Signal Processing 194 (2022) 108432

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

PSNA: A pathwise semismooth Newton algorithm for sparse recovery with optimal local convergence and oracle properties

Jian Huang[a,*], Yuling Jiao[b], Xiliang Lu[b,*], Yueyong Shi[c], Qinglong Yang[d], Yuanyuan Yang[b]

[a] Department of Statistics and Actuarial Sciences, University of Iowa, Iowa City, Iowa, USA
[b] School of Mathematics and Statistics, Wuhan University, Wuhan, China
[c] School of Economics and Management, China University of Geosciences, Wuhan, China
[d] School of Statistics and Mathematics, Zhongnan University of Economics and Law, Wuhan, China

ARTICLE INFO

Article history:
Received 3 October 2020
Revised 27 September 2021
Accepted 11 December 2021
Available online 14 December 2021

Keywords:
KKT conditions
Lasso
Newton derivative
semismooth functions
sign consistency
superlinear convergence

ABSTRACT

We propose a pathwise semismooth Newton algorithm (PSNA) for sparse recovery in high-dimensional linear models. PSNA is derived from a formulation of the KKT conditions for Lasso and Enet based on Newton derivatives. It solves the semismooth KKT equations efficiently by actively and continuously seeking the support of the regression coefficients along the solution path with warm start. At each knot in the path, PSNA converges locally superlinearly for the Enet criterion and achieves the best possible convergence rate for the Lasso criterion, i.e., PSNA converges in just one step at the cost of two matrix-vector multiplication per iteration. Under certain regularity conditions on the design matrix and the minimum magnitude of the nonzero elements of the target regression coefficients, we show that PSNA hits a solution with the same signs as the regression coefficients and achieves a sharp estimation error bound in finite steps with high probability. Extensive simulation studies support our theoretical results and indicate that PSNA is competitive with or outperforms state-of-the-art Lasso solvers in terms of efficiency and accuracy.

1. Introduction

In this paper, we propose a pathwise semismooth Newton algorithm (PSNA) for sparse recovery in the high-dimensional linear model

$$\boldsymbol{y} = X\boldsymbol{\beta}^{\dagger} + \boldsymbol{\eta}, \tag{1.1}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is a response vector, $X \in \mathbb{R}^{n\times p}$ is a design matrix, $\boldsymbol{\beta}^{\dagger} = (\beta_1^{\dagger}, \ldots, \beta_p^{\dagger})' \in \mathbb{R}^p$ is a vector of underlying regression coefficients, and $\boldsymbol{\eta} \in \mathbb{R}^n$ is a vector of random errors. We assume without loss of generality that $\boldsymbol{y}$ is centered and the columns of X are centered and $\sqrt{n}$-normalized. For this model, the Lasso [1,2] solves

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} L_\lambda(\boldsymbol{\beta}) := \frac{1}{2n}\|X\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{1.2}$$

where $\lambda > 0$ is a penalty parameter. Closely related to the Lasso is the elastic net (Enet) [3], which solves

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} J_{\lambda,\alpha}(\boldsymbol{\beta}) := L_\lambda(\boldsymbol{\beta}) + \frac{\alpha}{2n}\|\boldsymbol{\beta}\|_2^2,\ \alpha > 0. \tag{1.3}$$

This can be viewed as a regularized form of (1.2). Since $J_{\lambda,\alpha}(\cdot)$ is strongly convex for $\alpha > 0$, the Enet solution $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$ is unique. This enables us to characterize the unique minimum 2-norm Lasso solution (1.2) as the limit of $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$ as $\alpha \to 0^+$ (Proposition 3.2). In high-dimensional settings, it is nontrivial to efficiently solve (1.2) and (1.3) numerically since they are large scale nondifferentiable optimization problems.

The key ingredient of PSNA is a semismooth Newton algorithm (SNA), which is derived based on a suitable formulation of the KKT conditions. At each step in the iteration, the SNA works by first estimating the support of the solution based on a combination of the primal and dual information, and then finding the values of the nonzero coefficients on the support. Interestingly, our analysis shows that the SNA can be formally derived as a Newton algorithm based on the notion of Newton derivatives for nondifferentiable functions [4–6].

PSNA proceeds by running SNA along a grid of λ values: $\{\lambda_t = \lambda_0\gamma^t\}_{t=0,1,\ldots N}$ with the continuation strategy and warm start, where $\gamma \in (0,1)$, $\lambda_0 > 0$ and the integer N are user given parameters. It is

* Corresponding authors: Jian Huang and Xiliang Lu
E-mail addresses: jian-huang@uiowa.edu (J. Huang), yulingjiaomath@whu.edu.cn (Y. Jiao), xllv.math@whu.edu.cn (X. Lu), yueyongshi@cug.edu.cn (Y. Shi), yangqinglong@zuel.edu.cn (Q. Yang), yuanyuanyang@whu.edu.cn (Y. Yang).

https://doi.org/10.1016/j.sigpro.2021.108432
0165-1684/

easy to implement and computationally stable. Moreover, our simulation studies indicate that PSNA is nearly problem independent, in the sense that the computational cost of using PSNA to approximate the solution path is $O(Nnp)$, independent of the following aspects of the model, including the ambient dimension, sparsity level, correlation structure of the predictors, range of the magnitude of the nonzero regression coefficients and the noise level.

1.1. Contributions

The most popular algorithms for solving ℓ_1-regularized problems in the literature are mainly first-order methods. It is natural to ask whether we can develop a second-order method, i.e, the Newton-type method, which is a workhorse in low-dimensional problems, for such nonsmooth optimization problems which converge faster than first-order methods. We give a definitive answer to this question by establishing faster convergence results for SNA. We show that, for the Lasso, the SNA converges locally in just one step, which is obviously the best possible local convergence rate for any algorithms (Theorem 3.3). For the Enet, it converges locally superlinearly (Theorem 3.2). To the best of our knowledge, these are the best convergence rates for Lasso and Enet problems with $p \gg n$ in the literature. Our computational complexity analysis shows that the cost of each iteration in SNA is $O(np)$, which is the same as most existing Lasso solvers, including LARS and coordinate descent algorithms. Hence, the overall cost of using SNA to find the unique minimizer of $J_{\lambda,\alpha}(\boldsymbol{\beta})$ is still $O(np)$ due to its superlinear convergence if it is warm started.

Another contribution of this paper is that we establish the statistical properties of PSNA in the Gaussian noise case. Specifically, we show that under certain regularity conditions on the design matrix X, the solution sequence generated by PSNA enjoys the sign consistency property in finite steps if the minimum magnitude of the nonzero elements of $\boldsymbol{\beta}^\dagger$ is of the order $O(\sigma\sqrt{2\log(p)/n})$, which is the optimal magnitude of detectable signal. We also establish a sharp upper bound in supreme norm for the estimation error of the solution sequence.

1.2. Related work

[7] showed that the Lasso solution path is continuous and piecewise linear as a function of λ. They proposed a Homotopy algorithm that defines an active set of nonzero variables at the current vertex then moves to a new vertex by adding a new variable to or removing an existing one from the active set. [8] proposed the LARS algorithm to trace the whole solution path of (1.2) by omitting the removing steps in the Homotopy algorithm. [9] showed that, in the noiseless case with $\boldsymbol{\eta} = 0$ and under certain conditions on X and $\boldsymbol{\beta}^\dagger$, LARS (Homotopy) algorithm has the "$\|\boldsymbol{\beta}^\dagger\|_0$-step" convergence property with the cost of $O(\|\boldsymbol{\beta}^\dagger\|_0 np)$. However, the convergence property of LARS is unknown when the noise vector $\boldsymbol{\eta}$ is nonzero in the $p > n$ settings. Further connections of SNA with LARS, sure independence screening [10], and active set methods for accelerating coordinate descent [11] are discussed in Section 5.

Several authors have adopted a Gauss-Seidel type coordinate descent algorithm (CD-GS) [12–15], as well as Jacobi type coordinate descent (CD-J), or iterative thresholding [16,17] to solve (1.2). For the CD-GS proposed in [13], the results of [18], [19] and [20] only ensure the convergence and sublinear convergence rate of the sequence of the objective functions $\{L_\lambda(\boldsymbol{\beta}^k), k = 1, 2, \ldots\}$, but not the sequence of the solutions $\{\boldsymbol{\beta}^k, k = 1, 2, \ldots\}$. Since in high-dimensional settings with $p \gg n$, the global minimizers $\widehat{\boldsymbol{\beta}}_\lambda$ are generally not unique, hence, it is not clear which minimizer the sequence $\{\boldsymbol{\beta}^k, k = 1, 2, \ldots\}$ generated from CD-GS iterations converges to. The CD-GS proposed in [15] and [21] with refined sweep rules is guaranteed to converge. Other widely used algorithms include proximal gradient descent [22–25], alternative direction method of multiplier (ADMM) [26–28], among others. For more comprehensive reviews of the literature on the related topics, see the review papers by [29], and [30].

[24] considered the statistical properties of the proximal gradient descent path. But their analysis required knowing $\|\boldsymbol{\beta}^\dagger\|_1$, which is unknown or hard to estimate in practice. Although this can be remedied by using the techniques developed by [25], it does not achieve the sharp error bound as PSNA does.

1.3. Notation

Some notations used throughout this paper are defined below. With $\|\boldsymbol{\beta}\|_q = (\sum_{i=1}^p |\beta_i|^q)^{\frac{1}{q}}$ we denote the usual q ($q \in [1, \infty]$) norm of a vector $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)' \in \mathbb{R}^p$. $\|\boldsymbol{\beta}\|_0$ denotes the number of nonzero elements of $\boldsymbol{\beta}$. X' denotes the transpose of the covariate matrix $X \in \mathbb{R}^{n\times p}$ and $\|X\|$ denotes the operator norm of X induced by vector with 2-norm. $\mathbf{1}$ or $\mathbf{0}$ denote a vector $\in \mathbb{R}^p$ or a matrix with elements all 1 or 0. Define $S = \{1, \ldots, p\}$. For any $A, B \subseteq S$ with length $|A|, |B|$, we denote $\boldsymbol{\beta}_A \in \mathbb{R}^{|A|}$(or $X_A \in \mathbb{R}^{|A|\times p}$) as the subvector (or submatrix) whose entries (or columns) are listed in A. X_{AB} denotes submatrix of X whose rows and columns are listed in A and B, respectively. We use supp(z), sgn(z) to denote the support and sign of a vector $\boldsymbol{z}$, respectively. We use I, G and $\widetilde{\boldsymbol{y}}$ to denote the identity matrix, the regularized Gram matrix $X'X + \alpha I$ and $X'\boldsymbol{y}$, respectively.

1.4. Organization

In Section 2 we provide a heuristic and intuitive derivation of SNA for solving (1.3) (including (1.2) as a special case by setting $\alpha = 0$) and describe SNA for pathwise optimization (PSNA). In Section 3 we establish the locally superlinear convergence rate of SNA for (1.3) and local one-step convergence for (1.2), and analyze the computational complexity of SNA. In Section 4 we provide the conditions for the finite-step sign consistency of PSNA and the upper bounds for the estimation error. In Section 5 we discuss the relations of SNA with LARS, SIS, and active set methods for accelerating coordinate descent. The implementation detail and numerical comparison with LARS and coordinate descent methods are given in Section 6. We conclude in Section 7 with some comments and future work. The proofs of the main results and some background on Newton derivatives used for deriving SNA are included in the appendices.

2. A general description of PSNA

In this section we first give an intuitive description of the SNA for computing the Lasso and Enet solutions at a given λ and α. We then describe the PSNA, which uses SNA for computing the solution paths with warm start and a continuation strategy.

2.1. Motivating SNA based on the KKT conditions

The key idea in the proposed algorithm is to iteratively identify the active set in the optimization using both the primal and dual information, then solve the problem on the active set. Here the primal is simply $\boldsymbol{\beta}$ and its dual is $\boldsymbol{d} = (\widetilde{\boldsymbol{y}} - G\boldsymbol{\beta})/n$. Recall $\widetilde{\boldsymbol{y}} = X'\boldsymbol{y}$ and $G = X'X + \alpha I$. For the Lasso, the expression of $\boldsymbol{d}$ simplifies to $\boldsymbol{d} = X'(\boldsymbol{y} - X\boldsymbol{\beta})/n$, i.e., the correlation vector between the predictors and the residual. For any given (λ, α), the KKT conditions (Proposition 3.3) assert that $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$ is the unique Enet solution if and

only if the pair $\{\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}, \widehat{\boldsymbol{d}}_{\lambda,\alpha}\}$ satisfies

$$\begin{cases} \widehat{\boldsymbol{d}}_{\lambda,\alpha} = (\widetilde{\boldsymbol{y}} - G\widehat{\boldsymbol{\beta}}_{\lambda,\alpha})/n, \\ \widehat{\boldsymbol{\beta}}_{\lambda,\alpha} = T_\lambda(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} + \widehat{\boldsymbol{d}}_{\lambda,\alpha}), \end{cases} \tag{2.1}$$

where $T_\lambda(\boldsymbol{x})$ is the soft-threshold operator [31] acting on $\boldsymbol{x}$ component wise, that is, $T_\lambda(\boldsymbol{x}) = (T_\lambda(x_1), \ldots, T_\lambda(x_p))'$ with

$$T_\lambda(x) = x - \frac{|x+\lambda|}{2} + \frac{|x-\lambda|}{2}, x \in \mathbb{R}. \tag{2.2}$$

The KKT conditions in (2.1) are stated in equalities using the soft-threshold operator, rather than in the usual inequality form or in terms of set-valued subdifferentials. This is the basis for our derivation of the SNA, which seeks to solve these nonsmooth equations.

To simplify the notation we drop the subscripts of $(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}, \widehat{\boldsymbol{d}}_{\lambda,\alpha})$ and write them as $(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{d}})$, when it does not cause any confusion. By the second equation of (2.1) and the definition of the soft-threshold operator, we have

$$\widehat{\boldsymbol{\beta}}_B = \boldsymbol{0}, \tag{2.3}$$

$$\widehat{\boldsymbol{d}}_A = \lambda \operatorname{sgn}(\widehat{\boldsymbol{\beta}}_A + \widehat{\boldsymbol{d}}_A), \tag{2.4}$$

where

$$A = \left\{ j \in S : |\widehat{\beta}_j + \widehat{d}_j| \geq \lambda \right\} \text{ and } B = \left\{ j \in S : |\widehat{\beta}_j + \widehat{d}_j| < \lambda \right\}. \tag{2.5}$$

Substituting (2.3) into the first equation of (2.1) and observing G_{AA} is invertible, we can solve the resulting linear system to get

$$\widehat{\boldsymbol{\beta}}_A = G_{AA}^{-1}(\widetilde{\boldsymbol{y}}_A - n\widehat{\boldsymbol{d}}_A), \tag{2.6}$$

$$\widehat{\boldsymbol{d}}_B = (\widetilde{\boldsymbol{y}}_B - G_{BA}\widehat{\boldsymbol{\beta}}_A)/n. \tag{2.7}$$

Therefore, $\{\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{d}}\}$ can be obtained from (2.3)-(2.4) and (2.6)-(2.7) if A is known.

Let $\{\boldsymbol{\beta}^k, \boldsymbol{d}^k\}$ be the primal and dual approximation of $\{\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{d}}\}$ at the kth iteration. Based on (2.5), we approximate the active and inactive sets by

$$A_k = \left\{ j \in S : |\beta_j^k + d_j^k| > \lambda \right\} \text{ and } B_k = \left\{ j \in S : |\beta_j^k + d_j^k| \leq \lambda \right\}. \tag{2.8}$$

Based on (2.3)-(2.4) and (2.6)-(2.7) we obtain the updated approximation $\{\boldsymbol{\beta}^{k+1}, \boldsymbol{d}^{k+1}\}$,

$$\boldsymbol{\beta}_{B_k}^{k+1} = \boldsymbol{0}, \tag{2.9}$$

$$\boldsymbol{d}_{A_k}^{k+1} = (\lambda - \overline{\lambda})\operatorname{sgn}(\boldsymbol{\beta}_{A_k}^k + \boldsymbol{d}_{A_k}^k), \tag{2.10}$$

$$\boldsymbol{\beta}_{A_k}^{k+1} = G_{A_kA_k}^{-1}(\widetilde{\boldsymbol{y}}_{A_k} - n\boldsymbol{d}_{A_k}^{k+1}), \tag{2.11}$$

$$\boldsymbol{d}_{B_k}^{k+1} = (\widetilde{\boldsymbol{y}}_{B_k} - G_{B_kA_k}\boldsymbol{\beta}_{A_k}^{k+1})/n. \tag{2.12}$$

In (2.10) we introduce a (small) shift parameter $\overline{\lambda}$ with $0 \leq \overline{\lambda} < \lambda$ and use a slightly more general version of (2.4), replacing λ with $\lambda - \overline{\lambda}$ in (2.4). For $\overline{\lambda} > 0$, we solve a less shrunk version of the Enet. For the solution sequence $\{\boldsymbol{\beta}^k, k \geq 1\}$ with a suitable $\overline{\lambda} > 0$, we show that it achieves finite-step sign consistency and sharp estimation error bound (Theorem 4.1).

Based on the above discussion, we summarize SNA for minimizing (1.3) in Algorithm 1 below, where we write $\widehat{\boldsymbol{\beta}}(\lambda) = \widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$ for a fixed α.

Algorithm 1 $(\widehat{\boldsymbol{\beta}}(\lambda), \widehat{\boldsymbol{d}}(\lambda)) \longleftarrow \text{SNA}(\boldsymbol{\beta}^0, \boldsymbol{d}^0, \lambda, \overline{\lambda}, K)$

1: Input: $X, \boldsymbol{y}, \alpha, \lambda, \overline{\lambda}, K$, initial guess $\boldsymbol{\beta}^0, \boldsymbol{d}^0$, $A_{-1} = \text{supp}(\boldsymbol{\beta}^0)$. Set $k = 0$.
2: Compute $\widetilde{\boldsymbol{y}} = X'\boldsymbol{y}$ and store it.
3: **for** $k = 0, 1, \cdots, K$ **do**
4: Compute A_k, B_k using (**??**).
5: If $A_k = A_{k-1}$ or $k \geq K$. Stop and denote the last iteration by $\boldsymbol{\beta}_{\widehat{A}}, \boldsymbol{\beta}_{\widehat{B}}, \boldsymbol{d}_{\widehat{A}}, \boldsymbol{d}_{\widehat{B}}$.Else
6: Compute $\{\boldsymbol{\beta}^{k+1}, \boldsymbol{d}^{k+1}\}$ using (**??**) - (**??**). $k := k + 1$.End
7: **end for**
8: Output: $\widehat{\boldsymbol{\beta}}(\lambda) = \begin{pmatrix} \boldsymbol{\beta}_{\widehat{A}} \\ \boldsymbol{\beta}_{\widehat{B}} \end{pmatrix}$ and $\widehat{\boldsymbol{d}}(\lambda) = \begin{pmatrix} \boldsymbol{d}_{\widehat{A}} \\ \boldsymbol{d}_{\widehat{B}} \end{pmatrix}$

Remark 2.1. In the algorithm, we use a safeguard maximum number of iterations K that can be defined by the user. We usually set $K \leq 5$ due to the locally superlinear/one-step convergence of SNA.

Each line in Algorithm 1 consists of simple vector and matrix multiplications, except (2.11) in line 6, where we need to invert a $|A_k| \times |A_k|$ matrix. Note that A_k is usually a small subset of S if Algorithm 1 is warm started. Intuitively, at the kth step in the iteration, this algorithm tries to identify A_k, an approximation of the underlying support by using the estimated coefficients with a proper adjustment $\boldsymbol{d}^k$ determined by the KKT, and solves a low-dimensional adjusted least squares problem on A_k. Therefore, with a good starting estimation of A_k, which is guaranteed by using a continuation strategy with warm start described below, Algorithm 1 can find a good solution in a few steps. In Section 3, we derive Algorithm 1 formally from the semismooth Newton method and show that its convergence rate is locally superlinear for the Enet and locally one-step for the Lasso.

2.2. Solution path approximation

We are often interested in the whole solution path $\widehat{\boldsymbol{\beta}}(\lambda) \equiv \widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$ of (1.3) for $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ and some given $\alpha \geq 0$. Here we approximate the solution path by computing $\widehat{\boldsymbol{\beta}}(\lambda)$ on a given finite set $\Lambda = \{\lambda_0, \lambda_1 ..., \lambda_N\}$ for some integer N, where $\lambda_0 > \cdots > \lambda_N > 0$. Obviously, $\widehat{\boldsymbol{\beta}}(\lambda) = \boldsymbol{0}$ satisfies (2.1) and (2.1) if $\lambda \geq \|X'\boldsymbol{y}/n\|_\infty$. Hence we set $\lambda_{\max} = \lambda_0 = \|X'\boldsymbol{y}/n\|_\infty$, $\lambda_t = \lambda_0\gamma^t, t = 0, 1, ..., N$, and $\lambda_{\min} = \lambda_0\gamma^N$, where $\gamma \in (0, 1)$.

We adopt a simple continuation technique with warm start in computing the solution path. This strategy has been successfully used for computing the Lasso and Enet paths [13,32]. We use the solution at λ_t as the initial value for computing the solution at λ_{t+1}. The shift parameter $\overline{\lambda}$ can vary at different path knots λ_t, so here we use $\overline{\lambda}_t$ to demonstrate this. We summarize this in the following PSNA algorithm

When running PSNA with warm start, SNA usually converges in a few steps, since SNA converges locally superlinearly and warm start provides a good initial value.

3. Derivation of SNA and convergence analysis

3.1. KKT conditions

In this subsection, we first discuss the relationship between the minimizers of (1.2) and (1.3). We then characterize the unique minimizer (1.3) by its KKT system.

Proposition 3.1. *Let M_λ be the set of the Lasso solutions given in (1.2). Then M_λ is nonempty, convex and compact.*

In general, the Lasso solution is not unique in the $p \gg n$ settings, however, the one in M_λ with the minimum Euclidean norm is unique. We denote this solution by $\widehat{\boldsymbol{\beta}}_\lambda$.

Proposition 3.2. *For $\alpha > 0$, the Enet (1.3) admits a unique minimizer denoted by $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$. Furthermore, $\|\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} - \widehat{\boldsymbol{\beta}}_\lambda\|_2 \to 0$ as $\alpha \to 0^+$.*

By Proposition 3.2, a good numerical solution of (1.3) is a good approximation of the minimum 2-norm minimizer of (1.2) for a sufficiently small α.

Proposition 3.3. *Let $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} \in \mathbb{R}^p$ be the Enet solution, which is the unique minimizer of $J_{\lambda,\alpha}$ in (1.3) for $\alpha > 0$. Then there exists a $\widehat{\boldsymbol{d}}_{\lambda,\alpha} \in \mathbb{R}^p$ such that (2.1) hold. Conversely, if there exists $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} \in \mathbb{R}^p$ and $\widehat{\boldsymbol{d}}_{\lambda,\alpha} \in \mathbb{R}^p$ satisfying (2.1), then $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$ is the unique minimizer of $J_{\lambda,\alpha}$ in (1.3).*

The KKT equations (2.1) with $\alpha = 0$ also characterize the Lasso solution (1.2), except that the solution may not be unique.

Here the KKT conditions are formulated in terms of equalities (2.1), which are different from but equivalent to the usual inequality form

$$\widehat{\boldsymbol{d}}_A = \lambda \operatorname{sgn}(\widehat{\boldsymbol{\beta}}_A),$$

$$\|\widehat{\boldsymbol{d}}_{A^c}\|_\infty \le \lambda,$$

where, $\widehat{\boldsymbol{d}} = (X'\boldsymbol{y} - G\widehat{\boldsymbol{\beta}})/n$ and $A = \operatorname{supp}(\widehat{\boldsymbol{\beta}})$. The reason we adopt the equation form (2.1) is that we can transform the minimization problem (1.3) into a root finding problem, which helps us derive SNA formally as a semismooth Newton method.

3.2. SNA as a Newton algorithm

We now formally derive the SNA based on the KKT conditions by using the semismooth Newton method [4–6] for finding a root of a nonsmooth equation. This enables us to prove its locally superlinear convergence stated in Theorem 3.2 below. The definition and related property on Newton derivative are given in Appendix A.

Let

$$\boldsymbol{z} = \begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{d} \end{pmatrix} \quad \text{and} \quad F(\boldsymbol{z}) = \begin{bmatrix} F_1(\boldsymbol{z}) \\ F_2(\boldsymbol{z}) \end{bmatrix} : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^{2p},$$

where

$$F_1(\boldsymbol{z}) := \boldsymbol{\beta} - T_\lambda(\boldsymbol{\beta} + \boldsymbol{d}),$$

$$F_2(\boldsymbol{z}) := G\boldsymbol{\beta} + n\boldsymbol{d} - \widetilde{\boldsymbol{y}}.$$

By Proposition 3.3, to find the minimizer of (1.3), it suffices to find a root of $F(\boldsymbol{z})$. Although the classical Newton algorithm cannot be applied directly since $F(\boldsymbol{z})$ is not Fréchet differentiable, we can resort to semismooth Newton algorithm since $F(\boldsymbol{z})$ is Newton differentiable.

Let

$$A := \{i \in S : |\beta_i + d_i| \ge \lambda\}, \quad B := \{i \in S : |\beta_i + d_i| < \lambda\}.$$

We reorder $(\boldsymbol{\beta}', \prime)'$ such that $\boldsymbol{z} = (\boldsymbol{d}_A', \boldsymbol{\beta}_B', \boldsymbol{\beta}_A', \boldsymbol{d}_B')'$. We also reorder $F_1(\boldsymbol{z})$ and $F_2(\boldsymbol{z})$ accordingly,

$$F(\boldsymbol{z}) = \begin{bmatrix} \boldsymbol{\beta}_A - T_\lambda(\boldsymbol{\beta}_A + \boldsymbol{d}_A) \\ \boldsymbol{\beta}_B - T_\lambda(\boldsymbol{\beta}_B + \boldsymbol{d}_B) \\ G_{AA}\boldsymbol{\beta}_A + G_{AB}\boldsymbol{\beta}_B + n\boldsymbol{d}_A - \widetilde{\boldsymbol{y}}_A \\ G_{BA}\boldsymbol{\beta}_A + G_{BB}\boldsymbol{\beta}_B + n\boldsymbol{d}_B - \widetilde{\boldsymbol{y}}_B \end{bmatrix}.$$

We have the following result concerning the Newton derivative of F.

Theorem 3.1. *$F(\boldsymbol{z})$ is Newton differentiable at any point $\boldsymbol{z}$. And*

$$H := \begin{bmatrix} -I_{AA} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & I_{BB} & \boldsymbol{0} & \boldsymbol{0} \\ nI_{AA} & X_A'X_B & G_{AA} & \boldsymbol{0} \\ \boldsymbol{0} & G_{BB} & X_B'X_A & nI_{BB} \end{bmatrix} \in \nabla_N F(\boldsymbol{z}). \tag{3.1}$$

Furthermore, H is invertible and H^{-1} is uniformly bounded with

$$\|H^{-1}\| \le 1 + 2(n + 1 + \alpha + \|X\|^2)^2/\alpha.$$

At the k_{th} iteration, the semismooth Newton method for finding the root of $F(\boldsymbol{z}) = \boldsymbol{0}$ consists of two steps.

(1) Solve $H_k D^k = -F(\boldsymbol{z}^k)$ for D^k, where H_k is an element of $\nabla_N F(\boldsymbol{z}^k)$.
(2) Update $\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + D^k$, set $k \leftarrow k + 1$ and go to step (1).

This has the same form as the classical Newton method, except that here we use an element of $\nabla_N F(Z^k)$ in step (1). Indeed, the key to the success of this method is to find a suitable and invertible H_k. We state this method in Algorithm 3.

Algorithm 3 SNA for finding a root of $F(\boldsymbol{z})$

1: Input: $X, \boldsymbol{y}, \lambda, \alpha$, initial guess $\boldsymbol{z}^0 = \begin{pmatrix} \boldsymbol{\beta}^0 \\ \boldsymbol{d}^0 \end{pmatrix}$. Set $k = 0$.
2: **for** $k = 0, 1, 2, 3, \cdots$ **do**
3: Choose $H_k \in \nabla_N F(\boldsymbol{z}^k)$.
4: Get the semismooth Newton directionD_k by solving

$$H_k D^k = -F(\boldsymbol{z}^k). \tag{17}$$

5: Update

$$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + D^k. \tag{18}$$

6: Check Stop conditionIf stopDenote the last iteration by $\widehat{\boldsymbol{z}}$.Else$k := k + 1$.
7: **end for**
8: Output:$\widehat{\boldsymbol{z}}$as an estimate of the roots of $F(\boldsymbol{z})$.

Remark 3.1. When $A_k = A_{k+1}$ holds for some k, Algorithm 1 converges. Hence it is natural to stop Algorithm 1 accordingly. A common condition that can be used as a stop rule of Algorithm 3 is when $\|F(\boldsymbol{z}^k)\|_2$ is sufficiently small, since this algorithm is a root finding process. Therefor we can use both stopping rules in Algorithm 1 due to the equivalence of Algorithm 1 and Algorithm 3. We also stop Algorithm 1 when the iteration number k exceeds a prespecified integer K.

It can be verified that Algorithm 1 with $\overline{\lambda} = 0$ is the same as Algorithm 3. However, Algorithm 1 is written in a form that is easier to implement computationally. The details are given in Appendix C. Thus it is indeed a semismooth Newton method. The more compact form of Algorithm 3 is better suited for its convergence analysis.

Theorem 3.2. *Let H_k in Algorithm 3 be given in (A.42). Then the sequence $\{\boldsymbol{\beta}^k, k = 1, 2, \ldots\}$ generated based on Algorithm 3 (and Algorithm 1 with $\overline{\lambda} = 0$) converges locally and superlinearly to $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$, the unique minimizer of (1.3).*

Theorem 3.2 shows the local supperlinear convergence rate of SNA, which is a superior property of Newton-type algorithms to first-order methods.

Theorem 3.3. *For a give $\lambda > 0$, let $\widehat{\boldsymbol{\beta}} \equiv \widehat{\boldsymbol{\beta}}_\lambda$ be a minimizer of (1.2), $\widehat{\boldsymbol{d}} = X'(\boldsymbol{y} - X\widehat{\boldsymbol{\beta}})/n$, $A = \{i : |\widehat{\beta}_i + \widehat{d}_i| \geq \lambda\}$, $\widetilde{A} = \{i : |\widehat{\beta}_i + \widehat{d}_i| \neq \lambda\}$, and $C = \min_{i\in\widetilde{A}} ||\widehat{\beta}_i + \widehat{d}_i| - \lambda| > 0$. Suppose $\operatorname{rank}(X_A) = |A|$ and the initial guess $\boldsymbol{\beta}^0, \boldsymbol{d}^0$ satisfies $\|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_\infty + \|\widehat{\boldsymbol{d}} - \boldsymbol{d}^0\|_\infty \leq C$. Then, $\boldsymbol{\beta}^1 = \widehat{\boldsymbol{\beta}}$, where $\boldsymbol{\beta}^1$ is generated by Algorithm 3 with $\alpha = 0$ and $\overline{\lambda} = 0$.*

Theorem 3.3 shows that the SNA has the optimal local convergence rate, since it converges in just one step. This is obviously the best possible convergence rate due to the special structure of (1.2) and improves the locally superliner convergence rate of semismooth Newton method in general situations, see for example, [4], [5] and [6].

3.3. Computational complexity analysis

We now consider the computational complexity of SNA (Algorithm 1). We look at the number of floating-point operations per iteration. Clearly it takes $O(p)$ flops to finish steps 4-7 in Algorithm 1. For step 8, we solve the linear equation iteratively by the conjugate gradient (CG) method initialized with the projection of the previous solution onto the current active set [33]. The main operations in CG include two matrix-vector multiplications, which take $2n|A_k|$ flops. Therefore, the number of CG iterations is smaller than $p/(2|A_{k+1}|)$ and there are at most $O(np)$ flops in step 8. For step 9, calculation of the matrix-vector product costs np flops. So, the total cost per iteration in Algorithm 1 is $O(np)$, which is also the cost for the state-of-the-art first-order Lasso solvers. The local superlinear/one-step convergence of SNA guaranteed that a good solution can be found in only a few iterations if it is warm started. Therefore, at each knot on the path, the whole cost of SNA can be still $O(np)$ if we use the continuation strategy. Thus we can use Algorithm 2 (PSNA) to compute the solution path accurately and efficiently with $O(Nnp)$ flops, where N is the number of knots on the path. See the numerical results in Section 6.

4. Error bounds and finite-step sign consistency

As shown in Theorems 3.2 and 3.3, SNA converges locally superlinearly for Enet and converges in one step for Lasso. In this section we prove that the simple warm start technique makes the PSNA converge globally under certain mutual coherence conditions on X and a condition on the minimum magnitude of the nonzero components of $\boldsymbol{\beta}^\dagger$. Specifically, we show that PSNA hits a solution with the same sign as $\boldsymbol{\beta}^\dagger$ and attains a sharp statistical error bound in finitely many steps with high probability, if we properly design the path $\{\lambda_t = \lambda_0\gamma^t\}_{t=0,1,..N}$ and run SNA along it with warm start.

We only consider the Lasso, so we set $\alpha = 0$ and $G = X'X$. The mutual coherence ν defined as $\nu = \max_{i\neq j} |G_{i,j}|/n$ [34,35] characterizes the minimum angle between different columns of $X/\sqrt{n}$. Let $A^\dagger = \operatorname{supp}(\boldsymbol{\beta}^\dagger)$ and $T = |A^\dagger|$. Define $|\boldsymbol{\beta}^\dagger|_{\min} = \min\{|\beta^\dagger_j| : j \in A^\dagger\}$, Denote the universal threshold value by $\lambda_u = \sigma\sqrt{2\log(p)/n}$. Let $\delta_u = 3\lambda_u$, $\lambda_0 = \|X'y/n\|_\infty$, and $\lambda_t = \lambda_0\gamma^t$, $t = 0, 1, ...$

We make the following assumptions on the design matrix X, the target coefficient $\boldsymbol{\beta}^\dagger$, and the noise vector $\boldsymbol{\eta}$.

(A1) The mutual coherence satisfies $T\nu \leq \frac{1}{4}$.
(A2) The smallest nonzero regression coefficient satisfies $|\boldsymbol{\beta}^\dagger|_{min} \geq 78\lambda_u$.
(A3) $\boldsymbol{\eta}$ satisfies $\boldsymbol{\eta} \sim N(0, \sigma^2 I_n)$.

Lemma 4.1. *Suppose that (A1) to (A3) hold. There exists an integer $N \in [1, \log_\gamma(\frac{10\delta_u}{\lambda_0}))$ such that $\lambda_N > 10\delta_u \geq \lambda_{N+1}$ and $|\boldsymbol{\beta}^\dagger|_{min} > 8\lambda_N/5$ hold with probability at least $1 - 1/(2\sqrt{\pi\log(p)})$.*

Theorem 4.1. *Suppose that (A1) to (A3) hold. Then with probability at least $1 - 1/(2\sqrt{\pi\log(p)})$, PSNA$(\lambda_0, \gamma, N, K)$ with $\gamma = 8/13$, N determined in Lemma 4.1, $K \geq T$, and $\overline{\lambda} = \frac{9}{10}\lambda_t + \delta_u$ at the t_{th} knot, has a finite step sign consistence property and achieves a sharp estimation error, i.e.,*

$$\operatorname{sgn}(\widehat{\boldsymbol{\beta}}(\lambda_N)) = \operatorname{sgn}(\boldsymbol{\beta}^\dagger), \tag{4.1}$$

and

$$\|\widehat{\boldsymbol{\beta}}(\lambda_N) - \boldsymbol{\beta}^\dagger\|_\infty < \frac{23}{6}\lambda_u. \tag{4.2}$$

Remark 4.1. From the proof of Theorem 4.1 we can see that PSNA (Algorithm 3) with $\overline{\lambda} = 0$ can recover $\boldsymbol{\beta}^\dagger$ exactly by letting $\lambda_t \to 0$ in the case $\boldsymbol{\eta} = \mathbf{0}$. However, if the observation contains noise we have to set the shift parameter $\overline{\lambda}$ in PSNA to be nonzero which reduce the amount of shrinkage of Lasso.

The properties of Lasso have been studied by many authors. For example, [36] and [37] showed that Lasso is sign consistent under a strong irrepresentable condition, which is a little weaker than (A1). They also required $|\boldsymbol{\beta}^\dagger|_{min}$ to be bounded below by $O(n^{-c/2})$ with $c \in (0, 1)$, which is a stronger assumption than (A2). [38] required X satisfy a sparse Rieze condition, which may be weaker than (A1); and $|\boldsymbol{\beta}^\dagger|_{min}$ larger than $O(\sqrt{T}\lambda_u)$, which is stronger than (A2). [39] assumed a condition stronger than the strong irrepresentable condition to guarantee the uniqueness of Lasso and its sign consistency with a condition on $|\boldsymbol{\beta}^\dagger|_{min}$ similar to (A1). [40] and [41] assumed the mutual coherence conditions with $T\nu < 1/7$ and $\nu < c/\log(p)$ for a constant c, respectively; and their requirements for $|\boldsymbol{\beta}^\dagger|_{min}$ are similar to (A1) with different constants. In deriving the ℓ_2 and ℓ_∞ error bounds of the Lasso, [35] and [42] assumed $T\nu < 1/4$ and $T\nu \leq 1/4$, respectively. The latter is exactly (A1). However, these existing results do not imply the finite-step sign consistency property established in Theorem 4.1.

All the results mentioned above concern the minimizer of the Lasso problem, but they did not directly address the statistical properties of the sequence generated by a specific solver. So there is a gap between those theoretical results and the computational solutions. There has been efforts to close this gap. For example, [24] considered the statistical properties of the proximal gradient descent path. However, their analysis required knowing $\|\boldsymbol{\beta}^\dagger\|_1$, which is hard to estimate in practice. [25] remedied this, but their result does not achieve the sharp error bound like PSNA does. Also, the technique used for deriving the statistical properties of PSNA (a Newton-type method), is quite different from the proximal gradient method (a gradient type method).

5. Comparison with LARS, SIS and an active set method for accelerating coordinate descent

The key idea in PSNA is using the Newton-type method SNA to iteratively identify the active set using both the primal and dual information, then solve the problem on the active set. In this section we discuss the connections between PSNA with other three dual active set methods, i.e., LARS, SIS [10], and sequential strong rule SSR (active set tricks for accelerating coordinate descent) [11].

LARS [8] also does not solve (1.2) exactly since it omits the removing procedure of Homotopy [7]. As discussed in [9], the LARS algorithm can be formulated as

$$\boldsymbol{\beta}^{k+1}_{B_k} = \mathbf{0},$$

$$\boldsymbol{\beta}^{k+1}_{A_k} = (X'_{A_k}X_{A_k})^{-1}(\widetilde{\boldsymbol{y}}_{A_k} - \overline{\lambda}^k \operatorname{sgn}(\boldsymbol{d}^k_{A_k})),$$

where A_k is the set of the indices of the variables with highest correlation with the current residual, $B_k = (A_k)^c$, $\overline{\lambda}^k = \|\boldsymbol{d}^k\|_\infty - \gamma_k$, $\boldsymbol{d}^k = X'(\boldsymbol{y} - X\boldsymbol{\beta}^k)/n$, and γ_k is the step size to the next breakpoint on the path [8,9]. Comparing Algorithm 2 (PSNA) (by setting $K = 0$)

Algorithm 2 $\widehat{\boldsymbol{\beta}}(\Lambda) \longleftarrow \text{PSNA}(\lambda_0, \gamma, N, K)$

1: Input: $\lambda_0 = \|X'\boldsymbol{y}/n\|_\infty, \widehat{\boldsymbol{\beta}}(\lambda_{-1}) = \mathbf{0}, \widehat{\boldsymbol{d}}(\lambda_{-1}) = X'\boldsymbol{y}/n, \gamma, N, K.$
2: **for** $t = 0, 1...N.$ **do**
3: Set $\lambda_t = \lambda_0\gamma^t$ and $(\boldsymbol{\beta}^0, \boldsymbol{d}^0) = (\widehat{\boldsymbol{\beta}}(\lambda_{t-1}), \widehat{\boldsymbol{d}}(\lambda_{t-1}))$.
4: $(\widehat{\boldsymbol{\beta}}(\lambda_t), \widehat{\boldsymbol{d}}(\lambda_t)) \longleftarrow \text{SNA}(\boldsymbol{\beta}^0, \boldsymbol{d}^0, \lambda_t, \overline{\lambda}_t, K)$
5: **end for**
6: Output: $\widehat{\boldsymbol{\beta}}(\Lambda) = [\widehat{\boldsymbol{\beta}}(\lambda_0), ..., \widehat{\boldsymbol{\beta}}(\lambda_N)]$.

with the above reformulation of the LARS algorithm, we see that both PSNA and LARS can be understood as approaches for estimating the support of the underlying solution, which is the essential aspect in fitting sparse, high-dimensional models. So, PSNA and LARS share some similarity both in formulation and in spirit although they were derived from different perspectives. However, the definitions of the active set in PSNA are based on the sum of primal approximation (current approximation $\boldsymbol{\beta}^k$) and the dual approximation (current correlation $\boldsymbol{d}^k = X'(\boldsymbol{y} - X\boldsymbol{\beta}^k)/n$) while LARS is based on dual only. The following low-dimensional small noise interpretation may clarify the difference between the two active set definitions. If $X'X/n \approx$ identity and $\boldsymbol{\eta} \approx 0$ we get

$$\boldsymbol{d}^k = X'(\boldsymbol{y} - X\boldsymbol{\beta}^k)/n = X'(X\boldsymbol{\beta}^\dagger + \boldsymbol{\eta} - X\boldsymbol{\beta}^k)/n \approx \boldsymbol{\beta}^\dagger - \boldsymbol{\beta}^k + X'\boldsymbol{\eta}/n \approx \boldsymbol{\beta}^\dagger - \boldsymbol{\beta}^k$$

and

$$\boldsymbol{\beta}^k + \boldsymbol{d}^k \approx \boldsymbol{\beta}^\dagger.$$

In addition, LARS selects variables one by one while PSNA can select more than one variable at each iteration. Also, the adjusted least squares fits on the active sets in PSNA and LARS are different. Under certain conditions on X both of them recover $\boldsymbol{\beta}^\dagger$ exactly in the noise-free case [9] even when $p > n$. But the convergence or consistency of LARS is unknown when the noise vector $\boldsymbol{\eta} \neq \mathbf{0}$.

Given a starting point λ_0, SNA is initialized with $\boldsymbol{\beta}^0 = \mathbf{0}, \boldsymbol{d}^0 = X'\boldsymbol{y}/n$. Therefore,

$$A_0 = \{j : |\beta_j^0 + d_j^0| > \lambda_0\} = \{j : |\boldsymbol{x}_j'\boldsymbol{y}/n| > \lambda_0\}.$$

Thus the first active set generated by PSNA contains the features that coorelated with y larger than λ_0, which are the same as those from the sure independence screening [10] with parameter λ_0 and include the one selected by the first step of LARS. We then use (2.9) - (2.12) to obtain $\{\boldsymbol{\beta}^1, \boldsymbol{d}^1\}$, and update the active set to A_1 using (2.8). Clearly, for $k \geq 1$, A_k are determined not just by the correlation $\boldsymbol{d}^k$, but by the primal ($\boldsymbol{\beta}^k$) and dual ($\boldsymbol{d}^k$) together.

[11] proposed a sequential strong rule (SSR) for discarting predictors in Lasso-type problems. At point λ_t on the solution path, this rule discards the jth predictor if

$$|\widehat{d}_j(\lambda_{t-1})| < 2\lambda_t - \lambda_{t-1},$$

where $\widehat{d}_j(\lambda) = \boldsymbol{x}_j'(\boldsymbol{y} - X\widehat{\boldsymbol{\beta}}(\lambda))/n$ for the Lasso penalty. They define active set

$$A_k = \{j : |d_j(\lambda_{t-1})| \geq 2\lambda_t - \lambda_{t-1}\},$$

and set $\widehat{\boldsymbol{\beta}}(\lambda_t)_{B_k} = 0$ for $B_k = A_k^c$ and solve the Lasso problem on A_k. By combining with a simple check of the KKT condition, it speeds up the computation considerably. So SNA shares some similarity in spirit with SSR in that both methods seek to identify an active set and solve a smaller optimization problem, although they are derived from quite different perspectives. However, there are some important differences. First, the active sets are determined differently. Specifically, SSR determines the active set only based on the dual approximation; while SNA uses both primal and dual approximation. Second, SNA does not need the unit slope assumption, and additional check of the KKT conditions is not needed (The cost of check KKT is $O(np)$). Third, as far as we know, the statistical properties of the solution sequence generated from SSR are unknown, while error bounds and sign consistency are established under suitable conditions for the solution sequence generated from the PSNA.

6. Numerical studies

In this section, we present numerical examples to evaluate the performance of the proposed PSNA algorithm 2 for solving Lasso. We have implemented PSNA in a Matlab package *psna*, which is available at https://github.com/jian94/psna. 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).

6.1. Comparison with existing popular algorithms

Both the LARS [8,9] and the CD [13,43] are popular algorithms capable of efficiently computing the Lasso solution, hence we compare the proposed PSNA with these two algorithms. In the implementation, we consider two solvers: (1) SolveLasso, the Matlab code for LARS with the Lasso modification, available online at http://sparselab.stanford.edu/SparseLab_files/Download_files/SparseLab21-Core.zip ; (2) glmnet, the Fortran based Matlab package using CD, available online at https://github.com/distrep/DMLT/tree/master/external/glmnet. The parameters in the solvers are the default values as their online versions. In addition to the default stopping parameters in the solvers, we stop LARS (SolveLasso), CD (glmnet) and PSNA if the number of nonzero elements at some iteration is larger than a given fixed quantity such as $n/\log(p)$ or even larger $0.5n$, since the upper bound of the estimated sparsity level of Lasso is $O(n/\log(p))$ when $n \ll p$ [44,45].

6.2. Tuning parameter selection

To choose a proper value of λ in (1.2) is a crucial issue for Lasso problems, since it balances the tradeoff between the data fidelity and the sparsity level of the solution. In practice, the Bayesian information criterion (BIC) is a widely used selector for the tuning parameter selection, due to its model selection consistency under some regularity conditions. We refer the readers to [46–51] and references therein for more details. In this paper, we use a modified BIC (MBIC) from [50] to choose λ, which is given as

$$\widehat{\lambda} = \arg\min_{\lambda \in \Lambda} \left\{ \frac{1}{2n}\|X\widehat{\boldsymbol{\beta}}(\lambda) - \boldsymbol{y}\|_2^2 + |\widehat{A}(\lambda)| \frac{\log(n)\log(p)}{n} \right\}, \tag{6.1}$$

where $\Lambda = \{\lambda_t\}_t$ is the candidate set for λ, and $\widehat{A}(\lambda) = \{j : \widehat{\beta}(\lambda) \neq 0\}$ is the model identified by $\widehat{\boldsymbol{\beta}}(\lambda)$. Besides, the high-dimensional BIC (HBIC) in [51] defined by

$$\widehat{\lambda} = \arg\min_{\lambda \in \Lambda} \left\{ \log\left(\|X\widehat{\boldsymbol{\beta}}(\lambda) - \boldsymbol{y}\|_2^2/n\right) + |\widehat{A}(\lambda)| \frac{\log(\log n)\log(p)}{n} \right\} \tag{6.2}$$

is also a good candidate for the selection of λ. Unless otherwise specified, the MBIC (6.1) is the default one to select λ.

6.3. Simulation

6.3.1. Implementation

The $n \times p$ design matrix X is generated as follows.

(i) Classical Gaussian matrix with correlation parameter ρ. The rows of X are drawn independently from $N(0, \Sigma)$ with $\Sigma_{jk} = \rho^{|j-k|}$, $1 \leq j, k \leq p$, $\rho \in (0, 1)$.

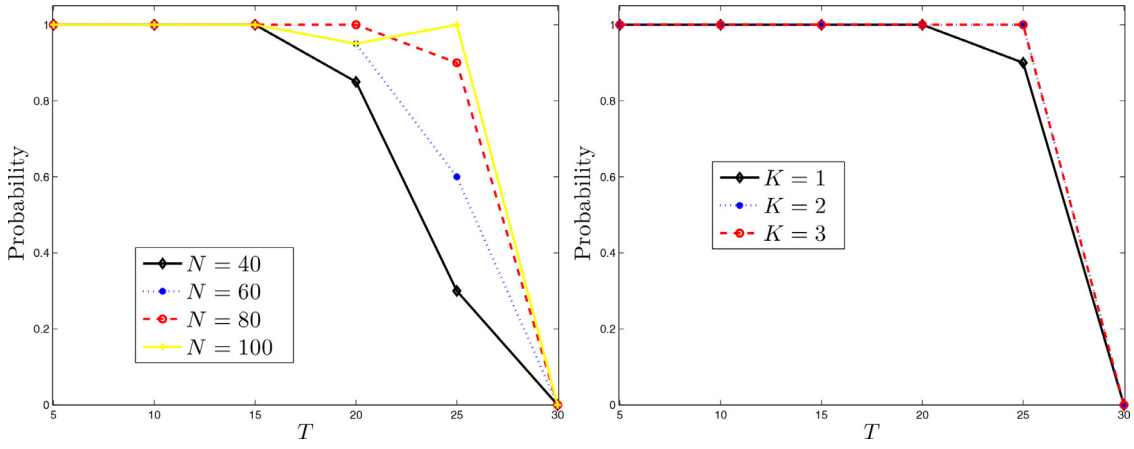

Fig. 1. The influence of the PSNA parameters N (left panel) and K (right panel) on the exact support recovery probability.

(ii) Random Gaussian matrix with auto-correlation parameter ν. First we generate a random Gaussian matrix $\widetilde{X} \in \mathbb{R}^{n\times p}$ with its entries following i.i.d. $N(0,1)$. Then we define a matrix $X \in \mathbb{R}^{n\times p}$ by setting $X_1 = \widetilde{X}_1$,

$$X_j = \widetilde{X}_j + \nu * (\widetilde{X}_{j-1} + \widetilde{X}_{j+1}),\ \ j = 2, ..., p-1,$$

and $X_p = \widetilde{X}_p$.

The elements of the error vector $\boldsymbol{\eta}$ are generated independently with $\eta_i \sim N(0, \sigma^2)$, $i = 1, 2, ..., n$. Let $A^\dagger = \mathrm{supp}(\boldsymbol{\beta}^\dagger)$ be the support of $\boldsymbol{\beta}^\dagger$, and let $R^\dagger = \max\{|\boldsymbol{\beta}^\dagger_{A^\dagger}|\}/\min\{|\boldsymbol{\beta}^\dagger_{A^\dagger}|\}$ be the range of magnitude of nonzero elements of $\boldsymbol{\beta}^\dagger$. The underlying regression coefficient vector $\boldsymbol{\beta}^\dagger \in \mathbb{R}^p$ is generated in a way that $A^\dagger$ is a randomly chosen subset of S with $|A^\dagger| = T$. As in [52], [53] and [54], each nonzero entry of $\boldsymbol{\beta}^\dagger$ is generated as follows:

$$\beta^\dagger_j = \xi_{1j} 10^{\xi_{2j}}, \tag{6.3}$$

where $j \in A^\dagger$, $\xi_{1j} = \pm 1$ with probability $\frac{1}{2}$ and ξ_{2j} is uniformly distributed in [0,1]. Then the observation vector $\boldsymbol{y} = X\boldsymbol{\beta}^\dagger + \boldsymbol{\eta}$. For convenience, we use $(n, p, \rho, \sigma, T, R^\dagger)$ and $(n, p, \nu, \sigma, T, R^\dagger)$ to denote the data generated as above, respectively.

6.3.2. The performance of PSNA

The algorithm parameters of PSNA. We study the influence of the free parameters N and K in the PSNA algorithm on the exact support recovery probability (Probability for short), that is, the percentage of the estimated model $\widehat{A}$ agrees with the true model $A^\dagger$. To this end, we independently generate 20 datasets from $(n = 200, p = 1000, \rho = 0.1, \sigma = 0.01, T = 5:5:30, R^\dagger = 10)$ for each combination of (N, K). Here $5:5:30$ means the sparsity level starts from 5 to 30 with an increment of 5. The numerical results are summarized in Fig. 1, which consider the following two settings: (a) $K = 1$, and varying $N \in \{40, 60, 80, 100\}$; (b) $N = 100$, and varying $K \in \{1, 2, 3\}$.

It is observed from Fig. 1 that the influence of K is very mild on the exact support recovery probability and $K = 1$ generally works well in practice, due to the locally superlinear convergence of SNA and the continuation technique with warm start on the solution path, which is consistent with the conclusions in Section 2. It is also found in Fig. 1 that Larger N values make the algorithm have better exact support recovery probability, but the enhancement decreases as N increases. Thus, unless otherwise specified, we set $(N, K) = (100, 1)$ for the PSNA solver.

The MBIC selector for PSNA. We illustrate the performance of the MBIC selector (6.1) for PSNA with simulated data $(n = 400, p = 2000, \rho = 0.5, \sigma = 0.1, T = 10, R^\dagger = 10)$. The results are summarized in Fig. 2. It can be observed from Fig. 2 that the MBIC selector performs very well for the PSNA algorithm on the continuation solution path introduced in Section 2.2. *The local superlinear convergence of PSNA.* To gain further insight into the PSNA algorithm, we illustrate the convergence behavior of the algorithm using the simulated data as that of Fig. 2. Let $\widehat{A}_t = \{j : \widehat{\beta}_j(\lambda_t) \neq 0\}$, where $\widehat{\boldsymbol{\beta}}(\lambda_t)$ is the solution to the λ_t-problem. Set $(N, K) = (100, 5)$. The convergence history is shown in Fig. 3, which presents the change of the active sets and the number of iterations for each fixed λ_t along the path $\lambda_0 > \lambda_1 > \cdots > \widehat{\lambda}$. It is observed in Fig. 3 that $\widehat{A}_t \subset A^\dagger$, and the size $|\widehat{A}_t|$ increases monotonically as the path proceeds and eventually equals the true model size $|A^\dagger|$. In particular, for each λ_{t+1} problem with $\widehat{\boldsymbol{\beta}}(\lambda_t)$ as the initial guess, PSNA generally reaches convergence within two iterations (typically one, noting that the maximum number of iterations $K = 5$ here). This is attributed to the local superlinear/one-step convergence of the algorithm for Lasso, which is consistent with the results in Theorem 3.3. Hence, the overall procedure is very efficient. *The accuracy versus the sparsity level.* We now consider the influence of the sparsity level T on the performance of PSNA, LARS and CD in terms of the exact support recovery probability. Data are generated from the model with $(n = 1000, p = 2000, \rho = 0.2, \sigma = 0.1, T = 5:5:100, R^\dagger = 1)$. The results are summarized in Fig. 4. As shown in the figure, our method performs well when the sparsity level varies from small to large.

6.3.3. Efficiency and accuracy

To further evaluate the efficiency and accuracy of the proposed PSNA algorithm, we independently generate $M = 100$ datasets from two settings: (i) the classical Gaussian matrix with $(n, p, \rho, \sigma, T, R^\dagger) = (600, 3000, 0.3:0.2:0.7, 0.2:0.2:0.4, 40, 10)$; (ii) the random Gaussian matrix with $(n, p, \nu, \sigma, T, R^\dagger) = (1000, 10000, 0.3:0.2:0.7, 0.2:0.2:0.4, 50, 10)$. Based on M independent runs, we compare PSNA with CD and LARS in terms of the average CPU time (Time, in seconds), the estimated average model size (MS) $M^{-1}\sum_{m=1}^M |\widehat{A}^{(m)}|$, the proportion of correct models (CM, in percentage terms) $M^{-1}\sum_{m=1}^M I\{\widehat{A}^{(m)} = A^\dagger\}$, the average ℓ_∞ absolute error (AE) $M^{-1}\sum_{m=1}^M \|\widehat{\boldsymbol{\beta}}^{(m)} - \boldsymbol{\beta}^\dagger\|_\infty$, and the average ℓ_2 relative error (RE) $M^{-1}\sum_{m=1}^M \left(\|\widehat{\boldsymbol{\beta}}^{(m)} - \boldsymbol{\beta}^\dagger\|_2/\|\boldsymbol{\beta}^\dagger\|_2\right)$. The measure Time reflects the efficiency of the solvers, while measures MS, CM, AE and RE evaluate the accuracy (quality) of the solutions. Simulation results are summarized in Table 1 and Table 2, respectively.

For each (ρ, σ) combination, it can be observed from Table 1 that PSNA has better speed performance than CD and LARS.

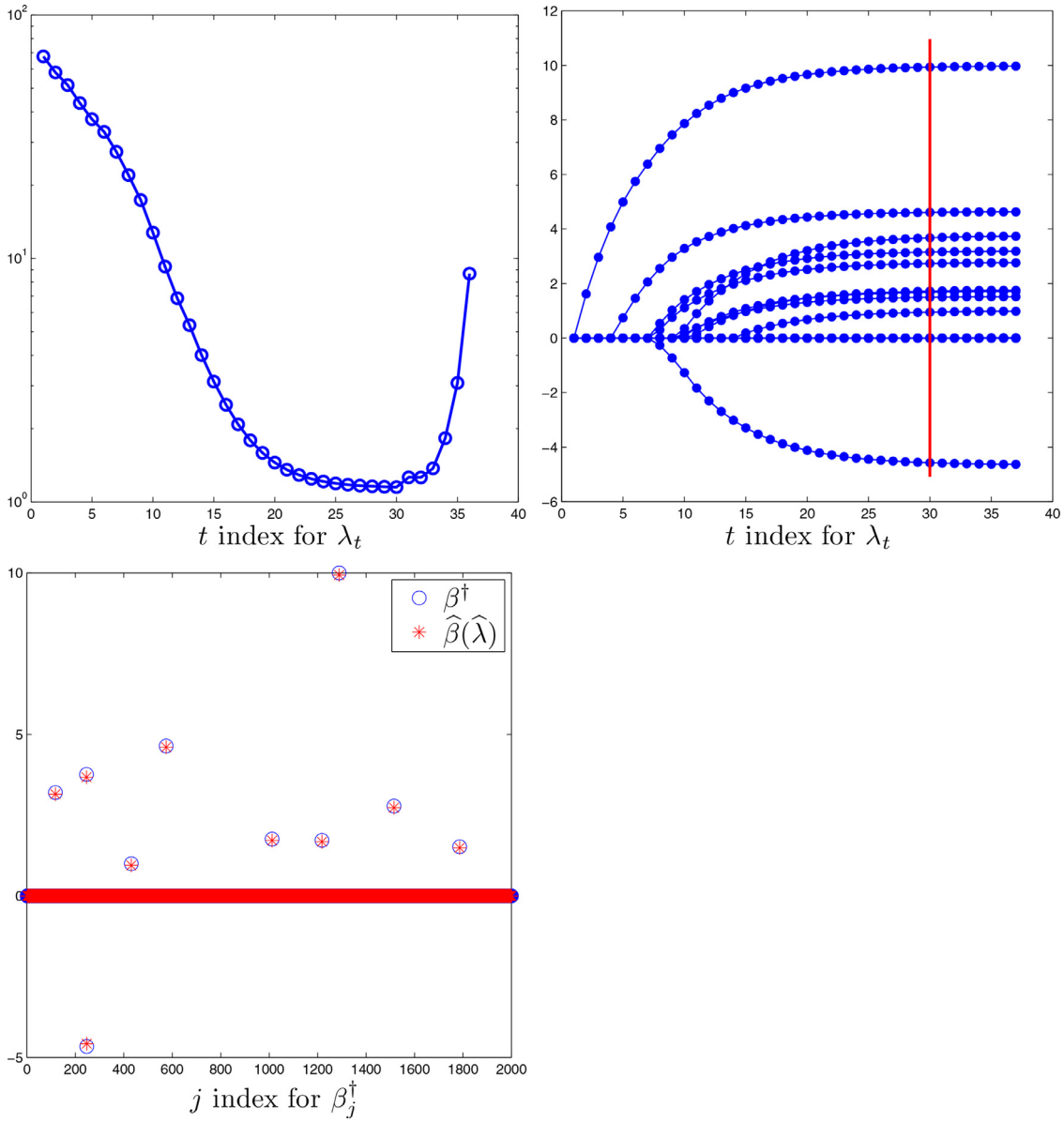

Fig. 2. Plots for PSNA using the MBIC selector with data ($n = 400, p = 2000, \rho = 0.5, \sigma = 0.1, T = 10, R^\dagger = 10$): MBIC curve (top left panel), the solution path (top right panel), and the comparison between the underlying true parameter $\boldsymbol{\beta}^\dagger$ and the selected solution $\widehat{\boldsymbol{\beta}}(\widehat{\lambda})$ (bottom left panel). The red vertical line in the middle panel shows the solutions selected by MBIC.

With ρ fixed, the CPU time of CD and PSNA slightly decreases as σ increases, while higher σ increases the timing of LARS in general. Given σ, the CPU time of CD and PSNA is relatively robust with respect to ρ, while that of LARS generally increases as ρ increases. According to MS, all solvers tend to overestimate the true model and PSNA usually selects a smaller model, while PSNA can select the correct model far more frequently than CD and LARS in terms of CM. The errors of all solvers AE and RE are small, which means they all can produce estimates that are very close to the true values of $\boldsymbol{\beta}^\dagger$, while the AE and RE of PSNA are smaller than that of the other two, indicating that PSNA is generally more accurate than CD and LARS. Unsurprisingly, larger ρ or σ will degrade the accuracy of all solvers. In addition, it is shown from Table 1 that PSNA generally has smaller ((or comparable) standard errors, especially in accuracy metrics MS, AE and RE, which means the results of PSNA are stable and robust. Similar phenomena also hold for the random Gaussian matrix setting in Table 2. In particular, since the value of p is large in Table 2, the timing advantage of PSNA is more obvious, which implies that PSNA is capable of handling much larger data sets. In summary, PSNA behaves very well in simulation studies and generally outperforms the state-of-the-art solvers such as LARS and CD in terms of both efficiency and accuracy.

6.4. Application

We analyze the breast cancer data which comes from breast cancer tissue samples deposited to The Cancer Genome Atlas (TCGA) project and compiles results obtained using Agilent mRNA expression microarrays to illustrate the application of the PSNA algorithm in high-dimensional settings. This data, which is named bcTCGA, is available at https://portal.gdc.cancer.gov/. In this bcTCGA dataset, we have expression measurements of 17814 genes from 536 patients (all expression measurements are recorded on the log scale). There are 491 genes with missing data, which we have excluded. We restrict our attention to the 17323 genes without missing values. The response variable $\boldsymbol{y}$ measures one of the

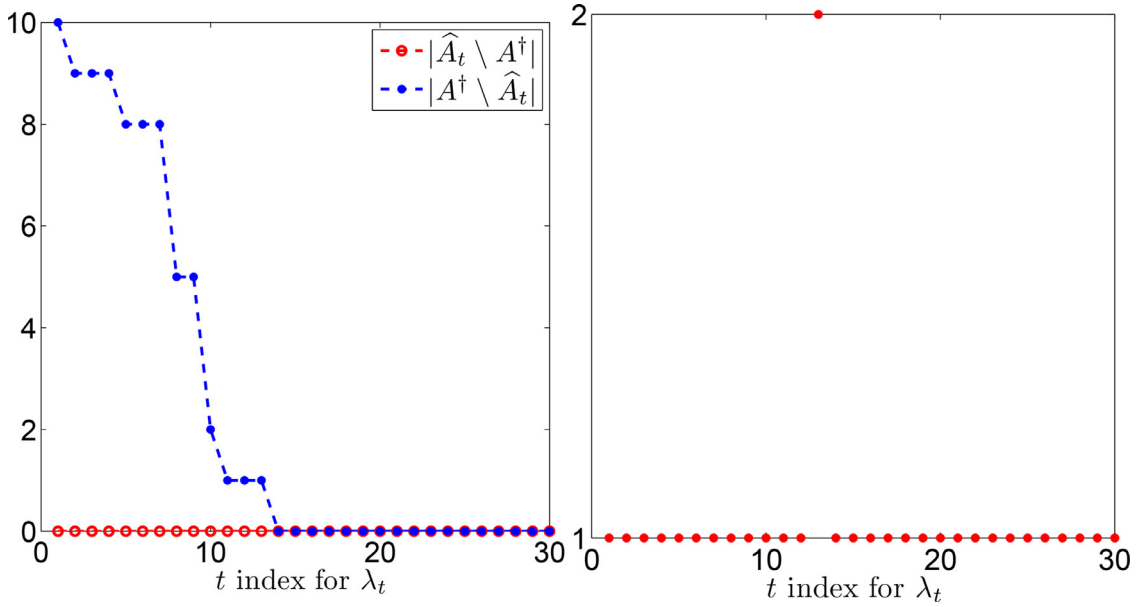

Fig. 3. Convergence behavior of PSNA with data ($n = 400$, $p = 2000$, $\rho = 0.5, \sigma = 0.1, T = 10, R^\dagger = 10$): the change of the active sets (left panel) and the number of iterations (right panel) for each λ_t-problem along the path. $\widehat{A}_t \setminus A^\dagger$ ($A^\dagger \setminus \widehat{A}_t$) denotes the set difference of sets $\widehat{A}_t$ and $A^\dagger$ ($A^\dagger$ and $\widehat{A}_t$). In the left panel, the vertical axis is the size of sets; in the right panel, the vertical axis is the number of iterations.

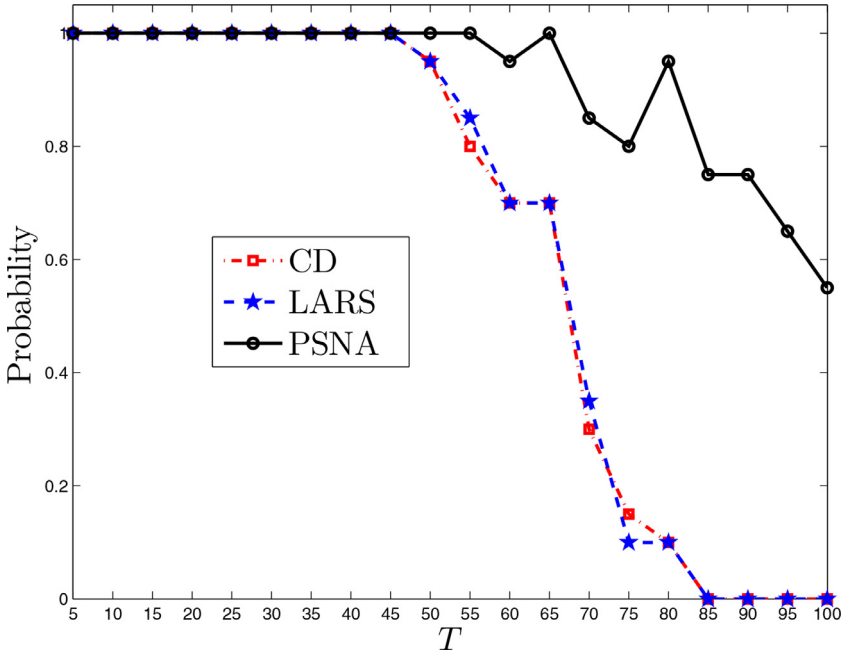

Fig. 4. The influence of the sparsity level T on the exact support recovery probability of the solvers.

17323 genes, a numeric vector of length 536 giving expression level of gene BRCA1, which is the first gene identified that increases the risk of early onset breast cancer, and the design matrix X is a 536×17322 matrix, which represents the remaining expression measurements of 17322 genes. Because BRCA1 is likely to interact with many other genes, it is of interest to find genes with expression levels related to that of BRCA1. This has been studied by using different methods in the recent literature; see, for example, [54–58]. In this subsection, we apply methods CD (glmnet), LARS (SolveLasso) and PSNA, coupled with the HBIC selector, to analyze this dataset.

First, we analyze the complete dataset of 536 patients. The genes selected by each method along with their corresponding nonzero coefficient estimates, the CPU time (Time, in seconds), the model size (MS) and the prediction error (PE) calculated by $n^{-1}\sum_{i=1}^{n}(\widehat{y}_i - y_i)^2$ are provided in Table 3. It can be seen from Table 3 that PSNA runs faster than LARS and CD, while the PE by PSNA is smaller than that by LARS and CD, which demonstrates that PSNA performs better than the other two solvers in terms of both efficiency and accuracy. Further, CD, LARS and PSNA identify 7, 9 and 4 genes respectively, with 3 identified probes in common, namely, C17orf53, NBR2 and TIMELESS. Although the magnitudes of estimates for the common genes are not equal, they have the same signs, which suggests similar biological conclusions.

To further evaluate the performance of the three methods, we implement the cross validation (CV) procedure similar to

Table 1
Simulation results for the classical Gaussian matrix with $n = 600$, $p = 3000$, $T = 40$ and $R^{\dagger} = 10$ based on 100 independent runs. The numbers in the parentheses are the corresponding standard errors.

ρ	σ	Method	Time	MS	CM	AE	RE
0.3	0.2	CD	0.2287(0.0060)	41.35(1.1135)	23%(0.4230)	0.1270(0.0286)	0.0172(0.0042)
		LARS	0.2066(0.0336)	41.25(1.1315)	29%(0.4560)	0.1208(0.0273)	0.0163(0.0039)
		PSNA	0.1784(0.0111)	40.07(0.2932)	94%(0.2387)	0.0808(0.0192)	0.0105(0.0030)
	0.4	CD	0.2233(0.0023)	42.71(1.5973)	8%(0.2727)	0.2041(0.0349)	0.0275(0.0047)
		LARS	0.2121(0.0316)	42.14(3.0978)	13%(0.3380)	0.2672(0.6802)	0.0344(0.0767)
		PSNA	0.1569(0.0041)	40.23(0.4894)	80%(0.4020)	0.1528(0.0309)	0.0200(0.0046)
0.5	0.2	CD	0.2272(0.0025)	42.74(1.8836)	11%(0.3145)	0.1484(0.0454)	0.0193(0.0057)
		LARS	0.2086(0.0318)	42.53(2.0863)	15%(0.3589)	0.1808(0.2411)	0.0223(0.0317)
		PSNA	0.1746(0.0036)	40.19(0.4861)	84%(0.3685)	0.0925(0.0319)	0.0117(0.0039)
	0.4	CD	0.2236(0.0027)	44.48(2.1057)	0%(0.0000)	0.2333(0.0616)	0.0299(0.0066)
		LARS	0.2162(0.0364)	43.58(5.0835)	1%(0.1000)	0.3518(0.9235)	0.0427(0.1029)
		PSNA	0.1548(0.0039)	40.62(0.8138)	55%(0.5000)	0.1693(0.0439)	0.0213(0.0045)
0.7	0.2	CD	0.2280(0.0023)	47.96(3.2315)	0%(0.0000)	0.2062(0.0956)	0.0223(0.0061)
		LARS	0.2325(0.0317)	47.85(3.4855)	0%(0.0000)	0.3177(0.6256)	0.0255(0.0219)
		PSNA	0.1737(0.0047)	41.25(1.2340)	34%(0.4761)	0.1274(0.0596)	0.0136(0.0040)
	0.4	CD	0.2249(0.0023)	50.59(3.6517)	0%(0.0000)	0.3323(0.1500)	0.0349(0.0081)
		LARS	0.2437(0.0310)	50.61(5.4028)	0%(0.0000)	0.5283(0.7663)	0.0463(0.0679)
		PSNA	0.1561(0.0040)	41.92(1.4885)	16%(0.3685)	0.2337(0.0914)	0.0245(0.0054)

Table 2
Simulation results for the random Gaussian matrix with $n = 1000$, $p = 10000$, $T = 50$ and $R^{\dagger} = 10$ based on 100 independent runs. The numbers in the parentheses are the corresponding standard errors.

ν	σ	Method	Time	MS	CM	AE	RE
0.3	0.2	CD	1.6607(0.0185)	51.59(1.5511)	26%(0.4408)	0.0902(0.0229)	0.0121(0.0027)
		LARS	1.0458(0.1729)	51.48(1.5007)	29%(0.4560)	0.0859(0.0215)	0.0116(0.0027)
		PSNA	0.8685(0.0289)	50.08(0.3075)	93%(0.2564)	0.0553(0.0133)	0.0072(0.0017)
	0.4	CD	1.6416(0.0074)	52.76(1.8916)	9%(0.2876)	0.1403(0.0314)	0.0184(0.0031)
		LARS	1.1354(0.2245)	52.40(2.0792)	12%(0.3266)	0.1560(0.2001)	0.0204(0.0272)
		PSNA	0.7764(0.0149)	50.28(0.5519)	76%(0.4292)	0.1007(0.0217)	0.0128(0.0022)
0.5	0.2	CD	1.6719(0.0058)	55.71(2.6678)	0%(0.0000)	0.0959(0.0583)	0.0111(0.0028)
		LARS	1.1819(0.2301)	55.59(3.3937)	0%(0.0000)	0.2002(0.4392)	0.0164(0.0377)
		PSNA	0.8980(0.0108)	50.82(1.0767)	49%(0.5024)	0.0559(0.0320)	0.0064(0.0019)
	0.4	CD	1.6504(0.0064)	57.86(3.0847)	0%(0.0000)	0.1583(0.1068)	0.0164(0.0044)
		LARS	1.3180(0.2388)	56.85(4.2530)	0%(0.0000)	0.1954(0.4346)	0.0220(0.0584)
		PSNA	0.8098(0.0172)	51.20(1.1192)	31%(0.4648)	0.1091(0.0674)	0.0112(0.0028)
0.7	0.2	CD	1.6962(0.0100)	65.22(4.6113)	0%(0.0000)	0.1411(0.1521)	0.0114(0.0060)
		LARS	1.4585(0.2683)	64.90(7.2202)	0%(0.0000)	0.5015(1.0090)	0.0249(0.0411)
		PSNA	0.9439(0.0506)	53.09(1.7529)	6%(0.2387)	0.0858(0.1374)	0.0068(0.0100)
	0.4	CD	1.6715(0.0090)	67.12(4.8996)	0%(0.0000)	0.1880(0.2146)	0.0156(0.0082)
		LARS	1.5399(0.2448)	67.07(6.4499)	0%(0.0000)	0.5493(0.9422)	0.0271(0.0299)
		PSNA	0.8441(0.0889)	52.84(3.3536)	6%(0.2387)	0.1907(0.5365)	0.0187(0.0611)

Table 3
The genes identified by CD, LARS and PSNA that correlated with BRCA1 based on the complete dataset of bcTCGA ($n = 536$, $p = 17322$). The zero entries correspond to variables omitted.

No.	Term	Gene	CD	LARS	PSNA
	Intercept		-1.0865	-1.0217	-0.4985
1	β_{1743}	C17orf53	0.1008	0.0983	0.4140
2	β_{2739}	CCDC56	0	0.0108	0
3	β_{2964}	CDC25C	0	0.0136	0
4	β_{4543}	DTL	0.0764	0.0844	0
5	β_{9230}	MFGE8	0	0	-0.1168
6	β_{9941}	NBR2	0.1519	0.1885	0.4673
7	β_{12146}	PSME3	0.0480	0.0615	0
8	β_{15122}	TIMELESS	0.0157	0.0279	0.2854
9	β_{15535}	TOP2A	0.0259	0.0331	0
10	β_{16315}	VPS25	0.1006	0.1083	0
	Time		3.5436	2.1070	0.7884
	MS		7	9	4
	PE		0.3298	0.3023	0.2345

Table 4
The CPU time (Time), model size (MS) and prediction error (PE) averaged across 100 random partitions of the bcTCGA data (numbers in parentheses are standard deviations)

Method	Time	MS	PE
CD	2.0598(0.0393)	8.72(3.3667)	0.3503(0.0764)
LARS	0.5861(0.3625)	7.90(3.5689)	0.3514(0.0831)
PSNA	0.6554(0.0906)	6.10(3.0830)	0.2742(0.0593)

[54–57,59,60]. We conduct 100 random partitions of the data. For each partition, we randomly choose 3/4 observations and 1/4 observations as the training and test data, respectively. We compute the CPU time (Time, in seconds) and the model size (MS, i.e., the number of selected genes) using the training data, and calculate the prediction error (PE) based on the test data. Table 4 presents the average values over 100 random partitions, along with corresponding standard deviations in the parentheses.

Due to the CV procedure, the working sample size decreases to $n_{CV} = \frac{3}{4}n$. Hence, the CPU time of three solvers in Table 4 decreases accordingly compared with the counterpart in Table 3. Obviously, it is shown in Table 4 that PSNA is still running faster than CD, and is quite comparable to LARS in speed. Compared to LARS, the CPU time of PSNA is less sensitive to the sample size, which means PSNA has more potential than LARS to be applied to a larger volume of noisy data. Also, as clearly shown in the Table 4, PSNA selects fewer genes and has a smaller PE, which implies that PSNA could provide a more targeted list of the gene sets. Based on 100 random partitions, we report the selected genes and their corre-

Table 5
Frequency table for 100 random partitions of the bcTCGA data. To save space, only the genes with Freq ≥ 5 are listed.

CD		LARS		PSNA	
Gene	Freq	Gene	Freq	Gene	Freq
C17orf53	98	C17orf53	93	NBR2	95
DTL	91	NBR2	89	C17orf53	91
NBR2	91	VPS25	87	DTL	32
VPS25	90	DTL	86	MFGE8	29
PSME3	77	PSME3	73	CCDC56	25
TOP2A	73	TOP2A	67	CDC25C	23
TIMELESS	49	TIMELESS	41	TUBG1	23
CCDC56	41	CCDC56	33	LMNB1	18
CDC25C	35	CDC25C	30	GNL1	18
CENPK	26	CENPK	24	TIMELESS	16
SPRY2	20	RDM1	20	VPS25	16
SPAG5	18	CDC6	17	TOP2A	15
RDM1	18	TUBG1	15	ZYX	14
TUBG1	17	SPRY2	15	KIAA0101	14
CDC6	17	C16orf59	12	KHDRBS1	12
UHRF1	16	CCDC43	11	PSME3	11
C16orf59	13	UHRF1	10	SPAG5	10
CCDC43	13	SPAG5	10	TUBA1B	8
ZWINT	9	NSF	9	FGFRL1	8
KIAA0101	9	KIAA0101	8	CMTM5	7
NSF	8	ZWINT	5	SYNGR4	5
MLX	6				
TRAIP	5				

sponding frequency (Freq) in Table 5, where the genes are ordered such that the frequency is decreasing. To save space, we only list genes with frequency greater than or equal to 5 counts. It is observed from Table 5 that some genes such as NBR2, C17orf53, DTL and VPS25 have quite high frequencies (Freq ≥ 80) with all three solvers, which largely implies these genes are related to BRCA1. Combining the findings in Table 5 and taking into account the small MS and PE of PSNA in Table 4, we have a strong belief that genes NBR2 and C17orf53 selected by PSNA are particularly associated with BRCA1.

7. Concluding Remarks

Starting from the KKT conditions we developed SNA for computing the Lasso and Enet solutions in high-dimensional linear regression models. We approximate the whole solution paths using PSNA by utilizing the continuation technique with warm start. PSNA is easy to implement, stable, fast and accurate. We established the locally superlinear of SNA for the Enet and local one-step convergence for the Lasso. We provided sufficient conditions under which SANP enjoys the sign consistency property in finite steps. Moreover, PSNA has the same computational complexity as LARS and CD. Our simulation studies demonstrate that PSNA is competitive with these state-of-the-art solvers in accuracy and outperforms them in efficiency. These theoretical and numerical results suggest that PSNA is a promising new method for dealing with large-scale ℓ_1-regularized linear regression problems.

We have only considered the linear regression model with convex penalties. It would be interesting to generalize PSNA to other models such as the generalized linear and Cox models. It would also be interesting to extend the idea of PSNA to problems with nonconvex penalties such as SCAD [61] and MCP [62]. Coordinate descent algorithms for these penalties have been considered by [63] and [64]. In our paper we adopt simple continuation strategy to globalize SNA, globalization via smoothing Newton methods [65–67] is also an interesting future work.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Jian Huang: Conceptualization, Methodology, Writing – review & editing. **Yuling Jiao:** Conceptualization, Methodology, Software, Formal analysis, Writing – original draft, Writing – review & editing. **Xiliang Lu:** Conceptualization, Methodology, Writing – review & editing. **Yueyong Shi:** Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Qinglong Yang:** Validation, Investigation, Writing – review & editing. **Yuanyuan Yang:** Validation, Investigation, Writing – review & editing.

Acknowledgment

The authors are grateful to Professor Defeng Sun and Professor Bangti Jin for the helpful discussions and suggestions on the topics related to this work. J. Huang is supported by the U.S. National Science Foundation grant DMS-1916199. Y. Jiao is supported by the National Natural Science Foundation of China (Grant No. 11871474) and by the research fund of KLATASDSMOE of China, X. Lu is supported by the partially supported by the National Key Research and Development Program of China (No. 2020YFA0714200) and National Natural Science Foundation of China (Grant No. 11871385), Y. Shi is supported by the National Natural Science Foundation of China (Grant Nos. 11501578, 11701571, 11801531 and 41572315), and Q. Yang is supported by the National Natural Science Foundation of China (Grant No. 11671311).

Appendix A. Appendices

A1. Background on convex analysis and Newton derivative

In order to derive the KKT system (2.1) and prove the locally superlinear convergence of Algorithm 1, we recall some background in convex analysis [68] and describe the concept and some properties of Newton derivative [4–6].

The standard Euclidean inner product for two vector $\boldsymbol{z}, \boldsymbol{w} \in \mathbb{R}^p$ is defined by $\langle \boldsymbol{z}, \boldsymbol{w} \rangle := \sum_{i=1}^p z_i w_i$. The class of all proper lower semicontinuous convex functions on $\mathbb{R}^p$ is denoted by $\Gamma_0(\mathbb{R}^p)$. The subdifferential of $f : \mathbb{R}^p \to \mathbb{R}^1$ denoted by ∂f is a set-value mapping defined as

$$\partial f(z) := \{\boldsymbol{w} \in \mathbb{R}^p : f(\boldsymbol{v}) \geq f(\boldsymbol{z}) + \langle \boldsymbol{w}, \boldsymbol{v} - \boldsymbol{z} \rangle, \text{for all} \quad \boldsymbol{v} \in \mathbb{R}^p\}.$$

If f is convex and differentiable it holds that

$$\partial f(\boldsymbol{z}) = \nabla f(\boldsymbol{z}) \tag{A.1}$$

Furthermore, if $f, g \in \Gamma_0(\mathbb{R}^p)$ then

$$\partial (f + g)(\boldsymbol{z}) = \partial f(\boldsymbol{z}) + \partial g(\boldsymbol{z}) \tag{A.2}$$

Recall the classical Fermat's rule [68],

$$\mathbf{0} \in \partial f(\boldsymbol{z}^*) \Leftrightarrow \boldsymbol{z}^* \in \underset{\boldsymbol{z} \in \mathbb{R}^p}{\operatorname{argmin}} \; f(z). \tag{A.3}$$

Moreover, a more general case is [69]

$$\boldsymbol{w} \in \partial f(\boldsymbol{z}) \Leftrightarrow \boldsymbol{z} = \operatorname{Prox}_f(\boldsymbol{z} + \boldsymbol{w}), \tag{A.4}$$

where Prox_f is the proximal operator for $f \in \Gamma_0(\mathbb{R}^p)$ defined as

$$\operatorname{Prox}_f(\boldsymbol{z}) := \underset{\boldsymbol{x} \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{z}\|_2^2 + f(\boldsymbol{x}).$$

Here we should mention that the proximal operator of $\lambda\|\cdot\|_1$ is given in a closed form by the componentwise soft-threshold operator, i.e.,

$$\mathrm{Prox}_{\lambda\|\boldsymbol{x}\|_1}(z) = T_\lambda(\boldsymbol{x}), \tag{A.5}$$

where $T_\lambda(\boldsymbol{x})$ is defined in (2.2).

Let $F:\mathbb{R}^m \to \mathbb{R}^l$ be a nonlinear map. [70] generalized classical Newton's algorithm for finding a root of $F(\boldsymbol{z})=\boldsymbol{0}$ when F is not Fréchet differentiable but only Newton differentiable [6].

Definition 7.1. $F:\mathbb{R}^m \to \mathbb{R}^l$ is called Newton differentiable at $\boldsymbol{x}\in\mathbb{R}^m$ if there exists an open neighborhood $N(\boldsymbol{x})$ and a family of mappings $D: N(x)\to\mathbb{R}^{l\times m}$ such that

$$\|F(\boldsymbol{x}+\boldsymbol{h})-F(\boldsymbol{x})-D(\boldsymbol{x}+\boldsymbol{h})\boldsymbol{h}\|_2 = o(\|\boldsymbol{h}\|_2) \quad \text{for} \quad \|\boldsymbol{h}\|_2 \longrightarrow 0.$$

The set of maps $\{D(\boldsymbol{z}):\boldsymbol{z}\in N(\boldsymbol{x})\}$ denoted by $\nabla_N F(\boldsymbol{x})$ is called the Newton derivative of F at $\boldsymbol{x}$.

It can be easily seen that $\nabla_N F(\boldsymbol{x})$ coincides with the Fréchet derivative at $\boldsymbol{x}$ if F is continuously Fréchet differentiable. An example that is Newton differentiable but not Fréchet differentiable is the absolute function $F(z)=|z|$ defined on $\mathbb{R}^1$. In fact, let $G(z+h)h=\frac{z+h}{|z+h|}h$ and $G(0)h=rh$ with r be any constant in $\mathbb{R}^1$. Then

$$\nabla_N F(z)=\begin{cases}1, & z>0,\\ -1, & z<0,\\ \mathrm{r}\in\mathbb{R}^1, & z=0.\end{cases} \tag{A.6}$$

follows from the definition of Newton derivative.

Suppose $F_i:\mathbb{R}^m\to\mathbb{R}^1$ is Newton differentiable at $\boldsymbol{x}$ with Newton derivative $\nabla_N F_i(\boldsymbol{x})$, $i=1,\dots,l$. Then $F=(F_1,\dots,F_l)'$ is also Newton differentiable at $\boldsymbol{x}$ with Newton derivative

$$\nabla_N F(\boldsymbol{x})=\begin{pmatrix}\nabla_N F_1(\boldsymbol{x})\\ \nabla_N F_2(\boldsymbol{x})\\ \vdots\\ \nabla_N F_l(\boldsymbol{x})\end{pmatrix}. \tag{A.7}$$

Furthermore, if F_1 and F_2 are Newton differentiable at $\boldsymbol{x}$, then the linear combination of them are also Newton differentiable at $\boldsymbol{x}$, i.e., for any $\theta,\gamma\in\mathbb{R}^1$,

$$\nabla_N(\theta F_1+\gamma F_2)(\boldsymbol{x})=\theta\nabla_N F_1(\boldsymbol{x})+\gamma\nabla_N F_2(\boldsymbol{x}). \tag{A.8}$$

Let $F_1:\mathbb{R}^s\to\mathbb{R}^l$ be Newton differentiable with Newton derivative $\nabla_N F_1$. Let $L\in\mathbb{R}^{s\times m}$ and define $F(\boldsymbol{x})=F_1(L\boldsymbol{x}+\boldsymbol{z})$. It can be verified that the chain rule holds, i.e., $F(\boldsymbol{x})$ is Newton differentiable at $\boldsymbol{x}$ with Newton derivative

$$\nabla_N F(\boldsymbol{x})=\nabla_N F_1(L\boldsymbol{x}+\boldsymbol{z})L. \tag{A.9}$$

With the above preparation we can calculate the Newton derivative of the componentwise soft threshold operator $T_\lambda(\boldsymbol{x})$.

Lemma 7.1. *$T_\lambda(\cdot):\mathbb{R}^p\to\mathbb{R}^p$ is Newton differentiable at any point $\boldsymbol{x}\in\mathbb{R}^p$. And $\mathrm{diag}(\boldsymbol{b})\in\nabla_N T_\lambda(\boldsymbol{x})$, where $\mathrm{diag}(\boldsymbol{b})$ is a diagonal matrix with*

$$\boldsymbol{b}=(\mathbf{1}_{\{|x_1|>\lambda\}},\dots,\mathbf{1}_{\{|x_p|>\lambda\}})',$$

and $\mathbf{1}_A$ is the indicator function of set A.

This lemma is used in the derivation of the SNA given in Subsection 3.2.

Proof of Lemma7.1.. As shown in (A.6), $\mathbf{1}_{\{|z|>0\}}\in\nabla_N|z|$. Then, it follows from (A.8)-(A.9) that the scalar function $T_\lambda(z)=z-|z+\lambda|/2+|z-\lambda|/2$ is Newton differentiable by with

$$\mathbf{1}_{\{|z|>\lambda\}}\in\nabla_N T_\lambda(z). \tag{A.10}$$

Let

$$F_i(\boldsymbol{x})=T_\lambda(e_i'\boldsymbol{x}):\boldsymbol{x}\in\mathbb{R}^p\to\mathbb{R}^1,\ i=1,\dots,p,$$

where the column vector e_i is the i_{th} orthonormal basis in $\mathbb{R}^p$. Then, it follow from (A.9) and (A.10) that

$$e_i'\mathbf{1}_{\{|x_i|>\lambda\}}\in\nabla_N F_i(\boldsymbol{x}). \tag{A.11}$$

By using (A.7) and (A.11) we have $T_\lambda(\boldsymbol{x})=(F_1(\boldsymbol{x}),\dots,F_p(\boldsymbol{x}))'$ is Newton differentiable and $\mathrm{diag}\{\boldsymbol{b}\}\in\nabla_N T_\lambda(\boldsymbol{x})$. This completes the proof of Lemma 7.1. □

A2. Proofs

Proof of Proposition 3.1.. This is a standard result in convex optimization, we include a proof here for completeness. Obviously $L_\lambda(\cdot)$ is bounded below by 0, thus, has infimum denoted by L^*. Let $\{\boldsymbol{\beta}^k\}_k$ be a sequence such that $L_\lambda(\boldsymbol{\beta}^k)\to L^*$. Then $\{\boldsymbol{\beta}^k\}_k$ is bounded due to

$$L_\lambda(\boldsymbol{\beta})\to+\infty \quad \text{as} \quad \|\boldsymbol{\beta}\|_1\to+\infty. \tag{A.12}$$

Hence $\{\boldsymbol{\beta}^k\}_k$ has a subsequence still denoted by $\{\boldsymbol{\beta}^k\}_k$ that converge to some $\boldsymbol{\beta}_\lambda$. Then the continuity of $L_\lambda(\cdot)$ implies $\boldsymbol{\beta}_\lambda\in M_\lambda$, i.e., M_λ is nonempty. The boundedness of M_λ follows from (A.12) and the closeness follows from the continuity of $L_\lambda(\cdot)$, i.e., M_λ is compact. The convexity of M_λ follows from the convexity of $L_\lambda(\cdot)$. This completes the proof of Proposition 3.1. □

Proof of Proposition 3.2.. By the same argument in the proof of Proposition 3.1, there exists a minimizer of $J_{\lambda,\alpha}(\cdot)$. We denote this minimizer by $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$. It follow from the strict convexity of $J_{\lambda,\alpha}(\cdot)$ that $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$ is unique. Let $\widehat{\boldsymbol{\beta}}_\lambda$ be the one in M_λ with the minimum Euclidean. We have

$$\begin{aligned}L_\lambda(\widehat{\boldsymbol{\beta}}_\lambda)+\frac{\alpha}{2n}\|\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}\|_2^2 &\le L_\lambda(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha})+\frac{\alpha}{2n}\|\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}\|_2^2 = J_{\lambda,\alpha}(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha})\\ &\le J_{\lambda,\alpha}(\widehat{\boldsymbol{\beta}}_\lambda)=L_\lambda(\widehat{\boldsymbol{\beta}}_\lambda)+\frac{\alpha}{2n}\|\boldsymbol{\beta}_\lambda\|_2^2,\end{aligned} \tag{A.13}$$

where the first inequality use the the property that $\widehat{\boldsymbol{\beta}}_\lambda$ is a minimizer of $L_\lambda(\cdot)$, and the second inequality use the the property that $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$ is a minimizer of $J_{\lambda,\alpha}(\cdot)$. Then it follows from (A.13) that

$$\|\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}\|_2^2\le\|\widehat{\boldsymbol{\beta}}_\lambda\|_2^2. \tag{A.14}$$

This implies $\{\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}\}_\alpha$ is bounded and thus there exist a subsequence of $\{\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}\}_\alpha$ denoted by $\{\overline{\boldsymbol{\beta}}_{\lambda,\alpha}\}_\alpha$ that converge to some $\boldsymbol{\beta}_*$ as $\alpha\to0^+$. Let $\alpha\to0^+$ in (A.13) and (A.14) we get

$$L_\lambda(\boldsymbol{\beta}_*)\le L_\lambda(\widehat{\boldsymbol{\beta}}_\lambda)$$

and

$$\|\boldsymbol{\beta}_*\|_2\le\|\widehat{\boldsymbol{\beta}}_\lambda\|_2.$$

The above two inequality imply $\boldsymbol{\beta}_*$ is a minimizer of $L_\lambda(\cdot)$ with minimum 2-norm. Thus, $\boldsymbol{\beta}_*=\widehat{\boldsymbol{\beta}}_\lambda$ due to the uniqueness of such a minimizer. Hence $\overline{\boldsymbol{\beta}}_{\lambda,\alpha}$ converges to $\boldsymbol{\beta}_\lambda$. The same argument shows that any subsequence of $\{\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}\}_\alpha$ has a further subsequence converging to $\boldsymbol{\beta}_\lambda$. This implies that the whole sequence $\{\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}\}_\alpha$ converges to $\widehat{\boldsymbol{\beta}}_\lambda$. This completes the proof of Proposition 3.2. □

Proof of Proposition 3.3.. We first assume $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}\in\mathbb{R}^p$ is a minimizer of (1.3). Then it follows from (A.1)-(A.3) that

$$\boldsymbol{0}\in X'(X\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}-\boldsymbol{y})/n+\alpha\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}+\lambda\partial\|\cdot\|_1(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}).$$

Therefore, there exists $\widehat{\boldsymbol{d}}_{\lambda,\alpha}\in\lambda\partial\|\cdot\|_1(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha})$ such that

$$\boldsymbol{0}=X'(X\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}-\boldsymbol{y})/n+\alpha\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}+\widehat{\boldsymbol{d}}_{\lambda,\alpha},$$

i.e. the first equation of (2.1) holds by noticing $G=X'X+\alpha I$ and $\widetilde{\boldsymbol{y}}=X'\boldsymbol{y}$. Furthermore, it follow from (A.4) that

$$\widehat{\boldsymbol{d}}_{\lambda,\alpha}\in\lambda\partial\|\cdot\|_1(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha})$$

is equivalent to

$$\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} = \mathrm{Prox}_{\lambda\partial\|\cdot\|_1}(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} + \widehat{\boldsymbol{d}}_{\lambda,\alpha}).$$

By using (A.5), we have

$$\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} = T_\lambda(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} + \widehat{\boldsymbol{d}}_{\lambda,\alpha}),$$

which is the second equation of (2.1).

Conversely, if (2.1) are satisfied for some $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} \in \mathbb{R}^p$, $\widehat{\boldsymbol{d}}_{\lambda,\alpha} \in \mathbb{R}^p$. By using (A.4) and (A.5) again, we deduce

$$\widehat{\boldsymbol{d}}_{\lambda,\alpha} \in \lambda\|\cdot\|_1(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha})$$

from the second equation of (2.1). Substituting this into the first equation of (2.1) we have

$$\mathbf{0} \in (G\widehat{\boldsymbol{\beta}}_{\lambda,\alpha} - \widetilde{\boldsymbol{y}})/n + \lambda\|\cdot\|_1(\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}),$$

which implies that $\widehat{\boldsymbol{\beta}}_{\lambda,\alpha}$ is a minimizer of (1.3) by Fermat's rule (A.3).

The proof for (1.2) can be derived similarly. This completes the proof of Proposition 3.3. □

Proof of Theorem 3.1. It follows from Lemma 7.1 and (A.8)-(A.9) that $F_1(\boldsymbol{z})$ is Newton differentiable. Furthermore, by using Lemma 7.1 and the definition of A and B, we have

$$\begin{bmatrix} -I_{AA} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_{BB} & \mathbf{0} & \mathbf{0} \end{bmatrix} \in \nabla_N F_1(\boldsymbol{z}). \tag{A.15}$$

Obviously, $F_2(\boldsymbol{z})$ is continuously differentiable with

$$\nabla F_2(\boldsymbol{z}) = \begin{bmatrix} nI_{AA} & X_A'X_B & G_{AA} & \mathbf{0} \\ \mathbf{0} & G_{BB} & X_B'X_A & nI_{BB} \end{bmatrix}. \tag{A.16}$$

Then it follows from (A.15)-(A.16) and (A.7) that F is Newton differentiable $\boldsymbol{z}$ with $H \in \nabla_N F(\boldsymbol{z})$.

Let

$$H_1 = \begin{bmatrix} -I_{AA} & \mathbf{0} \\ \mathbf{0} & I_{BB} \end{bmatrix}, H_2 = \begin{bmatrix} nI_{AA} & X_A'X_B \\ \mathbf{0} & G_{BB} \end{bmatrix},$$

$$H_3 = \begin{bmatrix} G_{AA} & \mathbf{0} \\ X_B'X_A & nI_{BB} \end{bmatrix}.$$

Obviously, $H_i, i = 1, 2, 3$ is invertible and

$$H^{-1} = \begin{bmatrix} H_1^{-1} & \mathbf{0} \\ -H_3^{-1}H_2H_1^{-1} & H_3^{-1} \end{bmatrix}.$$

Let $g = (g_1', g_2')'$ be an arbitrary vector in $\mathbb{R}^{2p}$. Then

$$\begin{aligned} \|H^{-1}g\|_2^2 &= \|\begin{bmatrix} H_1^{-1} & \mathbf{0} \\ -H_3^{-1}H_2H_1^{-1} & H_3^{-1} \end{bmatrix}\begin{pmatrix} g_1 \\ g_2 \end{pmatrix}\|_2^2 \\ &= \|H_1^{-1}g_1\|_2^2 + \| - H_3^{-1}H_2H_1^{-1}g_1 + H_3^{-1}g_2\|_2^2 \\ &\le \|H_1^{-1}\|\|g_1\|_2^2 + \|H_3^{-1}\|^2(\|H_2\|\|H_1^{-1}\|\|g_1\|_2 \\ &\quad + \|g_2\|_2)^2 \\ &\le (\|H_1^{-1}\| + \|H_3^{-1}\|(1 + \|H_2\|\|H_1^{-1}\|))^2\|g\|_2^2, \end{aligned}$$

which shows

$$\|H^{-1}\| \le \|H_1^{-1}\| + \|H_3^{-1}\|(1 + \|H_2\|\|H_1^{-1}\|). \tag{A.17}$$

The similar argument shows

$$\|H_2\| \le n + \alpha + 2\|X\|^2, \tag{A.18}$$

and

$$\|H_3^{-1}\| \le 1/n + (1 + \|X\|^2)/\alpha. \tag{A.19}$$

Combining (A.18)-(A.19) with (A.17) and observing $\|H_1^{-1}\| = 1$ we get

$$\|H^{-1}\| < 1 + 2(n + 1 + \alpha + \|X\|^2)^2/\alpha.$$

This completes the proof of Theorem 3.1. □

Proof of Theorem 3.2.. Let $\boldsymbol{z}_\alpha = (\widehat{\boldsymbol{\beta}}_\alpha', \widehat{\boldsymbol{d}}_\alpha')'$ be a root of $F(\boldsymbol{z})$. Let $\boldsymbol{z}^k$ be sufficiently close to $\boldsymbol{z}_\alpha$. By using the definition of Newton derivative and $H_k \in \nabla_N F(\boldsymbol{z}^k)$, we have

$$\|H_k(\boldsymbol{z}^k - \boldsymbol{z}_\alpha) - F(\boldsymbol{z}^k) + F(\boldsymbol{z}_\alpha)\|_2 \le \varepsilon\|\boldsymbol{z}_k - \boldsymbol{z}_\alpha\|_2, \tag{A.20}$$

where $\varepsilon \to 0$ as $\boldsymbol{z}^k \to \boldsymbol{z}_\alpha$. Then,

$$\begin{aligned} &\|\boldsymbol{z}^{k+1} - \boldsymbol{z}_\alpha\|_2 \\ &= \|\boldsymbol{z}^k - H_k^{-1}F(\boldsymbol{z}^k) - \boldsymbol{z}_\alpha\|_2 \\ &= \|\boldsymbol{z}^k - H_k^{-1}F(\boldsymbol{z}^k) - \boldsymbol{z}_\alpha + H_k^{-1}F(\boldsymbol{z}_\alpha)\|_2 \\ &\le \left\|H_k^{-1}\right\|\|H_k(\boldsymbol{z}^k - \boldsymbol{z}_\alpha) - F(\boldsymbol{z}^k) + F(\boldsymbol{z}_\alpha)\|_2 \\ &\le \varepsilon(1 + 2(n + 1 + \alpha + \|X\|^2)^2/\alpha)\|\boldsymbol{z}^k - \boldsymbol{z}_\alpha\|_2, \end{aligned}$$

where the first equality uses () - (), the second equality uses $F(\boldsymbol{z}_\alpha) = \mathbf{0}$, the first inequality is some algebra, and the last inequality uses (A.20) and the uniform boundedness of H_k^{-1} proved in Theorem 3.1. Then we get the sequence $\boldsymbol{z}^k$ generated by Algorithm 3 converge to $\boldsymbol{z}^\alpha$ locally superlinearly. The definition of $F(\boldsymbol{z})$ implies its root $\boldsymbol{z}_\alpha = (\widehat{\boldsymbol{\beta}}_\alpha', \widehat{\boldsymbol{d}}_\alpha')'$ satisfies the KKT conditions (2.1). Thus, it follows from Proposition 3.3 that $\widehat{\boldsymbol{\beta}}_\alpha$ is the unique minimizer of (1.3). Therefore, Theorem 3.2 holds by the equivalence between Algorithm 1 and Algorithm 3. This completes the proof of Theorem 3.2. □

Proof of Theorem 3.3.. First, we have

$$\begin{aligned} &\widehat{\beta}_i + \widehat{d}_i - \beta_i^0 - d_i^0 \\ &\le |\beta_i^0 + d_i^0 - \widehat{\beta}_i - \widehat{d}_i| \\ &\le \|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_\infty + \|\widehat{\boldsymbol{d}} - \boldsymbol{d}^0\|_\infty \\ &\le C \\ &\le \widehat{\beta}_i + \widehat{d}_i - \lambda, \quad \forall i \in \{j \in \widetilde{A} : \widehat{\beta}_j + \widehat{d}_j > \lambda\} \end{aligned}$$

where the last inequality uses the definition that $C = \min_{i\in\widetilde{A}}||\widehat{\beta}_i + \widehat{d}_i| - \lambda|$. This implies that $\widehat{\beta}_i + \widehat{d}_i > \lambda \Longrightarrow \beta_i^0 + d_i^0 > \lambda$ (similarly, we can show $\widehat{\beta}_i + \widehat{d}_i < -\lambda \Longrightarrow \beta_i^0 + d_i^0 < -\lambda$), i.e., $\{i : |\widehat{\beta}_i + \widehat{d}_i| > \lambda\} \subseteq A_0 = \{i : |\beta_i^0 + d_i^0| > \lambda\}$. Meanwhile, by the same argument we can show that $|\widehat{\beta}_i + \widehat{d}_i| < \lambda \Longrightarrow |\beta_i^0 + d_i^0| < \lambda$, i.e., $A_0 \subseteq A$. Then by the second equation of (2.1) and the definition of soft threshold operator we get $\widehat{\boldsymbol{d}}_A = \lambda\mathrm{sgn}(\widehat{\boldsymbol{d}}_A + \widehat{\boldsymbol{\beta}}_A)$ which implies $\widehat{\boldsymbol{d}}_{A_0} = \lambda\mathrm{sgn}(\widehat{\boldsymbol{d}}_{A_0} + \widehat{\boldsymbol{\beta}}_{A_0})$. This together with the first equation of (2.1) and (2.11) implies

$$X_{A_0}'X_{A_0}\widehat{\boldsymbol{\beta}}_{A_0} + n\widehat{\boldsymbol{d}}_{A_0} = X_{A_0}'\boldsymbol{y} = X_{A_0}'X_{A_0}\boldsymbol{\beta}_{A_0}^1 + n\boldsymbol{d}_{A_0}^1.$$

Then we get $X_{A_0}'X_{A_0}(\widehat{\boldsymbol{\beta}}_{A_0} - \boldsymbol{\beta}_{A_0}^1) = \mathbf{0}$, therefore, $\widehat{\boldsymbol{\beta}}_{A_0} = \boldsymbol{\beta}_{A_0}^1$ follows from the above equation and the assumption that $\mathrm{rank}(X_A) = |A|$. Let $B^0 = (A_0)^c$, by (2.9) and the fact $A \subset A_0$ we deduce that $\boldsymbol{\beta}_{B^0}^1 = \mathbf{0} = \widehat{\boldsymbol{\beta}}_{B^0}$. Hence, $\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}^1$. This completes the proof of Theorem 3.3. □

In order to prove Lemma 4.1, we need the following two lemmas. Lemma 7.2 collects some property on mutual coherence and Lemma 7.3 states that the effect of the noise $\boldsymbol{\eta}$ can be controlled with high probability.

Lemma 7.2. *Let A, B be disjoint subsets of $S = 1, 2, ..., p$, with $|A| = a$, $|B| = b$. Let ν be the mutual coherence of X. Then we have*

$$\|X_B' X_A \boldsymbol{u}\|_\infty \le na\nu \|\boldsymbol{u}\|_\infty, \forall \boldsymbol{u} \in \mathbb{R}^{|A|}, \tag{A.21}$$

$$\|X_A\| = \left\|X_A'\right\| \le \sqrt{n(1+(a-1)\nu)}. \tag{A.22}$$

Furthermore, if $\nu < 1/(a-1)$, then $\forall u \in \mathbb{R}^{|A|}$,

$$\|(X_A' X_A)\boldsymbol{u}\|_\infty \ge n(1-(a-1)\nu)\|\boldsymbol{u}\|_\infty, \tag{A.23}$$

$$\|(X_A' X_A)^{-1}\boldsymbol{u}\|_\infty \le \frac{\|\boldsymbol{u}\|_\infty}{n(1-(a-1)\nu)}, \tag{A.24}$$

$$\|(X_A' X_A - nI)\boldsymbol{u}\|_\infty \le n(1+(a-1)\nu)\|\boldsymbol{u}\|_\infty. \tag{A.25}$$

Proof. Let $G = X'X/n$. $\forall i \in B$, $|\sum_{j=1}^a G_{i,j}u_j| \le \mu a \|u\|_\infty$, which implies (A.21). For any $i \in A$, by using Gerschgorin's disk theorem, $|\left\|G_{A,A}\right\| - G_{i,i}| \le \sum_{i\ne j=1}^a |G_{i,j}| \le (a-1)\mu$, i.e., (A.21) holds. Let $i \in A$ such that $\|u\|_\infty = |u_i|$. (A.23) follows from that $|\sum_{j=1}^a G_{i,j}u_j| \ge |u_i| - \sum_{i\ne j=1}^a |G_{i,j}||u_j| \ge \|u\|_\infty - \mu(a-1)\|u\|_\infty$. (A.24) follows directly from (A.23). And (A.25) can be showed similarly as the (A.23). This complete the proof of Lemma 7.2. □

Lemma 7.3. *Suppose (A3) holds. We have*

$$\mathbf{P}\left(\|X'\boldsymbol{\eta}\|_\infty/n \le \lambda_u\right) \ge 1 - \frac{1}{2\sqrt{\pi \log(p)}}. \tag{A.26}$$

Proof. This inequality follows from standard probabilities calculations. □

Recall that $\lambda_u = \sigma\sqrt{2\log(p)/n}$, $\delta_u = 3\lambda_u$, $\gamma = 8/13$, $\lambda_0 = \|X'y/n\|_\infty$ and $\lambda_t = \lambda_0\gamma^t$, $t = 0, 1, ...$

Proof of Lemma 4.1.. We first show that under the assumption of Lemma 4.1,

$$\lambda_1 > 10\delta_u \tag{A.27}$$

holds with probability at least $1 - \frac{1}{2\sqrt{\pi\log(p)}}$. In fact,

$$\begin{aligned}
\lambda_1 = \lambda_0\gamma &= \frac{8}{13}\|X'\boldsymbol{y}/n\|_\infty = \frac{8}{13}\|X'(X\boldsymbol{\beta}^\dagger + \boldsymbol{\eta})/n\|_\infty \\
&\ge \frac{8}{13}(\|X_{A^\dagger}' X_{A^\dagger}\boldsymbol{\beta}_{A^\dagger}^\dagger/n\|_\infty - \|X'\boldsymbol{\eta}/n\|_\infty) \\
&\ge \frac{8}{13}((1-(T-1)\nu)\|\boldsymbol{\beta}^\dagger\|_\infty - \lambda_u) \quad \text{W. H. P.} \\
&> \frac{8}{13}(\frac{3}{4}26\delta_u - \frac{\delta_u}{3}) \\
&> 10\delta_u
\end{aligned}$$

where the first inequality is the triangle inequality, the second inequality uses Lemma (A.23)-(A.26), and the third one follows uses assumption (A1)-(A2). Here in the third line, "W. H. P." stands for with high probability, that is, with probability at least $1 - 1/(2\sqrt{\pi\log(p)})$. Then it follow from (A.27) and the definition of λ_t that there exist an integer $N \in [1, \log_\gamma(\frac{10\delta_u}{\lambda_0}))$ such that

$$\lambda_N > 10\delta_u \ge \lambda_{N+1} \tag{A.28}$$

holds with high probability. It follows from assumption (A2) and (A.28) that $\lambda_{N+1} = \lambda_N 8/13 \le 10\delta_u \le |\boldsymbol{\beta}^\dagger|_{min} 10/26$, which implies that with high probability $|\boldsymbol{\beta}^\dagger|_{min} > 8\lambda_N/5$ holds. This complete the proof of Lemma 4.1. □

The main idea behind the proof of Theorem 4.1 is that under assumption (A1)-(A3) the active generated by PSNA is contained in the underlying target support and increase in some sense with high probability. To show this we need the following two Lemmas. Lemma 7.4 gives one-step error estimations of SNA (Algorithm 1) and Lemma 7.4 shows that some monotone property of the active set.

Lemma 7.4. *Suppose assumption (A1) holds. Let $A^k, B^k, \boldsymbol{\beta}^{k+1}, \boldsymbol{d}^{k+1}$ are generated by $Sna(\boldsymbol{\beta}^0, \boldsymbol{d}^0, \lambda, \bar{\lambda}, K)$ with $\lambda > \bar{\lambda} = \frac{9\lambda}{10} + \delta_u$. Denote $E^k = A^\dagger \backslash A^k$ and $i_k = \{i \in B^k : |\boldsymbol{\beta}_i^\dagger| = \|\boldsymbol{\beta}^\dagger\|_\infty\}$. If $A^k \subset A^\dagger$, then with probability at least $1 - \frac{1}{2\sqrt{\pi\log(p)}}$ we have*

$$\|\boldsymbol{\beta}_{A^k}^{k+1} + \boldsymbol{d}_{A^k}^{k+1} - \boldsymbol{\beta}_{A^k}^\dagger\|_\infty < \frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| + \frac{\lambda}{30}, \tag{A.29}$$

$$|\boldsymbol{\beta}_i^{k+1} + \boldsymbol{d}_i^{k+1}| > |\boldsymbol{\beta}_i^\dagger| - \frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| - \frac{\lambda}{30}, \forall i \in A^k, \tag{A.30}$$

$$|\boldsymbol{d}_i^{k+1}| < \frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| + \frac{\lambda}{30}, \tag{A.31}$$

$$|\boldsymbol{d}_{i_k}^{k+1}| > \frac{2}{3}|\boldsymbol{\beta}_{i_k}^\dagger| - \frac{\lambda}{30}. \tag{A.32}$$

Proof. Since $\boldsymbol{\beta}^{k+1}, \boldsymbol{d}^{k+1}$ are generated by SNA with $\lambda > \bar{\lambda} = \frac{9\lambda}{10} + \delta_u$, $A^k \subset A^\dagger$, $E^k = A^\dagger \backslash A^k$ and $\boldsymbol{y} = X_{A^\dagger}\boldsymbol{\beta}_{A^\dagger}^\dagger + \boldsymbol{\eta}$ we have

$$\boldsymbol{\beta}_{A^k}^{k+1} = (X_{A^k}' X_{A^k})^{-1}(X_{A^k}'(X_{A^k}\boldsymbol{\beta}_{A^k}^\dagger + X_{E^k}\boldsymbol{\beta}_{E^k}^\dagger + \boldsymbol{\eta}) - n\boldsymbol{d}^{k+1}) \tag{A.33}$$

and

$$\begin{aligned}
&\|\boldsymbol{\beta}_{A^k}^{k+1} + \boldsymbol{d}_{A^k}^{k+1} - \boldsymbol{\beta}_{A^k}^\dagger\|_\infty \le \|(X_{A^k}' X_{A^k})^{-1}(X_{A^k}'(X_{E^k}\boldsymbol{\beta}_{E^k}^\dagger + \boldsymbol{\eta}))\|_\infty \\
&+ \|(X_{A^k}' X_{A^k})^{-1}(X_{A^k}' X_{A^k} - nI)\boldsymbol{d}^{k+1}\|_\infty \\
&\le \frac{n|E^k|\nu|\boldsymbol{\beta}_{i_k}^\dagger| + \|X_{A^k}'\boldsymbol{\eta}\|_\infty}{n(1-(|A^k|-1)\nu)} + \frac{n(|A^k|-1)\nu}{n(1-(|A^k|-1)\nu)}(\lambda - \bar{\lambda}) \\
&< \frac{T\nu|\boldsymbol{\beta}_{i_k}^\dagger| + \lambda_u}{(1-T\nu)} + \frac{T\nu}{(1-T\nu)}(\lambda - (\frac{9\lambda}{10} + \delta_u)) \quad \text{W.H.P.} \\
&\le \frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| + \frac{\lambda}{30}
\end{aligned}$$

where the first inequality uses (A.33) and the triangle inequality, the second inequality uses (A.21), (A.24) and (A.24), the third inequality uses (A.26), the last inequality uses assumption (A1). Thus, (A.29) holds. Then, (A.30) follows from (A.29) and the triangle inequality. $\forall i \in B^k$,

$$\begin{aligned}
|\boldsymbol{d}_i^{k+1}| &= |X_i'(X_{A^k}(\boldsymbol{\beta}_{A^k}^\dagger - \boldsymbol{\beta}_{A^k}^{k+1} - \boldsymbol{d}_{A^k}^{k+1}) + X_{A^k}\boldsymbol{d}_{A^k}^{k+1} + X_{E^k}\boldsymbol{\beta}_{E^k}^\dagger + \boldsymbol{\eta})/n| \\
&\le |X_i' X_{A^k}(\boldsymbol{\beta}_{A^k}^\dagger - \boldsymbol{\beta}_{A^k}^{k+1} - \boldsymbol{d}_{A^k}^{k+1})| + |X_i' X_{A^k}\boldsymbol{d}_{A^k}^{k+1} + X_i' X_{E^k}\boldsymbol{\beta}_{E^k}^\dagger + X_i'\boldsymbol{\eta}|/n \\
&\le \nu|A^k|\|\boldsymbol{\beta}_{A^k}^{k+1} + \boldsymbol{d}_{A^k}^{k+1} - \boldsymbol{\beta}_{A^k}^\dagger\|_\infty + \nu|A^k|(\lambda - \bar{\lambda}) + \nu|E^k||\boldsymbol{\beta}_{i_k}^\dagger| + \lambda_u \quad \text{W.H.P.} \\
&< \frac{1}{4}(\frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| + \frac{\lambda}{30}) + \frac{1}{4}(\lambda - \bar{\lambda}) + \frac{1}{4}|\boldsymbol{\beta}_{i_k}^\dagger| + \lambda_u \\
&= \frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| + \frac{\lambda}{30}
\end{aligned}$$

where the first equality uses (A.33), the first inequality is the triangle inequality, the second inequality is due to (A.21) and (A.26), and the third inequality uses (A.29), i.e., (A.31) holds. Observing $i_k \in E^k$ and (A.33) we get

$$\begin{aligned}
|\boldsymbol{d}_{i_k}^{k+1}| &= |X_{i_k}'(X_{A^k}(\boldsymbol{\beta}_{A^k}^\dagger - \boldsymbol{\beta}_{A^k}^{k+1} - \boldsymbol{d}_{A^k}^{k+1}) + X_{A^k}\boldsymbol{d}_{A^k}^{k+1} + X_{i_k}\boldsymbol{\beta}_{i_k}^\dagger + X_{E^k\backslash i_k}\boldsymbol{\beta}_{E^k\backslash i_k}^\dagger + \boldsymbol{\eta})/n| \\
&\ge |\boldsymbol{\beta}_{i_k}^\dagger| - |X_i' X_{A^k}(\boldsymbol{\beta}_{A^k}^\dagger - \boldsymbol{\beta}_{A^k}^{k+1} - \boldsymbol{d}_{A^k}^{k+1})| - |X_i'(X_{A^k}\boldsymbol{d}_{A^k}^{k+1} + X_{E^k\backslash i_k}\boldsymbol{\beta}_{E^k\backslash i_k}^\dagger + \boldsymbol{\eta}|/n \\
&\ge |\boldsymbol{\beta}_{i_k}^\dagger| - \nu|A^k|\|\boldsymbol{\beta}_{A^k}^{k+1} - \boldsymbol{d}_{A^k}^{k+1} - \boldsymbol{\beta}_{A^k}^\dagger\|_\infty - \nu|A^k|(\lambda - \bar{\lambda}) - \nu|E^k||\boldsymbol{\beta}_{i_k}^\dagger| - \lambda_u \quad \text{W.H.P.} \\
&> |\boldsymbol{\beta}_{i_k}^\dagger| - \frac{1}{4}(\frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| + \frac{\lambda}{30}) - \frac{1}{4}(\lambda - \bar{\lambda}) - \frac{1}{4}|\boldsymbol{\beta}_{i_k}^\dagger| - \lambda_u \\
&> \frac{2}{3}|\boldsymbol{\beta}_{i_k}^\dagger| - \frac{\lambda}{30}
\end{aligned}$$

where the first inequality is the triangle inequality, the second inequality is due to Lemma (A.21) and (A.26), and the third one uses A.29, i.e., (A.32) holds. This complete the proof of Lemma 7.4. □

For a given $\tau > 0$, we define $S_{\lambda,\tau} = \{i : |\boldsymbol{\beta}_i^\dagger| \geq \lambda\tau\}$.

Lemma 7.5. *Suppose assumption (A1) hold. Let $\kappa = \frac{8}{5}$ and $\tau = \kappa$ or $\kappa + 1$. Denote $E^k = A^\dagger \backslash A^k$ and $i_k = \{i \in B^k : |\boldsymbol{\beta}_i^\dagger| = \|\boldsymbol{\beta}^\dagger\|_\infty\}$. If $S_{\lambda,\tau} \subset A^k \subset A^\dagger$ then $S_{\lambda,\tau} \subset A^{k+1} \subset A^\dagger$. Meanwhile, if $S_{\lambda,\kappa+1} \subset A^k \subset A^\dagger$ and $S_{\lambda,\kappa} \nsubseteq A^k$ then $|\boldsymbol{\beta}_{i_k}^\dagger| > |\boldsymbol{\beta}_{i_{k+1}}^\dagger|$.*

Proof. Assume $S_{\lambda,\tau} \subset A^k \subset A^\dagger$. Since $E^k = A^\dagger \backslash A^k$ and $i_k \in E^k$, we get $i_k \notin A^k$ which implies $|\boldsymbol{\beta}_{i_k}^\dagger| < \lambda\tau$. $\forall i \in S_{\lambda,\tau} \subset A^k$. By using (A.30) we have

$$|\boldsymbol{\beta}_i^{k+1} + \boldsymbol{d}_i^{k+1}| > |\boldsymbol{\beta}_i^\dagger| - \frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| - \frac{\lambda}{30} > \lambda\tau - \frac{1}{3}\lambda\tau - \frac{\lambda}{30} > \lambda,$$

which implies $i \in A^{k+1}$, i.e., $S_{\lambda,\kappa} \subset A^{k+1}$ holds. $\forall i \in (A^\dagger)^c \subset B^k$. By using (A.31) we get

$$|\boldsymbol{\beta}_i^{k+1} + \boldsymbol{d}_i^{k+1}| = |\boldsymbol{d}_i^{k+1}| < \frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| + \frac{\lambda}{30} < \begin{cases} \lambda, & \tau = \kappa + 1, \\ \frac{\kappa}{\kappa+1}\lambda, & \tau = \kappa, \end{cases} \tag{A.34}$$

i.e., $i \notin A^{k+1}$ which implies $A^{k+1} \subset A^\dagger$. Next we turn to the second assertion. Assume $S_{\lambda,\kappa+1} \subset A^k \subset A^\dagger$, $S_{\lambda,\kappa} \nsubseteq A^k$. It suffice to show all the elements of $|\boldsymbol{\beta}^\dagger|$ that larger than $|\boldsymbol{\beta}_{i_k}^\dagger|$ move into A^{k+1}. It follows from the definition of $S_{\lambda,\kappa}$, $S_{\lambda,\kappa+1}$ and $i_k \in E^k = A^\dagger \backslash A^k$ that $i_k \in S_{\lambda,\kappa} \backslash S_{\lambda,\kappa+1}$, i.e., $|\boldsymbol{\beta}_{i_k}^\dagger| \in [\lambda\kappa, \lambda(\kappa+1))$. By using (A.32) we have

$$|\boldsymbol{\beta}_{i_k}^{k+1} + \boldsymbol{d}_{i_k}^{k+1}| = |\boldsymbol{d}_{i_k}^{k+1}| > \frac{2}{3}|\boldsymbol{\beta}_{i_k}^\dagger| - \frac{1}{30}\lambda > \frac{2}{3}\lambda\kappa - \frac{1}{30}\lambda > \lambda,$$

which implies $i_k \in A^{k+1}$. Let $i \in A^k$ satisfy $|\boldsymbol{\beta}_i^\dagger| \geq |\boldsymbol{\beta}_{i_k}^\dagger|$. Then it follows from (A.30) that

$$\begin{aligned} |\boldsymbol{\beta}_i^{k+1} + \boldsymbol{d}_i^{k+1}| &> |\boldsymbol{\beta}_i^\dagger| - \frac{1}{3}|\boldsymbol{\beta}_{i_k}^\dagger| - \frac{\lambda}{30} \\ &> \frac{2}{3}|\boldsymbol{\beta}_{i_k}^\dagger| - \frac{1}{30}\lambda \\ &> \frac{2}{3}\lambda\kappa - \frac{1}{30}\lambda > \lambda, \end{aligned}$$

which implies $i \in A^{k+1}$. This complete the proof of Lemma 7.5. □

With the above preparation, we now give the prove of Theorem 4.1.

Proof of Theorem 4.1. Let $\overline{\lambda_t} = \frac{9}{10}\lambda_t + \delta_u$. By using Lemma 4.1 and the definition of λ_t and we get $\lambda_t > \overline{\lambda_t}, t = 0, 1, ..., N$. At the t_{th} knot of $Snap(\lambda_0, \gamma, N, K)$, suppose it takes Algorithm $Sna(\boldsymbol{\beta}^0, \boldsymbol{d}^0, \lambda_t, \overline{\lambda_t}, K)$ k_t iterations to get the solution $(\widehat{\boldsymbol{\beta}}(\lambda_t), \widehat{\boldsymbol{d}}(\lambda_t))$, where $(\boldsymbol{\beta}^0, \boldsymbol{d}^0) = (\widehat{\boldsymbol{\beta}}(\lambda_{t-1}), \widehat{\boldsymbol{d}}(\lambda_{t-1}))$ and $k_t \leq K$ by the definition of PSNA. We denote the approximate primal dual solution pair and active set generated in $Sna(\widehat{\boldsymbol{\beta}}(\lambda_{t-1}), \widehat{\boldsymbol{d}}(\lambda_{t-1}), \lambda_t, \overline{\lambda_t}, K)$ by $(\boldsymbol{\beta}_t^k, \boldsymbol{d}_t^k)$ and A_t^k, respectively, $k = 0, 1, ..., k_t$. By the construction of PSNA we have $(\boldsymbol{\beta}_t^{k_t}, \boldsymbol{d}_t^{k_t}) = (\widehat{\boldsymbol{\beta}}(\lambda_t), \widehat{\boldsymbol{d}}(\lambda_t))$, i.e, the solution at the t_{th} stage is the initial value for the $t+1$ stage which implies

$$A_t^{k_t} \subseteq A_{t+1}^0. \tag{A.35}$$

We claim that

$$S_{\lambda_t,\kappa+1} \subseteq A_t^0 \subseteq A^\dagger, t = 0, 1, ..., N. \tag{A.36}$$

$$S_{\lambda_t,\kappa} \subseteq A_t^{k_t} \subseteq A^\dagger, t = 0, 1, ..., N. \tag{A.37}$$

We prove the above two claims by mathematical induction. First we show that $\emptyset = S_{\lambda_0,\kappa+1} \subseteq A_0^0 \subseteq A^\dagger$. Let $|\boldsymbol{\beta}_i^\dagger| = \|\boldsymbol{\beta}^\dagger\|_\infty$.

$$\begin{aligned} (\kappa+1)\lambda_0 &= \frac{13}{5}\|X'\boldsymbol{y}/n\|_\infty = \frac{13}{5}\|X'(X\boldsymbol{\beta}^\dagger + \boldsymbol{\eta})/n\|_\infty \\ &\geq \frac{13}{5}(\|X'_{A^\dagger}X_{A^\dagger}\boldsymbol{\beta}_{A^\dagger}^\dagger/n\|_\infty - \|X'\boldsymbol{\eta}/n\|_\infty) \\ &\geq \frac{13}{5}((1-(T-1)\nu)|\boldsymbol{\beta}_i^\dagger| - \lambda_u), \quad W.H.P \\ &> \frac{13}{5}(\frac{3}{4}|\boldsymbol{\beta}_i^\dagger| - \lambda_u) \\ &> |\boldsymbol{\beta}_i^\dagger| \end{aligned} \tag{A.38}$$

where the first inequality is the triangle equation and the second inequality uses (A.23) and (A.26), the third inequality uses assumption (A1), and the last inequality is derive from assumption (A2). This implies $\emptyset = S_{\lambda_0,\kappa+1}$. By the construction of $Snap(\lambda_0, \gamma, N, K)$ we get $A_0^0 = \{j : |X_j'\boldsymbol{y}/n| > \lambda_0 = \|X'\boldsymbol{y}/n\|_\infty\} = \emptyset$. Therefore, (A.36) holds when $t = 0$. Now we suppose (A.36) holds for some $t \geq 0$. Then by the first assertion of Lemma 7.5 we get

$$S_{\lambda_t,\kappa+1} \subseteq A_t^k \subseteq A^\dagger, k = 0, 1, ..., k_t. \tag{A.39}$$

By the stopping rule of $Sna(\beta^0, d^0, \lambda_t, \overline{\lambda_t}, K)$ it holds either $A_t^{k_t} = A_t^{k_t-1}$ or $k_t = K \geq T$ when it stops. In both cases, by using (A.39) and the second assertion of Lemma 7.5 we get

$$S_{\lambda_t,\kappa} \subseteq A_t^{k_t} \subseteq A^\dagger,$$

i.e., (A.37) holds for this given t. Observing the relation $S_{\lambda_{t+1},\kappa+1} = S_{\lambda_t,\kappa}$ and (A.34)-(A.35) we get $S_{\lambda_{t+1},\kappa+1} \subseteq A_{t+1}^0 \subseteq A^\dagger$, i.e., (A.36) holds for $t+1$. Therefore, (A.36) - (A.37) are verified by mathematical induction on t. That is all the active set generated in PSNA is contained in $A^\dagger$. Therefore, by Lemma 4.1 we get

$$A^\dagger \subseteq S_{\lambda_N,\kappa} \subseteq A_N^{k_N} \subseteq A^\dagger,$$

i.e.,

$$\operatorname{supp}(\widehat{\boldsymbol{\beta}}(\lambda_N)) = A^\dagger. \tag{A.40}$$

Then,

$$\begin{aligned} \|\boldsymbol{\beta}^\dagger - \widehat{\boldsymbol{\beta}}(\lambda_N)\|_\infty &= \|\boldsymbol{\beta}_{A^\dagger}^\dagger - (X'_{A^\dagger}X_{A^\dagger})^{-1}(\widetilde{\boldsymbol{y}}_{A^\dagger} - n\widehat{\boldsymbol{d}}(\lambda_N)_{A^\dagger})\|_\infty \\ &= \|\boldsymbol{\beta}_{A^\dagger}^\dagger - (X'_{A^\dagger}X_{A^\dagger})^{-1}(X'_{A^\dagger}(X_{A^\dagger}\boldsymbol{\beta}_{A^\dagger}^\dagger + \boldsymbol{\eta}) - n\widehat{\boldsymbol{d}}(\lambda_N)_{A^\dagger})\|_\infty \\ &\leq \frac{\|X'_{A^\dagger}\boldsymbol{\eta}\|_\infty + n(\lambda_N - \overline{\lambda_N})}{n(1-T\nu)} \\ &< \frac{\lambda_u + \frac{\lambda_N}{10} - 3\lambda_u}{1-\frac{1}{4}} \quad \text{W.H.P.} \\ &\leq \frac{\frac{39}{8}\lambda_u - 2\lambda_u}{\frac{3}{4}} = \frac{23}{6}\lambda_u, \end{aligned}$$

where the first inequality uses (A.24), the second inequality uses (A.26), and last inequality uses Lemma 4.1, i.e., (4.2) holds. The sign consistency (4.1) follows directly from (A.40), (4.2) and assumption (A2). This complete the proof of Theorem 4.1. □

A3. Details in Algorithm 3

We now describe in detail the quantities in the k_{th} iteration in Algorithm 3. This paves the way for showing that Algorithm 1 is a specialization of Algorithm 3. At $\boldsymbol{z}^k = (\boldsymbol{\beta}^{k\prime}, \boldsymbol{d}^{k\prime})'$, we define A_k and B_k by (2.8). By a similar reordering of $(\boldsymbol{\beta}^{k\prime}, \boldsymbol{d}^{k\prime})'$, $F_1(\boldsymbol{z}^k)$ and $F_2(\boldsymbol{z}^k)$ as concerning the Newton derivative of F in Theorem 3.1, and using the definition of $T_\lambda(\cdot)$, we get

$$\boldsymbol{z}^k = \begin{pmatrix} \boldsymbol{d}_{A_k}^k \\ \boldsymbol{\beta}_{B_k}^k \\ \boldsymbol{\beta}_{A_k}^k \\ \boldsymbol{d}_{B_k}^k \end{pmatrix}, F(\boldsymbol{z}^k) = \begin{bmatrix} -\boldsymbol{d}_{A_k}^k + \lambda\operatorname{sgn}(\boldsymbol{\beta}_{A_k}^k + \boldsymbol{d}_{A_k}^k) \\ \boldsymbol{\beta}_{B_k}^k \\ G_{A_kA_k}\boldsymbol{\beta}_{A_k}^k + G_{A_kB_k}\boldsymbol{\beta}_{B_k}^k + n\boldsymbol{d}_{A_k}^k - \widetilde{\boldsymbol{y}}_{A_k}^k \\ G_{B_kA_k}\boldsymbol{\beta}_{A_k}^k + G_{B_kB_k}\boldsymbol{\beta}_{B_k}^k + n\boldsymbol{d}_{B_k}^k - \widetilde{\boldsymbol{y}}_{B_k}^k \end{bmatrix}. \tag{A.41}$$

Then, by using Theorem 3.1 and noting that $G_{B_kA_k} = X'_{B_k}X_{A_k}$, $G_{A_kB_k} = X'_{A_k}X_{B_k}$ we have $H_k \in \nabla_N F(\boldsymbol{z}^k)$, where

$$H_k = \begin{bmatrix} -I_{A_kA_k} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_{B_kB_k} & \mathbf{0} & \mathbf{0} \\ nI_{A_kA_k} & X'_{A_k}X_{B_k} & G_{A_kA_k} & \mathbf{0} \\ \mathbf{0} & G_{B_kB_k} & X'_{B_k}X_{A_k} & nI_{B_kB_k} \end{bmatrix}. \tag{A.42}$$

Algorithm 3 is well defined if we choose H_k in the form of (A.42), since H_k is invertible as shown in Theorem 3.1.

In Section 2.1 we derived Algorithm 1 in an intuitive way. We now verify that Algorithm 1 is indeed Algorithm 3 in a form that can be easily and efficiently implemented computationally. Let

$$D^k = \begin{pmatrix} D^{\boldsymbol{d}}_{A_k} \\ D^{\boldsymbol{\beta}}_{B_k} \\ D^{\boldsymbol{\beta}}_{A_k} \\ D^{\boldsymbol{d}}_{B_k} \end{pmatrix}$$

and substitute (A.41) and (A.42) into () we get

$$\boldsymbol{d}^k_{A_k} + D^{\boldsymbol{d}}_{A_k} = \lambda \mathrm{sgn}(\boldsymbol{\beta}^k_{A_k} + \boldsymbol{d}^k_{A_k}), \tag{A.43}$$

$$\boldsymbol{\beta}^k_{B_k} + D^{\boldsymbol{\beta}}_{B_k} = \mathbf{0}, \tag{A.44}$$

$$G_{A_kA_k}(\boldsymbol{\beta}^k_{A_k} + D^{\boldsymbol{\beta}}_{A_k}) = \widetilde{\boldsymbol{y}}_{A_k} - n(\boldsymbol{d}^k_{A_k} + D^{\boldsymbol{d}}_{A_k}) - X'_{A_k}X_{B_k}(\boldsymbol{\beta}^k_{B_k} + D^{\boldsymbol{\beta}}_{B_k}), \tag{A.45}$$

$$n(\boldsymbol{d}^k_{B_k} + D^{\boldsymbol{d}}_{B_k}) = \widetilde{\boldsymbol{y}}_{B_k} - X'_{B_k}X_{A_k}(\boldsymbol{\beta}^k_{A_k} + D^{\boldsymbol{\beta}}_{A_k}) - G_{B_kB_k}(\boldsymbol{\beta}^k_{B_k} + D^{\boldsymbol{\beta}}_{B_k}). \tag{A.46}$$

Observing the relationship (by ()),

$$\begin{pmatrix} \boldsymbol{d}^{k+1}_{A_k} \\ \boldsymbol{\beta}^{k+1}_{B_k} \\ \boldsymbol{\beta}^{k+1}_{A_k} \\ \boldsymbol{d}^{k+1}_{B_k} \end{pmatrix} = \begin{pmatrix} \boldsymbol{d}^k_{A_k} + D^{\boldsymbol{d}}_{A_k} \\ \boldsymbol{\beta}^k_{B_k} + D^{\boldsymbol{\beta}}_{B_k} \\ \boldsymbol{\beta}^k_{A_k} + D^{\boldsymbol{\beta}}_{A_k} \\ \boldsymbol{d}^k_{B_k} + D^{\boldsymbol{d}}_{B_k} \end{pmatrix}.$$

and substituting (A.43) - (A.44) into (A.45)-(A.46), we obtain (2.9) - (2.12), which are the computational steps in Algorithm 1.

References

[1] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society. Series B (Methodological) (1996) 267–288.
[2] S.S. Chen, D.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit, SIAM Journal on Scientific Computing 20 (1) (1998) 33–61.
[3] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67 (2) (2005) 301–320.
[4] B. Kummer, Newton's method for non-differentiable functions, Advances in Mathematical Optimization 45 (1988) 114–125.
[5] L. Qi, J. Sun, A nonsmooth version of newton's method, Mathematical programming 58 (1-3) (1993) 353–367.
[6] K. Ito, K. Kunisch, Lagrange multiplier approach to variational problems and applications, SIAM, Philadelphia, 2008.
[7] M.R. Osborne, B. Presnell, B.A. Turlach, A new approach to variable selection in least squares problems, IMA Journal of Numerical Analysis 20 (3) (2000) 389–403.
[8] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, The Annals of Statistics 32 (2) (2004) 407–499.
[9] D.L. Donoho, Y. Tsaig, Fast solution of ℓ_1-norm minimization problems when the solution may be sparse, IEEE Transactions on Information Theory 54 (11) (2008) 4789–4812.
[10] J. Fan, J. Lv, Sure independence screening for ultrahigh dimensional feature space, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 70 (5) (2008) 849–911.
[11] R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, R.J. Tibshirani, Strong rules for discarding predictors in lasso-type problems, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 74 (2) (2012) 245–266.
[12] W.J. Fu, Penalized regressions: the bridge versus the lasso, Journal of Computational and Graphical Statistics 7 (3) (1998) 397–416.
[13] J. Friedman, T. Hastie, H. Höfling, R. Tibshirani, Pathwise coordinate optimization, The Annals of Applied Statistics 1 (2) (2007) 302–332.
[14] T.T. Wu, K. Lange, Coordinate descent algorithms for lasso penalized regression, The Annals of Applied Statistics 2 (1) (2008) 224–244.
[15] Y. Li, S. Osher, Coordinate descent optimization for ℓ^1 minimization with application to compressed sensing; a greedy algorithm, Inverse Problems and Imaging 3 (3) (2009) 487–503.
[16] I. Daubechies, M. Defrise, C. De Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Communications on Pure and Applied Mathematics 57 (11) (2004) 1413–1457.
[17] Y. She, Thresholding-based iterative selection procedures for model selection and shrinkage, Electronic Journal of Statistics 3 (2009) 384–415.
[18] P. Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, Journal of Optimization Theory and Applications 109 (3) (2001) 475–494.
[19] A. Saha, A. Tewari, On the nonasymptotic convergence of cyclic coordinate descent methods, SIAM Journal on Optimization 23 (1) (2013) 576–601.
[20] S. Yun, On the iteration complexity of cyclic coordinate gradient descent methods, SIAM Journal on Optimization 24 (3) (2014) 1567–1580.
[21] P. Tseng, S. Yun, A coordinate gradient descent method for nonsmooth separable minimization, Mathematical Programming 117 (2009) 387–423.
[22] Y. Nesterov, Smooth minimization of non-smooth functions, Mathematical Programming 103 (1) (2005) 127–152.
[23] Y. Nesterov, Gradient methods for minimizing composite functions, Mathematical Programming 140 (1) (2013) 125–161.
[24] A. Agarwal, S. Negahban, M.J. Wainwright, Fast global convergence of gradient methods for high-dimensional statistical recovery, The Annals of Statistics 40 (5) (2012) 2452–2482.
[25] L. Xiao, T. Zhang, A proximal-gradient homotopy method for the sparse least--squares problem, SIAM Journal on Optimization 23 (2) (2013) 1062–1091.
[26] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends® in Machine learning 3 (1) (2011) 1–122.
[27] L. Chen, D. Sun, K.-C. Toh, An efficient inexact symmetric Gauss–Seidel based majorized ADMM for high-dimensional convex composite conic programming, Mathematical Programming 161 (1-2) (2017) 237–270.
[28] D. Han, D. Sun, L. Zhang, Linear rate convergence of the alternating direction method of multipliers for convex composite programming, Mathematics of Operations Research 43 (2) (2017) 622–637.
[29] J.A. Tropp, S.J. Wright, Computational methods for sparse solution of linear inverse problems, Proceedings of the IEEE 98 (6) (2010) 948–958.
[30] N. Parikh, S. Boyd, Proximal algorithms, Foundations and Trends® in Optimization 1 (3) (2014) 127–239.
[31] D.L. Donoho, I.M. Johnstone, Adapting to unknown smoothness via wavelet shrinkage, Journal of the American Statistical Association 90 (432) (1995) 1200–1224.
[32] Y. Jiao, B. Jin, X. Lu, Iterative soft/hard thresholding with homotopy continuation for sparse recovery, IEEE Signal Processing Letters 24 (6) (2017) 784–788.
[33] G.H. Golub, C.F. Van Loan, Matrix computations, 3, Johns Hopkins University Press, Baltimore, 1996.
[34] D.L. Donoho, X. Huo, Uncertainty principles and ideal atomic decomposition, IEEE Transactions on Information Theory 47 (7) (2001) 2845–2862.
[35] D.L. Donoho, M. Elad, V.N. Temlyakov, Stable recovery of sparse overcomplete representations in the presence of noise, IEEE Transactions on Information Theory 52 (1) (2006) 6–18.
[36] P. Zhao, B. Yu, On model selection consistency of lasso, Journal of Machine Learning Research 7 (2006) 2541–2563.
[37] N. Meinshausen, P. Bühlmann, High-dimensional graphs and variable selection with the lasso, The Annals of Statistics 34 (3) (2006) 1436–1462.
[38] C.-H. Zhang, J. Huang, The sparsity and bias of the lasso selection in high-dimensional linear regression, The Annals of Statistics 36 (4) (2008) 1567–1594.
[39] M.J. Wainwright, Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (lasso), IEEE Transactions on Information Theory 55 (5) (2009) 2183–2202.
[40] K. Lounici, Sup-norm convergence rate and sign concentration property of Lasso and Dantzig estimators, Electronic Journal of Statistics 2 (2008) 90–102.
[41] E.J. Candès, Y. Plan, Near-ideal model selection by ℓ_1 minimization, The Annals of Statistics 37 (5A) (2009) 2145–2177.
[42] T. Zhang, Some sharp performance bounds for least squares regression with l_1 regularization, The Annals of Statistics 37 (5A) (2009) 2109–2144.
[43] J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, Journal of Statistical Software 33 (1) (2010) 1–22.
[44] E.J. Candès, J. Romberg, T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, IEEE Transactions on Information Theory 52 (2) (2006) 489–509.
[45] E.J. Candès, T. Tao, Near-optimal signal recovery from random projections: Universal encoding strategies? IEEE Transactions on Information Theory 52 (12) (2006) 5406–5425.
[46] H. Wang, R. Li, C.-L. Tsai, Tuning parameter selectors for the smoothly clipped absolute deviation method, Biometrika 94 (3) (2007) 553–568.
[47] J. Chen, Z. Chen, Extended bayesian information criteria for model selection with large model spaces, Biometrika 95 (3) (2008) 759–771.
[48] H. Wang, B. Li, C. Leng, Shrinkage tuning parameter selection with a diverging

number of parameters, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 71 (3) (2009) 671–683.
[49] J. Chen, Z. Chen, Extended BIC for small-*n*-large-*P* sparse GLM, Statistica Sinica 22 (2012) 555–574.
[50] Y. Kim, S. Kwon, H. Choi, Consistent model selection criteria on high dimensions, Journal of Machine Learning Research 13 (2012) 1037–1057.
[51] L. Wang, Y. Kim, R. Li, Calibrating nonconvex penalized regression in ultra-high dimension, The Annals of Statistics 41 (5) (2013) 2505–2536.
[52] S. Becker, J. Bobin, E.J. Candès, Nesta: A fast and accurate first-order method for sparse recovery, SIAM Journal on Imaging Sciences 4 (1) (2011) 1–39.
[53] Y. Shi, Y. Wu, D. Xu, Y. Jiao, An ADMM with continuation algorithm for non–convex SICA-penalized regression in high dimensions, Journal of Statistical Computation and Simulation 88 (9) (2018) 1826–1846.
[54] Y. Shi, J. Huang, Y. Jiao, Q. Yang, A semismooth Newton algorithm for high-dimensional nonconvex sparse learning, IEEE Transactions on Neural Networks and Learning Systems 31 (8) (2020) 2993–3006.
[55] A. Tan, J. Huang, Bayesian inference for high-dimensional linear regression under mnet priors, Canadian Journal of Statistics 44 (2) (2016) 180–197.
[56] C. Yi, J. Huang, Semismooth newton coordinate descent algorithm for elastic-net penalized huber loss regression and quantile regression, Journal of Computational and Graphical Statistics 26 (3) (2017) 547–557.
[57] S. Lv, H. Lin, H. Lian, J. Huang, Oracle inequalities for sparse additive quantile regression in reproducing kernel hilbert space, The Annals of Statistics 46 (2) (2018) 781–813.
[58] P. Breheny, Marginal false discovery rates for penalized regression models, Biostatistics https://doi.org/10.1093/biostatistics/kxy004 (2018).
[59] J. Huang, S. Ma, C.-H. Zhang, Adaptive lasso for sparse high-dimensional regression models, Statistica Sinica 18 (2008) 1603–1618.
[60] J. Huang, J.L. Horowitz, F. Wei, Variable selection in nonparametric additive models, The Annals of Statistics 38 (4) (2010) 2282–2313.
[61] J. Fan, R. Li, Variable selection via nonconcave penalized likelihood and its oracle properties, Journal of the American Statistical Association 96 (456) (2001) 1348–1360.
[62] C.-H. Zhang, Nearly unbiased variable selection under minimax concave penalty, The Annals of Statistics 38 (2) (2010) 894–942.
[63] P. Breheny, J. Huang, Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection, The Annals of Applied Statistics 5 (1) (2011) 232–253.
[64] R. Mazumder, J.H. Friedman, T. Hastie, SparseNet: Coordinate descent with nonconvex penalties, Journal of the American Statistical Association 106 (495) (2011) 1125–1138.
[65] X. Chen, L. Qi, D. Sun, 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 (222) (1998) 519–540.
[66] L. Qi, D. Sun, A survey of some nonsmooth equations and smoothing Newton methods, in: Progress in optimization, Springer, 1999, pp. 121–146.
[67] L. Qi, D. Sun, G. Zhou, A new look at smoothing Newton methods for nonlinear complementarity problems and box constrained variational inequalities, Mathematical Programming 87 (1) (2000) 1–35.
[68] R.T. Rockafellar, Convex analysis, Princeton University Press, Princeton, 1970.
[69] P.L. Combettes, V.R. Wajs, Signal recovery by proximal forward-backward splitting, Multiscale Modeling and Simulation 4 (4) (2005) 1168–1200.
[70] X. Chen, Z. Nashed, L. Qi, Smoothing methods and semismooth methods for nondifferentiable operator equations, SIAM Journal on Numerical Analysis 38 (4) (2000) 1200–1216.