Statistica Sinica **16**(2006), 441-457

OPTIMIZING ψ-LEARNING VIA MIXED INTEGER PROGRAMMING

Yufeng Liu and Yichao Wu

University of North Carolina at Chapel Hill

Abstract: As a new margin-based classifier, ψ-learning shows great potential for high accuracy. However, the optimization of ψ-learning involves non-convex minimization and is very challenging to implement. In this article, we convert the optimization of ψ-learning into a mixed integer programming (MIP) problem. This enables us to utilize the state-of-art algorithm of MIP to solve ψ-learning. Moreover, the new algorithm can solve ψ-learning with a general piecewise linear ψ loss and does not require continuity of the loss function. We also examine the variable selection property of 1-norm ψ-learning and make comparisons with the SVM.

Key words and phrases: Classification, norm, regularization, SVM, variable selection.

1. Introduction

With the ultimate goal of maximizing classification prediction accuracy, classification techniques with good generalization ability are desirable. The Support Vector Machine (SVM), originally invented in the machine learning literature, has attracted tremendous interest in both the machine learning and statistics communities. It has been widely applied to various disciplines, including engineering, biology, and medicine.

The SVM paradigm was first introduced by V. Vapnik and colleagues with the idea of searching for the optimal separating hyperplane, see Boser, Guyon, and Vapnik (1992) and Vapnik (1998). It is now known that SVM can be put in the regularization framework. Specifically, the objective cost function in regularization includes a data fit component as well as a penalty term. The data fit component ensures that the model has a good fit to the training data, while the penalty term avoids overfitting of the resulting model (Wahba (1998)). The most natural measure of the data fit is the classification error based on 0-1 loss on the training data. However, optimization of the 0-1 loss is very difficult. Therefore, most classification methods use convex losses as surrogates of the 0-1 loss, among which the hinge loss used by SVM is the closest convex surrogate.

Recently, Shen et al. (2003) proposed a new learning method called ψ-learning. In contrast to the SVM, ψ-learning uses a group of piecewise linear

nonconvex losses which are closer to the 0-1 loss than the hinge loss. They showed that although the optimization is more difficult, ψ-learning has the potential of better generalization than the SVM. In their paper, they used a direct complex algorithm for numerical comparisons. However, it is a local algorithm and may only work well for low-dimensional problems. Liu, Shen and Doss (2005) considered a special continuous ψ loss which can be decomposed as a sum of a convex function and a concave function. With this decomposition, the optimization can then be carried out using the so called d.c. algorithm. Although this algorithm can handle large-scale problems, it only deals with a special ψ loss that is continuous, and it may also have a problem with local minima. More discussions on the d.c. algorithm can be found in Liu, Shen and Wong (2005).

In this paper, we consider a group of piecewise linear ψ loss functions that convert the nonconvex optimization problem of ψ-learning into a mixed integer programming (MIP) problem. We compare the performance of different ψ losses and give recommendations on the choice of ψ. In addition to the standard 2-norm ψ-learning, we also explore the variable selection property of 1-norm ψ-learning and make comparisons with the SVM. A simplified algorithm utilizing a close relationship between ψ-learning and the SVM is proposed.

The rest of this paper is organized as follows. In Section 2, we briefly review the framework of the SVM and ψ-learning. We then present some basics of MIP and formulate the optimization problem of ψ-learning as a MIP problem in Section 3. In Section 4, we discuss the variable selection property of 1-norm ψ-learning. Numerical examples are given in Section 5, followed by some discussions in Section 6.

2. SVM and ψ-Learning

For simplicity, we only discuss binary classification problems with class label $Y \in \{\pm 1\}$ and input vector $X \in S \subset R^d$. In supervised learning, a training sample of n input/output pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, i.i.d. realizations from $P(\boldsymbol{x}, y)$, is given. Then a function f, mapping from S to R, can be constructed using the training set such that $\text{sign}(f(\boldsymbol{x}))$ is the classification rule.

Under the regulation framework, we consider classifiers solving the optimization problem

$$\underset{f}{\text{argmin}}\ J(f) + C\sum_{i=1}^{n} l(y_i f(\boldsymbol{x}_i)), \tag{2.1}$$

where $J(f)$ serves as a regularization term and $C > 0$ is a tuning parameter that controls the balance between the data fit measured by a loss l and the complexity of f.

For linear learning problems, $f(\boldsymbol{x}) = \boldsymbol{w}'\boldsymbol{x} + b$ is a hyperplane in R^d. In this case, $J(f)$ is some functional of a certain norm of $\boldsymbol{w}$. The most commonly used norm is the 2-norm (or squared norm) with $J(f) = \|\boldsymbol{w}\|_2^2/2 = \boldsymbol{w}'\boldsymbol{w}/2$. Another interesting one is the LASSO-type penalty, based on the 1-norm, with $J(f) = \|\boldsymbol{w}\|_1 = \sum_{i=1}^d |w_i|$ (Tibshirani (1996)). Nonlinear learning can be achieved by either basis expansion or kernel mapping (Cristianini and Shawe-Taylor (1999)). Basis expansion increases the number of basis functions to achieve nonlinear learning. For kernel learning, a kernel $K(\cdot,\cdot)$ that maps $S \times S$ to R is employed. Then $f(\boldsymbol{x}) = \sum_{i=1}^n \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b$, by the theory of reproducing kernel Hilbert spaces (RKHS), see Wahba (1990). In this case, the 2-norm of $f(\boldsymbol{x}) - b$ in the RKHS with $K(\cdot,\cdot)$ is typically used as $J(f)$. Using the representer theorem (Kimeldorf and Wahba (1971)), $J(f)$ can be represented as $\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}/2$, where $\mathbf{K}$ is the $n \times n$ dimensional kernel matrix with $\mathbf{K}(i,j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

A desirable classifier is one with good generalization ability, measured by the generalization error (GE). The GE, defined as the probability of misclassification, can be written as $Err(f) = P(Yf(X) < 0) = E(1 - \text{sign}(Yf(X))/2$. A classifier with a loss function $l(u)$ tries to minimize the GE through l. Hence l plays a similar rule as the $0-1$ loss, i.e., $(1-\text{sign})/2$. In the case of the SVM, one uses the hinge loss function with $l(u) = [1-u]_+$, a piecewise linear convex function. Shen et al. (2003) considered using the $0-1$ loss directly, with $l(u) = 1 - \text{sign}(u)$. However, since the sign function is scale invariant, the solution of (2.1) with $l(u) = 1 - \text{sign}(u)$ is then approximately equal to $f = 0$. To overcome this problem, they proposed a group of ψ loss functions satisfying

$$\begin{aligned} U \geq \psi(u) > 0 \text{ if } u \in [0, \tau]; \\ \psi(u) = 1 - \text{sign}(u) \text{ otherwise,} \end{aligned} \tag{2.2}$$

where $0 < U \leq 2$ and $\tau > 0$ are constants. Note that the positive values of $\psi(u)$ when $u \in [0, \tau]$ eliminate the scaling problem of the sign function and avoid too many points piling around the decision boundary.

The choice of ψ function is an interesting problem. Different ψ functions may deliver different classification performances. So far, two different choices have been used. In Shen et al. (2003), $\psi(u)$ is defined to be 0 if $u \geq 1$, $1 - u$ if $0 \leq u \leq 1$, and 2 otherwise. In order to utilize the so called d.c. algorithm, Liu, Shen and Doss (2005) employed a continuous ψ function with $\psi(u) = 2(1 - u)$ for $0 \leq u \leq 1$. It is not obvious, however, how the performance of different losses varies. In this paper, we consider piecewise linear ψ functions with $\psi(u) = a - au$ for $u \in [0, 1]$, and $1 - \text{sign}(u)$ otherwise, a a constant in $[0, 2]$. This covers the previously mentioned two choices with $a = 1$ and $a = 2$, respectively. Figure 1 displays a general piecewise linear ψ loss function. As a remark, we note that $\tau = 1$ in (2.2) covers a general situation since an equivalent loss ψ/τ belongs

to the group of loss functions we consider. Therefore, our ψ loss functions are governed only by a. From previous studies, it is not clear how to select a.

In Section 3, we show that ψ-learning with a general piecewise linear ψ loss can be converted into a MIP problem. Under the framework of MIP, we examine the performance of ψ-learning with different a's, and make recommendations based on the numerical results.

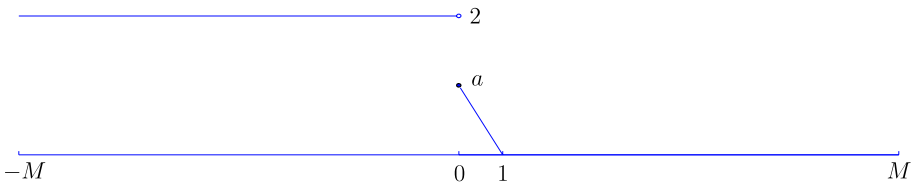

Figure 1. Plot of a piecewise linear ψ loss function.

3. Optimization of ψ-Learning

3.1. MIP

A MIP problem is the minimization of a linear or quadratic function subject to linear constraints with some of the variables being integers. When the objective function is quadratic, it becomes a mixed integer quadratic programming (MIQP) problem. Similarly, a mixed integer linear programming (MILP) problem has a linear objective function. For the purpose of illustration, a MILP problem with n variables $\mathbf{z} = (z_1, \ldots, z_n)$ and m constraints has the following form:

$$\min_{\mathbf{z}} \mathbf{g}^T \mathbf{z}, \tag{3.1}$$

subject to

$$A_1 \mathbf{z} = \mathbf{b}_1, \tag{3.2}$$

$$A_2 \mathbf{z} \leq \mathbf{b}_2, \tag{3.3}$$

$$\mathbf{l} \leq \mathbf{z} \leq \mathbf{u}, \tag{3.4}$$

$$z_j \text{ is an integer for } \forall j \in D, \tag{3.5}$$

where D is a subset of $\{1, \ldots, n\}$, A_1 is an $m_1 \times n$ matrix, A_2 is an $m_2 \times n$ matrix, $\mathbf{g}, \mathbf{l}, \mathbf{u}$ are vectors of length n, $\mathbf{b}_1$ and $\mathbf{b}_2$ are vectors of lengths m_1 and m_2 respectively, with $m_1 + m_2 = m$. When D is empty, (3.1) becomes a linear programming (LP) problem which can be solved in polynomial time.

MIP is an important optimization problem in the field of operations research and has been studied for many years. One common approach to solve a MIP is the so called branch and bound algorithm. The essence of the algorithm is to solve the linear relaxation of the original problem without the integer requirement. If the solution satisfies the integer constraints, then the optimal solution

is found. Otherwise one can create two new subproblems by branching on a fractional variable which is required to be an integer. Then one needs to keep solving subproblems created by branching and improving the lower bound of the objective by pruning out certain parts of the solution space. The algorithm stops when the optimal solution attaining the lower bound is obtained. Another popular approach is the cutting plane technique. The idea behind this approach is to add constraints to a LP problem until the optimal basic feasible solution satisfies the integer requirement. There are many other algorithms developed to solve a MIP. Garfinkel and Nemhauser (1972) and Wolsey and Nemhauser (1999) are excellent books on this topic.

The complexity of MIP varies, depending on the size of the problem, the numerical characteristics of the data, the algorithm used, and the computation hardware. Some MIP problems with hundreds of thousands of variables and constraints can be solved in a few minutes. On the other hand, there exist small MIP problems with a few hundred variables that are not yet solved. When it is impractical to compute an optimal solution, one can settle for a good solution that is not optimal. There are many optimization solvers available. A detailed software list can be found at http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/index.html. All the examples presented in this paper were computed via the commercial optimization software CPLEX with the AMPL interface (Fourer et al. (2002)). As a remark, we note MILP is typically easier than MIQP.

3.2. Converting ψ-learning into MIP

The linear ψ-learning with a general piecewise ψ loss solves the optimization problem

$$\underset{f}{\operatorname{argmin}}\ J(f) + C\sum_{i=1}^{n}\psi(y_i f(\boldsymbol{x}_i)), \tag{3.6}$$

where $\psi(u) = a - au$ for $u \in [0,1]$, and $1-\operatorname{sign}(u)$ otherwise. Since the ψ function is non-convex, the main challenge of (3.6) is how to deal with $\psi(u)$. Let M be a large positive constant so that $yf(\boldsymbol{x}) \in [-M, M]\ \forall(y, \boldsymbol{x})$. The value of M needs to be determined prior to training, $M = 10^3$ is usually sufficient. Then $\psi(u)$ can be split into three regions, namely Region 1: $\psi(u) = 2$ with $u \in [-M, 0)$, Region 2: $\psi(u) = a - au$ with $u \in [0,1]$, and Region 3: $\psi(u) = 0$ with $u \in [1, M]$ (see Figure 1). Note that ψ is a linear function within each of the three regions. Thus, $\psi(y_i f(\boldsymbol{x}_i))$ can be represented as a linear function if the specific region where $y_i f(\boldsymbol{x}_i)$ falls is known. To this end, we introduce binary (dyadic) variables δ_2^i and δ_3^i; $i = 1, \ldots, n$, which specify the assignments of the n training pairs among the three regions. Specifically, $(\delta_2^i, \delta_3^i) = (0,0), (1,0), (1,1)$ correspond to regions $1, 2, 3$ respectively. In addition, new variables (Z_1^i, Z_2^i, Z_3^i) are introduced to determine the values of $y_i f(\boldsymbol{x}_i)$; $i = 1, \ldots, n$. Then we express $y_i f(\boldsymbol{x}_i)$ as

$Z_1^i + Z_2^i + Z_3^i$ where $Z_1^i \in [-M, 0]$, $Z_2^i \in [0, 1]$, and $Z_3^i \in [0, M-1]$. Through imposing certain linear constraints, we aim to have $Z_2^i = Z_3^i = 0$ when $(\delta_2^i, \delta_3^i) = (0, 0)$, i.e., $y_i f(\boldsymbol{x}_i)$ belongs to Region 1; $Z_1^i = Z_3^i = 0$ when $(\delta_2^i, \delta_3^i) = (1, 0)$; and $Z_1^i = 0, Z_2^i = 1$ when $(\delta_2^i, \delta_3^i) = (1, 1)$. Therefore, (Z_1^i, Z_2^i, Z_3^i) give the exact location of $y_i f(\boldsymbol{x}_i)$ in $[-M, M]$. Although there are $3n$ of the (Z_1^i, Z_2^i, Z_3^i), their true dimension in optimization is only n. We are now ready to express $\psi(y_i f(\boldsymbol{x}_i))$ as

$$\psi(y_i f(\boldsymbol{x}_i)) = 2 - aZ_2^i - (2-a)\delta_2^i, \tag{3.7}$$

subject to the constraints

$$y_i f(\boldsymbol{x}_i) = Z_1^i + Z_2^i + Z_3^i, \tag{3.8}$$

$$\delta_3^i \le \delta_2^i \in \{0, 1\}, \tag{3.9}$$

$$\delta_3^i \le Z_2^i \le \delta_2^i, \tag{3.10}$$

$$Z_3^i \le (M-1)\delta_3^i, \tag{3.11}$$

$$-Z_1^i \le 2M(1 - \delta_2^i), \tag{3.12}$$

$$Z_1^i \in [-M, 0],\ Z_2^i \in [0, 1],\ Z_3^i \in [0, M-1]. \tag{3.13}$$

To show (3.7), we can consider the three cases under (3.9).

Case 1. ($\delta_2^i = \delta_3^i = 0$): By (3.10) and (3.11), $Z_2^i = Z_3^i = 0$. Then $Z_1^i \ge -2M$ by (3.12) and thus $Z_1^i \in [-M, 0]$ by (3.13). Consequently, we have $y_i f(\boldsymbol{x}_i) \in [-M, 0]$ with $\psi(y_i f(\boldsymbol{x}_i)) = 2$ by (3.7) and (3.8).

Case 2. ($\delta_2^i = 1, \delta_3^i = 0$): $Z_1^i = Z_3^i = 0$ follows immediately from (3.11) and (3.12). From (3.7), (3.8), and (3.10), we have $y_i f(\boldsymbol{x}_i) = Z_2^i \in [0, 1]$ with $\psi(y_i f(\boldsymbol{x}_i)) = a - a y_i f(\boldsymbol{x}_i)$.

Case 3. ($\delta_2^i = \delta_3^i = 1$): $Z_1^i = 0$ and $Z_2^i = 1$ by (3.10) and (3.12). From (3.11), we have $Z_3^i \le M - 1$. Thus, $\psi(y_i f(\boldsymbol{x}_i)) = 0$ and $y_i f(\boldsymbol{x}_i) = Z_3^i + 1 \in [1, M]$ by (3.7) and (3.8).

Therefore, $\psi(y_i f(\boldsymbol{x}_i))$ is reformulated as a linear function subject to linear constraints with respect to f, binary variables (δ_2^i, δ_3^i), as well as bounded continuous variables (Z_1^i, Z_2^i, Z_3^i). This is a critical step in the derivation of our MIP formulation for ψ-learning. Note that $y_i f(\boldsymbol{x}_i) = 0$ is included in both cases 1 and 2, but $\psi(0) = a$ will be selected in the minimization.

After plugging (3.7) into (3.6), (3.6) can be converted into the MIP problem

$$\underset{\{f, \delta_2^i, \delta_3^i, Z_1^i, Z_2^i, Z_3^i\}_{i=1}^n}{\operatorname{argmin}} J(f) + C \sum_{i=1}^{n} (2 - aZ_2^i - (2-a)\delta_2^i), \tag{3.14}$$

subject to constraints (3.8)$-$(3.13) for $i = 1, \ldots, n$. When the 2-norm is employed, $J(f) = \boldsymbol{w}'\boldsymbol{w}/2$ and (3.14) becomes a MIQP problem. Similarly, (3.14) turns into a MILP problem if $J(f) = \|\boldsymbol{w}\|_1$.

As a general remark, the MIP formulation (3.14) is equivalent to the optimization problem of ψ-learning with any a. This allows us to make use of the existing MIP tools to tackle ψ-learning. Although some MIP problems may not be solved in polynomial time, many MIP solvers like CPLEX can report the best possible solution obtained within a specified time.

3.3. Improved algorithm

The dimension of problem (3.14) is $5n + d + 1$. Among these variables, $2n$ binary variables (δ_2^i, δ_3^i) correspond to the assignments of $(\boldsymbol{x}_i, y_i)$ among the three regions, $3n$ continuous variables (Z_1^i, Z_2^i, Z_3^i) specify the exact locations of $y_i f(\boldsymbol{x}_i); i = 1, \ldots, n$. The remaining $d + 1$ variables $(\boldsymbol{w}, b)$ yield the solution $f(\boldsymbol{x})$. By (3.9)−(3.13), only n of $3n$ variables (Z_1^i, Z_2^i, Z_3^i) need to be searched. Because of the $2n$ binary variables $\{\delta_2^i, \delta_3^i\}$, the algorithm can be slow as n gets large. Therefore, improvement is needed in order to deal with real applications.

The main difference between the SVM and ψ-learning comes from their loss functions on Regions 1 and 2, i.e, when $y_i f(\boldsymbol{x}_i) \leq 1$. Specifically, the SVM assigns a loss for point $(\boldsymbol{x}_i, y_i)$ linearly in the value of $y_i f(\boldsymbol{x}_i)$ once it is less than 1. Consequently, the resulting classifiers could be sensitive to outliers. ψ-learning, by contrast, gives the same losses for points on the "wrong" side of the boundary, namely $y_i f(\boldsymbol{x}_i) < 0$. Thus, the decision boundaries yielded by ψ-learning differ from the ones by the SVM in the sense that ψ-learning tries to reassign points in a more robust way, especially points not assigned to Region 3 by the SVM. In the SVM context, those points satisfying $y_i f(\boldsymbol{x}_i) \leq 1$ are the "support vectors" (SVs) and the SVM classifiers only depend on those SVs. See Cristianini and Shawe-Taylor (1999) for more details. Once a SVM boundary is given, ψ-learning tries to reassign those SVs. The non-SVs of the SVM are more likely to stay in Region 3 although their exact locations in that region, determined by the value of $y_i f(\boldsymbol{x}_i)$, may vary. Therefore, we can reduce the complexity of the MIP of ψ-learning by restricting non-SVs in Region 3 of a SVM classifier. Specifically, those points satisfy $\delta_2^i = \delta_3^i = 1$. In this way, the number of binary variables in the MIP can be reduced substantially if the SVM solution only involves a small number of SVs.

Our improved algorithm for ψ-learning can be summarized in the following two steps.

Step 1. Compute the SVM classifier for the training data and obtain the set of non-SVs $\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : y_i f_{SVM}(\boldsymbol{x}) > 1\}$.

Step 2. Solve the MIP problem (3.14) with constraints $\delta_2^i = \delta_3^i = 1$ for all points in $\mathcal{L}$.

One issue in Step 1 is the choice of C for the SVM. There are two natural selections for C, one is the same C used by ψ in Step 2, and the other one is

the tuned C best for the SVM. In fact, the tuning parameter C for the SVM is not directly comparable to that of ψ-learning because of their different loss functions. Therefore, we use the best tuned SVM solution for Step 1 in this paper. In particular, we compute the SVM solutions for a number of candidate C's and select the one with the smallest tuning error. For presentation, we refer to the MIP problem of (3.14) as unrestricted ψ-learning and the improved algorithm as restricted ψ-learning. To further justify our improved algorithm, we compare the results of unrestricted and restricted ψ-learning in a simulated example. More details are given in Section 3.3.

3.4. Choice of a

In this section, we study the performance of ψ-learning with different choices of a using the simulated example in Shen et al. (2003). Specifically, a random training set is obtained as follows. First, two dimensional vectors $\{(X_{i1}, X_{i2})\}_{i=1}^{n}$ are generated uniformly over the unit disk $\{(x_1, x_2) : x_1^2 + x_2^2 \leq 1\}$, and then Y_i is assigned to be 1 if $X_{i1} \geq 0$ and -1 otherwise. After that, some randomly selected class labels $\{Y_i\}$ are changed to the other class in order to examine the robustness of ψ-learning to outliers.

We computed classifiers of 1-norm and 2-norm ψ-learning with four different flipping numbers (0, 1, 2, 5), four different values of tuning parameter C $(1, 10, 10^3, 10^7)$, and eight choices of a (0.1, 0.2, 0.5, 1, 1.5, 1.8, 1.9, 2). In this example, M is set to be 10^3 and seems to work well. To compare the performance of ψ-learning with restriction to that without restriction, we computed the solutions for both cases. The results of averaging 100 replications of 2-norm ψ-learning with $n = 50$ are summarized in Table 1. For comparison, the results of the 2-norm SVM are reported as well.

In each case, the best error rates with respect to different C's are highlighted in bold face. The smallest errors across different a's are underlined. From these results, we make the following observations. The errors indicate that the choice of a may slightly affect the classification performance of ψ-learning. Our numerical experience suggests that small or large a may be suboptimal for ψ-learning. Small values of a, for example 0.1 and 0.2 which are close to 0, can be numerically unstable and consequently are not good for implementation. On the other hand, large values of a, for example 1.8, 1.9, and 2, sometimes do not deliver the best performance for ψ-learning. One plausible explanation is that a ψ loss with large a does not distinguish points with $-\epsilon < yf(\boldsymbol{x}) < 0$ (wrongly classified points) and $0 < yf(\boldsymbol{x}) < \epsilon$ (correctly classified points) well because of their similar losses for these points. In contrast, a relatively smaller a can avoid such a problem. Based on numerical results, we recommend selecting $a \in [0.5, 1.5]$ although a ψ loss with $a = 2$ still outperforms the SVM in this example. For other numerical examples in the paper, we use $a = 0.5$.

Table 1. Error rates of the 2-norm SVM, unrestricted and restricted 2-norm ψ-learning for the example in Section 3.3 with $n = 50$. Entries marked by "—" are unavailable because of numerical difficulty.

2-norm SVM				
n flip	$c = 1$	$c = 10$	$c = 1,000$	$c = 10^7$
0	0.0486	0.0344	0.0228	**0.0222**
1	0.0546	0.0418	**0.0350**	0.0360
2	0.0571	0.0425	**0.0413**	0.0413
5	0.0675	0.0583	**0.0573**	0.0573

2-norm ψ						restricted 2-norm ψ			
n flip	a	$c = 1$	$c = 10$	$c = 1,000$	$c = 10^7$	$c = 1$	$c = 10$	$c = 1,000$	$c = 10^7$
0	0.1	0.0322	0.0275	**0.0209**	0.0222	0.0291	0.0253	**0.0203**	0.0222
	0.2	0.0314	0.0260	**0.0215**	0.0222	0.0292	0.0262	**0.0215**	0.0222
	0.5	0.0293	0.0258	**0.0219**	0.0222	0.0291	0.0254	**0.0219**	0.0222
	1.0	0.0313	0.0276	0.0227	**0.0222**	0.0287	0.0274	0.0227	**0.0222**
	1.5	0.0461	0.0361	0.0234	**0.0222**	0.0303	0.0292	0.0233	**0.0222**
	1.8	0.0576	0.0398	0.0237	**0.0222**	0.0310	0.0302	0.0237	**0.0222**
	1.9	0.0614	0.0411	0.0238	**0.0222**	0.0316	0.0303	0.0238	**0.0222**
	2.0	0.0633	0.0419	0.0238	**0.0222**	0.0319	0.0307	0.0238	**0.0222**
1	0.1	—	0.0322	0.0288	**0.0279**	0.0379	0.0309	**0.0289**	0.0289
	0.2	0.0349	0.0344	0.0283	**0.0279**	0.0381	0.0332	**0.0282**	0.0289
	0.5	0.0331	0.0311	0.0279	**0.0276**	0.0367	0.0317	**0.0278**	0.0286
	1.0	0.0370	0.0325	**0.0274**	0.0281	0.0367	0.0335	**0.0274**	0.0286
	1.5	0.0535	0.0382	**0.0269**	0.0276	0.0396	0.0386	**0.0269**	0.0279
	1.8	0.0643	0.0437	**0.0270**	0.0286	0.0440	0.0437	**0.0270**	0.0284
	1.9	0.0679	0.0436	**0.0270**	0.0286	0.0451	0.0433	**0.0270**	0.0284
	2.0	0.0691	0.0447	**0.0272**	0.0286	0.0465	0.0437	**0.0272**	0.0289
2	0.1	—	—	**0.0270**	0.0275	0.0382	0.0292	**0.0271**	0.0303
	0.2	—	0.0316	**0.0263**	0.0277	0.0379	0.0306	**0.0263**	0.0303
	0.5	0.0339	0.0309	**0.0271**	0.0277	0.0372	0.0305	**0.0271**	0.0303
	1.0	0.0427	0.0322	0.0287	**0.0273**	0.0387	0.0312	**0.0287**	0.0303
	1.5	0.0563	0.0383	0.0295	**0.0291**	0.0446	0.0386	**0.0295**	0.0305
	1.8	0.0672	0.0452	0.0300	**0.0294**	0.0521	0.0444	**0.0300**	0.0305
	1.9	0.0693	0.0456	0.0298	**0.0289**	0.0543	0.0448	**0.0298**	0.0305
	2.0	0.0725	0.0455	0.0298	**0.0285**	0.0553	0.0448	**0.0298**	0.0305
5	0.1	—	—	—	—	0.0470	0.0456	**0.0396**	0.0413
	0.2	—	—	—	—	0.0465	0.0439	**0.0392**	0.0413
	0.5	0.0484	0.0407	**0.0389**	0.0390	0.0462	0.0408	**0.0382**	0.0415
	1.0	0.0498	0.0442	**0.0354**	0.0396	0.0472	0.0437	**0.0346**	0.0422
	1.5	0.0601	0.0511	**0.0354**	0.0399	0.0562	0.0504	**0.0345**	0.0413
	1.8	0.0709	0.0559	**0.0351**	0.0408	0.0640	0.0559	**0.0343**	0.0421
	1.9	0.0721	0.0560	**0.0351**	0.0394	0.0666	0.0560	**0.0343**	0.0421
	2.0	0.0742	0.0561	**0.0362**	0.0415	0.0677	0.0561	**0.0354**	0.0421

As a remark, we note that a may be treated as an additional tuning parameter and determined in a data-driven fashion. This appears to be appealing theoretically. However, it will greatly increase the computational demand. Moreover, numerical problems with small a's can be a concern in practice. For this reason, we treat a as a fixed constant throughout the paper.

A comparison between restricted and unrestricted ψ-learning shows that restricted ψ-learning gives similar performance as that of unrestricted one. This justifies the use of our improved algorithm which can reduce the computation significantly. In fact, there are cases in Table 1 in which restricted ψ-learning even outperforms unrestricted ψ-learning. This is caused by the use of the best tuned SVM for restriction, which helps to improve the restricted ψ-learning with respect to a specific C.

Although restricting regions of training points can help to speed up the algorithm, it has the potential risk of casting (3.14) into a different optimization problem, and may weaken the performance of ψ-learning. Therefore, it is advisable to make as few restrictions on training points as possible if the computational cost is to be acceptable.

4. Variable Selection

Variable selection is important in statistical modeling, especially when the dimension of input variables is large. In classification, we quite often have a relatively large number of candidate covariates available for building accurate classifiers. However, it is not desirable to have a classifier that includes too many noise variables. On one hand, the accuracy of the resulting classifier may not be very satisfactory with many uninformative covariates in the model. On the other hand, a classifier with sparsity on input variables has good interpretability and makes it possible to further investigate the effects of those important covariates.

To achieve variable selection, one may carry out prescreening on covariates and construct classifiers using only those selected. However, it is preferable to build a classifier with a "built-in" variable selection ability. Then classification and variable selection can be achieved simultaneously. In the SVM case, it is known that the 1-norm SVM has variable selection ability (Bradley and Mangasarian (1998) and Zhu et al. (2003)). As discussed earlier, ψ-learning has a very close relationship with the SVM. Therefore, it is natural to study the property of 1-norm ψ-learning in terms of classification accuracy as well as sparsity of the resulting classifiers. The difference between the effects of 1-norm and 2-norm penalties comes from their different treatments of values of the resulting $\boldsymbol{w}$. The 1-norm puts a relatively large penalty on small coefficients and consequently sparsity of the solution can be achieved. Figure 2 depicts the two norms

for the one-dimensional case. As a result, the obtained solution has more zero coefficients if the 1-norm is employed.

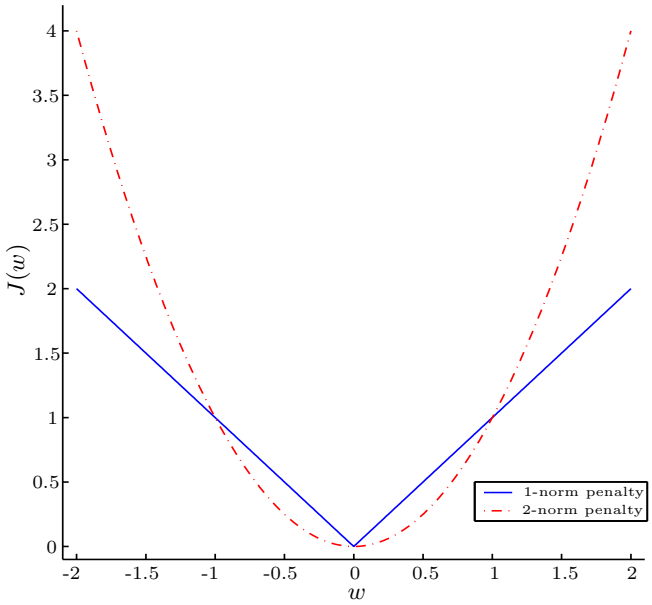

Figure 2. A comparison plot of the 1-norm and 2-norm penalties

In terms of computation, 1-norm ψ-learning requires MILP which is easier to compute than MIQP, as needed for 2-norm ψ-learning. Numerical results on the performance of the 1-norm/2-norm SVM and ψ-learning in classification, as well as variable selection, are reported in Section 5.

5. Numerical Examples

5.1. Simulation

We generate 100 random samples of size 100 with input $\mathbf{x} = (x_1, x_2, x_3, x_4)$. Each component x_i is distributed as Uniform$[-2, 2]$. Let $f(\mathbf{x}) = x_1^2 + x_2^2 - 2.8$. With $\mathbf{x} = (x_1, x_2, x_3, x_4)$ given, the label y is set to be $\text{sign}(f(\mathbf{x}))$ with probability 0.95 and $-\text{sign}(f(\mathbf{x}))$ with probability 0.05. Independent tuning data sets with the same size as the training set and a testing set of size 20,000 are generated in the same manner.

From the data generation mechanism, it is easy to see that only two variables, x_1^2 and x_2^2, are important. To examine the variable selection ability and classification accuracy of each method, we apply the 1-norm SVM, 2-norm SVM, 1-norm ψ-learning, and 2-norm ψ-learning on four different sets of inputs as follows.

Case 1. Two variables x_1^2 and x_2^2.

Case 2. Four variables x_1^2, x_2^2, x_3^2, and x_4^2.

Case 3. Eight variables x_1^2, x_2^2, x_3^2, x_4^2, x_1, x_2, x_3, and x_4.

Case 4. Fourteen variables x_1^2, x_2^2, x_3^2, x_4^2, x_1, x_2, x_3, x_4, x_1x_2, x_1x_3, x_1x_4, x_2x_3, x_2x_4, and x_3x_4.

The results are summarized in Table 2. From the table, we can make a general observation that 1-norm and 2-norm ψ-learning outperform the corresponding SVMs in terms of classification accuracy. Case 1 has no noise variables in the input. In this situation, the 2-norm ψ (SVM) appears to do slightly better than the 1-norm ψ (SVM). However, as more and more noise variables are added, classification accuracy gets worse and worse for all methods. As expected, the 1-norm ψ (SVM) can perform variable selection with some zero coefficients. In contrast, the 2-norm penalty has no ability in variable selection, although it is still feasible to interpret the results by examining the magnitudes of the estimated coefficients. As a result, the 1-norm ψ (SVM) has smaller testing errors. In this example, the 1-norm SVM has slightly more zero coefficients, yet larger testing errors, than that of 1-norm ψ-learning. The better accuracy of ψ-learning is realized through its robust ψ loss.

Table 2. Results of the simulation example in Section 5.1 for four different cases. Input vectors for the four cases have 2, 4, 8, and 14 input variables respectively, with 2 important ones.

Case	Method	Error			Avg. # of zero coef.
		Training	Tuning	Testing	
1	1-norm SVM	0.0688(0.0306)	0.0676(0.0324)	0.0770(0.0202)	0.00
	1-norm ψ	0.0537(0.0258)	0.0537(0.0261)	0.0620(0.0094)	0.00
	2-norm SVM	0.0682(0.0307)	0.0654(0.0320)	0.0761(0.0203)	0.00
	2-norm ψ	0.0549(0.0254)	0.0534(0.0268)	0.0617(0.0098)	0.00
2	1-norm SVM	0.0697(0.0302)	0.0706(0.0324)	0.0828(0.0208)	1.07
	1-norm ψ	0.0562(0.0255)	0.0571(0.0269)	0.0702(0.0144)	0.88
	2-norm SVM	0.0746(0.0299)	0.0721(0.0310)	0.0902(0.0194)	0.00
	2-norm ψ	0.0589(0.0270)	0.0589(0.0270)	0.0753(0.0177)	0.00
3	1-norm SVM	0.0686(0.0301)	0.0778(0.0369)	0.0895(0.0251)	3.48
	1-norm ψ	0.0576(0.0262)	0.0685(0.0330)	0.0807(0.0203)	3.03
	2-norm SVM	0.0750(0.0279)	0.0899(0.0364)	0.1079(0.0238)	0.00
	2-norm ψ	0.0559(0.0254)	0.0787(0.0330)	0.0965(0.0207)	0.00
4	1-norm SVM	0.0699(0.0304)	0.0828(0.0362)	0.1015(0.0282)	7.40
	1-norm ψ	0.0554(0.0258)	0.0710(0.0332)	0.0882(0.0246)	7.01
	2-norm SVM	0.0738(0.0297)	0.1069(0.0379)	0.1330(0.0263)	0.00
	2-norm ψ	0.0550(0.0259)	0.1010(0.0376)	0.1242(0.0266)	0.00

To illustrate the restricted points of our improved algorithm visually, we plot one specific example in Figure 3. From this plot, we can see that our improved algorithm aims to find those points on the correct sides as well as far away from the decision boundaries. Then the algorithm restricts those points in the region of $yf(\boldsymbol{x}) \in (1, M]$ to reduce the number of integer variables. Notice that those points are the non-SVs for the SVM solution and they do not contribute to the resulting decision boundaries. Since the difference between the ψ loss and the hinge loss only occurs for the points in the region of $yf(\boldsymbol{x}) \in [-M, 1]$, the restriction procedure does not affect the solution of ψ-learning much. It is interesting to point out that the SVM boundary is somewhat off from the Bayes boundary because of several points on the wrong side. ψ-learning, on the other hand, corrects the SVM's solution by reassigning losses for the unrestricted points, including those close to the boundary as well as the ones on the wrong side. Since the SVM only has a small number of SVs, the improved algorithm can greatly simplify the computation of the MIP for ψ-learning. In fact, one can further simplify the algorithm by restricting the points with $yf(\boldsymbol{x}) \in (\epsilon, 1)$ for the SVM solution to be on the correct sides, i.e., $\delta_2^i = 1$.

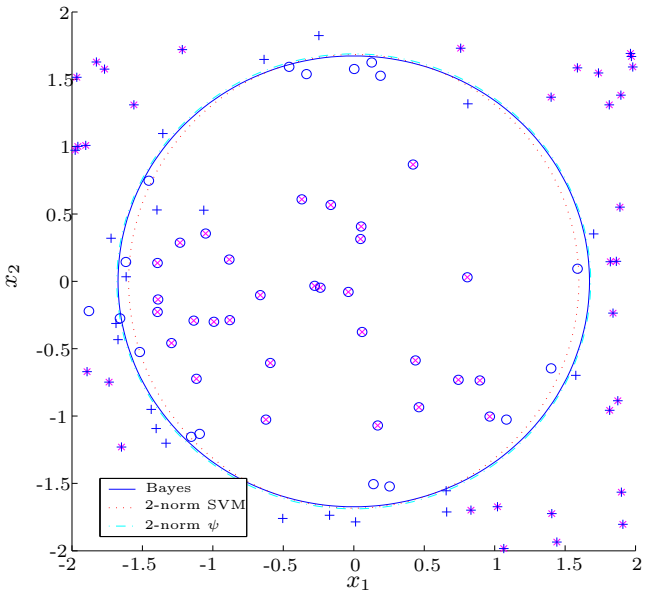

Figure 3. Boundaries of 2-norm SVM and ψ-learning for one selected sample of Case 1 in the simulation example in Section 5.1. The points for the corresponding two classes are in circles and pluses. The solid blue curve, red dotted curve, and cyan slashed curve represent boundaries of the Bayes rule, 2-norm SVM, and 2-norm ψ-learning respectively. The points marked by red crosses represent those restricted by the improved algorithm.

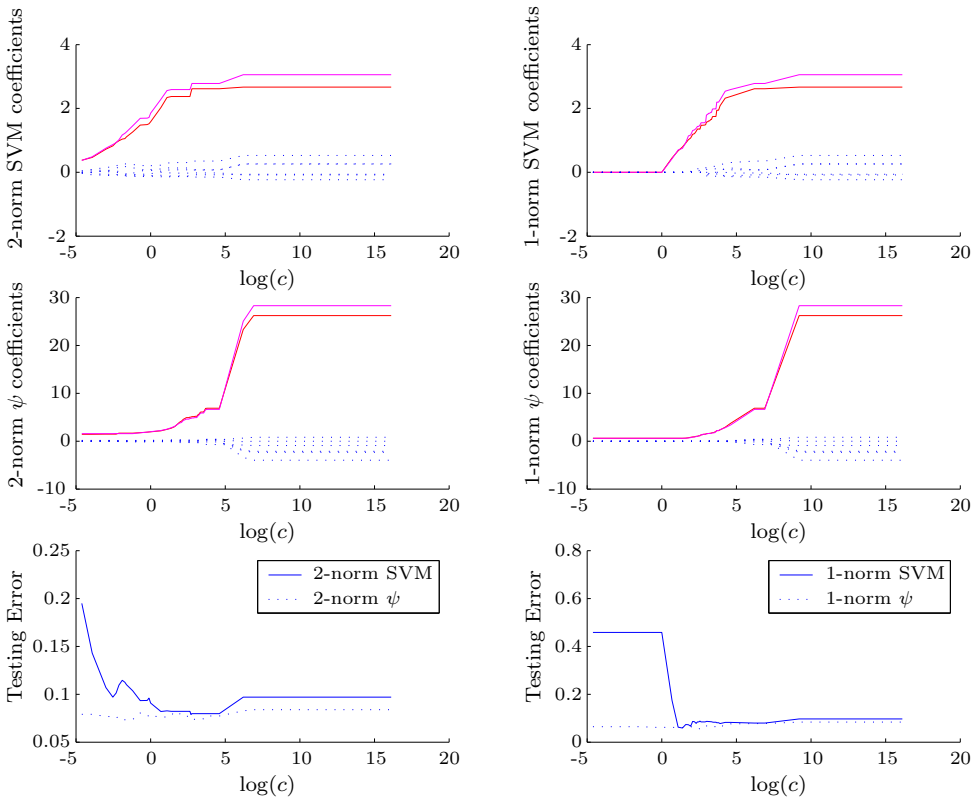

Figure 4. Results of SVM and ψ-learning for one selected sample of Case 3 in the simulation example in Section 5.1. The top four panels display the coefficients of the solutions of 1-norm and 2-norm SVM and ψ-learning. The solid lines represent coefficients of important variables while the dotted lines represent the noise variables. The bottom two panels illustrate the change of testing errors with different tuning parameters C. The solid lines are for the SVM and the dotted lines are for ψ-learning.

Figure 4 displays the solution paths of the 1-norm and 2-norm SVMs/ψ-learning as well as the corresponding testing errors for one specific sample with eight input variables, as in Case 3. The solution paths are generated by connecting the coefficients at the corresponding C values. The solid lines on the top four panels represent the coefficients of the two important variables, while dashed lines are for the remaining six noise variables. As discussed earlier, the 2-norm penalty does not have ability in variable selection. Therefore, the solutions for the 2-norm penalties have small coefficients for those noise variables even when the tuning parameter is close to 0. But the 1-norm penalty can effectively select the two important variables as C is small. The noise variables do not appear until the parameter gets relatively large. For C large, the solutions stay the

same as C increases since C balances the two components in (2.1). A large C forces the solution to have a good fit on the training data. However, once C is sufficiently large, the solution cannot further improve the second component of (2.1) and thus it is fixed for even larger C. This observation helps us to select a reasonable range of the tuning parameter. The bottom two panels of Figure 4 plot the testing errors of two methods. It appears here that ψ-learning is less sensitive to the choice of C in terms of classification performance than the SVM. One plausible explanation of the flat testing error curve of ψ-learning is the effect of restriction of the improved algorithm using a best tuned solution of the SVM. This may help us to develop an effective tuning procedure, and deserves further investigation.

5.2 Breast cancer data

We examine the performance of ψ-learning and the SVM on the Wisconsin Diagnostic Breast Cancer (WDBC) data. This dataset contains 569 observations, and each has a 30-dimensional real-valued input vector and a binary response (malignant or benign). More details can be found in Street et al. (1993) and at ftp://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-wisconsin/.

To examine performance of different methods, we randomly select 100 observations as a training dataset, another 100 observations for tuning, and the remaining 369 for testing. In particular, we compute classifiers using the training set with respect to a set of C in $[10^{-3}, 10^{3}]$, then find the one with the smallest tuning error to get the corresponding testing error via the testing set. Methods, including the 1-norm and 2-norm SVMs/ψ-learning, are implemented using all input variables. To carry out the ψ optimization, we apply the improved algorithm in Section 3.2. Specifically, we run the 1-norm and 2-norm SVMs on each training dataset, choose the best tuned solutions, and then restrict those observations with $yf > 1.1$ for the corresponding 1-norm and 2-norm ψ-learning. Here $yf > 1.1$ instead of $yf > 1$ is used for fewer restrictions and numerical accuracy. The average training/tuning/testing errors and average number of zero coefficients over 10 replications are reported in Table 3.

Table 3. Results of the WDBC example in Section 5.2.

	Error			Avg. # of
Method	Training	Tuning	Testing	zero coef.
1-norm SVM	2.50%	1.90%	4.74%	22.2
2-norm SVM	1.90%	1.60%	3.74%	0
1-norm ψ	1.20%	1.70%	4.20%	22.2
2-norm ψ	1.20%	2.70%	3.33%	0

The results indicate that the 2-norm SVM/ψ-learning have smaller testing errors than the corresponding methods with the 1-norm penalty. The 1-norm methods, on the other hand, can perform variable selections with only around 8 out of 30 variables selected. In this example, ψ-learning appears to perform better than the SVM.

6. Conclusion

In this paper, we convert the non-convex optimization problem of ψ-learning into a MIP problem. This algorithm is applicable for any general piecewise linear ψ losses. Through this framework, we study the properties of 1-norm and 2-norm ψ-learning in terms of classification accuracy as well as variable selection. In addition, we examine the effects of different a's on ψ-learning and make our recommendation on the choice of a. The close relationship between the hinge loss of the SVM and the ψ loss helps us to further improve the algorithm by restricting points unimportant in determining the solutions.

Although an improved version is offered to simplify the algorithm, it maybe problematic in situations where the dimension of the input variables is high. In that case, almost all training points may turn out to be SVs and the improved algorithm cannot provide any simplification. We may then have to reply on some heuristics of MIP to get reasonably good, but non-optimal, solutions.

The current paper focuses on binary ψ-learning. However, the same idea can be extended to the multicategory case. Then multicategory ψ-learning (Liu and Shen (2004)) can also be implemented via MIP. Development of the multicategory situation will be reported in a future paper.

Acknowledgement

The first author is grateful to Professor Xiaotong Shen for his guidance and encouragement. He would also like to thank Professors Gabor Pataki and Scott Proven for their helpful discussions on MIP. The authors are indebted to two referees, whose helpful comments and suggestions led to a much improved presentation.

References

Boser, B., Guyon, I. and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *The Fifth Annual Conference on Computational Learning Theory*, 142-152. ACM, Pittsburgh.

Bradley, P. and Mangasarian, O. (1998). Feature selection via concave minimization and support vector machines, In *ICML*'98 (Edited by J. Shavlik). Morgan Kaufmann.

Cristianini, N. and Shawe-Taylor, J. (1999). *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods.* Cambridge University Press.

Fourer, R., Gay, D. M. and Kernighan, B. W. (2002). *AMPL: A Modeling Language for Mathematical Programming.* Duxbury Press.

Garfinkel, R. S and Nemhauser, G. L. (1972). *Integer Programming.* Wiley, New York.

Kimeldorf, G. and Wahba, G. (1971). Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.* **33**, 82-95.

Liu, S., Shen, X. and Wong, W. (2005). Computational developments of ψ-learning. *Proceeding of The Fifth SIAM-ASA International Conference on Data Mining*, 1-12. Newport, CA.

Liu, Y. and Shen, X. (2004). Multicategory ψ-learning. *J. Amer. Statist. Assoc.* To appear.

Liu, Y., Shen, X. and Doss, H. (2005). Multicategory ψ-learning and support vector machine: computational tools. *J. Comput. Graph. Statist.* **14**, 219-236.

Shen, X., Tseng, G. C., Zhang, X. and Wong, W. H. (2003). On ψ-learning. *J. Amer. Statist. Assoc.* **98**, 724-734.

Street, W. N., Wolberg, W. H. and Mangasarian, O. L. (1993). Nuclear feature extraction for breast tumor diagnosis. *IS &T/SPIE* 1993 *International Symposium on Electronic Imaging: Science and Technology*, 861-870. San Jose, CA.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B* **58**, 267-288.

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

Wahba, G. (1990). Spline models for observational data. *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*, **59**.

Wahba, G. (1998). Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV. In *Advances in Kernel Methods: Support Vector Learning* (Edited by B. Schölkopf, C. J. C. Burges and A. J. Smola), 125-143. MIT Press.

Wolsey, L. A. and Nemhauser, G. L. (1999). *Integer and Combinatorial Optimization.* John Wiley, New York.

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

Department of Statistics and Operations Research, Carolina Center for Genome Sciences, University of North Carolina, CB 3260, Chapel Hill, NC 27599, U.S.A.

E-mail: yfliu@email.unc.edu

Department of Statistics and Operations Research, University of North Carolina, CB 3260, Chapel Hill, NC 27599, U.S.A.

E-mail: wuy@email.unc.edu

(Received April 2005; accepted August 2005)