

# Multicategory $\psi$ -Learning and Support Vector Machine: Computational Tools

Yufeng LIU, Xiaotong SHEN, and Hani DOSS

Many margin-based binary classification techniques such as support vector machine (SVM) and  $\psi$ -learning deliver high performance. An earlier article proposed a new multicategory  $\psi$ -learning methodology that shows great promise in generalization ability. However,  $\psi$ -learning is computationally difficult because it requires handling a nonconvex minimization problem. In this article, we propose two computational tools for multicategory  $\psi$ -learning. The first one is based on d.c. algorithms and solved by sequential quadratic programming, while the second one uses the outer approximation method, which yields the global minimizer via sequential concave minimization. Numerical examples show the proposed algorithms perform well.

**Key Words:** Classification; Concave minimization; d.c. algorithms; Nonconvex minimization; Outer approximation; Quadratic programming.

## 1. INTRODUCTION

There has been considerable interest in margin-based classification techniques recently. Examples include support vector machine (SVM, Boser, Guyon, and Vapnik 1992; Mason, Bartlett, and Baxter 2000), distance weighted discrimination (Marron and Todd 2002), and  $\psi$ -learning (Shen, Tseng, Zhang, and Wong 2003; Liu and Shen 2004), among others.

SVM, a powerful classification tool, has gained its popularity and attracted tremendous interest due to its theoretical merits and successes in real applications, ranging from cancer genomics classification from gene expression data to text classification and handwritten character recognition; see Vapnik (1998). Shen et al. (2003) proposed a new learning methodology, called  $\psi$ -learning, and showed that it has significant advantages over SVM in terms of generalization. Recently, Liu and Shen (2004) generalized  $\psi$ -learning and SVM

---

Yufeng Liu is Assistant Professor, Department of Statistics and Operations Research, Carolina Center for Genome Sciences, University of North Carolina, CB 3260, Chapel Hill, NC 27599 (E-mail: yfliu@email.unc.edu). Xiaotong Shen is Professor, School of Statistics, The University of Minnesota, Minneapolis, MN 55455 (E-mail: xshen@stat.umn.edu). Hani Doss is Professor, Department of Statistics, The Ohio State University, Columbus, OH 43210 (doss@stat.ohio-state.edu).

©2005 American Statistical Association, Institute of Mathematical Statistics,  
and Interface Foundation of North America

*Journal of Computational and Graphical Statistics*, Volume 14, Number 1, Pages 219–236

DOI: 10.1198/106186005X37238

from the binary case to the multicategory case, and showed the generalization retains the interpretation of margins as well as desirable properties of the binary case.

The practical use of  $\psi$ -learning is difficult without good computational heuristics, because the optimization involved in  $\psi$ -learning is nonconvex. To make the multicategory methodology more useful, we use state-of-the-art technology in global optimization to develop computational tools for  $\psi$ -learning. Particularly, we propose two computational tools. The first one uses the difference convex algorithms (DCA) of An and Tao (1997), which solves nonconvex minimization via sequential quadratic programming. As a trivial consequence, we solve the convex minimization problem of multicategory SVM of Liu and Shen (2004) via quadratic programming. The second one uses the outer approximation method for d.c. optimization proposed by Blanquero and Carrizosa (2000), yielding an  $\epsilon$ -global minimizer of  $\psi$ -learning through sequential concave minimization obtained by vertex enumeration. Our numerical experience indicates that the proposed computational algorithms perform well.

This article is organized as follows. Section 2 reviews multicategory  $\psi$ -learning and SVM. Sections 3 and 4 are devoted to their computational developments. Numerical results are presented in Section 5, followed by a discussion. The appendix collects proofs of the theoretical results.

## 2. MULTICATEGORY $\psi$ -LEARNING AND SUPPORT VECTOR MACHINE

For  $k$ -class classification, a training sample  $\{(\mathbf{x}_i, y_i); i = 1, \dots, n\}$  is distributed according to an unknown probability distribution  $P(\mathbf{x}, y)$ . Here  $\mathbf{x}_i \in R^d; i = 1, \dots, n$ , are input vectors, and we code the class label  $y_i \in \{1, \dots, k\}$ . For any given new input vector  $\mathbf{x} \in S$ , a decision function vector  $\mathbf{f} = (f_1, \dots, f_k)$ , one function representing each class, mapping from  $S \subset R^d$  to  $R$ , is employed to estimate the value of its label via a decision rule:  $\arg\max_{j=1, \dots, k} f_j(\mathbf{x})$ . In short,  $\mathbf{x} \in S$  is assigned to a class with the highest value of  $f_j(\mathbf{x}); j = 1, \dots, k$ . In our context, each  $f_j$  may not be a probability, although this decision rule mimics the maximum probability decision rule.

We now define notations to be used before introducing multicategory  $\psi$ -learning and SVM. In the sequel, vectors are column vectors. Denote the  $i$ th element of vector  $\mathbf{e}$  to be  $\mathbf{e}(i)$ , and  $\mathbf{0}_m$  and  $\mathbf{1}_m$  to be an  $m$ -dimensional vector of 0 and 1, respectively. Denote the  $i$ th element of an  $m_1 \times m_2$  matrix  $M$  to be  $M(i, j)$ , its transpose by  $M^T$ , and its  $i$ th row by  $M^i$ . Let  $\text{vec}(M)$  be an  $m_1 m_2$ -dimensional vector with its  $m_1(i_2 - 1) + i_1$ th element to be  $M(i_1, i_2)$ , which converts the matrix  $M$  to a vector. Denote  $I_m$  and  $\mathbf{0}_{m_1 \times m_2}$  to be the  $m \times m$  identity matrix and an  $m_1 \times m_2$  matrix of 0, respectively. Let  $\langle \cdot, \cdot \rangle$  be an inner product in the corresponding Euclidean space,  $\otimes$  be the Kronecker product, and  $\wedge$  be the minimum operator.

For a linear problem in which  $f_j(\mathbf{x}) = \langle \mathbf{w}_j, \mathbf{x} \rangle + b_j; \mathbf{w}_j \in R^d, b_j \in R$ , is a hyperplane, let  $\mathbf{b} = (b_1, \dots, b_k)^T \in R^k$ ,  $\mathbf{w} = \text{vec}(\mathbf{w}_1, \dots, \mathbf{w}_k) \in R^{kd}$ ,  $\tilde{\mathbf{w}}_j = \begin{bmatrix} b_j \\ \mathbf{w}_j \end{bmatrix} \in R^{d+1}; j = 1, \dots, k$ , and  $\tilde{\mathbf{w}} = \text{vec}(\tilde{\mathbf{w}}_1, \dots, \tilde{\mathbf{w}}_k) \in R^{k(d+1)}$ . According to Liu and Shen (2004),

the multicategory  $\psi$ -learning aims to solve the minimization problem:

$$\min_{\tilde{\mathbf{w}}} \left( \frac{1}{2} \sum_{j=1}^k \langle \mathbf{w}_j, \mathbf{w}_j \rangle + C \sum_{i=1}^n \psi(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) \right)$$

$$\text{subject to } \sum_{j=1}^k f_j(\mathbf{x}) = 0; \quad \forall \mathbf{x} \in S. \quad (2.1)$$

Here  $C$  ( $C > 0$ ) is the tuning parameter,  $\mathbf{g}(\mathbf{f}(\mathbf{x}), y) = (f_y(\mathbf{x}) - f_1(\mathbf{x}), \dots, f_y(\mathbf{x}) - f_{y-1}(\mathbf{x}), f_y(\mathbf{x}) - f_{y+1}(\mathbf{x}), \dots, f_y(\mathbf{x}) - f_k(\mathbf{x}))^T$  is a  $(k-1)$ -dimensional vector, which is used to perform multiple comparison of class  $y$  against other classes, and  $\psi(\mathbf{u}) = 0$  if  $u_{\min} = \min\{u_1, \dots, u_{k-1}\} \geq 1$ , 2 if  $u_{\min} < 0$ , and  $2(1 - u_{\min})$  otherwise, where  $\mathbf{u} = (u_1, \dots, u_{k-1})$ . Problem (2.1) involves nonconvex minimization. As pointed out by Shen et al. (2003) and Liu and Shen (2004), the method is not implementable without good computational heuristics.

The sum-to-zero constraint  $\sum_{j=1}^k f_j(\mathbf{x}) = 0$  in (2.1) involves all values of  $\mathbf{x} \in S$ . We therefore need to reduce the infinite constraints to finite constraints. Theorem 1 provides such a result.

**Theorem 1.** *The constraint  $\sum_{j=1}^k f_j(\mathbf{x}) = 0 \quad \forall \mathbf{x} \in S$  is equivalent to the constraint  $\tilde{X} \sum_{j=1}^k \tilde{\mathbf{w}}_j = 0$ , where  $\tilde{X} = [\mathbf{1}_n, X] = (\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n)^T$  is an  $n \times (d+1)$  matrix,  $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$  is the  $n \times d$  design matrix, and  $\tilde{\mathbf{x}}_i = \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}$  is a  $(d+1)$ -dimensional vector;  $i = 1, \dots, n$ .*

By Theorem 1, (2.1) reduces to

$$\min_{\tilde{\mathbf{w}}} \left( \frac{1}{2} \sum_{j=1}^k \langle \mathbf{w}_j, \mathbf{w}_j \rangle + C \sum_{i=1}^n \psi(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) \right) \quad \text{subject to } \tilde{X} \sum_{j=1}^k \tilde{\mathbf{w}}_j = 0. \quad (2.2)$$

In contrast to (2.1), (2.2) involves only finitely many constraints.

For a nonlinear problem, each decision function  $f_j(\mathbf{x})$  is represented as  $h_j(\mathbf{x}) + b_j$  with  $h_j \in H_K$  a reproducing kernel Hilbert space (RKHS). Here the kernel  $K(\cdot, \cdot)$  mapping  $S \times S$  to  $R$  is a positive definite function. By the representer theorem of Kimeldorf and Wahba (1971) (also see Wahba 1998), the nonlinear problem can be reduced to finding finite dimensional coefficients and  $h_j(\mathbf{x}) = \sum_{i=1}^n v_{ji} K(\mathbf{x}_i, \mathbf{x})$ ;  $j = 1, \dots, k$ . Using an argument similar to that of the linear case, nonlinear multicategory  $\psi$ -learning then becomes

$$\min_{\tilde{\mathbf{v}}} \left( \frac{1}{2} \sum_{j=1}^k \langle \mathbf{v}_j, \mathbf{K} \mathbf{v}_j \rangle + C \sum_{i=1}^n \psi(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) \right) \quad \text{subject to } \tilde{\mathbf{K}} \sum_{j=1}^k \tilde{\mathbf{v}}_j = 0, \quad (2.3)$$

where  $\tilde{\mathbf{K}} = [\mathbf{1}_n, \mathbf{K}]$ ,  $\mathbf{K}$  is an  $n \times n$  matrix whose  $i_1 i_2$ th entry is  $K(\mathbf{x}_{i_1}, \mathbf{x}_{i_2})$ ,  $\mathbf{v}_j = (v_{j1}, \dots, v_{jn})^T \in R^n$ ,  $\tilde{\mathbf{v}}_j = \begin{bmatrix} b_j \\ \mathbf{v}_j \end{bmatrix} \in R^{n+1}$ ,  $\mathbf{v} = \text{vec}(\mathbf{v}_1, \dots, \mathbf{v}_k) \in R^{nk}$ , and  $\tilde{\mathbf{v}} = \text{vec}(\tilde{\mathbf{v}}_1, \dots, \tilde{\mathbf{v}}_k) \in R^{k(n+1)}$ .

With the above formulation in place, the multicategory SVM can be obtained by replacing  $\psi$  by  $\psi_1$  in (2.2) and (2.3), where  $\psi_1(\mathbf{u}) = 0$  if  $u_{\min} \geq 1$  and  $2(1 - u_{\min})$  otherwise. Then, linear SVM solves the following minimization problem:

$$\min_{\tilde{\mathbf{w}}} \left( \frac{1}{2} \sum_{j=1}^k \langle \mathbf{w}_j, \mathbf{w}_j \rangle + C \sum_{i=1}^n \psi_1(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) \right) \quad \text{subject to} \quad \tilde{X} \sum_{j=1}^k \tilde{\mathbf{w}}_j = 0. \quad (2.4)$$

Analogous to nonlinear  $\psi$ -learning, nonlinear SVM becomes

$$\min_{\tilde{\mathbf{v}}} \left( \frac{1}{2} \sum_{j=1}^k \langle \mathbf{v}_j, \mathbf{K}\mathbf{v}_j \rangle + C \sum_{i=1}^n \psi_1(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) \right) \quad \text{subject to} \quad \tilde{\mathbf{K}} \sum_{j=1}^k \tilde{\mathbf{v}}_j = 0. \quad (2.5)$$

Finally, the classifier of  $\psi$ -learning or SVM is defined as  $\operatorname{argmax}_{j=1, \dots, k} \hat{f}_j(\mathbf{x})$ , where  $\hat{f}_j(\mathbf{x}) = \langle \hat{\mathbf{w}}_j, \mathbf{x} \rangle + \hat{b}_j$  and  $\hat{f}_j(\mathbf{x}) = \sum_{i=1}^n \hat{v}_{ji} K(\mathbf{x}_i, \mathbf{x}) + \hat{b}_j$ , respectively, for linear and nonlinear problems, defined by the minimizers  $\hat{\mathbf{w}}$  and  $\hat{\mathbf{v}}$  of (2.2) and (2.3) or (2.4) and (2.5).

### 3. D.C. OPTIMIZATION ALGORITHMS

The minimization involved in (2.2) and (2.3) is nonconvex. To handle nonconvex minimization, we use a global optimization technique called difference convex (d.c.) algorithms (DCA, An and Tao 1997) in particular the simplified DCA. To this end, we construct a d.c. decomposition of our cost function into a difference of two convex functions, or a sum of convex and concave functions.

For simplicity, we focus our discussion on (2.2) because (2.3) can be treated similarly. To solve (2.2), we first decompose  $\psi$  into  $\psi_1 + \psi_2$  with  $\psi_1$  defined immediately above (2.4) and  $\psi_2$  being 0 if  $u_{\min} \geq 0$  and  $2u_{\min}$  otherwise. Figure 1 depicts the decomposition of  $\psi = \psi_1 + \psi_2$  in the binary case. This thus yields a d.c. decomposition of  $s = s_1 + s_2$ , where  $s_1(\tilde{\mathbf{w}}) = \frac{1}{2} \sum_{j=1}^k \langle \mathbf{w}_j, \mathbf{w}_j \rangle + C \sum_{i=1}^n \psi_1(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i))$  is convex while  $s_2(\tilde{\mathbf{w}}) = C \sum_{i=1}^n \psi_2(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i))$  is concave in  $\tilde{\mathbf{w}}$ . Then (2.2) becomes

$$\min_{\tilde{\mathbf{w}}} s(\tilde{\mathbf{w}}) = s_1(\tilde{\mathbf{w}}) + s_2(\tilde{\mathbf{w}}) \quad \text{subject to} \quad \tilde{X} \sum_{j=1}^k \tilde{\mathbf{w}}_j = 0. \quad (3.1)$$

This d.c. decomposition has a nice interpretation in that  $s_1$  in (3.1) is the convex cost function of the multicategory SVM in (2.4) and  $s_2$  in (3.1) can be treated as a bias correction for generalization due to imposed convexity to  $s_1$ .

The basic idea of DCA is to construct a sequence of subproblems defined by the affine minorization  $s_1(\tilde{\mathbf{w}}) + s_2(\tilde{\mathbf{w}}^l) + \langle \nabla s_2(\tilde{\mathbf{w}}^l), \tilde{\mathbf{w}} - \tilde{\mathbf{w}}^l \rangle$  of  $s(\tilde{\mathbf{w}})$  and solve them iteratively, where  $\nabla s_2(\tilde{\mathbf{w}}^l)$  is the subgradient of  $s_2(\tilde{\mathbf{w}})$  at  $\tilde{\mathbf{w}}^l$ . Given the solution of the  $l$ th subproblem, the  $(l+1)$ th subproblem can be solved by minimizing  $s_1(\tilde{\mathbf{w}}) + \langle \nabla s_2(\tilde{\mathbf{w}}^l), \tilde{\mathbf{w}} \rangle$  with respect to  $\tilde{\mathbf{w}}$  after ignoring the constant term. By concavity of  $s_2$ , DCA yields a sequence of nonincreasing convex upper approximations  $s_1(\tilde{\mathbf{w}}) + s_2(\tilde{\mathbf{w}}^l) + \langle \nabla s_2(\tilde{\mathbf{w}}^l), \tilde{\mathbf{w}} - \tilde{\mathbf{w}}^l \rangle$  to  $s(\tilde{\mathbf{w}})$ .

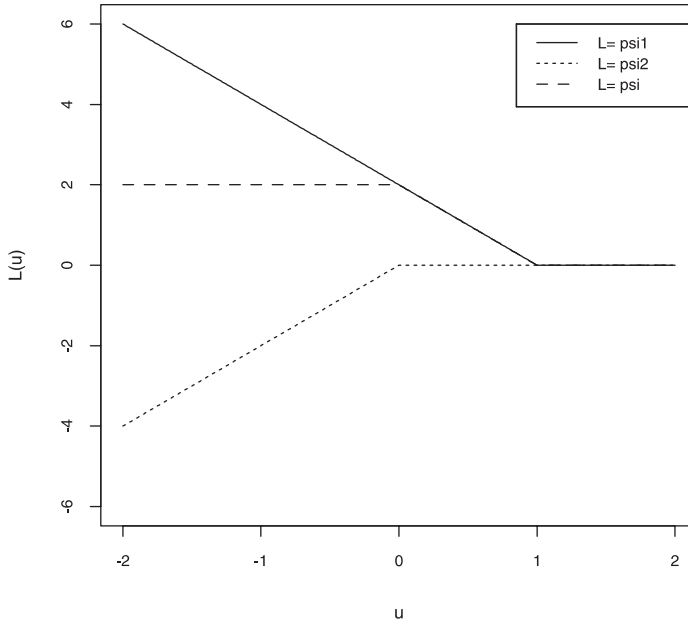


Figure 1. Plot of the  $\psi$ -function and a d.c. decomposition of  $\psi = \psi_1 + \psi_2$  in the binary case.

We now give technical details of the DCA. The subgradient  $\nabla(s_2(\tilde{\mathbf{w}}))$  of  $s_2$  can be expressed as  $C \sum_{i=1}^n \nabla \psi_2(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i))$ , where  $\nabla \psi_2$  is the subgradient of  $\psi_2$  and  $\nabla \psi_2 = (\nabla_1 \psi_2, \dots, \nabla_k \psi_2)$  with  $\nabla_j \psi_2$  being the subgradient of  $\psi_2$  with respect to  $\tilde{\mathbf{w}}_j$ ;  $j = 1, \dots, k$ . By definition, the subgradient here is not unique. For convenience, we choose  $\nabla_j \psi_2(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) = 2\tilde{\mathbf{x}}_i$  if  $j = y_i$  and  $\min_{m \in \{1, \dots, k\} \setminus \{y_i\}} \{\tilde{\mathbf{x}}_i^T (\tilde{\mathbf{w}}_{y_i} - \tilde{\mathbf{w}}_m)\} < 0$ ;  $-2\tilde{\mathbf{x}}_i$  if  $j \neq y_i$  and  $\tilde{\mathbf{x}}_i^T (\tilde{\mathbf{w}}_{y_i} - \tilde{\mathbf{w}}_j) < 0 \wedge \min_{m \in \{1, \dots, k\} \setminus \{y_i, j\}} \{\tilde{\mathbf{x}}_i^T (\tilde{\mathbf{w}}_{y_i} - \tilde{\mathbf{w}}_m)\}$ ; and  $\mathbf{0}_{d+1}$  otherwise.

At the  $(l + 1)$ th iteration, the  $l$ th subproblem is solved to yield  $\tilde{\mathbf{w}}^{l+1}$ :

$$\min_{\tilde{\mathbf{w}}} \left( \frac{1}{2} \sum_{j=1}^k \langle \mathbf{w}_j, \mathbf{w}_j \rangle + C \sum_{i=1}^n \psi_1(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) - \sum_{j=1}^k \langle \nabla \mathbf{w}_j^l, \mathbf{w}_j \rangle - \langle \nabla \mathbf{b}^l, \mathbf{b} \rangle \right),$$

$$\text{subject to } \tilde{X} \sum_{j=1}^k \tilde{\mathbf{w}}_j = \mathbf{0}, \quad (3.2)$$

where  $\langle \nabla s_2(\tilde{\mathbf{w}}^l), \tilde{\mathbf{w}} \rangle = -\sum_{j=1}^k \langle \nabla \mathbf{w}_j^l, \mathbf{w}_j \rangle - \langle \nabla \mathbf{b}^l, \mathbf{b} \rangle$ . To apply quadratic programming to (3.2), we introduce  $n$  nonnegative slack variables  $\xi_i$  for  $\psi_1(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i))$ ;  $i = 1, \dots, n$ , which yields the following primal problem:

$$\min_{\xi, \tilde{\mathbf{w}}} L_P = \frac{1}{2} \sum_{j=1}^k \langle \mathbf{w}_j, \mathbf{w}_j \rangle + C \sum_{i=1}^n \xi_i - \sum_{j=1}^k \langle \nabla \mathbf{w}_j^l, \mathbf{w}_j \rangle - \langle \nabla \mathbf{b}^l, \mathbf{b} \rangle, \quad (3.3)$$

subject to

$$\xi_i \geq 0; \quad i = 1, \dots, n, \quad (3.4)$$

$$2(1 - \tilde{\mathbf{x}}_i^T(\tilde{\mathbf{w}}_{y_i} - \tilde{\mathbf{w}}_j)) \leq \xi_i; \quad i = 1, \dots, n, \quad j = 1, \dots, k, \quad j \neq y_i, \quad (3.5)$$

$$\tilde{X} \sum_{j=1}^k \tilde{\mathbf{w}}_j = 0, \quad (3.6)$$

where  $\boldsymbol{\xi} = (\xi_1, \dots, \xi_n)^T$ .

The primal problem (3.3) will be solved via its dual form, which we obtain by employing the Lagrange multipliers and solve via quadratic programming. After some calculations, we derive the dual problem in the following form.

**Theorem 2.** *The dual problem of (3.3) is equivalent to*

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \boldsymbol{\beta}^T H \boldsymbol{\beta} + \mathbf{g}^T \boldsymbol{\beta}, \quad (3.7)$$

subject to

$$\mathbf{1}_n^T Q_j \boldsymbol{\beta} = \nabla \mathbf{b}_j^l; \quad j = 1, \dots, k, \quad A \boldsymbol{\beta} \leq C \mathbf{1}_n, \quad \text{and} \quad -B \boldsymbol{\beta} \leq \mathbf{0}_{n(k-1)}.$$

This yields solution  $\mathbf{w}$  of (3.3):  $\mathbf{w}_j^{l+1} = \nabla \mathbf{w}_j^l - X^T Q_j \boldsymbol{\beta}^*$ ;  $j = 1, \dots, k$ , where  $\boldsymbol{\beta}^*$  is the solution of (3.7).

In Theorem 2,  $\boldsymbol{\beta}$  is an  $nk$ -dimensional vector corresponding to  $nk$  Lagrange multipliers, obtained from  $n(k+1)$ -dimensional vector  $\boldsymbol{\beta}^0 = \text{vec}(\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_k, \boldsymbol{\delta})$  by removing the  $r_i$ th element of  $\boldsymbol{\beta}^0$ , where  $r_i = (y_i - 1)n + i$ ;  $i = 1, \dots, n$ , correspond to the  $n$  redundant multipliers in  $\boldsymbol{\beta}^0$ . Moreover,  $H = \sum_{j=1}^k Q_j^T X X^T Q_j$  is an  $nk \times nk$  matrix and  $\mathbf{g} = \mathbf{a} - \sum_{j=1}^k Q_j^T X \nabla \mathbf{w}_j^l$  is an  $nk$ -dimensional vector, where  $\mathbf{a}$  is an  $nk$ -dimensional vector with  $\mathbf{a}(m) = -2$  for  $m = 1, \dots, n(k-1)$  and  $0$  for  $m = n(k-1) + 1, \dots, nk$ , and  $Q_j$  is an  $n \times nk$  matrix obtained from  $Q_j^0 = 2U_j - 2V_j U + U_{k+1}$  by removing the  $r_i$ th column of  $Q_j^0$  with  $U_m = I_{k+1}^m \otimes I_n$ ,  $U = \sum_{j=1}^k U_j = (\mathbf{1}_k^T, 0) \otimes I_n$ , and  $V_j$  being an  $n \times n$  diagonal matrix with its diagonal element being  $I(y_i = j)$ ;  $m = 1, \dots, k+1$ ,  $j = 1, \dots, k$ ,  $i = 1, \dots, n$ . Finally,  $B = [I_{n(k-1)}, \mathbf{0}_{n(k-1) \times n}]$  is an  $n(k-1) \times nk$  matrix and  $A$  is an  $n \times nk$  matrix obtained from  $A^0 = [I_n - V_1, \dots, I_n - V_k, \mathbf{0}_{n \times n}]$  by removing the  $r_i$ th column of  $A^0$ ;  $i = 1, \dots, n$ .

After  $\mathbf{w}^{l+1}$  is obtained,  $\mathbf{b}^{l+1}$  is determined by the Karush-Kuhn-Tucker (KKT) complementary conditions. Specifically, by Equations (A.1), (A.2), and (A.5), the KKT conditions can be reduced to a set of equations

$$(b_{y_{i^*}} - b_{j^*}) = 1 - \mathbf{x}_{i^*}^T (\mathbf{w}_{y_{i^*}}^{l+1} - \mathbf{w}_{j^*}^{l+1}) \quad \text{with} \quad i^*, j^* \quad \text{satisfying}$$

$$0 < \sum_{j \in \{1, \dots, k\} \setminus \{y_{i^*}\}} \alpha_{i^* j} < C, \quad 0 < \alpha_{i^* j^*} < C. \quad (3.8)$$

Then  $\mathbf{b}^{l+1}$  may be obtained by solving the set of linear Equations (3.8). Alternatively, after substituting  $\mathbf{w}$  by  $\mathbf{w}^{l+1}$ , we can obtain  $\mathbf{b}^{l+1}$  from (3.2) via linear programming. For linear

programming, we introduce  $n$  nonnegative slack variables  $\eta_i$  and obtain  $\mathbf{b}^{l+1}$  by solving the following problem:

$$\min_{\boldsymbol{\eta}, \mathbf{b}} \left( C \sum_{i=1}^n \eta_i - \langle \nabla \mathbf{b}_j^l, \mathbf{b} \rangle \right), \quad (3.9)$$

subject to

$$\begin{aligned} \eta_i &\geq 0; & i &= 1, \dots, n, \\ \eta_i + 2(b_{y_i} - b_j) &\geq 2[1 - \mathbf{x}_i^T (\mathbf{w}_{y_i}^{l+1} - \mathbf{w}_j^{l+1})]; & i &= 1, \dots, n, j = 1, \dots, k, j \neq y_i, \\ \sum_{j=1}^k b_j &= 0, \end{aligned}$$

where  $\boldsymbol{\eta} = (\eta_1, \dots, \eta_n)^T$ .

Algorithm 1 solves (3.2), and thus yields a (possibly local) solution of (3.1).

**Algorithm 1 (Linear):**

**Step 1:** Initial value. Compute the SVM solution  $\tilde{\mathbf{w}}^0 = \text{vec}(\tilde{\mathbf{w}}_1^0, \dots, \tilde{\mathbf{w}}_k^0)$ . Specifically, obtain  $\mathbf{w}^0$  and  $\mathbf{b}^0$  by solving (3.7) and (3.9) with  $\nabla \mathbf{w}_j^0 = 0$  and  $\nabla \mathbf{b}_j^0 = 0$ ;  $j = 1, \dots, k$ , respectively.

At iteration  $l + 1$  ( $l \geq 0$ ), for given  $\tilde{\mathbf{w}}^l = \text{vec}(\tilde{\mathbf{w}}_1^l, \dots, \tilde{\mathbf{w}}_k^l)$ :

**Step 2:** Iteration. Solve (3.7) to yield  $\mathbf{w}^{l+1} = \text{vec}(\mathbf{w}_1^{l+1}, \dots, \mathbf{w}_k^{l+1})$  and (3.8) or (3.9) to obtain  $\mathbf{b}^{l+1}$  after substituting  $\mathbf{w}$  by  $\mathbf{w}^{l+1}$  in (3.2). Finally,  $\tilde{\mathbf{w}}^{l+1} = \text{vec}(\tilde{\mathbf{w}}_1^{l+1}, \dots, \tilde{\mathbf{w}}_k^{l+1})$ .

**Step 3:** Stopping rule. Stop if  $|s(\tilde{\mathbf{w}}^{l+1}) - s(\tilde{\mathbf{w}}^l)| \leq \epsilon$  for prespecified  $\epsilon > 0$ . The final solution is then  $\hat{\tilde{\mathbf{w}}} = \text{argmin}_{0 \leq h \leq l+1} s(\tilde{\mathbf{w}}^h)$ .

For a nonlinear problem, (2.3) can be solved in a similar manner as that of (2.2) with  $\tilde{\mathbf{w}}$  being replaced by  $\tilde{\mathbf{v}}$ . In particular, at the  $(l + 1)$ th iteration, the following subproblem is solved to yield  $\tilde{\mathbf{v}}^{l+1}$ :

$$\begin{aligned} \min_{\tilde{\mathbf{v}}} s(\tilde{\mathbf{v}}) &= \frac{1}{2} \sum_{j=1}^k \langle \mathbf{v}_j, \mathbf{K} \mathbf{v}_j \rangle + C \sum_{i=1}^n \psi_1(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) - \sum_{j=1}^k \langle \nabla \mathbf{v}_j^l, \mathbf{v}_j \rangle - \langle \nabla \mathbf{b}^l, \mathbf{b} \rangle, \\ &\text{subject to} \quad \tilde{\mathbf{K}} \sum_{j=1}^k \tilde{\mathbf{v}}_j = 0, \quad (3.10) \end{aligned}$$

where  $\sum_{j=1}^k \langle \nabla \mathbf{v}_j^l, \mathbf{v}_j \rangle + \langle \nabla \mathbf{b}^l, \mathbf{b} \rangle$  is the affine minorization of  $-C \sum_{i=1}^n \psi_2(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i))$  for given  $\tilde{\mathbf{v}}^l$  after removing the constant term.

Algorithm 2 solves (3.10), and thus yields a (possibly local) solution of (2.3).

**Algorithm 2 (Nonlinear):**

**Step 1:** Initial value. Compute the SVM solution  $\tilde{\mathbf{v}}^0$  by minimizing (2.5).

At iteration  $l + 1$  ( $l \geq 0$ ), for given  $\tilde{\mathbf{v}}^l = \text{vec}(\tilde{\mathbf{v}}_1^l, \dots, \tilde{\mathbf{v}}_n^l)$ :

**Step 2:** Iteration. Solve (3.7) with  $H = \sum_{j=1}^k Q_j^T \mathbf{K} Q_j$  and  $\mathbf{g} = \mathbf{a} - \sum_{j=1}^k Q_j^T \nabla \mathbf{v}_j^l$  to obtain solution  $\beta^*$  and (3.10) to yield  $\mathbf{b}^{l+1}$  after substituting  $\mathbf{v}$  by  $\mathbf{v}^{l+1} = (\mathbf{v}_1^{l+1}, \dots, \mathbf{v}_n^{l+1})$  in (3.10) using the formula  $\mathbf{v}_j^{l+1} = \mathbf{K}^{-1} \nabla \mathbf{v}_j^l - Q_j \beta^*$ ;  $j = 1, \dots, k$ . Finally,  $\tilde{\mathbf{v}}^{l+1} = \text{vec}(\tilde{\mathbf{v}}_1^{l+1}, \dots, \tilde{\mathbf{v}}_n^{l+1})$ .

**Step 3:** Stopping rule. Stop if  $|s(\tilde{\mathbf{v}}^{l+1}) - s(\tilde{\mathbf{v}}^l)| \leq \epsilon$  for prespecified  $\epsilon > 0$ . The final solution is then  $\text{argmin}_{0 \leq h \leq l+1} s(\tilde{\mathbf{v}}^h)$ .

**Remark:** Algorithm 2 involves the inverse of the kernel matrix  $\mathbf{K}$ , as defined in (2.3), for  $\mathbf{K} \mathbf{v}_j^{l+1} = \nabla \mathbf{v}_j^l - \mathbf{K} Q_j \beta^*$ , which may be ill-posed for example because of numerical errors when the training size  $n$  is large. To overcome this difficulty, we may solve for  $\mathbf{K}^{-1} \nabla \mathbf{v}_j^l$  directly to avoid the inverse operation, based on an observation that  $\mathbf{K}^{-1} \nabla \mathbf{v}_j^l = -2C \mathbf{1}_n$  if  $j = y_i$  and  $\min_{h=1, \dots, k, h \neq y_i} \{\tilde{\mathbf{K}}_i(\tilde{\mathbf{v}}_{y_i}^l - \tilde{\mathbf{v}}_h^l)\} < 0$ ,  $2C \mathbf{1}_n$  if  $j \neq y_i$  and  $\tilde{\mathbf{K}}_i(\tilde{\mathbf{v}}_{y_i}^l - \tilde{\mathbf{v}}_j^l) < 0 \wedge \min_{h=1, \dots, k, h \neq y_i, h \neq j} \{\tilde{\mathbf{K}}_i(\tilde{\mathbf{v}}_{y_i}^l - \tilde{\mathbf{v}}_h^l)\}$ , and  $\mathbf{0}_n$  otherwise, where  $\tilde{\mathbf{K}}_i$  is the  $i$ th row of matrix  $\tilde{\mathbf{K}} = [\mathbf{1}_n, \mathbf{K}]$ .

**Theorem 3.** *Algorithm 1 terminates finitely and generates a sequence  $\tilde{\mathbf{w}}^l$  such that  $s(\tilde{\mathbf{w}}^l)$  is nonincreasing with respect to  $l$ ,  $\lim_{l \rightarrow \infty} s(\tilde{\mathbf{w}}^l) \geq \min_{\tilde{\mathbf{w}}} s(\tilde{\mathbf{w}})$ , and  $\lim_{l \rightarrow \infty} \|\tilde{\mathbf{w}}^{l+1} - \tilde{\mathbf{w}}^l\| = 0$ . Algorithm 2 continues to share the same convergence properties as Algorithm 1 with  $\tilde{\mathbf{w}}$  replaced by  $\tilde{\mathbf{v}}$  when  $\mathbf{K}^{-1}$  exists.*

Theorem 3 says that Algorithm 1 or 2 terminates finitely, and may yield the global solution although it can not guarantee so. Furthermore, as suggested by our simulations, Algorithms 1 and 2 converge fast and usually stop within 20 iterations. Consequently, the algorithms are suited for large-scale problems. The complexity of Algorithms 1 and 2 is roughly  $O(n^3 k^3 p)$  when quadratic programming solver LOQO is applied (Vanderbei 1994), where  $p$ , the number of iterations, is determined by the error bound  $\epsilon > 0$  and is usually small.

As defined by Liu and Shen (2004), “support vectors” (SVs) are instances in the training dataset with  $y = j$  satisfying  $\min(\mathbf{g}(\mathbf{x}, j)) \leq 1$ ;  $j = 1, \dots, k$ . Lemma 1 shows that the classifiers yielded by Algorithm 1 or 2 only depend on SVs. Therefore, sparsity of the solutions is achieved if the number of SVs is small.

## 4. OUTER APPROXIMATION METHOD

Algorithms 1 and 2 implement the d.c. optimization by solving a sequence of quadratic programming problems. In particular, they solve the nonconvex optimization problem by minimizing a sequence of convex upper approximations of the objective function. In this section, we develop a different global optimization technique using the outer approximation method for d.c. optimization. Our technique will be implemented via an outer approximation construction (Blanquero and Carrizosa 2000) and an improved version of the reverse vertex enumeration method (Avis 2000). This technique solves a d.c. minimization problem via minimizing a sequence of its lower envelopes, and solves a sequence of concave

minimization via vertex enumeration. As a result, it yields an  $\epsilon$ -global optimizer.

We now describe the outer approximation method for (3.1) in detail. The construction of the lower envelope  $E^l(\tilde{\mathbf{w}})$  of  $s(\tilde{\mathbf{w}}) = s_1(\tilde{\mathbf{w}}) + s_2(\tilde{\mathbf{w}})$  proceeds as follows. At iteration  $l + 1$ ,  $E^l(\tilde{\mathbf{w}}) = \max_{1 \leq i \leq l} L_i(\tilde{\mathbf{w}})$ , where  $L_i(\tilde{\mathbf{w}})$  is  $s_1(\tilde{\mathbf{w}}^i) + \langle \nabla s_1(\tilde{\mathbf{w}}^i), \tilde{\mathbf{w}} - \tilde{\mathbf{w}}^i \rangle + s_2(\tilde{\mathbf{w}})$  with  $\nabla s_1(\tilde{\mathbf{w}}^i)$  being the subgradient of  $s_1(\tilde{\mathbf{w}})$  at  $\tilde{\mathbf{w}}^i$  and  $\tilde{\mathbf{w}}^i$ ;  $i = 0, \dots, l$ , are obtained from the previous steps. Then  $\tilde{\mathbf{w}}^{l+1} = \operatorname{argmin}_{\tilde{\mathbf{w}}} E^l(\tilde{\mathbf{w}})$  defines  $L_{l+1}(\tilde{\mathbf{w}}) = s_1(\tilde{\mathbf{w}}^{l+1}) + \langle \nabla s_1(\tilde{\mathbf{w}}^{l+1}), \tilde{\mathbf{w}} - \tilde{\mathbf{w}}^{l+1} \rangle + s_2(\tilde{\mathbf{w}})$ , thus yielding  $E^{l+1}(\tilde{\mathbf{w}}) = \max(L_{l+1}(\tilde{\mathbf{w}}), E^l(\tilde{\mathbf{w}}))$ . By construction, we have  $s(\tilde{\mathbf{w}}) \geq E^{l+1}(\tilde{\mathbf{w}}) \geq E^l(\tilde{\mathbf{w}}) \geq \dots \geq E^1(\tilde{\mathbf{w}})$ .

To obtain the minimizer  $\tilde{\mathbf{w}}^{l+1}$  at iteration  $l + 1$ , we solve the following concave minimization problem subject to linear constraints:

$$\min_{(t, \tilde{\mathbf{w}}) \in B^l} t + s_2(\tilde{\mathbf{w}}), \quad (4.1)$$

where  $B^l = \{(t, \tilde{\mathbf{w}}) : t \geq s_1(\tilde{\mathbf{w}}^i) + \langle \nabla s_1(\tilde{\mathbf{w}}^i), \tilde{\mathbf{w}} - \tilde{\mathbf{w}}^i \rangle, i = 1, \dots, l; \tilde{X} \sum_{j=1}^k \tilde{\mathbf{w}}_j = 0\}$  is a polyhedron. This amounts to minimizing a concave function in (4.1) over the polyhedron  $B^l$ . To avoid degenerate polyhedrons, we may begin with a bounded polyhedron  $B^0$ , called polytope, where  $B^0$  is constructed by  $2(k(d+1) + 1)$  linear constraints defined by the initial bounds of  $(t, \tilde{\mathbf{w}})$ . Because  $s_1(\tilde{\mathbf{w}}) \geq 0$  holds,  $t \geq 0$  is used as a lower bound of  $t$ .

By concavity, the minimizer  $\tilde{\mathbf{w}}^{l+1}$  is attained at a vertex of the polytope, defined by the  $l$  linear constraints in  $B^0$ . Equivalently, minimization (4.1) reduces to a problem of finding new vertices of a polytope when a new linear constraint is added. The new linear constraint generated from iteration  $l + 1$  yields  $B^{l+1} \subset B^l$ , thus new vertices for iteration  $l + 2$ .

Any value  $\tilde{\mathbf{w}}$  inside  $B^0$  can be chosen as an initial value. In fact, this technique is insensitive to the choice of initial values because the final solution is an  $\epsilon$ -global minimizer if it is contained in  $B^0$ . But a good initial value may expedite convergence.

### Algorithm 3:

**Step 1:** Initialization. Set up initial polytope  $B^0$  using the bounds of  $(t, \tilde{\mathbf{w}})$ . Choose initial value  $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}^0$  and set  $\bar{s} = s(\tilde{\mathbf{w}}^0)$ , where  $\tilde{\mathbf{w}}^0$  is the SVM solution.

At iteration  $l + 1$  ( $l \geq 0$ ), for given  $\tilde{\mathbf{w}}^i$ ,  $i = 0, \dots, l$ :

**Step 2:** Vertex enumeration and discrete minimization. Construct  $E^l(\tilde{\mathbf{w}})$  based on  $\tilde{\mathbf{w}}^i$ ;  $i = 1, \dots, l$ . Set  $\underline{s} = \min_{\tilde{\mathbf{w}}} E^l(\tilde{\mathbf{w}}) = E^l(\tilde{\mathbf{w}}^{l+1})$ , where  $\tilde{\mathbf{w}}^{l+1}$  is the solution of (4.1), obtained as follows. First, seek all new vertices when a new constraint  $t \geq s_1(\tilde{\mathbf{w}}^l) + \langle \nabla s_1(\tilde{\mathbf{w}}^l), \tilde{\mathbf{w}} - \tilde{\mathbf{w}}^l \rangle$  is added, which is performed via vertex enumeration. Then, find the minimizer  $(t^{l+1}, \tilde{\mathbf{w}}^{l+1})$  of  $t + s_2(\tilde{\mathbf{w}})$  among these new  $(t, \tilde{\mathbf{w}})$ -vertices.

**Step 3:** Redundant constraints. Drop any inactive constraints.

**Step 4:** Stopping rule. If  $\bar{s} - \underline{s} \leq \epsilon$  for prespecified  $\epsilon \geq 0$ , then stop and set  $\hat{\tilde{\mathbf{w}}} = \tilde{\mathbf{w}}^{l+1}$ . Otherwise, construct  $E^{l+1}(\tilde{\mathbf{w}})$  from  $E^l(\tilde{\mathbf{w}})$ . If  $s(\tilde{\mathbf{w}}^{l+1}) < \bar{s}$ , then set  $\bar{s} = s(\tilde{\mathbf{w}}^{l+1})$  and go to iteration  $l + 2$ .

**Theorem 4.** *Algorithm 3 yields an  $\epsilon$ -global minimizer in the sense that the difference between the function value of the minimizer and the global minimizer is no greater than a*

prespecified tolerance bound  $\epsilon$ , provided that  $B^0$  contains the global minimizer  $(\hat{t}, \hat{\mathbf{w}})$ .

The choice of  $B^0$  appears important for the speed of convergence of Algorithm 3. If the region of  $B^0$  is too large, the algorithm converges slowly. On the other hand,  $B^0$  should be sufficiently large to contain an  $\epsilon$ -global minimizer. In practice, we choose  $B^0$  based on consideration of computational cost and prior knowledge about  $(\hat{t}, \hat{\mathbf{w}})$ .

Algorithm 3 has an attractive feature in that it yields a global optimum ( $\epsilon$ -global minimizer). However, vertex enumeration is computationally intensive and is infeasible for large-scale problems. Although the solutions of Algorithms 1 and 2 may not be global, they yield global solutions with probability normally exceeding 60%, as suggested by An and Tao (1997). Most importantly, Algorithms 1 and 2 improve the generalization ability of SVM by correcting the bias introduced by the imposed convexity to  $s_1$ , when the SVM solution is used as an initial value, even though they may not reach an  $\epsilon$ -global minimizer. In practice, Algorithms 1 and 2 are preferable based on the consideration of computational cost.

## 5. NUMERICAL EXAMPLES

### 5.1 SIMULATIONS

In this section, we investigate the performance of Algorithms 1–3 in four simulated examples. Example 1 examines globality of the solution of Algorithm 1 via Algorithm 3, while Examples 2 and 3 compare  $\psi$ -learning with SVM for linear problems, where the solution of  $\psi$ -learning is computed via Algorithm 1. Example 4 is nonlinear and solved by Algorithm 2. In what is to follow, the amount of improvement of  $\psi$ -learning over SVM is measured by

$$(T(\text{SVM}) - T(\psi)) / (T(\text{SVM}) - \text{Bayes error}), \quad (5.1)$$

where  $T(\cdot)$  denotes the testing error of a given method. The testing and Bayes errors are computed via independent testing samples of size 100,000, generated from the same mechanism as its corresponding training sample.

**Example 1:** Linear. A random sample of size  $n = 100$  is generated as follows. First, class label  $y$  is generated with probability of  $1/3$  for each class. Second, for a given  $y = 1, 2, 3$ ,  $x$  is sampled from  $\text{Beta}(\alpha, \beta)$ , where  $(\alpha, \beta) = (1, 5), (2, 2), (5, 1)$ . Figure 2 displays  $P(Y = j | X = x)$ ;  $j = 1, 2, 3$ . The estimated Bayes error is .2553.

To maximize the performance of the proposed method, we perform a grid search to choose the tuning parameter  $C$ . In this example,  $C = 1,000$  is obtained by a grid search for  $\psi$ -learning. For comparison, the SVM solution with the same  $C$  is chosen to be an initial value for both Algorithms 1 and 3, where its cost function value, as defined in (2.2) on the training dataset, is 91355.6 with training error .29 and testing error .2595. Algorithm 1 terminates after 3 iterations with a cost function value of 57418.8, training error .26, and

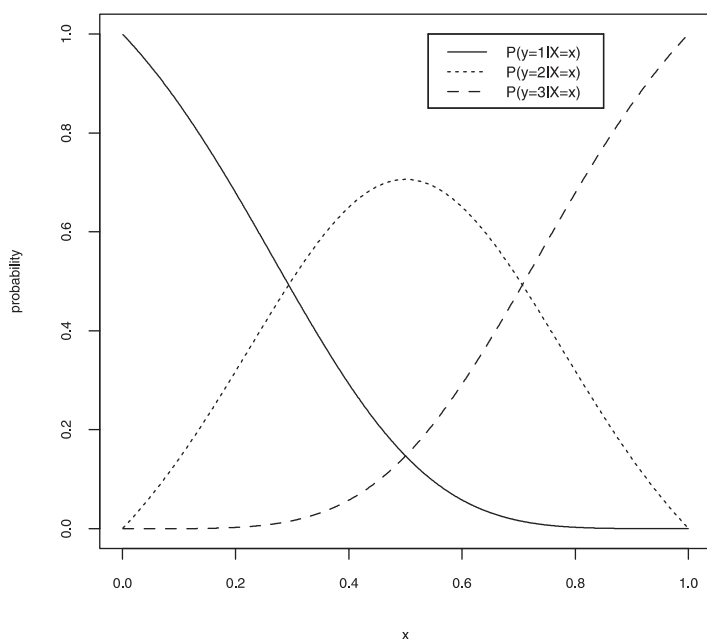


Figure 2. Plot of the conditional probabilities as described in Example 1, represented by solid, dotted, and dashed lines for classes 1–3, respectively. The marginal distribution of  $Y$  has probability  $1/3$  for each class and the conditional distribution of  $X$  given  $Y = j$  follows  $\text{Beta}(\alpha, \beta)$  with  $(\alpha, \beta) = (1, 5), (2, 2), (5, 1)$  for  $j = 1, 2, 3$ , respectively.

testing error .2576. Algorithm 3 terminates after 117 iterations, yielding the global minimum 57265.3 of the cost function in (2.2) with training error .26 and testing error .2564. Figure 3 displays the plot of the estimated  $\max(f_1, f_2, f_3)$  of  $\psi$ -learning using Algorithms 1 and 3. In this example, although Algorithm 1 did not attain the global minimum, its solution is close to the global minimizer obtained by Algorithm 3.

This example reinforces our view that an improvement over SVM can be usually made even when Algorithm 1 or 2 could not reach the global minimum.

**Example 2:** Linear. Random samples of size  $n = 100$  are generated as follows. First, generate  $(t_1, t_2)$  from the standard bivariate  $t$ -distribution with degrees of freedom  $\nu$ , where  $\nu = 1, 3$  in Cases 1 and 2, respectively. Second, randomly assign  $\{1, 2, 3, 4\}$  to its label index for each  $(t_1, t_2)$ . Third, generate  $(x_1, x_2)$  as follows:  $x_1 = t_1/4.5 + a_1$  and  $x_2 = t_2/4.5 + a_2$  with four different values of  $(a_1, a_2) = (0, .5), (.5, 1), (1, .5), (.5, 0)$  for classes 1–4, respectively. We perform a grid search to choose  $C$  in order to eliminate the dependence of the performances of the methods on  $C$ .

The results are obtained by averaging 100 repeated simulations for each case and are reported in Table 1, which indicate that  $\psi$ -learning has a smaller testing error thus better generalization ability. In addition,  $\psi$ -learning has better data reduction ability than SVM. Finally, Figure 4 illustrates how SVM and  $\psi$ -learning perform on one random training sample of size 100. Indeed,  $\psi$ -learning yields a better decision boundary than SVM.

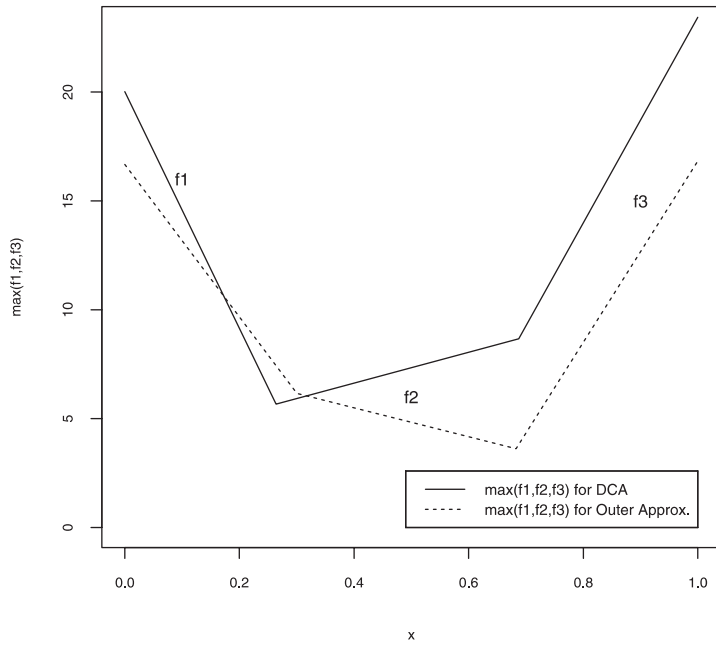


Figure 3. Plot of  $\max(f_1, f_2, f_3)$  estimated by Algorithms 1 and 3 in Example 1, represented by solid and dotted lines, respectively, with  $n = 100$  and  $C = 1,000$ . Here, the cost function values of  $\psi$ -learning are 57418.8 and 57265.3, obtained from Algorithms 1 and 3, respectively.

**Example 3:** Linear. Random samples of size  $n = 100$  are generated as follows. First, generate  $(x_1, x_2)$  according to the uniform distribution over  $[0, 1]^2$ . Assign  $\{1, 2, 3, 4\}$  to its label index for each  $(x_1, x_2)$  when it belongs respectively to one of the four regions: Region 1  $\{x_2 \geq x_1, x_2 \leq (1 - x_1)\}$ , Region 2  $\{x_2 > x_1, x_2 > (1 - x_1)\}$ , Region 3  $\{x_2 \leq x_1, x_2 \geq (1 - x_1)\}$ , and Region 4  $\{x_2 < x_1, x_2 < (1 - x_1)\}$ . Second, randomly select  $\nu$  instances and flip their label indices to the remaining three classes with equal probabilities, where  $\nu = 10\%, 20\%$  for Cases 1 and 2, respectively. Outliers occur due to random flipping. We perform 100 repeated simulations for both cases and the results are reported in Table 2. The results show that  $\psi$ -learning does much better than SVM and its performance is closer to the Bayes rule. This confirms our view that  $\psi$ -learning is more

Table 1. Testing, Training Errors, and Number of SVs of SVM and  $\psi$ -Learning Using the Best  $C$  in Example 2 with  $n = 100$ , Averaged Over 100 Simulation Replications and Their Standard Errors in Parenthesis. In Case 1,  $df = 1$ , the Bayes error is .3015 with the improvement of  $\psi$ -learning over SVM 33.48%. In Case 2,  $df = 3$ , the Bayes error is .1937 with the improvement of  $\psi$ -learning over SVM 21.65%.

Case	Method	Training (s.e.)	Testing (s.e.)	No. SV(s.e.)
df = 1	SVM	.4567(.1260)	.5082(.1071)	96.35(6.34)
	$\psi$ -L	.3839(.1094)	.4390(.0965)	57.17(12.09)
df = 3	SVM	.1799(.0392)	.2034(.0140)	68.13(11.07)
	$\psi$ -L	.1763(.0367)	.2013(.0102)	47.05(15.19)

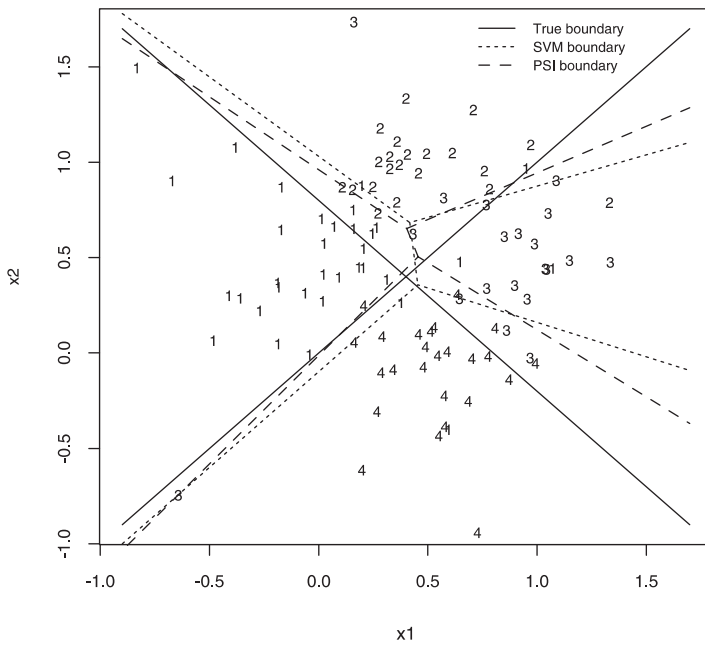


Figure 4. The true, SVM, and  $\psi$ -learning decision boundaries for one training set in Example 2 are represented by solid, dotted, and dashed lines, respectively. The parameter  $C$  is tuned independently for each method. Here Bayes error is .1937. For SVM, the training error is .19 with testing error of .2079. For  $\psi$ -learning, the training error is .16 with testing error of .1987.

robust to outliers than SVM. To view the effects of outliers on both methods, we plot the decision boundaries for one random training sample of size 100 in Figure 5. As expected,  $\psi$ -learning is more robust than SVM to outliers.

**Example 4:** Nonlinear. A random sample of size  $n = 100$  is generated as follows. First, obtain  $(t_1, t_2)$  as in Example 2. Second, randomly assign  $\{1, 2, 3\}$  to the label index for each  $(t_1, t_2)$ . Third, generate  $(x_1, x_2)$  as follows:  $x_1 = t_1/4.5 + a_1$  and  $x_2 = t_2/4.5 + a_2$ , where  $(a_1, a_2) = (.5, 1), (.5, 0)$  for classes 2 and 3, respectively,  $(a_1, a_2)$  is  $(0, .5)$  or  $(1, .5)$  with probability 1/2 for class 1. This generates a three-class nonlinear problem with class 1

Table 2. Testing, Training Errors, and Number of SVs of SVM and  $\psi$ -Learning Using the Best  $C$  in Example 3 With  $n = 100$ , Averaged Over 100 Simulation Replications and Their Standard Errors in Parenthesis. In Case 1 with 10% flipping, the Bayes error is .1 with the improvement of  $\psi$ -learning over SVM 48.29%. In Case 2 with 20% flipping, the Bayes error is .2 with the improvement of  $\psi$ -learning over SVM 75.23%.

Case	Method	Training (s.e.)	Testing (s.e.)	No. SV (s.e.)
10%	SVM	.1298(.0333)	.1615(.0256)	57.92(11.68)
	$\psi$ -L	.1005(.0282)	.1318(.0211)	22.99(9.13)
20%	SVM	.2267(.0444)	.2646(.0339)	76.77(9.12)
	$\psi$ -L	.1819(.0349)	.2160(.0227)	33.16(10.05)

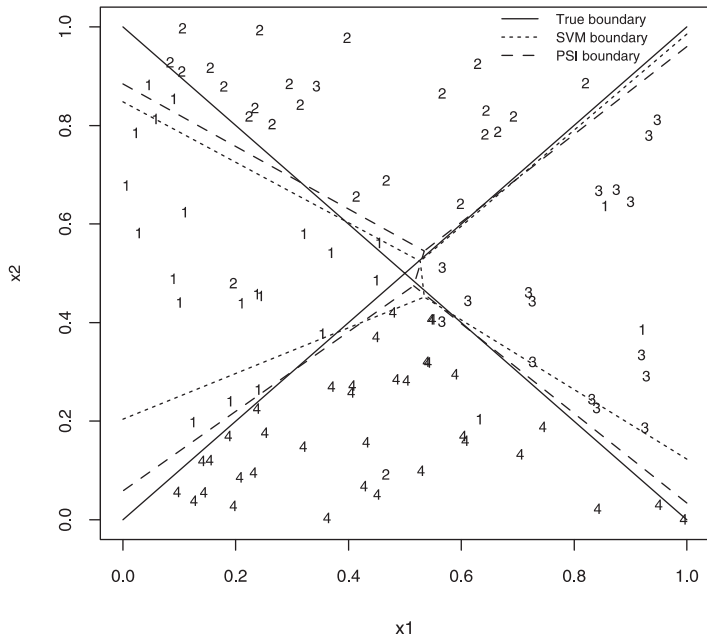


Figure 5. The true, SVM, and  $\psi$ -learning decision boundaries for one training set in Example 3 are represented by solid, dotted, and dashed lines, respectively. The parameter  $C$  is tuned independently for each method. Here Bayes error is .10. For SVM, the training error is .14 with testing error of .1743. For  $\psi$ -learning, training error is .09 with testing error of .1242.

from a mixture of  $t$  distributions. Figure 6 displays the results of SVM and  $\psi$ -learning with Gaussian kernel  $K(s, t) = \exp(-\frac{1}{\sigma^2} \|s - t\|^2)$ , where  $C$  and  $\sigma$  are chosen via a  $20 \times 20$  grid search for each method to maximize its performance. Algorithm 2 converges in four iterations.

As suggested by Figure 6, Algorithm 2 performs well for this nonlinear problem.

In summary, Algorithms 1 and 2 are effective, and  $\psi$ -learning outperforms SVM in the linear cases, with the amount of improvements over SVM ranging from 21.65% to 75.23% in Examples 2 and 3, respectively. In addition, the average number of SVs for  $\psi$ -learning is much smaller than that for SVM. Therefore, comparing to SVM,  $\psi$ -learning yields a more sparse solution. Moreover,  $\psi$ -learning is insensitive to extreme observations while SVM seems quite sensitive. Finally,  $\psi$ -learning performs well in the nonlinear case as shown in Example 4 although the choice of  $C$  and  $\sigma$  becomes more critical in such cases.

## 5.2 APPLICATION

In this section, we study the performance of DCA on the leukemia dataset of Golub et al. (1999). This dataset summarizes a study of gene expression of 7,129 genes in two types of acute leukemias, acute lymphoblastic leukemia (ALL), and acute myeloid leukemia (AML). ALL can be further divided into ALL-B or ALL-T corresponding to B-cell or T-cell

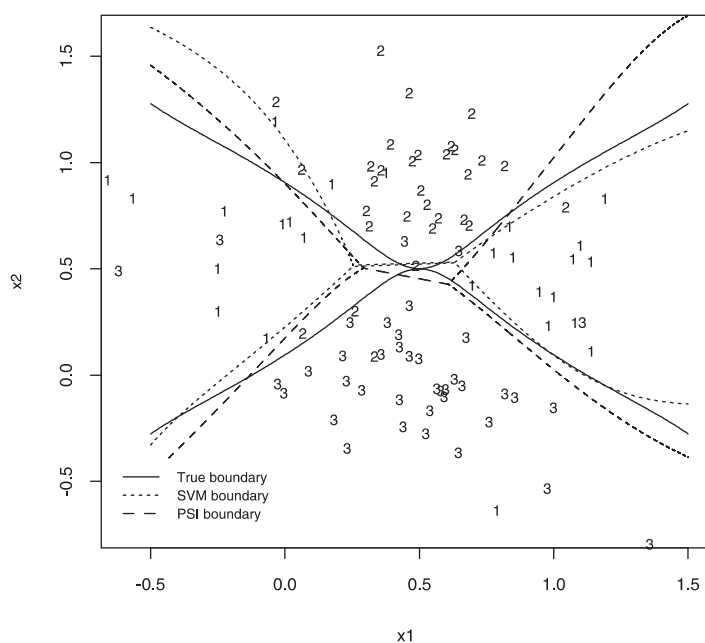


Figure 6. The true, SVM, and  $\psi$ -learning decision boundaries in Example 4 are represented by solid, dotted, and dashed lines, respectively. Parameters  $C$  and  $\sigma$  are tuned independently for each method. Here Bayes error is .176. For SVM, the training error is .15 with testing error of .1914. For  $\psi$ -learning, training error is .13 with testing error of .1857.

lineage. The dataset consists of a training set of size 38 (19 ALL-B, 8 ALL-T, and 11 AML) and a testing set of size 34 (19 ALL-B, 1 ALL-T, and 14 AML).

First, we standardize each sample across genes. Because many genes are nearly constant across tumor samples, we apply the gene selection method proposed by Dudoit, Fridlyand, and Speed (2002), that is, prescreen genes by choosing those genes with largest between-group to within-group sums of squares. For illustration, we choose  $d = 50$  genes to apply Algorithm 1 on the training dataset with  $k = 3$ . To remove the effect of tuning parameter  $C$ , we seek the best performance of  $\psi$ -learning and SVM with respect to a set of discretized  $C$ -values in  $[10^{-3}, 10^3]$ . It turns out that the classifier for  $\psi$ -learning (yielded by Algorithm 1) has one misclassified tumor (Number 67: classify ALL-T as AML), which is smaller than that of SVM with two misclassified tumors (Number 66: classify AML as ALL-B; Number 67: classify ALL-T as AML).

We also applied Algorithm 1 to other number of preselected genes (between 40 and 100) and the results do not alter much. In general, since the complexity of Algorithm 1 or 2 only depends on  $n$  and  $k$ , the computational cost is roughly the same for different  $d$ 's.

## 6. DISCUSSION

This article is devoted to computational developments of multicategory  $\psi$ -learning as well as SVM. For multicategory  $\psi$ -learning, because its cost function to be minimized

is nonconvex and multiple categories are involved, the computational task becomes challenging. Two computational tools are proposed here via d.c. programming, and are based, respectively, on DCA and the outer approximation method. The first one reduces to sequential quadratic programming, while the second one yields the global minimizer via sequential concave minimization. For multicategory SVM, the optimization is performed via quadratic programming as in the binary case.

Although our computational tools perform well in simulations, further investigation is necessary before these tools become applicable in practice, especially for large-scale problems where both the computational complexity and storage are of concern from a computational point of view.

## APPENDIX

**Lemma A.1.** *The classifiers of multicategory  $\psi$ -learning (2.2) or (2.3) yielded by Algorithms 1 or 2 and SVM (2.4) or (2.5) depend only on the “support vectors.”*

**Proof:** Without loss of generality, we only prove the linear case. To treat (2.2), we consider the Karush-Kuhn-Tucker (KKT) complementary conditions

$$\gamma_i \xi_i = 0; \quad i = 1, \dots, n, \quad (\text{A.1})$$

$$\alpha_{ij} [1 - \mathbf{x}_i^T (\mathbf{w}_{y_i} - \mathbf{w}_j) - (b_{y_i} - b_j) - \xi_i / 2] = 0; \quad j = 1, \dots, k, \quad j \neq y_i. \quad (\text{A.2})$$

These equations, together with (A.5), imply that the slack variable  $\xi_i$  is nonzero only when  $\sum_{j \in \{1, \dots, k\} \setminus \{y_i\}} \alpha_{ij} = C$ . It then follows from (A.2) that  $\sum_{j \in \{1, \dots, k\} \setminus \{y_i\}} \alpha_{ij} (f_{y_i} - f_j) < C$ , equivalently,  $\min(\mathbf{g}(\mathbf{x}_i, y_i)) < 1$ . This is to say that instance  $(x_i, y_i)$  falls outside of polyhedron  $D_{y_i}$ . On the other hand, instances with  $0 < \sum_{j \in \{1, \dots, k\} \setminus \{y_i\}} \alpha_{ij} < C$  lie on the boundary of polyhedron  $D_{y_i}$  because  $\xi_i = 0$ , hence that  $\min(\mathbf{g}(\mathbf{x}_i, y_i)) \geq 1$ . Note that the equality  $\min(\mathbf{g}(\mathbf{x}_i, y_i)) = 1$  holds following the fact that there exists a  $j_0 \in \{1, \dots, k\} \setminus \{y_i\}$  such that  $\alpha_{ij_0} > 0$  and  $f_{y_i} - f_{j_0} = 1$ .  $\square$

Instances falling inside polyhedron  $D_{y_i}$  satisfy  $\sum_{j \in \{1, \dots, k\} \setminus \{y_i\}} \alpha_{ij} = 0$  and slack variable  $\xi_i = 0$ . Thus, these instances are not SVs by definition with  $\alpha_{ij} = 0$ ;  $j \in \{1, \dots, k\} \setminus \{y_i\}$ ,  $\nabla \psi_2(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) = 0$ , thus do not affect the solutions. The desired result then follows.

**Proof of Theorem 1:** It suffices to show that  $\sum_{j=0}^k f_j^0(\mathbf{x}) = \sum_{j=1}^k b_j^0 + (\mathbf{w}_j^0)^T \mathbf{x} = 0$  for  $\forall \mathbf{x} \in S$  if  $(\mathbf{b}^0, \mathbf{w}^0)$  is the minimizer of (2.2). To prove this, suppose  $\sum_{j=0}^k f_j^0(\mathbf{x}^*) \neq 0$  for some  $\mathbf{x}^* \in S$ . Now, define new decision functions  $f_j^1(\mathbf{x}) = b_j^1 + (\mathbf{w}_j^1)^T \mathbf{x} = (b_j^0 - \bar{b}^0) + (\mathbf{w}_j^0 - \bar{\mathbf{w}}_j^0)^T \mathbf{x}$ ;  $j = 1, \dots, k$ , where  $\bar{b}^0 = \sum_{j=1}^k b_j^0 / k$  and  $\bar{\mathbf{w}}^0 = \sum_{j=1}^k \mathbf{w}_j^0 / k$ . Clearly, both  $f_j^0(\mathbf{x})$  and  $f_j^1(\mathbf{x})$ ;  $j = 1, \dots, k$ , satisfy the constraint in (2.2), in addition that they give the same values of  $C \sum_{i=1}^n \psi(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i))$ . On the other hand, after substituting  $f_j^1(\mathbf{x})$ ;  $j = 1, \dots, k$ , into the norm part of the cost function, we obtain that  $\sum_{j=1}^k \langle \mathbf{w}_j^0 - \bar{\mathbf{w}}^0, \mathbf{w}_j^0 - \bar{\mathbf{w}}^0 \rangle = \sum_{j=1}^k \langle \mathbf{w}_j^0, \mathbf{w}_j^0 \rangle - 2 \sum_{j=1}^k \langle \mathbf{w}_j^0, \bar{\mathbf{w}}^0 \rangle + k \langle \bar{\mathbf{w}}^0, \bar{\mathbf{w}}^0 \rangle = \sum_{j=1}^k \langle \mathbf{w}_j^0, \mathbf{w}_j^0 \rangle - k \langle \bar{\mathbf{w}}^0, \bar{\mathbf{w}}^0 \rangle \leq \sum_{j=1}^k \langle \mathbf{w}_j^0, \mathbf{w}_j^0 \rangle$ . The inequality holds because  $\sum_{j=0}^k f_j^0(\mathbf{x}^*) \neq 0$  for some  $\mathbf{x}^* \in S$ . Thus,

$(\mathbf{b}^1, \mathbf{w}^1)$  yields a smaller value of the cost function than that of  $(\mathbf{b}^0, \mathbf{w}^0)$ . This contradicts the assumption that  $(\mathbf{b}^0, \mathbf{w}^0)$  is the minimizer of (2.2), hence yields the desired result.  $\square$

**Proof of Theorem 2:** To derive its dual problem of (3.3), we introduce Lagrange multipliers  $\gamma_i \geq 0$ ,  $\alpha_j = (\alpha_{1j}, \dots, \alpha_{nj}) \in R^n$  with  $\alpha_{ij} \geq 0$ , and  $\delta \in R^n$ ;  $i = 1, \dots, n$ ,  $j = 1, \dots, k$ , respectively, for (3.4), (3.5), and (3.6). Then, the primal problem (3.3) is equivalent to the following dual problem:

$$\begin{aligned} \max_{\gamma, \alpha, \delta} L_D &= \frac{1}{2} \sum_{j=1}^k \langle \mathbf{w}_j, \mathbf{w}_j \rangle + C \sum_{i=1}^n \xi_i - \sum_{j=1}^k \langle \nabla \mathbf{w}_j^l, \mathbf{w}_j \rangle \\ &\quad - \langle \nabla \mathbf{b}^l, \mathbf{b} \rangle + \delta^T \sum_{j=1}^k (b_j \mathbf{1}_n + X \mathbf{w}_j) \\ &\quad - \sum_{i=1}^n \gamma_i \xi_i + \sum_{i=1}^n \sum_{j=1, j \neq y_i}^k 2\alpha_{ij} [1 - \mathbf{x}_i^T (\mathbf{w}_{y_i} - \mathbf{w}_j) - (b_{y_i} - b_j) - \xi_i / 2], \end{aligned} \quad (\text{A.3})$$

subject to

$$\alpha_{ij}(1 - I(y_i = j)) \geq 0; \quad j = 1, \dots, k, \quad \gamma_i \geq 0; \quad i = 1, \dots, n, \quad (\text{A.4})$$

$$\frac{\partial L_D}{\partial \xi_i} = C - \gamma_i - \sum_{j=1}^k \alpha_{ij}(1 - I(y_i = j)) = 0; \quad i = 1, \dots, n, \quad (\text{A.5})$$

$$\frac{\partial L_D}{\partial \mathbf{w}_j} = \mathbf{w}_j - \nabla \mathbf{w}_j^l + \sum_{i=1}^n 2\alpha_{ij} \mathbf{x}_i - \sum_{i=1}^n I(y_i = j) \sum_{u=1}^k 2\alpha_{iu} \mathbf{x}_i + X^T \delta = 0, \quad (\text{A.6})$$

$$\frac{\partial L_D}{\partial b_j} = -\nabla b_j^l + \sum_{i=1}^n 2\alpha_{ij} - \sum_{i=1}^n I(y_i = j) \sum_{u=1}^k 2\alpha_{iu} + \delta^T \mathbf{1}_n = 0; \quad j = 1, \dots, k. \quad (\text{A.7})$$

After substituting (A.5)–(A.7) into (A.3), we simplify  $L_D$  as  $-\frac{1}{2} \sum_{j=1}^k \langle \mathbf{w}_j, \mathbf{w}_j \rangle + \sum_{i=1}^n \sum_{j=1}^k \alpha_{ij}(1 - I(y_i = j))$ . Clearly,  $\alpha_{ij}$ 's become redundant when  $y_i = j$ ;  $i = 1, \dots, n$ ,  $j = 1, \dots, k$ . For simplicity, we retain redundant multipliers and remove them later. By definition, we have  $\alpha_j = U_j \beta^0$ ,  $\sum_{j=1}^k \alpha_j = U \beta^0$  and  $\delta = U_{k+1} \beta^0$ ;  $j = 1, \dots, k$ . Using (A.6) and relations  $Q_j^0 = 2U_j - 2V_j U + U_{k+1}$ , we have  $\mathbf{w}_j = \nabla \mathbf{w}_j^l - X^T Q_j^0 \beta^0$ ;  $j = 1, \dots, k$ . After some calculations, (A.3) reduces to  $\min_{\beta^0} L_D = \frac{1}{2} \beta^{0T} H^0 \beta^0 + \mathbf{g}^{0T} \beta^0$  with  $H^0 = \sum_{j=1}^k Q_j^0 X X^T Q_j^0$  and  $\mathbf{g}^0 = \sum_{j=1}^k (-2U_j^T (I_n - V_j) \mathbf{1}_n) - \sum_{j=1}^k Q_j^{0T} X \nabla \mathbf{w}_j^l$ . Because the  $r_m$ th element of  $\beta^0$  corresponds to the redundant  $\alpha_{ij}$  satisfying  $j(n-1) + i = r_m$ , we then remove the redundant Lagrange multipliers by deleting the  $r_m$ th element of  $\beta^0$ , the  $r_m$ th column of  $Q_j^0$  and the  $r_m$ th element of  $\mathbf{g}^0$ ;  $m = 1, \dots, n$ . Hence, (A.3) reduces to  $\min_{\beta} L_D = \frac{1}{2} \beta^T H \beta + \mathbf{g}^T \beta$ . Similarly, (A.5) becomes the inequality constraint  $A\beta \leq C \mathbf{1}_n$  after removing the nonnegative Lagrange multipliers  $\gamma_i$ ;  $i = 1, \dots, n$ , (A.7) and (A.4) reduce to  $\mathbf{1}_n^T Q_j \beta = \nabla b_j^l$ ;  $j = 1, \dots, k$ , and  $B\beta \geq \mathbf{0}_{n(k-1)}$ , respectively.

Moreover, (A.6) yields that  $\mathbf{w}_j = \nabla \mathbf{w}_j^l - X^T Q_j \beta$ ;  $j = 1, \dots, k$ . The desired result then follows.  $\square$

**Proof of Theorem 3:** Because the cost function  $s$  satisfies  $0 \leq C \sum_{i=1}^n \psi(\mathbf{g}(\mathbf{f}(\mathbf{x}_i), y_i)) \leq 2nC$ , the minimum of  $s$  is finite. The result then follows from Theorem 6 of An and Tao (1997).  $\square$

**Proof of Theorem 4:** The result follows from the nondecreasing lower approximations as well as the nonincreasing upper bounds of the objective function.  $\square$

## ACKNOWLEDGMENTS

The authors thank the editor, the AE, and the referees for their helpful comments. The research is supported in part by NSF Grant IIS-0328802, DMS-0072635, and NSA grant MDA904-03-1-0021.

[Received April 2003. Revised May 2004.]

## REFERENCES

- 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.
- Avis, D. (2000), "A Revised Implementation of the Reverse Search Vertex Enumeration," in *Polytopes-Combinatorics and Computation*, eds. G. Kalai and G. Ziegler, Boston: Birkhauser-Verlag, DMV Seminar Band 29, pp. 177–198.
- Blanquero, R., and Carrizosa, E. (2000), "On Covering Methods for d.c. Optimization," *Journal of Global Optimization*, 18, 265–274.
- 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, pp. 142–152.
- Dudoit, S., Fridlyand, J., and Speed, T. (2002), "Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data," *Journal of the American Statistical Association*, 97, 77–87.
- Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999), "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring," *Science*, 286, 531–537.
- Kimeldorf, G., and Wahba, G. (1971), "Some Results on Tchebycheffian Spline Functions," *Journal of Mathematical Analysis and Applications*, 33, 82–95.
- Liu, Y., and Shen, X. (2004), "Multicategory  $\psi$ -Learning," manuscript.
- Marron, J. S., and Todd, M. J. (2002), "Distance Weighted Discrimination," Technical Report No. 1339, School of Operations Research and Industrial Engineering, Cornell University.
- Mason, L., Bartlett, P., and Baxter, J. (2000), "Improved Generalization Through Explicit Optimization of Margins," *Machine Learning*, 38, 243–255.
- Shen, X., Tseng, G. C., Zhang, X., and Wong, W. H. (2003), "On  $\psi$ -Learning," *Journal of the American Statistical Association*, 98, 724–734.
- Wahba, G. (1998), "Support Vector Machines, Reproducing Kernel Hilbert Spaces, and Randomized GACV," Technical report 984, Department of Statistics, University of Wisconsin.
- Vanderbei, R. J. (1994), "Loqo: An Interior Point Code for Quadratic Programming," technical report, Program in Statistics & Operations Research, Princeton University.
- Vapnik, V. (1998), *Statistical Learning Theory*, Chichester, UK: Wiley.