Hybrid huberized support vector machines for microarray classification and gene selection

Li Wang¹, Ji Zhu²,* and Hui Zou³

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

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

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 (HH SVM). The HH SVM combines the huberized hinge loss function and the elastic-net penalty. By doing so, the HH SVM performs automatic gene selection in a way similar to the $L_1$-norm SVM. In addition, the HH SVM encourages highly correlated genes to be selected (or removed) together. We also develop an efficient algorithm to compute the entire solution path of the HH SVM. Numerical results indicate that the HH SVM 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 the recursive 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.
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 \( \{x_1, x_2, \ldots, x_n\} \) 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:

\[
\max_{\beta, b} \frac{1}{\|\beta\|_2^2} \quad (1)
\]

subject to

\[
y_i(\beta_0 + x_i^T \beta) \geq 1 - \varepsilon_i \quad (2)
\]

\[
\sum_{i=1}^n \varepsilon_i \leq C \quad (3)
\]

\[
\varepsilon_i \geq 0, \text{ for } i = 1, 2, \ldots, n, \quad (4)
\]

where \( \varepsilon_i (i = 1, \ldots, n) \) are slack variables, and \( C \geq 0 \) is a tuning parameter that controls the overlap. Denote the solution as \( \beta \) and \( \hat{b} \), a new point with input vector \( x \) is then assigned with the label \( \text{sign}(\beta_0 + x^T \hat{\beta}) \).

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

\[
\min_{\beta, b} \sum_{i=1}^n [1 - y_i(\beta_0 + x_i^T \beta)]_+ + \frac{\lambda}{2} \|\beta\|_2^2, \quad (5)
\]

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

\[
\min_{\beta, b} \sum_{i=1}^n [1 - y_i(\beta_0 + x_i^T \beta)]_+ + \lambda \|\beta\|_1, \quad (6)
\]

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 \( \lambda_1 \) is large enough, it tends to shrink the coefficients of irrelevant variables to exactly zero. When \( \lambda \) 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:

\[
\lambda_1 \|\beta\|_1 + \frac{\lambda_2}{2} \|\beta\|_2^2.
\]

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:

\[
\min_{\beta, b} \sum_{i=1}^n \phi(y_i(\beta_0 + x_i^T \beta)) + \lambda_1 \|\beta\|_1 + \frac{\lambda_2}{2} \|\beta\|_2^2, \quad (7)
\]

where \( \lambda_1, \lambda_2 \geq 0 \) are regularization parameters. Increasing \( \lambda_1 \) tends to eliminate more irrelevant variables; and increasing \( \lambda_2 \) 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’:

\[
\phi(yf) = \begin{cases} 
0, & \text{for } \big|\frac{yf - yf}{\delta}\big| > 1, \\
(1 - \frac{yf - yf}{\delta})^2, & \text{for } 1 - \delta < \big|\frac{yf - yf}{\delta}\big| \leq 1, \\
1 - \big|\frac{yf - yf}{\delta}\big|, & \text{for } \big|\frac{yf - yf}{\delta}\big| \leq 1 - \delta,
\end{cases}
\]

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.
respect to a piecewise linear function (in a multi-dimensional space) with
the input vector \( x \). If the input vector \( x \) and \( x' \) are centered and normalized, then
the entire solution path is piecewise linear in \( \lambda_2 \).

3 ALGORITHM

The HHSVM involves two tuning parameters \( \lambda_1 \) and \( \lambda_2 \). According to our experience, the prediction performance is often more sensitive to \( \lambda_1 \), since it has more impact on selecting variables; therefore, the value of \( \lambda_1 \) 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 \( \lambda_1 \) (when \( \lambda_2 \) is fixed). The algorithm is based on the fact that the solution \( \hat{\beta} \) is a piecewise linear function (in a multi-dimensional space) with respect to \( \lambda_1 \).

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)
- \( \mathcal{E} = \{i : 1 - \delta < y_i f(x_i) \leq 1\} \) (Elbow)
- \( \mathcal{L} = \{i : y_i f(x_i) \leq 1 - \delta\} \) (Left)

where \( f(x_i) = \beta_0 + x_i^T \beta \). We also define the indices for non-zero \( \beta_j \) as the active set \( A \):

- \( A = \{j : \beta_j \neq 0, j = 1, 2, \ldots, p\} \) (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 \( \beta_0 \) to zero, we get:

\[
\sum_{i \in \mathcal{R}} \frac{1}{\delta} (\beta_0 + x_i^T \beta - y_i) - \sum_{i \in \mathcal{E}} y_i = 0.
\]

Setting the derivative with respect to \( \beta_j \) (\( j \in A \)) to zero, we get:

\[
\sum_{i \in \mathcal{R}} \frac{1}{\delta} (\beta_0 + x_i^T \beta - y_i)x_j - \sum_{i \in \mathcal{E}} y_i x_j + \lambda_2 \beta_j + \lambda_1 \text{sign}(\beta_j) = 0, \text{ for } j \in A.
\]

In the linear system (10)–(11), there are \( |A| + 1 \) unknowns and \( |A| + 1 \) equations, where \( |A| \) represents the number of elements in set \( A \). Therefore, the solution \( \beta_0 \) and \( \beta_j \) (\( j \in A \)) can be uniquely determined, given that the system is non-singular.

When sets \( \mathcal{E}, \mathcal{R} \) and \( A \) are fixed, the structure of the linear system (10)–(11) is also fixed. Under this condition, \( \beta_0 \) and \( \beta_j \) (\( j \in A \)) are linear functions of \( \lambda_1 \), which can be seen from (11). However, as \( \lambda_1 \) decreases, sooner or later some of the sets \( \mathcal{E}, \mathcal{R} \) and \( A \) will change. We call this an event. After an event occurs the new \( \beta_0 \) and \( \beta_j \) (\( j \in A \)) are still linear functions of \( \lambda_1 \), but their derivatives with respect to \( \lambda_1 \) will change. Therefore, the entire solution path is piecewise linear in \( \lambda_1 \), and between any two consecutive events, \( \beta_0 \) and \( \beta_j \) (\( j \in A \)) change linearly with \( \lambda_1 \). Each event corresponds to a kink on the piecewise linear solution path.

Our algorithm starts from \( \lambda_1 = \infty \), continuously decreases \( \lambda_1 \), solves the solutions along this path, and terminates if \( \lambda_1 \) reaches 0. The algorithm provides the solutions on each kink. For any \( \lambda_1 \) 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 \( \beta = 0 \) at this stage. Therefore, the problem reduces to:

\[
\min_{\beta_0} \sum_{i=1}^{n} \phi(y_i, \beta_0),
\]

which involves only one parameter \( \beta_0 \). Since the huberized hinge loss function is convex and differentiable everywhere, this one-variable optimization problem can be easily solved. In fact,
one can easily develop an analytical solution for the initial $\beta_0$, but due to the lack of space, we skip the details.

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

Since Equation (11) must hold for any $j \in A$, we can determine the critical point $\lambda_1^0$ by:

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

The corresponding variable that joins $A$ first can be identified as:

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

and the sign for $\beta_j$ is:

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

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

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 $\lambda_1^j$ and the corresponding $A_k$, $L$, $E$ and $R_k$ 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 $\lambda_1$ until it reaches 0. Let $\lambda_1 = \lambda_1^j + \Delta \lambda_1$, where $\Delta \lambda_1 < 0$. When $\lambda_1$ is reduced by a small enough amount, sets $L$, $E$, $R$ and $A$ do not change, because of the continuity of $f(x)$ with respect to $\lambda_1$. Therefore, based on the systems (10)-(11), the derivatives of $\beta_0$ and $\beta_j$ ($j \in A$) with respect to $\lambda_1$ can be solved from the following equations:

$$\sum_{i \in E} \frac{\Delta \beta_0}{\Delta \lambda_1} + \sum_{i \in A} x_{ij} \frac{\Delta \beta_j}{\Delta \lambda_1} = 0, \quad \text{(13)}$$

$$\sum_{i \in E} \frac{\Delta \beta_0}{\Delta \lambda_1} + \sum_{i \in A} x_{ij} \frac{\Delta \beta_j}{\Delta \lambda_1} x_{ij} + \lambda_1 \frac{\Delta \beta_0}{\Delta \lambda_1} + \text{sign}(\beta_j) = 0, \text{ for } j \in A. \quad \text{(14)}$$

Since there are $|A| + 1$ unknowns and $|A| + 1$ equations, $\frac{\Delta \beta_0}{\Delta \lambda_1}$ and $\frac{\Delta \beta_j}{\Delta \lambda_1}$ ($j \in A$) 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 $\lambda_1$:

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

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

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

If we keep reducing $\lambda_1$, some of the sets $L$, $E$, $R$ and $A$ will change. We call this an 'event', and four types of ‘events’ may occur:

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

The boundary between $L$ and $E$ is $1 - \delta$ and the boundary between $R$ and $E$ is 1. Therefore, when $y_i f(x)$ 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 $i \in L$ or $E$ we calculate:

$$\Delta \lambda_1^i = \frac{1 - \delta - y_i f(x)}{y_i \frac{\Delta \beta_0}{\Delta \lambda_1} + \sum_{j \in A} \frac{\Delta \beta_j}{\Delta \lambda_1} x_{ij}}.$$

Let $\Delta \lambda_{1,1}$ represent the step size for the first ‘event’. Since $\lambda_1$ only decreases, $\Delta \lambda_{1,1} \leq 0$ and its value should be determined by:

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

Similarly, for the second ‘event’ we calculate:

$$\Delta \lambda_1^i = \frac{1 - y_i f(x)}{y_i \frac{\Delta \beta_0}{\Delta \lambda_1} + \sum_{j \in A} \frac{\Delta \beta_j}{\Delta \lambda_1} x_{ij}},$$

for each $i \in R$ or $E$. And the step size for the second ‘event’ is:

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

When a non-zero $\beta_j$ reduces to 0, the third ‘event’ occurs. Therefore, we calculate for each $j \in A$:

$$\Delta \lambda_1^j = -\frac{\beta_j}{\Delta \lambda_1^j}.$$
The step size for the third ‘event’ is:
\[ \Delta \lambda_{1,3} = \max(\Delta \lambda^j_1 : j \in A, \Delta \lambda^j_1 \leq 0). \]

The fourth ‘event’ (a zero-valued \( \beta_j \) becomes non-zero) is a little complicated to determine. For \( j = 1, \ldots, p \), we define:
\[ C_j = \sum_{i \in L} \frac{1}{\delta} (\beta_0 + x_i^T \beta - y_i) x_i y_i - \sum_{i \in L} y_i x_i + \lambda_2 \beta_j, \]
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 \( j \in A \);
- \( |C_j| < \lambda_1 \), for \( j \notin A \).

Notice that when \( |\Delta \lambda_1| \) is sufficiently small, \( C_j \) is also a linear function of \( \lambda_1 \):
\[ C_j = C_j^0 + \left( \frac{1}{\delta} \sum_{i \in L} \frac{\Delta \beta_0}{\Delta \lambda_1} + \sum_{i \in E} \frac{\Delta \beta_i}{\Delta \lambda_1} \right) x_i (\lambda_1 - \lambda_1^0). \]

As \( \lambda_1 \) decreases, the value for a \( |C_j| \) will first meet the decreasing \( \lambda_1 \), after which the corresponding \( \beta_j \) will become non-zero if we further reduce \( \lambda_1 \).

Therefore, to determine the step size for the fourth ‘event’, for each \( j \notin A \) we calculate:
\[ \Delta \lambda^j_1 = \begin{cases} 0, & \text{if } C_j^0 + \frac{1}{\delta} \sum_{i \in L} \frac{\Delta \beta_0}{\Delta \lambda_1} + \sum_{i \in E} \frac{\Delta \beta_i}{\Delta \lambda_1} x_i \leq 0; \\ \frac{1}{\sum_{i \in E} \frac{\Delta \beta_i}{\Delta \lambda_1}} x_{i_0}, & \text{otherwise}. \end{cases} \]

The step size \( \Delta \lambda_{1,4} \) for the fourth ‘event’ is:
\[ \Delta \lambda_{1,4} = \max(\Delta \lambda^j_1 : j \notin A). \]

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 \( L, E, R \) and \( A \) according to the corresponding ‘event’, and also update \( \lambda_1, \beta_0, \beta_i, C_j \) and the fitted value \( f(x) \). 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 \( \lambda_1 \) reaches 0.

Between any two consecutive ‘events’, the solutions are linear in \( \lambda_1 \), and after an ‘event’ occurs, the derivative of the solution with respect to \( \lambda_1 \) is changed. Therefore, the solution path is piecewise linear in \( \lambda_1 \), where each ‘event’ corresponds to a kink on the path. The algorithm provides solutions at these kinks, and for any \( \lambda_1 \) between two consecutive kinks the solutions can be calculated precisely via linear interpolation.

### 3.4 Computation cost

The major computational cost in each iteration comes from solving the linear system (13)–(14). Since this system has \(|A| + 1\) unknowns, the computational cost seems to be \( O(|A|^3) \) for each iteration. However, between any two iterations, only one element is changed for sets \( L, E, R \) and \( A \), so using inverse updating and downdating the computational cost can be reduced to \( O(|A|^2) \). 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 \( R \) and \( O(p) \) steps to add all the parameters into \( A \). Since \( p \) is the upper bound for \(|A|\), the overall computational cost is upper bounded by \( O(\min(n,p)^2) \).

### 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
\[ \mu_+ = (0.5, \ldots, 0.5, 0, \ldots, 0)^T, \]
and covariance \( \Sigma = I_p \times p \). The ‘−’ class has a similar distribution except that
\[ \mu_- = (-0.5, \ldots, -0.5, 0, \ldots, 0)^T, \]

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 \( x_i \) 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>
<thead>
<tr>
<th>Scenario</th>
<th>L2-norm SVM</th>
<th>L1-norm SVM</th>
<th>HHSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario I</td>
<td>0.214 (0.004)</td>
<td>0.143 (0.007)</td>
<td>0.133 (0.005)</td>
</tr>
<tr>
<td>Scenario II</td>
<td>0.160 (0.003)</td>
<td>0.160 (0.002)</td>
<td>0.143 (0.001)</td>
</tr>
</tbody>
</table>

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.
HHSVM is slightly better than that of the 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 $q_{signal} =$ number of selected relevant variables, and $q_{noise} =$ 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

$$
\mu_+ = \left(1, \ldots, 1, 0, \ldots, 0\right)^T
$$

and covariance

$$
\Sigma = \begin{pmatrix}
\Sigma^*_1 & 0_{10 \times (p-10)} \\
0_{(p-10) \times 10} & I_{(p-10) \times (p-10)}
\end{pmatrix},
$$

where the diagonal elements of $\Sigma^*_1$ are 1 and the off-diagonal elements are all equal to $\rho = 0.8$. The ‘−’ class has a similar distribution except that

$$
\mu_- = \left(-1, \ldots, -1, 0, \ldots, 0\right)^T.
$$

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.

### 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% of the 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.

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

<table>
<thead>
<tr>
<th>Test error</th>
<th>Number of genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colon cancer dataset</td>
<td>Lung cancer dataset</td>
</tr>
<tr>
<td>SVM</td>
<td>14.65% (1.34%)</td>
</tr>
<tr>
<td>SVM-RFE</td>
<td>17.10% (0.87%)</td>
</tr>
<tr>
<td>$L_1$-norm SVM</td>
<td>15.84% (1.02%)</td>
</tr>
<tr>
<td>HHSVM</td>
<td>12.69% (0.93%)</td>
</tr>
<tr>
<td>SVM</td>
<td>29.0% (0.6%)</td>
</tr>
<tr>
<td>SVM-RFE</td>
<td>28.4% (0.7%)</td>
</tr>
<tr>
<td>$L_1$-norm SVM</td>
<td>29.3% (0.4%)</td>
</tr>
<tr>
<td>HHSVM</td>
<td>27.8% (0.7%)</td>
</tr>
</tbody>
</table>

The numbers in the parentheses are the corresponding standard errors.
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 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,
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


