Statistics and Its Interface Volume 14 (2021) 339–350

Generalized Newton-Raphson algorithm for high dimensional LASSO regression*

Yueyong Shi, Jian Huang, Yuling Jiao, Yicheng Kang, and Hu Zhang†

The least absolute shrinkage and selection operator (LASSO) penalized regression is a state-of-the-art statistical method in high dimensional data analysis, when the number of predictors exceeds the number of observations. The commonly used Newton-Raphson algorithm is not very successful in solving the non-smooth optimization in LASSO. In this paper, we propose a fast generalized Newton-Raphson (GNR) algorithm for LASSO-type problems. The proposed algorithm, derived from a suitable Karush-Kuhn-Tucker (KKT) conditions based on generalized Newton derivatives, is a non-smooth Newton-type method. We first establish the local one-step convergence of GNR and then show that it is very efficient and accurate when coupled with a constinuation strategy. We also develop a novel parameter selection method. Numerical studies of simulated and real data analysis suggest that the GNR algorithm, with better (or comparable) accuracy, is faster than the algorithm implemented in the popular glmnet package.

AMS 2000 subject classifications: Primary 62F12; secondary 62J05, 62J07.
Keywords and phrases: LASSO, Generalized Newton-Raphson, Continuation, Local one-step convergence, Voting.

1. INTRODUCTION

Methods for high-dimensional regression and variable selection have been applied in many scientific fields, particularly in high-throughput biomedical studies [3, 4, 5]. Among these methods, LASSO regression [41] is favored because of its sparsity and convexity properties [18, 6, 35].

In the setup of LASSO regression, one considers the linear model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\varepsilon}, \tag{1}$$

where $\mathbf{y}$ is an $n \times 1$ response vector, $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_p)$ is an $n \times p$ design matrix, $\boldsymbol{\varepsilon}$ is an $n \times 1$ error (noise) vector, and $\boldsymbol{\beta}^* = (\beta_1^*, \ldots, \beta_p^*)^\top \in \mathbb{R}^p$ is the underlying regression coefficient vector. We assume that $\boldsymbol{\beta}^*$ is sparse in the sense that only a small portion of $\boldsymbol{\beta}^*$ are nonzero. We focus on the high dimensional case where $p > n$. Our goal is to reconstruct the unknown vector $\boldsymbol{\beta}^*$. To use sparsity, LASSO is formulated as the following optimization problem:

$$\widehat{\boldsymbol{\beta}} \triangleq \widehat{\boldsymbol{\beta}}(\lambda) := \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \left\{ Q_\lambda(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \right\}, \tag{2}$$

where $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^p |\beta_j|$ denotes the ℓ_1 norm of the coefficient vector, and $\lambda > 0$ is a tuning (or regularization) parameter. The problem (2) is called the LASSO penalized least squares (PLS), and is also known as "basis pursuit denoising" in the signal processing literature [8].

Theoretically, under certain regularity conditions on the design matrix $\mathbf{X}$ (e.g., the restricted isometry property [7], the strong irrepresentable condition [30, 53]) and the sparsity condition on the regression coefficients, LASSO enjoys certain attractive statistical properties. Computationally, since (2) is a convex optimization problem, the typical algorithms such as Homotopy [31, 13, 12] and Gauss-Seidel type coordinate descent (CD) [17, 43, 15, 50, 29, 44, 16, 42] can be applied for computing the solution of (2). In the literature, many existing methods (e.g., [15, 43, 37, 51]) only ensure the convergence and sublinear convergence rate of the sequence of the objective functions $\{Q_\lambda(\boldsymbol{\beta}^k), k = 1, 2, \ldots\}$, but they have not established the convergence rate of the sequence of the solutions given by $\{\boldsymbol{\beta}^k, k = 1, 2, \ldots\}$.

To avoid such a limitation, we propose a local one step convergent algorithm for (2). In particular, we first show that the global minimizer of (2) is equivalent to roots of some nonsmooth Karush-Kuhn-Tucker (KKT) equations. Then, we solve the non-smooth KKT equations using a generalized Newton-Raphson (GNR) algorithm [28, 34, 23]. We derive the local one step convergence property of the proposed algorithm. We show that it enhances the local superlinear convergence results given by the literature [28, 34, 23]. The computational cost of each iteration in the GNR algorithm is at most $O(np)$. It is the same as that given by the coordinate descent algorithms. If it is warm-started [24, 25, 39, 40], the GNR algorithm usually converges after only a few iterations. Consequently, for a given λ, our GNR algorithm for (2) has overall computational cost $O(np)$, implying that it is efficient in computing the whole solution path of (2).

* Yueyong Shi is supported by the National Natural Science Foundation of China (Grant Nos. 11801531 and 11701571), Yuling Jiao is supported by the National Natural Science Foundation of China (Grant No. 11871474), and Hu Zhang is supported by the National Social Science Fund of China (Grant No. 17BTJ017).
†Corresponding author.

The remainder of this paper is organized as follows. In Section 2, we introduce the GNR algorithm and study its one step convergence property and computational complexity. We also propose our tuning parameter selection method. In Section 3, we compare numerical properties of our algorithm with that given by the glmnet package [16]. Some concluding remarks are provided in Section 4 and technical details are given in the appendix.

2. GENERALIZED NEWTON-RAPHSON ALGORITHM

Classical Newton-Raphson algorithm is an important numerical scheme for solving the maximum log-likelihood estimator (MLE) via finding the root of the gradient of MLE. However, Newton-Raphson is not applicable to the LASSO optimization in (2) since it is not differentiable. In the following, we turn to study (3), the equivalent dual problem of (2), whose solution can be further characterized via a nonsmooth equations (4). We propose a generalized Newton-Raphson method to find the root of equations (4) based on techniques in nonsmooth analysis [28, 34].

2.1 Methodology

We first transform (2) into its equivalent dual formulation.

Lemma 1. *The dual problem of the primal problem (2) is*

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \langle \boldsymbol{\beta}, G\boldsymbol{\beta}\rangle \quad s.t. \quad L \le G\boldsymbol{\beta} \le U, \tag{3}$$

where $G = \mathbf{X}^\top\mathbf{X}$, $L = \widetilde{\mathbf{y}} - \lambda\mathbf{1}$, $U = \widetilde{\mathbf{y}} + \lambda\mathbf{1}$, $\widetilde{\mathbf{y}} = \mathbf{X}^\top\mathbf{y}$ *and* $\mathbf{1} = (1,\dots,1)^\top \in \mathbb{R}^p$.

By Lemma 1, we immediately have (2)$\Leftrightarrow$ (3). Next, we derive the KKT condition of (3). For convenience, we denote $c(\boldsymbol{\beta}) = (G\boldsymbol{\beta} - U, L - G\boldsymbol{\beta})^\top : \mathbb{R}^p \to \mathbb{R}^{2p}$ and $\widetilde{G} = (G, -G)$.

Theorem 1. *Let* $\overline{\boldsymbol{\beta}}$ *be a minimizer of (3), then there exists* $\overline{\mu} \in \mathbb{R}^{2p}$ *such that*

$$\begin{cases} 2G\overline{\boldsymbol{\beta}} + \widetilde{G}\overline{\mu} = 0, \\ \overline{\mu} = P_{\mathbb{R}^{2p}_+}(\overline{\mu} + c(\overline{\boldsymbol{\beta}})), \end{cases} \tag{4}$$

where $P_{\mathbb{R}^{2p}_+}(\cdot)$ *is the pointwise projection onto* $\mathbb{R}_+$, *and* $\mathbb{R}_+ = [0,\infty)$. *Conversely, if there exist* $\overline{\boldsymbol{\beta}} \in \mathbb{R}^p$ *and* $\overline{\mu} \in \mathbb{R}^{2p}$ *such that (4) holds, then* $\overline{\boldsymbol{\beta}}$ *is a minimizer of (3) or (2).*

Let

$$F(\boldsymbol{\beta},\mu) = \begin{pmatrix} 2G\boldsymbol{\beta} + \widetilde{G}\mu \\ \mu - P_{\mathbb{R}^{2p}_+}(\mu + c(\boldsymbol{\beta})) \end{pmatrix} : \mathbb{R}^p \times \mathbb{R}^{2p} \to \mathbb{R}^p \times \mathbb{R}^{2p}. \tag{5}$$

Theorem 1 implies that finding a (global) minimizer of (3) is equivalent to finding a root of $F(\boldsymbol{\beta},\mu)$. However, the classical Newton-Raphson method cannot be used here since the projection operator $P_{\mathbb{R}^{2p}_+}(\cdot)$ is not Frechét differentiable. Fortunately, we can resort to the generalized Newton-Raphson method.

Recall the concept of generalized derivative and the generalized Newton-Raphson method. Let $g : \mathbb{R}^m \to \mathbb{R}^l$ be a Lipschitz (locally Lipschitz) function. By Rademacher's theorem, g is differentiable almost everywhere with respect to the Lebesgue measure. Let D_g be the set of differentiable points of g.

Definition 1. Let $x \in \mathbb{R}^m$. The generalized derivative [34] of g at x is defined as $\partial g(x) = \mathrm{co}\{\lim_{x_i \to 0, x_i \in D_g} \nabla g(x_i)\}$, where $\mathrm{co}(A)$ is the convex hull of A, i.e, the smallest convex set that contains set A. The generalized Newton-Raphson method [28, 34] for solving $g(x) = 0$ can be defined by $x^{k+1} = x^k - H_k^{-1} g(x^k)$, where $H_k \in \partial g(x^k)$.

Lemma 2. *Let* $g(x) = P_{[a,b]}(x) : \mathbb{R}^1 \to \mathbb{R}^1$ *with* $a < b$. *Then*

$$\partial g(x) = \begin{cases} \{0\}, & x < a \text{ or } x > b, \\ \{1\}, & a < x < b, \\ [0,1], & x = b \text{ or } x = a. \end{cases}$$

Moreover, $H(\boldsymbol{\beta},\mu) = P_{\mathbb{R}^{2p}_+}(\mu + c(\boldsymbol{\beta})) : \mathbb{R}^p \times \mathbb{R}^{2p} \to R^{2p}$ *is generalized derivatiable everywhere, with* $\partial H(\boldsymbol{\beta},\mu) = [-\Lambda\widetilde{G}, \mathbf{I} - \Lambda]$, *where* $\Lambda = diag(\partial P_{\mathbb{R}^1_+}(\mu_i + c_i(\boldsymbol{\beta})))$.

We plot the projection function $g(x)$ and the set value mapping (the generalized derivative of g) $\partial g(x)$ with $x \in [-2,8]$, $a = 2$ and $b = 6$ in Figure 1. Although $g(x)$ is not differentiable, we can compute its generalized derivative $\partial g(x)$ by Definition 1. Based on this simple result and the chain rule, we can compute the generalized Jacobian matrix of $F(\boldsymbol{\beta},\mu)$ in (5).

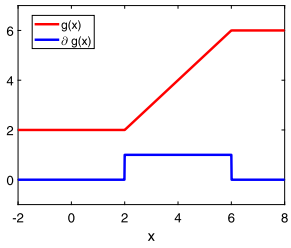

Figure 1. Plot for $g(x)$ *and* $\partial g(x)$.

Let $\mu = (\eta^\top, \gamma^\top)^\top$, $\eta \in \mathbb{R}^p$, $\gamma \in \mathbb{R}^p$, $\mathcal{S} = \{1,\cdots,p\}$, $\mathcal{A}_1 = \{j | \eta_j + (G\boldsymbol{\beta} - U)_j \le 0\}$, $\mathcal{I}_1 = \mathcal{S}/\mathcal{A}_1$, $\mathcal{A}_2 = \{j | \gamma_j + (G\boldsymbol{\beta} - U)_j \le 0\}$, and $\mathcal{I}_2 = \mathcal{S}/\mathcal{A}_2$. Recall $G = \mathbf{X}^\top\mathbf{X}$, $U = \mathbf{X}^\top\mathbf{y} + \lambda\mathbf{1}$ and $L = \mathbf{X}^\top\mathbf{y} - \lambda\mathbf{1}$, we get $\mathcal{A}_1 = \{j | -\eta_j + (\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}))_j \ge -\lambda\}$ and $\mathcal{A}_2 = \{j | \gamma_j + (\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}))_j \le \lambda\}$.

Theorem 2. *If we reformulate $F(\boldsymbol{\beta},\mu)$ in (5) according to $\mathcal{A}_1,\mathcal{I}_1,\mathcal{A}_2$ and $\mathcal{I}_2$ as*

$$F(\boldsymbol{\beta},\eta_{\mathcal{A}_1},\eta_{\mathcal{I}_1},\gamma_{\mathcal{A}_2},\gamma_{\mathcal{I}_2})=\begin{pmatrix}\eta_{\mathcal{A}_1}\\ -G_{\mathcal{I}_1}\boldsymbol{\beta}+U_{\mathcal{I}_1}\\ \gamma_{\mathcal{A}_2}\\ G_{\mathcal{I}_2}\boldsymbol{\beta}-L_{\mathcal{I}_2}\end{pmatrix},$$

then we have

$$\partial F(\boldsymbol{\beta},\mu)=\begin{pmatrix}2G & \widetilde{G}\\ A & B\end{pmatrix},$$

with

$$A=\begin{pmatrix}0 & 0\\ -G_{\mathcal{I}_1\mathcal{A}_1} & -G_{\mathcal{I}_1\mathcal{I}_1}\\ 0 & 0\\ G_{\mathcal{I}_2\mathcal{A}_2} & -G_{\mathcal{I}_2\mathcal{I}_2}\end{pmatrix} \text{ and } B=\begin{pmatrix}\mathbf{I}_{\mathcal{A}_1} & & & \\ & \mathbf{0} & & \\ & & \mathbf{I}_{\mathcal{A}_2} & \\ & & & \mathbf{0}\end{pmatrix}.$$

Now, we state the proposed GNR in Algorithm 1 for solving $F(\boldsymbol{\beta},\mu)=0$, which is also an algorithm for minimizing (2) by Theorem 1.

Algorithm 1 GNR for solving $F(\boldsymbol{\beta},\mu)=0$

Input: $\mathbf{X}$, $\mathbf{y}$, λ and initial guesses $\boldsymbol{\beta}^0$ and $\mu^0=(\eta_0^\top,\gamma_0^\top)^\top$. Set $k=0$.

1: **for** $k=0,1,2,\cdots$ **do**
2: Choose $H_k\in\partial F(\boldsymbol{\beta}^k,\mu^k)$.
3: Obtain $\Delta^k=(\Delta\boldsymbol{\beta}^k,\Delta\mu^k)^\top$ by solving

$$H_k\Delta^k=-F(\boldsymbol{\beta}^k,\mu^k).$$

4: Update

$$\boldsymbol{\beta}^{k+1}=\boldsymbol{\beta}^k+\Delta\boldsymbol{\beta}^k,\mu^{k+1}=\mu^k+\Delta\mu^k.$$

5: Check the stopping rule.
6: **if** stop **then**
7: Denote the last iteration by $\widehat{\boldsymbol{\beta}}$, $\widehat{\mu}$.
8: **else**
9: $k=k+1$
10: **end if**
11: **end for**

Output: $(\widehat{\boldsymbol{\beta}},\widehat{\mu})$, as an estimate of the roots of $F(\boldsymbol{\beta},\mu)=0$.

The key idea of the proposed generalized Newton-Raphson algorithm is using generalized Newton iteration $(\boldsymbol{\beta}^{k+1},\mu^{k+1})=(\boldsymbol{\beta}^k,\mu^k)-H_k^{-1}F((\boldsymbol{\beta}^k,\mu^k))$, where $H_k\in\partial F((\boldsymbol{\beta}^k,\mu^k))$, to find a root, say $(\widehat{\boldsymbol{\beta}},\widehat{\mu})$ of $F(\boldsymbol{\beta},\mu)$ in (5). According to Theorem 1, $\widehat{\boldsymbol{\beta}}$ is the minimizer of the LASSO problem (2).

The main step of the GNR lives in getting Δ^k by solving $H_k\Delta^k=-F(\boldsymbol{\beta}^k,\mu^k)$ and updating

$$\begin{pmatrix}\boldsymbol{\beta}^{k+1}\\ \mu^{k+1}\end{pmatrix}=\begin{pmatrix}\boldsymbol{\beta}^k\\ \mu^k\end{pmatrix}+\Delta^k. \tag{6}$$

Next we examine these steps in details.

By Theorem 2 and the above discussion, $H_k\Delta^k=-F(\boldsymbol{\beta}^k,\mu^k)$ reads

$$\begin{pmatrix}2G & \widetilde{G}\\ A^k & B^k\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{\beta}^k\\ \Delta\eta^k_{\mathcal{A}_1^k}\\ \Delta\eta^k_{\mathcal{I}_1^k}\\ \Delta\gamma^k_{\mathcal{A}_2^k}\\ \Delta\gamma^k_{\mathcal{I}_2^k}\end{pmatrix}=\begin{pmatrix}-2G\boldsymbol{\beta}^k\\ \eta^k_{\mathcal{A}_1^k}\\ -G_{\mathcal{I}_1^k}\boldsymbol{\beta}^k\\ \gamma^k_{\mathcal{A}_2^k}\\ -G_{\mathcal{I}_2^k}\boldsymbol{\beta}^k\end{pmatrix}, \tag{7}$$

where $\mathcal{A}_1^k=\{j|-\eta_j^k+(\mathbf{X}^\top(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}^k))_j\geq-\lambda\}$, $\mathcal{A}_2^k=\{j|\gamma_j^k+(\mathbf{X}^\top(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}^k))_j\leq\lambda\}$, $\mathcal{I}_1^k=\mathcal{S}/\mathcal{A}_1^k$, $\mathcal{I}_2^k=\mathcal{S}/\mathcal{A}_2^k$,

$$A^k=\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ -G_{\mathcal{I}_1^k\mathcal{A}_1^k} & -G_{\mathcal{I}_1^k\mathcal{I}_1^k}\\ \mathbf{0} & \mathbf{0}\\ G_{\mathcal{I}_2^k\mathcal{A}_2^k} & -G_{\mathcal{I}_2^k\mathcal{I}_2^k}\end{pmatrix}, \text{ and } B^k=\begin{pmatrix}\mathbf{I}_{\mathcal{A}_1^k} & & & \\ & \mathbf{0} & & \\ & & \mathbf{I}_{\mathcal{A}_1^k} & \\ & & & \mathbf{0}\end{pmatrix}.$$

It is noteworthy that (7) is equivalent to

$$\begin{cases}2G(\boldsymbol{\beta}^k+\Delta\boldsymbol{\beta}^k)+G(\eta^k+\Delta\eta^k)-G(\gamma^k+\Delta\gamma^k)=\mathbf{0},\\ \eta^k_{\mathcal{A}_1^k}+\Delta\eta^k_{\mathcal{A}_1^k}=\mathbf{0},\\ G_{\mathcal{I}_1^k}(\beta^k+\Delta\boldsymbol{\beta}^k)=U_{\mathcal{I}_1^k},\\ \gamma^k_{\mathcal{A}_2^k}+\Delta\gamma^k_{\mathcal{A}_2^k}=\mathbf{0},\\ G_{\mathcal{I}_2^k}(\beta^k+\Delta\boldsymbol{\beta}^k)=L_{\mathcal{I}_2^k},\end{cases} \tag{8}$$

and (6) is equivalent to

$$\begin{pmatrix}\boldsymbol{\beta}^{k+1}\\ \eta^k_{\mathcal{A}_1^k}\\ \eta^{k+1}_{\mathcal{I}_1^k}\\ \gamma^{k+1}_{\mathcal{A}_2^k}\\ \gamma^{k+1}_{\mathcal{I}_2^k}\end{pmatrix}=\begin{pmatrix}\boldsymbol{\beta}^k\\ \eta^k_{\mathcal{A}_1^k}\\ \eta^k_{\mathcal{I}_1^k}\\ \gamma^k_{\mathcal{A}_2^k}\\ \gamma^k_{\mathcal{I}_2^k}\end{pmatrix}+\begin{pmatrix}\Delta\boldsymbol{\beta}^k\\ \Delta\eta^k_{\mathcal{A}_1^k}\\ \Delta\eta^k_{\mathcal{I}_1^k}\\ \Delta\gamma^k_{\mathcal{A}_2^k}\\ \Delta\gamma^k_{\mathcal{I}_2^k}\end{pmatrix}. \tag{9}$$

Substituting (9) into (8), we get

$$\begin{cases}2G\boldsymbol{\beta}^{k+1}+G\eta^{k+1}-G\gamma^{k+1}=\mathbf{0},\\ \eta^{k+1}_{\mathcal{A}_1^k}=\mathbf{0},\\ G_{\mathcal{I}_1^k}\boldsymbol{\beta}^{k+1}=U_{\mathcal{I}_1^k},\\ \gamma^{k+1}_{\mathcal{A}_2^k}=\mathbf{0},\\ G_{\mathcal{I}_2^k}\boldsymbol{\beta}^{k+1}=L_{\mathcal{I}_2^k}.\end{cases} \tag{10}$$

We obtain

$$\begin{cases}\eta^{k+1}=-\boldsymbol{\beta}^{k+1}, & \gamma^{k+1}=\boldsymbol{\beta}^{k+1},\\ \boldsymbol{\beta}^{k+1}_{\mathcal{A}^k}=\mathbf{0}, & \mathcal{A}^k=\mathcal{A}_1^k\cap\mathcal{A}_2^k,\\ G_{\mathcal{I}^k\mathcal{I}^k}\boldsymbol{\beta}^{k+1}_{\mathcal{I}^k}=\begin{pmatrix}U_{\mathcal{I}_1^k}\\ L_{\mathcal{I}_2^k}\end{pmatrix}, & \mathcal{I}^k=\mathcal{I}_1^k\cup\mathcal{I}_2^k=(\mathcal{A}^k)^c,\end{cases} \tag{11}$$

which satisfies (10) for each k.

We show that (11) is well defined. Observing that

$$\begin{cases} \mathcal{A}_1^k = \{j|\beta_j^k + (\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^k))_j \geq -\lambda\}, \quad \mathcal{I}_1^k = \mathcal{S}/\mathcal{A}_1^k, & (12)\\ \mathcal{A}_2^k = \{j|\beta_j^k + (\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^k))_j \leq \lambda\}, \quad \mathcal{I}_2^k = \mathcal{S}/\mathcal{A}_2^k, & (13)\\ \mathcal{A}^k = \mathcal{A}_1^k \cap \mathcal{A}_2^k, & (14)\\ \mathcal{I}^k = \mathcal{I}_1^k \cup \mathcal{I}_2^k = (\mathcal{A}^k)^c, & (15) \end{cases}$$

if $|\mathcal{I}^k|$ is small such that $\mathbf{X}_{\mathcal{I}^k}^\top \mathbf{X}_{\mathcal{I}^k}$ is invertible, then (11) is well defined. From the above details, we summarize the GNR algorithm for (2) in Algorithm 2.

Algorithm 2 GNR for (2)

Input: $\mathbf{X}$, $\mathbf{y}$, λ, $J \in \mathbb{N}$ and initial guess $\boldsymbol{\beta}^0$. Set $k = 0$.

1: Pre-compute $\widetilde{\mathbf{y}} = \mathbf{X}^\top \mathbf{y}$ and store it.
2: **for** $k = 0, 1, 2, \cdots$ **do**
3: Compute $\mathcal{A}^k$ and $\mathcal{I}^k$ by (12)-(15).
4: Compute

$$\boldsymbol{\beta}_{\mathcal{A}^k}^{k+1} = \mathbf{0}, \quad \boldsymbol{\beta}_{\mathcal{I}^k}^{k+1} = G_{\mathcal{I}^k\mathcal{I}^k}^{-1} \begin{pmatrix} \widetilde{\mathbf{y}}_{\mathcal{I}_1^k} + \lambda \mathbf{1}_{\mathcal{I}_1^k} \\ \widetilde{\mathbf{y}}_{\mathcal{I}_2^k} - \lambda \mathbf{1}_{\mathcal{I}_2^k} \end{pmatrix}.$$

5: Check the stopping rule: either $\mathcal{A}_k = \mathcal{A}_{k+1}$ or $k > J$.
6: **if** stop **then**
7: Denote the last iteration by $\widehat{\boldsymbol{\beta}}(\lambda)$.
8: **else**
9: $k = k + 1$
10: **end if**
11: **end for**

Output: $\widehat{\boldsymbol{\beta}}(\lambda)$, the estimate of $\boldsymbol{\beta}$ in (2).

2.2 Convergence analysis

In this subsection, we will prove the local one step convergence of GNR algorithm 2, which enhances the local superlinear convergence rate of generalized Newton-Raphson method [28, 34, 9, 23].

Theorem 3. *Let $\widehat{\boldsymbol{\beta}}$ be the global minimizer of (2) and $\widehat{\mathbf{d}} = \mathbf{X}^\top(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})$. Denote $\mathcal{I} = \{j|\widehat{\mathbf{d}}_j| \geq \lambda\}$. If $\mathbf{X}_{\mathcal{I}}$ is of full column rank, and $\boldsymbol{\beta}^0$ is close enough to $\widehat{\boldsymbol{\beta}}$ in the sense that $\|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_\infty + \|\widehat{\mathbf{d}} - \mathbf{d}^0\|_\infty \leq C_\lambda$, where C_λ is defined in (28), then $\boldsymbol{\beta}^1 = \widehat{\boldsymbol{\beta}}$.*

2.3 Complexity analysis

We consider the computational complexity of GNR in Algorithm 2 by examining the number of floating point operations per iteration. It takes $O(np)$ flops in step 3 in Algorithm 2. Here $b = O(a)$ means $b \leq Ca$ for absolute constant C that does not depend on parameters of interest. For step 4, inverting the positive definite matrix $G_{\mathcal{I}^k\mathcal{I}^k}$ by Cholesky factorization needs $O(|\mathcal{I}^k|^3)$ flops. The overall cost per iteration for Algorithm 2 is $O(\max(|A_k|^3, pn))$. Numerically, $|A_k|$ usually increases and converges to $O(\|\boldsymbol{\beta}^*\|_0)$ if the algorithm is warm started; see Section 2.4 for details. Therefore, if the underling solution is sufficiently sparse such that $\|\boldsymbol{\beta}^*\|_0^3 \leq O(np)$ then it takes $O(np)$ flops per iteration for Algorithm 2. The computational cost of the papular CD algorithms [17, 15, 50, 29] for (2) is also $O(np)$ per iteration. Hence the overall cost of Algorithm 2 to compute the solution path with warmstarting will be cheaper than that of CD since Algorithm 2 enjoys superlinear convergence while the convergence rate of CD is sublinear. This is supported by the numerical results presented in Section 3.

2.4 Continuation and selection on tuning parameter

To solve (2) successfully by Algorithm 2, we need to provide a good initial guess for $\boldsymbol{\beta}^0$ and specify the choice of the tuning parameter λ. We solve the two issues separately below.

The proposed GNR is a nonsmooth Newton type method with fast local convergence rate [34, 23]. An important advantage of the GNR is that it converges after one-step iteration if the initial value is good enough. To fully exploit the fast local convergence, we adopt a continuation (warmstarting) strategy [24, 26, 25, 39, 21, 40] to achieve a good initial guess. Specifically, we define $\lambda_s = \lambda_0 \rho^s$ with $\rho \in (0, 1)$ for $s = 1, 2, \ldots, M$, where λ_0 is selected such that $\widehat{\boldsymbol{\beta}} = 0$ whenever $\lambda \geq \lambda_0$ in (2), ρ is the decreasing factor and M is a given positive integer denoting the number of grid points for λ. Then we apply Algorithm 2 on the decreasing sequence $\{\lambda_s\}_s$ by solving the λ_s-problem initialized with the solution of λ_{s-1} problem. In particular, we can set $\lambda_0 = \|\mathbf{X}^\top \mathbf{y}\|_\infty$ for (2) after some simple algebra. The GNR with continuation is denoted as GNRC, and it is summarized in Algorithm 3. In practice, we let $\lambda_{\max} = \lambda_0$ and $\lambda_{\min} = \lambda_M = 1e - 8\lambda_{\max}$, and then divide the interval $[\lambda_{\min}, \lambda_{\max}]$ into M equally distributed subintervals in the logarithmic scale. Numerically, ρ is determined by M. Clearly, a large M implies a large ρ.

Algorithm 3 GNRC

Input: $\lambda_0 = \|\mathbf{X}^\top \mathbf{y}\|_\infty$, $\widehat{\boldsymbol{\beta}}(\lambda_0) = \mathbf{0}$, $\rho \in (0, 1)$ and $M \in \mathbb{N}$.

1: **for** $s = 1, 2, \cdots, M$ **do**
2: Set $\lambda_s = \lambda_0 \rho^s$ and $\boldsymbol{\beta}^0 = \widehat{\boldsymbol{\beta}}(\lambda_{s-1})$.
3: Get $\widehat{\boldsymbol{\beta}}(\lambda_s)$ by Algorithm 2 with $\widehat{\boldsymbol{\beta}}(\lambda_s) \leftarrow \text{GNR}(\boldsymbol{\beta}^0)$.
4: Check the stopping rule.
5: **end for**

Output: Solution path $\{\widehat{\boldsymbol{\beta}}(\lambda_s)\}_{s=1,2,\ldots}$.

The selection of λ in (2) is an important issue, since it compromises the tradeoff between the data fidelity and the sparsity level of the solution. We refer readers to [46, 45, 48] and references therein for more details about tuning parameter selection. In this paper, we recommend using a novel voting criterion (VC) [20, 21] to select a proper λ in the high dimensional setting. Specifically, we run the GNRC on the sequence $\{\lambda_s\}_{s=1}^M$ to yield a solution path until, for instance, $\|\widehat{\boldsymbol{\beta}}(\lambda_s)\|_0 > \lfloor n/\log(p) \rfloor$ for some $s = S$, where $S \leq M$. Let

$$\Lambda_\ell = \{\lambda_s : \|\widehat{\boldsymbol{\beta}}(\lambda_s)\|_0 = \ell, s = 1, 2, \ldots, S\}$$

be the set of tuning parameters at which the output of GNR has ℓ nonzero elements, where $\ell = 1, \ldots, \lfloor n/\log(p) \rfloor$. Then we determine $\widehat{\lambda}$ by VC, i.e.,

$$\widehat{\lambda} = \max\{\Lambda_{\overline{\ell}}\}, \quad \text{where} \quad \overline{\ell} = \arg\max_{\ell} \{|\Lambda_\ell|\}. \tag{16}$$

3. NUMERICAL EXAMPLES

In this section, we numerically evaluate the performance of the proposed GNRC algorithm (3) and the VC tuning parameter selector (16). All experiments are performed in MATLAB R2010b on a quad-core laptop with an Intel Core i5 CPU (2.60 GHz) and 8 GB RAM running Windows 8.1 (64 bit).

3.1 Comparison with glmnet

There are generally three ways of solving LASSO in (2): the least angle regression (LARS) [13], the CD algorithms, and the proximal methods [1]; see [35] for more details. Among the LASSO solvers, the glmnet by [16] is a well-known, widely used and highly efficient implementation of the standard CD algorithm [15], which is particularly popular in statistical community. Thus, we compare our GNRC with the glmnet. Since the GNRC solver is written in MATLAB, we use a MATLAB version of glmnet, which is a MATLAB wrapper for Fortran code (available online at https://github.com/distrep/DMLT/tree/master/external/glmnet). Besides using the default stopping parameters in the glmnet solver, we run glmnet on the same path as the one used in GNRC with the VC tuning parameter selector (16) for a fair comparison.

3.2 Simulation

3.2.1 Implementation setting

We generate synthetic data from (1). The rows of the $n \times p$ matrix $\mathbf{X}$ are sampled as i.i.d. copies from $N(\mathbf{0}, \Sigma)$ with $\Sigma = (r^{|j-k|})$, $1 \le j, k \le p$, where r is the correlation coefficient of $\mathbf{X}$. The noise vector $\boldsymbol{\varepsilon}$ is generated independently from $N(\mathbf{0}, \sigma^2 \mathbf{I}_n)$, where σ is the noise level. The underlying regression coefficient vector $\boldsymbol{\beta}^*$ is a random sparse vector chosen as T-sparse with a dynamic range (DR) defined by

$$\text{DR} := \frac{\max\{|\beta_j^*| : \beta_j^* \neq 0\}}{\min\{|\beta_j^*| : \beta_j^* \neq 0\}} = 10^{\alpha}, \tag{17}$$

where α is a parameter quantifies the dynamic range. Let $\mathcal{A} = \{j : |\beta_j^*| \neq 0\}$ be the true model and $\widehat{\mathcal{A}} = \{j : \widehat{\beta}_j(\lambda) \neq 0\}$ be the estimated model. Then $|\mathcal{A}| = \|\boldsymbol{\beta}^*\|_0 = T$. Following [2], each nonzero entry of $\boldsymbol{\beta}^*$ is generated as follows:

$$\beta_j^* = \eta_{1j} 10^{\alpha \eta_{2j}}, \tag{18}$$

where $j \in \mathcal{A}$, $\eta_{1j} = \pm 1$ with probability $\frac{1}{2}$ and η_{2j} is uniformly distributed in $[0, 1]$. Unless otherwise stated, α is fixed at 1. Then the outcome vector $\mathbf{y}$ is generated via $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\varepsilon}$. For convenience, we denote the data generated above by (n, p, r, σ, T).

3.2.2 The accuracy of the VC tuning parameter selector

We give an example to show the accuracy of the proposed VC tuning parameter selector (16) for two solvers with data $(n = 200, p = 1000, r = 0.5, \sigma = 0.1, T = 10)$. The results for GNRC and glmnet are summarized in Figure 2.

As shown in the left column in Figure 2, the majority vote selection rule (16) will determine the support size that occurs most frequently, i.e., $\arg\max\{|\Lambda_i|\} = 10$, which coincides with the support size of the true regression coefficient. The tuning parameter and the final solution to be selected are $\widehat{\lambda} = \max\{\Lambda_{10}\}$ and $\widehat{\boldsymbol{\beta}}_{\widehat{\lambda}}$, respectively. The right column in Figure 2 shows the selected estimators $\widehat{\boldsymbol{\beta}}_{\widehat{\lambda}}$ and the true target $\boldsymbol{\beta}^*$. We can see that the selected estimator and the true value are very close. Therefore, the data driven VC selector (16) performs well for both GNRC and glmnet.

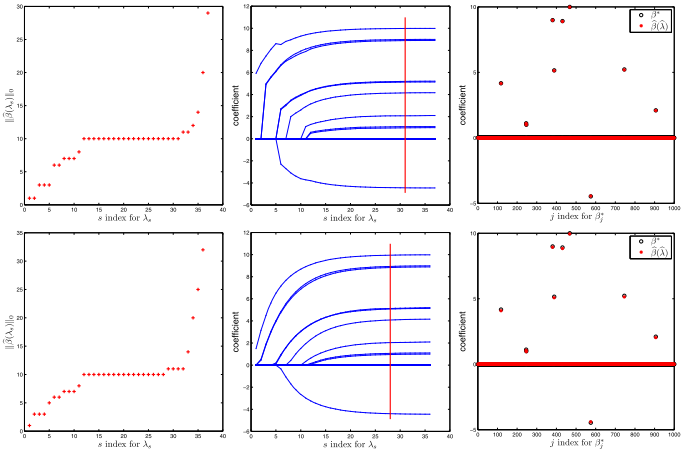

Figure 2. Plots for GNRC (first row) and glmnet (second row) using the VC tuning parameter selector on data $(n = 200, p = 1000, r = 0.5, \sigma = 0.1, T = 10)$: $\|\widehat{\boldsymbol{\beta}}(\lambda_s)\|_0$ on the path (left panel), the solution path $\{\widehat{\boldsymbol{\beta}}(\lambda_s)\}_s$ and the solution selected by the VS selector (middle panel), and the comparison between the underlying true parameter $\boldsymbol{\beta}^*$ and the selected solution $\widehat{\boldsymbol{\beta}}(\widehat{\lambda})$ (right panel).

It is noteworthy that the VC selector is seamlessly integrated with the continuation strategy without any extra computational overhead, since the sequence $\{\widehat{\boldsymbol{\beta}}(\lambda_s)\}$ is already generated along the continuation solution path.

3.2.3 The behavior of the GNRC algorithm

Next, we study the influence of the free parameters M and J in the GNRC algorithm on the exact support recovery probability, i.e., the percentage that the estimated model $\widehat{\mathcal{A}}$ in agreement with the true model $\mathcal{A}$. To this end, we independently generate 100 datasets from $(n = 200, p = 1000, r = 0.1, \sigma = 0.1, T = 5 : 5 : 30)$ for each combination of (M, J). Here $5 : 5 : 30$ means the sparsity level starts from 5 to 30 with an increment of 5. The numerical results

are summarized in Figure 3, which contain the following two settings: (a) $J = 1$ and $M \in \{40, 60, 80, 100\}$; (b) $M = 100$ and $J \in \{1, 2, 3\}$.

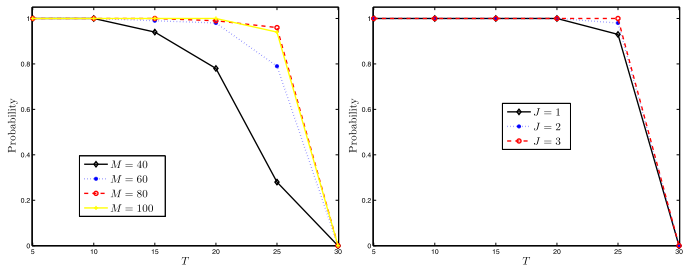

Figure 3. The influence of the GNRC parameters M (left panel) and J (right panel) on the exact support recovery probability.

It can be seen from Figure 3 that the influence of the parameter J is very mild on the exact support recovery probability. Specifically, as J increases, the reconstruction accuracy is hardly improved, even though the algorithm gets computationally more expensive. We can also find in Figure 3 that Larger M improves the exact support recovery probability, but the incremental improvement diminishes as M increases. Unsurprisingly, a relatively small value of M (e.g., $M = 40$) can degrade the accuracy of support recovery when the sparsity level T is relatively large, due to insufficient resolution of the solution path. Thus, it is reasonable to choose $(M, J) = (100, 1)$ for the GNRC algorithm in most cases. Unless otherwise specified, we set $(M, J) = (100, 1)$ throughout this paper.

3.2.4 Efficiency and accuracy

To evaluate the efficiency and accuracy of the proposed GNRC algorithm on variable selection, we independently generate $N = 100$ datasets from $(n, p, r, \sigma, T) \in \{200\} \times \{1000, 2000\} \times \{0.3, 0.5, 0.7\} \times \{0.4, 0.8\} \times \{10\}$. Based on N replications, we compare GNRC with glmnet in terms of the average CPU time (Time, in seconds), the estimated average model size (MS) $N^{-1}\sum_{m=1}^{N}|\widehat{\mathcal{A}}^{(m)}|$, the proportion of correct models (CM) $N^{-1}\sum_{m=1}^{N} I\{\widehat{\mathcal{A}}^{(m)} = \mathcal{A}\}$ (in percentage terms), the average ℓ_∞ absolute error (AE) $N^{-1}\sum_{m=1}^{N}\|\widehat{\boldsymbol{\beta}}^{(m)} - \boldsymbol{\beta}\|_\infty$, and the average ℓ_2 relative error (RE) $N^{-1}\sum_{m=1}^{N}(\|\widehat{\boldsymbol{\beta}}^{(m)} - \boldsymbol{\beta}\|_2/\|\boldsymbol{\beta}\|_2)$. The cpu time measures the efficiency of the solvers. MS, CM, AE and RE measures the accuracy (quality) of the solutions. Obviously, Time, AE and RE are the smaller the better. Ideally, MS $= T$ and CM $= 100\%$. Simulation results for variable selection with different parameter tuples are summarized in Table 1.

For each (p, r, σ) combination, we observe from Table 1 that GNRC has better speed performance than glmnet by Time. Further, considering that the GNRC is written in pure Matlab while the glmnet is a Matlab wrapped Fortran solver, it is fair to say that the proposed GNRC is a highly efficient method for solving the LASSO problem (2). Compared with glmnet, GNRC can produce solutions with smaller AE and RE. In terms of MS and CM, GNRC selects the correct model more frequently than glmnet. The CPU time of GNRC generally decreases as σ increases, increases linearly with p, and is relatively robust to the choice of r, with the other two parameters in the 3-tuple (p, r, σ) fixed. Unsurprisingly, larger σ or r tends to degrade the accuracy of GNRC. Similar phenomena are observed for glmnet as well. In summary, GNRC outperforms glmnet in terms of both efficiency and accuracy.

As shown in Table 1, the efficiency and accuracy of GNRC or glmnet will be affected by the variation of model parameters. In next subsection, we conduct the sensitivity analysis for the model parameters (n, p, r, σ, T) in a wider range in terms of both CPU time and solution quality.

3.2.5 Sensitivity analysis of model parameters

Next, we consider the sensitivity of the model performance of both GNRC and glmnet to their model parameters (i.e., $\{n, p, r, \sigma, T\}$). More specifically, we calculate the changes in the exact support recovery probability (denoted by *Probability*) and CPU time (denoted by *Time* and measured in seconds), as one parameter changes and the others are held constant. Based on 100 independent runs, our simulation is set up as follows.

- *Sensitivity to sample size n.* The results on *Probability* and *Time* are respectively shown in the top left panels of Figure 4 and Figure 5, where the grid of values for n is specified to be $\{100+50\cdot k : k = 0, 1, \ldots, 4\}$, and p, r, σ, and T are fixed at 1500, 0.1, 0.1, and 10, respectively.
- *Sensitivity to dimension p.* The results on *Probability* and *Time* are respectively shown in the top right panels of Figure 4 and Figure 5, where the grid of values for p is specified to be $\{500 + 500 \cdot k : k = 0, 1, \ldots, 4\}$, and n, r, σ, and T are fixed at 200, 0.1, 0.1, and 10, respectively.
- *Sensitivity to correlation level r.* The results on *Probability* and *Time* are respectively shown in the middle left panels of Figure 4 and Figure 5, where the grid of values for r is specified to be $\{0.1 + 0.2 \cdot k : k = 0, 1, \ldots, 4\}$, and n, p, σ, and T are fixed at 200, 1500, 0.1, and 10, respectively.
- *Sensitivity to noise level σ.* The results on *Probability* and *Time* are respectively shown in the middle right panels of Figure 4 and Figure 5, where the grid of values for σ is specified to be $\{0.2, 0.4, 0.8, 1.2, 1.6\}$, and n, p, r, and T are fixed at 200, 1500, 0.1, and 10, respectively.
- *Sensitivity to sparsity level T.* The results on *Probability* and *Time* are respectively shown in the bottom left panels of Figure 4 and Figure 5, where the grid of values for T is specified to be $\{3 + 3 \cdot k : k = 0, 1, \ldots, 4\}$,

Table 1. Simulation results for variable selection with $n = 200$ and $T = 10$ based on 100 replications

p	r	σ	Method	Time	MS	CM	AE	RE
1000	0.3	0.4	glmnet	0.0150	9.96	95%	0.2786	0.0409
			GNRC	0.0143	10.00	100%	0.1079	0.0132
		0.8	glmnet	0.0144	9.11	71%	0.9109	0.1251
			GNRC	0.0126	9.97	98%	0.2387	0.0281
	0.5	0.4	glmnet	0.0145	9.97	82%	0.3651	0.0518
			GNRC	0.0137	10.00	100%	0.1117	0.0140
		0.8	glmnet	0.0142	9.19	55%	1.0496	0.1379
			GNRC	0.0123	9.54	90%	0.4500	0.0519
	0.7	0.4	glmnet	0.0147	10.32	38%	0.5988	0.0828
			GNRC	0.0139	9.99	99%	0.1307	0.0154
		0.8	glmnet	0.0144	8.33	20%	1.8025	0.2306
			GNRC	0.0126	9.72	92%	0.3404	0.0412
2000	0.3	0.4	glmnet	0.0302	9.70	83%	0.5027	0.0703
			GNRC	0.0210	10.00	100%	0.1182	0.0148
		0.8	glmnet	0.0289	8.51	56%	1.3183	0.1789
			GNRC	0.0189	9.89	96%	0.2851	0.0334
	0.5	0.4	glmnet	0.0295	9.77	86%	0.4933	0.0681
			GNRC	0.0205	10.00	100%	0.1175	0.0147
		0.8	glmnet	0.0282	8.84	50%	1.1584	0.1616
			GNRC	0.0182	9.91	98%	0.2521	0.0318
	0.7	0.4	glmnet	0.0304	10.11	34%	0.7325	0.1026
			GNRC	0.0213	10.00	100%	0.1188	0.0147
		0.8	glmnet	0.0289	7.55	20%	1.9223	0.2599
			GNRC	0.0190	9.57	92%	0.4272	0.0530

and n, p, r, and σ are fixed at 200, 1500, 0.1, and 0.1, respectively.

From Figure 4 and Figure 5, one can make the following findings: (1) Larger n enhances the exact support recovery probabilities of both solvers, while larger p, r, σ and T reduce the probabilities. In particular, the exact support recovery probabilities of GNRC are greater than those of glmnet in all the scenarios considered, suggesting that GNRC identifies the true model more frequently than glmnet does. In addition, the probabilities of GNRC exhibit notable robustness with respect to n, p and T. (2) GNRC is faster than glmnet in all cases except when both n and p are relatively small. The time of both solvers increases linearly with p and n. This is consistent with the complexity analysis in Section 2.3. Thus, both solvers can be efficiently scaled up to other larger datasets. Also, the CPU time increases as ρ increases and decreases as σ increases. Overall, the numerical results shown in Figures 4–5 demonstrate that the two solvers have similar variation tendency, with GNRC being more accurate and efficient.

3.3 Application

As an illustration of our method with a real data, we applied the method to the genetics data in [38], which consists of 18975 probes obtained from 120 rats and is publicly available at http://myweb.uiowa.edu/pbreheny/data/Scheetz2006.html. See [22, 49, 5] for a more detailed description of this data set. The goal is to find the probes that are related to the gene TRIM32 known to cause Bardet-Biedl syndrome [11].

Following [22], we first selected 3000 probes with the largest variance in expression level and then chose the top 200 probes that have the largest absolute correlation with TRIM32 among the 3000 probes. We then applied GNRC and glmnet to these 200 probes in the analysis, where the numerical response variable $\mathbf{y}$ is of length 120 giving the expression levels of the gene TRIM32, and the design matrix $\mathbf{X}$ (of size 120×200) represents the data of 120 rats with 200 probes. Table 2 summarizes the comparison in terms of the following metrics: the selected probes along with their corresponding nonzero coefficient estimates, CPU time (in seconds) and prediction error (PE) calculated as $n^{-1}\sum_{i=1}^{n}(\widehat{y}_i - y_i)^2$. It can be seen from the table that GNRC took less computing time and achieved smaller PE than glmnet did. GNRC and glmnet respectively identified 5 and 10 probes, with 5 probes in common (i.e., 1383110_at, 1389584_at, 1374106_at, 1370429_at and 1383996_at). Although of different magnitudes, the estimates for the common coefficients have the same signs, leading to similar biological conclusions.

Next, we examined the quality of variable selection by GNRC and glmnet via cross-validation with $(M, J) = (200, 1)$ and $p = 1000$ and 2000. Specifically, the 120 rats were randomly partitioned into two groups 100 times. In each partition, one group of 100 rats was used as training data and the remaining group of 20 rats was used as

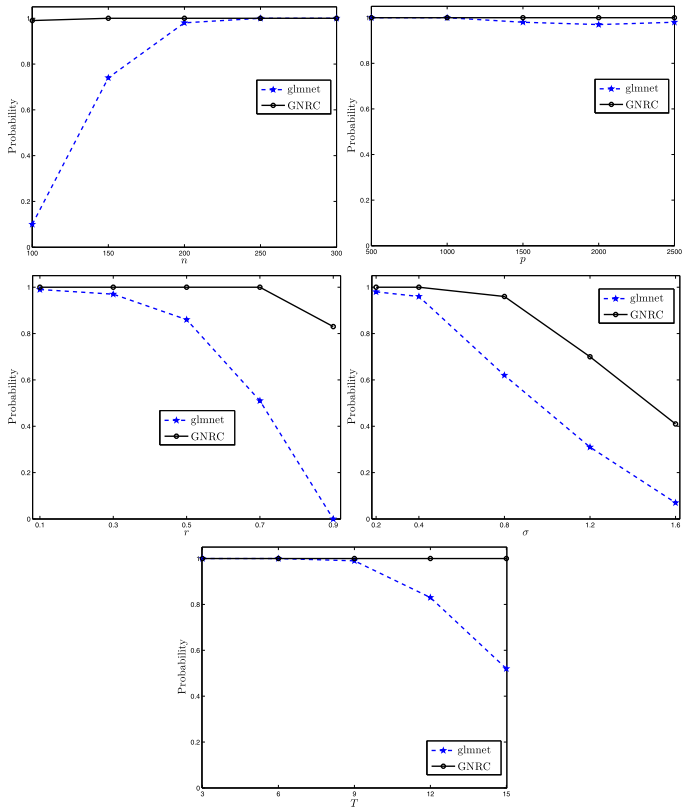

Figure 4. Numerical results of the influence of the sample size n (top left panel), the dimension p (top right panel), the correlation level r (middle left panel), the noise level σ (middle right panel), and the sparsity level T (bottom middle panel) on the exact support recovery probability of GNRC and glmnet.

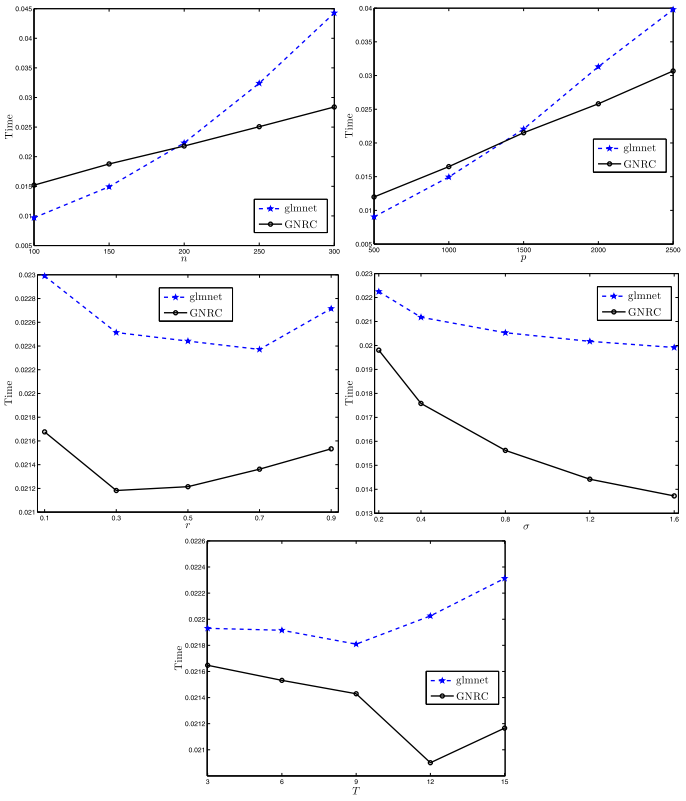

Figure 5. Numerical results of the influence of the sample size n (top left panel), the dimension p (top right panel), the correlation level r (middle left panel), the noise level σ (middle right panel), and the sparsity level T (bottom middle panel) on the CPU time of GNRC and glmnet.

test data. The CPU time (Time, in seconds) and model size (MS, i.e., the number of selected probes) were calculated using the training data, and the prediction errors were calculated using the test data. Table 3 presents the average values over the 100 random partitions. The numbers in parenthesis are the corresponding standard deviations. As shown in Table 3, in comparison with glmnet, GNRC took less computing time, selected fewer probes and achieved smaller prediction error. It suggests that GNRC is able to provide the medical researchers with a more targeted list of probes in subsequent studies. Based on our cross-validation results, we report the selected probes and their corresponding frequency (denoted as Freq) in Table 4. It can be seen that the probes 1383110_at and 1389584_at are the top 2 probes, with Freq $>$ 75, indicating that these two probes are closely related to the gene TRIM32. Probes 1370429_at, 1374106_at, 1379971_at, 1383673_at and 1386683_at also display moderately high frequencies (Freq $>$ 20), suggesting their possible association with the target gene.

4. CONCLUDING REMARKS

Starting with the KKT condition, we develop a fast generalized Newton-Raphson algorithm for solving LASSO-type problems. We establish the local one step convergence of the proposed algorithm and show that its computational complexity is $O(np)$. When paired with the continuation strategy, our algorithm is able to produce solution path efficiently. A novel tuning parameter selector is suggested as well. Numerical comparison with glmnet on both real and synthetic datasets show that our algorithm works well in applications.

There are several avenues for further study. First, whether the theoretical and computational results still hold for non-convex regularizers (e.g., SCAD [14] and MCP [52]) requires more research. Second, extensions of the GNRC algorithm are possible for structured sparsity models (e.g., group sparsity penalty and the matrix analogue) and other regression problems (e.g., logistic regression and Cox models). Third, implementing GNRC as a distributed algorithm [47, 27] for large-scale inference is another interesting direction to pursue. Last, we globalize the local convergence of GNR by a

Table 2. The probes identified by glmnet and GNRC that correlated with TRIM32 based on the subset of the real data ($n = 120, p = 200$)

Term	Probe	glmnet	GNRC
Intercept		7.1536	6.3125
1	1383110_at	0.0450	0.0598
2	1389584_at	0.0560	0.1061
3	1383673_at	0.0189	
4	1386683_at	0.0131	
5	1379971_at	0.0270	
6	1374106_at	0.0186	0.0734
7	1370429_at	−0.0299	−0.0667
8	1383749_at	−0.0226	
9	1369353_at	−0.0105	
10	1383996_at	0.0470	0.1083
Time		0.0466	0.0301
PE		0.0077	0.0062

Table 3. The CPU time (Time), model size (MS) and prediction error (PE) averaged across 100 random partitions of the real data (numbers in parentheses are standard deviations)

p	Method	Time	MS	PE
1000	glmnet	0.0164(0.0089)	7.66(5.8573)	0.0165(0.0165)
	GNRC	0.0037(0.0041)	5.74(2.3979)	0.0150(0.0138)
2000	glmnet	0.0257(0.0021)	7.11(5.2989)	0.0172(0.0173)
	GNRC	0.0057(0.0037)	5.79(2.7718)	0.0157(0.0145)

simple continuation strategy. Globalization via smoothing Newton methods [10, 32, 33] is also of immense interest.

APPENDIX A. APPENDIX

A.1 Proof of Lemma 1

Proof. According to the fact $\lambda\|\boldsymbol{\beta}\|_1 = \max_{\|v\|_\infty \le \lambda}\langle \boldsymbol{\beta}, v\rangle$ with $v \in \mathbb{R}^p$ and the Minimax Theorem [36], we have

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} Q_\lambda(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta}\in\mathbb{R}^p}\left\{\frac{1}{2}\|\mathbf{X}\boldsymbol{\beta}-\mathbf{y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1\right\} \tag{19}$$

$$= \min_{\boldsymbol{\beta}\in\mathbb{R}^p}\left\{\frac{1}{2}\|\mathbf{X}\boldsymbol{\beta}-\mathbf{y}\|_2^2 + \max_{\|v\|_\infty\le\lambda}\langle\boldsymbol{\beta}, v\rangle\right\} \tag{20}$$

$$= \min_{\boldsymbol{\beta}\in\mathbb{R}^p}\left\{\max_{\|v\|_\infty\le\lambda}\left[\frac{1}{2}\|\mathbf{X}\boldsymbol{\beta}-\mathbf{y}\|_2^2 + \langle\boldsymbol{\beta}, v\rangle\right]\right\} \tag{21}$$

$$= \max_{\|v\|_\infty\le\lambda}\left\{\min_{\boldsymbol{\beta}\in\mathbb{R}^p}\left[\frac{1}{2}\|\mathbf{X}\boldsymbol{\beta}-\mathbf{y}\|_2^2 + \langle\boldsymbol{\beta}, v\rangle\right]\right\} \tag{22}$$

$$= \max_{\|v\|_\infty\le\lambda}\left\{\min_{\boldsymbol{\beta}\in\mathbb{R}^p}\left[\frac{1}{2}\boldsymbol{\beta}^\top G\boldsymbol{\beta} - \boldsymbol{\beta}^\top(\widetilde{\mathbf{y}} - v)\right]\right\}. \tag{23}$$

Noting that $\min_{\boldsymbol{\beta}\in\mathbb{R}^p}\left[\frac{1}{2}\boldsymbol{\beta}^\top G\boldsymbol{\beta} - \boldsymbol{\beta}^\top(\widetilde{\mathbf{y}} - v)\right]$ implies that

$$\begin{cases}\boldsymbol{\beta} = G^{-1}(\widetilde{\mathbf{y}} - v), & (24)\\ v = \widetilde{\mathbf{y}} - G\boldsymbol{\beta} = \mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}), & (25)\end{cases}$$

then (23) becomes

$$\max_{\|\mathbf{X}^\top(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\|_\infty\le\lambda}\left\{-\frac{1}{2}\boldsymbol{\beta}^\top G\boldsymbol{\beta}\right\} \Leftrightarrow \min_{\|\widetilde{\mathbf{y}}-G\boldsymbol{\beta}\|_\infty\le\lambda}\langle\boldsymbol{\beta}, G\boldsymbol{\beta}\rangle, \tag{26}$$

which completes the proof. □

A.2 Proof of Theorem 1

Proof. (3) can be reformulated as

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p}\langle\boldsymbol{\beta}, G\boldsymbol{\beta}\rangle \quad s.t. \quad c(\boldsymbol{\beta}) \le 0. \tag{27}$$

By standard convex optimization, we know the KKT condition of $\overline{\boldsymbol{\beta}}$ being a minimizer of (27) is

$$\begin{cases} 2G\overline{\boldsymbol{\beta}} + \widetilde{G}\overline{\mu} = 0, \\ \overline{\mu} \ge 0, c(\overline{\boldsymbol{\beta}}) \le 0, \overline{\mu} \perp c(\overline{\boldsymbol{\beta}}), \end{cases}$$

where $\overline{\mu}$ is the lagrange multiplier of the inequality constraint $c(\boldsymbol{\beta}) \le 0$. Observing $\overline{\mu} \ge 0, c(\overline{\boldsymbol{\beta}}) \le 0, \overline{\mu} \perp c(\overline{\boldsymbol{\beta}}) \Leftrightarrow \overline{\mu} = P_{\mathbb{R}^{2p}_+}(\overline{\mu} + c(\overline{\boldsymbol{\beta}}))$, we obtain that (4) holds if $\overline{\boldsymbol{\beta}}$ is a minimizer of (3). The converse results follow from the fact that (3) is a convex optimization with constraint. □

A.3 Proof of Lemma 2

Proof. Similar as the derivation in Section 2.4.5.1 in [19]. □

A.4 Proof of Theorem 2

Proof. Direct calculation based on the Theorem 2.10 in [19] and Lemma 2 above yields the results. □

A.5 Proof of Theorem 3

Proof. Let $\widetilde{\mathcal{I}} = \{j : |\widehat{\boldsymbol{\beta}}_j + \widehat{\mathbf{d}}_j| \ne \lambda\}$, and $\mathbf{d}^0 = \mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^0)$. Define

$$C_\lambda = \min_{i\in\widetilde{\mathcal{A}}}||\widehat{\boldsymbol{\beta}}_i + \widehat{\mathbf{d}}_i| - \lambda|. \tag{28}$$

$\forall i \in \{j : \widehat{\boldsymbol{\beta}}_j + \widehat{\mathbf{d}}_j > \lambda\}$, we have

$$\begin{aligned} &\widehat{\boldsymbol{\beta}}_i + \widehat{\mathbf{d}}_i - \boldsymbol{\beta}_i^0 - \mathbf{d}_i^0 \\ &\le |\boldsymbol{\beta}_i^0 + \mathbf{d}_i^0 - \widehat{\boldsymbol{\beta}}_i - \widehat{\mathbf{d}}_i| \\ &\le \|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_\infty + \|\widehat{\mathbf{d}} - \mathbf{d}^0\|_\infty \\ &\le \min_{i\in\widetilde{\mathcal{A}}}||\widehat{\boldsymbol{\beta}}_i + \widehat{\mathbf{d}}_i| - \lambda| \\ &\le \widehat{\boldsymbol{\beta}}_i + \widehat{\mathbf{d}}_i - \lambda, \end{aligned}$$

where the third inequality uses the assumption $\boldsymbol{\beta}^0$ is close to $\widehat{\boldsymbol{\beta}}$ in the sense that $\|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_\infty + \|\widehat{\mathbf{d}} - \mathbf{d}^0\|_\infty \le C_\lambda$. This implies that $\widehat{\boldsymbol{\beta}}_i + \widehat{\mathbf{d}}_i > \lambda \Longrightarrow \boldsymbol{\beta}_i^0 + \mathbf{d}_i^0 > \lambda$ (similarly, we can show $\widehat{\boldsymbol{\beta}}_i + \widehat{\mathbf{d}}_i < -\lambda \Longrightarrow \boldsymbol{\beta}_i^0 + \mathbf{d}_i^0 < -\lambda$), i.e., $\{i : |\widehat{\boldsymbol{\beta}}_i + \widehat{\mathbf{d}}_i| > \lambda\} \subseteq \mathcal{I}_0 = \{i : |\boldsymbol{\beta}_i^0 + \mathbf{d}_i^0| > \lambda\}$. Meanwhile, by the

Table 4. Frequency table for 100 random partitions of the real data (only probes with Freq ≥ 10 are listed for the sake of brevity and focus)

$p = 1000$				$p = 2000$			
glmnet		GNRC		glmnet		GNRC	
Probe	Freq	Probe	Freq	Probe	Freq	Probe	Freq
1389584_at	83	1383110_at	83	1389584_at	82	1383110_at	84
1383110_at	78	1389584_at	82	1383110_at	76	1389584_at	77
1383673_at	55	1383673_at	68	1383673_at	54	1383673_at	67
1379971_at	50	1386683_at	62	1374106_at	49	1386683_at	58
1383996_at	43	1379971_at	40	1370429_at	44	1370551_a_at	42
1374106_at	42	1370551_a_at	38	1383996_at	44	1374106_at	35
1386683_at	35	1370429_at	32	1379971_at	41	1379971_at	34
1383749_at	35	1379495_at	27	1386683_at	37	1370429_at	28
1370429_at	32	1374106_at	24	1383749_at	31	1379495_at	27
1369353_at	24	1394455_at	17	1379495_at	18	1394455_at	19
1379495_at	18	1379597_at	15	1369353_at	14	1382263_at	14
1393817_at	17	1382263_at	15	1382263_at	12	1393817_at	11
1382452_at	13	1393817_at	13	1393817_at	11	1382517_at	10
1384204_at	13	1382517_at	11	1381787_at	11		
1383522_at	13			1393979_at	10		
1382835_at	12						
1382263_at	11						
1379597_at	10						

same arguments we can show that $|\widehat{\boldsymbol{\beta}}_i + \widehat{\mathbf{d}}_i| < \lambda \Longrightarrow |\boldsymbol{\beta}_i^0 + \mathbf{d}_i^0| < \lambda$, i.e., $\mathcal{I}_0 \subseteq \mathcal{I}$. Then by the definition of $\widehat{\mathbf{d}}$ and the KKT condition of $\widehat{\boldsymbol{\beta}}$, we deduce $\widehat{\mathbf{d}}_{\mathcal{I}_0} = \lambda \mathrm{sgn}(\widehat{\mathbf{d}}_{\mathcal{I}_0} + \widehat{\boldsymbol{\beta}}_{\mathcal{I}_0})$. The definition of $\boldsymbol{\beta}^1$ together with the KKT condition implies

$$\mathbf{X}_{\mathcal{I}_0}^\top \mathbf{X}_{\mathcal{I}_0} \widehat{\boldsymbol{\beta}}_{\mathcal{I}_0} + \widehat{\mathbf{d}}_{\mathcal{I}_0} = \mathbf{X}_{\mathcal{I}_0}^\top \mathbf{y} = \mathbf{X}_{\mathcal{I}_0}^\top \mathbf{X}_{\mathcal{I}_0} \boldsymbol{\beta}_{\mathcal{I}_0}^1 + \lambda \begin{pmatrix} \mathbf{1}_{\mathcal{I}_1^0} \\ -\lambda \mathbf{1}_{\mathcal{I}_2^0} \end{pmatrix}.$$

Then we get $\mathbf{X}_{\mathcal{I}_0}^\top \mathbf{X}_{\mathcal{I}_0} (\widehat{\boldsymbol{\beta}}_{\mathcal{I}_0} - \boldsymbol{\beta}_{\mathcal{I}_0}^1) = \mathbf{0}$, therefore, $\widehat{\boldsymbol{\beta}}_{\mathcal{I}0} = \boldsymbol{\beta}_{\mathcal{I}_0}^1$ follows from the the full rank assumption of $\mathbf{X}_{\mathcal{I}}$. Let $\mathcal{A}_0 = (\mathcal{I}_0)^c$. By the fact $\mathcal{I} \subset \mathcal{I}_0$ and the KKT condition we deduce that $\boldsymbol{\beta}_{\mathcal{A}_0}^1 = \mathbf{0} = \widehat{\boldsymbol{\beta}}_{\mathcal{A}_0}$. Hence, $\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}^1$. This completes the proof. □

ACKNOWLEDGEMENTS

The authors would like to thank an associate editor, an anonymous referee, and the Editor-in-Chief Professor Yuedong Wang for their careful reading and many constructive and insightful comments that led to significant improvements in this article. They would also like to thank Professor Defeng Sun of The Hong Kong Polytechnic University and Professor Bangti Jin of University College London for their valuable discussions and suggestions.

Received 24 July 2019

REFERENCES

[1] Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* **2** 183–202. MR2486527

[2] Becker, S., Bobin, J. and Candès, E. J. (2011). NESTA: A fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences* **4** 1–39. MR2765668

[3] Breheny, P. and Huang, J. (2009). Penalized methods for bi-level variable selection. *Statistics and Its Interface* **2** 369–380. MR2540094

[4] Breheny, P. and Huang, J. (2011). Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The Annals of Applied Statistics* **5** 232–253. MR2810396

[5] Breheny, P. and Huang, J. (2015). Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and Computing* **25** 173–187. MR3306699

[6] Bühlmann, P. and van de Geer, S. (2011). *Statistics for high-dimensional data: methods, theory and applications*. Springer, Berlin. MR2807761

[7] Candes, E. J. and Tao, T. (2005). Decoding by linear programming. *IEEE Transactions on Information Theory* **51** 4203–4215. MR2243152

[8] Chen, S. S., Donoho, D. L. and Saunders, M. A. (2001). Atomic decomposition by basis pursuit. *SIAM Review* **43** 129–159. MR1854649

[9] Chen, X., Nashed, Z. and Qi, L. (2000). Smoothing methods and semismooth methods for nondifferentiable operator equations. *SIAM Journal on Numerical Analysis* **38** 1200–1216. MR1786137

[10] Chen, X., Qi, L. and Sun, D. (1998). Global and superlinear convergence of the smoothing Newton method and its application to general box constrained variational inequalities. *Mathematics of Computation of the American Mathematical Society* **67** 519–540. MR1458218

[11] Chiang, A. P., Beck, J. S., Yen, H.-J., Tayeh, M. K., Scheetz, T. E., Swiderski, R. E., Nishimura, D. Y., Braun, T. A., Kim, K.-Y. A., Huang, J., Elbedour, K., Carmi, R., Slusarski, D. C., Casavant, T. L., Stone, E. M. and Sheffield, V. C. (2006). Homozygosity mapping with SNP arrays identifies TRIM32, an E3 ubiquitin ligase, as a Bardet–Biedl syndrome gene (BBS11). *Proceedings of the National Academy of*

Sciences **103** 6287–6292.
[12] Donoho, D. L. and Tsaig, Y. (2008). Fast solution of ℓ_1-norm minimization problems when the solution may be sparse. *IEEE Transactions on Information Theory* **54** 4789–4812. MR2589865
[13] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics* **32** 407–499. MR2060166
[14] 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. MR1946581
[15] Friedman, J., Hastie, T., Höfling, H. and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics* **1** 302–332. MR2415737
[16] Friedman, J., Hastie, T. and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software* **33** 1–22.
[17] Fu, W. J. (1998). Penalized regressions: the bridge versus the lasso. *Journal of Computational and Graphical Statistics* **7** 397–416. MR1646710
[18] Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The elements of statistical learning*, 2 ed. Springer, New York. MR2722294
[19] Hinze, M., Pinnau, R., Ulbrich, M. and Ulbrich, S. (2009). *Optimization with PDE constraints*. Springer. MR2516528
[20] Huang, J., Jiao, Y., Jin, B., Liu, J., Lu, X. and Yang, C. (2018). A Unified Primal Dual Active Set Algorithm for Nonconvex Sparse Recovery. arXiv preprint arXiv:1310.1147v4.
[21] Huang, J., Jiao, Y., Lu, X. and Zhu, L. (2018). Robust decoding from 1-bit compressive sampling with least squares. *SIAM Journal on Scientific Computing* **40** A2062–A2086. MR3820388
[22] Huang, J., Ma, S. and Zhang, C.-H. (2008). Adaptive Lasso for sparse high-dimensional regression models. *Statistica Sinica* **18** 1603–1618. MR2469326
[23] Ito, K. and Kunisch, K. (2008). *Lagrange multiplier approach to variational problems and applications*. SIAM, Philadelphia. MR2441683
[24] Jiao, Y., Jin, B. and Lu, X. (2015). A primal dual active set with continuation algorithm for the ℓ^0-regularized optimization problem. *Applied and Computational Harmonic Analysis* **39** 400–426. MR3398943
[25] Jiao, Y., Jin, B. and Lu, X. (2017). Iterative soft/hard thresholding with homotopy continuation for sparse recovery. *IEEE Signal Processing Letters* **24** 784–788.
[26] Jiao, Y., Jin, B. and Lu, X. (2017). Group sparse recovery via the $\ell^0(\ell^2)$ penalty: theory and algorithm. *IEEE Transactions on Signal Processing* **65** 998–1012. MR3583009
[27] Jordan, M. I., Lee, J. D. and Yang, Y. (2019). Communication-efficient distributed statistical inference. *Journal of the American Statistical Association* **114** 668–681. MR3963171
[28] Kummer, B. (1988). Newton's method for non-differentiable functions. *Advances in Mathematical Optimization* **45** 114–125. MR0953328
[29] Li, Y. and Osher, S. (2009). Coordinate descent optimization for ℓ^1 minimization with application to compressed sensing; a greedy algorithm. *Inverse Problems and Imaging* **3** 487–503. MR2557916
[30] Meinshausen, N. and Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics* **34** 1436–1462. MR2278363
[31] Osborne, M. R., Presnell, B. and Turlach, B. A. (2000). A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis* **20** 389–403. MR1773265
[32] Qi, L. and Sun, D. (1999). A survey of some nonsmooth equations and smoothing Newton methods. In *Progress in optimization* 121–146. Springer. MR1719516
[33] Qi, L., Sun, D. and Zhou, G. (2000). A new look at smoothing Newton methods for nonlinear complementarity problems and box constrained variational inequalities. *Mathematical Programming* **87** 1–35. MR1734657
[34] Qi, L. and Sun, J. (1993). A nonsmooth version of Newton's method. *Mathematical Programming* **58** 353–367. MR1216791
[35] Rish, I. and Grabarnik, G. Y. (2014). *Sparse modeling: theory, algorithms, and applications*. CRC, Boca Raton. MR3328718
[36] Rockafellar, R. T. (1970). *Convex analysis*. Princeton University Press, Princeton. MR0274683
[37] Saha, A. and Tewari, A. (2013). On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization* **23** 576–601. MR3037002
[38] Scheetz, T. E., Kim, K.-Y. A., Swiderski, R. E., Philp, A. R., Braun, T. A., Knudtson, K. L., Dorrance, A. M., DiBona, G. F., Huang, J., Casavant, T. L., Sheffield, V. C. and Stone, E. M. (2006). Regulation of gene expression in the mammalian eye and its relevance to eye disease. *Proceedings of the National Academy of Sciences of the United States of America* **103** 14429–14434.
[39] Shi, Y., Wu, Y., Xu, D. and Jiao, Y. (2018). An ADMM with continuation algorithm for non-convex SICA-penalized regression in high dimensions. *Journal of Statistical Computation and Simulation* **88** 1826–1846. MR3784583
[40] Shi, Y., Zhou, Z., Jiao, Y. and Wang, J. (2019). A primal dual active set with continuation algorithm for high-dimensional nonconvex SICA-penalized regression. *Journal of Statistical Computation and Simulation* **89** 864–883. MR3910941
[41] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* **58** 267–288. MR1379242
[42] Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J. and Tibshirani, R. J. (2012). Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **74** 245–266. MR2899862
[43] Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications* **109** 475–494. MR1835069
[44] Tseng, P. and Yun, S. (2009). A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming* **117** 387–423. MR2421312
[45] Wang, H., Li, B. and Leng, C. (2009). Shrinkage tuning parameter selection with a diverging number of parameters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **71** 671–683. MR2749913
[46] Wang, H., Li, R. and Tsai, C.-L. (2007). Tuning parameter selectors for the smoothly clipped absolute deviation method. *Biometrika* **94** 553–568. MR2410008
[47] Wang, J., Kolar, M., Srebro, N. and Zhang, T. (2017). Efficient distributed learning with sparsity. In *Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research* **70** 3636–3645. PMLR, International Convention Centre, Sydney, Australia.
[48] Wang, L., Kim, Y. and Li, R. (2013). Calibrating nonconvex penalized regression in ultra-high dimension. *The Annals of Statistics* **41** 2505–2536. MR3127873
[49] Wang, L., Wu, Y. and Li, R. (2012). Quantile regression for analyzing heterogeneity in ultra-high dimension. *Journal of the American Statistical Association* **107** 214–222. MR2949353
[50] Wu, T. T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics* **2** 224–244. MR2415601
[51] Yun, S. (2014). On the iteration complexity of cyclic coordinate gradient descent methods. *SIAM Journal on Optimization* **24** 1567–1580. MR3264573
[52] Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics* **38** 894–942. MR2604701
[53] Zhao, P. and Yu, B. (2006). On model selection consistency of Lasso. *Journal of Machine Learning Research* **7** 2541–2563. MR2274449

Yueyong Shi
School of Economics and Management
China University of Geosciences
Wuhan, Hubei 430074
P. R. China
Center for Resources and Environmental Economic Research
China University of Geosciences
Wuhan, Hubei 430074
P. R. China
E-mail address: yueyongshi@cug.edu.cn

Jian Huang
Department of Statistics and Actuarial Science
University of Iowa
Iowa City, Iowa 52246
USA
E-mail address: jian-huang@uiowa.edu

Yuling Jiao
School of Mathematics and Statistics
Wuhan University
Wuhan, Hubei 430072
P. R. China
E-mail address: yulingjiaomath@whu.edu.cn

Yicheng Kang
Department of Mathematical Sciences
Bentley University
Waltham, MA 02452
USA
E-mail address: ykang@bentley.edu

Hu Zhang
School of Statistics and Mathematics
Zhongnan University of Economics and Law
Wuhan, Hubei 430073
P. R. China
E-mail address: zczyde@163.com