Hybrid Huberized Support Vector Machines for Microarray Classification

Li Wang WANG@BUS.UMICH.EDU
Ross School of Business, University of Michigan, Ann Arbor, MI 48109, U.S.

Ji Zhu JIZHU@UMICH.EDU
Department of Statistics, University of Michigan, Ann Arbor, MI 48109, U.S.

Hui Zou HZOU@STAT.UMN.EDU
School of Statistics, University of Minnesota, Minneapolis, MN 55455, U.S.

Abstract

The large number of genes and the relatively small number of samples are typical characteristics for microarray data. These characteristics pose challenges for both sample classification and relevant gene selection. The support vector machine (SVM) is a widely used classification technique, and previous studies have demonstrated its superior classification performance in microarray analysis. However, a major limitation is that the SVM can not perform automatic gene selection. To overcome this limitation, we propose the hybrid huberized support vector machine (HHSVM). The HHSVM uses the huberized hinge loss function and the elastic-net penalty. It has two major benefits: 1. automatic gene selection; 2. the *grouping effect*, where highly correlated genes tend to be selected/removed together. We also develop an efficient algorithm that computes the entire regularized solution path for HHSVM. We have applied our method to real microarray data and achieved promising results.

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 data sets usually contain only a small number of samples. These characteristics pose great challenges for sample classification and gene selection. In the previous studies, many machine learning techniques have been applied to microarray analysis, for example, the penalized logistic regression, the boosting algorithm, the support vector machine (SVM), the nearest neighbor method. The SVM is one of the most effective methods for microarray classification (Mukherjee et al., 2000; Guyon et al., 2002; Ramaswamy et al., 2001); however, its major limitation is that the SVM can not perform automatic variable selection. Since in microarray analysis people are also interested in identifying the informative genes, it is desirable to have a tool that can achieve both classification and variable selection simultaneously.

Guyon et al. (2002) proposed the recursive feature elimination (RFE) method to overcome this limitation of the SVM. The method, called the SVM-RFE, recursively eliminates irrelevant genes. At each iteration, the SVM-RFE trains for an 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 a small number of variables are left in the model. However, the SVM-RFE is computationally intensive, especially for microarray data, which include a large number of variables. Bradley and Mangasarian (1998) proposed the L_1-norm SVM, which can perform variable selection using the L_1-norm regularization. The L_1-norm SVM, however, also has its limitations. Firstly, due to the L_1-norm penalty function, the number of selected variables is upper bounded by the sample size. Therefore, when the number of relevant variables exceeds the sample size, it can only discover a portion of them. Secondly, for the highly correlated and relevant variables, the L_1-norm SVM tends to pick only one or few of them.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

In this paper, we propose the hybrid huberized support vector machine (HHSVM) for microarray classification. The HHSVM has the form of "loss" + "penalty", where it uses the huberized hinge loss function to measure misclassification and the elastic-net penalty to control the complexity of the model. The HHSVM has two major benefits:

- It performs automatic variable selection;
- It has the *grouping effect* (Zou & Hastie, 2005), where highly correlated variables tend to be selected or removed together.

Inspired by the LAR/LASSO method (Efron et al., 2004) and the general piece-wise linear solution path strategy of Rosset and Zhu (2006) , we develop an efficient algorithm, which solves the optimal solutions for every possible value of the regularization parameter.

The rest of the paper 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 real microarray data sets; and we conclude the paper with Section 5.

2. Model

2.1. The Support Vector Machine

The support vector machine (SVM) is a widely used tool for 2-class classification and it is inspired by the idea of maximizing the geometric *margin* (Vapnik, 1995; Cortes & Vapnik, 1995). This section provides a brief review of the SVM and interested readers can refer to (Burges, 1998; Scholkopf et al., 1999) for detailed tutorials. Let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ represent the n input vectors, where $\boldsymbol{x}_i \in \mathbb{R}^p$, and let $\{y_1, y_2, ..., y_n\}$ be the corresponding output labels, where $y_i \in \{1, -1\}$. The SVM solves the following optimization problem:

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

$$\text{s.t.} \quad y_i(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}) \geq 1 - \varepsilon_i \tag{2}$$

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

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

where ε_i $(i = 1, ..., n)$ are slack variables. $C \geq 0$ is a tuning parameter, and it controls the overlap between the two classes of input data. After solving the optimal solution $\widehat{\beta}_0$ and $\widehat{\boldsymbol{\beta}}$, a point with input vector $\boldsymbol{x}$ is assigned with the label $sign(\widehat{\beta}_0 + \boldsymbol{x}^T \widehat{\boldsymbol{\beta}})$.

Many researchers have recognized that the SVM can be equivalently transformed into the "loss" + "penalty" format:

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

where the loss function $(1-t)_+ = \max(1-t, 0)$ is called the *hinge loss* and the "penalty" is the L_2-norm function 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 & 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 achieves the bias-variance trade-off, resulting in better prediction accuracy.

2.2. The L_1-norm SVM

The L_1-norm penalty was first used in the LASSO method (Tibshirani, 1996) for regression analysis. Inspired by this idea, Bradley and Mangasarian (1998) proposed the L_1-norm SVM. It replaces the L_2-norm penalty in the standard SVM with the L_1-norm penalty:

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

Similar to the L_2-norm penalty, the L_1-norm penalty can reduce the variance of the estimates and improve the prediction accuracy. Furthermore, it has a unique property: performing automatic variable selection. Because of the nature of the L_1-norm function, when λ_1 is large enough, it tends to reduce the coefficients of irrelevant variables to exactly zero. Therefore, when λ increases, more and more irrelevant variables are eliminated from the model, achieving automatic variable selection. This is desirable for microarray analysis, where scientists are often interested in identifying the relevant variables.

2.3. The Hybrid Huberized SVM

Zou and Hastie (2005) argue 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 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 of them if they are related to the underlying biological process.

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

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

which is a hybrid between the L_1-norm and the L_2-norm penalties. The elastic-net penalty retains the benefits of the L_1-norm penalty, but the number of variables that it can select is no longer bounded by n. Another nice property of the elastic-net penalty is that it tends to generate similar coefficients for the highly correlated variables (*grouping effect*); hence, highly correlated variables tend to be selected or removed together.

In this paper, we apply the elastic-net penalty to the SVM and propose the hybrid huberized support vector machine (HHSVM):

$$\min_{\beta_0,\boldsymbol{\beta}} \sum_{i=1}^{n} \ell_H\left(y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta})\right) + \lambda_1\|\boldsymbol{\beta}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\beta}\|_2^2, \quad (7)$$

where $\lambda_1, \lambda_2 \geq 0$ are regularization parameters. Increasing λ_1 tends to eliminate more irrelevant variables; and increasing λ_2 makes the *grouping effect* more prominent, which will be illustrated by the 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 & Zhu, 2006) to measure misclassification:

$$\ell_H(t) = \begin{cases} 0, & \text{for } t > 1; \\ \frac{(1-t)^2}{2\delta}, & \text{for } 1-\delta < t \leq 1; \\ 1 - t - \frac{\delta}{2}, & \text{for } t \leq 1-\delta, \end{cases}$$

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 a similar shape when yf is negative (misclassification): they both increase linearly as yf decreases. Therefore, the classification performance of these two functions should be similar to each other. Also notice that the huberized hinge loss function is differentiable everywhere, which is not the case for the hinge loss function. As shown in the next section, this differentiability can significantly reduce the computational cost for the HHSVM algorithm, especially for the initial setup.

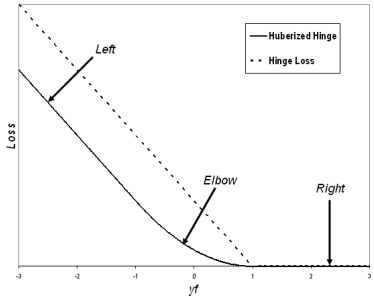

Figure 1. The Hinge and Huberized Hinge Loss Functions ($\delta = 2$). Note that the *Elbow* corresponds to the region $(1-\delta, 1]$; the *Left* and the *Right* correspond to the regions $(-\infty, 1-\delta]$ and $(1, \infty)$, respectively.

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 optimal solution for problem (7). For any pair (j, j'), we have*

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

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

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

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

Due to the lack of space, we skip the proof in this manuscript. 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, people usually specify a number of candidates, test each of them and choose 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 paper, we propose an efficient algorithm, which solves the optimal solutions for every possible value of λ_1. The algorithm is based on the fact that the solution $(\widehat{\beta}_0, \widehat{\boldsymbol{\beta}})$ is a piece-wise linear function (in a multi-dimensional space) of 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:

- $\mathcal{R} = \{i : y_i f(\boldsymbol{x}_i) > 1\}$ (Right),
- $\mathcal{E} = \{i : 1 - \delta < y_i f(\boldsymbol{x}_i) \leq 1\}$ (Elbow),
- $\mathcal{L} = \{i : y_i f(\boldsymbol{x}_i) \leq 1 - \delta\}$ (Left),

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

- $\mathcal{A} = \{j : \beta_j \neq 0,\ j = 1, 2, ..., p\}$ (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:

$$\sum_{i \in \mathcal{E}} \frac{1}{\delta}(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta} - y_i) - \sum_{i \in \mathcal{L}} y_i = 0 \tag{8}$$

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

$$\sum_{i \in \mathcal{E}} \frac{1}{\delta}(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta} - y_i) x_{ij} - \sum_{i \in \mathcal{L}} y_i x_{ij} + \lambda_2 \beta_j + \lambda_1 sign(\beta_j) = 0, \text{ for } j \in \mathcal{A} \tag{9}$$

In the linear system (8)-(9), there are $|\mathcal{A}|+1$ unknowns and $|\mathcal{A}|+1$ equations, where $|\mathcal{A}|$ represents the number of elements in set $\mathcal{A}$. So, the optimal solution $\widehat{\beta}_0$ and $\widehat{\boldsymbol{\beta}}_j$ ($j \in \mathcal{A}$) can be uniquely determined, given that the system is nonsingular.

When sets $\mathcal{L}, \mathcal{E}, \mathcal{R}$ and $\mathcal{A}$ are fixed, the structure for equation (8)-(9) is also fixed. Under this condition, $\widehat{\beta}_0$ and $\widehat{\boldsymbol{\beta}}_j$ ($j \in \mathcal{A}$) are linear functions of λ_1, which can be seen from (9). However, as λ_1 keeps decreasing, 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 $\widehat{\beta}_0$ and $\widehat{\boldsymbol{\beta}}_j$ ($j \in \mathcal{A}$) are still linear functions of λ_1, but their derivatives with respect to λ_1 will change. Therefore, the entire optimal solution path is piece-wise linear in λ_1, and between any two consecutive events, $\widehat{\beta}_0$ and $\widehat{\boldsymbol{\beta}}_j$ ($j \in \mathcal{A}$) change linearly with λ_1. Each *event* corresponds to a kink on the piece-wise linear solution path.

Our algorithm starts from $\lambda_1 \to \infty$; continuously decreases λ_1; solves the optimal solutions along this path; and terminates if λ_1 reaches 0 or some other stopping criterion is met. The algorithm provides the optimal 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 HHSVM algorithm.

Table 1. Outline of the HHSVM Algorithm

INITIALIZATION:	`Calculate` β_0^0, β_j^0 $(j = 1, ..., p)$, 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 system` (11)-(12);
STEP 2:	`Calculate` $\Delta\lambda_1$ `and determine the next event according to Section 3.3;`
STEP 3:	`If the stopping criterion is met, then stop the algorithm;`
STEP 4:	`Otherwise, let` $k = k + 1$ `and update` β_0^k, β_j^k $(j = 1, ..., p)$, 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;`

3.2. Initial Solution

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

$$\min_{\beta_0} \sum_{i=1}^{n} \ell_H (y_i \beta_0), \tag{10}$$

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. Let $\widehat{\beta}_0$ represent the optimal solution. In fact, we can develop the analytical solution for $\widehat{\beta}_0$, but due to the lack of space, we skip the details.

When $\lambda_1 \to \infty$, $\mathcal{A} = \Phi$ ($\boldsymbol{\beta} = 0$) and sets $\mathcal{L}, \mathcal{E}$ and $\mathcal{R}$ can be determined using the values of $y_i \widehat{\beta}_0$ $(i = 1, ..., n)$. Reducing λ_1 tends to increase the magnitude of $\boldsymbol{\beta}$, and we can find a critical point, denoted as λ_1^*, where exactly one β_j $(j = 1, ..., p)$ joins $\mathcal{A}$. The parameter will become non-zero if λ_1 is further reduced.

Since equation (9) must hold for any $j \in \mathcal{A}$, we can determine the critical point λ_1^* by:

$$\lambda_1^* = \max_{j \in \{1,..,p\}} \left(|\sum_{i \in \mathcal{E}} \frac{1}{\delta}(\widehat{\beta}_0 - y_i) x_{ij} - \sum_{i \in \mathcal{L}} y_i x_{ij}| \right)$$

At the critical point, $\mathcal{A}$ contains exactly one element ($\mathcal{A} = \{j^*\}$), and this element j^* can be identified as:

$$j^* = \text{argmax}_{j \in \{1,..,p\}} \left(|\sum_{i \in \mathcal{E}} \frac{1}{\delta}(\widehat{\beta}_0 - y_i) x_{ij} - \sum_{i \in \mathcal{L}} y_i x_{ij}| \right)$$

From equation (9), the sign for β_{j^*} is:

$$sign(\beta_{j^*}) = sign\left(-\sum_{i\in\mathcal{E}}\frac{1}{\delta}(\widehat{\beta}_0 - y_i)x_{ij^*} + \sum_{i\in\mathcal{L}} y_i x_{ij^*}\right)$$

Let the superscript k indicate the iteration number and assume $k = 0$ at the initial stage. Now, we have $\mathcal{A}^k = \{j^*\}$, $\beta_0^k = \widehat{\beta}_0$, $\beta_j^k = 0$ $(j = 1, ..., p)$, $\lambda_1^k = \lambda_1^*$, and $\mathcal{L}^k, \mathcal{R}^k, \mathcal{E}^k$ are determined by $y_i\widehat{\beta}_0$ $(i = 1, ..., n)$.

We note that the initial conditions can be easily solved 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 differentiable everywhere, it is difficult to determine the critical point λ_1^* and the corresponding $\mathcal{A}, \mathcal{L}, \mathcal{E}$ and $\mathcal{R}$ at the initial stage. Zhu et al. (2004) proposed an approach for solving the initial solution for the L_1-norm SVM. The approach was based on linear programming, and it is 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 step, sets $\mathcal{L}, \mathcal{E}, \mathcal{R}$ and $\mathcal{A}$ do not change, because of their continuity with respect to λ_1. Therefore, based on system (8)-(9), the derivatives of β_0 and β_j $(j \in \mathcal{A})$ with respect to λ_1 can be solved from the following equations:

$$\sum_{i\in\mathcal{E}}\left(\frac{\Delta\beta_0}{\Delta\lambda_1} + \sum_{k\in\mathcal{A}} x_{ik}\frac{\Delta\beta_k}{\Delta\lambda_1}\right) = 0 \tag{11}$$

$$\sum_{i\in\mathcal{E}}\frac{1}{\delta}\left(\frac{\Delta\beta_0}{\Delta\lambda_1} + \sum_{k\in\mathcal{A}} x_{ik}\frac{\Delta\beta_k}{\Delta\lambda_1}\right)x_{ij} + \lambda_2\frac{\Delta\beta_j}{\Delta\lambda_1} + sign(\beta_j) = 0, \text{ for } j \in \mathcal{A} \tag{12}$$

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 nonsingular. When $|\Delta\lambda_1|$ is sufficiently small, the optimal solution and fitted values are linear in λ_1:

$$\begin{aligned}
\beta_0 &= \beta_0^k + \frac{\Delta\beta_0}{\Delta\lambda_1}(\lambda_1 - \lambda_1^k)\\
\beta_j &= \beta_j^k + \frac{\Delta\beta_j}{\Delta\lambda_1}(\lambda_1 - \lambda_1^k), \qquad \text{for } j \in \mathcal{A}\\
f(\boldsymbol{x}_i) &= f^k(\boldsymbol{x}_i) + \left(\frac{\Delta\beta_0}{\Delta\lambda_1} + \sum_{j\in\mathcal{A}}\frac{\Delta\beta_j}{\Delta\lambda_1}\right)(\lambda_1 - \lambda_1^k)
\end{aligned}$$

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 $y_i f(\boldsymbol{x}_i)$ 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:

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

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:

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

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. First, 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 for equation (9). From (9) and the initial conditions, we know that:

- $|C_j| = \lambda_1,\ sign(C_j) = -sign(\beta_j),$ 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:

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

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:

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

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:

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

and we update $\mathcal{L}, \mathcal{E}, \mathcal{R}$ and $\mathcal{A}$ according to the corresponding *event*. Variable $\lambda_1, \beta_0, \beta_j$, C_j and the fitted value $f(\boldsymbol{x}_i)$ should also be updated. Finally, let $k = k + 1$ and the algorithm goes to the next iteration: solving linear system (11)-(12) and calculating the step size $\Delta\lambda_1$. This entire process is repeated, until λ_1 reaches 0.

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

3.4. Computational Cost

The major computational cost for each iteration is for solving linear system (11)-(12). Since in this system there are $|\mathcal{A}| + 1$ unknowns, the computational cost seems to be $O(|\mathcal{A}|^3)$ for each iteration. However, between any two steps 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 $O(\min(n, p)p^2)$. However, the two classes usually can be perfectly separated when there are more active variables than the sample size ($|\mathcal{A}| > n$), and the algorithm should terminate when the two classes are already perfectly separated. Therefore, when $p \gg n$, $O(\min(n, p)p^2)$ is a loose upper bound for the computational cost of our algorithm.

4. Experiment

In this section, we apply the HHSVM method on two real microarray data sets, and we compare its performance with the standard SVM, SVM-RFE and L_1-norm SVM.

4.1. Leukemia Data

The SVM, SVM-RFE, L_1-norm SVM and HHSVM are applied to a leukemia data set (Golub et al., 1999). The data set includes two types of acute leukemia, acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). It contains 38 samples for training and 34 samples for testing, and each sample includes 2,185 genes. We apply the three methods to the training data, use 10-fold cross-validation for parameter tuning and evaluate their performance on the testing data. Table 2 shows our results. It can be seen that the HHSVM has better prediction performance than the other methods. For the SVM, we tried different kernel methods and the best performance is achieved on the linear kernel. For the SVM-RFE, we use the same approach as Guyon et al. (2002), eliminating half of the remaining genes at each iteration. It is also worth mentioning that all the 22 genes selected by the L_1-norm SVM are also selected by the HHSVM.

The previous results are based on the original separation for the training and testing data. We also test the performance of the four methods using a randomly-splitting approach. The original training and testing data are combined together and we randomly split it into 38 and 34 samples for training and testing, respectively. Following the same procedure as before, we train each method on the training data and test

Table 2. Results on the Leukemia Data: based on the original separation of the training and testing data

	CV Error	Testing Error	# of Genes
Golub	3/38	4/34	50
SVM	0/38	1/34	All
SVM-RFE	0/38	2/34	128
L_1-norm SVM	3/38	1/34	22
HHSVM	0/38	0/34	84

Table 3. Results on the Leukemia Data: based on the randomly-splitting approach; numbers in parentheses are the corresponding standard errors

	Ave. Testing Error	Ave. # of Genes
SVM	2.33% (0.39%)	All
SVM-RFE	2.25% (0.46%)	256
L_1-norm SVM	8.22% (0.48%)	20.3 (0.4)
HHSVM	1.67% (0.24%)	87.9 (5.5)

them on the testing data. The entire process is repeated 50 times and the results are summarized in Table 3. Table 4 lists the top 10 genes that have been frequently selected by the HHSVM. The SVM, SVM-RFE and HHSVM still work well, where the HHSVM has the best prediction error. However, the randomly-splitting result for the L_1-norm SVM is not as good as before. It is likely that some highly correlated and relevant genes can not be captured by the L_1-norm SVM in the randomly-splitting experiment.

4.2. Breast Cancer Data

The breast cancer data (West et al., 2001) consist of 49 tumor samples, among which 25 are estrogen receptor (ER) positive and 24 are ER negative. Each sample consists of 7,129 genes. Similar to what we did to the leukemia data, we use the randomly-splitting procedure to evaluate the four methods. At each iteration 31 samples (16 positive and 15 negative) are used for training and the rest 18 samples (9 positive and 9 negative) for testing. Table 5 shows the testing results, and the HHSVM has the best prediction performance. Table 6 lists the top 10 frequently selected genes by the HHSVM. Shevade and Keerthi (2003) identified 6 relevant genes from the same data set, where *Y box binding protein-1 (YB-1) mRNA* and *H.sapiens mRNA for cathepsin C* are also in our top 10 list.

Table 4. Some Genes Selected by the HHSVM on the Leukemia Data: based on the randomly-splitting approach

Gene annotation	Selected frequency
Lysosomal Mal Pro-X Carboxypeptidase Precursor	50/50
ARH9 Aplysia ras-related homolog 9	50/50
RAS-related Protein RAB-1A	48/50
Mitochondrial 1,25-dihydroxyvitamin D3 24-hydroxylase mRNA	47/50
GUSB Glucuronidase, beta	47/50
NCA Non-specific cross reacting antigen	46/50
GIP Gastric inhibitory polypeptide	45/50
Rhodanese	44/50
Calcitonin	44/50
Alpha-2,8-polysialyltransferase (PST) gene	44/50

Table 5. Results on the Breast Cancer Data: based on the randomly-splitting approach; numbers in parentheses are the corresponding standard errors

	Ave. Testing Error	Ave. # of Genes
SVM	9.56% (0.54%)	All
SVM-RFE	10.11% (0.80%)	128
L_1-norm SVM	18.93% (0.88%)	12.1 (0.4)
HHSVM	8.67% (0.87%)	199.0 (6.4)

5. Conclusion

In microarray analysis, the large number of genes and the relatively small number of samples pose great challenges for tissue classification and gene selection. To tackle these problems, we propose the hybrid huberized support vector machine (HHSVM). The HHSVM uses the huberized hinge loss function to measure misclassification and the elastic-net penalty to control the complexity of the model. The elastic-net penalty helps achieve the automatic variable selection and it provides the *grouping effect*. The huberized hinge loss function enables us to develop an efficient algorithm, which solves the entire regularized solution path. We compare the HHSVM with other methods on two microarray data sets and the HHSVM achieves promising results on both classification and gene selection.

Table 6. Some Genes Selected by the HHSVM on the Breast Cancer Data: based on the randomly-splitting approach

Gene annotation	Selected frequency
Y box binding protein-1 (YB-1) mRNA	50/50
Human high mobility group protein (HMG-I(Y)) gene exons 1-8	50/50
Homo sapiens (clone zap128) mRNA	50/50
Human hGATA3 mRNA for trans-acting T-cell specific transcription factor	50/50
H.sapiens mRNA for cathepsin C	50/50
Human DR-nm23 mRNA	49/50
Human mRNA for oestrogen receptor	49/50
Human androgen receptor mRNA	47/50
Human X box binding protein-1 (XBP-1) mRNA	47/50
Human insulin-like growth factor binding protein 4 (IGFBP4) mRNA	47/50

References

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

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

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*.

Efron, B., Johnstone, I., Hastie, T., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, *32*, 407–499.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., & Lander, E. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, *286*, 531–536.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*, 389–422.

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

Mukherjee, S., Tamayo, P., Slonim, D., Verri, A., Golub, T., Mesirov, J., & Poggio, T. (2000). *Support vector machine classification of microarray data* (Technical Report). Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J., Poggio, T., Gerald, W., Loda, M., Lander, E., & Golub, T. (2001). Multiclass cancer diagnosis using tumor gene expression signature. *Proceedings of National Academy of Sciences of the United States of America*.

Rosset, S., & Zhu, J. (2006). Piecewise linear regularized solution paths. *The Annals of Statistics*. In press.

Scholkopf, B., Burges, C., & Smola, A. (Eds.). (1999). *Advances in kernel methods: Support vector learning*. Cambridge: MIT Press.

Shevade, S., & Keerthi, S. (2003). A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, *19*, 2246–2253.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, *58*, 267–288.

Vapnik, V. (Ed.). (1995). *The nature of statistical learning theory*. New York: Springer-Verlag.

West, M., Blanchette, C., Dressman, H., Huang, E., Ishida, S., Spang, R., Zuzan, H., Olson, J., Marks, J., & Nevins, J. (2001). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of National Academy of Sciences of the United States of America* (pp. 11462–11467).

Zhu, J., Rosset, S., Hastie, T., & Tibshirani, R. (2004). 1-norm svms. *Proceedings of the Neural Information Processing Systems*.

Zou, H., & Hastie, T. (2005). Regualization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, *67*, 301–320.