BIOINFORMATICS ORIGINAL PAPER

Vol. 24 no. 3 2008, pages 412–419
doi:10.1093/bioinformatics/btm579

Data and text mining

Hybrid huberized support vector machines for microarray classification and gene selection

Li Wang[1], Ji Zhu[2,*] and Hui Zou[3]

[1]Ross School of Business, [2]Department of Statistics, University of Michigan, Ann Arbor, MI 48109 and [3]School of Statistics, University of Minnesota, Minneapolis, MN 55455, USA

Received on June 15, 2007; revised on October 09, 2007; accepted on November 18, 2007

Advance Access publication January 5, 2008

Associate Editor: David Rocke

ABSTRACT

Motivation: The standard L_2-norm support vector machine (SVM) is a widely used tool for microarray classification. Previous studies have demonstrated its superior performance in terms of classification accuracy. However, a major limitation of the SVM is that it cannot automatically select relevant genes for the classification. The L_1-norm SVM is a variant of the standard L_2-norm SVM, that constrains the L_1-norm of the fitted coefficients. Due to the singularity of the L_1-norm, the L_1-norm SVM has the property of automatically selecting relevant genes. On the other hand, the L_1-norm SVM has two drawbacks: (1) the number of selected genes is upper bounded by the size of the training data; (2) when there are several highly correlated genes, the L_1-norm SVM tends to pick only a few of them, and remove the rest.

Results: We propose a hybrid huberized support vector machine (HHSVM). The HHSVM combines the huberized hinge loss function and the elastic-net penalty. By doing so, the HHSVM performs automatic gene selection in a way similar to the L_1-norm SVM. In addition, the HHSVM encourages highly correlated genes to be selected (or removed) together. We also develop an efficient algorithm to compute the entire solution path of the HHSVM. Numerical results indicate that the HHSVM tends to provide better variable selection results than the L_1-norm SVM, especially when variables are highly correlated.

Availability: R code are available at http://www.stat.lsa.umich.edu/~jizhu/code/hhsvm/

Contact: jizhu@umich.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

The DNA microarray technology is a powerful tool for biological and medical research. It can detect thousands of gene expression levels simultaneously, providing a wealth of information. On the other hand, however, microarray datasets usually contain only a small number of samples. These characteristics pose great challenges for sample classification and gene selection. The support vector machine (SVM) is one of the most effective methods for microarray classification (Guyon *et al.*, 2002; Mukherjee *et al.*, 2000; Ramaswamy *et al.*, 2001); however, a major limitation of the SVM is that it cannot perform automatic gene selection. Since in microarray analysis, researchers are often interested in identifying informative genes, it is desirable to have a tool that can achieve both classification and gene selection simultaneously.

Guyon *et al.* (2002) proposed therecursive feature elimination (RFE) method for the SVM. The method, called the SVM-RFE, recursively eliminates irrelevant genes. At each step, the SVM-RFE trains for a SVM classifier, ranks the genes according to some score function and eliminates one or more genes with the lowest ranking scores. This process is repeated until classification accuracy starts to degrade. The SVM-RFE method is computationally intensive, especially for microarray datasets, which usually has a large number of genes. Bradley and Mangasarian (1998) proposed the L_1-norm SVM, which can automatically select genes via the L_1-norm regularization. The L_1-norm SVM, however, has two limitations:

(1) The number of selected genes is upper bounded by the sample size. Therefore, when the number of relevant genes exceeds the sample size, the L_1-norm SVM can only discover a portion of them.

(2) For the highly correlated and relevant genes, the L_1-norm SVM tends to pick only one or a few of them.

In this article, we propose an HHSVM for microarray classification. The HHSVM has the form of 'loss' + 'penalty', where it uses the huberized hinge function to measure the loss and the elastic-net penalty (Zou and Hastie, 2005) to control the complexity of the model. The HHSVM has several major benefits:

- Similar to the L_1-norm SVM, it automatically selects genes.
- The number of selected genes is no longer upper bounded by the sample size.
- When genes are highly correlated, they tend to be selected or removed together, i.e. the *grouping effect*.

Furthermore, inspired by the LAR/LASSO method (Efron *et al.*, 2004) and the general piecewise linear solution path strategy (Rosset and Zhu, 2007), we develop an efficient algorithm, which solves the entire solution path for every possible value of the regularization parameter.

*To whom correspondence should be addressed.

The rest of the article is organized as follows. In Section 2, we describe the HHSVM model. In Section 3, we develop the solution path algorithm. In Section 4, we apply the HHSVM method to simulation and real microarray datasets. We conclude the article with Section 5.

2 MODEL

2.1 The support vector machine

We briefly introduce the SVM and refer interested readers to (Burges, 1998; Evgeniou *et al.*, 1999; Vapnik, 1995) for detailed tutorials. Let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ represent the n input vectors, where $x_i \in \mathbb{R}^p$, and let $\{y_1, y_2, \ldots, y_n\}$ be the corresponding output labels, where $y_i \in \{1, -1\}$. The SVM finds a hyperplane that separates the two classes of data points by the largest distance:

$$\max_{\beta_0, \boldsymbol{\beta}} \quad \frac{1}{\|\boldsymbol{\beta}\|_2^2} \tag{1}$$

$$\text{subject to} \quad y_i(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\beta) \geq 1 - \varepsilon_i \tag{2}$$

$$\sum_{i=1}^{n} \varepsilon_i \leq C \tag{3}$$

$$\varepsilon_i \geq 0, \text{ for } i = 1, 2, \ldots, n, \tag{4}$$

where ε_i $(i = 1, \ldots, n)$ are slack variables, and $C \geq 0$ is a tuning parameter that controls the overlap. Denote the solution as $\widehat{\beta}_0$ and $\widehat{\beta}$, a new point with input vector x is then assigned with the label $\text{sign}(\widehat{\beta}_0 + \boldsymbol{x}^{\mathrm{T}}\widehat{\boldsymbol{\beta}})$.

Many researchers have recognized that the SVM can be equivalently transformed into the 'loss' + 'penalty' format, i.e.

$$\min_{\beta_0, \beta} \sum_{i=1}^{n} [1 - y_i(\boldsymbol{\beta}_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})]_+ + \frac{\lambda}{2}\|\beta\|_2^2, \tag{5}$$

where the loss function $(1 - \cdot)_+ = \max(1 - \cdot, 0)$ is called the *hinge loss* and the penalty is the L_2-norm of the fitted coefficients. $\lambda \geq 0$ is a regularization parameter, which controls the balance between the 'loss' and the 'penalty'. The same L_2-norm penalty has also been used in the ridge regression (Hoerl and Kennard, 1970) and neural networks. By shrinking the magnitude of the coefficients, the L_2-norm penalty reduces the variance of the estimated coefficients and may result in better prediction accuracy.

2.2 The L_1-norm SVM

Bradley and Mangasarian (1998) proposed to replace the L_2-norm penalty in the standard SVM with the L_1-norm penalty (Tibshirani, 1996):

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} [1 - y_i(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})]_+ + \lambda\|\boldsymbol{\beta}\|_1. \tag{6}$$

Similar to the L_2-norm penalty, the L_1-norm penalty also reduces the variance of the estimates and improves the prediction accuracy. Furthermore, due to the singularity of the L_1-norm function, when λ_1 is large enough, it tends to shrink the coefficients of irrelevant variables to exactly zero. When λ changes, different coefficients are set to zero, hence the L_1-norm penalty allows a kind of automatic continuous variable selection.

2.3 The hybrid huberized SVM

Zou and Hastie (2005) argued that the L_1-norm penalty has two major limitations:

(1) The number of variables selected by the L_1-norm penalty is upper bounded by the sample size n. In microarray analysis, we nearly always have the case $p \gg n$, but the L_1-norm SVM can identify at most n relevant genes.

(2) For highly correlated and relevant variables, the L_1-norm penalty tends to select only one or a few of them. In microarray analysis, genes sharing the same biological pathway tend to have highly correlated expression levels. It is often desirable to identify all, rather than a few, of them if they are related to the underlying biological process.

To overcome these limitations, Zou and Hastie (2005) proposed the elastic-net penalty:

$$\lambda_1\|\boldsymbol{\beta}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\beta}\|_2^2,$$

which is a hybrid of the L_1-norm and the L_2-norm penalties. The elastic-net penalty retains the variable selection feature of the L_1-norm penalty, and the number of selected variables is no longer bounded by n. Furthermore, the elastic-net penalty tends to provide similar estimated coefficients for highly correlated variables, i.e. the grouping effect; hence, highly correlated variables tend to be selected or removed together.

In this article, we apply the elastic-net penalty to the SVM and propose the HHSVM:

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} \phi\big(y_i(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})\big) + \lambda_1\|\boldsymbol{\beta}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\beta}\|_2^2, \tag{7}$$

where λ_1, $\lambda_2 \geq 0$ are regularization parameters. Increasing λ_1 tends to eliminate more irrelevant variables; and increasing λ_2 makes the 'grouping effect' more prominent, which we will illustrate by a theorem at the end of this section.

Notice that instead of using the standard hinge loss function of the SVM, we use the huberized hinge loss function (Rosset and Zhu, 2007) to measure 'badness-of-fit':

$$\phi(yf) = \begin{cases} 0, & \text{for } yf > 1, \\ (1 - yf)^2/2\delta, & \text{for } 1 - \delta < yf \leq 1, \\ 1 - yf - \delta/2, & \text{for } yf \leq 1 - \delta, \end{cases}$$

where $\delta \geq 0$ is a pre-specified constant.

Figure 1 compares the standard hinge loss function and the huberized hinge loss function. Notice that they have similar shapes: when yf decreases (misclassification), they both increase linearly; when yf is bigger than 1, they are both equal to zero. Therefore, we expect the classification performance of these two functions should be similar to each other. Also notice that, unlike the hinge loss, the huberized hinge loss function is differentiable everywhere. As we will see in the next section, this differentiability can significantly reduce the computational cost for the HHSVM algorithm.

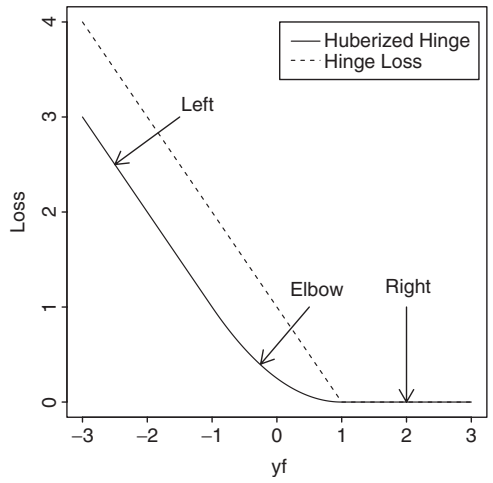

Fig. 1. The hinge and the huberized hinge loss functions (with $\delta = 2$). Note that the *Elbow* corresponds to the region $(1-\delta; 1)$; the *Left* and the *Right* correspond to the regions $(-\infty, 1-\delta)$ and $(1,\infty)$, respectively.

Before delving into details for the HHSVM algorithm, we illustrate the 'grouping effect' with the following theorem:

THEOREM 1. *Let $\widehat{\beta}_0$ and $\widehat{\boldsymbol{\beta}}$ denote the solution for problem (7). For any pair (j, j'), we have*

$$|\widehat{\beta}_j - \widehat{\beta}_{j'}| \le \frac{1}{\lambda_2}\|\boldsymbol{x}_j - \boldsymbol{x}_{j'}\|_1 = \frac{1}{\lambda_2}\sum_{i=1}^{n}|x_{ij} - x_{ij'}|. \tag{8}$$

If the input vector $\boldsymbol{x}_j$ and $\boldsymbol{x}_{j'}$ are centered and normalized, then

$$|\widehat{\beta}_j - \widehat{\beta}_{j'}| \le \frac{\sqrt{n}}{\lambda_2}\sqrt{2(1-\rho)}, \tag{9}$$

where ρ is the sample correlation between $\boldsymbol{x}_j$ and $\boldsymbol{x}_{j'}$.

Details of the proof are in the Supplementary Material. Theorem 1 suggests that highly correlated variables tend to have similar estimated coefficients; hence, they tend to be selected or removed together when λ_2 is sufficiently large.

3 ALGORITHM

The HHSVM involves two tuning parameters λ_1 and λ_2. According to our experience, the prediction performance is often more sensitive to λ_1, since it has more impact on selecting variables; therefore, the value of λ_1 must be chosen more carefully. For parameter tuning, one usually specifies a number of candidates, tests each of them and chooses the best one according to some criterion. However, this trial-and-error approach is computationally expensive and the optimal parameter setting can be easily missed. In this section, we propose an efficient algorithm, which solves the entire solution path for every possible value of λ_1 (when λ_2 is fixed). The algorithm is based on the fact that the solution $(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}})$ is a piecewise linear function (in a multi-dimensional space) with respect to λ_1.

3.1 Problem setup

Since the huberized hinge loss function has different definitions in different regions, for simplicity we define each region as:

- $\mathcal{R} = \{i: y_i f(x_i) > 1\}$ (Right)
- $\mathcal{E} = \{i: 1-\delta < y_i f(x_i) \le 1\}$ (Elbow)
- $\mathcal{L} = \{i: y_i f(x_i) \le 1-\delta\}$ (Left)

where $f(\boldsymbol{x}_i) = \beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}$. We also define the indices for non-zero β_j as the active set $\mathcal{A}$:

- $\mathcal{A} = \{j: \beta_j \ne 0, j = 1,2,\ldots,p\}$ (Active)

Since problem (7) is an unconstrained convex optimization problem, for any optimal solution the derivatives of its objective function must be zero. Setting the derivative with respect to β_0 to zero, we get:

$$\sum_{i\in\mathcal{E}}\frac{1}{\delta}(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta} - y_i) - \sum_{i\in\mathcal{L}} y_i = 0. \tag{10}$$

Setting the derivative with respect to β_j $(j \in \mathcal{A})$ to zero, we get:

$$\sum_{i\in\mathcal{E}}\frac{1}{\delta}(\beta_0 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta} - y_i)x_{ij} - \sum_{i\in\mathcal{L}} y_i x_{ij} + \lambda_2\beta_j + \lambda_1\mathrm{sign}(\beta_j) = 0, \text{ for } j \in \mathcal{A}. \tag{11}$$

In the linear system (10)–(11), there are $|\mathcal{A}|+1$ unknowns and $|\mathcal{A}|+1$ equations, where $|\mathcal{A}|$ represents the number of elements in set $\mathcal{A}$. Therefore, the solution β_0 and β_j $(j \in \mathcal{A})$ can be uniquely determined, given that the system is non-singular.

When sets $\mathcal{L}$, $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{A}$ are fixed, the structure of the linear system (10)–(11) is also fixed. Under this condition, β_0 and β_j $(j \in \mathcal{A})$ are linear functions of λ_1, which can be seen from (11). However, as λ_1 decreases, sooner or later some of the sets $\mathcal{L}$, $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{A}$ will change. We call this an *event*. After an 'event' occurs the new β_0 and $\boldsymbol{\beta}_j$ $(j \in \mathcal{A})$ are still linear functions of λ_1, but their derivatives with respect to λ_1 will change. Therefore, the entire solution path is piecewise linear in λ_1, and between any two consecutive events, β_0 and $\boldsymbol{\beta}_j$ $(j \in \mathcal{A})$ change linearly with λ_1. Each 'event' corresponds to a kink on the piecewise linear solution path.

Our algorithm starts from $\lambda_1 = \infty$, continuously decreases λ_1, solves the solutions along this path, and terminates if λ_1 reaches 0. The algorithm provides the solutions on each kink. For any λ_1 between two consecutive kinks, the solution can be precisely obtained using linear interpolation. Table 1 shows the outline of the our algorithm.

3.2 Initial solution

The algorithm starts from $\lambda_1 = \infty$. From the objective function of problem (7), we can see that $\boldsymbol{\beta} = \mathbf{0}$ at this stage. Therefore, the problem reduces to:

$$\min_{\beta_0}\sum_{i=1}^{n}\phi(y_i\beta_0), \tag{12}$$

which involves only one parameter β_0. Since the huberized hinge loss function is convex and differentiable everywhere, this one-variable optimization problem can be easily solved. In fact,

Table 1. Outline of the HHSVM algorithm

Initialization: calculate β_0^0, β_j^0 ($j=1,\ldots,p$), λ_1^0, $\mathcal{A}^0$,$\mathcal{L}^0$, $\mathcal{R}^0$, $\mathcal{E}^0$ according to Section 3.2 and set $k=0$.
Step 1: solve the linear systems (13)–(14).
Step 2: calculate $\Delta\lambda_1$ and determine the next event according to Section 3.3.
Step 3: if the stopping criterion is satisfied, terminate the algorithm.
Step 4: otherwise, let $k=k+1$ and update β_0^k, β_j^k ($j=1,\ldots,p$), λ_1^k,$\mathcal{A}^k$, $\mathcal{L}^k$,$\mathcal{R}^k$ and $\mathcal{E}^k$ according to Section 3.3.
Step 5: goto Step 1.

one can easily develop an analytical solution for the initial β_0, but due to the lack of space, we skip the details.

Let β_0^0 represent the optimal solution when $\lambda_1=\infty$. Hence $\mathcal{A}=\emptyset$ ($\beta=0$) and sets $\mathcal{L}$, $\mathcal{E}$ and $\mathcal{R}$ can be determined using the values of $y_i\beta_0^0$ ($i=1,\ldots,n$). Reducing λ_1 tends to increase the magnitude of β, and we can find a critical point, denoted as λ_1^0, where exactly one β_j ($j=1,\ldots,p$) joins $\mathcal{A}$, i.e. the estimate β_j will become non-zero if λ_1 is further reduced.

Since Equation (11) must hold for any $j\in\mathcal{A}$, we can determine the critical point λ_1^0 by:

$$\lambda_1^0=\max_{j\in\{1,\ldots,p\}}\left|\sum_{i\in\mathcal{E}}\frac{1}{\delta}(\beta_0^0-y_i)x_{ij}-\sum_{i\in\mathcal{L}}y_ix_{ij}\right|.$$

The corresponding variable that joins $\mathcal{A}$ first can be identified as:

$$j^*=\arg\max_{j\in\{1,\ldots,p\}}\left|\sum_{i\in\mathcal{E}}\frac{1}{\delta}(\beta_0^0-y_i)x_{ij}-\sum_{i\in\mathcal{L}}y_ix_{ij}\right|,$$

and the sign for β_{j^*} is:

$$\text{sign}(\beta_{j^*})=\text{sign}\left(-\sum_{i\in\mathcal{E}}\frac{1}{\delta}(\beta_0^0-y_i)x_{ij^*}+\sum_{i\in\mathcal{L}}y_ix_{ij^*}\right).$$

Let the superscript k indicate the iteration number and $k=0$ for the initial stage. Now, we have $\mathcal{A}^k=\{j^*\}$, $\beta_0^k=\beta_0^0$, $\beta_j^k=0$ ($j=1,\ldots,p$), $\lambda_1^k=\lambda_1^0$, and the $\mathcal{L}^k$, $\mathcal{R}^k$, $\mathcal{E}^k$ are determined by $y_i\beta_0^0$ ($i=1,\ldots,n$).

We emphasize again that the initial solution can be easily determined here because of the differentiability of the huberized hinge loss function; however, this is not the case for the hinge loss function. Since the hinge loss is not everywhere differentiable, it is difficult to determine the critical point λ_1^0 and the corresponding $\mathcal{A}^0$, $\mathcal{L}^0$, $\mathcal{E}^0$ and $\mathcal{R}^0$ at the initial stage. Zhu *et al.* (2004) proposed an approach for solving the initial solution of the L_1-norm SVM. The approach was based on linear programming, but it can be computationally intensive, especially when p is large.

3.3 Solution path

The algorithm continuously decreases λ_1 until it reaches 0. Let $\lambda_1=\lambda_1^k+\Delta\lambda_1$, where $\Delta\lambda_1<0$. When λ_1 is reduced by a small enough amount, sets $\mathcal{L}$, $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{A}$ do not change, because of the continuity of $f(x)$ with respect to λ_1. Therefore, based on the systems (10)–(11), the derivatives of β_0 and β_j ($j\in\mathcal{A}$) with respect to λ_1 can be solved from the following equations:

$$\sum_{i\in\mathcal{E}}\left(\frac{\Delta\beta_0}{\Delta\lambda_1}+\sum_{k\in\mathcal{A}}x_{ik}\frac{\Delta\beta_k}{\Delta\lambda_1}\right)=0, \tag{13}$$

$$\sum_{i\in\mathcal{E}}\frac{1}{\delta}\left(\frac{\Delta\beta_0}{\Delta\lambda_1}+\sum_{k\in\mathcal{A}}x_{ik}\frac{\Delta\beta_k}{\Delta\lambda_1}\right)x_{ij}+\lambda_2\frac{\Delta\beta_j}{\Delta\lambda_1}+\text{sign}(\beta_j)=0,\ \text{for } j\in\mathcal{A}. \tag{14}$$

Since there are $|\mathcal{A}|+1$ unknowns and $|\mathcal{A}|+1$ equations, $\frac{\Delta\beta_0}{\Delta\lambda_1}$ and $\frac{\Delta\beta_j}{\Delta\lambda_1}$ ($j\in\mathcal{A}$) are uniquely determined, given that the system is non-singular. Then when $|\Delta\lambda_1|$ is sufficiently small, the solution and fitted values are linear in λ_1:

$$\beta_0=\beta_0^k+\frac{\Delta\beta_0}{\Delta\lambda_1}(\lambda_1-\lambda_1^k),$$

$$\beta_j=\beta_j^k+\frac{\Delta\beta_j}{\Delta\lambda_1}(\lambda_1-\lambda_1^k),\ \text{for } j\in\mathcal{A},$$

$$f(\boldsymbol{x}_i)=f^k(\boldsymbol{x}_i)+\left(\frac{\Delta\beta_0}{\Delta\lambda_1}+\sum_{j\in\mathcal{A}}\frac{\Delta\beta_j}{\Delta\lambda_1}\right)(\lambda_1-\lambda_1^k).$$

If we keep reducing λ_1, some of the sets $\mathcal{L}$, $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{A}$ will change. We call this an *event*, and four types of 'events' may occur:

(1) A point i reaches the boundary between $\mathcal{L}$ and $\mathcal{E}$.
(2) A point i reaches the boundary between $\mathcal{R}$ and $\mathcal{E}$.
(3) A parameter β_j becomes zero (j leaves $\mathcal{A}$).
(4) A zero-valued parameter β_j becomes non-zero (j joins $\mathcal{A}$).

The boundary between $\mathcal{L}$ and $\mathcal{E}$ is $1-\delta$ and the boundary between $\mathcal{R}$ and $\mathcal{E}$ is 1. Therefore, when $y_i\ f(x_i)$ for a point crosses $1-\delta$ or 1, one of the first two 'events ' occurs. To determine the step size $\Delta\lambda_1$ for the first 'event', for each $i\in\mathcal{L}$ or $\mathcal{E}$ we calculate:

$$\Delta\lambda_1^i=\frac{1-\delta-y_if^k(\boldsymbol{x}_i)}{y_i(\frac{\Delta\beta_0}{\Delta\lambda_1}+\sum_{j\in\mathcal{A}}\frac{\Delta\beta_j}{\Delta\lambda_1})}.$$

Let $\Delta\lambda_{1,1}$ represent the step size for the first 'event'. Since λ_1 only decreases, $\Delta\lambda_{1,1}\le 0$ and its value should be determined by:

$$\Delta\lambda_{1,1}=\max\{\Delta\lambda_1^i: i\in\ \mathcal{L}\text{ or }\mathcal{E},\ \Delta\lambda_1^i\le 0\}.$$

Similarly, for the second 'event' we calculate:

$$\Delta\lambda_1^i=\frac{1-y_if^k(\boldsymbol{x}_i)}{y_i(\frac{\Delta\beta_0}{\Delta\lambda_1}+\sum_{j\in\mathcal{A}}\frac{\Delta\beta_j}{\Delta\lambda_1})},$$

for each $i\in\mathcal{R}$ or $\mathcal{E}$. And the step size for the second 'event' is:

$$\Delta\lambda_{1,2}=\max\{\Delta\lambda_1^i: i\in\mathcal{R}\text{ or }\mathcal{E},\ \Delta\lambda_1^i\le 0\}.$$

When a non-zero β_j reduces to 0, the third 'event' occurs. Therefore, we calculate for each $j\in\mathcal{A}$:

$$\Delta\lambda_1^j=-\beta_j^k/\frac{\Delta\beta_j}{\Delta\lambda_1}.$$

The step size for the third 'event' is:

$$\Delta\lambda_{1,3} = \max\{\Delta\lambda_1^j : j \in \mathcal{A}, \Delta\lambda_1^j \le 0\}.$$

The fourth 'event' (a zero-valued β_j becomes non-zero) is a little complicated to determine. For $j = 1, \ldots, p$, we define:

$$C_j = \sum_{i\in\mathcal{E}} \frac{1}{\delta}(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta} - y_i)x_{ij} - \sum_{i\in\mathcal{L}} y_i x_{ij} + \lambda_2\beta_j,$$

which is part of the left-hand side of Equation (11). From (11) and the initial conditions, we know that:

- $|C_j| = \lambda_1$, $\text{sign}(C_j) = -\text{sign}(\beta_j)$, for $j \in \mathcal{A}$;
- $|C_j| < \lambda_1$, for $j \notin \mathcal{A}$.

Notice that when $|\Delta\lambda_1|$ is sufficiently small, C_j is also a linear function of λ_1:

$$C_j = C_j^k + \left[\sum_{i\in\mathcal{E}} \frac{1}{\delta}\left(\frac{\Delta\beta_0}{\Delta\lambda_1} + \sum_{k\in\mathcal{V}} \frac{\Delta\beta_k}{\Delta\lambda_1}\right)x_{ij}\right](\lambda_1 - \lambda_1^k).$$

As λ_1 decreases, the value for a $|C_j|$ ($j \notin \mathcal{A}$) will first meet the decreasing λ_1, after which the corresponding β_j will become non-zero if we further reduce λ_1.

Therefore, to determine the step size for the fourth 'event', for each $j \notin \mathcal{A}$ we calculate:

$$\Delta\lambda_1^j = \begin{cases} \dfrac{C_j^k + \lambda_1^k}{-1 - \sum_{i\in\mathcal{E}} \frac{1}{\delta}\left(\frac{\Delta\beta_0}{\Delta\lambda_1} + \sum_{k\in\mathcal{V}} \frac{\Delta\beta_k}{\Delta\lambda_1}\right)x_{ij}}, \\ \text{if } C_j \text{ decreases as } \lambda_1 \text{ is reduced;} \\ \dfrac{C_j^k - \lambda_1^k}{1 - \sum_{i\in\mathcal{E}} \frac{1}{\delta}\left(\frac{\Delta\beta_0}{\Delta\lambda_1} + \sum_{k\in\mathcal{V}} \frac{\Delta\beta_k}{\Delta\lambda_1}\right)x_{ij}}, \text{otherwise.} \end{cases}$$

The step size $\Delta\lambda_{1,4}$ for the fourth 'event' is:

$$\Delta\lambda_{1,4} = \max\{\Delta\lambda_1^j : j \notin \mathcal{A}\}.$$

After solving for $\Delta\lambda_{1,1}$, $\Delta\lambda_{1,2}$, $\Delta\lambda_{1,3}$ and $\Delta\lambda_{1,4}$, the overall step size $\Delta\lambda_1$ can be obtained:

$$\Delta\lambda_1 = \max\{\Delta\lambda_{1,1}, \Delta\lambda_{1,2}, \Delta\lambda_{1,3}, \Delta\lambda_{1,4}\}.$$

We then update $\mathcal{L}$, $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{A}$ according to the corresponding 'event', and also update λ_1, β_0, β_j, C_j and the fitted value $f(x_i)$. Finally, let $k = k + 1$ and the algorithm goes to the next iteration: solving the linear systems (13)–(14) and calculating the step size $\Delta\lambda_1$. This entire process is repeated, until λ_1 reaches 0.

Between any two consecutive 'events', the solutions are linear in λ_1, and after an 'event' occurs, the derivative of the solution with respect to λ_1 is changed. Therefore, the solution path is piecewise linear in λ_1, where each 'event' corresponds to a kink on the path. The algorithm provides solutions at these kinks, and for any λ_1 between two consecutive kinks the solutions can be calculated precisely via linear interpolation.

3.4 Computational cost

The major computational cost in each iteration comes from solving the linear system (13)–(14). Since this system has $|\mathcal{A}| + 1$ unknowns, the computational cost seems to be $O(|\mathcal{A}|^3)$ for each iteration. However, between any two iterations, only one element is changed for sets $\mathcal{L}$, $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{A}$, so using inverse updating and downdating the computational cost can be reduced to $O(|\mathcal{A}|^2)$. It is difficult to predict the number of iterations. According to our experience, $O(\min(n,p))$ is a reasonable estimate. The heuristic is that the algorithm needs $O(n)$ to move all the points to $\mathcal{R}$ and $O(p)$ steps to add all the parameters into $\mathcal{A}$. Since p is the upper bound for $|\mathcal{A}|$, the overall computational cost is upper bounded by $O(\min(n,p)p^2)$.

Table 2. Comparison of test errors

	Scenario I	Scenario II
L_2-norm SVM	0.214 (0.004)	0.160 (0.003)
L_1-norm SVM	0.143 (0.007)	0.160 (0.002)
HHSVM	0.133 (0.005)	0.143 (0.001)

The number of training observations is 50, the number of total input variables is 300 and the number of relevant variables is 10. The results are averages of test errors over 100 repetitions on a 10 000 test set, and the numbers in parentheses are the corresponding standard errors. In 'Scenario I', the input variables are independent, and in 'Scenario II', the relevant variables are highly correlated.

4 NUMERICAL RESULTS

In this section, we use both simulation data and real microarray datasets to illustrate the HHSVM.

4.1 Simulation

The main purpose of the simulation is to demonstrate that when input variables are independent, the L_1-norm SVM and the HHSVM perform similarly, but the HHSVM can have an advantage when the variables are highly correlated, especially at identifying the relevant variables.

We first consider the scenario where all input variables are independent. The '+' class has a normal distribution with mean

$$\boldsymbol{\mu}_+ = (\underbrace{0.5, \ldots, 0.5}_{10}, \underbrace{0, \ldots, 0}_{p-10})^{\mathrm{T}},$$

and covariance $\boldsymbol{\Sigma} = \boldsymbol{I}_{p\times p}$. The '−' class has a similar distribution except that

$$\boldsymbol{\mu}_- = (\underbrace{-0.5, \ldots, -0.5}_{10}, \underbrace{0, \ldots, 0}_{p-10})^{\mathrm{T}}.$$

So the Bayes optimal classification rule only depends on $x_1, \ldots, x_{10}$, and the Bayes error is about 0.06, independent of the dimension p.

We generated $50 = 25 + 25$ training data, each input $\boldsymbol{x}_i$ is a $p = 300$-dimensional vector. We compared the standard L_2-norm SVM, the L_1-norm SVM, and the HHSVM. We used 50 validation data to select the tuning parameters for each method, then applied the selected models to a separate 10 000 testing dataset. Each experiment was repeated 100 times. The means of the prediction errors and the corresponding standard errors (in parentheses) are summarized in Table 2. As we can see, the prediction errors of the L_1-norm SVM and the HHSVM are similar and both are better than that of the

Table 3. Comparison of variable selection

	Scenario I		Scenario II	
	q_{signal}	q_{noise}	q_{signal}	q_{noise}
L_1-norm SVM	7.2 (0.3)	6.5 (1.4)	2.5 (0.2)	2.9 (1.2)
HHSVM	7.6 (0.3)	7.1 (1.3)	7.9 (0.4)	3.3 (2.5)

The setup are the same as those described in Table 2. q_{signal} is the number of selected relevant variables, and q_{noise} is the number of selected noise variables.

L_2-norm SVM. This is due to the fact that the L_2-norm SVM uses all input variables, and its prediction accuracy is 'polluted' by the noise variables.

Besides the prediction error, we also compared the selected variables of the L_1-norm SVM and the HHSVM (the L_2-norm SVM keeps all input variables). In particular, we consider $q_{\text{signal}} =$ number of selected relevant variables, and $q_{\text{noise}} =$ number of selected noise variables. The results are in Table 3. Again, we see that the L_1-norm SVM and the HHSVM perform similarly; both are able to identify the relevant variables and remove most of the irrelevant variables.

Now we consider the scenario when the relevant variables are correlated. Similar as the independent scenario, the '+' class has a normal distribution, with mean

$$\boldsymbol{\mu}_+ = (\underbrace{1, \ldots, 1}_{10}, \underbrace{0, \ldots, 0}_{p-10})^{\mathrm{T}}$$

and covariance

$$\boldsymbol{\Sigma} = \begin{pmatrix} \Sigma^*_{10\times 10} & 0_{10\times(p-10)} \\ \mathbf{0}_{(p-10)\times 10} & I_{(p-10)\times(p-10)} \end{pmatrix},$$

where the diagonal elements of Σ^* are 1 and the off-diagonal elements are all equal to $\rho = 0.8$. The '−' class has a similar distribution except that

$$\boldsymbol{\mu}_- = (\underbrace{-1, \ldots, -1}_{10}, \underbrace{0, \ldots, 0}_{p-10})^{\mathrm{T}}.$$

So the Bayes optimal classification rule depends on $x_1, \ldots, x_{10}$, which are highly correlated. The Bayes error is 0.13, independent of the dimension p.

Again, we considered $n = 25 + 25$ and $p = 300$. Each experiment was repeated 100 times. The results for the prediction errors are shown in Table 2. Now the performance of the HHSVM is slightly better than that of the L_1-norm SVM, after taking into account the standard error. The right part of Table 3 compares the variables selected by the L_1-norm SVM and the HHSVM, which sheds some light on what happened. Both the L_1-norm SVM and the HHSVM are able to identify relevant variables. However, when the relevant variables are highly correlated, the L_1-norm SVM tends to keep only a small subset of the relevant variables, and overlook the others, while the HHSVM tends to identify all of them, due to the grouping effect. Both methods seem to work well in removing irrelevant variables.

Table 4. Results on 100 randomsplits of the original datasets: the upper part is for the colon cancer dataset, and the lower part is for the lung cancer dataset

	Test error	Number of genes
Colon cancer dataset		
SVM	14.65% (1.34%)	All
SVM-RFE	17.10% (0.87%)	64
L_1-norm SVM	15.84% (1.02%)	14.4 (3.6)
HHSVM	12.69% (0.93%)	94.5 (42.4)
Lung cancer dataset		
SVM	29.0% (0.6%)	All
SVM-RFE	28.4% (0.7%)	128
L_1-norm SVM	29.3% (0.4%)	7.2 (0.3)
HHSVM	27.8% (0.7%)	22.6 (1.9)

The numbers in the parentheses are the corresponding standard errors.

4.2 Real Data Analysis

The first dataset we considered is the colon cancer dataset in Alon and Barkai (1999). This dataset consists of 62 samples (40 colon cancer tumors and 22 normal tissues). Each sample consists of $p = 2000$ genes. We randomly split the samples into the training and the test sets for 100 times; for each split, 42 samples (27 cancer samples and 15 normal tissues) were used for training and the rest 20 samples (13 cancer samples and 7 normal tissues) were for testing. For each split, we applied four methods, the standard L_2-norm SVM, the L_1-norm SVM, the SVM-RFE and the HHSVM, to the training data. Tuning parameters were chosen using 10-fold cross-validation on the training data, and the chosen models were evaluated on the test data. For the standard L_2-norm SVM, we tried different kernels and found the linear kernel has the best performance. For the SVM-RFE, we used the same strategy as in Guyon *et al*. (2002), i.e. eliminating 10% ofthe remaining genes in each iteration. The results are summarized in the upper part of Table 4. We can see that the HHSVM seemed to have a slightly better classification accuracy than other methods. However, we note that this is not necessarily conclusive, for the sample size is small.

Tables 5–6 summarize the genes that were 'frequently' selected by the L_1-norm SVM and the HHSVM. As we can see, only 5 genes were selected for more than 50 times by the L_1-norm SVM, while 40 genes were selected for more than 50 times by the HHSVM. The selection frequency implies that these genes may have certain power in differentiating the colon cancer samples from the normal tissues, however, the L_1-norm SVM was not always able to identify them. Figure 2 shows the number of selected genes for each selection frequency out of the 100 random splits. Again, we can see that the HHSVM is more stable than the L_1-norm SVM in terms of selecting genes, i.e. over the 100 random splits, for important genes, the HHSVM selects them more frequently than the L_1-norm SVM (lower part of Fig. 2), while for unimportant genes, the HHSVM selects them less frequently (upper part of Fig. 2).

To assess the stability of prediction, we also recorded the frequency that each sample, as a test observation, was

Table 5. The 32 most frequently selected genes by the L_1-SVM from the colon cancer dataset

Gene number	Selection frequency	Gene number	Selection frequency	Gene number	Selection frequency	Gene number	Selection frequency
249	82	1993	27	515	16	897	9
377	65	245	25	281	16	698	8
765	64	83	24	1771	15	1546	7
1870	61	365	22	1892	13	187	6
1582	54	802	21	66	13	266	6
1772	33	1325	20	1042	12	780	5
1423	30	75	19	1047	10	1058	5
625	27	493	17	822	9	1644	5

'Selection Frequency' is the number of times that the corresponding gene was selected out of 100 random splits of the training and test data.

Table 6. The 100 most frequently selected genes by the HHSVM from the colon cancer dataset

Gene number	Selection frequency	Gene number	Selection frequency	Gene number	Selection frequency	Gene number	Selection frequency
245	100	1836	66	1666	42	617	35
249	100	1153	65	75	41	780	35
377	100	70	63	164	41	795	34
493	100	1582	62	444	40	1048	34
765	100	799	60	1002	40	1256	34
1423	100	802	59	1440	39	1411	34
267	98	897	58	26	39	1679	34
286	97	527	56	49	39	1843	34
698	93	625	55	411	39	15	33
1772	92	1110	54	419	39	237	33
66	90	350	53	467	38	260	33
1870	87	1671	52	513	38	261	33
1325	84	1042	51	590	38	262	33
1993	83	1473	50	679	38	263	33
43	80	1644	50	812	38	279	33
1221	78	83	49	878	37	306	33
1058	77	561	48	1599	37	365	33
1873	76	627	48	1727	37	621	32
792	73	1024	47	1887	37	661	32
822	72	1635	47	807	36	682	32
14	71	639	46	1047	36	1102	32
281	70	1567	45	1196	36	1258	32
1346	70	175	44	341	35	1370	32
187	69	391	43	427	35	1771	32
1892	68	1073	42	581	35	1798	32

'Selection Frequency' is the number of times that the corresponding gene was selected out of 100 random splits of the training and test data.

correctly classified. For example, if a sample appears in 70 test sets among the 100 random splits, and out of the 70 predictions, the sample was correctly classified for 60 times, then we recorded 60/70 for this sample. The results are shown in Figure 3. As we can see, for most samples, both the L_1-norm SVM and the HHSVM classified them correctly for most of the random splits, but there are also some samples where both methods

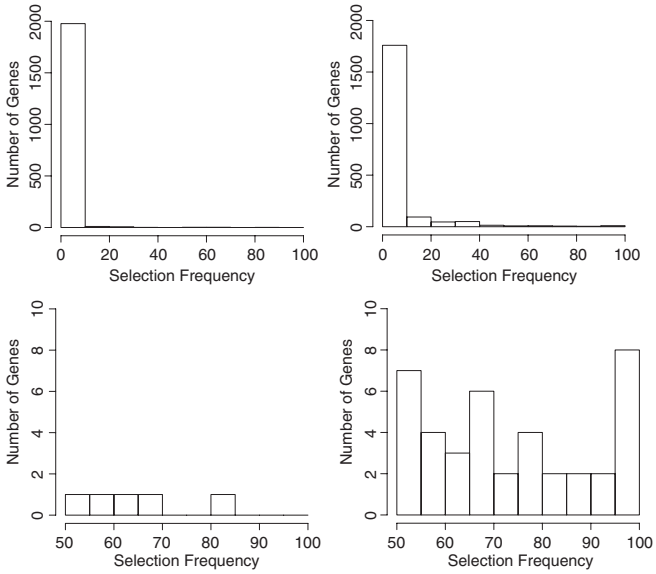

Fig. 2. The upper part shows the number of selected genes versus the selection frequency for the L_1-norm SVM (left) and the HHSVM (right) on 100 random splits of the colon cancer dataset. The lower part 'zooms in' to the region where the 'selection frequency' is bigger than 50 (the left panel is for the L_1-norm SVM and the right panel is for the HHSVM).

tended to misclassify. Overall, the HHSVM seems to be slightly more stable than the L_1-norm SVM in terms of prediction.

The second dataset we considered is a more recent lung cancer dataset (Potti *et al.*, 2006), which consists of 198 samples (54 cancer tumors and 144 normal tissues). Each sample consists of expression measurements on 22 215 genes. We randomly splitted the data into training and test sets, with sample sizes 137 (37 cancer samples and 100 normal tissues) and 61, respectively. We repeated it 100 times. The lower part of Table 4 summarizes the results. The gene selection behavior of the L_1-norm SVM and the HHSVM for the lung cancer dataset are similar to those for the colon cancer dataset. Due to the lack of space, we skip the results.

5 CONCLUSION

In this article, we have proposed the HHSVM for microarray classification and gene selection. The HHSVM uses the huberized hinge loss function to measure the 'badness-of-fit' and the elastic-net penalty to control the model complexity. The huberized hinge loss function allows efficient computation for calculating the entire solution path. The elastic-net penalty allows automatic flexible variable selection. We have presented some evidence that the new method tends to select more relevant variables (than the L_1-norm SVM method), especially when variables are highly correlated.

ACKNOWLEDGEMENTS

We thank the Associate Editor, David Rocke, and two referees for their feedback which led us to improve the article,

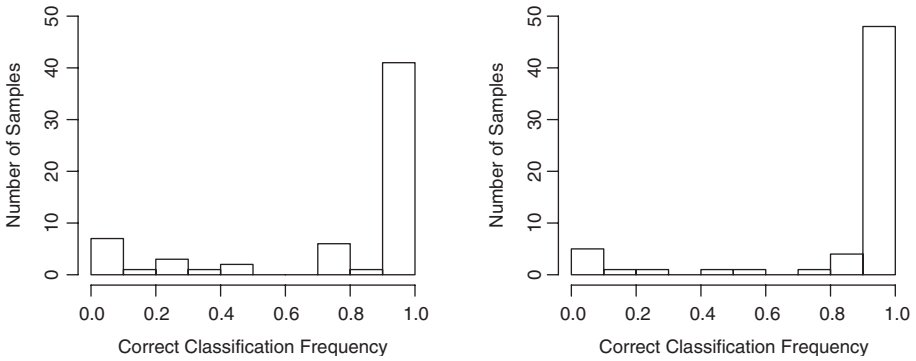

Fig. 3. Number of samples vs the frequency that a sample (as a test observation) was correctly classified on 100 random splits of the colon cancer dataset. The left panel is for the L_1-norm SVM, and the right panel is for the HHSVM.

particularly the numerical result section. We also thank Saharon Rosset for helpful comments. L.W. and J.Z. are partially supported by grant DMS-0505432 and DMS-0705532 from the National Science Foundation.

Conflict of Interest: none declared.

REFERENCES

Alon,U. and Barkai,N. (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon cancer tissues probed by oligonucleotide arrays. *Proc. Natl Acad. Sci. USA*, **96**, 6745–6750.

Bradley,P. and Mangasarian,O. (1998) Feature selection via concave minimization and support vector machines. In *Proceedings of the 15th International Conference on Machine Learning*.

Burges,C. (1998) A tutorial on support vector machines for pattern recognition. *Data Mining Knowl. Discov.*, **2**, 121–167.

Efron,B. *et al.* (2004) Least angle regression. *Ann. Stat.*, **32**, 407–499.

Evgeniou,T. *et al.* (1999) Regularization networks and support vector machines. In Smola,A. *et al.* (eds.) *Advances in Large Margin Classifiers*. MIT Press.

Guyon,I. *et al.* (2002) Gene selection for cancer classification using support vector machines. *Mach. Learn.*, **46**, 389–422.

Hoerl,A. and Kennard,R. (1970) Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, **12**, 55–67.

Mukherjee,S. *et al.* (2000) Support vector machine classification of microarray data. *Technical report*. Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Potti,A. *et al.* (2006) A genomic strategy to refine prognosis in early-stage non-small-cell lung cancer. *N. Eng. J. Med.*, **355**, 570–580.

Ramaswamy,S. *et al.* (2001) Multiclass cancer diagnosis using tumor gene expression signatures. *Proc. Natl Acad. Sci. USA*, **98**, 15149–15154.

Rosset,S. and Zhu,J. (2007) Piecewise linear regularized solution paths. *Ann. Stat.*, **35**, 1012–1030.

Tibshirani,R. (1996) Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. B*, **58**, 267–288.

Vapnik,V. (1995) *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Zhu,J. *et al.* (2004) 1-norm SVMs. In *Proceedings of the Neural Information Processing Systems*.

Zou,H. and Hastie,T. (2005) Regularization and variable selection via the elastic net. *J. R. Stat. Soc. B*, **67**, 301–320.