Statistics and Its Interface Volume 2 (2009) 481–491

Robust and sparse bridge regression[*]

Bin Li[†] and Qingzhao Yu

It is known that when there are heavy-tailed errors or outliers in the response, the least squares methods may fail to produce a reliable estimator. In this paper, we proposed a generalized Huber criterion which is highly flexible and robust for large errors. We applied the new criterion to the bridge regression family, called *robust and sparse bridge regression* (RSBR). However, to get the RSBR solution requires solving a nonconvex minimization problem, which is a computational challenge. On the basis of recent advances in difference convex programming, coordinate descent algorithm and local linear approximation, we provide an efficient computational algorithm that attempts to solve this nonconvex problem. Numerical examples show the proposed RSBR algorithm performs well and suitable for large-scale problems.

Keywords and phrases: Coordinate descent, D.C. programming, Huber loss, Local linear approximation, Regularization.

1. INTRODUCTION

Consider a linear regression model: given p predictors x_{ij}, $j = 1, 2, \dots, p$ and the response y_i for the ith observation, $i = 1, 2, \dots, n$. The ordinary least squares (OLS) estimate $\hat{\beta}^{ols}$ minimizes

$$RSS = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2. \tag{1}$$

Despite its simplicity and unbiasedness, the OLS estimator is, however, not always satisfactory. Due to its high variability, the OLS estimate often does poorly in both prediction and interpretation, especially when the sample size n is not large compared to the number of variables p.

To achieve better prediction, Hoerl and Kennard (1970) introduced *ridge regression*, which stabilizes the estimates by placing a restriction on the joint solution value. That is

$$\hat{\beta}^{ridge} = \arg\min_{\beta} RSS \quad \text{s.t.} \sum_{j=1}^{p} |\beta|^2 \le S. \tag{2}$$

The ridge solution can also be obtained through the equivalent penalized formulation

$$\hat{\beta}^{ridge}(\lambda) = \arg\min_{\beta}\left(RSS + \lambda\sum_{j=1}^{p}|\beta|^2\right) \tag{3}$$

where $\sum |\beta|^2$ is the L_2 penalty on β and $\lambda \ge 0$ is the tuning parameter which regulates the strength of the penalty.

Frank and Friedman (1993) introduced the *bridge regression*, which minimizes RSS subject to a constraint $\sum |\beta|^\gamma \le S$ with $\gamma \ge 0$, without solving for the solution for any given γ. The bridge family includes ridge regression with $\gamma = 2$ and subset selection with $\gamma = 0$ as special cases. Notice that for $\gamma > 1$ the coefficients of bridge solutions are usually non-zero for all the variables. In addition, the L_γ penalty $(\sum |\beta|^\gamma)$ with $\gamma \ge 1$ is convex in β, while non-convex for $\gamma < 1$.

Tibshirani (1996) introduced the *lasso*, which minimizes RSS subject to the constraint $\sum |\beta| \le S$, as a special case of the bridge with $\gamma = 1$. Equivalently, the lasso solution can be defined as

$$\hat{\beta}^{lasso}(\lambda) = \arg\min_{\beta} RSS + \lambda\sum_{j=1}^{p}|\beta_j| \tag{4}$$

where $\lambda \ge 0$. By imposing an L_1-penalty on the regression coefficients, the lasso does continuous shrinkage and automatic variable selection simultaneously.

In general, the penalized optimization problem can often be described as:

$$\hat{\beta}(\lambda) = \arg\min_{\beta} L(Y, X\beta) + \lambda J(\beta), \tag{5}$$

where $L(Y, X\beta)$ is a non-negative loss function for goodness-of-fit, $J(\beta)$ is a non-negative penalty of model complexity, and λ is the non-negative tuning parameter that balances between goodness-of-fit and model complexity. Besides the bridge regression, SVM (Vapnik, 1998), SCAD (Fan and Li, 2001), group lasso (Yuan and Lin, 2006) and elastic net (Zou and Hastie, 2005) fall into this category.

Although the lasso is much more appealing in modern data analysis owing to its sparse representation, it suffers from the heavy-tailed errors or outliers in the response like other least squares methods. Namely, it may fail to produce a reliable estimator. To alleviate this problem, Wang et al. (2006) proposed *LAD-lasso* and Rosset and Zhu (2004)

[*]The authors would like to thank the referee for pointing out related work, and for constructive comments and suggestions that helped to improve the presentation of the paper.
[†]Corresponding author.

suggested *Huberized lasso* which uses the well-known Huber criterion as the loss function. Both methods use the least absolute deviation (LAD) cost for the large errors.

In this article we suggest an algorithm that not only can do regression shrinkage and variable selection (like lasso) and but also is resistant to outliers or heavy-tailed errors (like LAD). The basic idea is to use a class of nonconvex Huber-like criteria as the loss functions (see the left panel in Figure 1) and the L_γ norm of β (with $0 < \gamma \leq 1$) as the penalty. To efficiently solve this optimization problem (which is nonconvex), we utilize three optimization and approximation methods which were recently proposed in solving regularized statistical learning problems: (1) difference convex programming (An and Tao (1997), Liu et al. (2005) and Wu and Liu (2007)), (2) coordinate descent algorithm (Friedman et al. (2007) and Wu and Lange (2008)), (3) local linear approximation (Zou and Li, 2008).

The outline of this paper is as follows. Section 2 introduces the generalized Huber criterion and the proposed robust and sparse bridge regression (RSBR). Section 3 reviews the three key computation components used in the proposed algorithm to solve RSBR. The details of the algorithm are presented in Section 4. In Section 5, we examine the prediction and variable selection performance for RSBR via simulation studies, followed by an application on a real data set in Section 6. Related issues are discussed in Section 7.

2. ROBUST AND SPARSE BRIDGE REGRESSION

First, generalized Huber criterion is introduced. Then we describe the proposed robust and sparse bridge regression followed by an example of illustration.

2.1 Generalized Huber criterion and RSBR

The least squares criterion is well suited to normally distributed errors but can give poor performance for heavy-tailed errors or outliers in the response. One remedy is to remove influential observations from the least-squares fit. Another approach, termed *robust regression*, is to employ a fitting criterion that is not as vulnerable as least squares to unusual observations.

The common method of robust regression is *M-estimation*, introduced by Huber (1964). The general *M*-estimator minimizes the objective function

$$(6) \qquad \sum_{i=1}^{n} \rho(e_i) = \sum_{i=1}^{n} \rho\left(y_i - \beta_0 - \sum_{j=1}^{p} x_j\beta_j\right)$$

where the function ρ has the following properties: (1) $\rho(e) \geq 0$, (2) $\rho(0) = 0$, (3) $\rho(e) = \rho(-e)$, (4) $\rho(e_i) \geq \rho(e_j)$ for $|e_i| > |e_j|$. For example, for least-squares estimation, $\rho(e) = e^2$. Huber (1981) describes a well-known robust *M*-estimator employing a loss function that is less affected by very large residual values. The Huber criterion can be written as

$$(7) \qquad \rho_H(e) = \begin{cases} e^2 & |e| < K \\ K^2 + 2K(|e| - K) & |e| \geq K. \end{cases}$$

The parameter K describes where the transition from quadratic to linear takes place. Namely, errors smaller than K get squared while larger errors increase the loss linearly.

Rosset and Zhu (2004) suggested *Huberized lasso* which uses the Huber criterion as the loss function in the lasso to minimize

$$(8) \qquad \sum_{i=1}^{n} \rho_H\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right) + \lambda \sum_{j=1}^{p} |\beta_j|.$$

In this paper, we generalized the Huber criterion described in (7) to a class of *M*-estimators as follows

$$(9) \qquad \rho_\eta(e) = \begin{cases} e^2 & |e| < K \\ K^2 + 2\eta K(|e| - K) & |e| \geq K \end{cases}$$

where $0 \leq \eta \leq 1$. The left panel of Figure 1 illustrates the family of generalized Huber criteria with three different values of η at $K = 2$. The black solid line is the square loss function. The grey shaded region represents the region for all the possible generalized Huber criteria with η ranges from zero to one.

Remarks.

- The Huber criterion ρ_H corresponds to $\rho_{\eta=1}$, while the truncated least-squares criterion corresponds to $\rho_{\eta=0}$.
- The smaller the η, the more robust the loss function is against the outliers in response.
- Although the generalized Huber criterion is not convex (in $e \in \mathbb{R}$) for $0 \leq \eta < 1$, it can be decomposed as a difference of two convex functions (of e) as follows:

$$(10) \qquad \rho_\eta(e) = e^2 - \mathbf{I}(|e| > K)\left[e^2 + 2\eta K(K - |e|) - K^2\right],$$

  where $\mathbf{I}(a)$ is an indicator function, equal to one when a holds, otherwise zero. Note that the leading convex function is the square loss function. In Figure 1, the right panel shows the trailing convex function, i.e. the second term on the right-hand side of (10). Notice that the trailing convex function is *K-insensitive*, i.e. it is a constant within the range $[-K, K]$.
- The class of generalized Huber criteria falls into the category of *redescending M-estimator*, which includes Tukeys bi-weight, S-estimators (Maronna et al., 2006), and *t*-type scores (He et al., 2000).

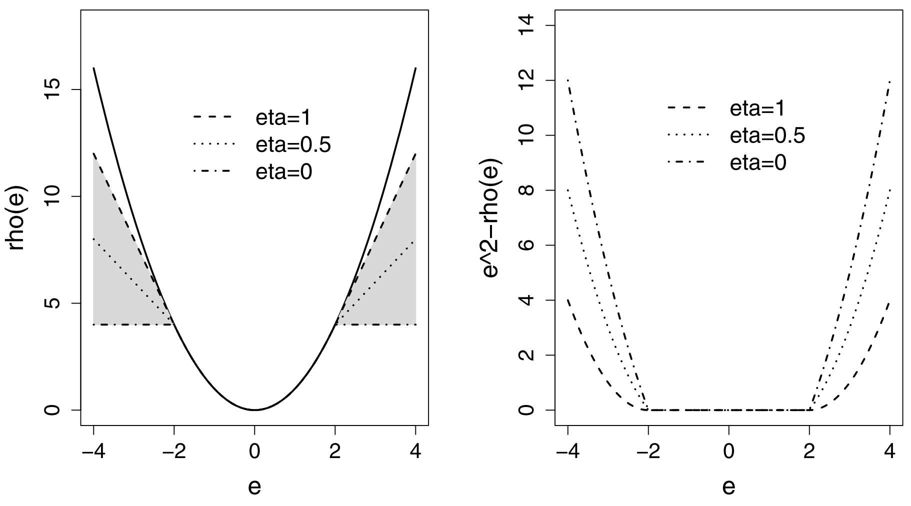


Figure 1. Illustration of three generalized Huber criteria with different values of η at $K = 2$ (left) and the corresponding trailing convex functions (right).

Based on the proposed generalized Huber criterion, we define the robust and sparse bridge regression which minimizes

$$Q(\beta) = \frac{1}{2}\sum_{i=1}^{n} \rho_\eta(e_i) + \lambda \sum_{j=1}^{p} |\beta_j|^\gamma, \tag{11}$$

where $e_i = y_i - \beta_0 - \sum x_{ij}\beta_j$ and $0 < \gamma \leq 1$. RSBR differs from bridge regression in two aspects: (1) the value of γ cannot go beyond one in order to have sparse solution, (2) the square loss function is replaced by generalized Huber criterion. Note that the Huberized lasso (Rosset and Zhu, 2004) belongs to the RSBR family, i.e. it is a special case of RSBR with $\gamma = 1$ and $\eta = 1$.

The reason of using generalized Huber criterion rather than the Huber criterion is that the former provides more flexibility of down-weighting the outliers in response than the latter. Namely, when the datasets are subject to extreme response outliers, which are commonly encountered in applications, RSBR can adaptively use more robust loss function ($\eta < 1$) than the Huber criterion ($\eta = 1$). As a result, RSBR can achieve superior performance to the Huberized lasso in the existence of outliers. This is illustrated by an example presented in the following section. Compared to other commonly used redescending estimators, the proposed class of generalized Huber criteria (used in the bridge family) is relatively easier to solve due to the recent advances in statistical computation and programming for regularized statistical learning problems. The details of the algorithm which attempts to minimize (11) are presented in Section 4.

2.2 An illustrative example

We simulated an example from the true model $\mathbf{y} = \mathbf{X}\beta + \epsilon$ where $\beta = (1, 0, 0, 0, 0)$ and $\epsilon \sim N(0, 0.5^2)$. The pairwise correlation between $\mathbf{x}_i$ and $\mathbf{x}_j$ was set to be 0.5 for all $i \neq j$.

Table 1. Summary of performance on simulation

	Lasso	Huberized Lasso	Truncated Lasso
Average MSE	0.747	0.612	0.588
PTTS	0.290	0.250	0.470
NDR	0.100	0.010	0.010

The model was fitted on the training set which consists of 20 observations. An independent set of 20 observations was used to select a tuning parameter λ. The test set consists of 1,000 independent observations. To include an outlier in the response, we randomly pick one observation from the training set and multiply its response value by ten, mimicking a typo of shifting the decimal point one place to the left. We compare the following three methods: the lasso; Huberized lasso (RSBR with $\eta = 1$ and $\gamma = 1$); *truncated lasso* (RSBR with $\eta = 0$ and $\gamma = 1$). For the last two methods, we set K at 95^{th} percentile for the distribution of $\{|e_i|\}_1^{20}$ for the training samples.

Table 1 summarizes the performance of three methods based on 100 replications. For prediction, we compare their averages of mean square errors (on test sets). For variable selection, we use the nondiscovery rates (NDR) and proportions of times that the true model is selected (PTTS). Based on Table 1, we see that truncated lasso does the best in both prediction and variable selection.

3. REVIEW OF RELATED COMPUTATIONAL METHODS

Solving RSBR requires handling a nonconvex minimization problem which is computationally difficult especially for high-dimensional problems. We propose an algorithm that attempts to solve RSBR based on three computation components developed recently. The outline of the algorithm is the following.

1. The L_γ bridge penalty is nonconvex (in β) for $0 < \gamma < 1$. By using the *local linear approximation* (LLA), we can approximate the L_γ penalty by a weighted L_1 penalty (like adaptive lasso), which is convex in β.
2. Although the generalized Huber criterion is nonconvex for $0 < \eta < 1$, it can be represented as a difference of two convex functions as in (10). Hence, by applying the *difference convex* (d.c.) programming, minimizing the nonconvex problem becomes solving a sequence of convex *lasso-type* subproblems.
3. To solve the convex lasso-type subproblems, we use the *coordinate descent* algorithm.

We review these three computation components in the next three sections, and describe the algorithm in Section 4.

3.1 Coordinate descent algorithm

For simplicity we assume that the predictors are standardized so that $\sum_i x_{ij}/n = 0$ and $\sum_i x_{ij}^2 = 1$, and the response is centered (i.e. $\sum_i y_i/n = 0$) from now on. With a single predictor, the lasso solution is a soft-thresholded version of the ordinary least squares estimate $\hat\beta^{ols}$ (Donoho and Johnstone, 1995):

$$\hat\beta^{lasso}(\lambda) = S(\hat\beta^{ols}, \lambda)$$
$$= \begin{cases} \hat\beta^{ols} - \lambda & \text{if } \hat\beta^{ols} > 0 \text{ and } |\hat\beta^{ols}| > \lambda \\ \hat\beta^{ols} + \lambda & \text{if } \hat\beta^{ols} < 0 \text{ and } |\hat\beta^{ols}| > \lambda \\ 0 & \text{if } |\hat\beta^{ols}| \le \lambda. \end{cases} \tag{12}$$

When the predictors are uncorrelated, the lasso solutions are still the soft-thresholded versions of the individual ordinary least squares estimates. But this is not the case when the predictors are correlated.

Efficient algorithms for solving the lasso solution have been proposed; see e.g. the "shooting" algorithm in Fu (1998), the homotopy algorithm in Osborne et al. (2000) and the least angle regression (LARS) approach in Efron et al. (2004). Friedman et al. (2007) and Wu and Lange (2008) developed the coordinate descent algorithms for lasso and lasso-related optimization problems. The rationale behind the coordinate descent algorithms is to iteratively solve a sequence of univariate problems (which is assumed to be simple to solve) with "partial residuals" as the response variable.

Denote $\langle \mathbf{x}_j, y\rangle = \sum_i x_{ij} y_i$, where $\mathbf{x}_j = (x_{1j}, \dots, x_{nj})$ for $j = 1, \dots, p$. To estimate β_j, the coordinate descent algorithm applies soft-thresholding on the partial residuals $y_i - \tilde y_i^{(j)}$ with all $\{\hat\beta_k\}_{k\ne j}$ fixed,

$$\hat\beta_j = \arg\min_{\beta_j} \frac{1}{2}\sum_{i=1}^n \left(y_i - \tilde y_i^{(j)} - x_{ij}\beta_j\right)^2 + \lambda|\hat\beta_j| + \lambda \sum_{k:k\ne j} |\hat\beta_k| \tag{13}$$

where $\tilde y_i^{(j)} = \sum_{k\ne j} x_k \hat\beta_k$. The soft-thresholding estimate of β_j in (13) can also be written as

$$\hat\beta_j \leftarrow \left(\sum_{i=1}^n x_{ij}(y_i - \tilde y_i^{(j)}), \lambda\right) \tag{14}$$

$$= S\left(\langle \mathbf{x}_j, y\rangle - \sum_{k:k\ne j} \langle \mathbf{x}_j, \mathbf{x}_k\rangle \hat\beta_k, \lambda\right) \tag{15}$$

In this paper we consider a particular form of the lasso-type problem, which minimizes

$$L_\lambda(\beta, \mathbf{v}) = RSS + \sum_{j=1}^p \lambda_j |\beta_j| + \sum_{j=1}^p v_j \beta_j \tag{16}$$

where $\lambda_j \ge 0$ and $v_j \in \mathcal{R}$. Notice that the objective function in (16) is very similar to the adaptive lasso (Zou, 2006) with an extra term for a weighted coefficient vector. The coordinate descent algorithm for minimizing (16) is the following.

Algorithm 1 Coordinate descent algorithm for minimizing (16)

```
1. Initialize $\{\hat\beta_j\}_{j=1}^p$.
2. Repeat until convergence of $\hat\beta$'s.
    For $j = 1, \dots, p$.
        $\hat\beta_j \leftarrow S\left(\langle \mathbf{x}_j, y\rangle - \sum_{k:k\ne j} \langle \mathbf{x}_j, \mathbf{x}_k\rangle \hat\beta_k - v_j, \lambda\right)$.
    End for loop.
3. End algorithm
```

By computing the inner products of each variable $\mathbf{x}_j$ with $\mathbf{y}$ initially and updating the covariance matrix sequentially, each coordinate step (Step 2 in Algorithm 1) can be easily updated. Studies (Friedman et al. 2007) show that the coordinate descent algorithm is highly competitive with the LARS and homotopy procedure in terms of computation efficiency for large-scale problems. For the convergence properties of coordinate descent algorithm in convex optimization problems, we refer the readers to Tseng (2001).

3.2 One-step LLA estimate

Coordinate descent algorithm has been successfully applied to solve the lasso and lasso-related methods (see Friedman et al., 2007). However, it cannot be applied to solve the bridge regression with $\gamma < 1$. Zou and Li (2008) proposed a unified algorithm based on the local linear approximation for maximizing the penalized likelihood for a broad class of nonconcave penalty functions.

Consider a well-chosen initial value $\beta_j^{(0)}$ for β_j, where β_j is close to $\beta_j^{(0)}$, and the penalty can be written as $\sum p_{\lambda_j}(|\beta_j|)$, which includes the bridge regression family. Based on LLA, the penalty function can be approximated by

$$p_{\lambda_j}(|\beta_j|) \approx p_{\lambda_j}(|\beta_j^{(0)}|) + p'_{\lambda_j}(|\beta_j^{(0)}|)(|\beta_j| - |\beta_j^{(0)}|),$$

where $p'_{\lambda_j}(|\beta_j^{(0)}|)$ is the derivative of $p_{\lambda_j}(\cdot)$ at $|\beta_j^{(0)}|$. For the bridge regression model, the LLA algorithm naturally provides a sparse one-step estimator by

$$\beta^{(1)} = \arg\min_{\beta} \left(RSS + \lambda \sum_{j=1}^{p} \gamma |\beta_j^{(0)}|^{\gamma-1} |\beta_j| \right). \tag{17}$$

Regarding the initial value, Zou and Li (2008) suggested to use the unpenalized estimate (e.g. OLS solution for linear regression model) as $\beta^{(0)}$. Note that (17) has the similar weighted penalty for each variable as the adaptive lasso (Zou, 2006), which can be solved efficiently by coordinate descent algorithm in Algorithm 1.

3.3 Difference convex programming

Although the LLA algorithm provides an approximate and sparse solution for the penalized regression model with a nonconvex penalty, it cannot be used to approximate nonconvex loss function such as generalized Huber criterion. However, by using the difference convex programming (An and Tao, 1997), which employs a decomposition of the loss function into a difference of two convex functions, we are able to find the approximate solution for the nonconvex problem.

Consider minimizing a nonconvex objective function $g(\mathbf{w})$ which is a difference of two convex functions, i.e. $g(\mathbf{w}) = g_1(\mathbf{w}) - g_2(\mathbf{w})$ where both $g_1(\mathbf{w})$ and $g_2(\mathbf{w})$ are convex in $\mathbf{w}$. The basic idea of d.c. programming is to construct a sequence of subproblems, which are obtained by replacing the trailing convex function, e.g. $g_2(\mathbf{w})$, by its affine minorization function $g_2(\mathbf{w}) + \langle \mathbf{w} - \mathbf{w}^{(o)}, \triangledown g_2(\mathbf{w}^{(o)}) \rangle$ and solve them iteratively, where $\triangledown g_2(\mathbf{w}^{(o)})$ is the subgradient of $g_2(\mathbf{w})$ at $\mathbf{w}^{(o)}$ with respect to $\mathbf{w}$. Specifically, given the solution for the $(m-1)$th subproblem $\mathbf{w}^{(m)}$, the mth subproblem solves

$$\begin{aligned} \mathbf{w}^{(m)} = \arg\min_{\mathbf{w}} g_1(\mathbf{w}) - \Big[g_2(\mathbf{w}^{(m-1)}) \\ + \langle \mathbf{w} - \mathbf{w}^{(m-1)}, \triangledown g_2(\mathbf{w}^{(m-1)}) \rangle \Big], \; m = 1, 2, \ldots, M. \end{aligned} \tag{18}$$

Note that after removing the constant terms in (18), minimizing the mth subproblem is equivalent to

$$\mathbf{w}^{(m)} = \arg\min_{\mathbf{w}} g_1(\mathbf{w}) - \langle \mathbf{w}, \triangledown g_2(\mathbf{w}^{(m-1)}) \rangle. \tag{19}$$

In our decomposition, the leading convex function has the same form as the adaptive lasso criterion (see (22) in Section 4), while the trailing one is the *K-insensitive loss* (see the right panel in Figure 1 and (22)). With this decomposition, the d.c. programming yields a solution by solving a sequence of lasso-type problems, which can be solved by the coordinate descent algorithm described in Algorithm 1. Liu et al. (2005) and Wu and Liu (2007) applied the difference convex programming to solve the ψ-learning problems and multicategory SVM.

4. ALGORITHM

By applying the LLA, the one-step estimator for (11) is:

$$Q(\beta)^* = \frac{1}{2} \sum_{i=1}^{n} \rho_\eta(e_i) + \lambda \sum_{j=1}^{p} \gamma |\beta_j^{(0)}|^{\gamma-1} |\beta_j|. \tag{20}$$

On the other hand, (10) shows that the generalized Huber criterion can be represented as the difference of two convex functions, which allows us to use d.c. programming to minimize (20). Namely, (20) can be represented as a diference of two convex functions as follows:

$$Q(\beta)^* = g_1(\beta) - g_2(\beta) \tag{21}$$

where

$$g_1(\beta) = RSS + \lambda \sum_{j=1}^{p} \gamma |\beta_j^{(0)}|^{\gamma-1} |\beta_j| \tag{22}$$

$$\begin{aligned} g_2(\beta) = \frac{1}{2} \sum_{i=1}^{n} \mathbf{I}(|e_i| > K) \big[e_i^2 \\ + 2\eta K(K - |e_i|) - K^2 \big]. \end{aligned} \tag{23}$$

The subgradient of $g_2(\beta)$ with respect to β at $\beta^{(o)}$ is

$$\triangledown g_2(\beta^{(o)}) = \sum_{i:|e_i|>K} \mathbf{x}_i \left(\eta K \times Sign(e_i) - e_i \right), \tag{24}$$

where $Sign(a)$ is the sign function, equal to one if a is positive, -1 if a is negative and zero otherwise. The inner product of β and subgradient $\triangledown g_2(\beta^{(o)})$ is

$$\langle \beta, \triangledown g_2(\beta^{(o)}) \rangle = - \sum_{j=1}^{p} v_j \beta_j \tag{25}$$

where

$$v_j = \sum_{i:|e_i|>K} Sign(e_i)\,(e_i - \eta K)\, x_{ij}. \tag{26}$$

Hence, by using d.c. programming, minimizing the objective function (20) becomes minimizing a sequence of subproblems

$$\hat{\beta} = \arg\min_{\beta} RSS + \lambda \sum_{j=1}^{p} \gamma |\beta_j^{(0)}|^{\gamma-1} |\beta_j| + \sum_{j=1}^{p} v_j \beta_j. \tag{27}$$

Note that the objective function in (27) is a special case of the lasso-type problems described in (16) with

$$\lambda_j = \lambda\gamma|\beta_j^{(0)}|^{\gamma-1} \quad \text{for } j = 1, \ldots, p. \tag{28}$$

The algorithm that attempts to solve RSBR is as follows.

Algorithm 2 RSBR Algorithm with fixed λ, γ, η and K.

```
1. Initialize $\hat{\beta}$, $m \leftarrow 0$ and $\beta^{(0)} \leftarrow \hat{\beta}$.
2. Calculate residuals $\{e_i\}_{i=1}^n$. Update $\{v_j\}_{j=1}^p$ and $\{\lambda_j\}_{j=1}^p$ as in (26) and (28).
3. Apply Algorithm 1 to estimate $\beta$ in (27).
4. Update $m \leftarrow m+1$ and $\beta^{(m)} \leftarrow \hat{\beta}$.
5. Repeat Step 2–4 above until convergence of $\beta^{(m)}$.
```

Remarks.

- Instead of using unpenalized estimate, we use the lasso estimate (with the same value of λ) as the initial values $\beta^{(0)}$ in Algorithm 2.
- Denote $||\beta||_2^2 = \sum_{j=1}^p \beta_j^2$. The algorithm terminates when $||\beta^{(m+1)} - \beta^{(m)}||_2/||\beta^{(m)}||_2 < \epsilon$, where ϵ is a pre-specified convergence tolerance. In this study, we set ϵ at 10^{-4}.

4.1 Tuning parameters

How to select the tuning parameters is an important issue in penalized regression problems. For RSBR, there are two main practical issues for the choice of tuning parameters.

(1) Although the ranges of γ and η are known ($(0,1]$ and $[0,1]$, respectively), the investigators usually don't know the ranges for λ and K in advance. The values of λ and K can differ substantially between different datasets.

To fix the ranges of tuning parameters, we reparametrize the tuning parameters as follows.

1. Let $\lambda^* = \lambda/\max\{|\sum_i \mathbf{x}_{ij}y_i|\}_1^p$.
2. Let K be the α-quantile of the distribution of $\{|e_i|\}_1^n$.

Notice that since $\mathbf{x}$ is standardized, $\sum \mathbf{x}_{ij}y_i$ is the univariate regression coefficient for $\mathbf{x}_j$. Although it is possible that λ^* can be larger than one, in practice the optimal value of λ^* usually falls between zero and one. In fact, the optimal value of λ^* is usually much smaller than one. On the other hand, consider the coordinate descent algorithm for the lasso with the initial values $\beta_j = 0$, $\forall j$. If λ^* is set to be greater than one, then the algorithm will not select any variable into the model. Hence, instead of λ and K, we tune λ^* and α, which both fall between zero and one.

One thing worth mentioning is that after reparametrization, the value of K depends on the distribution of residuals, which further depends on $\hat{\beta}$. Thus, in Algorithm 2 the value of K is updated in Step 2 after each iteration.

(2) Algorithm 2 has four tuning parameters. Hence the selection of tuning parameters from four dimensional space is computation intensive, especially for the large scale problems.

Although Algorithm 2 has four tuning parameters, the results shown in Section 5 imply that η and λ have greater effects on the RSBR performance than α (or K) and γ. Hence, in practice, we can fix α and γ at a few levels (on a coarse grid) and tune the values for λ and η (on a fine grid). This can substantially reduce the computation burden.

In practice, we can determine the value of α as follows. (1) Apply the lasso on the data and use cross-validation to find the value of tuning parameter λ. (2) Determine the proportion of outliers on the residuals after fitting the lasso. For example, the proportion of the residuals that are either greater than $(Q_3 + 1.5 \times IQR)$ or less than $(Q_1 - 1.5 \times IQR)$, where Q_1, Q_3 and IQR are the first quartile, third quartile and the inter-quartile range ($IQR = Q_3 - Q_1$) of the residuals, respectively. (3) Use the proportion of outliers we got in the previous step as the value of α.

The values of tuning parameters can be chosen by optimizing the performance via cross-validation or monitoring the performance on an independent validation set through grid search. In practice, we did a grid search for the optimal values of tuning parameters by using the efficient gradient *cleversearch*$(\cdot)$ function developed by Susanne Heim for R/S-PLUS.

5. SIMULATION STUDIES

The purpose of this simulation study is to show that the proposed RSBR is competitive with the lasso for the normal errors. However, when large errors exist (e.g. the error term has a Cauchy distribution), RSBR achieves better performance than both the lasso and Huberized lasso in terms of prediction accuracy and/or variable selection.

We simulated data from the true model $\mathbf{y} = \mathbf{X}\beta + \epsilon$ in four examples which were used in the original lasso paper (Tibshirani, 1996) to compare the prediction performance of the lasso and ridge regression systematically. For each example, we also considered two types of error distributions: normal (i.e. $\epsilon \sim N(0, \sigma^2)$) and Cauchy (i.e. $\epsilon \sim Cauchy(0,1)$, whose density function is $\frac{1}{\pi(1+x^2)}$).

For each example, our simulated data consist of a training set, an independent validation and test set. Models were fitted on training data only, and the validation data were used to select the tuning parameters. We computed the test error on the test data set. We use the notation $\cdot/\cdot/\cdot$ to describe the number of observations in the training, validation and test set respectively, e.g. 50/50/1000. Here are the details of the four examples.

1. In example 1, we simulated 100 data sets consisting of eight predictors, which are $\beta = (3, 1.5, 0, 0, 2, 0, 0, 0)$. The pairwise correlation between $\mathbf{x}_i$ and $\mathbf{x}_j$ was set to be $\text{corr}(i, j) = 0.5^{|i-j|}$. We set $\sigma = 3$ and sample size 50/50/1000.
2. Example 2 is the same as example 1, except that $\beta_j = 0.85$ for all j.
3. Example 3 is the same as example 1, except that $\beta_1 = 5$ and $\beta_j = 0$ for $j = \{2, \ldots, 8\}$ and $\sigma = 2$.
4. In example 4, we simulated 100 data sets consisting of 40 predictors which are

$$\beta = (\underbrace{0, \ldots, 0}_{10}, \underbrace{2, \ldots, 2}_{10}, \underbrace{0, \ldots, 0}_{10}, \underbrace{2, \ldots, 2}_{10})$$

and pairwise $\text{corr}(i, j) = 0.5$ for all i and j. We set $\sigma = 15$ and sample size 600/400/5000.

We compare the following six methods: (1) the lasso, (2) Huberized lasso (denoted as hlasso), (3) RSBR with $\gamma = 1$ and $\alpha = 0.90$ (denoted as 1/0.9), (4) RSBR with $\gamma = 1$ and $\alpha = 0.80$ (1/0.8), (5) RSBR with $\gamma = 0.01$ and $\alpha = 0.90$ (0.01/0.9), (6) RSBR with $\gamma = 0.01$ and $\alpha = 0.80$ (0.01/0.8). Notice that for the Huberized lasso, we did a two-dimensional search on α and the log-scale of λ^*, while for the other four RSBR methods, we did the two-dimensional search on η and the log-scale of λ^*. The searching ranges for both λ^* and η were set to be $[10^{-4}, 1]$, while the range for α was $[0.8, 1.0]$.

For the normal error case, the prediction performance is measured by the mean square error while for the Cauchy case it is measured by the *median* square error. Let d_{ij} be the prediction measure for the j^{th} competitor $(j = 1, \ldots, 6)$ in the i^{th} replication $(i = 1, \ldots, 100)$. To compare the prediction performance, we use the *comparative test error*, defined by

$$c_{i,j} = \frac{100 \times d_{ij}}{min\{d_{i,l}\}_{l=1,\ldots,6}},\ i = 1, \ldots, 100,\ j = 1, 2, 3, 4, 5, 6$$

over 100 replications for each of the six methods. This quantity facilitates individual comparisons by using the test error of the best method for each data set to calibrate the difficulty of the problem. Table 2 shows the average of comparative errors for the lasso and its competitors based on 100 random replications.

Based on Table 2, we have the following remarks. (1) For the normal error case, all six methods have very close prediction performance. On the average, no two methods differ more than 2% in terms of the prediction performance in all four examples (i.e. all pairwise differences within each example are less than 2% in Table 2 for the normal error case). (2) However, for the Cauchy error case, the lasso has substantially higher error rates than others, particularly in example 4. This is because in example 4 we have a relatively large training sample size such that more extreme outliers were included than the first three examples. (3) Huberized lasso performs inferior to other four RSBR methods which shows the advantage of using generalized Huber criterion over Huber loss. It also implies that η has greater effects on the prediction performance of RSBR than α and γ do. (4) "0.01/0.8" performs consistantly the best in all four examples.

Table 2. Summary of prediction performance in four examples

	Normal errors					
	lasso	hlasso	1/.9	1/.8	.01/.9	.01/.8
Ex. 1	102.55	102.24	102.10	102.91	104.02	103.84
Ex. 2	102.53	102.17	102.69	102.87	103.60	104.18
Ex. 3	103.04	102.84	102.56	102.66	103.47	103.11
Ex. 4	100.31	100.23	100.48	100.61	101.20	101.31
	Cauchy errors					
	lasso	hlasso	1/.9	1/.8	.01/.9	.01/.8
Ex. 1	199.42	128.21	117.65	113.11	109.12	106.82
Ex. 2	176.46	125.31	116.27	115.04	112.81	111.61
Ex. 3	194.46	135.77	122.51	121.81	107.57	105.91
Ex. 4	941.14	220.89	163.70	152.94	107.85	104.98

To compare the variable selection performance, we consider the nondiscovery rate (NDR) and false discovery rate (FDR) as well as the proportion of times that the true model is selected (PTTS). Table 3 presents the values of NDR, FDR and PTTS for six competitors.

Based on Table 3, we have the following remarks. (1) For the normal error case, the six methods have similar variable selection performance in general, and the lasso tends to have smaller NDR than its competitors. (2) However, for the Cauchy error case, the lasso has substantially higher NDR than others. On the other hand, the lasso has smaller FDR than the others in Example 1 & 3. For PTTS, we see that the lasso has lower chance to select the true model than others. It implies that when large errors exist, the lasso tends to select fewer variables into the model than RSBR does. (3) In both normal and Cauchy error cases, the Huberized lasso has similar variable selection performance as its RSBR competitors.

Model stability is an important issue in regression analysis. Generally, it is desired to have a stable estimate of regression coefficients. To compare model stability, we use the L_2 *distance variance* (L_2DV) criterion, defined by

$$L_2DV = Var\{||\hat{\beta}^{(i)} - \bar{\beta}||_2\}_{i=1}^{100}$$

where $\bar{\beta} = (\bar{\beta}_1, \ldots, \bar{\beta}_p)$ for $\bar{\beta}_j = \sum_{i=1}^{100} \hat{\beta}_j^{(i)}/100$. Note that $||\hat{\beta}^{(i)} - \bar{\beta}||$ is the Euclidean distance between the i^{th} estimate

Table 3. Summary of variable selection performance in four examples

	Method	Normal errors			Cauchy errors		
		NDR	FDR	PTTS	NDR	FDR	PTTS
1	lasso	.003	.544	.090	.417	.368	.040
	hlasso	.007	.554	.090	.110	.572	.060
	1/.9	.013	.594	.070	.110	.572	.060
	1/.8	.013	.598	.080	.083	.610	.090
	.01/.9	.007	.506	.110	.090	.602	.080
	.01/.8	.007	.492	.120	.117	.516	.090
2	lasso	.060	-	.640	.502	-	.180
	hlasso	.062	-	.650	.215	-	.450
	1/0.9	.066	-	.640	.164	-	.580
	1/0.8	.071	-	.610	.184	-	.560
	.01/.9	.075	-	.590	.220	-	.450
	.01/.8	.076	-	.580	.222	-	.420
3	lasso	0	.381	.190	.230	.320	.160
	hlasso	0	.373	.190	.060	0.443	.210
	1/.9	0	.436	.140	.040	0.550	.190
	1/.8	0	.440	.120	.060	0.531	.190
	.01/.9	0	.304	.200	.060	0.416	.210
	.01/.8	0	.303	.190	.060	0.389	.220
4	lasso	.022	.557	0	.310	.379	.000
	hlasso	.020	.572	0	.026	.503	.010
	1/0.9	.024	.546	0	.017	.495	.010
	1/0.8	.027	.546	0	.019	.470	.000
	.01/.9	.023	.533	0	.026	.356	.070
	.01/.8	.024	.520	0	.027	.342	.020

of β and the average of the estimate over 100 trials. Table 4 shows the values of L_2DV for each method under different scenarios.

Based on Table 4, we have the following remarks. (1) For the normal error case, the L_2DV values for the first four methods (i.e. the lasso, Huberized lasso, "1/0.9" and "1/0.8") are close to each other in all examples and are slightly lower than the ones for "0.01/0.9" and "0.01/0.8". This is because that when γ is very small, it tends to behave like the best subset selection method which corresponds to $\gamma = 0$. (2) For the Cauchy error case, the lasso has larger variation in estimating β than others in Example 1, 3 and 4 (particularly in Example 4). This agrees with the known fact that the least squares methods are unstable for the presence of large outliers in the response. In Example 2, the L_2DV values for all six methods are close to each other. This may be due to the fact that the underlying model is dense. In addition, the L_2DV values for the Huberized lasso are slightly higher than the other four RSBR competitors in all examples.

6. CALIFORNIA HOUSING DATA

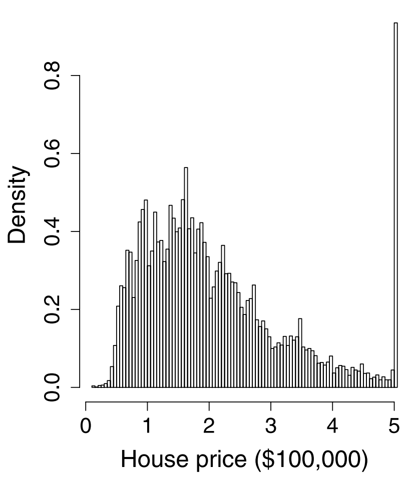
This data set, available at CMU *StatLib* repository (http://lib.stat.cmu.edu/datasets/), was originally used by Pace and Barry (1997). It consists of aggregated data from each of 20640 neighborhoods (1990 census block groups) in California. The response variable is the median house value in each neighborhood measured in units of \$100,000. There are eight continuous input variables, which are demographics (e.g. median income), housing density and occupancy, housing properties (e.g. number of rooms/bedrooms), and location of each neighborhood. Since the response variable median house value is highly skewed to right, logarithm transformation was applied before the analysis (see Pace and Barry, 1997). To examine the robustness property in the presence of outlying responses, in this study we use the raw median house value as the response without any transformation. Figure 2 shows the histogram of median house values (left) and the normal probability plot of the residuals after fitting the OLS regression model on all eight input variables (right). We see that there are many neighborhoods that have extremely large median house values. For example, in the data set, the skewness and kurtosis of the residuals shown in the right panel is as large as 1.25 and 9.25, respectively.

To examine the performance for RSBR in high-dimensional problems (i.e. p is large), we first applied the *sequential* Importance Sampled Learning Ensembles (ISLE, Friedman and Popescu, 2003) on the housing data. ISLE is a two-stage strategy to construct the final model. In the first stage, an ensemble of learning models is generated. In the second stage, the final quadrature coefficients (i.e. the weights of the generated learning models) are estimated through a regularized regression such as the lasso. In this study, we use the best performing tree ensemble considered in Friedman and Popescu (2003). Particularly, the sequential ISLE with the subsample proportion at 0.2 and shrinkage parameter $\nu = 0.01$. The sequential ensemble consists of 500 regression trees with the maximum depth of variable interaction at five (i.e. a model with up to five-way interactions). For the details of ISLE, we refer the readers to Friedman and Popescu (2003).

We randomly split the data ($n = 13760$) into three equal size sets: training, validation and test set (i.e. 6880/6880/6880). We compare the same six methods as

Table 4. Summary of L_2DV in four examples

Error	Method	Ex. 1	Ex. 2	Ex. 3	Ex. 4
Normal	lasso	0.991	1.343	0.408	4.431
	hlasso	0.976	1.320	0.416	4.487
	1/.9	0.943	1.332	0.442	4.562
	1/.8	1.012	1.363	0.412	4.590
	.01/.9	1.175	1.412	0.532	4.996
	.01/.8	1.111	1.460	0.518	4.973
Cauchy	lasso	1.978	1.599	2.317	6.844
	hlasso	1.364	1.734	1.035	1.063
	1/.9	1.262	1.549	1.001	0.918
	1/.8	1.158	1.426	1.004	0.842
	.01/.9	1.234	1.606	0.833	0.824
	.01/.8	1.150	1.573	0.865	0.750



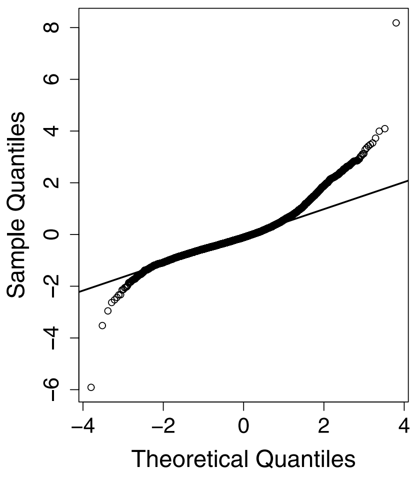


Figure 2. Histogram of median house values (left) and the normal probability plot of the residuals after fitting the OLS regression (right).

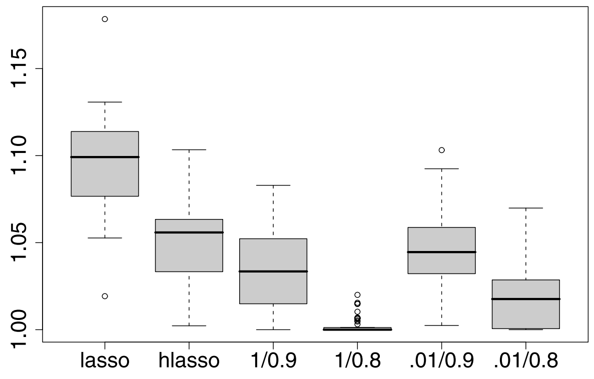


Figure 3. Boxplots of comparative test errors in California housing data.

the ones in the simulation study and use the same ranges for searching the tuning parameters. The prediction performance is measured by the median square errors on the test set. Figure 3 shows the boxplots of comparative test errors for the lasso and its competitors based on 50 random replications. We see that "1/0.8" achieves the best prediction performance. For example, it wins the best performance (i.e. the smallest median square errors on the test set) over 25 times out of 50 replications; and in the worst case (out of 50 replications) its median square error is only about 2% higher than the best method.

6.1 Computation issues

In this section we examine the computational efficiency for Algorithm 2 under various scenarios. The Algorithm 2 is implemented as R language functions, with the coordinate descent algorithm calling FORTRAN routine. All timings were carried out on an Intel Xeon 2.66GHz processor.

To vary the dimensionality of the problem, we randomly select a particular number of regression trees (without replacement) generated from sequential ISLE. To vary the sample size, we randomly select (without replacement) a particular number of neighborhoods from the entire data set. The average run times are calculated based on 10 random trials, in which the tuning parameters λ^*, η and α are randomly selected from a uniform distribution and γ is fixed at two levels: 1 and 0.01. Figure 4 shows the average CPU timings for various sample sizes and dimensionalities. We see that the average run times are approximately linear with the sample size n and dimensionality p in both cases.

In Figure 4 we see the average run times for $\gamma = 0.01$ are very close to the ones for $\gamma = 1$, which may contradict to the intuition that smaller γ should cost us more time to solve (i.e. for small value of γ, RSBR tends to behave like the best subset selection method which is NP-hard to solve.) Notice that in the proposed algorithm, for the computation consideration, the *one-step* LLA estimator is used. Both Meng (2008) and Bühlmann and Meier (2008) have suggested the possibility to go beyond the one-step estimator. Namely, they suggested to use multiple-step estimators in order to have some "safety-net" for guarding against accidental "unreasonable" initial values. If multiple-step LLA estimator is used, then solving the small value of γ may have considerably more computation than the large value of γ does.

An important factor related to the computational efficiency for Algorithm 2 is the number of iterations (of Step 2–4) needed to achieve the convergence of β. Here we use all 500 trees generated from sequential ISLE ($p = 500$). Table 5 shows the average numbers of iterations based on ten random trials under various numbers of training observations.

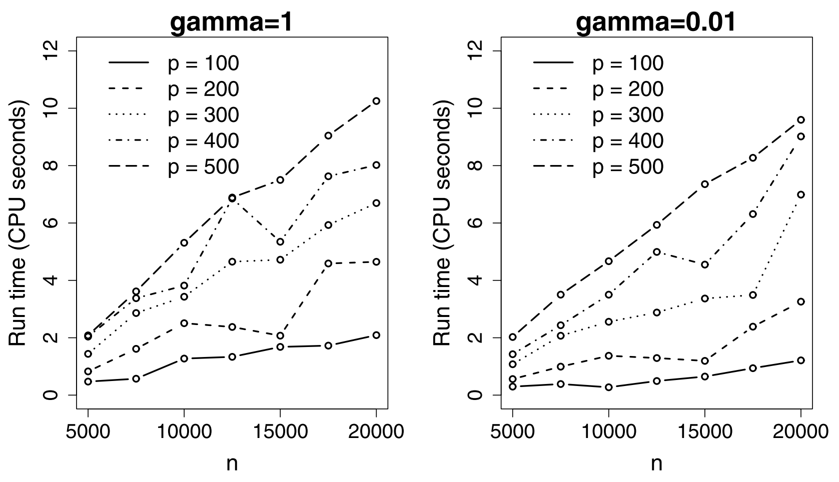


Figure 4. Average CPU timings (in seconds) for various sample sizes and dimensionalities with $\gamma = 1$ (left) and $\gamma = 0.01$ (right).

Table 5. Average numbers of iterations with different sample size

Sample size ($\times 10^3$)	5.0	7.5	10	12.5	15	17.5	20
Iterations	6.3	7.2	11.9	9.7	9.5	8.0	8.8

7. DISCUSSION

This article is devoted to computational developments of robust and sparse regression in the bridge family. Since both the generalized Huber loss function and the bridge penalty (for $\gamma < 1$) are nonconvex, the computational task becomes challenging. Two computational tools are used here: (1) d.c. programming, and (2) local linear approximation. The former decompose the generalized Huber loss into a difference of two convex functions, while the latter approximates the nonconvex penalty by a linear combination of weighted lasso penalties. Hence, minimizing the original nonconvex problem in RSBR becomes solving a sequence of convex lasso-type subproblems. By using the coordinate descent algorithm, we can solve the subproblems efficiently.

Although our focus in this article is on the bridge family, we believe the operation of generalized Huber loss also can be applied to other regression methods. An example of straightforward extension is to combine the generalized Huber loss with the elastic net penalty (Zou and Hastie, 2005), which consists of a linear combination of the lasso and ridge penalty. It is known that elastic net is equivalent to a lasso-type problem (see Section 2.2 in Zou and Hastie, 2005) and can be solved by coordinate descent algorithm (see Section 2 in Friedman et al., 2007). Hence, Algorithm 2 can also be applied to the robust elastic net regression using generalized Huber loss.

Received 25 August 2009

REFERENCES

[1] An, L.T.H. and Tao, P.D. (1997). Solving a class of linearly constrained indefinite quadratic problems by d.c. algorithms. *Journal of Global Optimization* **11** 253–285.

[2] Bühlmann, P. and Meier, L. (2008). Discussion: One-step sparse estimates in nonconcave penalized likelihood models. *Annals of Statistics* **36** 1534–1541.

[3] Donoho, D. and Johnstone, I. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association* **90** 1200–1224.

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

[5] Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association* **96** 1348–1360.

[6] Frank, I.E. and Friedman, J.H. (1993). A statistical view of some chemometrics regression tools. *Technometrics* **35** 109–148.

[7] Friedman, J.H., Hastie, T., Höfling, H. and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics* **1** 302–332.

[8] Friedman, J.H. and Popescu, B. (2003). Importance sampled learning ensembles. *Stanford University, Department of Statistics,* technical report.

[9] Fu, W. (1998). Penalized regressions: the bridge vs the lasso. *Journal of Computational and Graphical Statistics* **7** 397–416.

[10] He, X., Simpson, D.G. and Wang, G. (2000). Breakdown points of t-type regression estimators. *Biometrika* **87** 675–687.

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

[12] Huber, P.J. (1981). *Robust Statistics*. John-Wiley and Sons, New York.

[13] Liu, Y., Shen, X. and Doss, H. (2005). Multicategory psi-learning and support vector machine: computational tools. *Journal of Computational and Graphical Statistics* **14** 219–236.

[14] Maronna, R.A., Martin, D.R. and Yohai, V.J. (2006). *Robust Statistics: Theory and Methods*. Wiley.

[15] Meng, X. (2008). Discussion: One-step sparse estimates in nonconcave penalized likelihood models: Who cares if it is a white cat or a black cat? *Annals of Statistics* **36** 1542–1552.

[16] 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.

[17] Pace, R.K. and Barry, R. (1997). Sparse spatial autoregressions.

Statistics and Probability Letters **33** 291–297.

[18] Rosset, S. and Zhu, J. (2004). Discussion of "Least angle regression" by Efron, Hastie, Johnstone and Tibshirani. *Annals of Statistics* **32** 469–475.

[19] Tibshirani, R. (2004). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society*, Series B. **58** 267–288.

[20] Tseng, P. (2001). Convergence of block coordinate descent method for nondifferentiable maximization. *Journal of Optimization Theory and Applications* **109** 474–494.

[21] Wang, H., Li, G. and Jiang, G. (2006). Robust regression shrinkage and consistent variable selection via the lad-lasso. *Journal of Business and Economics Statistics* **11** 1–6.

[22] Wu, Y. and Liu, Y. (2007). Robust truncated-hinge-loss support vector machines. *Journal of the American Statistical Association* **102** 974–983.

[23] Wu, T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *Annals of Applied Statistics* **2** 224–244.

[24] Vapnik, V. (1998). *Statistical Learning Theory.* John Wiley, New York.

[25] Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*, Series B. **68** 49–67.

[26] Zou, H. (2006). The Adaptive lasso and its oracle properties. *Journal of the American Statistical Association* **101** 1418–1429.

[27] 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.

[28] Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models (with discussion). *Annals of Statistics* **36** 1509–1533.

Bin Li
Assistant Professor, Department of Experimental Statistics
Louisiana State University
E-mail address: bli@lsu.edu

Qingzhao Yu
Assistant Professor, School of Public Health
Louisiana State University Health Sciences Center
E-mail address: qyu@lsuhsc.edu