair of $(c_1', j'), (c_2', j') \in \mathcal{A}(\hat{w}(s_l))^c$ such that $j' \in \mathcal{J}(\hat{w}(s_l))^c$, update $\mathcal{A}(\hat{w}(s)) = \mathcal{A}(\hat{w}(s_l)) \cup \{(c_1', j')\} \cup \{(c_2', j')\}$, $\mathcal{J}(\hat{w}(s)) = \mathcal{J}(\hat{w}(s_l)) \cup \{j'\}$, and solve

$$\begin{cases} \sum_{j:(c,j)\in\mathcal{A}(\hat{w}(s))} x_{ij} d_{c,j} = 0, \\ \qquad (i,c) \in \mathcal{E}(\hat{w}(s_l)), \\ \sum_{c:(c,j)\in\mathcal{A}(\hat{w}(s))} d_{c,j} = 0, \\ \qquad j \in \mathcal{J}(\hat{w}(s)), \\ \sum_{(c,j)\in\mathcal{A}(\hat{w}(s_l)), j\neq 0} \operatorname{sign}(\hat{w}_{c,j}(s)) d_{c,j} + |d_{c_1',j'}| + |d_{c_2',j'}| = 1 \end{cases}$$

with respect to $\{d_{c,j} : (c,j) \in \mathcal{A}(\hat{w}(s))\}$. Then, compute

$$\frac{\Delta l(\hat{w}(s))}{\Delta s} = \sum_{(i,c)\in\mathcal{E}_+(\hat{w}(s_l))} \Big(\sum_{j:(c,j)\in\mathcal{A}(\hat{w}(s))} x_{ij} d_{c,j} \Big).$$

(c) Remove an element from $\mathcal{E}(\hat{w}(s_l))$.
For each $(i', c') \in \mathcal{E}(\hat{w}(s_l))$, update $\mathcal{E}(\hat{w}(s)) = \mathcal{E}(\hat{w}(s_l)) \setminus \{(i', c')\}$. Solve

$$\begin{cases} \sum_{j:(c,j)\in\mathcal{A}(\hat{w}(s_l))} x_{ij} d_{c,j} = 0, \quad (i,c) \in \mathcal{E}(\hat{w}(s)), \\ \sum_{c:(c,j)\in\mathcal{A}(\hat{w}(s_l))} d_{c,j} = 0, \quad j \in \mathcal{J}, \\ \sum_{(c,j)\in\mathcal{A}(\hat{w}(s_l)), j\neq 0} \operatorname{sign}(\hat{w}_{c,j}(s)) d_{c,j} = 1 \end{cases}$$

with respect to $\{d_{c,j} : (c,j) \in \mathcal{A}(\hat{w}(s_l))\}$. Then, compute

$$\frac{\Delta l(\hat{w}(s))}{\Delta s} = \sum_{(i,c)\in\mathcal{E}_+(\hat{w}(s_l))} \Big(\sum_{j:(c,j)\in\mathcal{A}(\hat{w}(s_l))} x_{ij} d_{c,j} \Big) + \Big[\sum_{j:(c',j)\in\mathcal{A}(\hat{w}(s_l))} x_{i'j} d_{c',j} \Big]_+.$$

If the minimum of $\Delta l(\hat{w}(s))/\Delta s$ in (a), (b) and (c) is less than 0, the corresponding solution $d_{c,j}$ is the right derivative of $\hat{w}_{c,j}(s)$ for $(c,j) \in \mathcal{A}(\hat{w}(s))$ at current joint s_l, and $d_{c,j} = 0$ is the right derivative of $\hat{w}_{c,j}(s)$ for $(c,j) \in \mathcal{A}(\hat{w}(s))^c$. Otherwise, the algorithm terminates.

Step 3. Given the current right derivative $d_{c,j}(s)$, $\mathcal{A}(\hat{w}(s))$, $\mathcal{J}(\hat{w}(s))$ and $\mathcal{E}(\hat{w}(s))$, compute the next joint s_{l+1}.

When s hits s_{l+1}, either $\mathcal{A}(\hat{w}(s))$ decreases by 1 or $\mathcal{E}(\hat{w}(s))$ increases by 1.

(a) One $(c,j) \in \mathcal{A}(\hat{w}(s))$ leaves $\mathcal{A}(\hat{w}(s))$.
For each $(c,j) \in \mathcal{A}(\hat{w}(s))$, $j \neq 0$, compute $\Delta s = -\hat{w}_{c,j}(s_l)/d_{c,j}$.

(b) One $(i,c) \in \mathcal{E}(\hat{w}(s))^c$ enters $\mathcal{E}(\hat{w}(s))$.

For each $(i,c) \in \mathcal{E}(\hat{w}(s))^c$, compute $\Delta s = -\sum_j \hat{w}_{c,j}(s_l)x_{ij} + 1/\sum_j d_{c,j}x_{ij}$.

Compare all Δs's in (a) and (b). Find the minimum of all positive Δs's. Then the next joint is $s_{l+1} = s_l + \Delta s$, we update $\hat{w}_{c,j}(s_{l+1}) = \hat{w}_{c,j}(s_l) + \Delta s \cdot d_{c,j}$, as well as the corresponding $\mathcal{A}(\hat{w}(s_{l+1}))$, $\mathcal{J}(\hat{w}(s_{l+1}))$ and $\mathcal{E}(\hat{w}(s_{l+1}))$.

Step 4. Iterate Steps 2 and 3 until the algorithm terminates.

Remark A. The perturbation to x_{i0} is important. Without this perturbation, one solution is obtained for the initial step $s = 0$ with $\hat{w}_{c,0}(0) = k - 1$ for $c = \arg\max_c n_c$ and $\hat{w}_{c,0}(0) = -1$ for $c \neq \arg\max_c n_c$, where n_c is the number of instances in class c. This results in $|\mathcal{A}(\hat{w}(0))| = k$ and $|\mathcal{E}(\hat{w}(0))| = n(k-2) + \max_c n_c$. Therefore the equations in Theorem 2 no longer hold, which implies that multiple elements may enter $\mathcal{A}(\hat{w}(0))$ simultaneously, as s increases. In this case, the path seems intractable, and a linear program needs to be solved to determine $d_{c,j}$, and this is computationally intensive.

Remark B. As s increases, $\hat{w}_{c,j}(s)$ becomes nonzero when (c,j) enters $\mathcal{A}(\hat{w}(s))$ in a certain order. This permits stepwise feature selection, where the order of features indicates their importance in terms of contributions to the classifier. A feature x_j with $\hat{w}_{c,j}(s) = 0$ for all $c = 1, \ldots, k$, is considered to be redundant to the classification.

3.2. Computational complexity

At joint s_l, a system of equations $M(s_l)u = r(s_l)$ of size $m_l = |\mathcal{A}(\hat{w}(s_l))|$ need to be solved, which has a complexity of order $O(m_l^3)$. In our algorithm, however, this complexity reduces to $O(m_l^2)$ because $M(s_l)$ changes only slightly with respect to iteration, and the Woodbury formula is applied to update the inverse matrix $M(s_l)^{-1}$ at each joint. Consequently, the computational cost at each iteration is of order $O(pkm_l^2)$. Experience suggests that the number of joints on the path is roughly $O(\min(n,p)k)$, which yields a complexity of $O(\min(n,p)pk^2m^2)$ for our algorithm, where m is the average size of $M(s_l)$ and no larger than $\min(n,p)k$.

4. Numerical Studies

4.1. Simulations

In this section, we illustrate our algorithm with a three-class 70-dimensional classification problem. In this example, 60 training samples are generated as follows. First, sample $(u_1, \ldots, u_{70})$ from the 70-dimensional standard normal distribution. Second, randomly assign 60 instances to the three classes, with 20 in each one. Third, perform linear transformation: $x_j = u_j + a_j$; $j = 1, 2$ and

$x_j = u_j$; $j = 3, \ldots, 70$, with $(a_1, a_2) = (\sqrt{2}, \sqrt{2})$, $(-\sqrt{2}, -\sqrt{2})$ and $(\sqrt{2}, -\sqrt{2})$ for classes 1-3, respectively. Evidently, the Bayes decision function only depends on x_1 and x_2, while the other features are irrelevant. For the purpose of comparison, we compute the optimal Bayes risk of approximately 0.104.

In this numerical example, we select the optimal tuning parameter by five-fold cross-validation. First, we randomly assign 60 training samples into five groups of equal size. Next, we use four groups for training and the other one for testing, and apply our algorithm to construct the whole solution path. This procedure is repeated five times, and the averaged cross-validation error is then obtained as a function of s. Then, the solution path is constructed for the whole training data set and the test error is obtained over 15,000 testing samples generated with the same scheme for all s. Note that feature x_j is excluded from the decision functions if and only if $\hat{w}_{c,j}(s)$, $c = 1, 2, 3$, are all zero. To illustrate the effect of s on feature selection, we use functions $v_j(s) = \sum_{c=1}^{3} |\hat{w}_{c,j}(s)|$, $j = 1, \ldots, k$, where a nonzero $v_j(s)$ implies feature x_j is selected. The results are displayed in Figure 2. As shown in Figure 2, the cross validation error achieves its minimum of 0.117 at $s = 2.516$, which is used for the final prediction model. At $s = 2.516$, L1MSVM achieves a test error of 0.137, which is fairly close to the Bayes risk, given a relatively low sample size $n = 60$ and a high dimension $p = 70$. Furthermore, at $s = 2.516$, only x_1 and x_2 are included in the final prediction model, which coincides with the truth.

4.2. Application to genome classification

An important application of L1MSVM is to cancer genome classification via microarrays. Consider a benchmark example concerning the small round blue cell tumors (SRBCTs) of childhood. This example consists of 63 training samples in 4 classes (EMS, BL, NB and RMS), and 20 test samples with each sample a vector of 2,308 genes. We apply our algorithm, in addition to five-fold cross-validation, for selecting the tuning parameter. As shown in Figure 3, the five-fold cross-validation error decreases as s increases, and is 0 at $s = 11.373$. Therefore, $s = 11.373$ is selected as the optimal tuning parameter. This yields a perfect classification (no test error) and a sparse representation, with only 83 relevant genes selected in the final prediction model.

This example was previously analyzed in Khan et al. (2001) using an artificial neural network (ANN) and in Lee, Lin and Wahba (2004) using the standard MSVM. Although these two methods achieve zero test error, they depend heavily on preprocessing to select important genes. For instance, without pre-screening genes, MSVM may have three errors as reported in Lee, Lin and Wahba (2004). In contrast, our method has the advantage of automatic gene selection without a requirement of preprocessing.

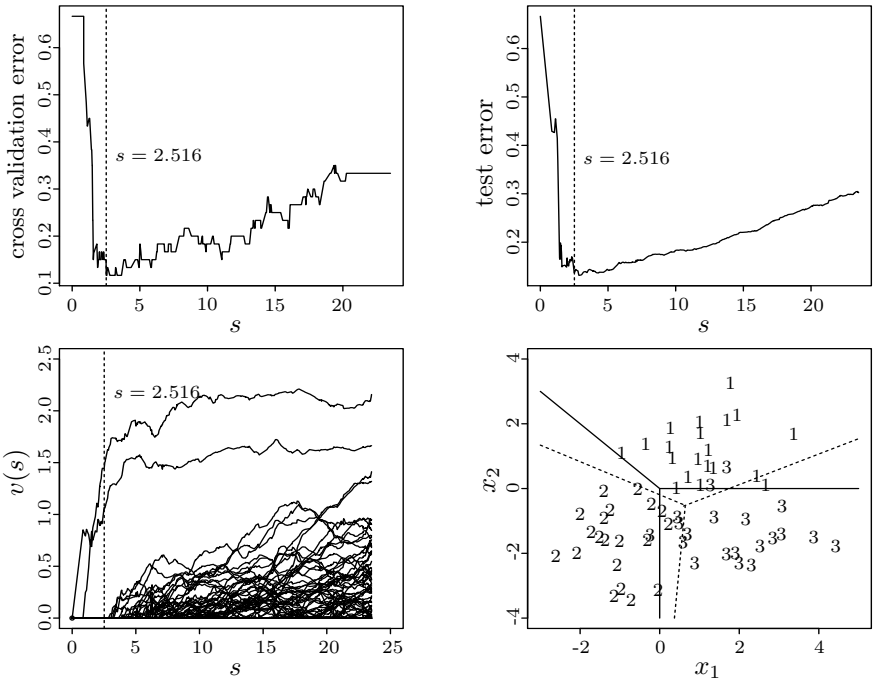

Figure 2. *Top Left Panel*: Five-fold cross-validation error as a function of s. *Top Right Panel*: Test error (over 10,000 test samples) as a function of s. *Lower Left Panel*: $v_j(s)$; $j = 1, \ldots, 70$, as functions of s, where $v_j(s) = \sum_{c=1}^{3} |\hat{w}_{c,j}(s)|$. The upper two paths are $v_1(s)$ and $v_2(s)$, corresponding to features x_1 and x_2. *Lower Right Panel*: The true decision boundaries (solid lines) and L1MSVM decision boundaries (dashed lines) in the example with $s = 2.516$, each depending only on x_1 and x_2.

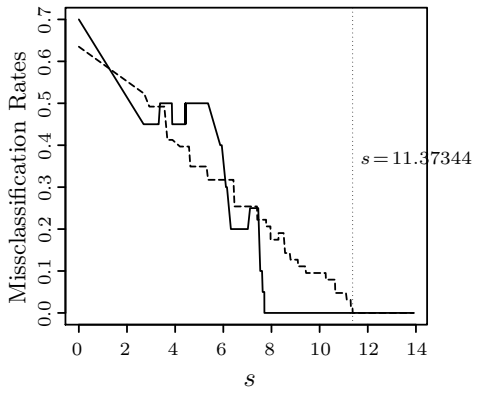

Figure 3. Test error rate (solid line) and five-fold cross-validation error rate (dashed line) against tuning parameter s. The cross-validation error is 0 at $s = 11.373$, the optimal s.

5. Conclusions

This article is devoted to the computational development of L1MSVM. To meet the computational challenge of L1MSVM in high-dimensional settings, an efficient algorithm is proposed, that yields an entire solution path. This algorithm reduces the computational cost and storage involved in adaptive selection of the tuning parameter based on a piecewise linearity property.

Although our algorithm performs well in simulations and some applications, further investigation is needed to make it applicable to more challenging problems, such as the multi-class cancer diagnosis problem with 14 classes and 16,063 genes discussed in Ramaswamy et al. (2001).

Acknowledgement

This research is supported by NSF grant IIS-0328802. We thank the reviewers for helpful comments and suggestions.

Appendix

Proof of Theorem 1. We need some notations. Let $\mathcal{A}(w) = \{(c,j) : w_{c,j} \neq 0; c = 1,\ldots,k; j = 0,1,\ldots,p\}$, $\mathcal{E}(w) = \{(i,c) : \sum_{j=0}^{p} w_{c,j}x_{ij} + 1 = 0, c \neq y_i\}$, $\mathcal{E}_+(w) = \{(i,c) : \sum_{j=0}^{p} w_{c,j}x_{ij} + 1 > 0, c \neq y_i\}$, $\mathcal{E}_-(w) = \{(i,c) : \sum_{j=0}^{p} w_{c,j}x_{ij} + 1 < 0, c \neq y_i\}$ and $\mathcal{J}(w) = \{j : w_{c,j} \neq 0 \text{ for some } c\}$.

Uniqueness. Suppose that $s < t^*$. Then a minimizer w^* of (2.1)−(2.2) must be located on the boundary $\sum_{(c,j):j\neq 0} |w_{c,j}| = s$. Otherwise, $\sum_{(c,j):j\neq 0} |w_{c,j}| < s$, implying that w^* is a global minimizer of $l(w)$ by convexity of $l(w)$. This contradicts the definition of t^*.

Before proving uniqueness, we first prove that the nonzero components of w^*, $w^*_{c,j}$; $(c,j) \in \mathcal{A}(w^*)$, are uniquely determined by the following system of equations involving $\{u_{c,j} : (c,j) \in \mathcal{A}(w^*)\}$.

$$\begin{cases} \displaystyle\sum_{j:(c,j)\in\mathcal{A}(w^*)} x_{ij}u_{c,j} = -1, & (i,c) \in \mathcal{E}(w^*), \\ \displaystyle\sum_{c:(c,j)\in\mathcal{A}(w^*)} u_{c,j} = 0, & j \in \mathcal{J}(w^*), \\ \displaystyle\sum_{(c,j)\in\mathcal{A}(w^*),j\neq 0} \operatorname{sign}(w^*_{c,j})u_{c,j} = s. \end{cases} \tag{A.1}$$

For simplicity, write (A.1) as $M(w^*)u = r(w^*)$, where u is a vector of unknown variables, $M(w^*)$ and $r(w^*)$ are the coefficient matrix and vector, respectively, determined by $\mathcal{A}(w^*)$, $\mathcal{J}(w^*)$ and s.

By construction of $\mathcal{E}(w^*)$ and $\mathcal{J}(w^*)$, $\{w^*_{c,j} : (c,j) \in \mathcal{A}(w^*)\}$ is a solution of (A.1). To prove $\{w^*_{c,j} : (c,j) \in \mathcal{A}(w^*)\}$ is a unique solution of $M(w^*)u = r(w^*)$, it suffices to show $M(w^*)u = 0$ has a unique solution $\vec{0}$.

Let u^* be any solution of $M(w^*)u = 0$. Define d as a $k \times (p+1)$ matrix with $d_{c,j} = u^*_{c,j}$ for $(c,j) \in \mathcal{A}(w^*)$, and $d_{c,j} = 0$ otherwise. Note that, for sufficiently small $\epsilon > 0$, $l(w^*+\epsilon d) - l(w^*) = \epsilon \sum_{(i,c)\in\mathcal{E}_+(w^*)} [\sum_{j:(c,j)\in\mathcal{A}(w^*)} x_{ij} d_{c,j}]$ and $l(w^* - \epsilon d) - l(w^*) = -\epsilon \sum_{(i,c)\in\mathcal{E}_+(w^*)} [\sum_{j:(c,j)\in\mathcal{A}(w^*)} x_{ij} d_{c,j}]$, which implies $l(w^*+\epsilon d) - l(w^*) = -(l(w^* - \epsilon d) - l(w^*))$. Since w^* is a minimizer of (2.1)−(2.2), $l(w^*+\epsilon d) - l(w^*) = -(l(w^* - \epsilon d) - l(w^*)) = 0$, i.e., $\sum_{(i,c)\in\mathcal{E}_+(w^*)} [\sum_{j:(c,j)\in\mathcal{A}(w^*)} x_{ij} d_{c,j}] = 0$. This implies that any solution of $M(w^*)u = 0$ satisfies $\sum_{(i,c)\in\mathcal{E}_+(w^*)} [\sum_{j:(c,j)\in\mathcal{A}(w^*)} x_{ij} u_{c,j}] = 0$. Note that the coefficients of $u_{c,j}$ in $\sum_{(i,c)\in\mathcal{E}_+(w^*)} [\sum_{j:(c,j)\in\mathcal{A}(w^*)} x_{ij} u_{c,j}] = 0$ are continuous random variables. This implies that $u = \vec{0}$ with probability 1. Therefore, $M(w^*)u = 0$ has a unique solution $u = \vec{0}$, implying that $M(w^*)u = r(w^*)$ has a unique solution.

Now we are ready to prove uniqueness of the minimizer of (2.1). Suppose that there exist two minimizers $w_{(j)}$, $j = 1, 2$. Let $d = w_{(1)} - w_{(2)}$. Consider $\tilde{w} = (w_{(1)} + w_{(2)})/2$ and $\tilde{w} + \epsilon d$ with $\epsilon > 0$ sufficiently small. Note that both $\tilde{w}$ and $\tilde{w} + \epsilon d$ are convex combinations of $w_{(1)}$ and $w_{(2)}$. By convexity of $l(w)$, $\tilde{w}$ and $\tilde{w} + \epsilon d$ are also minimizer of (2.1), which implies that they are on the boundary, i.e., $\sum_{(c,j):j\neq 0} |\tilde{w}_{c,j}| = s$ and $\sum_{(c,j):j\neq 0} |\tilde{w}_{c,j} + \epsilon d_{c,j}| = s$. It can be verified that, for sufficiently small ϵ, $\mathcal{A}(\tilde{w}) = \mathcal{A}(\tilde{w} + \epsilon d)$ and $\mathcal{E}(\tilde{w}) \supset \mathcal{E}(\tilde{w} + \epsilon d)$. Thus, the equations $M(\tilde{w} + \epsilon d)u = r(\tilde{w} + \epsilon d)$ also appear in $M(\tilde{w})u = r(\tilde{w})$, implying that the solution of $M(\tilde{w})u = r(\tilde{w})$ is a solution of $M(\tilde{w} + \epsilon d)u = r(\tilde{w} + \epsilon d)$. This contradicts the fact that $M(\tilde{w} + \epsilon d)u = r(\tilde{w} + \epsilon d)$ has a unique solution.

Continuity. Denote by $\hat{w}(s)$ the unique minimizer of (2.1)−(2.2). It suffices to show $\hat{w}(s)$ is both right continuous and left continuous at any point s_0.

If $\hat{w}(s)$ is not right continuous, there exists a sequence $s_n \downarrow s_0$ such that $\lim_{n\to\infty} \hat{w}(s_n) = w^* \neq \hat{w}(s_0)$. Note that $l(\hat{w}(s_n)) \leq l(\hat{w}(s_0))$ and $\sum_{c=1}^k \|w^*_c\|_1 = s_0$. It follows that $l(w^*) \leq l(\hat{w}(s_0))$, implying that w^* minimizes (2.1)−(2.2). This contradicts uniqueness of $\hat{w}(s_0)$.

If $\hat{w}(s)$ is not left continuous, there exists a sequence $s_n \uparrow s_0$ such that $\lim_{n\to\infty} \hat{w}(s_n) = w^* \neq \hat{w}(s_0)$. Note that $l(\hat{w}(s_n)) \leq l((s_n/s_0)\hat{w}(s_0))$, implying $\lim_{n\to\infty} l(\hat{w}(s_n)) \leq \lim_{n\to\infty} l((s_n/s_0)\hat{w}(s_0))$, i.e., $l(w^*) \leq l(\hat{w}(s_0))$. This contradicts uniqueness of $\hat{w}(s_0)$. Continuity follows.

Piecewise Linearity. It suffices to prove that $d = [\hat{w}(s+\epsilon) - \hat{w}(s)]/\epsilon$ is constant for sufficiently small $\epsilon > 0$. We use the fact that d is the solution of a linear program that is independent of ϵ.

First, because $\hat{w}(s)$ is continuous in s, we can take ϵ sufficiently small such that

1. for $\sum_{j=0}^{p} \hat{w}_{c,j}(s)x_{ij} + 1 \neq 0$, $\sum_{j=0}^{p} \hat{w}_{c,j}(s+\epsilon)x_{ij} + 1$ has the same sign as $\sum_{j=0}^{p} \hat{w}_{c,j}(s)x_{ij} + 1$, which implies $\mathcal{E}_+(\hat{w}(s)) \subset \mathcal{E}_+(\hat{w}(s+\epsilon))$ and $\mathcal{E}_-(\hat{w}(s)) \subset \mathcal{E}_-(\hat{w}(s+\epsilon))$;
2. for $\hat{w}_{c,j}(s) \neq 0$ and $j \neq 0$, $\hat{w}_{c,j}(s+\epsilon)$ has the same sign as $\hat{w}_{c,j}(s)$.

Consequently,

$$\begin{aligned}\frac{l(\hat{w}(s+\epsilon)) - l(\hat{w}(s))}{\epsilon} &= \sum_{(i,c)\in\mathcal{E}_+(\hat{w}(s+\epsilon))} \Big(\sum_j x_{ij}\frac{\hat{w}_{c,j}(s+\epsilon) - \hat{w}_{c,j}(s)}{\epsilon}\Big) \\ &\quad + \sum_{(i,c)\in\mathcal{E}(\hat{w}(s+\epsilon))} \Big[\sum_j x_{ij}\frac{\hat{w}_{c,j}(s+\epsilon) - \hat{w}_{c,j}(s)}{\epsilon}\Big]_+ \\ &= \sum_{(i,c)\in\mathcal{E}_+(\hat{w}(s+\epsilon))} \Big(\sum_j x_{ij}d_{c,j}\Big) + \sum_{(i,c)\in\mathcal{E}(\hat{w}(s+\epsilon))} \Big[\sum_j x_{ij}d_{c,j}\Big]_+ \\ &\stackrel{\text{def}}{=} g(d).\end{aligned}$$

Because $\hat{w}(s+\epsilon)$ is the minimizer of $l(w)$ with respect to w, subject to $\sum_{c,j\neq 0}|w_{c,j}| \leq s+\epsilon$ and $\sum_c w_c = \vec{0}$, it also minimizes $[l(w) - l(\hat{w}(s))]/\epsilon = g((w-\hat{w}(s))/\epsilon)$ with respect to w, subject to

$$\sum_{(c,j)\in\mathcal{A}(\hat{w}(s)),j\neq 0} \operatorname{sign}(\hat{w}(s))w_{c,j} + \sum_{(c,j)\in\mathcal{A}(\hat{w}(s))^c,j\neq 0} |w_{c,j}| \leq s+\epsilon,$$

equivalently,

$$\sum_{(c,j)\in\mathcal{A}(\hat{w}(s)),j\neq 0} \operatorname{sign}(\hat{w}(s))\frac{w_{c,j} - \hat{w}(s)}{\epsilon} + \sum_{(c,j)\in\mathcal{A}(\hat{w}(s))^c,j\neq 0} \Big|\frac{w_{c,j} - \hat{w}(s)}{\epsilon}\Big| \leq 1.$$

It follows that $d = (\hat{w}(s+\epsilon) - \hat{w}(s))/\epsilon$ solves

$$\min_d g(d), \tag{A.2}$$

$$\text{s.t.} \sum_{(c,j)\in\mathcal{A}(\hat{w}(s)),j\neq 0} \operatorname{sign}(\hat{w}(s))d_{c,j} + \sum_{(c,j)\in\mathcal{A}(\hat{w}(s))^c,j\neq 0} |d_{c,j}| \leq 1, \tag{A.3}$$

$$\text{and} \quad \sum_c d_c = \vec{0}. \tag{A.4}$$

Because (A.2)−(A.4) are independent of ϵ, the solution $d = [\hat{w}(s+\epsilon) - \hat{w}(s)]/\epsilon$ is constant for sufficient small ϵ. This completes the proof.

Proof of Theorem 2. Our proof uses the fact from the proof of Theorem 1 that for any $\hat{w}(s)$, $M(\hat{w}(s))u = r(\hat{w}(s))$ has a unique solution.

Note that $\hat{w}(s)$ is piecewisely linear. It follows from (A.2)−(A.4) that, as ϵ increases, the right derivative of $\hat{w}(s+\epsilon)$ remains constant until $\hat{w}_{c,j}(s+\epsilon)$ becomes

zero for some (c, j), or $(\sum_{j=0}^{p} w_{c,j}(s+\epsilon)x_{ij}+1)$ becomes zero for some (i, c). Let $\hat{w}(s+\epsilon) = \hat{w}(s) + d\epsilon$, for sufficiently small $\epsilon > 0$. Now consider $\hat{w}(s+\epsilon_1)$ and $\hat{w}(s+\epsilon_2)$, with $0 < \epsilon_1, \epsilon_2 < \epsilon$. It can be verified that $\mathcal{A}(\hat{w}(s+\epsilon_1)) = \mathcal{A}(\hat{w}(s+\epsilon_2))$ and $\mathcal{E}(\hat{w}(s+\epsilon_1)) = \mathcal{E}(\hat{w}(s+\epsilon_2))$. It follows from the proof of Theorem 1 that $u_{c,j} = \hat{w}_{c,j}(s+\epsilon_1)$, $(c, j) \in \mathcal{A}(\hat{w}(s+\epsilon_1))$, is the unique solution of the system of equations

$$\begin{cases} \sum_{j:(c,j)\in\mathcal{A}(\hat{w}(s+\epsilon_1))} x_{ij}u_{c,j} &= -1, \quad (i,c) \in \mathcal{E}(\hat{w}(s+\epsilon_1)), \\ \sum_{c:(c,j)\in\mathcal{A}(\hat{w}(s+\epsilon_1))} u_{c,j} &= 0, \quad j \in \mathcal{J}(\hat{w}(s+\epsilon_1)), \\ \sum_{(c,j)\in\mathcal{A}(\hat{w}(s+\epsilon_1));j\neq 0} \operatorname{sign}(\hat{w}_{c,j}(s+\epsilon_1))u_{c,j} &= s+\epsilon_1, \end{cases}$$

and $u_{c,j} = \hat{w}_{c,j}(s+\epsilon_2)$, $(c, j) \in \mathcal{A}(\hat{w}(s+\epsilon_1))$, is the unique solution to the system of equations

$$\begin{cases} \sum_{j:(c,j)\in\mathcal{A}(\hat{w}(s+\epsilon_1))} x_{ij}u_{c,j} &= -1, \quad (i,c) \in \mathcal{E}(\hat{w}(s+\epsilon_1)), \\ \sum_{c:(c,j)\in\mathcal{A}(\hat{w}(s+\epsilon_1))} u_{c,j} &= 0, \quad j \in \mathcal{J}(\hat{w}(s+\epsilon_1)), \\ \sum_{(c,j)\in\mathcal{A}(\hat{w}(s+\epsilon_1));j\neq 0} \operatorname{sign}(\hat{w}_{c,j}(s+\epsilon_1))u_{c,j} &= s+\epsilon_2. \end{cases}$$

This yields that $d_{c,j} = (\hat{w}(s+\epsilon_2) - \hat{w}(s+\epsilon_2))/(\epsilon_2 - \epsilon_1)$, $(c, j) \in \mathcal{A}(\hat{w}(s+\epsilon_1))$, is the unique solution of

$$\begin{cases} \sum_{j:(c,j)\in\mathcal{A}(\hat{w}(s+\epsilon_1))} x_{ij}d_{c,j} &= 0, \quad (i,c) \in \mathcal{E}(\hat{w}(s+\epsilon_1)), \\ \sum_{c:(c,j)\in\mathcal{A}(\hat{w}(s+\epsilon_1))} d_{c,j} &= 0, \quad j \in \mathcal{J}(\hat{w}(s+\epsilon_1)), \\ \sum_{(c,j)\in\mathcal{A}(\hat{w}(s+\epsilon_1));j\neq 0} \operatorname{sign}(\hat{w}_{c,j}(s+\epsilon_1))d_{c,j} &= 1. \end{cases}$$

Because the number of unknown variables is $|\mathcal{A}(\hat{w}(s+\epsilon_1))|$, while the number of equations is $|\mathcal{E}(\hat{w}(s+\epsilon_1))| + |\mathcal{J}(\hat{w}(s+\epsilon_1))| + 1$, $|\mathcal{A}(\hat{w}(s))| = |\mathcal{E}(\hat{w}(s))| + |\mathcal{J}(\hat{w}(s))| + 1$ if $\hat{w}(s)$ is not at a joint. Note that, as s increases and $\hat{w}(s)$ hits a joint, either $|\mathcal{A}(\hat{w}(s))|$ decreases by 1 or $|\mathcal{E}(\hat{w}(s))|$ increases by 1. Thus $|\mathcal{A}(\hat{w}(s))| = |\mathcal{E}(\hat{w}(s))| + |\mathcal{J}(\hat{w}(s))| + 2$ if $\hat{w}(s)$ is at a joint. This completes the proof.

References

Bertsimas, D. and Tsitsiklis, J. N. (1997). Introduction to Linear Optimization. Athena Scientific, Belmont, Massachusetts.

Bredensteiner, E. J. and Bennett, K. P. (1999). Multicategory classification by support vector machines. *Computat. Optim. Appl.* **12**, 35-46.

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004) Least angle regression. *Ann. Statist.* **32**, 407-451.

Guermeur, Y. (2002). Combining discriminant models with new multi-class SVMs. *Pattern Anal. Appl.* **5**, 168-179.

Hastie, T., Rosset, S., Tibshirani, R. and Zhu, J. (2004). The entire regularization path for the support vector machine. *J. Mach. Learn. Res.* **5**, 1391-1415.

Khan, J., Wei, J., Rignr, M., Saal, L., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C., Peterson, C. and Meltzer, P. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine* **7**, 673-679.

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. *J. Amer. Statist. Assoc.* **99**, 67-81.

Liu, Y. and Shen, X (2005). Multicategory psi-learning. *J. Amer. Statist. Assoc.* To appear.

Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J., Poggio, T., Gerald, W., Loda, M., Lander, E. and Golub, T. (2001). Multiclass cancer diagnosis using tumor gene expression signatures. *Proc. Nat. Acad. Sci.* **98**, 15149-15154.

Vapnik, V. (1998). *Statistical Learning Theory*, Wiley, New York.

Wang, L. and Shen, X. (2005). On L_1-norm multi-category support vector machines: methodology and theory. Manuscript.

Weston, J. and Watkins, C. (1998). Support vector machines for multiclass pattern recognition. *Proceedings of the Seventh European Symposium on Artificial Neural Networks*.

Zhu, J., Hastie, T., Rosset, S. and Tibshirani, R. (2003). 1-norm support vector machines. *Neural Inf. Process. Systems* **16**.

Zhang, T. (2004). Statistical analysis of some multi-category large margin classification methods. *J. Mach. Learn. Res.* **5**, 1225-1251.

School of Statistics, University of Minnesota, U.S.A.

E-mail: iamwlf@stat.umn.edu

School of Statistics, University of Minnesota, U.S.A.

E-mail: xshen@stat.umn.edu

(Received June 2005; accepted October 2005)