

# Computational developments of $\psi$ -learning\*

Sijin Liu  
Department of Statistics  
The Ohio State University  
Columbus, OH 43210

Xiaotong Shen  
School of Statistics  
University of Minnesota  
Minneapolis, MN 55455

Wing Hung Wong  
Department of Statistics  
Stanford University  
Stanford, CA 94305

## Summary

One central problem in science and engineering is predicting unseen outcome via relevant knowledge gained from data, where accuracy of generalization is the key. In the context of classification, we argue that higher generalization accuracy is achievable via  $\psi$ -learning, when a certain class of non-convex rather than convex cost functions are employed. To deliver attainable higher generalization accuracy, we propose two computational strategies via a global optimization technique—difference convex programming, which relies on a decomposition of the cost function into a difference of two convex functions. The first strategy solves sequential quadratic programs. The second strategy, combining this with the method of Branch-and-Bound, is more computationally intensive but is capable of producing global optima. Numerical experiments suggest that the algorithms realize the desired generalization ability of  $\psi$ -learning.

\* Shen's research is supported in part by NSF Grant IIS-0328802. Wong's research is supported by a grant from NSF-DMS.

Key words: DC programming, global optimization, sequential quadratic programming, support vectors.

## 1 Introduction

How does one accurately predict unseen outcome using relevant information? The simplest problem of this kind is classification. In statistical and computer sciences, classification has been particularly vital. Margin-based classification techniques are at the core of progress. Essentially all these techniques construct a classifier by minimizing a convex cost function. Examples include support

vector machines (SVM, Boser, Guyon and Vapnik, 1992; Cortes and Vapnik, 1995), import vector machines (Zhu and Hastie, 2004), among others.

**Key issues: Convex vs Non-convex.** While the margin-based techniques have proven effective and have achieved state-of-the-art performance, the recent work of Shen, Tseng, Zhang, and Wong (2003) suggests that substantial higher generalization accuracy can be achieved if one steps out from the paradigm of convexity; see also Lin (2000, 2002). Specifically, they constructed a certain class of  $\psi$  cost functions and showed that  $\psi$ -learning realizes sharp generalization error rates in some examples. The rationale behind their methodology is that classification is non-convex in nature and ultimately should be treated via non-convex cost functions. Indeed, any convex cost function is generally bound to suffer a loss in generalization accuracy as the price for easing computation (Fung and Mangasarian, 2000). An important practical issue then is how to meet the computational challenge of non-convex minimization.

**Global optimization.** On the basis of recent advances in global optimization, we develop computational tools for  $\psi$ -learning. This allows us to realize the generalization ability of  $\psi$ -learning in practice. The key ingredient of our proposed methods is difference convex (DC) programming, which uses a decomposition of the cost function into a difference of two convex functions. In our decomposition, the leading convex function is an equivalent SVM cost function, while the trailing one can be thought of as a correction to the SVM cost function so that the result is closer to the true generalization error. With this decomposition, a sequence of monotone approximations to the SVM cost function are constructed, and a sequence of quadratic programs are solved to yield an approximate

solution. We develop an efficient algorithm and refer this as SQP. This algorithm can be further enhanced by combining with the branch-and-bound (BB) search to obtain a provable convergence to global optima, which we call SQP-BB.

Numerical experiments suggest SQP performs well for large problems, whose termination requires only a small number of iterations, for instance, 4-7 iterations would be common. We use SQP-BB to check globality of the solutions of SQP in some examples. The result indicates that SQP yields global optima with high likelihood of occurrence.

Six benchmark examples are examined using SQP-BB. Computational results demonstrate that the significant generalization advantage of  $\psi$ -learning is realized by the computational tools developed here. For every single example,  $\psi$ -learning yields higher generalization accuracy than SVM.

## 2 $\psi$ -learning

From a statistical perspective, training data  $(x_i, y_i)_{i=1}^n$  are sampled from a true yet unknown probability distribution, with  $y_i = \pm 1$  in binary classification.

Linear classification uses hyperplanes  $f(\tilde{x}) = \langle w, \tilde{x} \rangle$  as decision functions, with  $\tilde{x} = (x, 1)$  and  $w = (w^*, w_{d+1}) = (w_1, \dots, w_d, w_{d+1}) \in \mathcal{R}^{d+1}$ . Here  $\langle \cdot, \cdot \rangle$  represents the inner product in the corresponding Euclidean space. The basic form of linear SVM, originated from the optimal separating hyper-plane in the separable case, minimizes a convex cost function:  $s(w) = \frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \psi_{svm}(y_i f(\tilde{x}_i))$ , where  $\psi_{svm}(z) = (1 - z)_+$  is the hinge loss, and  $z_+$  represents the positive part of  $z$ . Instead of using  $\psi_{svm}$ , linear  $\psi$ -learning seeks  $w$  to minimize

$$s(w) = \frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \psi(y_i f(\tilde{x}_i)), \quad (1)$$

where  $C > 0$  controls the balance between the margin and training, and  $\frac{2}{\|w^*\|^2}$  is the geometric margin in the separable case. Here  $\psi$  is required to satisfy the property:

$$\begin{aligned} U \geq \psi(z) &> 0 && \text{if } z \in (0, \tau] \\ \psi(z) &= U(1 - \text{Sign}(z)) && \text{otherwise,} \end{aligned} \quad (2)$$

where  $0 < \tau \leq 1$  and  $U > 0$  are some constants.

In implementation, a specific choice of  $\psi$  should be chosen depending on one's optimization strategy. In what follows, we shall use a  $\psi$  function, defined as  $\psi(z) = 0$  if  $z \geq 1$ ,  $\psi(z) = 2(1 - z)$  if  $0 \leq z \leq 1$ , and 2 otherwise. This  $\psi$  function, as displayed in Figure 2, has the desirable DC property in that it has a DC representation. This property is the key to develop efficient computational algorithms. Since no differentiability is used in our algorithms, there is no obvious advantage of applying a smooth version of  $\psi$ . See Shen et. al (2003) for some discussions with regard to the choice of  $\psi$ .

Nonlinear classification uses flexible representations, with  $f(\tilde{x}) = \langle \tilde{x}, w \rangle$ ,  $w = (w^*, w_{n+1}) = (w_1, \dots, w_{n+1}) \in \mathcal{R}^{n+1}$ , and  $\tilde{x} = (K(x, x_1), \dots, K(x, x_n), 1)$ , defined by kernel  $K(\cdot, \cdot)$  mapping from  $S \times S$  to  $\mathcal{R}$ . The kernel is required to satisfy Mercer's condition, which assures that  $\|w^*\| = (\sum_{i=1}^n \sum_{j=1}^n w_i w_j K(x_i, x_j))^{1/2}$  is a proper norm. The theory of reproducing kernel Hilbert space (RKHS), c.f., Wahba (1990, 1999), is useful to construct such a kernel. Then the cost function of nonlinear  $\psi$ -learning becomes

$$s(w) = \frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \psi(y_i f(\tilde{x}_i)). \quad (3)$$

In the sequel, we shall adopt a generic form  $f(\tilde{x}) = \langle \tilde{x}, w \rangle$  and the norm  $\|w^*\| = \langle w^*, w^* \rangle$  to represent both the linear and nonlinear cases with  $w$  being respectively  $d+1$  and  $n+1$  dimensional. The estimated decision function of  $\psi$ -learning is then  $\hat{f}(\tilde{x}) = \langle \hat{w}, \tilde{x} \rangle$ , where  $\hat{w}$  is the minimizer of (1) or (3).

## 3 Non-convex minimization

For high-dimensional non-convex minimization, there is generally no efficient method to compute global optima. Figure 1 illustrates the level of difficulty of optimization in (1) and (3). Fortunately, by exploiting the DC property of the  $\psi$ -function, we are able to develop efficient algorithms.

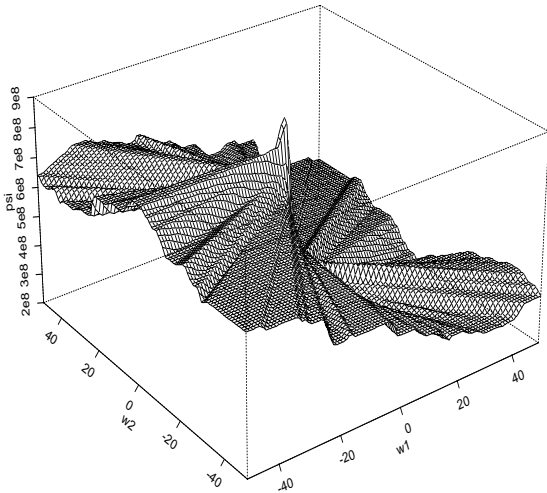


Figure 1: Perspective plot of  $s$  as a function  $(w_1, w_2)$  and  $w_3 = 0.25$  for the example described in Section 4 with  $n = 50$ ,  $C = 10^7$  and 20% flipping.

### 3.1 DC decompositions

There have been major advances in computation of global optima when an objective function has a DC representation (An and Tao, 1997). Such a decomposition plays an **extremely critical** role in determining the speed of convergence, stability, robustness, and globality of sought solutions. For our problem, we utilize the problem structure and decompose our cost function  $s$  in (1) or (3) into:

$$s(w) = s_1(w) - s_2(w), \quad (4)$$

where  $s_1 = \frac{1}{2}\|w^*\|^2 + C \sum_{i=1}^n \psi_1(y_i f(\tilde{x}_i))$  and  $s_2 = C \sum_{i=1}^n \psi_2(y_i f(\tilde{x}_i))$  are both convex in  $w$ . This decomposition is obtained from a DC decomposition of  $\psi(z) = \psi_1(z) - \psi_2(z)$ , where  $\psi_1(z)$  is 0 if  $z \geq 1$  and  $-2(z-1)$  otherwise;  $\psi_2(z)$  is 0 if  $z \geq 0$  and  $-2z$  otherwise. The plot of this decomposition is given in Figure 2. Note that  $s_1$  is equivalent to the SVM cost function induced by twice of the hinge loss. Further, since the true generalization error is defined by a bounded loss function  $1 - \text{Sign}$ , the unbounded nature of the SVM cost function introduces obvious bias when it is used to estimate

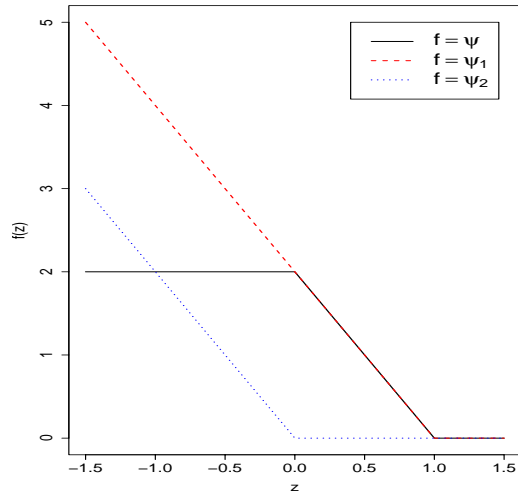


Figure 2: Plot of functions  $\psi_1$ ,  $\psi_2$  and  $\psi$ , where  $\psi = \psi_1 - \psi_2$  is a DC decomposition of  $\psi$ .

the generalization error. In light of this, we interpret  $s_2$  in (4) as a bias correction to the SVM cost function.

### 3.2 Differenced Convex Algorithms

Differenced convex algorithm (DCA) is among the rare algorithms which allow to solve large-scale non-convex minimization problems. As shown in An and Tao (1997), when a DC decomposition is available, DCA constructs non-increasing upper envelopes of  $s$ , which yield sequential convex subproblems. This permits developing efficient algorithms for  $\psi$ -learning, especially so for large-scale problems.

There are basically two versions of DCA, regular and simplified, and we apply the simplified DCA. When (4) is given, the simplified DCA solves a sequence of primal and dual subproblems. It proceeds with construction of two sequences  $(w^{(k)}, y^{(k)})$  iteratively. Given  $(w^{(k)}, y^{(k)})$ , the  $k$ th primal subproblem is  $s_1(w) - s_2(w^{(k)}) - \langle w - w^{(k)}, y^{(k)} \rangle$ , obtained by replacing  $s_2$  by its affine minorization function  $s_2(w^{(k)}) + \langle w - w^{(k)}, y^{(k)} \rangle$ . Minimizing it with respect to  $w$  yields  $w^{(k+1)}$ . Similarly,  $y^{(k+1)}$  is the minimizer of the  $k$ th dual subproblem after obtaining  $w^{(k+1)}$ , which amounts to selecting a suitable sub-

gradient of  $s_2$  at  $w^{(k)}$ . By convexity, these subproblems provide a sequence of non-increasing upper approximations to the original problem, leading to convergence of  $w^{(k)}$ .

In our case, we derive a subgradient of  $s_2$  at  $w^{(k)}$  without solving the dual problem. Specifically, this subgradient  $\nabla s_2(w^{(k)})$  is defined as  $(V_1^{(k)}, V_2^{(k)})$ , where  $V_1^{(k)}$  is  $C \sum_{i=1}^n \nabla \psi_2(y_i f^{(k)}(\tilde{x}_i)) y_i x_i$ ,  $V_2^{(k)}$  is  $C \sum_{i=1}^n \nabla \psi_2(y_i f^{(k)}(\tilde{x}_i)) y_i$ , and  $f^{(k)}(\tilde{x}_i)$  is  $\langle w^{(k)}, \tilde{x}_i \rangle$ . Here  $\nabla \psi_2(z) = 0$  if  $z > 0$  and  $\nabla \psi_2(z) = -2$  otherwise.

Our algorithm solves a sequence of subproblems. At iteration  $k$ , only the primal subproblem is required to solve, which is equivalent to

$$\min_w (s_1(w) - \langle w, \nabla s_2(w^{(k)}) \rangle). \quad (5)$$

This problem can be solved via quadratic programming (QP). By Kuhn-Tucker (KKT)'s condition, it is equivalent to the dual QP in Theorems 1 and 2.

**Theorem 1:** (Linear) The  $k$ th dual subproblem of (5) with  $\alpha = (\alpha_1, \dots, \alpha_n)$  is

$$\begin{aligned} \max_{\alpha} W(\alpha) &= \sum_{i=1}^n \alpha_i [1 - y_i \langle V_1^{(k)}, x_i \rangle] \\ &\quad - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle, \end{aligned} \quad (6)$$

subject to  $\sum_{i=1}^n \alpha_i y_i = -V_2^{(k)}$ ,  $2C \geq \alpha_i \geq 0$ ;  $i = 1, \dots, n$ . Then the solution of (5)  $(w_1^{(k+1)}, \dots, w_d^{(k+1)})$  is  $V_1 + \sum_{i=1}^n \alpha_i^{(k)} y_i x_i$ ,  $w_{d+1}^{(k+1)}$  satisfies KKT's condition:  $y_i \langle w^{(k+1)}, \tilde{x}_i \rangle = 1$  for any  $i$  with  $2C > \alpha_i^{(k)} > 0$ . Here  $\{\alpha_i^{(k)}\}_{i=1}^n$  is the solution of (6).

**Theorem 2:** (Nonlinear) The dual subproblem of (5) is (6) with  $\langle x_i, x_j \rangle$  being replaced by  $K(x_i, x_j)$  and  $\langle V_1^{(k)}, x_i \rangle$  being replaced by  $C \sum_{j=1}^n \psi_2'(y_j f^{(k)}(\tilde{x}_j)) y_j K(x_i, x_j)$ . The solution  $\{\alpha_i^{(k)}\}_{i=1}^n$  yields that of (5)  $w_j^{(k+1)} = y_j (\alpha_j^{(k)} + C \nabla \psi_2(y_j f^{(k)}(\tilde{x}_j)))$ ;  $j = 1, \dots, n$ , and  $w_{n+1}^{(k+1)}$  satisfies KKT's condition:  $y_i \langle w^{(k+1)}, \tilde{x}_i \rangle = 1$  for any  $i$  with  $2C > \alpha_i^{(k)} > 0$ .

**Algorithm 1: (SQP, Linear and Nonlinear)**

**Step 1:** (Initialization) Specify initial value  $w^{(0)}$  and tolerance error  $\varepsilon > 0$ .

**Step 2:** (Iteration) At iteration  $k$ , compute  $w^{(k+1)}$  by solving (6).

**Step 3:** (Stopping rule) Stop if  $|s(w^{(k+1)}) - s(w^{(k)})| \leq \varepsilon$ . Then the final solution  $w^s$  is  $w^{(k+1)}$ , which yields  $\hat{f}(\hat{x}) = \langle w^s, \tilde{x} \rangle$  for (1) or (3).

Our numerical experience suggests that a good initial value enhances the chance of DCA to locate global optima. For both Algorithms 1 and 2, we recommend to use a SVM solution or any point with a smaller cost function value.

Two important features are built into SQP to guard against potential numerical problems and enhance its stability. First, linear programming (LP) is employed for  $w_{d+1}$  or  $w_{n+1}$  when there are no instances  $2C > \hat{\alpha}_i > 0$  such that they can be determined by KKT's condition. Specifically, minimize (5) with respect to  $w_{d+1}$  or  $w_{n+1}$  via LP after substituting the values of  $\{\alpha_i^{(k)}\}_{i=1}^n$  in  $f^{(k+1)}$ . Second, a regularization technique is applied to replace the leading matrix  $K$  in QP by  $K + \rho I$  for small  $\rho > 0$  when it becomes ill-posed, although  $K$  is supposed to be positive definite. This regularization technique is equivalent to a different DC decomposition:  $s = (s_1 + \rho) \|\cdot\|^2 - (s_2 + \rho) \|\cdot\|^2$ , for some  $\rho > 0$  for improving the strength of convexity of the decomposition.

**Theorem 3:** (Convergence) The sequence  $s(w^{(k)})$  is non-increasing,  $\lim_{k \rightarrow \infty} s(w^{(k)}) \geq \min_w s(w)$ , and  $\lim_{k \rightarrow \infty} \|w^{(k+1)} - w^{(\infty)}\| = 0$  for some  $w^{(\infty)}$ . Moreover, convergence of SQP is superlinear in that  $\lim_{k \rightarrow \infty} \|w^{(k+1)} - w^{(\infty)}\| / \|w^{(k)} - w^{(\infty)}\| = 0$  provided that there does not exist an instance  $x^*$  on the decision boundary such that  $f^{(\infty)}(x^*) = \langle w^{(\infty)}, \tilde{x}^* \rangle = 0$ .

As shown in Theorem 3, SQP converges superlinear in that the number of iterations required for it to achieve precision  $\varepsilon$  is  $o(\log(1/\varepsilon))$ . Based on our numerical experience, it normally terminates in 4-10 steps. The computational complexity of  $o(\log(1/\varepsilon))$  multiplied by that of QP, which is usually  $O(n^3)$ .

An improvement over SVM in generalization usually occurs even when global optima have not been reached by SQP; see Tables 1 and 2. This is mainly because  $s_2$  corrects the bias due to imposed convexity to  $s_1$  in (4). This aspect has been confirmed by our numerical experi-

ence.

### 3.3 DCA and Branch-and-Bound

The method of BB can be used to globally solve minimization in (1) and (3). When it is suitably combined with SQP, it leads to a promising global minimization routine, which can substantially improve efficiency of BB and enhance globality of DCA. In what follows, we shall derive such an algorithm.

BB is composed of two critical operations: bounding and subdivision. The bounding operation constructs both upper and lower bounds of  $s$ , while the subdivision operation divides regions. A combination of both exclude infeasible regions and determine optimality of a solution. Because SQP tends to yield a sharp upper bound when it does not give global optima, convergence of BB expedites.

For the bounding operation, we obtain a good upper bound via the solution  $w^s$  of SQP in that  $s(w^s) \geq \min_w s(w)$ . To construct a good lower bound, we first construct a tight convex envelop of the concave function  $-s_2$  in (4) over a simplex  $S$  using a result of Falk and Hooeman (1976). Specifically in the linear case, let  $v_j \in V(S)$ ;  $j = 1, \dots, d+2$ , be a collection of vertices of  $S$ . The convex envelop  $l_s(w)$  of  $-s_2(w)$  is  $\langle a, w \rangle + a_{d+2}$ , obtained by solving a linear system of  $(d+2)$  equations with respect to  $(a = (a_1, \dots, a_{d+1}, a_{d+2}))$ :  $l_s(v_j) = -s_2(v_j)$ ;  $j = 1, \dots, d+2$ . By concavity,  $-s_2(w) \geq l_s(w)$  for  $w \in S$ . Second, solve a quadratic problem:

$$L(S) = \min_{w \in S} (s_1(w) + l_s(w)), \quad (7)$$

yielding a lower bound  $L(S)$ , which is equivalent to solving the following problem.

**Theorem 4:** (Lower bound, Linear) The dual problem of (7) is

$$\min_z W(z) = \frac{1}{2} z^T H z - g^T z \quad (8)$$

subject to  $2C \geq \alpha_i \geq 0$ ;  $i = 1, 2, \dots, n$ ,  $\beta \geq 0$ ,  $\theta \geq 0$ ,  $t \geq 0$ , and  $\sum_{i=1}^n \alpha_i y_i + \beta^T A_2 + \theta A_4 - t a_6 = a_2$ . The solution  $(\tilde{w}_1, \dots, \tilde{w}_d)$  of (7) is  $\sum_{i=1}^n \hat{\alpha}_i y_i x_i + A_1^T \hat{\beta} + \hat{\theta} A_3^T - a_1 - \hat{t} a_5$ , and  $\tilde{w}_{d+1}$  is chosen so that  $y_i \langle \tilde{w}, \tilde{x}_i \rangle = 1$  for any  $i$  with  $2C > \hat{\alpha}_i > 0$ . Here  $(\hat{\alpha}_1, \dots, \hat{\alpha}_n, \hat{\beta}, \hat{\theta}, \hat{t})$

is the solution of (8), and  $H, g, A_i$ ;  $i = 1, \dots, 4$ ,  $a_i$ ;  $i = 1, \dots, 6$ , are defined in the Appendix.

For the subdivision operation, we use a simplicial subdivision, and combine bisection via the longest edge with the radial partition, c.f., Horst and Tuy (1989). An adaptive partition is possible but will not be studied in here. A simplex subdivision divides  $S$  into  $d+2$  subsimplices  $\{S_i\}_{i=1}^{d+2}$ , and replaces each vertex  $v_i$  of  $S$  by a subdivision point  $v$ , which is the average of the vertices of  $S$  for the radial partition and is the middle point of the longest edge for the longest edge partition. Note that the latter partition only generates two non-degenerate subsimplices.

Our subdivision rule first uses the radial partition, then switches to bisection via the longest edge once the lower bounds become sufficiently good. This allows us to combine the advantages of the both partitions to enhance computational efficiency. Note that the first yields good lower bounds due to division of smaller subsimplices but does not assure convergence, while the second one assures convergence but is inefficient with computation of lower bounds for a large simplex. As a result, computational efficiency is enhanced. In the partition process, switching takes place for  $S$  if one of three conditions is met: 1) Five consecutive partitions have been completed, 2)  $L(S)$  becomes positive, and 3) the shortest edge of  $S$  is less than 10. Here 3) is for numerical stability.

To define an initial simplex  $S_0$  centered at  $w^{(0)} = (w_1^{(0)}, \dots, w_{d+1}^{(0)})$ , we first construct a cube centered at  $w^{(0)}$  with half width  $h$ . Let  $\alpha_j = w_j^{(0)} - h$ ;  $j = 1, \dots, d+1$  and  $\alpha = \sum_{j=1}^{d+1} w_j^{(0)} + (d+1)h$ . This defines the smallest simplex containing the cube:  $S_0 = \{w \in R^{d+1} : \alpha_i - w_j \leq 0, \sum_{j=1}^{d+1} w_j - \alpha \leq 0\}$  with vertices  $v_0 = (\alpha_1, \dots, \alpha_{d+1})$  and  $v_j = (\alpha_1, \dots, \alpha_{j-1}, \alpha - \sum_{i \neq j} \alpha_i, \alpha_{j+1}, \dots, \alpha_{d+1})$ ;  $j = 1, \dots, d+1$ .

Let  $R_k, \epsilon > 0, w^s$ , and  $N$  be a collection of feasible regions at iteration  $k$ , prespecified tolerance error, the final solution, and the upper bound of the numbers of iterations, respectively. Here  $N$  and  $h$  control run time with suitable choice of  $N$  and  $h$  yielding global optima while expedite convergence.

#### Algorithm 2:(SQP-BB)

**Step 1:** (Initialization) Specify  $w^{(0)}$ ,  $h$ ,  $N$ , and  $\epsilon$ . With  $w^{(0)}$  the current best feasible point, compute upper bound  $U(S_0)$  obtained via the solution of SQP with initial value as the solution from lower bound  $L(S_0)$  via the solution of

(7) via (8) with  $S = S_0$ . If  $U(S_0) - L(S_0) \leq \varepsilon(L(S_0) + 1)$ , then  $R_0 = \emptyset$  and terminate; otherwise  $R_0 = S_0$ .

Iteration  $k = 0, \dots$

**Step 2:** (Check optimality) If  $R_k = \emptyset$ , then terminate with  $w^s = w^{(k)}$ .

**Step 3:** (Selection) If  $R_k \neq \emptyset$ , then select  $S_k \in R_k$  such that  $U(S_k) = \min\{U(S) : S \in R_k\}$ .

**Step 4:** (Division) For each  $S_{kj}; j = 1, \dots, j_k$ , apply the subdivision rule to divide  $S_k$  into subsimplices  $S_{kj}$  and compute  $L(S_{kj})$  via (8) with  $S = S_{kj}$ .

**Step 5:** (Updating and elimination) Apply SQP to  $S_{kj}$  using the lower bound solution as an initial value and compute  $U(S_{kj}); j = 1, \dots, j_k$ . Update the current best feasible point  $w^{(k+1)}$  and the best upper bound  $\alpha_{k+1} = \min\{U(S_{kj}) : j = 1, \dots, j_k\}$  at iteration  $k$ . Let  $R_{k+1} = \{S \in \Delta_{k+1} : \alpha_{k+1} - L(S) > \varepsilon(L(S) + 1)\}$ , where  $\Delta_{k+1} = (R_k \setminus S_k) \cup \{S_{kj} : j = 1, \dots, j_k\}$ .

**Theorem 5:** (Convergence) The sequence  $w^{(k)}$  converges to the global minimizer, i.e.,  $\lim_{k \rightarrow \infty} s(w^{(k+1)}) = \min_w s(w)$ . Moreover, SQP-BB terminates finitely with precision  $\varepsilon > 0$  in that  $|s(w^s) - \min_w s(w)| \leq \varepsilon$ .

Theorem 5 says that  $w^s$  is an  $\varepsilon$ -global minimizer. Our numerical experience suggests that SQP-BB usually converges reasonably fast but more slowly than SQP. Computational complexity of SQP-BB is roughly  $o(\log(1/\varepsilon)Nn^3)$ , with an upper bound of  $N$  being of order of  $1/\varepsilon$ . Note that SQP is a special case of SQP-BB when  $h = \infty$  and  $N = 0$ .

As shown in Table 1, even if  $N$  is set to be small, it usually yields better solution than SQP. The computational cost of SQP-BB for linear problems is acceptable, while SQP with a good initial value is recommended for large problems. Interestingly, SQP-BB is parallelizable, permitting fast computation.

## 4 Numerical Analysis

The following numerical example examines the effectiveness of SQP and SQP-BB in terms of speed of convergence and globality of sought solutions. Here the QP and LP involved in SQP and SQP-BB are implemented via the IMSL QP and LP routines.

Consider a two-dimensional linear example of Shen et al. (2003), in which  $f(x) = \sum_{i=1}^2 w_i x_i + w_3$ . A random

Table 1: Globality of SQP and SQP-BB(N) in percent over 100 simulation replications as well as the average number of iterations for SQP.

| C=1               |     |            |                   |                   |
|-------------------|-----|------------|-------------------|-------------------|
| Flip              | SQP | Ave # iter | SQP-BB<br>(N=100) | SQP-BB<br>(N=200) |
| 0%                | 95% | 2.43       | 100%              | 100%              |
| 10%               | 92% | 2.82       | 100%              | 100%              |
| 20%               | 83% | 2.85       | 99%               | 100%              |
| C=10 <sup>3</sup> |     |            |                   |                   |
| Flip              | SQP | Ave # iter | SQP-BB<br>(N=100) | SQP-BB<br>(N=200) |
| 0%                | 99% | 1.20       | 99%               | 99%               |
| 10%               | 46% | 2.07       | 95%               | 97%               |
| 20%               | 29% | 2.10       | 85%               | 93%               |

training sample  $\{X_{i1}, X_{i2}, Y_i\}_{i=1}^n$  is generated as follows. First,  $(X_{i1}, X_{i2})_{i=1}^n$  are sampled from the uniform distribution over the unit disk  $\{(x_1, x_2) : x_1^2 + x_2^2 \leq 1\}$ , and  $Y_i$  is assigned to 1 if  $X_{i1} \geq 0$  and  $-1$  otherwise. Then randomly selected labels  $\{Y_i\}_{i=1}^n$  are flipped, which generates a random sample for non-separable cases.

Three levels of contamination are considered: 0-flip, 10%-flip and 20%-flip, each with two different values  $C = 1, 10^3$ . In each case, the percents of time for SQP to yield global optima based on 100 simulation replications are reported in Table 1. The globality of solutions of SQP is determined by its agreement with the solutions of SQP-BB(N= $\infty$ ), ignoring numerical rounding error that is less than  $\varepsilon > 0$  used by SQP-BB. In the simulations,  $\varepsilon$  is set to be  $10^{-3}C$ .

Numerical analyses for this linear problem indicate that SQP yields global optima with high likelihood, and termination occurs in 2-3 steps on average. It appears that whether it does so is random. This conclusion seems concordant with that of An and Tao (1997) for different problems, where numerical experiments for up to 30-dimensional problems were conducted. Furthermore, BB with N=100 seems to suffice in the case.

## 5 Performance Comparison

In this section we investigate the effectiveness of  $\psi$ -learning via SQP and SQP-BB(N), and compare it to SVM, in both simulated and benchmark examples. A testing error  $T(\cdot)$  is used for any given method, which is averaged over 100 independent testing samples.

For simulation comparisons, we define the amount of improvement of  $\psi$ -learning over SVM as the percent of improvement of in terms of corresponding Bayesian regrets, that is,

$$\frac{(T(SVM) - T(Bayes)) - ((T(\psi) - T(Bayes)))}{T(SVM) - T(Bayes)}, \quad (9)$$

where  $T(SVM)$ ,  $T(\psi)$ , and  $T(Bayes)$  are the testing errors for SVM,  $\psi$ -learning, and the Bayes error, respectively, with  $T(SVM) - T(Bayes)$  and  $T(\psi) - T(Bayes)$  the corresponding Bayesian regrets. This measure seems to be more sensible because a comparison is performed against the baseline error—the Bayes error  $T(Bayes)$ , which is the testing error over a testing sample of large size, say  $10^5$ .

For benchmark comparisons, because  $T(Bayes)$  is unknown, we then define the amount of improvement as

$$\frac{T(SVM) - T(\psi)}{T(SVM)}, \quad (10)$$

which may underestimate the improvement from the baseline error.

### 5.1 Simulation

For  $\psi$ -learning, SQP is applied to Examples 1 and 2 with the corresponding SVM solution as an initial value. To eliminate dependence of the performances of SVM and  $\psi$ -learning on tuning parameters, we perform a grid search to maximize the performances with respect to the tuning parameters.

**Example 1 (Linear):** A random training sample of size  $n = 150$  is generated as follows. First, generate  $(t_1, t_2)$  from the standard bivariate  $t$ -distribution with degree 1. Second, randomly assign  $\pm 1$  to each  $(t_1, t_2)$ . Third, generate  $(x_1, x_2)$  as:  $x_j = t_j + a_j$ ;  $j = 1, 2$ , with  $(a_1, a_2) = ((\sqrt{3.5}, 0.5), (-\sqrt{3.5}, -0.5))$  for positive and negative classes, respectively. In this example,

Table 2: Bayesian regrets defined in (9), for SVM and  $\psi$ -learning as well as the standard errors (in parentheses) in Example 1, minimized over tuning parameter  $C$ . The Bayes error is 15.18%.

|       |          | Testing       | # SV   |
|-------|----------|---------------|--------|
| n=150 | SVM      | 19.04(0.826)% | 109.88 |
|       | $\psi$   | 16.10(0.469)% | 54.05  |
|       | % Improv | 76.2%         |        |

we maximize the performance with respect to  $C$  over an interval  $(0, 10^4]$ , with 9, 9, 9, 9, 99, and 10 uniformly grid points over  $[10^{-3}, 10^{-2}]$ ,  $[10^{-2}, 10^{-1}]$ ,  $[10^{-1}, 1]$ ,  $[1, 10]$ ,  $[10, 10^3]$  and  $[10^3, 10^4]$ , for evaluation, that is,  $10^{-3}i$ ;  $i = 1, \dots, 9$ ,  $10^{-2}i$ ;  $i = 1, \dots, 9$ ,  $10^{-1}i$ ;  $i = 1, \dots, 9$ ,  $i$ ;  $i = 1, \dots, 9, 10i$ ;  $i = 1, \dots, 99$ ,  $10^3i$ ;  $i = 1, \dots, 10$ . The smallest average testing errors as well as the average number of support vectors of SVM and  $\psi$ -learning are summarized in Table 2.

Example 1 shows that  $\psi$ -learning is more robust to outliers than SVM.

**Example 2 (Nonlinear):** A random sample of size  $n = 150, 300$  is generated as follows. First, randomly sample positive and negative class labels. For the positive class, generate  $(x_1, x_2)$  from the standard bivariate  $t$ -distribution with degree 1. For the negative class, randomly generate  $(x_1, x_2)$  from the mixture of the standard bivariate  $t$ -distribution with degree 1 and the standard bivariate normal distribution. Gaussian kernel  $K(x, y) = \exp(-\frac{1}{\sigma^2}\|x - y\|^2)$  is applied to SVM and  $\psi$ -learning. Here the optimal  $C$  is chosen via grid search over interval  $(0, 10^4]$ , to maximize the performance of each method, The grid points are chosen in the same manner as in Example 1. For  $\sigma$ , it is set to be the median distance between the positive and negative classes. This is because  $C$  and  $\sigma^2$  play the similar role, and it is easier to optimize over  $C$  if  $\sigma^2$  is estimated. The numerical results are summarized in Table 3.

As expected,  $\psi$ -learning outperforms SVM as in Example 1. The amount of improvement, however, depends on the sample size. In this nonlinear case, the choice of tuning parameters  $C$  and  $\sigma^2$  appears to be more critical.

In summary,  $\psi$ -learning outperforms SVM in both the linear and nonlinear cases with improvement ranging from 20.5% to 76.2%. In addition, the average number

Table 3: Bayesian regrets defined in (9), for SVM and  $\psi$ -learning as well as the standard errors (in parentheses) in Example 2, minimized over tuning parameters  $C$  and  $\sigma^2$ . The Bayes error is 24.90%.

|       |          | Testing       | # SV   |
|-------|----------|---------------|--------|
| n=150 | SVM      | 29.16(0.467)% | 132.77 |
|       | $\psi$   | 28.29(0.432)% | 87.9   |
|       | % Improv | 20.5%         |        |
| n=300 | SVM      | 27.47(0.361)% | 23.71  |
|       | $\psi$   | 26.40(0.319)% | 15.72  |
|       | % Improv | 41.6%         |        |

of support vectors of  $\psi$ -learning is smaller than that of SVM in all the cases. This suggests that  $\psi$ -learning yields more sparse solutions than SVM. Moreover,  $\psi$ -learning is insensitive to outliers while SVM seems quite sensitive.

## 5.2 Benchmark

We now examine  $\psi$ -learning using SQP-BB( $N=100$ ) and compare it to SVM on 6 different benchmark examples: Wisconsin Breast Cancer (WBC, Wolberg and Mangasarian, 1990), Liver-disorders and Page-Block (the UCI Machine Learning Repository, Murphy and Aha, 1992), and Heart, Breast Cancer (Breast C) and Thyroid (Rätsch, Onoda and Müller, 2001). The examples used here are those reasonable for linear or kernel-based learning. For WBC, Liver and Page-Block examples, we randomly divide each data set into two halves, for testing and training. In the Page-Block example, in particular, we choose the horizontal line and picture classes with 329 and 115 cases respectively to be the binary classes. In the case where the sample size is odd, the size of training is one larger than that of testing. For Heart, Breast C, and Thyroid examples, they are originally not binary classification, hence that a random partition into two classes is applied and are available on <http://mlg.anu.edu.au/~raetsch/data/>, c.f., Rätsch et al. (2001).

For the same randomly selected training and testing sets, SVM and  $\psi$ -learning are compared, where SQP-BB( $N=100$ ) is used for  $\psi$ -learning, with  $N = 100$  and  $h = 25$  for linear training. Their performances averaged over these 100 randomly selected pairs are compared, which are minimized over  $C$  in interval  $(0, 10^4]$ .

Table 4: Averages of testing errors of linear SVM and  $\psi$ -learning as well as the standard errors (in parentheses), minimized over tuning parameter  $C$  in the five benchmark examples.

| Data                       |          | Testing      | # SV   |
|----------------------------|----------|--------------|--------|
| Obs $\times$ Dim           |          |              |        |
| WBC<br>682 $\times$ 9      | SVM      | 3.48(0.05)%  | 30.71  |
|                            | $\psi$   | 3.05(0.04)%  | 16.26  |
|                            | % Improv | 12.4%        |        |
| Liver<br>345 $\times$ 6    | SVM      | 32.00(0.29)% | 123.46 |
|                            | $\psi$   | 30.38(0.28)% | 51.69  |
|                            | % Improv | 5.1%         |        |
| Heart<br>270 $\times$ 13   | SVM      | 16.99(0.30)% | 60.35  |
|                            | $\psi$   | 16.58(0.33)% | 32.80  |
|                            | % Improv | 2.4%         |        |
| Breast C<br>277 $\times$ 9 | SVM      | 28.87(0.43)% | 138.17 |
|                            | $\psi$   | 23.56(0.37)% | 53.70  |
|                            | % Improv | 18.4%        |        |
| Thyroid<br>215 $\times$ 5  | SVM      | 9.43(0.25)%  | 35.26  |
|                            | $\psi$   | 7.39(0.27)%  | 16.02  |
|                            | % Improv | 21.6%        |        |

Table 5: Averages of testing errors of SVM and  $\psi$ -learning as well as the standard errors (in parentheses), minimized over  $C$  in polynomial learning in the two benchmark examples.

| Data                       |          | Testing       | # SV   |
|----------------------------|----------|---------------|--------|
| Obs $\times$ Dim           |          |               |        |
| Liver<br>345 $\times$ 6    | SVM      | 27.46(0.225)% | 101.29 |
|                            | $\psi$   | 26.63(0.210)% | 53.13  |
|                            | % Improv | 3.0%          |        |
| Breast C<br>277 $\times$ 9 | SVM      | 29.09(0.475)% | 92.82  |
|                            | $\psi$   | 27.26(0.462)% | 67.51  |
|                            | % Improv | 6.3%          |        |

Table 6: Averages of testing errors of SVM and  $\psi$ -learning as well as the standard errors (in parentheses), minimized over tuning parameters  $C$  and  $\sigma^2$  in learning with Gaussian kernels for the benchmark example.

| Data             |          | Testing      | # SV  |
|------------------|----------|--------------|-------|
| Obs $\times$ Dim |          |              |       |
| Page Block       | SVM      | 5.06(0.202)% | 50.46 |
| 444 $\times$ 10  | $\psi$   | 4.96(0.222)% | 43.50 |
|                  | % Improv | 1.87%        |       |

Particularly, 5, 45, 10 uniformly grid points respectively over  $(0, 10]$ ,  $(10, 10^3]$ , and  $(10^3, 10^4]$  are used, which are  $2i; i = 1, \dots, 5, 20i; i = 1, \dots, 50, 2000i; i = 1, \dots, 5$ . For the Gaussian kernel,  $\sigma^2$  is set to the same as in Example 2. The average smallest testing errors as well as the average number of support vectors for SVM and  $\psi$ -learning are summarized in Table 4 for linear learning with SQP-BB(N=100), in Table 5 for polynomial kernels, and in Table 6 for Gaussian kernels with SQP. Here  $K(x, y) = \langle x, y \rangle$  in the linear case,  $K(x, y) = (1 + \langle x, y \rangle)^2$  in the polynomial kernel case, and  $K(x, y) = \exp(-\frac{1}{\sigma^2}\|x - y\|^2)$  in the Gaussian kernel case.

We make the following observations regarding SVM and  $\psi$ -learning based on Tables 4-6.

- (1) Testing correctness of  $\psi$ -learning is higher than SVM on all benchmark datasets, for both linear and nonlinear learning.
- (2) On average,  $\psi$ -learning reduces the number of support vectors of SVM. The percent of reduction, however, varies.
- (3) Computing times for  $\psi$ -learning were about 7 times higher than those of SVM on average. The additional times are used for iteration, correcting the bias introduced by imposed convexity for SVM, while storage space for  $\psi$ -learning is only slightly higher.

In Tables 5 and 6, the percentage improvement is very modest. This is likely due to the fact that with kernel learning, the excess error rate (9) of the SVM over the Bayes error rate is probably of less than 10% already; so even if  $\psi$ -learning can reduce the excess error rate (9) by 50%, it will only show up as an improvement of just a

few percent according the computable index (10), which is only a lower bound of (9).

The numerical results here are consistent with the theoretical findings in Shen et al. (2003).

## 6 Appendix

**Proof of Theorem 1:** The  $k$ th subproblem (5) can be written as  $\frac{1}{2}\|w^*\|^2 + C \sum_{i=1}^n \psi_1(y_i f(\tilde{x}_i)) - \langle w, V^{(k)} \rangle$ , which, after introducing  $n$  slack variables  $\xi_i; i = 1, \dots, n$ , is equivalent to  $\min_w (\frac{1}{2}\|w^*\|^2 + C \sum_{i=1}^n \xi_i - \langle w, V^{(k)} \rangle)$  with constraints:  $\xi_i \geq 2(1 - y_i f(\tilde{x}_i)); \xi_i \geq 0$ , or  $y_i \langle w, \tilde{x}_i \rangle \geq 1 - \frac{1}{2}\xi_i; i = 1, \dots, n$ .

To solve this problem, we introduce Lagrangian multipliers to yield

$$L(w, \xi, \alpha, \gamma) = \frac{1}{2}\langle w^*, w^* \rangle + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [y_i (\langle w, \tilde{x}_i \rangle) - 1 + \frac{1}{2}\xi_i] - \sum_{i=1}^n \gamma_i \xi_i - \langle w, \nabla s_2(w^{(k)}) \rangle, \quad (11)$$

where  $\alpha_i \geq 0$  and  $\gamma_i \geq 0; i = 1, \dots, n$ . After differentiating  $L$  with respect to  $(w, \xi, \alpha, \gamma)$  and letting the derivatives be zero, we obtain that  $w^* = V_1^{(k)} - \sum_{i=1}^n \alpha_i y_i x_i = 0$ ,  $C - \frac{1}{2}\alpha_i - \gamma_i = 0$ , and  $-V_2^{(k)} = \sum_{i=1}^n \alpha_i y_i$ . Substituting these identities in (11),  $L(w, \xi, \alpha, \gamma) = \sum_{i=1}^n \alpha_i [1 - y_i \langle V_1^{(k)}, x_i \rangle] - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \frac{1}{2} \langle V_1^{(k)}, V_1^{(k)} \rangle$ . This yields (6) after ignoring constant terms. To derive the corresponding constraints, we note that  $C - \frac{1}{2}\alpha_i - \gamma_i = 0$ , together with  $\gamma_i \geq 0$ , implies  $\alpha_i \leq 2C$ . Furthermore,  $\xi_i \neq 0$  implies that  $\gamma_i = 0$  and  $\alpha_i = 2C$ . Hence, KKT's condition becomes  $\alpha_i [y_i \langle w, \tilde{x}_i \rangle - 1 + \frac{1}{2}\xi_i] = 0$  and  $\xi_i [\alpha_i - 2C] = 0; i = 1, \dots, n$ , implying that non-zero  $\xi_i$ 's can only occur when  $\alpha_i = 2C$ .

**Proof of Theorem 2:** The proof is essentially the same as that in Theorem 1 with slight modifications, and thus is omitted.

**Proof of Theorem 3:** We will only prove the linear case as the nonlinear case can be treated similarly. It follows from Theorem 6 of An and Tao (1997) that  $s(w^{(k)})$  is

non-increasing with respect to  $k$  and  $\lim_{k \rightarrow \infty} \|w^{(k+1)} - w^{(\infty)}\| = 0$ . From (6), we know that  $w^{(k+1)} = w^{(k)} + F(w^{(k)})$ , where  $F$  is a continuous mapping from  $\mathbb{R}^{d+1}$  to  $\mathbb{R}^{d+1}$ . By the assumption, there does not exist  $x^*$  such that  $\langle w^{(\infty)}, x^* \rangle = 0$ . By continuity, this property holds in a small  $w$ -neighborhood of  $w^{(\infty)}$ . Because  $\psi(z)$  is smooth when  $z$  stays away from the origin, the derivative  $F'$  exists and satisfies the Lipschitz condition  $\|F'(w) - F'(w^{(\infty)})\| \leq c\|w - w^{(\infty)}\|$  for a constant  $c > 0$  for any  $w$  in a small neighborhood of  $w^{(\infty)}$ . The convergence result then follows from Theorem 2.2 of Dennis and Moré (1974).

**Proof of Theorem 4:** We will solve the problem via QP. Let  $v = (v_1, \dots, v_{d+2})$  be the vertices of a simplex, with  $v_i = (v_{i1}, \dots, v_{i(d+1)})$ . Invoking the simplex representation, any point  $w = (w^*, w_{d+1})$  within the simplex can be written as  $w = \sum_{i=1}^{d+2} \lambda_i v_i, \lambda_i \geq 0; i = 1, \dots, d+2, \mathbf{1}^T \lambda = 1$ , where  $\mathbf{1} = (1, 1, \dots, 1)^T$  and  $\lambda = (\lambda_1, \dots, \lambda_{d+1})$ . Since  $\lambda_{d+2} = 1 - \sum_{i=1}^{d+1} \lambda_i$ ,  $w$  can be written as  $A\lambda + v_{d+2}$  with  $A = (v_1 - v_{d+2}, \dots, v_{d+1} - v_{d+2})$ . This yields that  $\lambda = A^{-1}(w - v_{d+2})$ , subject to  $\lambda \geq 0$  and  $\mathbf{1}^T \lambda \leq 1$ . Write the convex envelop  $l_s(w)$  as  $\langle a_1, w^* \rangle + a_2 w_{d+1} + a_{d+2}$ , where  $a_1$  and  $a_2$  are  $(d+1)$ -dimensional and one-dimensional, respectively. Then (7) becomes  $\min_w (\frac{1}{2}\|w^*\|^2 + C \sum_{i=1}^n \psi_1(y_i f(\tilde{x}_i)) + a_1^T w^* + a_2 w_{d+1})$  subject to  $A^{-1}(w - v_{d+2}) \geq 0, \mathbf{1}^T A^{-1}(w - v_{d+2}) \leq 1$ . After introducing  $n$  slack variables  $\xi_i; i = 1, \dots, n$ , it reduces to

$$\min_w \left( \frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \xi_i + a_1^T w^* + a_2 w_{d+1} \right)$$

subject to  $A_1 w^* + A_2 w_{d+1} \geq a_3, A_3 w^* + A_4 w_{d+1} \geq a_4, a_5 w^* + a_6 w_{d+1} \leq a_7, y_i \langle w, \tilde{x}_i \rangle \geq 1 - \frac{1}{2} \xi_i, \xi_i \leq 0; i = 1, \dots, n$ . Let  $A^{-1}$  be  $\begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$ , where  $A_i; i = 1, 2, 4$  are  $d \times d, d \times 1$  and  $1 \times 1, v_{d+2} = (v_{d+2}^*, vt_{d+2}), a_3 = A_1 v_{d+2}^* + A_2 vt_{d+2}, a_4 = A_3 v_{d+2}^* + A_4 vt_{d+2}, a_5 = \mathbf{1}^T A_1 + A_3, a_6 = \mathbf{1}^T A_2 + A_4$ , and  $a_7 = 1 + (\mathbf{1}^T A_1 + A_3) v_{d+2}^* + (\mathbf{1}^T A_2 + A_4) vt_{d+2} = 1 + \mathbf{1}^T a_3 + a_4$ . Now introduce the Lagrangian multipli-

ers:

$$\begin{aligned} L = & \frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^n \xi_i + a_1^T w^* + a_2 w_{d+1} \\ & - \sum_{i=1}^n \alpha_i (y_i \langle w, \tilde{x}_i \rangle - 1 + \frac{1}{2} \xi_i) - \sum_{i=1}^n r_i \xi_i \\ & - \beta^T (A_1 w^* + A_2 w_{d+1} - a_3) \\ & - \theta (A_3 w^* + A_4 w_{d+1} - a_4) \\ & + t (a_5 w^* + a_6 w_{d+1} - a_7), \end{aligned}$$

where  $\alpha \geq 0, r \geq 0, \beta \geq 0, \theta \geq 0$ , and  $t \geq 0$ . After differentiating  $L$  with respect to  $(w, \xi)$  and letting the derivatives be zero, we obtain that  $C = \frac{1}{2} \alpha_i + r_i, w^* = \sum_{i=1}^n \alpha_i y_i x_i + A_1^T \beta + \theta A_3^T - a_1 - t a_5^T, \sum_{i=1}^n \alpha_i y_i = a_2 + t a_6 - \beta^T A_2 - \theta A_4$ . Therefore,  $L = -\frac{1}{2} z^T H z + g^T z - \frac{1}{2} a_1 a_1^T$ , where the matrix  $H = (h_{ij}), h_{11} = y_x y_x^T, h_{12} = h_{21}^T = y_x A_1^T, h_{13} = h_{31}^T = y_x A_3^T, h_{14} = h_{41}^T = -y_x a_5^T, h_{22} = A_1 A_1^T, h_{23} = h_{32}^T = A_1 A_3^T, h_{24} = h_{42}^T = -A_1 a_5^T, h_{33} = A_3 A_3^T, h_{34} = h_{43}^T = -A_3 a_5^T, h_{44} = a_5 a_5^T, g = (a_1^T y_x^T + \mathbf{1}^T, a_1^T A_1^T + a_3^T, a_1^T A_3^T + a_4, -a_1^T a_5^T - a_7)$ , and  $y_x^T = (y_1 x_1, \dots, y_n x_n)$ . This yields the desired result. Similarly as in the proof of Theorem 1, the corresponding constraints can be derived.

**Proof of Theorem 5:** Note that  $U(S) - L(S) \leq |\langle w, \nabla s_2(w^s) \rangle - \langle w, \alpha \rangle|$ , where  $\alpha \in \mathcal{R}^{d+1}$  is the coefficient vector for  $l_s$ . Furthermore, when the longest edge partition is employed, the volume of  $S$  shrinks to zero, thus  $\|\alpha - \nabla s_2(w^s)\| \rightarrow 0$ . Consequently,  $|U(S) - L(S)|$  shrinks to zero as the number of iterations increases. The algorithm stops finitely when  $\varepsilon > 0$ .

## References

- [1] An, L. T. H., and Tao, P.D. (1997). Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *J. global opt.*, **11**, 253-285.
- [2] Boser, B., Guyon, I., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Fifth Annual Conference on Computational Learning Theory*, Pittsburgh ACM, 142-152.
- [3] Cortes, C., and Vapnik, V. N. (1995). Support-vector networks. *Machine Learning*, **20**, 273-297.

- [4] Dennis and Moré (1974). A characterization of super-linear convergence and its application to quasi-Newton methods. *Math. comput.*, **28**, 549-560.
- [5] Fung, G., and Mangasarian, O.L. (2000). Data selection for support vector machine classifiers. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 20-23, 2000, Boston, MA, R. Ramakrishnan & S. Stolfo, editors, ACM, NY 2000, 64-70.
- [6] Falk, J. E., and Hooeman, K. L. (1976). A successive underestimation method for concave minimization problems. *Mathematics of operations research*, **1**, 251-259.
- [7] Horst, R., and Tuy, H. (1989). *Global optimization-deterministic approaches*. Springer-Verlag.
- [8] Lin, Y. (2000) Some Asymptotic Properties of the Support Vector Machine. Technical report 1029r. Revised February 2002. Department of Statistics, University of Wisconsin.
- [9] Lin, Y. (2002) Support Vector Machines and the Bayes Rule in Classification. *Data Mining and Knowledge Discovery.*, **6**, 259-275.
- [10] Murphy, P.M., and Aha, D.W. (1992). UCI repository of machine learning databases. ([www.ics.uci.edu/~mllearn/MLRepository.html](http://www.ics.uci.edu/~mllearn/MLRepository.html)).
- [11] Rätsch, G., Onoda, T., and Müller, K.R (2001). Soft Margins for AdaBoost. *Machine Learning*, **42**, 287-320.
- [12] Shen, X., Tseng, G., Zhang, X., and Wong, W. H. (2003). On  $\psi$ -learning. *J. Ameri. Statist. Assoc.*, **98**, 724-734.
- [13] Wahba, G. (1990). *Spline models for observational data*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, xii + 169 pp, Vol. 59.
- [14] Wahba, G. (1999). Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. *Advances in Kernel Methods Support Vector Learning*, MIT Press, 69-88.
- [15] Wolberg, W.H., and Mangasarian, O. L. (1990). Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. Natl. Acad. Sci.*, **87**, 9193-9196.
- [16] Zhu, J., and Hastie, T. (2004). Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, to appear.