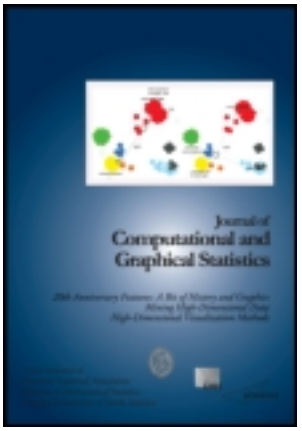

Journal of Computational and Graphical Statistics

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ucgs20

# An Efficient Algorithm for Computing the HHSVM and Its Generalizations

Yi Yang [a] & Hui Zou [a]

[a] University of Minnesota , Minneapolis , MN , 55455-0213

Accepted author version posted online: 25 Apr 2012.Published online: 30 May 2013.

To cite this article: Yi Yang & Hui Zou (2013) An Efficient Algorithm for Computing the HHSVM and Its Generalizations, Journal of Computational and Graphical Statistics, 22:2, 396-415, DOI: 10.1080/10618600.2012.680324

To link to this article: http://dx.doi.org/10.1080/10618600.2012.680324

Supplementary materials for this article are available online.
Please go to www.tandfonline.com/r/JCGS

# An Efficient Algorithm for Computing the HHSVM and Its Generalizations

Yi YANG and Hui ZOU

The hybrid Huberized support vector machine (HHSVM) has proved its advantages over the $\ell_1$ support vector machine (SVM) in terms of classification and variable selection. Similar to the $\ell_1$ SVM, the HHSVM enjoys a piecewise linear path property and can be computed by a least-angle regression (LARS)-type piecewise linear solution path algorithm. In this article, we propose a generalized coordinate descent (GCD) algorithm for computing the solution path of the HHSVM. The GCD algorithm takes advantage of a majorization–minimization trick to make each coordinatewise update simple and efficient. Extensive numerical experiments show that the GCD algorithm is much faster than the LARS-type path algorithm. We further extend the GCD algorithm to solve a class of elastic net penalized large margin classifiers, demonstrating the generality of the GCD algorithm. We have implemented the GCD algorithm in a publicly available R package gcdnet.

**Key Words:** Coordinate descent; Elastic net; Hubernet; Large margin classifiers; Majorization–minimization; SVM.



## 1. INTRODUCTION

The support vector machine (SVM; Vapnik 1995) is a very popular large margin classifier. Despite its competitive performance in terms of classification accuracy, a major limitation of the SVM is that it cannot automatically select relevant variables for classification, which is very important in high-dimensional classification problems such as tumor classification with microarrays. The SVM has an equivalent formulation as the $\ell_2$ penalized hinge loss (Hastie, Tibshirani, and Friedman 2001). Bradley and Mangasarian (1998) proposed the $\ell_1$ SVM by replacing the $\ell_2$ penalty in the SVM with the $\ell_1$ penalty. The $\ell_1$ penalization (aka lasso) (Tibshirani 1996) is a very popular technique for achieving sparsity with high-dimensional data. There has been a large body of theoretical work to support the $\ell_1$ regularization. A comprehensive reference is Bühlmann and van de Geer (2011). Wang, Zhu, and Zou (2008) later proposed a hybrid Huberized support vector machine (HHSVM) that uses the elastic

Yi Yang (E-mail: *yiyang@umn.edu*) is Ph.D. student and Hui Zou (E-mail: *zouxx019@umn.edu*) is Associate Professor in School of Statistics at University of Minnesota, Minneapolis, MN 55455-0213.


*Journal of Computational and Graphical Statistics*, Volume 22, Number 2, Pages 396–415
*DOI: 10.1080/10618600.2012.680324*

net penalty (Zou and Hastie 2005) for regularization and variable selection and uses the Huberized squared hinge loss for efficient computation. Wang, Zhu, and Zou (2008) showed that the HHSVM outperforms the standard SVM and the $\ell_1$ SVM for high-dimensional classification. Wang, Zhu, and Zou (2008) extended the least-angle regression (LARS) algorithm for the lasso regression model (Osborne, Presnell, and Turlach 2000; Efron et al. 2004, Rosset and Zhu 2007) to compute the solution paths of the HHSVM.

The main purpose of this article is to introduce a new efficient algorithm for computing the HHSVM. Our study was motivated by the recent success of using coordinate descent algorithms for computing the elastic net penalized regression and logistic regression (Friedman, Hastie, and Tibshirani 2010; see also Van der Kooij 2007). For the lasso regression, the coordinate descent algorithm amounts to an iterative cyclic soft-thresholding operation. Despite its simplicity, the coordinate descent algorithm can even outperform the LARS algorithm, especially when the dimension is much larger than the sample size (see table 1 of Friedman, Hastie, and Tibshirani 2010). Other articles on coordinate descent algorithms for the lasso include Fu (1998), Daubechies, Defrise, and De Mol (2004), Friedman et al. (2007), Genkin, Lewis, and Madigan (2007), Wu and Lange (2008), and among others. The HHSVM poses a major challenge for applying the coordinate descent algorithm, because the Huberized hinge loss function does not have a smooth first derivative everywhere. As a result, the coordinate descent algorithm for the elastic net penalized logistic regression (Friedman, Hastie, and Tibshirani 2010) cannot be used for solving the HHSVM.

To overcome the computational difficulty, we propose a new generalized coordinate descent (GCD) algorithm for solving the solution paths of the HHSVM. We also give a further extension of the GCD algorithm to general problems. Our algorithm adopts the majorization–minimization (MM) principle into the coordinate descent loop. We use an MM trick to make each coordinatewise update simple and efficient. In addition, the MM principle ensures the descent property of the GCD algorithm which is crucial for all coordinate descent algorithms. Extensive numerical examples show that the GCD can be much faster than the LARS-type path algorithm in Wang, Zhu, and Zou (2008). Here we use the prostate cancer data (Singh et al. 2002) to briefly illustrate the speed advantage of our algorithm (see Figure 1). The prostate data have 102 observations and each has 6033 gene expression values. It took the LARS-type path algorithm about 5 min to compute the HHSVM paths, while the GCD used only 3.5 sec to get the identical solution paths.

A closer examination of the GCD algorithm reveals that it can be used for solving other large margin classifiers. We have derived a generic GCD algorithm for solving a class of elastic net penalized large margin classifiers. We further demonstrate the generic GCD algorithm by considering the squared SVM loss and the logistic regression loss. We study through numeric examples how the shape and smoothness of the loss function affect the computational speed of the generic GCD algorithm.

In Section 2, we review the HHSVM and then introduce the GCD algorithm for computing the HHSVM. Section 3 studies the descent property of the GCD algorithm by making an intimate connection to the MM principle. The analysis motivates us to further consider a generic GCD algorithm for handling a class of elastic net penalized large margin classifiers. Numerical experiments are presented in Section 5.

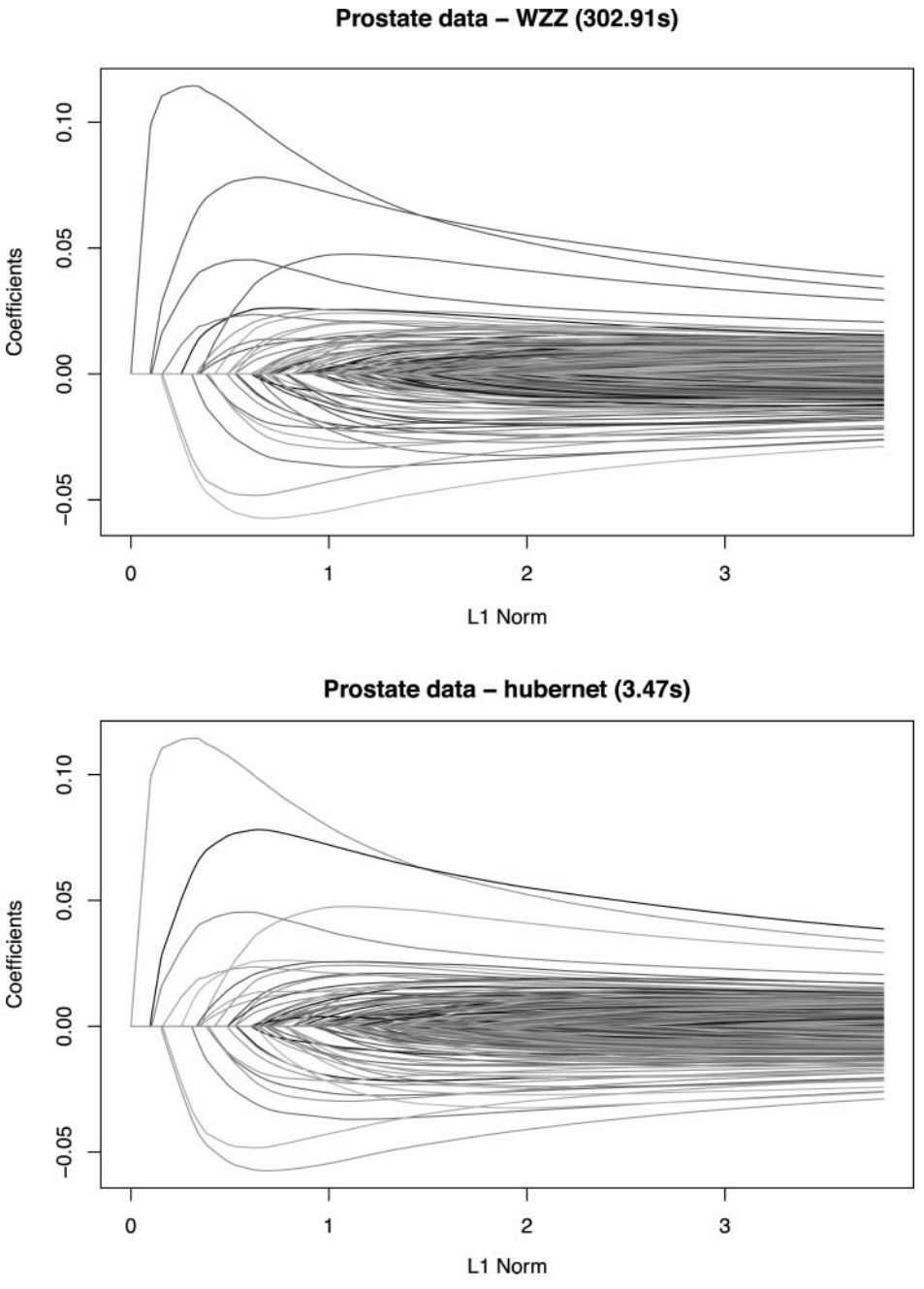


Figure 1. Solution paths and timings of the HHSVM on the prostate cancer data with 102 observations and 6033 predictors. The top panel shows the solution paths computed by the LARS-type algorithm in Wang, Zhu, and Zou (2008); the bottom panel shows the solution paths computed by GCD. GCD is 87 times faster.

## 2. THE HHSVM AND GCD ALGORITHM

### 2.1 The HHSVM

In a standard binary classification problem, we are given $n$ pairs of training data $\{\mathbf{x}_i, y_i\}$ for $i = 1, \ldots, n$ where $\mathbf{x}_i \in \mathbb{R}^p$ are predictors and $y_i \in \{-1, 1\}$ denotes class labels. The linear SVM (Vapnik 1995; Burges 1998; Evgeniou, Pontil, and Poggio 2000) looks for the hyperplane with the largest margin that separates the input data for class 1 and class $-1$

$$\min_{(\beta_0,\boldsymbol{\beta})} \frac{1}{2}||\boldsymbol{\beta}||^2 + \gamma \sum_{i=1}^{n} \xi_i$$
$$\text{subject to } \xi_i \geq 0,\ y_i\left(\beta_0 + \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta}\right) \geq 1 - \xi_i\ \forall i, \tag{2.1}$$

where $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$ are the slack variables and $\gamma > 0$ is a constant. Let $\lambda = 1/(n\gamma)$. Then (2.1) has an equivalent $\ell_2$ penalized hinge loss formulation (Hastie, Tibshirani, and Friedman 2001)

$$\min_{(\beta_0,\boldsymbol{\beta})} \frac{1}{n}\sum_{i=1}^{n}\left[1 - y_i\left(\beta_0 + \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta}\right)\right]_+ + \frac{\lambda}{2}||\boldsymbol{\beta}||_2^2. \tag{2.2}$$

The loss function $L(t) = [1 - t]_+$ has the expression

$$[1-t]_+ = \begin{cases} 0, & t > 1 \\ 1 - t, & t \leq 1, \end{cases}$$

which is called the hinge loss in the literature. The SVM has very competitive performance in terms of classification. However, because of the $\ell_2$ penalty the SVM uses all variables in classification, which could be a great disadvantage in high-dimensional classification (Zhu et al. 2004). The $\ell_1$ SVM proposed by Bradley and Mangasarian (1998) is defined by

$$\min_{(\beta_0,\boldsymbol{\beta})} \frac{1}{n}\sum_{i=1}^{n}\left[1 - y_i\left(\beta_0 + \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta}\right)\right]_+ + \frac{\lambda}{2}||\boldsymbol{\beta}||_1. \tag{2.3}$$

Just like in the lasso regression model, the $\ell_1$ penalty produces a sparse $\boldsymbol{\beta}$ in (2.3). Thus the $\ell_1$ SVM is able to automatically discard irrelevant features. In the presence of many noise variables, the $\ell_1$ SVM has significant advantages over the standard SVM (Zhu et al. 2004).

The elastic net penalty (Zou and Hastie 2005) is an important generalization of the lasso penalty. The elastic net penalty is defined as

$$P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}) = \sum_{j=1}^{p} p_{\lambda_1,\lambda_2}(\beta_j) = \sum_{j=1}^{p}\left(\lambda_1|\beta_j| + \frac{\lambda_2}{2}\beta_j^2\right), \tag{2.4}$$

where $\lambda_1, \lambda_2 \geq 0$ are regularization parameters. The $\ell_1$ part of the elastic net is responsible for variable selection. The $\ell_2$ part of the elastic net helps handle strong correlated variables, which is common in high-dimensional data, and improves prediction.

Wang, Zhu, and Zou (2008) used the elastic net in the SVM classification:

$$\min_{(\beta_0,\boldsymbol{\beta})} \frac{1}{n}\sum_{i=1}^{n}\phi_c\left(y_i\left(\beta_0 + \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta}\right)\right) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}). \tag{2.5}$$

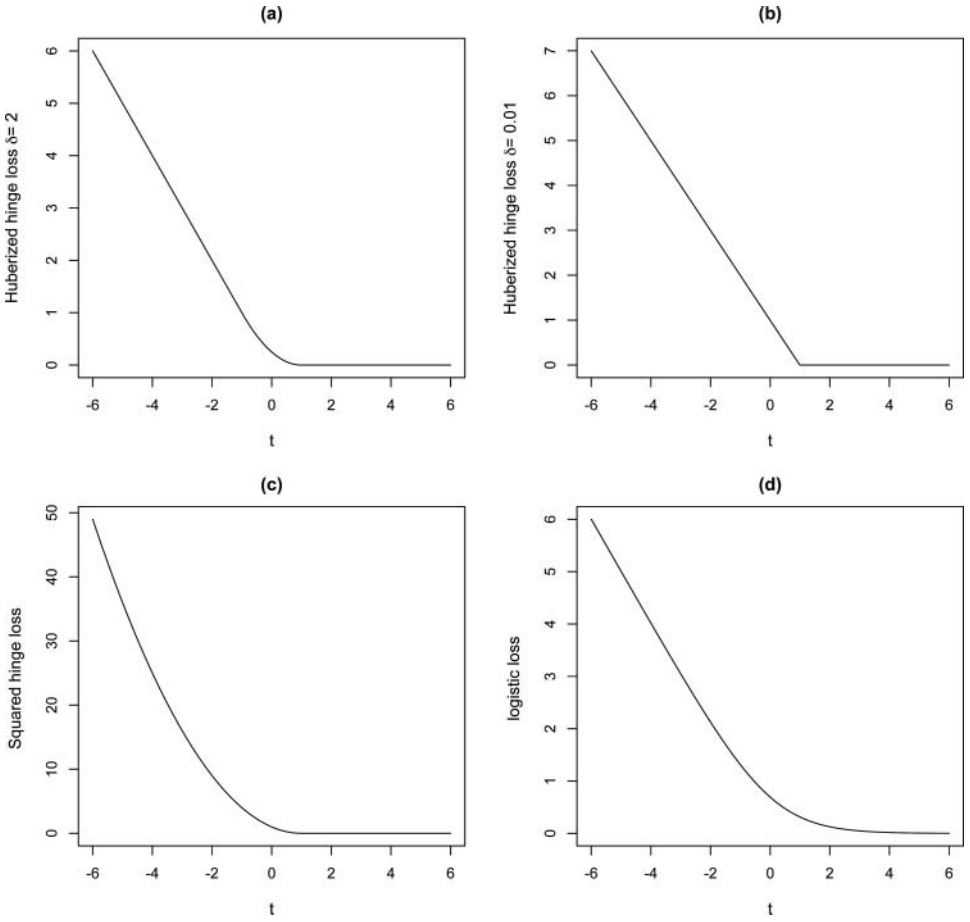


Figure 2. (a) The Huberized hinge loss function (with $\delta = 2$); (b) the Huberized hinge loss function (with $\delta = 0.01$); (c) the squared hinge loss function; (d) the logistic loss function.

Note that $\phi_c(\cdot)$ in (2.5) is the Huberized hinge loss

$$\phi_c(t) = \begin{cases} 0, & t > 1 \\ (1-t)^2/2\delta, & 1-\delta < t \le 1 \\ 1-t-\delta/2, & t \le 1-\delta, \end{cases}$$

where $\delta > 0$ is a pre-specified constant. The default choice for $\delta$ is 2 unless specified otherwise. Displayed in Figure 2 panel (a) is the Huberized hinge loss with $\delta = 2$. The Huberized hinge loss is very similar to the hinge loss in shape. In fact, when $\delta$ is small, the two loss functions are almost identical. See Figure 2 panel (b) for a graphical illustration. Unlike the hinge loss, the Huberized hinge loss function is differentiable everywhere and has continuous first derivative. Wang, Zhu, and Zou (2008) derived a LARS-type path algorithm for computing the solution paths of the HHSVM. Compared with the LARS-type algorithm for the $\ell_1$ SVM (Zhu et al. 2004), the LARS-type algorithm for the HHSVM has significantly lower computational cost, thanks to the differentiability property of the Huberized hinge loss.

### 2.2 A Generalized Coordinate Descent Algorithm

Besides the LARS-type algorithm, coordinate descent algorithms have been successfully used to compute the elastic net penalized linear model and generalized linear models. See the R package glmnet by Friedman, Hastie, and Tibshirani (2010). Despite its simplicity, the coordinate descent can even outperform the more sophisticated LARS algorithm (Friedman, Hastie, and Tibshirani 2010). Motivated by the great success of glmnet, we consider developing a fast coordinate descent algorithm for computing the HHSVM, with the goal to outperform the LARS-type algorithm derived in Wang, Zhu, and Zou (2008).

Without loss of generality assume that the input data are standardized: $\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0$, $\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 = 1$, for $j = 1, \ldots, p$. For the HHSVM, we can write down the standard coordinate descent algorithm as follows. Define the current margin $r_i = y_i(\tilde{\beta}_0 + \mathbf{x}_i^{\mathsf{T}}\tilde{\boldsymbol{\beta}})$ and

$$F(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) = \frac{1}{n}\sum_{i=1}^{n} \phi_c(r_i + y_i x_{ij}(\beta_j - \tilde{\beta}_j)) + p_{\lambda_1,\lambda_2}(\beta_j). \tag{2.6}$$

For fixed $\lambda_1$ and $\lambda_2$, the standard coordinate descent algorithm (Tseng 2001) proceeds as follows:

1. Initialization: $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$
2. Cyclic coordinate descent: for $j = 0, 1, 2, \ldots, p$: update $\tilde{\beta}_j$ by minimizing the objective function

$$\tilde{\beta}_j = \arg\min_{\beta_j} F(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}). \tag{2.7}$$

3. Repeat Step 2 till convergence.

The major difficulty in using the above coordinate descent procedure is that the univariate minimization problem in (2.7) does not have a closed form solution, unlike the penalized least squares. The univariate $\ell_1$ penalized least squares has a neat solution by soft-thresholding. However, solving (2.7) requires an iterative algorithm. The same problem occurs when computing the elastic net penalized logistic regression. In glmnet, Friedman, Hastie, and Tibshirani (2010) handled this difficulty by using the Newton–Raphson idea on the top of the coordinate descent. After computing the partial derivatives and the Hessian matrix of the logistic regression loss, we face an elastic net penalized weighted least squares that can be easily solved by invoking an iterative coordinatewise soft-thresholding procedure (Friedman, Hastie, and Tibshirani 2010). However, the Huberized hinge loss does not even have the second derivative, so the idea in glmnet is not directly applicable in the HHSVM.

We show that the computational obstacle can be resolved by a neat trick. We approximate the $F$ function in (2.6) by a penalized quadratic function defined as

$$\begin{aligned} Q(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) = {} & \frac{\sum_{i=1}^{n} \phi_c(r_i)}{n} + \frac{\sum_{i=1}^{n} \phi_c'(r_i) y_i x_{ij}}{n}(\beta_j - \tilde{\beta}_j) \\ & + \frac{1}{\delta}(\beta_j - \tilde{\beta}_j)^2 + p_{\lambda_1,\lambda_2}(\beta_j), \end{aligned} \tag{2.8}$$

where $\phi_c'(t)$ is the first derivative of $\phi_c(t)$. We can easily solve the minimizer of (2.8) by a simple soft-thresholding rule (Zou and Hastie 2005):

$$\hat{\beta}_j^{\mathrm{C}} = \arg\min_{\beta_j} Q(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) = \frac{S\left(\frac{2}{\delta}\tilde{\beta}_j - \frac{\sum_{i=1}^n \phi_c'(r_i)y_i x_{ij}}{n}, \lambda_1\right)}{\frac{2}{\delta} + \lambda_2}, \tag{2.9}$$

where $S(z, t) = (|z| - t)_+ \mathrm{sgn}(z)$. We then set $\tilde{\beta}_j = \hat{\beta}_j^{\mathrm{C}}$ as the new estimate.

We use the same trick to update intercept $\beta_0$. Similarly to (2.8), we consider minimizing a quadratic approximation

$$Q(\beta_0|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) = \frac{\sum_{i=1}^n \phi_c(r_i)}{n} + \frac{\sum_{i=1}^n \phi_c'(r_i)y_i}{n}(\beta_0 - \tilde{\beta}_0) + \frac{1}{\delta}(\beta_0 - \tilde{\beta}_0)^2, \tag{2.10}$$

which has a minimizer

$$\hat{\beta}_0^{\mathrm{C}} = \tilde{\beta}_0 - \frac{\delta}{2}\frac{\sum_{i=1}^n \phi_c'(r_i)y_i}{n}. \tag{2.11}$$

We set $\tilde{\beta}_0 = \hat{\beta}_0^{\mathrm{C}}$ as the new estimate.

To sum up, we have a GCD algorithm for solving the HHSVM, see Algorithm 1. The beauty of Algorithm 1 is that it is remarkably simple and almost identical to the coordinate descent algorithm for computing the elastic net penalized regression. In Section 3, we provide a rigorous justification of the use of these $Q$ functions and prove that each univariate update decreases the objective function of the HHSVM.

**Algorithm 1.** The generalized coordinate descent algorithm for the HHSVM.

- Initialize $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$.
- Iterate 2(a)–2(b) until convergence:
  - 2(a). Cyclic coordinate descent: for $j = 1, 2, \ldots, p$,
    - (2.a.1) Compute $r_i = y_i(\tilde{\beta}_0 + \mathbf{x}_i^{\mathsf{T}}\tilde{\boldsymbol{\beta}})$.
    - (2.a.2) Compute
    $$\hat{\beta}_j^{\mathrm{C}} = \frac{S\left(\frac{2}{\delta}\tilde{\beta}_j - \frac{\sum_{i=1}^n \phi_c'(r_i)y_i x_{ij}}{n}, \lambda_1\right)}{\frac{2}{\delta} + \lambda_2}.$$
    - (2.a.3) Set $\tilde{\beta}_j = \hat{\beta}_j^{\mathrm{C}}$.
  - 2(b). Update the intercept term
    - (2.b.1) Recompute $r_i = y_i(\tilde{\beta}_0 + \mathbf{x}_i^{\mathsf{T}}\tilde{\boldsymbol{\beta}})$.
    - (2.b.2) Compute
    $$\hat{\beta}_0^{\mathrm{C}} = \tilde{\beta}_0 - \frac{\delta}{2}\frac{\sum_{i=1}^n \phi_c'(r_i)y_i}{n}.$$
    - (2.b.3) Set $\tilde{\beta}_0 = \hat{\beta}_0^{\mathrm{C}}$.

### 2.3 IMPLEMENTATION

We have implemented Algorithm 1 in a publicly available R package gcdnet. As demonstrated in glmnet, some implementation tricks can boost the speed of a coordinate descent algorithm. We use these tricks in our implementation of Algorithm 1.

For each fixed $\lambda_2$, we compute the solutions for a fine grid of $\lambda_1$'s. We start with $\lambda_{\max}$ which is the smallest $\lambda_1$ to set all $\beta_j$, $1 \le j \le p$ to be zero. To compute $\lambda_{\max}$, we first obtain estimates $\hat{y}$ for the null model without any predictor:

$$\hat{y} = \arg\min_y \frac{1}{n}\sum_{i=1}^{n} \phi_c(y_i y) = \arg\min_y \left[\frac{n_+}{n}\phi_c(y) + \frac{n_-}{n}\phi_c(-y)\right].$$

Then by the Karush–Kuhn–Tucker (KKT) conditions $\frac{1}{n}|\sum_{i=1}^{n} \phi_c'(\hat{y}) y_i x_{ij}| \le \lambda_1$ for all $j = 1, \ldots, p$, we have

$$\lambda_{\max} = \frac{1}{n}\max_j |\sum_{i=1}^{n} \phi_c'(\hat{y}) y_i x_{ij}|.$$

We set $\lambda_{\min} = \tau\lambda_{\max}$ and the default value of $\tau$ is $\tau = 10^{-2}$ for $n < p$ data and $\tau = 10^{-4}$ for $n \ge p$ data. Between $\lambda_{\max}$ and $\lambda_{\min}$, we place $K$ points uniformly in the log-scale. The default value for $K$ is 98 such that there are 100 $\lambda_1$ values.

We use the warm-start trick to implement the solution paths. Without loss of generality let $\lambda_1[1] = \lambda_{\max}$ and $\lambda_1[100] = \lambda_{\min}$. We already have $\hat{\boldsymbol{\beta}} = 0$ for $\lambda_1[1]$. For $\lambda_1[k+1]$, the solution at $\lambda_1[k]$ is used as the initial value in Algorithm 1.

For computing the solution at each $\lambda_1$, we also use the active-set cycling idea. The active-set contains those variables whose current coefficients are nonzero. After a complete cycle through all the variables, we only apply coordinatewise operations on the active set till convergence. Then we run another complete cycle to see if the active set changes. If the active set is not changed, then the algorithm is stopped. Otherwise, the active-set cycling process is repeated.

We need to repeatedly compute $r_i$ in Steps 2(a) and 2(b) of Algorithm 1. For that we use an updating formula. For $j = 1, \ldots, p, 0$, suppose $\beta_j$ is updated, let $\delta_j = \hat{\beta}_j^{\mathrm{C}} - \tilde{\beta}_j$. Then we update $r_i$ by $r_i = r_i + y_i x_{ij}\delta_j$.

We mention the convergence criterion used in Algorithm 1. After each cyclic update, we declare convergence if $\frac{2}{\delta}\max_j(\hat{\beta}_j^{\text{current}} - \hat{\beta}_j^{\text{new}})^2 < \epsilon$, as done in glmnet (Friedman, Hastie, and Tibshirani 2010). In glmnet, the default value for $\epsilon$ is $10^{-6}$. In gcdnet, we use $10^{-8}$ as the default value of $\epsilon$.

### 2.4 APPROXIMATING THE SVM

Although the default value of $\delta$ is 2 unless specified otherwise, Algorithm 1 works for any positive $\delta$. This flexibility allows us to explore the possibility of using Algorithm 1 to obtain an approximate solution path of the elastic net penalized SVM. The motivation for doing so comes from the fact that $\lim_{\delta\to 0}\phi_c(t) = [1-t]_+$. In Figure 2 panel (b), we show the Huberized hinge functions with $\delta = 0.01$, which is nearly identical to the hinge loss.

Define

$$R(\boldsymbol{\beta}, \beta_0) = \frac{1}{n}\sum_{i=1}^{n}\left[1 - y_i\left(\beta_0 + \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta}\right)\right]_+ + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}),$$

and

$$R(\boldsymbol{\beta}, \beta_0|\delta) = \frac{1}{n}\sum_{i=1}^{n}\phi_c\left(y_i(\beta_0 + \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta})\right) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}).$$

Let $(\widehat{\boldsymbol{\beta}}^{\text{svm}}, \hat{\beta}_0^{\text{svm}})$ be the minimizer of $R(\boldsymbol{\beta}, \beta_0)$. Algorithm 1 gives the unique minimizer of $R(\boldsymbol{\beta}, \beta_0|\delta)$ for each given $\delta$. Denote the solution as $(\widehat{\boldsymbol{\beta}}(\delta), \hat{\beta}_0(\delta))$. We notice that

$$\phi_c(t) \le (1-t)_+ \le \phi_c(t) + \delta/2 \quad \forall t,$$

which yields the following inequalities

$$R(\boldsymbol{\beta}, \beta_0|\delta) \le R(\boldsymbol{\beta}, \beta_0) \le R(\boldsymbol{\beta}, \beta_0|\delta) + \delta/2. \tag{2.12}$$

From (2.12), we conclude that

$$\inf_{\boldsymbol{\beta},\beta_0} R(\boldsymbol{\beta}, \beta_0) \le R(\widehat{\boldsymbol{\beta}}(\delta), \hat{\beta}_0(\delta)) \le \inf_{\boldsymbol{\beta},\beta_0} R(\boldsymbol{\beta}, \beta_0) + \delta/2. \tag{2.13}$$

Here is a quick proof of (2.13):

$$\begin{aligned}
\inf_{\boldsymbol{\beta},\beta_0} R(\boldsymbol{\beta}, \beta_0) &\le R(\widehat{\boldsymbol{\beta}}(\delta), \hat{\beta}_0(\delta)) \\
&\le R\left(\widehat{\boldsymbol{\beta}}(\delta), \hat{\beta}_0(\delta)|\delta\right) + \delta/2 \\
&\le R\left(\widehat{\boldsymbol{\beta}}^{\text{svm}}, \hat{\beta}_0^{\text{svm}}|\delta\right) + \delta/2 \\
&\le R\left(\widehat{\boldsymbol{\beta}}^{\text{svm}}, \hat{\beta}_0^{\text{svm}}\right) + \delta/2 \\
&= \inf_{\boldsymbol{\beta},\beta_0} R(\boldsymbol{\beta}, \beta_0) + \delta/2.
\end{aligned}$$

The two inequalities in (2.13) are independent of $\lambda_1, \lambda_2$. They suggest that we can use Algorithm 1 to compute the solution of a HHSVM with a tiny $\delta$ (say, $\delta = 0.01$) and then treat the outcome of Algorithm 1 as a good approximation to the solution of the elastic net penalized SVM. We have observed that a smaller $\delta$ results in a longer computing time. Our experience suggests that $\delta = 0.01$ delivers a good trade-off between approximation accuracy and computing time.

## 3. GCD AND THE MM PRINCIPLE

In this section, we show that Algorithm 1 is a genuine coordinate descent algorithm. These $Q$ functions used in each univariate update are closely related to the MM principle (De Leeuw and Heiser 1977; Lange, Hunter, and Yang 2000; Hunter and Lange 2004). The MM principle is a more liberal form of the famous expectation-maximization (EM) algorithm (Dempster, Laird, and Rubin 1977) in that the former often does not work with "missing data" (see Wu and Lange (2010) and references therein).

We show that the $Q$ function in (2.8) majorizes the $F$ function in (2.6):

$$F(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) \le Q(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}), \tag{3.1}$$

$$F(\tilde{\beta}_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) = Q(\tilde{\beta}_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}). \tag{3.2}$$

With (3.1) and (3.2), we can easily verify the descent property of majorization update given in (2.9):

$$\begin{aligned} F(\hat{\beta}_j^{\mathrm{C}}|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) &= Q\big(\hat{\beta}_j^{\mathrm{C}}|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}\big) + F\big(\hat{\beta}_j^{\mathrm{C}}|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}\big) - Q\big(\hat{\beta}_j^{\mathrm{C}}|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}\big) \\ &\le Q\big(\hat{\beta}_j^{\mathrm{C}}|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}\big) \\ &\le Q(\tilde{\beta}_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) \\ &= F(\tilde{\beta}_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}). \end{aligned}$$

We now prove (3.1) (note that (3.2) is trivial). By the mean value theorem

$$\phi_c(r_i + y_i x_{ij}(\beta_j - \tilde{\beta}_j)) = \phi_c(r_i) + \phi_c'(r_i + t^* y_i x_{ij}(\beta_j - \tilde{\beta}_j)) y_i x_{ij}(\beta_j - \tilde{\beta}_j) \tag{3.3}$$

for some $t^* \in (0, 1)$. Moreover, we observe that the difference of the first derivatives for the function $\phi$ satisfies

$$|\phi_c'(a) - \phi_c'(b)| = \begin{cases} 0 & \text{if } (a > 1, b > 1) \text{ or } (a < 1 - \delta, b < 1 - \delta), \\ |a - b|/\delta & \text{if } (1 - \delta < a \le 1, 1 - \delta < b \le 1), \\ |a - (1 - \delta)|/\delta & \text{if } (1 - \delta < a \le 1, b < 1 - \delta), \\ |b - (1 - \delta)|/\delta & \text{if } (1 - \delta < b \le 1, a < 1 - \delta), \\ |a - 1|/\delta & \text{if } (1 - \delta < a \le 1, b > 1), \\ |b - 1|/\delta & \text{if } (1 - \delta < b \le 1, a > 1). \end{cases}$$

Therefore we have

$$|\phi_c'(a) - \phi_c'(b)| \le |a - b|/\delta \quad \forall a, b, \tag{3.4}$$

and hence,

$$|\phi_c'(r_i + t^* y_i x_{ij}(\beta_j - \tilde{\beta}_j)) - \phi_c'(r_i)| \le 1/\delta |t^* y_i x_{ij}(\beta_j - \tilde{\beta}_j)| \tag{3.5}$$

$$\le 1/\delta |y_i x_{ij}(\beta_j - \tilde{\beta}_j)|. \tag{3.6}$$

Combining (3.3) and (3.6), we have

$$\begin{aligned} \phi_c(r_i + y_i x_{ij}(\beta_j - \tilde{\beta}_j)) &= \phi_c(r_i) + \phi_c'(r_i) y_i x_{ij}(\beta_j - \tilde{\beta}_j) \\ &\quad + \big(\phi_c'(r_i + t^* y_i x_{ij}(\beta_j - \tilde{\beta}_j)) - \phi_c'(r_i)\big) y_i x_{ij}(\beta_j - \tilde{\beta}_j) \\ &\le \phi_c(r_i) + \phi_c'(r_i) y_i x_{ij}(\beta_j - \tilde{\beta}_j) + 1/\delta \left(y_i x_{ij}(\beta_j - \tilde{\beta}_j)\right)^2. \end{aligned}$$

Summing over $i$ at the both sides of the above inequality and using $y_i^2 = 1$, $\frac{1}{n}\sum_{i=1}^n x_{ij}^2 = 1$, we get (3.1).

## 4. A FURTHER EXTENSION OF THE GCD ALGORITHM

In this section, we further develop a generic GCD algorithm for solving a class of large margin classifiers. Define an elastic net penalized large margin classifier as follows:

$$\min_{(\beta_0,\boldsymbol{\beta})} \frac{1}{n}\sum_{i=1}^{n} L(y_i(\beta_0 + \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta})) + P_{\lambda_1,\lambda_2}(\boldsymbol{\beta}), \tag{4.1}$$

where $L(\cdot)$ is a convex loss function. The coordinate descent algorithm cyclically minimizes

$$F(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) = \frac{1}{n}\sum_{i=1}^{n} L(r_i + y_i x_{ij}(\beta_j - \tilde{\beta}_j)) + p_{\lambda_1,\lambda_2}(\beta_j) \tag{4.2}$$

with respect to $\beta_j$, where $r_i = y_i(\tilde{\beta}_0 + \mathbf{x}_i^{\mathsf{T}}\tilde{\boldsymbol{\beta}})$ is the current margin. The analysis in Section 3 shows that the foundation of the GCD algorithm for the HHSVM lies in the fact that for the Huberized hinge loss, the $F$ function has a simple quadratic majorization function. To generalize the GCD algorithm, we assume that the loss function $L$ satisfies the following *quadratic majorization condition*

$$L(t + a) \le L(t) + L'(t)a + \frac{M}{2}a^2 \quad \forall t, a. \tag{4.3}$$

Under (4.3), we have

$$\begin{aligned} &F(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) \\ &\quad \le \frac{1}{n}\sum_{i=1}^{n}[L(r_i) + L'(r_i)y_i x_{ij}(\beta_j - \tilde{\beta}_j) + \frac{M}{2}x_{ij}^2(\beta_j - \tilde{\beta}_j)] + p_{\lambda_1,\lambda_2}(\beta_j) \\ &\quad = \frac{\sum_{i=1}^{n} L(r_i)}{n} + \frac{\sum_{i=1}^{n} L'(r_i)y_i x_{ij}}{n}(\beta_j - \tilde{\beta}_j) + \frac{M}{2}(\beta_j - \tilde{\beta}_j)^2 + p_{\lambda_1,\lambda_2}(\beta_j) \\ &\quad = Q(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}), \end{aligned}$$

which means that $Q$ is a quadratic majorization function of $F$. Therefore, like in Algorithm 1, we set the minimizer of $Q$ as the new update

$$\hat{\beta}_j^{\mathrm{C}} = \arg\min_{\beta_j} Q(\beta_j|\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$$

and the solution is given by

$$\hat{\beta}_j^{\mathrm{C}} = \frac{S(M\tilde{\beta}_j - \frac{\sum_{i=1}^{n} L'(r_i)y_i x_{ij}}{n}, \lambda_1)}{M + \lambda_2}.$$

Likewise, the intercept is updated by

$$\hat{\beta}_0^{\mathrm{C}} = \tilde{\beta}_0 - \frac{\sum_{i=1}^{n} L'(r_i)y_i}{Mn}.$$

To sum up, we now have a generic GCD algorithm for a class of large margin classifiers whose loss function satisfies the quadratic majorization condition, see Algorithm 2.

We now show that the quadratic majorization condition is actually satisfied by popular margin-based loss functions.

*Lemma 1.* (a) If $L$ is differentiable and has Lipschitz continuous first derivative, that is,

$$|L'(a) - L'(b)| \le M_1|a - b| \quad \forall a, b. \tag{4.4}$$

then $L$ satisfies the quadratic majorization condition with $M = 2M_1$.
(b) If $L$ is twice differentiable and has bounded second derivative, that is,

$$L''(t) \le M_2 \quad \forall t,$$

then $L$ satisfies the quadratic majorization condition with $M = M_2$.

*Proof.* For part (a) of Lemma 1, we notice that

$$L(t + a) = L(t) + L'(t_1)a = L(t) + L'(t)a + (L'(t_1) - L'(t))a$$

for some $t_1$ between $t$ and $t + a$. Using (4.4), we have

$$|(L'(t_1) - L'(t))a| \le M_1|(t_1 - t)a| \le M_1a^2,$$

which implies

$$L(t + a) \le L(t) + L'(t)a + \frac{2M_1}{2}a^2.$$

For part (b), we simply use Taylor's expansion to the second order

$$L(t + a) = L(t) + L'(t)a + \frac{L''(t_2)}{2}a^2 \le L(t) + L'(t)a + \frac{M_2}{2}a^2.$$

Figures 2 and 3 show several popular loss functions and their derivatives. We use Lemma 1 to show that those loss functions satisfy the quadratic majorization condition.

In Section 3, we have shown that the Huberized hinge loss has Lipschitz continuous first derivative in (3.4) where $M_1 = 1/\delta$. By Lemma 1, it satisfies the quadratic majorization condition with $M = 2/\delta$. In this case, Algorithm 2 reduces to Algorithm 1.

The squared hinge loss function has the expression $L(t) = [(1 - t)_+]^2$ and its derivative is $L'(t) = -2(1 - t)_+$. Direct calculation shows that (4.4) holds for the squared hinge loss with $M_1 = 2$. By Lemma 1, it satisfies the quadratic majorization condition with $M = 4$.

The logistic regression loss has the expression $L(t) = \log(1 + e^{-t})$ and its derivative is $L'(t) = -(1 + e^t)^{-1}$. The logistic regression loss is actually twice differentiable and its second derivative is bounded by $1/4$: $L''(t) = \sum_{i=1}^{n} \frac{e^t}{(1+e^t)^2} \le \frac{1}{4}$. By Lemma 1, it satisfies the quadratic majorization condition with $M = 1/4$.

We have implemented both Algorithms 1 and 2 in an R package gcdnet. In this article, we use hubernet, sqsvmnet, and logitnet to denote the funcion in gcdnet for computing the solution paths of the elastic net penalized Huberized SVM, squared SVM, and logistic regression.

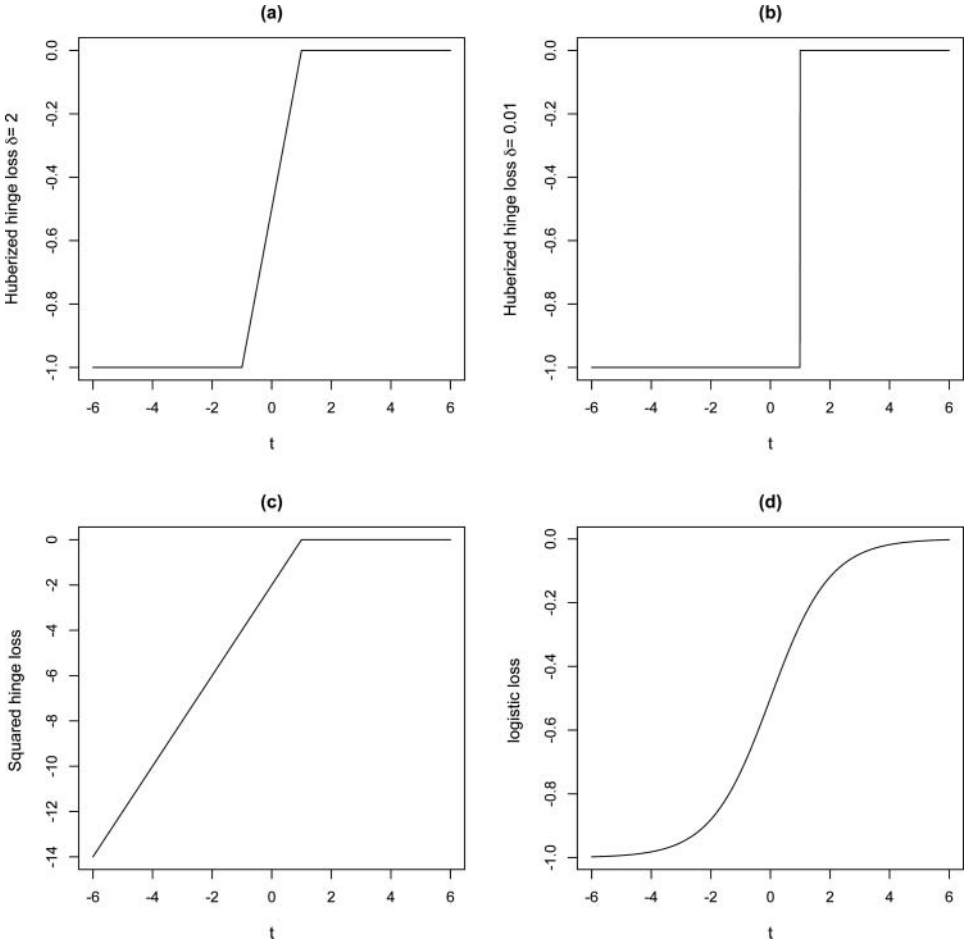


Figure 3. (a) The first derivative of the Huberized hinge loss function (with $\delta = 2$); (b) the first derivative of the Huberized hinge loss function (with $\delta = 0.01$); (c) the first derivative of the squared hinge loss function; (d) the first derivative of the logistic loss function.

## 5. NUMERICAL EXPERIMENTS

### 5.1 Comparing Two Algorithms for the HHSVM

We compare the run-times of hubernet and a competing algorithm by Wang, Zhu, and Zou (2008) for the HHSVM. The latter method is a LARS-type path algorithm that exploits the piecewise linearity of the HHSVM solution paths. The source code is available at *http://www.stat.lsa.umich.edu/~jizhu/code/hhsvm/* .

For notation convenience, the LARS-type algorithm is denoted by WZZ. For each given $\lambda_2$, WZZ finds the piecewise linear solution paths of the HHSVM. WZZ automatically generates a sequence of $\lambda_1$ values. To make a fair comparison, the same $\lambda_1$ sequence is used in hubernet. All numerical experiments were carried out on an Intel Xeon X5560 (Quad-core 2.8 GHz) processor.

**Algorithm 2.** The generic GCD algorithm for a class of large margin classifiers.

- Initialize $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$.
- Iterate 2(a)–2(b) until convergence:
  - 2(a) Cyclic coordinate descent: for $j = 1, 2, \ldots, p$,
    - (2.a.1) Compute $r_i = y_i(\tilde{\beta}_0 + \mathbf{x}_i^{\mathsf{T}}\tilde{\boldsymbol{\beta}})$.
    - (2.a.2) Compute
$$\hat{\beta}_j^{\mathrm{C}} = \frac{S\left(M\tilde{\beta}_j - \frac{\sum_{i=1}^n L'(r_i)y_i x_{ij}}{n}, \lambda_1\right)}{M + \lambda_2}.$$
    - (2.a.3) Set $\tilde{\beta}_j = \hat{\beta}_j^{\mathrm{C}}$.
  - 2(b) Update the intercept term
    - (2.b.1) Recompute $r_i = y_i(\tilde{\beta}_0 + \mathbf{x}_i^{\mathsf{T}}\tilde{\boldsymbol{\beta}})$.
    - (2.b.2) Compute
$$\hat{\beta}_0^{\mathrm{C}} = \tilde{\beta}_0 - \frac{\sum_{i=1}^n L'(r_i)y_i}{Mn}.$$
    - (2.b.3) Set $\tilde{\beta}_0 = \hat{\beta}_0^{\mathrm{C}}$.

We first use the *FHT model* introduced in Friedman, Hastie, and Tibshirani (2010) to generate simulated data with $n$ observations and $p$ predictors from the following model:

$$Z = \sum_{j=1}^{p} X_j\beta_j + k \cdot N(0, 1),$$

where the covariance between predictors $X_j$ and $X_{j'}$ has the same correlation $\rho$, with $\rho$ ranging from 0 to 0.95 and $\beta_j = (-1)^j \exp(-(2j-1)/20)$. We choose $k$ such that the signal-to-noise ratio is 3.0. Generate a binary response $Y$ according to $\Pr(Y = -1) = 1/(1 + \exp(-Z))$ and $\Pr(Y = +1) = 1/(1 + \exp(Z))$.

For each $\lambda_2$ in $(0, 10^{-4}, 10^{-2}, 1)$, we used WZZ and hubernet to compute the solution paths of the HHSVM over the same grid of $\lambda_1$ values. The process was repeated 10 times over 10 independent datasets. Tables 1 and 2 compare the run-times of hubernet and WZZ, where $\frac{t_{\mathrm{WZZ}}}{t_{\mathrm{hubernet}}}$ is the ratio of their run-times.

In Table 1, $n = 5000$ and $p = 100$, which corresponds to the traditional moderate-dimension larger-sample-size case. We see that hubernet is about 25–30 times faster than WZZ. It is also interesting to note that the relative improvement in speed is nearly independent of the correlation level and $\lambda_2$, although the two factors have noticeable impact on the actual computing times.

In Table 2, $n = 100$ and $p = 5000$, which corresponds to the high-dimension lower-sample-size case. When $\lambda_2 = 0$, the HHSVM uses the lasso penalty. In this case, the correlation has little impact on the speed of WZZ while higher correlation slightly increases the timing of hubernet. Nevertheless, hubernet is about 4–7 times faster than WZZ. When $\lambda_2 > 0$, the HHSVM uses a true elastic net penalty. We see that the advantage of hubernet over

Table 1. Timings (in seconds) for WZZ and hubernet for $n = 5000$, $p = 100$ data. Total time over the same grid of $\lambda_1$ values chosen by WZZ, averaged over 10 independent runs

| $n = 5000$, $p = 100$ | | | | | | |
|---|---|---|---|---|---|---|
| $\rho$ | 0 | 0.1 | 0.2 | 0.5 | 0.8 | 0.95 |
| | | | $\lambda_2 = 0$ | | | |
| WZZ | 88.42 | 85.51 | 86.97 | 86.56 | 96.24 | 113.83 |
| hubernet | 2.97 | 2.87 | 2.93 | 2.84 | 3.21 | 4.03 |
| $\frac{t_{\text{WZZ}}}{t_{\text{hubernet}}}$ | 29.79 | 29.68 | 30.48 | 29.98 | 28.25 | 28.25 |
| | | | $\lambda_2 = 10^{-4}$ | | | |
| WZZ | 87.87 | 84.93 | 86.43 | 86.08 | 95.78 | 113.24 |
| hubernet | 2.99 | 2.87 | 2.93 | 2.86 | 3.24 | 4.05 |
| $\frac{t_{\text{WZZ}}}{t_{\text{hubernet}}}$ | 29.39 | 29.59 | 29.50 | 30.10 | 29.56 | 27.96 |
| | | | $\lambda_2 = 10^{-2}$ | | | |
| WZZ | 80.53 | 76.65 | 79.38 | 78.73 | 87.45 | 107.24 |
| hubernet | 2.68 | 2.59 | 2.65 | 2.57 | 2.96 | 3.81 |
| $\frac{t_{\text{WZZ}}}{t_{\text{hubernet}}}$ | 30.05 | 29.59 | 29.95 | 30.63 | 29.54 | 28.15 |
| | | | $\lambda_2 = 1$ | | | |
| WZZ | 12.43 | 12.37 | 12.37 | 12.37 | 12.54 | 27.54 |
| hubernet | 0.45 | 0.48 | 0.49 | 0.50 | 0.53 | 1.03 |
| $\frac{t_{\text{WZZ}}}{t_{\text{hubernet}}}$ | 27.62 | 25.77 | 25.24 | 24.74 | 23.66 | 26.74 |

WZZ increases as $\lambda_2$ becomes larger. A closer examination shows that WZZ is significantly lowered down by a larger $\lambda_2$, while $\lambda_2$ has a much weaker impact on hubernet's timings.

We now use some popular benchmark datasets to compare WZZ and hubernet. Five datasets were considered in this study (see Table 3 for details). The first dataset Arcene

Table 2. Timings (in seconds) for WZZ and hubernet for $n = 100$, $p = 5000$ data. Total time over the same grid of $\lambda_1$ values chosen by WZZ, averaged over 10 independent runs

| $n = 100$, $p = 5000$ | | | | | | |
|---|---|---|---|---|---|---|
| $\rho$ | 0 | 0.1 | 0.2 | 0.5 | 0.8 | 0.95 |
| | | | $\lambda_2 = 0$ | | | |
| WZZ | 5.86 | 5.67 | 5.93 | 5.56 | 5.73 | 5.54 |
| hubernet | 0.87 | 0.82 | 0.86 | 0.86 | 1.01 | 1.16 |
| $\frac{t_{\text{WZZ}}}{t_{\text{hubernet}}}$ | 6.74 | 6.91 | 6.90 | 6.47 | 5.67 | 4.78 |
| | | | $\lambda_2 = 10^{-4}$ | | | |
| WZZ | 208.39 | 208.13 | 203.15 | 208.46 | 203.87 | 188.75 |
| hubernet | 2.71 | 2.69 | 2.70 | 2.76 | 2.94 | 3.01 |
| $\frac{t_{\text{WZZ}}}{t_{\text{hubernet}}}$ | 76.90 | 77.37 | 75.24 | 75.53 | 69.34 | 62.71 |
| | | | $\lambda_2 = 10^{-2}$ | | | |
| WZZ | 256.96 | 256.76 | 255.94 | 257.79 | 252.89 | 234.48 |
| hubernet | 2.97 | 2.96 | 3.03 | 2.95 | 2.95 | 2.96 |
| $\frac{t_{\text{WZZ}}}{t_{\text{hubernet}}}$ | 86.52 | 86.74 | 84.47 | 87.39 | 85.73 | 79.22 |
| | | | $\lambda_2 = 1$ | | | |
| WZZ | 292.75 | 292.87 | 292.91 | 292.84 | 291.45 | 282.51 |
| hubernet | 2.63 | 2.64 | 2.61 | 2.56 | 2.46 | 2.41 |
| $\frac{t_{\text{WZZ}}}{t_{\text{hubernet}}}$ | 111.31 | 110.94 | 112.23 | 114.39 | 118.48 | 117.22 |

Table 3. Timings (in seconds) of WZZ and hubernet for some benchmark real data, averaged over 10 runs

| HHSVM on benchmark datasets | | | | | |
|---|---|---|---|---|---|
| Data | $n$ | $p$ | WZZ | hubernet | $\frac{t_{\text{WZZ}}}{t_{\text{hubernet}}}$ |
| Arcene | 100 | 10000 | 37.17 | 2.46 | 15.09 |
| Breast cancer | 42 | 22283 | 3.16 | 0.46 | 6.85 |
| Colon | 62 | 2000 | 14.17 | 0.42 | 33.96 |
| Leukemia | 72 | 7128 | 52.20 | 1.42 | 36.69 |
| Prostate | 102 | 6033 | 302.91 | 3.47 | 87.2 |



is obtained from UCI Machine Learning Repository (Frank and Asuncion 2010). We also considered the breast cancer data in Graham et al. (2010), the colon cancer data in Alon et al. (1999), the leukemia data in Golub et al. (1999), and the prostate cancer data in Singh et al. (2002). For each dataset, we randomly split the data into a training set and a test set with ratio 4:1. The HHSVM was trained and tuned by five-fold cross-validation on the training set. Note that we did a two-dimensional cross-validation to find the best pair of $(\lambda_2, \lambda_1)$ that incurs minimal misclassification error. Let $\lambda_2^{\text{CV}}$ denote the chosen $\lambda_2$ by cross-validation. We reported the timing of WZZ and hubernet for computing solution paths of the HHSVM with $\lambda_2 = \lambda_2^{\text{CV}}$. The whole process was repeated 10 times. As can be seen from Table 3, hubernet is much faster than WZZ in all examples. It is also interesting to see that hubernet is very fast on all five datasets but WZZ can be very slow on prostate data.

### 5.2 Comparing hubernet, sqsvmnet, and logitnet

As shown in Section 4, the GCD algorithm provides a unified solution to three elastic net penalized large margin classifiers using Huberized hinge loss, squared hinge loss, and logistic regression loss. Here we compare the run times of hubernet for the HHSVM, sqsvmnet for the elastic net penalized squared hinge loss, and logitnet for the elastic net penalized logistic regression. We want to see how the loss function affects the computing time of GCD.

For the elastic net penalized logistic regression, Friedman, Hastie, and Tibshirani (2010) had developed a very efficient algorithm by combining Newton–Raphson and penalized weighted least squares. Their software is available in the R package glmnet. However glmnet uses a different form of the elastic net penalty: $P_{\lambda,\alpha}(\beta) = \lambda \sum_{j=1}^{p} \left[\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|\right]$. Fortunately, the glmnet code can be easily modified for the elastic net penalty $P_{\lambda_1,\lambda_2}(\beta)$ in (2.4). We have reimplemented glmnet and denote it by logitnet-FHT for notation convenience.

We first use the *FHT model* to generate simulation data with $n = 100$, $p = 5000$. Table 4 shows the run times (in seconds) of hubernet ($\delta = 2$), hubernet ($\delta = 0.01$), sqsvmnet, logitnet, and logitnet-FHT. Each method solves the solution paths for 100 $\lambda_1$ values for each $(\lambda_2, \rho)$ combination. First of all, Table 4 shows that all these methods are computationally efficient. Relatively speaking, the Huberized hinge loss ($\delta = 2$) and squared hinge loss lead to the fastest classifiers computed by GCD. As argued in Section 2.4, hubernet ($\delta = 0.01$) can be used to approximate the elastic net penalized SVM. Compared with the default hubernet (with $\delta = 2$), hubernet ($\delta = 0.01$) is about 10 times slower.

Table 4. Timings (in seconds) for hubernet ($\delta = 2$), hubernet ($\delta = 0.01$), sqsvmnet, logitnet, and logitnet-FHT in the elastic net penalized classification methods for $n = 100$, $p = 5000$ data. Total time for 100 values of $\lambda_1$, averaged over 10 independent runs

| $\rho$ | 0 | 0.1 | 0.2 | 0.5 | 0.8 | 0.95 |
|---|---|---|---|---|---|---|
| | | | $\lambda_2 = 0$ | | | |
| hubernet ($\delta = 2$) | 0.64 | 0.63 | 0.65 | 0.65 | 0.81 | 0.89 |
| hubernet ($\delta = 0.01$) | 6.25 | 5.46 | 5.87 | 7.21 | 9.11 | 8.92 |
| sqsvmnet | 0.49 | 0.47 | 0.48 | 0.50 | 0.59 | 0.65 |
| logitnet | 3.85 | 3.85 | 3.82 | 4.77 | 6.06 | 8.02 |
| logitnet-FHT | 0.48 | 0.50 | 0.54 | 0.62 | 0.81 | 1.24 |
| | | | $\lambda_2 = 10^{-4}$ | | | |
| hubernet ($\delta = 2$) | 0.64 | 0.62 | 0.64 | 0.67 | 0.81 | 0.88 |
| hubernet ($\delta = 0.01$) | 6.20 | 5.62 | 5.71 | 6.76 | 8.60 | 8.94 |
| sqsvmnet | 0.50 | 0.50 | 0.50 | 0.53 | 0.62 | 0.68 |
| logitnet | 3.78 | 3.81 | 3.79 | 4.69 | 6.07 | 8.05 |
| logitnet-FHT | 0.48 | 0.50 | 0.54 | 0.62 | 0.81 | 1.23 |
| | | | $\lambda_2 = 10^{-2}$ | | | |
| hubernet ($\delta = 2$) | 0.83 | 0.72 | 0.80 | 0.78 | 0.79 | 0.87 |
| hubernet ($\delta = 0.01$) | 6.07 | 5.31 | 5.38 | 7.05 | 9.47 | 9.39 |
| sqsvmnet | 0.51 | 0.47 | 0.47 | 0.50 | 0.58 | 0.63 |
| logitnet | 4.24 | 4.18 | 4.49 | 5.11 | 7.29 | 8.14 |
| logitnet-FHT | 0.51 | 0.52 | 0.56 | 0.63 | 0.80 | 1.15 |
| | | | $\lambda_2 = 1$ | | | |
| hubernet ($\delta = 2$) | 0.76 | 0.75 | 0.72 | 0.72 | 0.76 | 0.76 |
| hubernet ($\delta = 0.01$) | 12.92 | 13.29 | 13.69 | 15.69 | 20.99 | 11.45 |
| sqsvmnet | 0.64 | 0.61 | 0.60 | 0.62 | 0.74 | 0.73 |
| logitnet | 4.89 | 4.63 | 4.53 | 4.69 | 4.98 | 5.17 |
| logitnet-FHT | 0.78 | 0.74 | 0.76 | 0.75 | 0.71 | 0.77 |



We also observe that logitnet-FHT is about 8–10 times faster than logitnet. A possible explanation is that logitnet-FHT takes advantages of the Hessian matrix of the logistic regression model in its Newton–Raphson step, while logitnet simply uses $1/4$ to replace the second derivatives, which can be conservative. On the other hand, hubernet ($\delta = 2$) and sqsvmnet are at least comparable to logitnet-FHT and the former two even have noticeable advantages when the correlation is high.

We now compare timings and classification accuracy of hubernet ($\delta = 2$), hubernet ($\delta = 0.01$), sqsvmnet, logitnet, and logitnet-FHT on the five benchmark datasets (See details in Table 5). Each model was trained and tuned in the same way as described in Section 5.1. Average misclassification error on the test set from 10 independent splits is reported. Also reported is the run time (in seconds) for computing the solution paths with $\lambda_2$ chosen by five-fold cross-validation. We can see that the hubernet with $\delta = 2$ or $\delta = 0.01$ has better classification accuracy than other classifiers on Arcene, Colon, Leukemia, and Prostate, while sqsvmnet has the smallest error on breast cancer. In terms of timings, there are three datasets for which sqsvmnet is the winner and logitnet-FHT wins on the other two. The timing results for hubernet ($\delta = 2$) are very close to those of sqsvmnet and logitnet-FHT. Overall, hubernet ($\delta = 2$) delivers very competitive performance on these benchmark datasets.

Table 5. Testing error (%) and timings (in seconds) for some benchmark real data. The timings are for computing solution paths for hubernet ($\delta = 2$), hubernet ($\delta = 0.01$), sqsvmnet, logitnet, and logitnet-FHT with $\lambda_2$ chosen by cross-validation and over the grid of 100 $\lambda_1$ values, averaged over 10 runs

| Classification methods comparison on real data | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Arcene | | Breast cancer | | Colon | | Leukemia | | Prostate | |
| Method | Error | Time | Error | Time | Error | Time | Error | Time | Error | Time |
| hubernet ($\delta = 2$) | 21.00 | 1.35 | 19.71 | 1.12 | 17.90 | 0.23 | 3.16 | 0.73 | 9.35 | 0.77 |
| hubernet ($\delta = 0.01$) | 23.58 | 23.92 | 19.71 | 15.36 | 15.10 | 7.35 | 2.41 | 5.55 | 8.47 | 6.12 |
| sqsvmnet | 23.00 | 1.13 | 19.14 | 1.15 | 19.30 | 0.18 | 3.25 | 0.54 | 8.88 | 0.66 |
| logitnet | 25.00 | 9.71 | 19.57 | 6.53 | 21.20 | 1.13 | 3.91 | 6.23 | 9.05 | 4.77 |
| logitnet-FHT | 25.00 | 1.85 | 19.57 | 1.08 | 21.20 | 0.14 | 3.91 | 0.65 | 9.05 | 0.67 |

## 6. DISCUSSION

In this article, we have presented a GCD algorithm for efficiently computing the solution paths of the HHSVM. We also generalized the GCD to other large margin classifiers and demonstrated their performances. The GCD algorithm has been implemented in an R package gcdnet, which is publicly available from The Comprehensive R Archive Network at *http://cran.r-project.org/web/packages/gcdnet/index.html*.

As pointed out by a referee, in the optimization literature, there are other efficient algorithms for solving the HHSVM and the elastic net penalized squared SVM, logistic regression. Tseng (2009) proposed a coordinate gradient descent method for solving $\ell_1$ penalized smooth optimization problems. Nesterov (2007) proposed the composite gradient mapping idea for minimizing the sum of a smooth convex function and a nonsmooth convex function such as the $\ell_1$ penalty. These algorithms can be applied to the large margin classifiers considered in this article. We do not argue here that GCD is superior than these alternatives. The message we wish to convey is that the marriage between coordinate descent and MM principle could yield an elegant, stable, and yet very efficient algorithm for high-dimensional classification.

## SUPPLEMENTARY MATERIALS

The supplementary materials are available in a single zip file, which contains the R package gcdnet and a readme file (README.txt).

## ACKNOWLEDGMENTS

The authors thank the editor, an associate editor, and two referees for their helpful comments and suggestions. This work is supported in part by NSF grant DMS-08-46068.

*[Received October 2011. Revised March 2012.]*

## REFERENCES

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., and Levine, A. (1999), "Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays," *Proceedings of the National Academy of Sciences*, 96, 6745–6750. [411]

Bradley, P., and Mangasarian, O. (1998), "Feature Selection via Concave Minimization and Support Vector Machines," in *Machine Learning Proceedings of the Fifteenth International Conference (ICML'98)*, pp. 82–90.

Bühlmann, P., and van de Geer, S. (2011), *Statistics for High Dimensional Data*, Heidelberg: Springer. [396]

Burges, C. (1998), "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, 2, 121–167. [399]

Daubechies, I., Defrise, M., and De Mol, C. (2004), "An Iterative Thresholding Algorithm for Linear Inverse Problems With a Sparsity Constraint," *Communications on Pure and Applied Mathematics*, 57, 1413–1457. [397]

De Leeuw, J., and Heiser, W. (1977), "Convergence of Correction Matrix Algorithms for Multidimensional Scaling," in *Geometric Representations of Relational Data*, ed. J. C. Lingoes, Ann Arbor, MI: Mathesis Press, pp. 735–752.

Dempster, A., Laird, N., and Rubin, D. (1977), "Maximum Likelihood From Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, Series B, 39, 1–38. [404]

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004), "Least Angle Regression," *The Annals of Statistics*, 32, 407–451. [397]

Evgeniou, T., Pontil, M., and Poggio, T. (2000), "Regularization Networks and Support Vector Machines," *Advances in Computational Mathematics*, 13, 1–50. [399]

Frank, A., and Asuncion, A. (2010), "Arcene Data Set: UCI" Machine Learning Repository, Available at *http://archive.ics.uci.edu/ml/datasets/Arcene*.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007), "Pathwise Coordinate Optimization," *The Annals of Applied Statistics*, 1, 302–332. [397]

Friedman, J., Hastie, T., and Tibshirani, R. (2010), "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software*, 33, 1. [397,401,403,409,411]

Fu, W. (1998), "Penalized Regressions: The Bridge Versus the Lasso," *Journal of Computational and Graphical Statistics*, 7, 397–416. [397]

Genkin, A., Lewis, D., and Madigan, D. (2007), "Large-Scale Bayesian Logistic Regression for Text Categorization," *Technometrics*, 49, 291–304. [397]

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., and Lander, E. (1999), "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring," *Science*, 286, 531–537. [411]

Graham, K., de las Morenas, A., Tripathi, A., King, C., Kavanah, M., Mendez, J., Stone, M., Slama, J., Miller, M., Antoine, G., Willers, H., Sebastiani, P., and Rosenberg, C. (2010), "Gene Expression in Histologically Normal Epithelium From Breast Cancer Patients and From Cancer-Free Prophylactic Mastectomy Patients Shares a Similar Profile," *British Journal of Cancer*, 102, 1284–1293. [411]

Hastie, T., Tibshirani, R., and Friedman, J. (2001), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, New York: Springer Verlag. [396,399]

Hunter, D., and Lange, K. (2004), "A Tutorial on MM Algorithms," *The American Statistician*, 58, 30–37. [404]

Lange, K., Hunter, D., and Yang, I. (2000), "Optimization Transfer Using Surrogate Objective Functions," *Journal of Computational and Graphical Statistics*, 9, 1–20. [404]

Nesterov, Y. (2007), "Gradient Methods for Minimizing Composite Objective Function," Technical Report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain (UCL).

Osborne, M., Presnell, B., and Turlach, B. (2000), "A New Approach to Variable Selection in Least Squares Problems," *IMA Journal of Numerical Analysis*, 20, 389–404. [397]

Rosset, S., and Zhu, J. (2007), "Piecewise Linear Regularized Solution Paths," *The Annals of Statistics*, 35, 1012–1030. [397]

Singh, D., Febbo, P., Ross, K., Jackson, D., Manola, J., Ladd, C., Tamayo, P., Renshaw, A., D'Amico, A., Richie, J., Lander, E. S., Loda, M., Kantoff, P. W., Golub, T. R., and Sellers, W. R. (2002), "Gene Expression Correlates of Clinical Prostate Cancer Behavior," *Cancer Cell*, 1, 203–209. [397,411]

Tibshirani, R. (1996), "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society*, Series B, 58, 267–288. [396]

Tseng, P. (2001), "Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization," *Journal of Optimization Theory and Applications*, 109, 475–494. [401]

Tseng, P., and Yun, S. (2009), "A coordinate Gradient Descent Method for Nonsmooth Separable Minimization," *Mathematical Programming*, 117, 387–423. [413]

Van der Kooij, A. (2007), "Prediction Accuracy and Stability of Regression With Optimal Scaling Transformations," Ph.D. thesis, Child & Family Studies and Data Theory (AGP-D), Department of Education and Child Studies, Faculty of Social and Behavioural Sciences, Leiden University.

Vapnik, V. (1995), *The Nature of Statistical Learning Theory*, New York: Springer-Verlag. [396,399]

Wang, L., Zhu, J., and Zou, H. (2008), "Hybrid Huberized Support Vector Machines for Microarray Classification and Gene Selection," *Bioinformatics*, 24, 412–419. [396,397,399,400,401,408]

Wu, T. T., and Lange, K. (2008), "Coordinate Descent Algorithms for Lasso Penalized Regression," *The Annals of Applied Statistics*, 2, 224–244. [397]

——— and Lange, K. (2010), "The MM Alternative to EM," *Statistical Science*, 25, 492–505. [404]

Zhu, J., Rosset, S., Hastie, T., and Tibshirani, R. (2004), "1-Norm Support Vector Machines," *The Annual Conference on Neural Information Processing Systems*, 16.

Zou, H., and Hastie, T. (2005), "Regularization and Variable Selection via the Elastic Net," *Journal of the Royal Statistical Society*, Series B, 67, 301–320. [397,399,402]