Published in final edited form as:

Comput Stat Data Anal. 2018 August ; 124: 235–251. doi:10.1016/j.csda.2018.03.006.

A Forward and Backward Stagewise Algorithm for Nonconvex Loss Functions with Adaptive Lasso

Xingjie Shi[a], **Yuan Huang**[b], **Jian Huang**[c], and **Shuangge Ma**[d,*]

[a]Department of Statistics, Nanjing Univ6ersity of Finance and Economics

[b]Department of Biostatistics, University of Iowa

[c]Department of Applied Mathematics, The HongKong Polytechnic University

[d]Department of Biostatistics, Yale School of Public Health

Abstract

Penalization is a popular tool for multi- and high-dimensional data. Most of the existing computational algorithms have been developed for convex loss functions. Nonconvex loss functions can sometimes generate more robust results and have important applications. Motivated by the BLasso algorithm, this study develops the Forward and Backward Stagewise (Fabs) algorithm for nonconvex loss functions with the adaptive Lasso (aLasso) penalty. It is shown that each point along the Fabs paths is a δ-approximate solution to the aLasso problem and the Fabs paths converge to the stationary points of the aLasso problem when δ goes to zero, given that the loss function has second-order derivatives bounded from above. This study exemplifies the Fabs with an application to the penalized smooth partial rank (SPR) estimation, for which there is still a lack of effective algorithm. Extensive numerical studies are conducted to demonstrate the benefit of penalized SPR estimation using Fabs, especially under high-dimensional settings. Application to the smoothed 0–1 loss in binary classification is introduced to demonstrate its capability to work with other differentiable nonconvex loss function.

Keywords

Forward and backward stagewise; Penalization; Nonconvex loss; Adaptive Lasso

1. Introduction

In multi- and high-dimensional regression analysis, penalization has been a popular technique. Consider a generic regression problem with a vector of unknown regression coefficients β. A penalization approach considers the objective function

*Correspondence to: Shuangge Ma. Address: 60 College ST, LEPH 206, New Haven, CT 06520, USA. shuangge.ma@yale.edu.

Supplementary document

Supplementary material related to this article can be found online.

$$Q(\beta;\lambda) = L(\beta) + \lambda\rho(\beta), \quad (1)$$

where $L(\beta)$ is the loss function, $\lambda(> 0)$ is the data-dependent tuning parameter, and $\rho(\cdot)$ is the penalty function. In the literature, multiple families of loss functions have been considered, including for example likelihood-based, estimating-equation-based, U-statistic-based, and others. A large number of penalty functions have been developed, including for example Lasso (and its variants such as adaptive Lasso), Bridge, MCP, SCAD, and many others. We refer to Bühlmann and van de Geer (2011), Hastie et al. (2015), and others for detailed discussions on penalization methods, theories, and applications.

In the literature, most effort has been on convex loss functions, for which both concave and convex penalties have been examined. In either case, non-differentiability at the origin is usually introduced for sparsity-inducing in order to conduct variable selection, making the study on computation as challenging as that on methodology and theory. Concave penalties, such as Bridge, SCAD, and MCP, have received extensive attention because of their satisfactory theoretical properties (Gasso et al., 2009). On the other hand, convex penalties have also been popular because of their significant advantage that global optimal solutions can be efficiently computed. In the convex penalty family, the most popular is Lasso (and its variants). Multiple effective algorithms have been developed for computing Lasso estimates for convex loss functions, such as the class of iterative shrinkage-thresholding algorithm (Becker et al., 2011), alternating direction method of multipliers (Boyd et al., 2011), LARS (Efron et al., 2004), coordinate descent (Friedman et al., 2007), and others.

The aforementioned algorithms, although successful in multiple aspects, are not directly applicable to nonconvex loss functions. Nonconvex loss functions have certain desirable features and have been the choice of many studies. Let X_i be the length-p vector of covariates and y_i be the response for observation i. One example is the smoothed maximal rank loss (Lin and Peng, 2013)

$$L(\beta) = -\frac{1}{n(n-1)}\sum_{i \neq j}^{n} I(y_i \geq y_j)S(X_i^\top\beta - X_j^\top\beta), \quad (2)$$

where S is a smoothing function used to approximate the indicator function and obtain a more approachable loss than the original maximal rank loss proposed by Han (1987). This smoothed maximal rank loss is a special case of the smoothed partial rank loss discussed in Section 4. Another popular example is the 0–1 loss for which a smoothed version is

$$L(\beta) = \frac{1}{n}\sum_{i=1}^{n} S(-y_iX_i^\top\beta). \quad (3)$$

Nguyen and Sanner (2013) proposes several approximate algorithms for minimizing the 0–1 loss. However, their algorithms can only accommodate low dimensional data for which the number of covariates is much smaller than the sample size. Other nonconvex loss functions can be seen in a wide array of rank-based estimations including the AUC loss, pairwise-difference rank loss, maximum score loss, and others.

In general, nonconvex loss functions are computationally more challenging than convex loss functions. Many gradient-free optimization methods can be used to minimize the loss functions mentioned above, and the Nelder-Mead (NM) simplex algorithm and the genetic algorithm (GA) are the most commonly used ones. The major advantage of these algorithms lies in their generality. The NM algorithm has been developed for minimizing the rank loss (Cavanagh and Sherman, 1998). The GA is a heuristic search algorithm inspired by the basic principles of biological evolution and natural selection. It performs well in many optimization problems with highly complex search spaces. With low dimensional covariates, their superior performance has been well documented (Cavanagh and Sherman, 1998; Sivanandam and Deepa, 2007). However, they do not scale well to high dimensionality.

In this article, our goal is to develop an effective computational algorithm for aLasso penalized estimation with differentiable nonconvex loss functions. The aLasso penalty takes the form of $\sum_{l=1}^{p} w_l |\beta_l|$, where $w = (w_1, \ldots, w_p)'$ is the weight vector. For a tuning parameter λ, the nonconvex minimization problem is

$$\min_{\beta} \left\{ L(\beta) + \lambda \sum_{l=1}^{p} w_l |\beta_l| \right\}. \quad (4)$$

We assume that the sets of stationary points for estimation (4) are nonempty for all λ. The development has been partly motivated by the BLasso (Zhao and Yu, 2007) – which has been developed to approximate the solution paths of Lasso penalization – because of its simplicity, computational efficiency, and parallelizable strategy. It has been proved that if the loss function is differentiable and *strictly convex*, then the BLasso paths converge to the Lasso paths. However, the applicability of BLasso to nonconvex and convex-but-not-strictly-convex loss functions is not clear. Advancing from the BLasso and other studies, we consider nonconvex loss functions. We develop the Fabs (forward and backward stagewise) algorithm, which simplifies the minimization procedures in BLasso. We establish the approximate optimality and convergence of the Fabs solution paths. As an application, we consider the aLasso penalized estimation with the nonconvex SPR (smoothed partial rank) and smoothed 0–1 losses. The "SPR+penalization" method has been shown in the literature to be useful for many practical problems (Song and Ma, 2010; Lin and Peng, 2013). However, the published studies have focused on low dimensional data with a small p. There is still a lack of effective computational algorithms for high dimensional data. Applying Fabs to this method not only demonstrates its application but more importantly provides an effective solution to this important method. Applications of Fabs are also demonstrated with the "0–1 loss+aLasso" method.

The rest of the article is organized as follows. For the integrity of this study and also to motivate the proposed algorithm, we briefly review the BLasso algorithm in Section 2. In

Section 3, we develop the Fabs algorithm and establish its approximate optimality and convergence properties. In Section 4, we apply the Fabs to penalized estimation with the nonconvex SPR loss. In Section 5, we apply the Fabs to solve the binary classification problem under the 0–1 loss with the aLasso penalty. Concluding remarks are in Section 6.

2. The BLasso algorithm

BLasso is closely related to the forward stagewise (FS) algorithm. FS has received a lot of attention because of not only its simplicity and computational efficiency but also its remarkable connection to the Lasso penalization with a convex loss. Hastie et al. (2007) characterizes FS as a monotone Lasso in that the FS and Lasso paths coincide if the Lasso solution has a monotone coordinate path, which, however, rarely happens in practice. With novel modifications to the FS, the BLasso algorithm produces an approximation to the Lasso path without making the monotonicity assumption. Similar to the FS, Blasso is iterative. At each iteration, Blasso can take a backward step which moves the estimate along one coordinate towards zero if this step can lead to a decrease of the objective function. Otherwise, it takes a forward step, in the same way as the FS. BLasso requires a convex loss but poses no constraint on differentiability, although its convergence to the Lasso paths is only established under strict convexity and differentiability. Hence here we describe the Blasso algorithm under the assumption of a differentiable $L(\beta)$.

To better motivate the Fabs, we show the BLasso's forward and backward steps for estimation (4). At iteration t, the coordinate k for taking the backward step is determined by

$$k = \arg \min_{l \in \mathscr{A}^t} L\left(\hat{\beta}^t - \frac{\text{sign}(\hat{\beta}_l^t)}{w_l} 1_l \varepsilon\right), \quad (5)$$

where 1_l denotes the length-p vector with the lth component being 1 and the rest being 0, ε is the step size, and $\mathscr{A}^t$ is the index set of $\hat{\beta}^t$'s nonzero components. If $Q(\hat{\beta}^t - \frac{\text{sign}(\hat{\beta}_k^t)}{w_k} 1_k \varepsilon; \lambda^t) < Q(\hat{\beta}^t; \lambda^t)$, a backward update is made by

$$\beta^{t+1} = \hat{\beta}^t - \frac{\text{sign}(\hat{\beta}_k^t)}{w_k} 1_k \varepsilon, \quad \lambda^{t+1} = \lambda^t. \quad (6)$$

Otherwise a forward step is taken on coordinate k that satisfies

$$k = \arg \min_{l = 1, \ldots, p} L\left(\hat{\beta}^t - \frac{\text{sign}\left(\nabla_l L(\hat{\beta}^t)\right)}{w_l} 1_l \varepsilon\right), \quad (7)$$

where ∇ denotes partial derivative. And the forward update is

$$\beta^{t+1} = \hat{\beta}^t - \frac{\mathrm{sign}\left(\nabla_k L(\hat{\beta}^t)\right)}{w_k} 1_k \varepsilon, \quad \lambda^{t+1} = \min\left\{\lambda^t, \frac{L(\beta^t) - L(\beta^{t+1})}{\varepsilon}\right\}. \quad (8)$$

The complete BLasso algorithm is presented in Appendix A.

3. The Fabs algorithm

Motivated by BLasso, we develop the Fabs which works for a differentiable but not-necessarily-convex loss function to approximately solve estimation (4). By assuming differentiability, the Fabs effectively uses the gradient information which helps reduce computational cost dramatically. Without the convexity constraint, the Fabs is more flexible and able to cover many high-dimensional models ($n \ll p$) and nonconvex loss functions.

3.1. Method

The Fabs is an iterative algorithm with each iteration conducting either a backward or forward step on the selected coordinate with a fixed step size. When evaluating the loss function, the Fabs takes advantage of the differentiability and uses the partial gradient $\nabla L(\beta^t)$. Specifically, we use the first-order Taylor expansion to obtain dominant terms of $L(\beta^t)$ at each iteration. In a backward step, we have

$$L\left(\hat{\beta}^t - \frac{\mathrm{sign}(\hat{\beta}^t_l)}{w_l} 1_l \varepsilon\right) = L(\hat{\beta}^t) - \nabla_l L(\hat{\beta}^t) \frac{\mathrm{sign}(\hat{\beta}^t_l)}{w_l} \varepsilon + O(\varepsilon^2). \quad (9)$$

In a forward step, we have

$$L\left(\hat{\beta}^t - \frac{\mathrm{sign}\left(\nabla_l L(\hat{\beta}^t)\right)}{w_l} 1_l \varepsilon\right) = L(\hat{\beta}^t) - \frac{|\nabla_l L(\hat{\beta}^t)|}{w_l} \varepsilon + O(\varepsilon^2). \quad (10)$$

When ε is small, a backward step determines the coordinate k by

$$k = \arg\min_{l \in \mathscr{A}^t} \left\{-\nabla_l L(\hat{\beta}^t) \frac{\mathrm{sign}(\hat{\beta}^t_l)}{w_l}\right\}. \quad (11)$$

A backward step can reduce the penalty $\lambda^t \rho(\beta^t)$ by $\lambda^t \varepsilon$, at the risk of increasing $L(\beta^t)$ by $\frac{|\nabla L_k|}{w_k}$. It will be taken if the former is bigger. Otherwise, a forward step will be taken, where the coordinate k can be determined by

$$k = \arg \max_{l = 1, \dots, p} \frac{|\nabla_l L(\hat{\beta}^t)|}{w_l}. \quad (12)$$

The complete algorithm is presented in Algorithm 1.

The intuition behind (12) is simple: if a forward step is taken, we update the current estimate in the direction that has the largest absolute gradient of L (evaluated at $\hat{\beta}^t$ and normalized by the aLasso weight). It is clear from (10) that a forward step always reduces the loss function $L(\beta^t)$. By applying forward/backward steps iteratively, the Fabs approximately solves (4).

Remark 1 (Connection with BLasso)—Blasso evaluates $L(\beta^t \pm \frac{1_l \varepsilon}{w_l})(l = 1, \dots, p)$ in searching for the direction for update in both forward and backward steps. Unlike BLasso which directly evaluates the loss function, the Fabs conducts the first-order Taylor expansion and uses the gradient $\nabla L(\beta^t)$. This not only simplifies the minimization procedure but also reduces the computational burden dramatically (See Example 1 in Section 4 for numerical results).

3.2. Statistical properties

In this section, we examine performance of the Fabs paths and discuss their convergence property. We first describe the behaviour of a forward step. Then we introduce the δ-optimality conditions, which are the popular Karush-Kuhn-Tucker (KKT) conditions up to a tolerance δ (Gasso et al., 2009). Using these conditions, we establish in Theorem 1 that the Fabs generates a δ-approximate solution to estimation (4), and in Proposition 1 the convergence as the stepsize $\varepsilon \to 0$. Proofs are provided in Appendix B.

Lemma 1—For any t such that $\hat{\beta}^{t+1}$ is updated by a forward step, Fabs has the following properties:

1. If a forward step is made by updating $\hat{\beta}_l^t$ where $l \in \mathscr{A}^t$, it can only be made by

$$\hat{\beta}_l^{t+1} = \hat{\beta}_l^t + \frac{sign(\hat{\beta}_l^t)}{w_l}\varepsilon.$$

2. $\rho(\hat{\beta}^{t+1}) = p(\hat{\beta}^t) + \varepsilon.$

Algorithm 1

Fabs for the aLasso penalty

Step 1 (Initialization) With a small step size $\varepsilon(> 0)$, set $t = 0$, and take an initial forward step:

$$k = \arg \max_{l = 1, \dots, p} \frac{|\nabla_l L(0)|}{w_l} \text{ and } \mathscr{A}^0 = \{k\},$$

$$\hat{\beta}^0 = -\frac{\text{sign}(\nabla_k L(0))}{w_k} 1_k \varepsilon,$$

$$\lambda^0 = \frac{1}{\varepsilon}\left[L(0) - L(\hat{\beta}^0)\right].$$

Step 2 (Backward and Forward steps)

2.1 Determine the backward direction:

$$k = \arg\min_{l \in \mathscr{A}^t} \left\{ -\nabla_l L(\hat{\beta}^t) \frac{\text{sign}(\hat{\beta}_l^t)}{w_l} \right\},$$

$$\Delta_k = -\frac{\text{sign}(\hat{\beta}_k^t)}{w_k} 1_k.$$

2.2 If $Q(\hat{\beta}^t + \Delta_k \varepsilon; \lambda^t) < Q(\hat{\beta}^t; (\lambda^t)$, take a backward step $\hat{\beta}^{t+1} = \hat{\beta}^t + \Delta_k \varepsilon$, and $\lambda^{t+1} = \lambda^t$.

Otherwise, force a forward step, and relax λ if necessary:

$$k = \arg\max_{l = 1, \ldots, p} \frac{|\nabla_l L(\hat{\beta}^t)|}{w_l} \text{ and } \mathscr{A}^{t+1} = \mathscr{A}^t \cup \{k\},$$

$$\hat{\beta}^{t+1} = \hat{\beta}^t - \frac{\text{sign}(\nabla_k L(\hat{\beta}^t))}{w_k} 1_k \varepsilon,$$

$$\lambda^{t+1} = \min\left\{\lambda^t, \frac{1}{\varepsilon}[L(\hat{\beta}^t) - L(\hat{\beta}^{t+1})]\right\}.$$

Step 3 (Iteration) Update $t = t + 1$, and repeat Steps 2 and 3 until $\lambda^t \le \lambda_{\min}$, where $\lambda_{\min}$ is the lower bound of the tuning parameter.

3. $Q(\hat{\beta}^{t+1}; \lambda^{t+1}) \le Q(\hat{\beta}^t; \lambda^t)$.

Lemma 1 shows that with a forward step, the updated coefficient increases in absolute magnitude, indicating that the solution path is monotone if no backward step is taken. Therefore, Lemma 1 (1) provides an intuitive connection between the FS algorithm and monotone Lasso. Second, Lemma 1 (2) shows that the penalty function always increases by a constant ε in a forward step. Last, Lemma 1 (3) shows that by taking a forward step, the objective function $Q(\beta^{t+1}; \lambda^{t+1})$ is no greater than $Q(\beta^t; \lambda^t)$. Since a backward step is taken if $Q(\hat{\beta}^{t+1}; \lambda^t) \le Q(\hat{\beta}^t; \lambda^t)$ (in this case, $\lambda^{t+1} = \lambda^t$), the objective function Q is guaranteed to decrease in both forward and backward steps along the Fabs' iterations.

Definition 1 (δ-approximate solution)—If a vector $\beta \in \mathbb{R}^p$ satisfies the following δ-optimality conditions ($\delta \ge 0$) for all $1 \le l \le p$

$$|\nabla_l L(\beta) + \lambda w_l sign(\beta_l)| \le \delta, \ if \ \beta_l \ne 0, \quad (13)$$

$$|\nabla_l L(\beta)| \le \lambda w_l + \delta, \ if \ \beta_l = 0, \quad (14)$$

then we call β a δ-approximate solution for estimation (4)

Theorem 1—If the second-order derivatives of L are bounded from above, then every point $\hat{\beta}^t$ along the Fabs path is a δ-approximate solution for one of the stationary points of estimation (4), with the associated tuning parameter λ^t and $\delta = \frac{m}{w_*}\varepsilon$, where m is the upper bound of the second-order derivatives of L and $w_* = \min_j\{w_j\}$.

Theorem 1 guarantees the satisfactory properties of each point along the Fabs path. When $\varepsilon = 0$, the δ-optimality conditions reduce to the KKT conditions. Immediately we have Proposition 1.

Proposition 1—If L has upper-bounded second-order derivatives, as $\varepsilon \to 0$, each point along the Fabs paths converges to one stationary point of estimation (4).

Proposition 1 holds under mild conditions. Without any convexity assumption on L, it guarantees that the solution found by the Fabs converges to one stationary point for estimation (4). It accommodates loss functions that are not convex, including those popular ones mentioned in Section 1.

Remark 2—When the loss function is differentiable and convex with continuous predictors, estimation (4) leads to a unique and well-defined Lasso path (Tibshirani, 2013). For our case here with a nonconvex loss, it is not clear whether estimation (4) has a unique solution and whether the Fabs can find the global or even local optimizers. The Fabs path consists of a sequence of solutions such that each solution is an approximation to one of the stationary points of the aLasso problem with the same tuning λ. We note that convergence to a stationary point is a common theoretical result in nonconvex optimization. For the Fabs, the initial estimate zero is a stationary point and global optimal for $\lambda = \infty$. For all the following steps, $\hat{\beta}^{t+1}$ is updated by a forward or backward step from $\hat{\beta}^t$ with a stepsize ε. Even when the stepsize $\varepsilon \to 0$, it is not clear whether the obtained solution is the global optimal, a local one, or even a saddle point. To escape from potential saddle points, stronger conditions on the loss functions, for example the strict saddle property (Ge et al., 2015), are needed, but further investigation is beyond the scope here. Our numerical Example 1 in Section 4 shows that the Fabs' performance is competitive to NM and GA which likely provide local optimal solutions.

Remark 3—When L is convex but not necessarily strictly convex, the KKT conditions are necessary and sufficient for global optimal solutions. Therefore, the Fabs solutions converge to the global optimizers by Proposition 1. A visualization of such convergence can be found in the supplementary document. The convergence of BLasso is developed under strict convexity, which requires $n \geq p$ for the linear model. In this sense, the Fabs extends the convergence result of BLasso to accommodate high-dimensional settings. Another interesting finding for Proposition 1 under the additional convexity assumption is that there is an intuitive connection between the Fabs and FS algorithms: the Fabs path is an approximation to the Lasso path since it finds solutions that satisfy an approximate version of Lasso's KKT conditions. The FS algorithm is equivalent to the Fabs when there is no backward step.

3.3. Software

To facilitate data analysis in this study and beyond, we have developed an R program with the computational core written in C to reduce computer time. The code and description are publicly available at https://github.com/shuanggema. With a modular structure, the code can be easily revised and extended to other models and settings.

4. Application to the penalized SPR estimation

4.1. Penalized SPR estimation

The SPR has been investigated in multiple publications. It effectively "avoids" specifying the link function and thus has the much desired robustness properties (robust to model mis-specification). Despite its usefulness and extensive attention to methodology and theory, there is a lack of research on computation with the penalized SPR. In what follows, we apply Fabs to the penalized SPR, which not only demonstrates the application of Fabs but also provides an effective computational solution to the useful SPR technique. The SPR is applicable to multiple data types. Here we consider censored survival data, and other simpler data types can be analyzed similarly.

Denote T and C as the survival and censoring times, respectively. Under right censoring, we observe $y = \min(T, C)$ and $\delta = I(T \le C)$. Consider the transformation model

$$g(T) = \beta^\top X + e,$$

where e is the random error with an unknown distribution, and $g(\cdot)$ is monotone increasing but otherwise unspecified. Denote the observed data on n iid subjects as $\{(y_i, \delta_i, X_i), i = 1, \ldots, n\}$. The rank-based loss function is defined as

$$L(\beta) = -\frac{1}{n(n-1)} \sum_{i \ne j}^{n} \delta_j I(y_i \ge y_j) I(X_i^\top \beta \ge X_j^\top \beta). \quad (15)$$

As this loss function is not continuous, approximations are usually adopted. Specifically, the SPR loss function is defined as

$$L(\beta; \sigma) = -\frac{1}{n(n-1)} \sum_{i \ne j}^{n} \delta_j I(y_i \ge y_j) S_\sigma(X_i^\top \beta - X_j^\top \beta), \quad (16)$$

where σ is a tuning parameter, and $S_\sigma(u) = \frac{1}{1 + \exp(-u/\sigma)}$ is the sigmoid function (Song et al., 2007). When p is moderate to large, penalization has been applied to the SPR estimation. Multiple published studies have established satisfactory statistical properties of the SPR approach, both with and without penalization. However, the computational aspect has not been well examined mostly because of the nonconvexity of the loss function. Here we apply Fabs to the SPR estimation with an aLasso penalty.

Computational complexity—The computational complexity of each Fabs step is lucid. The gradient ∇L can be evaluated in $O(pn^2)$ calculations. $O(p \log p)$ calculations are needed to find the coordinate k that has the largest $\frac{|\nabla_k L|}{w_k}$. Thus, at each step, $O(pn^2)$ calculations are required. To produce the whole path, roughly $O(\frac{1}{\varepsilon})$ steps are needed (Zhao and Yu, 2007). Thus, overall, $O(\frac{pn^2}{\varepsilon})$ calculations are needed to compute the solution path. With the computational complexity increasing slowly with p, the whole Fabs path is computationally affordable. This advantage is clearly observed from the computer time presented in Table 1. It is also worth noting that the forward and backward steps are both simple, and computation can be conducted in a parallel manner to further reduce computer time.

Tuning parameter—Accuracy of the sigmoid approximation depends on the tuning parameter σ. Theoretical investigation on its order under low-dimensional settings is conducted in Song et al. (2007), which develops a one-step approach to select σ with an initial value $\sigma = 1/\sqrt{n}$. Below, we show that the selection of σ is not needed with Fabs. Let $\hat{\beta}^t(\sigma, \varepsilon)$ denote the solution of the penalized SPR at step t with σ, ε, and the associated $\lambda^t(\sigma, \varepsilon)$. We have the following property.

Theorem 2—For any positive constant c, $\hat{\beta}^t(c\sigma, c\varepsilon) = c\hat{\beta}^t(\sigma, \varepsilon)$ and $\lambda^t(c\sigma, c\varepsilon) = \lambda^t(\sigma, \varepsilon)/c$ for each t.

Remark 4—Theorem 2 implies that the solution path $\bar{\beta}^t(\sigma, \varepsilon)$ is determined up to location by the ratio of σ and ε. This property holds for other loss functions that involve smoothing using the sigmoid function, such as the smoothed 0–1 loss illustrated in Section 5.

According to Theorem 2, we can run a grid of ε on a small fixed σ. In our simulation, we set $\sigma = 1/\sqrt{n}$ and use the grid $\varepsilon \in \{0.02, 0.01, 0.001\}$. Our numerical study suggests that Fabs is relatively robust to the choice of ε. Following Lin and Peng (2013), we use the number of nonzero coefficient estimates ($|\mathscr{A}^t|$) as an approximation to the generalized degree of freedom and select the optimal t or λ^t along the solution path by maximizing $\mathrm{BIC}^t = \log\left(-L(\hat{\beta}^t; \sigma)\right) - |\mathscr{A}^t| \frac{\log n}{2n}$.

4.2. Simulation

Simulation is designed to demonstrate the favorable performance of the Fabs compared to the benchmark algorithms (Example 1) and to showcase the advantages of SPR by making use of the Fabs compared to other model-based estimators (Examples 2 and 3).

In Example 1, the benchmark algorithms considered include BLasso, NM, and GA. For the Lasso penalty, NM and GA algorithms are directly applicable to the penalized SPR loss function using the R function *optim* and package *GA* (Scrucca, 2013), respectively. The benchmark algorithms, especially GA, are computationally extensive. Therefore, we consider low-dimensional cases with $p = 10$ and 20. In Example 2, we consider a linear model with three random error distributions. The alternative estimators considered include the least squares (LS) and least absolute deviation (LAD). In Example 3, we consider a right censored response under the accelerated failure time model with three random error distributions. We consider the Cox model as an alternative. As the Fabs is computationally fast, we consider high-dimensional cases for Example 2 and 3 with $p = 500$. For GA and LAD, the Lasso penalty is ready to be applied using existing packages, therefore, we adopt Lasso in Examples 1 and 2, which is a special case of aLasso with weights all equal to 1. In all examples, the true coefficient $\beta^* = (1, 1, -1, -1, \mathbf{0}_{p-4})$. For covariates, we consider multivariate normal distributions with mean $\mathbf{0}_p$ and correlation matrix AR(ϱ) (auto-regressive). More model specifications are described below.

Example 1—Consider data with a right censored response under the accelerated failure time model

$$\log(T_i) = \beta^{*\top} X_i + e_i,$$

where e_i is the random error. The censoring times are generated independently, and the censoring rate is about 20%. We consider two distributions for the random errors: N(0,1) and t_4. We consider the correlation matrix AR(0.3) for X. Combinations of $n = (200, 400)$ and $p = (10, 20)$ are used to observe the increase of computational cost. For all algorithms that involve gradients, the gradient computation is written in C, and the rest of the computation is in R. GA requires no knowledge or gradient information about the loss surface (Sivanandam and Deepa, 2007), but is extremely time-consuming and takes a large number of iterations to converge. Therefore, we present the GA results based on maximal 100 iterations under which the estimates are roughly comparable to the other algorithms.

We consider the runtime (in second) for generating the whole solution path to evaluate computational cost. We evaluate the loss function's percentage decrease (PD) as $\frac{L(0) - L(\hat{\beta})}{L(0)}$ and estimation error (Error) as $\left\| \frac{\hat{\beta}}{\|\hat{\beta}\|_2} - \frac{\beta^*}{\|\beta^*\|_2} \right\|^2$ for $\hat{\beta}$ at the finishing end of the solution path corresponding to $\lambda = 0$. For the low-dimensional case here, this unpenalized estimator is a natural choice. Results are based on 200 replicates and shown in Table 1. We see that Fabs is much faster than BLasso, and both are significantly faster than NM and GA. When p increases from 10 to 20, the computer time of Fabs increases almost linearly. This enables Fabs to accommodate high dimensionality, which is not feasible with NM and GA. We observe a decrease in PD and an increase in Error when the noise distribution changes from $N(0, 1)$ to t_4. NM and GA have slightly large PD and Error. Fabs and BLasso have similar PD and Error, but Fabs is considerably faster. Figure 1 plots the solution paths of one replicate with $(n, p) = (200, 20)$ and t_4 random error. For Fabs, the solution paths generated under different ε values are similar with a smaller ε corresponding to a smoother path. In contrast, the NM and GA paths show more fluctuations. For BLasso, the solution paths are similar to those of Fabs.

Example 2—Consider data with a continuous response under the linear regression model

$$T_i = \beta^{*\top} X_i + e_i.$$

With $n = 200$ and $p = 500$, three distributions are considered for the random error: N(0,1), $0.7N(0, 1)$+0.3Cauchy(0, 1), and t_2. With the high dimensionality, penalization is necessary. Tuning parameter λ is selected by 10-fold cross-validation for LS and LAD, and by BIC for SPR.

Besides Error (defined in the same way as under Example 1), we also evaluate variable selection performance using the true positive rate (TPR) and false positive rate (FPR):

$$\mathrm{TPR} = \frac{\sum_{i=1}^{p} I(\beta_i^* \neq 0) I(\hat{\beta}_i \neq 0)}{\sum_{i=1}^{p} I(\beta_i^* \neq 0)}, \quad \mathrm{FPR} = \frac{\sum_{i=1}^{p} I(\beta_i^* = 0) I(\hat{\beta}_i \neq 0)}{\sum_{i=1}^{p} I(\beta_i^* = 0)}.$$

Results are based on 200 replicates and shown in Table 2. With both Fabs and BLasso, the numerical results of SPR are robust across all values of ε's. Fabs and BLasso have similar Error, TPR, and FPR across all scenarios. All methods have better performance with $\varrho = 0.3$. When the random errors are normally distributed, LS is appropriate. In this case, SPR has comparable performance as LS for $\varrho = 0.3$, indicating its good efficiency. When the correlation is high, LS is preferred with smaller FPR/Error and higher TPR. With the contaminated and heavy-tailed random errors, SPR significantly outperforms LS and LAD for all ε.

Example 3—Consider data with a right censored response under the accelerated failure time model. With $n = 200$ and $p = 500$, three error distributions are considered: the extreme value (EV) distribution with density $f(e) = \exp(e - \exp(e))$ that corresponds to the Cox model, N(0,1), and t_4. The censoring times are generated independently, and the censoring rate is about 20%. We consider the popular Cox model based analysis. Both SPR and Cox are penalized by aLasso, where the weights are calculated from marginal analysis.

Summary results based on 200 replicates are shown in Table 3. As previously observed, both the Fabs and BLasso estimates are not sensitive to ε, and all three methods have better performance under weak correlation. With the EV and Normal errors, SPR shows slightly better performance than Cox. With the t_4 errors, SPR generated by both Fabs and BLasso is favored with significantly better estimation and variable selection accuracy.

4.3. Real data examples

We demonstrate the practical application of Fabs to the SPR estimation using two real survival data examples. First, we consider the loan data which has $n \gg p$. For a more challenging task, we consider the data on skin cutaneous melanoma (SKCM) which has $n \ll p$. We compare the SPR estimation with the popular Cox model penalized by aLasso.

Example 1 (loan data)—The dataset is available at the Lending Club (LC) website (https://www.lendingclub.com/info/download-data.action). We retrieve data for the period between June 2007 and March 2015 and retain the records of loans scheduled with 64 payments for two high risk groups (LC grade F and G). Records with loans that were in the process of being repaid are removed, and records with loans that did not default over the study period are censored. The goal of analysis is to identify borrowers' characteristics (both main effects and interactions) that are associated with default risk. The processed data contain records for a total of 2626 high-risk borrowers with 797 defaults. The financial characteristics analyzed include self-reported annual income, loan amount, employment length, number of inquiries, verification status of self-reported annual income, debt to income ratio, number of derogatory public record, credit balance, number of accounts, and account utilization rate.

We apply aLasso-SPR and present the Fabs paths in the supplementary document. Estimates using both aLasso-SPR and aLasso-Cox are presented in Table 4. With different modeling strategies, the two sets of coefficients are not directly comparable. Specifically, for aLasso-SPR, positive estimates are associated with a lower risk, while for aLasso-Cox, positive estimates are associated with a higher risk. As shown in Table 4, both methods identify self-reported annual income as having an important main effect with a higher income associated with a lower default risk. aLasso-SPR also identifies loan amount as positively associated with default risk. Unlike aLasso-Cox which selects only main effects, aLasso-SPR also selects a few interaction terms. Quick literature search suggests that the aLasso-SPR findings are consistent with those published.

We also evaluate prediction performance. We conduct 300 random splits of the 2626 observations. In each split, we randomly select 1970 (75%) as the training set and the rest 656 as the testing set. We apply all three methods to the training set, obtain estimated coefficients, and predict default risk in the testing set. Prediction accuracy is evaluated using the censoring-adjusted AUC (C-statistics) as described in Uno et al. (2011) and implemented using the R package *survAUC*. The average AUCs are 0.571, 0.569, and 0.568 for aLasso-SPR, non-penalized SPR, and aLasso-Cox, respectively. Comparing these AUCs using a paired nonparametric test suggests that the differences are significant with p-values 1.4×10^{-7} (aLasso-SPR versus aLasso-Cox) and 9.4×10^{-10} (aLasso-SPR versus non-penalized SPR).

Example 2 (SKCM data)—We analyze the SKCM dataset downloaded from The Cancer Genome Atlas (https://tcga-data.nci.nih.gov). After removing samples with missing survival time and genes with minimal expression variations, we obtain 17,944 gene expressions on 278 patients for downstream analysis.

We report the estimates of aLasso-SPR and aLasso-Cox in Table 4. See the supplementary document for the whole Fabs path and tuning parameter selection plot for aLasso-SPR. aLasso-SPR identifies genes CREG1, DDX60, IFI5, MAD1L1, and TTYH2. Literature search suggests that these genes have strong associations with cancer and the signs are consistent with biomedical literature. For example, Znidar et al. (2016) suggests that upregulation of DDX60 is associated with reduced tumor growth. Zhang et al. (2013) reports a consistently significant increase of anti-viral responses of host cells in the presence of IFIT5 and concludes that IFIT5 is an important enhancer in innate immune response. Toiyama et al. (2007) proposes that TTYH2 is implicated in tumor growth and metastasis by directly altering tumor cell properties (aggregation) after observing enhanced adhesion and cell-cell aggregation in low TTYH2-expressing Caco-2 and DLD-1 cell lines. Ohta et al. (2002) shows that Mad1 (alias for MAD1L1) gene transfer inhibits the proliferation of human melanoma cells.

Prediction performance is evaluated. The average AUCs are 0.585 and 0.531 for aLasso-SPR and aLasso-Cox, respectively. Comparing the AUCs using a paired nonparametric test suggests that the difference is significant with p-value 2.2×10^{-16}.

5. Application to the penalized smoothed 0–1 loss

To demonstrate the application of Fabs to other nonconvex loss functions, we consider binary classification with the Lasso penalized 0–1 loss (0–1-Lasso) for which the smoothed 0–1 loss is defined in (3). Alternative methods for comparison include the Lasso penalized SVM (SVM-Lasso, Zhu et al., 2004) and Lasso penalized logistic regression (LR-Lasso). We also include results from unpenalized 0–1 loss (0–1), which is feasible when the sample size is larger than the dimension. We follow the simulation setup in Zhu et al. (2004) and directly copy their numerical results for SVM-Lasso. LR-Lasso is implemented by the R package *glmnet*.

Following Zhu et al. (2004), two classes of equal sample size 50 are generated. In the first class, the two input variables x_1 and x_2 are independently generated from N(0,1). In the second class, an additional constraint is applied such that $4.5 \le x_1^2 + x_2^2 \le 8$. Besides x_1 and x_2, we sequentially augment the inputs with additional two, four, six, and eight standard normal noise inputs. For input variables $\{x_j, j = 1, \ldots, p\}$, the dictionary of basis functions is $\mathscr{D} = \{\sqrt{2}x_j, \sqrt{2}x_j x_{j'}, x_j^2, j, j' = 1, \ldots, p\}$. Let q be the number of basis functions. Classification error is evaluated on a testing dataset of sample size 1000, and the results based on 200 replicates are shown in Table 5.

We observe that 0–1-Lasso and SVM-Lasso outperform LR-Lasso. The 0–1-Lasso has comparable performance as SVM-Lasso for a small to moderate number of basis functions (q = 5, 14, and 27). When the number of basis functions increase further (q = 44 and 65), the 0–1-Lasso outperforms SVM-Lasso. We note that among all the basis functions, only the quadratic basis functions x_1^2 and x_2^2 are truly effective. Therefore, when q increases, noises increase, posing more challenges for classification. The 0–1-Lasso handles the noises well and performs robustly as noises increase. Comparing 0–1-Lasso and 0–1, we can see that without variable selection using Lasso, the non-penalized 0–1 loss is significantly affected by noise inputs. Figure 2 depicts the solution path of 0–1-Lasso generated by the Fabs for q = 5 based on one replicate. We can see that 0–1-Lasso picks up x_1^2 and x_2^2 earlier than LR-Lasso. This confirms Fabs' effectiveness in this nonconvex example.

6. Discussion

For practical data analysis, the development of effective computational algorithms is as important as the development of methodologies and theories. In this study, we have developed the Fabs approach for computing penalized estimation with nonconvex loss functions and the aLasso penalty. Statistical properties of the Fabs paths have been investigated theoretically and numerically. Development in this study complements the existing literature on computing penalized convex loss functions. Another important contribution is to deliver an effective computational solution to the penalized SPR estimation, which has been well developed in the literature but with its computational aspect not carefully studied. Simulation and data analysis show that the Fabs algorithm can generate parameter paths and estimates with satisfactory properties

Beyond our emphasis on the SPR estimation, we have also included a brief application to the smoothed 0–1 loss in binary classification to demonstrate the capability of the Fabs to work with other differentiable nonconvex loss functions. For high-dimensional cases, penalty terms are usually included which bring additional complexity to the problems. The Fabs algorithm, developed for aLasso in this study, can generate solution paths with performance guaranteed to be more efficient than the heuristic algorithms. It may be of interest to have more focused studies on various interesting loss functions in the future.

Our development has been focused on aLasso which is a popular sparsity-inducing penalty. It is of interest to generalize the development to other convex penalties. In the supplementary document, we develop an Fabs algorithm for general convex penalties and illustrate its performance using a data example. However, we have not been able to establish the convergence properties for more general penalties and postpone such research to the future.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

Shi's work was supported by the National Natural Science Foundation of China (Grant No. 71501089). Ma's work was supported by National Institutes of Health (R01 CA204120) and National Bureau of Statistics of China (Grant No. 2016LD01). We are grateful to the editor and two referees for their valuable comments and constructive suggestions, which led to a substantial improvement of this manuscript.

Appendix A

The BLasso algorithm

Algorithm 0

BLasso for the aLasso penalty

Step 1 (Initialization) With a small step size $\varepsilon(>0)$, set $t=0$, and $k = \arg\max_{l=1,\ldots,p} \frac{|\nabla_l L(0)|}{w_l}$ and $\mathscr{A}^0 = \{k\}$, where ∇ denotes the partial derivative;

$\hat{\beta}^0 = -\frac{\text{sign}(\nabla_k L(0))}{w_k} 1_k \varepsilon$, where 1_k is a length-p vector with the kth element being 1 and the rest being 0;

$\lambda^0 = \frac{1}{\varepsilon}\left[L(0) - L(\hat{\beta}^0)\right]$.

Step 2 (Backward and Forward steps)

2.1 Find the backward direction:

$$k = \arg\min_{l \in \mathscr{A}^t} L\left(\hat{\beta}^t - \frac{\text{sign}(\hat{\beta}_l^t)}{w_l} 1_l \varepsilon\right),$$

$$\Delta_k = -\frac{\text{sign}(\hat{\beta}_k^t)}{w_k} 1_k.$$

2.2 If $Q(\hat{\beta}^t + \Delta_k \varepsilon; \lambda^t) < Q(\hat{\beta}^t; \lambda^t)$, take a backward step as $\hat{\beta}^{t+1} = \hat{\beta}^t + \Delta_k \varepsilon$, and $\lambda^{t+1} = \lambda^t$.

Otherwise, force a forward step, and relax λ if necessary:

$$k = \arg\min_{l = 1, \ldots, p} L\left(\hat{\beta}^t - \frac{\text{sign}(\nabla_l L(\hat{\beta}^t))}{w_l} 1_l \varepsilon\right) \text{ and } \mathscr{A}^{t+1} = \mathscr{A}^t \cup k,$$

$$\hat{\beta}^{t+1} = \hat{\beta}^t - \frac{\text{sign}(\nabla_k L(\hat{\beta}^t))}{w_k} 1_k \varepsilon,$$

$$\lambda^{t+1} = \min\left\{\lambda^t, \frac{1}{\varepsilon}[L(\hat{\beta}^t) - L(\hat{\beta}^{t+1})]\right\}.$$

Step 3 (Iteration) Update $t = t + 1$, and repeat Steps 2 and 3 until $\lambda^t \leq \lambda_{\min}$, where $\lambda_{\min}$ is the lower bound of the tuning parameter.

Appendix B

Proof

Proof of Lemma 1

1. When a forward step is taken, for all $l \in \mathscr{A}^t$,

$$Q\left(\hat{\beta}^t - \frac{\text{sign}(\hat{\beta}_l^t)}{w_l} 1_l \varepsilon; \lambda^t\right) \geq Q(\hat{\beta}^t; \lambda^t). \quad \text{(B.1)}$$

A forward step reduces $L(\hat{\beta}^t)$. That is, $L(\hat{\beta}^{t+1}) < L(\hat{\beta}^t)$. If $\hat{\beta}_l^{t+1} = \hat{\beta}_l^t - \frac{\text{sign}(\hat{\beta}_l^t)}{w_l}\varepsilon$, $\rho(\hat{\beta}^{t+1}) < \rho(\hat{\beta}^t)$. Then we have

$$L(\hat{\beta}^{t+1}) + \lambda^t \rho(\hat{\beta}^{t+1}) < L(\hat{\beta}^t) + \lambda^t \rho(\hat{\beta}^t),$$

which contradicts inequality (B.1). Therefore, if a forward step is made by updating $\hat{\beta}_l^t$ where $l \in \mathscr{A}^t$, it can only be made by

$$\hat{\beta}_l^{t+1} = \hat{\beta}_l^t + \frac{\text{sign}(\hat{\beta}_l^t)}{w_l}\varepsilon .$$

2. Without loss of generality, assume that in step $t+1$, the kth coordinate of $\hat{\beta}^t$ is updated by a forward step. Then

$$\hat{\beta}_k^{t+1} = \hat{\beta}_k^t - \frac{\text{sign}(\nabla_k L(\hat{\beta}^t))}{w_k}\varepsilon .$$

If $\hat{\beta}_k^t = 0$, we have $\rho(\hat{\beta}^{t+1}) = \rho(\hat{\beta}^t) + \varepsilon$. If $\hat{\beta}_k^t \neq 0$, we have $\hat{\beta}_k^{t+1} = \hat{\beta}_k^t + \frac{\text{sign}(\hat{\beta}_k^t)}{w_k}\varepsilon$ and $\rho(\hat{\beta}^{t+1}) = \rho(\hat{\beta}^t) + \varepsilon$.

3. In a forward step, $\lambda^{t+1} = \{\lambda^t, \frac{L(\hat{\beta}^t) - L(\hat{\beta}^{t+1})}{\varepsilon}\}$. If $\lambda^{t+1} = \frac{L(\hat{\beta}^t) - L(\hat{\beta}^{t+1})}{\varepsilon}$, then

$$Q(\hat{\beta}^{t+1}; \lambda^{t+1}) = L(\hat{\beta}^{t+1}) + \lambda^{t+1}\varepsilon + \lambda^{t+1}\rho(\hat{\beta}^t) \leq L(\hat{\beta}^{t+1}) + \lambda^{t+1}\varepsilon + \lambda^t\rho(\hat{\beta}^t) = Q(\hat{\beta}^t; \lambda^t) .$$

If $\lambda^{t+1} = \lambda^t$, then $\lambda^t \leq \frac{L(\hat{\beta}^t) - L(\hat{\beta}^{t+1})}{\varepsilon}$. Therefore,

$$\begin{aligned} Q(\hat{\beta}^{t+1}; \lambda^{t+1}) &= L(\hat{\beta}^{t+1}) + \lambda^{t+1}\rho(\hat{\beta}^{t+1}) = L(\hat{\beta}^{t+1}) + \lambda^t\left[\rho(\hat{\beta}^t) + \varepsilon\right] \\ &\leq L(\hat{\beta}^{t+1}) + \lambda^t\rho(\hat{\beta}^t) + \frac{L(\hat{\beta}^t) - L(\hat{\beta}^{t+1})}{\varepsilon}\varepsilon = Q(\hat{\beta}^t; \lambda^t) . \end{aligned}$$

Lemma 2

For any t such that $\hat{\beta}^{t+1}$ is updated by a forward step, if $\lambda^{t+1} = \frac{L(\hat{\beta}^t) - L(\hat{\beta}^{t+1})}{\varepsilon}$, then for all $l \in \mathscr{A}^t$, $Q(\hat{\beta}^t \pm \frac{1_l}{w_l}\varepsilon; \lambda^t) \geq Q(\hat{\beta}^t; \lambda^t)$.

Proof of Lemma 2

If a forward step is taken, from $\lambda^{t+1} = \frac{L(\hat{\beta}^t) - L(\hat{\beta}^{t+1})}{\varepsilon} \leq \lambda^t$, we have

$$L(\hat{\beta}^t) - L(\hat{\beta}^{t+1}) \leq \lambda^t\varepsilon = \lambda^t\left(\rho(\hat{\beta}^{t+1}) - \rho(\hat{\beta}^t)\right). \quad \text{(B.2)}$$

Therefore

$$\begin{aligned}
&L(\hat{\beta}^t) + \lambda^t \rho(\hat{\beta}^t) \le L(\hat{\beta}^{t+1}) + \lambda^t \rho\left(\hat{\beta}^{t+1}\right) = \min_l L\left(\hat{\beta}^t - \frac{\operatorname{sign}(\nabla_l L(\hat{\beta}^t))}{w_l} 1_l \varepsilon\right) + \lambda^t \rho\left(\hat{\beta}^{t+1}\right) \\
&\le \min_{l \in \mathscr{A}^t} L\left(\hat{\beta}^t - \frac{\operatorname{sign}(\nabla_l L(\beta^t))}{w_l} 1_l \varepsilon\right) + \lambda^t \rho\left(\hat{\beta}^{t+1}\right) = \min_{l \in \mathscr{A}^t} L\left(\hat{\beta}^t - \frac{\operatorname{sign}(\hat{\beta}_l^t)}{w_l} 1_l \varepsilon\right) + \lambda^t \rho\left(\hat{\beta}^{t+1}\right) \\
&\le L\left(\hat{\beta}^t + \frac{\operatorname{sign}(\hat{\beta}_l^t)}{w_l} 1_l \varepsilon\right) + \lambda^t \rho\left(\hat{\beta}^t + \frac{\operatorname{sign}(\hat{\beta}_l^t)}{w_l} 1_l \varepsilon\right) = Q\left(\hat{\beta}^t + \frac{\operatorname{sign}(\hat{\beta}_l^t)}{w_l} 1_l \varepsilon; \lambda^t\right).
\end{aligned}$$

Combining this inequality with (B.1), for $l \in \mathscr{A}^t$, we have

$$Q\left(\hat{\beta}^t \pm \frac{\operatorname{sign}(\hat{\beta}_l^t)}{w_l} 1_l \varepsilon; \lambda^t\right) \ge Q(\hat{\beta}^t; \lambda^t). \quad \text{(B.3)}$$

Proof of Theorem 1

We prove this theorem by checking the δ-optimality conditions for problem (4).

For $l \in \mathscr{A}^t$, with Taylor expansion,

$$L(\hat{\beta}^t \pm \frac{\operatorname{sign}(\hat{\beta}_l^t)}{w_l} 1_l \varepsilon) \le L(\hat{\beta}^t) \pm \nabla_l L(\hat{\beta}^t) \frac{\operatorname{sign}(\hat{\beta}_l^t)}{w_l} \varepsilon + \frac{m}{2w_l^2} \varepsilon^2, \quad \text{(B.4)}$$

where m is the upper bound of the second-order partial derivatives of L. Combining (B.3) from Lemma 2 with (B.4), we have

$$|\nabla_l L(\hat{\beta}^t) + \lambda w_l \operatorname{sign}(\hat{\beta}_l^t)| \le \delta, \quad \text{(B.5)}$$

where $\delta = \max_l \left\{\frac{m}{2w_l}\varepsilon\right\} = \frac{m}{2w_*}\varepsilon$. The δ-optimality condition (13) is satisfied.

For $l \notin \mathscr{A}^t$, with Taylor expansion,

$$L\left(\hat{\beta}^t - \frac{\operatorname{sign}(\nabla_l L(\beta^t))}{w_l} 1_l \varepsilon\right) \le L(\hat{\beta}^t) - \nabla_l L(\beta^t) \frac{\operatorname{sign}(\nabla_l L(\beta^t))}{w_l} \varepsilon + \frac{m}{2w_l^2} \varepsilon^2.$$

Combining this inequality with inequality (B.2), we have

$$|\nabla_l L(\hat\beta^t)| \le \frac{L(\hat\beta^t) - L\left(\hat\beta^t - \frac{\operatorname{sign}(\nabla_l L(\beta^t))}{w_l} 1_l \varepsilon\right)}{\varepsilon} w_l + \frac{m}{2w_l}\varepsilon \le \frac{L(\hat\beta^t) - L\left(\hat\beta^{t+1}\right)}{\varepsilon} w_l + \frac{m}{2w_l}\varepsilon \le \lambda^t w_l + \delta .$$

The δ-optimality condition (14) is satisfied.

Proof of Theorem 2

We prove this theorem by induction.

Initial step

$t = 0$.

Let k_c denote the coordinate selected with $c\sigma$ and $c\varepsilon$. By Algorithm 1, $k_c = k$ for $t = 0$. In this case,

$$\hat\beta^0(c\sigma, c\varepsilon) = -\frac{\operatorname{sign}(\nabla_{k_c} L(0; c\sigma))}{w_{k_c}} 1_{k_c} c\varepsilon = -\frac{\operatorname{sign}(\nabla_k L(0; \sigma))}{w_k} 1_k c\varepsilon = c\hat\beta^0(\sigma, \varepsilon) .$$

Then we have

$$\lambda^0(c\sigma, c\varepsilon) = \frac{L(0; c\sigma) - L\left(\hat\beta^0(c\sigma, c\varepsilon); c\sigma\right)}{c\varepsilon} = \frac{L(c0; c\sigma) - L\left(c\hat\beta^0(\sigma, \varepsilon); c\sigma\right)}{c\varepsilon} = \frac{L(0; \sigma) - L\left(\hat\beta^0(\sigma, \varepsilon); \sigma\right)}{c\varepsilon} = \frac{\lambda^0(\sigma, \varepsilon)}{c} .$$

Induction step

$t = t + 1$.

Assume $\hat\beta^t(c\sigma, c\varepsilon) = c\hat\beta^t(\sigma, \varepsilon)$ and $\lambda^t(c\sigma, c\varepsilon) = \lambda^t(\sigma, \varepsilon)/c$. Let $u_{ij}^t(\sigma, \varepsilon) = \frac{(X_i - X_j)^\top \hat\beta^t(\sigma, \varepsilon)}{\sigma}$. Immediately we have $u_{ij}^t(c\sigma, c\varepsilon) = u_{ij}^t(\sigma, \varepsilon)$. By definition, it also holds that $L(\hat\beta^t(c\sigma, c\varepsilon); c\sigma) = L(\hat\beta^t(\sigma, \varepsilon); \sigma)$. Then

$$\nabla_l L\left(\hat\beta^t(c\sigma, c\varepsilon); c\sigma\right) = -\frac{1}{n(n-1)} \sum_{i \ne j}^{n} \delta_j I(y_i \ge y_j) \frac{\exp(-u_{ij}^t(c\sigma, c\varepsilon))}{[1 + \exp(-u_{ij}^t(c\sigma, c\varepsilon))]^2} \frac{X_{il} - X_{jl}}{c\sigma}$$
$$= -\frac{1}{n(n-1)} \sum_{i \ne j}^{n} \delta_j I(y_i \ge y_j) \frac{\exp(-u_{ij}^t(\sigma, \varepsilon))}{[1 + \exp(-u_{ij}^t(\sigma, \varepsilon))]^2} \frac{X_{il} - X_{jl}}{c\sigma} = \nabla_l L\left(\hat\beta^t(\sigma, \varepsilon); \sigma\right)/c .$$

Obviously, the order of $\{\nabla_l L(\hat\beta^t(c\sigma, c\varepsilon); c\sigma), l = 1, \ldots, p\}$ is the same as that of $\{\nabla_l L(\hat\beta^t(\sigma, \varepsilon); \sigma), l = 1, \ldots, p\}$. Therefore, $k_c = k$, regardless of a forward or backward step.

Next we show that if a backward (forward) step is taken for $\hat{\beta}^{t+1}(\sigma, \varepsilon)$, then a backward (forward) step will be taken for $\hat{\beta}^{t+1}(c\sigma, c\varepsilon)$ as well. It is sufficient to show

$$Q\left(\beta^t(\sigma,\varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^t_k(\sigma,\varepsilon))}{w_k} 1_k \varepsilon; \lambda^t(\sigma,\varepsilon)\right) - Q\left(\hat{\beta}^t(\sigma,\varepsilon), \lambda^t(\sigma,\varepsilon)\right) \tag{B.6}$$
$$= Q\left(\hat{\beta}^t(c\sigma,c\varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^t_{k_c}(c\sigma,c\varepsilon))}{w_{k_c}} 1_{k_c} c\varepsilon; \lambda^t(c\sigma,c\varepsilon)\right) - Q\left(\hat{\beta}^t(c\sigma,c\varepsilon), \lambda^t(c\sigma,c\varepsilon)\right).$$

For the loss function,

$$L\left(\hat{\beta}^t(c\sigma,c\varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^t_{k_c}(c\sigma,c\varepsilon))}{w_{k_c}} 1_{k_c} c\varepsilon; c\sigma\right) - L\left(\hat{\beta}^t(c\sigma,c\varepsilon); c\sigma\right) \tag{B.7}$$
$$= L\left(c\hat{\beta}^t(\sigma,\varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^t_k(\sigma,\varepsilon))}{w_k} 1_k c\varepsilon; c\sigma\right) - L\left(\hat{\beta}^t(\sigma,\varepsilon); \sigma\right)$$
$$= L\left(\hat{\beta}^t(\sigma,\varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^t_k(\sigma,\varepsilon))}{w_k} 1_k \varepsilon; \sigma\right) - L\left(\hat{\beta}^t(\sigma,\varepsilon); \sigma\right).$$

For the penalty function,

$$\lambda^t(c\sigma,c\varepsilon)\rho\left(\hat{\beta}^t(c\sigma,c\varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^t_{k_c}(c\sigma,c\varepsilon))}{w_{k_c}} 1_{k_c} c\varepsilon\right) - \lambda^t(c\sigma,c\varepsilon)\rho(\hat{\beta}^t(c\sigma,c\varepsilon)) \tag{B.8}$$
$$= \frac{\lambda^t(\sigma,\varepsilon)}{c}\rho\left(c\hat{\beta}^t(\sigma,\varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^t_k(\sigma,\varepsilon))}{w_k} 1_k c\varepsilon\right) - \frac{\lambda^t(\sigma,\varepsilon)}{c}\rho(c\hat{\beta}^t(\sigma,\varepsilon))$$
$$= \lambda^t(\sigma,\varepsilon)\rho\left(\hat{\beta}^t(\sigma,\varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^t_k(\sigma,\varepsilon))}{w_k} 1_k \varepsilon\right) - \lambda^t(\sigma,\varepsilon)\rho(\hat{\beta}^t(\sigma,\varepsilon)).$$

By (B.7) and (B.8), we conclude that (B.6) holds.

If a backward step is taken, then

$$\begin{aligned}\hat{\beta}^{t+1}(c\sigma, c\varepsilon) &= \hat{\beta}^{t}(c\sigma, c\varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^{t}_{k_c}(c\sigma, c\varepsilon))}{w_{k_c}} 1_{k_c} c\varepsilon = c\hat{\beta}^{t}(\sigma, \varepsilon) - \frac{\operatorname{sign}(c\hat{\beta}^{t}_{k}(\sigma, \varepsilon))}{w_k} 1_{k} c\varepsilon \\ &= c\left[\hat{\beta}^{t}(\sigma, \varepsilon) - \frac{\operatorname{sign}(\hat{\beta}^{t}_{k}(\sigma, \varepsilon))}{w_k} 1_{k}\varepsilon\right] = c\hat{\beta}^{t+1}(\sigma, \varepsilon),\end{aligned}$$

and

$$\lambda^{t+1}(c\sigma, c\varepsilon) = \lambda^{t}(c\sigma, c\varepsilon) = \frac{\lambda^{t}(\sigma, \varepsilon)}{c} = \frac{\lambda^{t+1}(\sigma, \varepsilon)}{c}.$$

If a forward step is taken, then

$$\begin{aligned}\hat{\beta}^{t+1}(c\sigma, c\varepsilon) &= \hat{\beta}^{t}(c\sigma, c\varepsilon) - \frac{\operatorname{sign}\left(\nabla_{k_c} L\left(\hat{\beta}^{t}_{k_c}(c\sigma, c\varepsilon); c\sigma\right)\right)}{w_{k_c}} 1_{k_c} c\varepsilon = c\hat{\beta}^{t}(\sigma, \varepsilon) - \frac{\operatorname{sign}\left(\nabla_{k} L\left(c\hat{\beta}^{t}_{k}(\sigma, \varepsilon); c\sigma\right)\right)}{w_k} 1_{k} c\varepsilon \\ &= c\left[\hat{\beta}^{t}(\sigma, \varepsilon) - \frac{\operatorname{sign}\left(\nabla_{k} L\left(\hat{\beta}^{t}_{k}(\sigma, \varepsilon); \sigma\right)\right)}{w_k} 1_{k}\varepsilon\right] = c\hat{\beta}^{t+1}(\sigma, \varepsilon),\end{aligned}$$

and

$$\begin{aligned}\lambda^{t+1}(c\sigma, c\varepsilon) &= \min\left\{\lambda^{t}(c\sigma, c\varepsilon), \frac{1}{c\varepsilon}\left[L\left(\hat{\beta}^{t}(c\sigma, c\varepsilon); c\sigma\right) - L\left(\hat{\beta}^{t+1}(c\sigma, c\varepsilon); c\sigma\right)\right]\right\} \\ &= \min\left\{\frac{1}{c}\lambda^{t}(\sigma, \varepsilon), \frac{1}{c\varepsilon}\left[L\left(\hat{\beta}^{t}(\sigma, \varepsilon); \sigma\right) - L\left(\hat{\beta}^{t+1}(\sigma, \varepsilon); \sigma\right)\right]\right\} = \frac{1}{c}\min\left\{\lambda^{t}(\sigma, \varepsilon), \frac{1}{\varepsilon}\left[L\left(\hat{\beta}^{t}(\sigma, \varepsilon); \sigma\right) - L\left(\hat{\beta}^{t+1}(\sigma, \varepsilon); \sigma\right)\right]\right\} \\ &= \frac{\lambda^{t+1}(\sigma, \varepsilon)}{c}.\end{aligned}$$

Therefore, by induction, $\hat{\beta}^{t}(c\sigma, c\varepsilon) = c\hat{\beta}^{t}(\sigma, \varepsilon)$ and $\lambda^{t}(c\sigma, c\varepsilon) = \lambda^{t}(\sigma, \varepsilon)/c$ for each t.

References

Becker S, Bobin J, Candès EJ. Nesta: a fast and accurate first-order method for sparse recovery. SIAM Journal on Imaging Sciences. 2011; 4(1):1–39.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning. 2011; 3(1):1–122.

Bühlmann P, van de Geer S. Statistics for high-dimensional data: methods, theory and applications. Springer Science & Business Media; 2011.

Cavanagh C, Sherman RP. Rank estimators for monotonic index models. Journal of Econometrics. 1998; 84(2):351–381.

Efron B, Hastie T, Johnstone I, Tibshirani R, et al. Least angle regression. The Annals of statistics. 2004; 32(2):407–499.

Friedman J, Hastie T, Höfling H, Tibshirani R, et al. Pathwise coordinate optimization. The Annals of Applied Statistics. 2007; 1(2):302–332.

Gasso G, Rakotomamonjy A, Canu S. Recovering sparse signals with a certain family of nonconvex penalties and dc programming. IEEE Transactions on Signal Processing. 2009; 57(12):4686–4698.

Ge R, Huang F, Jin C, Yuan Y. Escaping from saddle points—online stochastic gradient for tensor decomposition. Conference on Learning Theory. 2015:797–842.

Han AK. Non-parametric analysis of a generalized regression model: the maximum rank correlation estimator. Journal of Econometrics. 1987; 35(2–3):303–316.

Hastie T, Taylor J, Tibshirani R, Walther G, et al. Forward stage-wise regression and the monotone lasso. Electronic Journal of Statistics. 2007; 1:1–29.

Hastie T, Tibshirani R, Wainwright M. Statistical learning with sparsity: the lasso and generalizations. CRC Press; 2015.

Lin H, Peng H. Smoothed rank correlation of the linear transformation regression model. Computational Statistics & Data Analysis. 2013; 57(1):615–630.

Nguyen T, Sanner S. Algorithms for direct 0–1 loss optimization in binary classification. ICML. 2013; (3):1085–1093.

Ohta Y, Hamada Y, Saitoh N, Katsuoka K. Effect of the transcriptional repressor mad1 on proliferation of human melanoma cells. Experimental dermatology. 2002; 11(5):439–447. [PubMed: 12366697]

Scrucca L. Ga: A package for genetic algorithms in r. Journal of Statistical Software. 2013; 053(1):1–37.

Sivanandam S, Deepa S. Introduction to genetic algorithms. Springer Science & Business Media; 2007.

Song X, Ma S. Penalised variable selection with u-estimates. Journal of nonparametric statistics. 2010; 22(4):499–515. [PubMed: 21904440]

Song X, Ma S, Huang J, Zhou X-H. A semiparametric approach for the nonparametric transformation survival model with multiple covariates. Biostatistics. 2007; 8(2):197–211. [PubMed: 16670240]

Tibshirani RJ. The lasso problem and uniqueness. Electronic Journal of Statistics. 2013; 7:1456–1490.

Toiyama Y, Mizoguchi A, Kimura K, Hiro J, Inoue Y, Tutumi T, Miki C, Kusunoki M. Ttyh2, a human homologue of the drosophila melanogaster gene tweety, is up-regulated in colon carcinoma and involved in cell proliferation and cell aggregation. World journal of gastroenterology: WJG. 2007; 13(19):2717. [PubMed: 17569141]

Uno H, Cai T, Pencina MJ, D'Agostino RB, Wei L. On the c-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. Statistics in medicine. 2011; 30(10): 1105–1117. [PubMed: 21484848]

Zhang B, Liu X, Chen W, Chen L. Ifit5 potentiates antiviral response through enhancing innate immune signaling pathways. Acta Biochim Biophys Sin. 2013; 45(10):867–874. [PubMed: 23942572]

Zhao P, Yu B. Stagewise lasso. The Journal of Machine Learning Research. 2007; 8:2701–2726.

Zhu J, Rosset S, Tibshirani R, Hastie TJ. 1-norm support vector machines. Advances in Neural Information Processing Systems. 2004; 16:49–56.

Znidar K, Bosnjak M, Cemazar M, Heller LC. Cytosolic dna sensor upregulation accompanies dna electrotransfer in b16. f10 melanoma cells. Molecular Therapy-Nucleic Acids. 2016; 5:e322. [PubMed: 27271988]

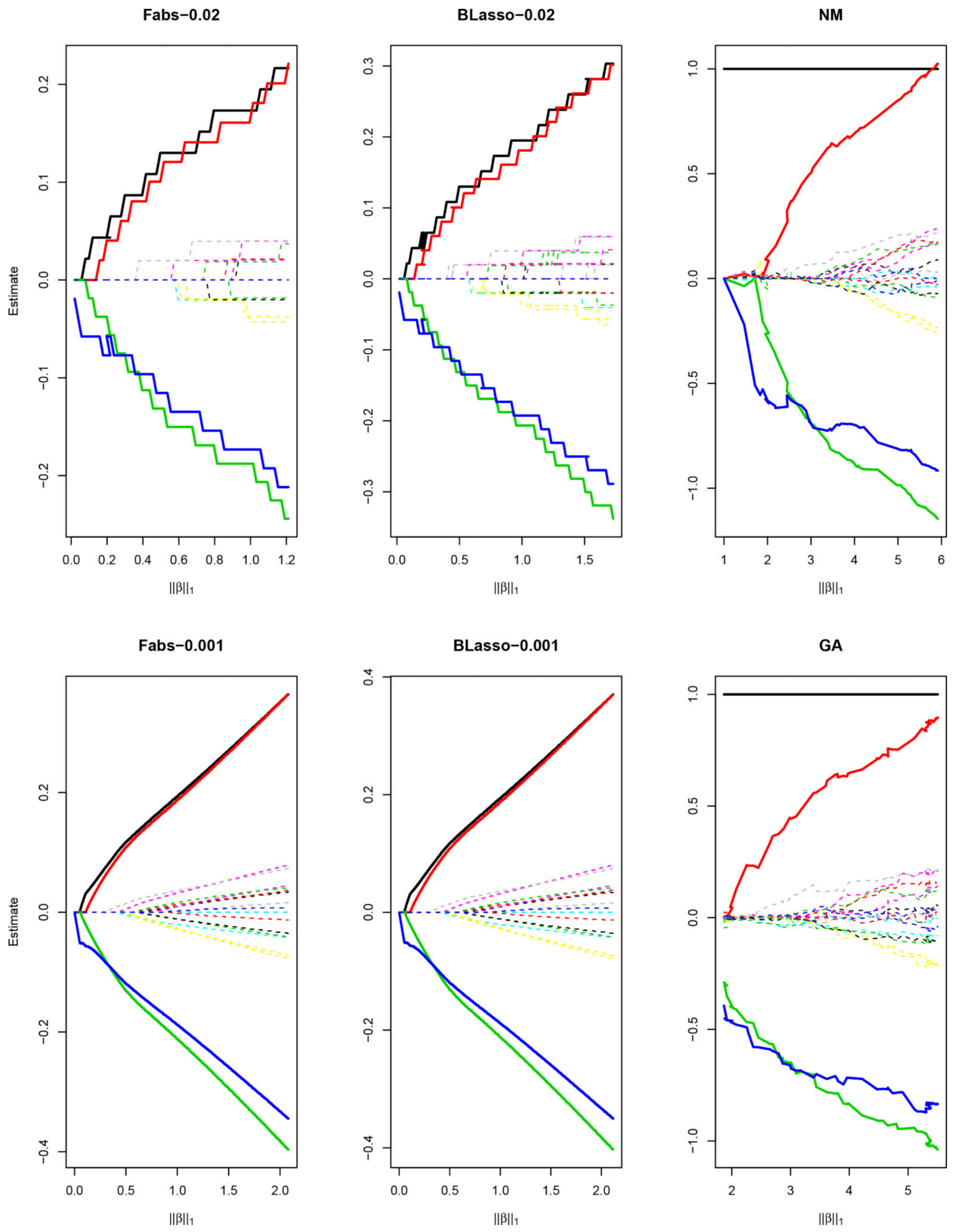

Figure 1.

Solution paths for one simulated replicate under Example 1 with $(n, p) = (200, 20)$ and t_4 random error. Solid/Dashed lines represent variables with truly nonzero/zero coefficients. Left: Fabs with step size 0.02 (top) and 0.001 (bottom). Middle: Blasso with step size 0.02 (top) and 0.001 (bottom). Right: NM (top) and GA (bottom), where the horizontal lines correspond to the anchor variable.

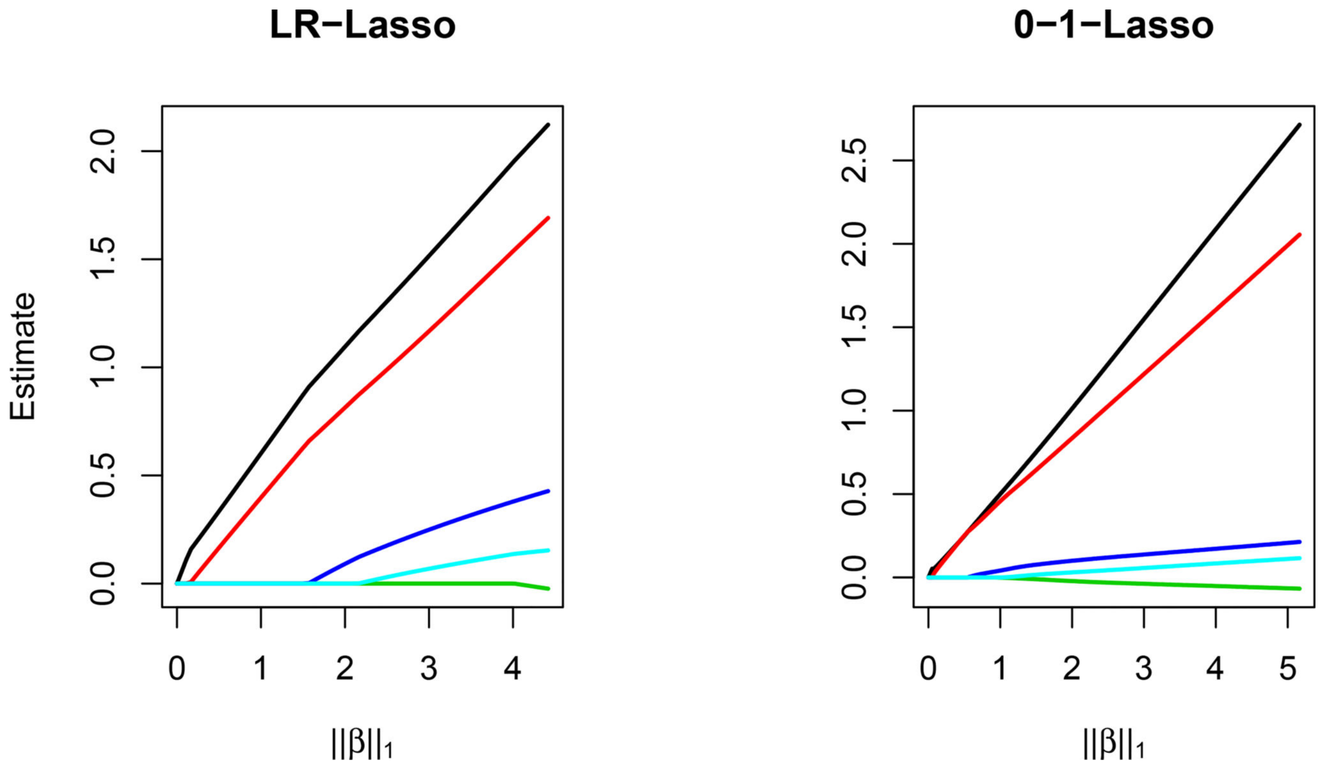

Figure 2.
Solution paths for binary classification when $q = 5$ based on one replicate. The black and red color represent the significant basis functions x_1^2 and x_2^2, respectively. Left: Lasso penalized logistic regression. Right: Lasso penalized 0–1 loss by the Fabs.

Table 1

Simulation Example 1. In each cell, the three rows are the means of PD, Error, and runtime (in seconds).

e	**(*n*, *p*)**	**$\varepsilon = 0.02$**		**$\varepsilon = 0.01$**		**$\varepsilon = 0.001$**		**NM**	**GA**
		Fabs	**BLasso**	**Fabs**	**BLasso**	**Fabs**	**BLasso**		
N(0,1)	(200, 10)	0.702	0.704	0.707	0.706	0.709	0.707	0.715	0.715
		0.014	0.017	0.014	0.017	0.014	0.016	0.017	0.017
		1.75	6.5	3.39	14.27	36.16	149.75	99.73	907.9
	(200, 20)	0.716	0.72	0.723	0.723	0.726	0.725	0.734	0.733
		0.035	0.037	0.034	0.036	0.035	0.037	0.044	0.039
		1.9	19.12	10.96	37.63	53.17	389.77	482.59	1076.45
	(400, 10)	0.698	0.702	0.706	0.707	0.709	0.708	0.715	0.715
		0.008	0.01	0.008	0.008	0.007	0.008	0.009	0.009
		2.92	14.01	7.77	34.08	91.66	390.48	373.24	3395.39
	(400, 20)	0.699	0.705	0.707	0.71	0.712	0.712	0.719	0.719
		0.017	0.019	0.016	0.017	0.016	0.016	0.021	0.021
		4.84	33.27	12.34	82.51	137.15	964.44	1682.4	4077.95
t_4	(200, 10)	0.65	0.65	0.654	0.654	0.656	0.654	0.662	0.662
		0.022	0.025	0.023	0.025	0.022	0.024	0.025	0.025
		1.51	5.51	3.5	13.32	33.19	139.62	91.53	819.73
	(200, 20)	0.65	0.653	0.658	0.658	0.661	0.659	0.668	0.668
		0.049	0.052	0.051	0.053	0.051	0.054	0.063	0.058
		1.82	20.14	8.06	39.04	55.51	392.56	489.37	1070.84
	(400, 10)	0.649	0.654	0.658	0.659	0.661	0.66	0.666	0.666
		0.013	0.014	0.011	0.012	0.01	0.012	0.012	0.012
		4.07	13.28	9.78	33.26	89.11	377.61	358.18	3171.15
	(400, 20)	0.653	0.659	0.661	0.662	0.665	0.664	0.672	0.672
		0.022	0.023	0.02	0.021	0.019	0.02	0.026	0.025

e	**(*n*, *p*)**	***e* = 0.02**		***e* = 0.01**		***e* = 0.001**		**NM**	**GA**
		Fabs	**BLasso**	**Fabs**	**BLasso**	**Fabs**	**BLasso**		
		4.66	33.95	14.05	81.87	128.78	925.51	1758.9	4009.01

Table 2

Simulation Example 2. In each cell, mean (upper) and sd (lower).

	$\varrho = 0.3$								$\varrho = 0.7$							
			$\varepsilon = 0.02$		$\varepsilon = 0.01$		$\varepsilon = 0.001$				$\varepsilon = 0.02$		$\varepsilon = 0.01$		$\varepsilon = 0.001$	
	LS	**LAD**	**BLasso**	**Fabs**	**BLasso**	**Fabs**	**BLasso**	**Fabs**	**LS**	**LAD**	**BLasso**	**Fabs**	**BLasso**	**Fabs**	**BLasso**	**Fabs**
	N(0,1)															
Error	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.15	0.42	0.16	0.13	0.22	0.18	0.29	0.30
	0.01	0.03	0.01	0.01	0.01	0.01	0.02	0.01	0.11	0.15	0.14	0.12	0.19	0.15	0.18	0.19
TPR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.82	0.96	0.98	0.90	0.95	0.88	0.86
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.17	0.11	0.10	0.18	0.13	0.19	0.20
FPR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.7N(0, 1) + 0.3Cauchy(0, 1)															
Error	3.63	0.26	0.04	0.03	0.03	0.03	0.04	0.04	3.67	0.76	0.30	0.29	0.38	0.37	0.42	0.45
	1.12	0.79	0.03	0.02	0.03	0.02	0.04	0.03	1.01	0.84	0.18	0.18	0.20	0.19	0.17	0.16
TPR	0.10	0.92	1.00	1.00	1.00	1.00	1.00	1.00	0.05	0.60	0.85	0.88	0.77	0.78	0.73	0.71
	0.29	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.21	0.19	0.18	0.19	0.20	0.19	0.19
FPR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	t_2															
Error	1.63	0.10	0.05	0.05	0.05	0.06	0.07	0.08	2.38	0.58	0.40	0.44	0.46	0.49	0.53	0.52
	1.81	0.14	0.04	0.04	0.05	0.08	0.06	0.10	1.70	0.10	0.22	0.21	0.19	0.17	0.14	0.15
TPR	0.56	0.98	1.00	1.00	1.00	0.99	1.00	0.99	0.28	0.60	0.76	0.71	0.69	0.66	0.63	0.62
	0.46	0.08	0.00	0.00	0.00	0.04	0.00	0.04	0.30	0.14	0.22	0.21	0.21	0.19	0.17	0.18
FPR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 3

Simulation Example 3. In each cell, mean (upper) and sd (lower).

	$\varrho = 0.3$							$\varrho = 0.7$						
		$e = 0.02$		$e = 0.01$		$e = 0.001$			$e = 0.02$		$e = 0.01$		$e = 0.001$	
	Cox	**BLasso**	**Fabs**	**BLasso**	**Fabs**	**BLasso**	**Fabs**	**Cox**	**BLasso**	**Fabs**	**BLasso**	**Fabs**	**BLasso**	**Fabs**
								EV						
Error	0.04	0.03	0.03	0.03	0.03	0.03	0.03	0.41	0.37	0.35	0.37	0.37	0.39	0.38
	0.05	0.03	0.02	0.02	0.02	0.02	0.02	0.21	0.20	0.21	0.19	0.21	0.20	0.20
TPR	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.80	0.78	0.78	0.78	0.77	0.76	0.76
	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.20	0.21	0.20	0.21	0.21	0.21
FPR	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
								N(0,1)						
Error	0.08	0.02	0.02	0.02	0.02	0.02	0.02	0.46	0.34	0.35	0.35	0.37	0.34	0.37
	0.10	0.02	0.01	0.02	0.01	0.02	0.01	0.21	0.21	0.20	0.21	0.20	0.21	0.20
TPR	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.73	0.78	0.80	0.78	0.77	0.78	0.78
	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.22	0.20	0.21	0.20	0.22	0.20	0.22
FPR	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
								t_4						
Error	0.53	0.05	0.04	0.05	0.04	0.05	0.04	1.32	0.43	0.43	0.43	0.43	0.43	0.43
	0.79	0.05	0.03	0.05	0.03	0.05	0.03	1.32	0.21	0.17	0.21	0.17	0.21	0.16
TPR	0.82	0.99	1.00	0.99	1.00	0.99	1.00	0.46	0.70	0.71	0.70	0.72	0.69	0.71
	0.27	0.04	0.00	0.04	0.00	0.04	0.00	0.29	0.19	0.18	0.19	0.18	0.20	0.18
FPR	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00

Table 4

Analysis of LC and SKCM data.

Variables	aLasso-Cox	aLasso-SPR
LC data		
self-reported annual income (log)	−0.256	0.081
loan amount (log)		−0.028
employment length*number of inquiries		−0.012
verification status*debt to income ratio		−0.005
verification status*number of public record		−0.013
debt to income ratio*credit balance		0.004
number of accounts*account utilization rate		−0.003
SKCM data		
ABCC1	0.047	
ABCC2	0.154	
FBP2	0.054	
INTS1	0.068	
MAD1L1	0.199	−0.005
SHPK	0.147	
SRCAP	0.195	
UBE2O	0.160	
UNC45A	0.001	
CREG1		0.009
DDX60		0.050
IFIT5		0.005
TTYH2		−0.011

Table 5

Simulation for binary classification. In each cell, mean (sd) of classification error.

$p - 2$	q	SVM-Lasso	LR-Lasso	0–1-Lasso	0–1
0	5	0.073 (0.010)	0.081 (0.023)	0.075 (0.012)	0.076 (0.016)
2	14	0.074 (0.014)	0.081 (0.022)	0.075 (0.010)	0.100 (0.025)
4	27	0.074 (0.009)	0.094 (0.043)	0.076 (0.011)	0.115 (0.023)
6	44	0.082 (0.009)	0.087 (0.033)	0.072 (0.013)	0.121 (0.029)
8	65	0.084 (0.011)	0.089 (0.043)	0.076 (0.019)	0.136 (0.033)