Statistics and Computing (2024) 34:59
https://doi.org/10.1007/s11222-023-10371-8

ORIGINAL PAPER

Accelerated gradient methods for sparse statistical learning with nonconvex penalties

Kai Yang[1] · **Masoud Asgharian**[2] · **Sahir Bhatnagar**[1]

Received: 24 November 2022 / Accepted: 2 December 2023 / Published online: 2 January 2024

Abstract
Nesterov's accelerated gradient (AG) is a popular technique to optimize objective functions comprising two components: a convex loss and a penalty function. While AG methods perform well for convex penalties, such as the LASSO, convergence issues may arise when it is applied to nonconvex penalties, such as SCAD. A recent proposal generalizes Nesterov's AG method to the nonconvex setting. The proposed algorithm requires specification of several hyperparameters for its practical application. Aside from some general conditions, there is no explicit rule for selecting the hyperparameters, and how different selection can affect convergence of the algorithm. In this article, we propose a hyperparameter setting based on the complexity upper bound to accelerate convergence, and consider the application of this nonconvex AG algorithm to high-dimensional linear and logistic sparse learning problems. We further establish the rate of convergence and present a simple and useful bound to characterize our proposed optimal damping sequence. Simulation studies show that convergence can be made, on average, considerably faster than that of the conventional proximal gradient algorithm. Our experiments also show that the proposed method generally outperforms the current state-of-the-art methods in terms of signal recovery.

Keywords Optimization · Statistical computing · Variable selection

1 Introduction

Sparse learning is an important component of modern data science and is an essential tool for the statistical analysis of high-dimensional data, with significant applications in signal processing and statistical genetics, among others. Penalization is commonly used to achieve sparsity in parameter estimation. The prototypical optimization problem for obtaining penalized estimators is

$$\hat{\boldsymbol{\beta}} \in \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{q+1}} \left[f(\boldsymbol{\beta}) + \sum_{j=1}^{q} p_\lambda(\beta_j) \right],$$

where $f : \mathbb{R}^{q+1} \mapsto \mathbb{R}$ is a convex loss function, $p_\lambda : \mathbb{R} \mapsto \mathbb{R}_{\geq 0}$ constitutes the penalty term, and $\lambda > 0$ is the tuning parameter for the penalty. Commonly used penalization methods for sparse learning include: LASSO (Least Absolute Shrinkage and Selection Operator) (Tibshirani 1996), Elastic Net (Zou and Hastie 2005), SCAD (Smoothly Clipped Absolute Deviation) (Fan and Li 2001) and MCP (Minimax Concave Penalty) (Zhang 2010). Among these penalties, parameter estimation with SCAD and MCP leads to a nonconvex objective function. The nonconvexity poses a challenge in statistical computing, as most methods developed for convex objective functions might not converge when applied to the nonconvex counterpart.

Various approaches have been proposed to carry out parameter estimation with SCAD or MCP penalties. Zou and Li (2008) proposed a local linear approximation, which yields a first-order majorization-minimization (MM) algorithm. Kim et al. (2008) discussed a difference-of-convex programming (DCP) method for ordinary least square esti-

✉ Kai Yang
kai.yang2@mail.mcgill.ca

Masoud Asgharian
masoud.asgharian2@mcgill.ca

Sahir Bhatnagar
sahir.bhatnagar@mcgill.ca

[1] Department of Epidemiology, Biostatistics and Occupational Health, McGill University, 845 Rue Sherbrooke O, Montreal, QC H3A 0G4, Canada

[2] Department of Mathematics and Statistics, McGill University, 845 Rue Sherbrooke O, Montreal, QC H3A 0G4, Canada

mators penalized by the SCAD penalty, which was later generalized by Wang et al. (2013) to a general class of nonconvex penalties to produce a first-order algorithm. These first-order methods belong to the class of proximal gradient descent methods, which are usually inefficient as relaxation is often expensive (Nesterov 2004). The objective function is often ill-conditioned for sparse learning problems, and gradient descent with constant step size is especially inefficient for high-dimensional problems. Indeed, previous studies have suggested that the condition number of a square random matrix grows linearly with respect to its dimension (Edelman 1988). Therefore, high-dimensional problems have a large condition number with high probability. Specific to gradient descent with constant step size, the trajectory will oscillate in the directions with a large eigenvalue, moving very slowly toward the directions with a small eigenvalue, making the algorithm inefficient. Lee et al. (2016) developed a modified second-order method originally designed for the ordinary least square loss function penalized by LASSO with extensions to SCAD and MCP; this attempt was later extended to generalized linear models, such as logistic and Poisson regression, and Cox's proportional hazard model. Quasi-Newton methods, or a mixture of first and second-order descent methods, have also been applied on nonconvex penalties (Ibrahim et al. 2012; Ghosh and Thoresen 2018). However, for high-dimensional problems, these second-order methods are slow due to the computational cost of evaluating the secant condition. Concurrently, most first and second-order methods discussed above require a line-search procedure at each step to ensure global convergence, which is prohibitive when the number of parameters to estimate grows large. Breheny and Huang (2011) implemented a coordinate descent method in the `ncvreg` `R` package to carry out estimation for linear models with least squares loss or logistic regression, penalized by SCAD and MCP. Mazumder et al. (2011) also implemented a coordinate descent method in the `sparsenet` `R` package, which carries out a closed-form root-finding update in a coordinate-wise manner for penalized linear regression. Similar to how ill-conditioning makes gradient descent inefficient, coordinate descent methods are generally inefficient when the covariate correlations are high (Friedman et al. 2077). Previous studies have also found that coordinate-wise minimization might not converge for some nonsmooth objective functions (Spall 2012). Furthermore, it is naturally challenging to run coordinate-wise minimization in parallel, as the algorithm must run in a sequential coordinate manner.

Due to the low computational cost and adequate memory requirement per iteration, first-order methods without a line search procedure have become the primary approach for high-dimensional problems arising from various areas (Beck 2017). For smooth convex objective functions, Nesterov proposed the *accelerated gradient method* (AG) to improve the rate of convergence from $O(1/N)$ for gradient descent to $O(1/N^2)$ while achieving global convergence (Nesterov 1983). Subsequently, Nesterov extended AG to composite convex problems (Nesterov 2012), whereas the objective is the sum of a smooth convex function and a simple nonsmooth convex function. With proper step-size choices, Nesterov's AG was later shown optimal to solve both smooth and nonsmooth convex programming problems (Lan 2011).

Given that sparse learning problems are often high-dimensional, Nesterov's AG has been frequently used for *convex* problems in statistical machine learning (e.g., Simon et al. 2013; Yang and Zou 2014; Yu et al. 2015; Akyildiz and Míguez 2021). However, convergence is questionable if the convexity assumption is violated. Recently, Ghadimi and Lan (2015) generalized the AG method to nonconvex objective functions, hereafter referred to as the nonconvex AG method, and derived the rates of convergence for both smooth and composite objective functions. While this method can be applied to nonconvex sparse learning problems, several hyperparameters must be set prior to running the algorithm and can be difficult to choose in practice. Indeed, the nonconvex AG method has never been applied in the context of sparse statistical learning problems with nonconvex penalties, such as SCAD and MCP.

This manuscript presents a detailed analysis of the complexity upper bound of the nonconvex AG algorithm and proposes a hyperparameter setting to accelerate convergence (Theorem 1). We further establish the rate of convergence (Theorem 2) and present a simple and useful bound to characterize our proposed optimal damping sequence (Theorem 3 and Corollary 1). Our simulation studies on penalized linear and logistic models show that the nonconvex AG method with the proposed hyperparameter selector converges considerably faster than other first-order methods. We also compare the signal recovery performance of the algorithm to that of `ncvreg`, the state-of-the-art method based on coordinate descent, showing that the proposed method outperforms the state-of-the-art coordinate descent method.

The rest of this manuscript is organised as follows. In Sects. 2, 3, 4, we will present an analysis of the nonconvex AG algorithm by Ghadimi and Lan to illustrate the algorithm as a generalization of Nesterov's AG. We also present formal results about the effect of hyperparameter settings on the complexity upper bound. Section 5 will include simulation studies for linear and logistic models penalized by SCAD and MCP penalties. The simulation studies show that i) The AG method using our proposed hyperparameter settings converges faster than commonly used first-order methods for data with various q/n and covariate correlation settings; and ii) our method outperforms the current state-of-the-art method, i.e. `ncvreg`, in terms of signal recovery performance, especially when the signal-to-noise ratios are low. The proofs for the theorems are included in the "Appendix A".

2 Motivation and setup

Having built on Nesterov's seminal work, Ghadimi and Lan (2015) considered the following composite optimization problem:

$$\min_{x\in\mathbb{R}^{q+1}} \Psi(x)+\chi(x),\ \Psi(x):=f(x)+h(x), \qquad (\mathcal{P})$$

where $f\in\mathcal{C}^{1,1}_{L_f}(\mathbb{R}^{q+1},\mathbb{R})$ is convex, $h\in\mathcal{C}^{1,1}_{L_h}(\mathbb{R}^{q+1},\mathbb{R})$ is possibly nonconvex, and χ is a convex function over a bounded domain, and $\mathcal{C}^{1,1}_{L}$ denotes the class of first-order Lipschitz smooth functions with L being the Lipschitz constant. They devised Algorithm 1 discussed in details in next section, and presented a theoretical analysis of their algorithm.

Some commonly used nonconvex penalties, such as SCAD and MCP, have a form that can naturally be decomposed into summation of a convex and a nonconvex function satisfying the conditions required by Ghadimi and Lan (2015). When such penalties are added to a smooth convex deviance measure, such as negative of typical log-likelihoods, the resulting optimization problem follows the form of optimization problem ($\mathcal{P}$). As we show below this is, in particular, the case when the deviance measure is a quadratic loss and the penalty is either SCAD or MCP. The quadratic loss plays the role of f. The other two functions, i.e. h and χ are specified for both SCAD and MCP penalties. Define

$$p_{\lambda,a,\mathrm{SCAD}}(\boldsymbol{\beta})=\chi(\boldsymbol{\beta})+h_{\mathrm{SCAD}}(\boldsymbol{\beta}), \qquad (1)$$

$$p_{\lambda,\gamma,\mathrm{MCP}}(\boldsymbol{\beta})=\chi(\boldsymbol{\beta})+h_{\mathrm{MCP}}(\boldsymbol{\beta}); \qquad (2)$$

where $\boldsymbol{\beta}:=\left[\beta_0,\beta_1,\ldots,\beta_q\right]^T$, $\chi(\boldsymbol{\beta})=\sum_{j=1}^{q}\lambda|\beta_j|$, and

$$h_{\mathrm{SCAD}}(\boldsymbol{\beta})=\sum_{j=1}^{q}\begin{cases}0; & |\beta_j|\le\lambda\\ \frac{2\lambda|\beta_j|-\beta_j^2-\lambda^2}{2(a-1)}; & \lambda<|\beta_j|<a\lambda\\ \frac{1}{2}(a+1)\lambda^2-\lambda|\beta_j|; & |\beta_j|\ge a\lambda\end{cases}\in\mathcal{C}^{1,1}_{L_{\mathrm{SCAD}}} \qquad (3)$$

$$h_{\mathrm{MCP}}(\boldsymbol{\beta})=\sum_{j=1}^{q}\begin{cases}-\frac{\beta_j^2}{2\gamma}; & |\beta_j|<\gamma\lambda\\ \frac{1}{2}\gamma\lambda^2-\lambda|\beta_j|; & |\beta_j|\ge\gamma\lambda\end{cases}\in\mathcal{C}^{1,1}_{L_{\mathrm{MCP}}} \qquad (4)$$

In the above equations, $\lambda>0, a>2, \gamma>1$ are the penalty tuning parameters. It is trivial that, in (1) and (2), $\chi(\boldsymbol{\beta})$ is convex and the remaining term is a first-order smooth concave function. In view of the optimization problem ($\mathcal{P}$), when applying SCAD/MCP on a convex $\mathcal{C}^{1,1}_{L_\ell}$ statistical learning objective function, $f=-2\ell$ will be the convex component; $h_{\mathrm{SCAD}}, h_{\mathrm{MCP}}$ will be the smooth nonconvex component with $L_{SCAD}=\frac{1}{a-1}$ and $L_{MCP}=\frac{1}{\gamma}$; and $\chi=\sum_{j=1}^{q}\lambda|\beta_j|$ will be the nonsmooth convex component. For high-dimensional statistical learning problems, the L-smoothness constant for the smooth nonconvex component, L_{SCAD} and L_{MCP}, are often negligible when compared to the greatest singular value of the design matrix (Meckes 2020). In statistical learning applications, most unconstrained problems can, in fact, be reduced to problems over a bounded domain, as information often suggests the boundedness of the variables.

3 The accelerated gradient algorithm

This Section comprises two subsections. Section 3.1 includes an algorithm proposed by Ghadimi and Lan (2015) for solving the composite optimization problem ($\mathcal{P}$). In Sect. 3.2 we propose an approach for selecting the hyperparameters of the algorithm by minimizing the complexity upper bound (10)

3.1 Nonconvex accelerated gradient method

Building on Nesterov's AG algorithm, Ghadimi and Lan (2015) proposed the following algorithm for solving the composite optimization problem ($\mathcal{P}$).

Algorithm 1 Accelerated Gradient Algorithm

Input: starting point $x_0\in\mathbb{R}^{q+1}$, $\{\alpha_k\}$ s.t. $\alpha_1=1$ and $\forall k\ge 2, 0<\alpha_k<1$, $\{\omega_k>0\}$, and $\{\delta_k>0\}$

0. Set $x_0^{ag}=x_0$ and $k=1$

1. Set

$$x_k^{md}=\alpha_k x_{k-1}^{ag}+(1-\alpha_k)x_{k-1} \qquad (5)$$

2. Compute $\nabla\Psi\left(x_k^{md}\right)$ and set

$$x_k=x_{k-1}-\delta_k\nabla\Psi\left(x_k^{md}\right)\ \text{(smooth)}\qquad x_k=\mathcal{P}\left(x_{k-1},\nabla\Psi\left(x_k^{md}\right),\delta_k\right)\ \text{(composite)} \qquad (6)$$

$$x_k^{ag}=x_k^{md}-\omega_k\nabla\Psi\left(x_k^{md}\right)\ \text{(smooth)}\qquad x_k^{ag}=\mathcal{P}\left(x_k^{md},\nabla\Psi\left(x_k^{md}\right),\omega_k\right)\ \text{(composite)} \qquad (7)$$

3. Set $k=k+1$ and go to step 1

Output: Minimizer x_N^{md}

In Algorithm 1, "smooth" represents the updating formulas for smooth problems, and "composite" represents the update formulas for composite problems, and $\mathcal{P}$ is the proximal operator defined as:

$$\mathcal{P}(x,y,c):=\arg\min_{u\in\mathbb{R}^{q+1}}\left\{\langle y,u\rangle+\frac{1}{2c}\|u-x\|^2+\chi(u)\right\}.$$

It is evident that the composite counter-part of the algorithm is the Moreau envelope smoothing of the simple nonconvex

function; for this reason, in later analysis of the algorithm, we will use smooth updating formulas for the sake of parsimony. As an interpretation of the algorithm, $\{\alpha_k\}$ controls the damping of the system, and ω_k controls the step size for the "gradient correction" update for momentum method. In what follows, Γ_k is defined recursively as:

$$\Gamma_k := \begin{cases} 1, & k = 1; \\ (1-\alpha_k)\,\Gamma_{k-1}, & k \geq 2. \end{cases}$$

Ghadimi and Lan (2015) proved that under the following conditions:

$$\alpha_k \delta_k \leq \omega_k < \frac{1}{L_\Psi},\ \forall k = 1, 2, \ldots N-1 \text{ and} \tag{8}$$

$$\frac{\alpha_1}{\delta_1 \Gamma_1} \geq \frac{\alpha_2}{\delta_2 \Gamma_2} \geq \cdots \geq \frac{\alpha_N}{\delta_N \Gamma_N}, \tag{9}$$

the rate of convergence for composite optimization problems can be illustrated by the following complexity upper bound:

$$\begin{aligned} \min_{k=1,\ldots,N} \left\| \mathcal{G}\left(x_k^{md}, \nabla\Psi\left(x_k^{md}\right), \omega_k\right)\right\|^2 \\ \leq \left[\sum_{k=1}^{N} \Gamma_k^{-1} \omega_k \left(1 - L_\Psi \omega_k\right)\right]^{-1} \\ \left[\frac{\|x_0 - x^*\|^2}{\delta_1} + \frac{2L_h}{\Gamma_N}\left(\|x^*\|^2 + M^2\right)\right]. \end{aligned} \tag{10}$$

In the above inequality, $\mathcal{G}\left(x_k^{md}, \nabla\Psi\left(x_k^{md}\right), \omega_k\right)$ is the analogue to the gradient for smooth functions defined by:

$$\mathcal{G}(x, y, c) := \frac{1}{c}\left[x - \mathcal{P}(x, y, c)\right].$$

In accelerated gradient settings, x corresponds to the past iteration, y corresponds to the smooth gradient at x, and c corresponds to the step size taken.

3.2 Hyperparameters for nonconvex accelerated gradient method

Here we discuss how hyperparameters, α_k, ω_k and δ_k can be selected to accelerate convergence of Algorithm 1 by minimizing the complexity upper bound. From Lemma 1, it is clear that the conditions (8) and (9) merely present a lower bound for the vanishing rate of $\{\alpha_k\}$. We also observe that the right-hand side of (A1) is monotonically increasing with respect to α_k; thus, to obtain the maximum values for $\{\alpha_k\}$, it is sufficient to maximize α_k recursively.

Using 5, 6, and 7, we have

$$\begin{aligned} \frac{x_{k+1}^{md} - (1-\alpha_{k+1})\,x_k^{ag}}{\alpha_{k+1}} =& \frac{x_k^{md} - (1-\alpha_k)\,x_{k-1}^{ag}}{\alpha_k} \\ & - \delta_k \nabla\Psi\left(x_k^{md}\right) \text{ and} \\ x_k^{ag} =& x_k^{md} - \omega_k \nabla\Psi\left(x_k^{md}\right). \end{aligned}$$

By sorting out the terms in the above equations, we obtain the following updating formulas:

$$x_k^{ag} = x_k^{md} - \omega_k \nabla\Psi\left(x_k^{md}\right) \tag{11}$$

$$\begin{aligned} x_{k+1}^{md} =& x_k^{ag} + \alpha_{k+1} \cdot \left(\frac{1}{\alpha_k} - \frac{\delta_k}{\omega_k}\right) \cdot \left(\omega_k \nabla\Psi\left(x_k^{md}\right)\right) \\ & + \alpha_{k+1} \cdot \left(\frac{1}{\alpha_k} - 1\right)\left(x_k^{ag} - x_{k-1}^{ag}\right) \end{aligned} \tag{12}$$

Compared to Nesterov's AG, the AG method proposed by Ghadimi and Lan differs by the convergence conditions (8) and (9), and the inclusion of the term $\alpha_{k+1} \cdot \left(\frac{1}{\alpha_k} - \frac{\delta_k}{\omega_k}\right) \cdot \left(\omega_k \nabla\Psi\left(x_k^{md}\right)\right)$ in (12). Since $\alpha_{k+1} \cdot \left(\frac{1}{\alpha_k} - \frac{\delta_k}{\omega_k}\right) \geq 0$ is implied by convergence condition (8), this added term functions as a step to reduce the magnitude of "gradient correction" presented in (11): the resulting framework will keep the same momentum compared to Nesterov's AG, but the momentum step update will occur at a midpoint between x_k^{ag} and x_k^{md} to yield x_{k+1}^{md}. Such a framework suggests that the proposed algorithm is merely a midpoint generalization in the gradient correction step of Nesterov's AG. Therefore, *the acceleration occurs to the convex component f of the objective function* Ψ. Following this intuition, we proceed to investigate the optimization hyperparameter settings for the most accelerating effect in Theorem 1 based on the idea of minimizing the complexity upper bound (10) when the objective function is convex; i.e., when $h \equiv 0$.

It can be deduced from (A1) that an increasing sequence of $\{\delta_k\}$ allows a slower vanishing rate for $\{\alpha_k\}$. Specifically, the existence of δ_1 in (10) can be explained as the following: the momentum initialization step in Algorithm 1 indicates that $x_1^{md} = x_0^{ag} = x_0$. We also have $x_1^{ag} = x_1^{md} - \omega_1 \nabla\Psi\left(x_1^{md}\right) = x_0^{ag} - \omega_1 \nabla\Psi\left(x_0\right)$ for smooth problems or $x_1^{ag} = \mathcal{P}\left(x_1^{md}, \nabla\Psi\left(x_1^{md}\right), \omega_1\right) = \mathcal{P}\left(x_0^{ag}, \nabla\Psi\left(x_0\right), \omega_1\right)$ for composite problems. In view of (12), the momentum initializes as $x_1^{ag} - x_0^{ag} = -\omega_1 \nabla\Psi\left(x_0\right)$ for smooth problems. Thus, should $\delta_1 < \omega_1$ take a smaller value, $\alpha_2 \cdot \left(\frac{1}{\alpha_1} - \frac{\delta_1}{\omega_1}\right) > 0$; i.e., x_2^{md} is a convex combination of x_1^{ag} and the initial point x_0, and the smaller δ_1 is, the closer x_2^{md} is to x_0. Meanwhile, a smaller δ_1 allows a faster increasing sequence $\{\delta_k\}$; hence a slower-vanishing sequence $\{\alpha_k\}$ can be achieved to incorporate more momentum. This process can be interpreted as follows: when x_2^{md} does not retain the full step update from the initial point x_0, more initial momentum will be allowed

to accumulate, as the initial momentum is in the same direction as the update. We therefore choose $\delta_1 = \omega_1$; i.e., to let x_2^{md} retain fully the update from x_0 in the direction of $-\omega_1 \nabla\Psi(x_0)$, such that no *excess* initial momentum will be needed to account for initial update deficiency in this direction.

4 Theoretical analysis of the algorithm

For gradient methods without a line-search procedure, the step size for the gradient correction is usually set to be a constant. Based on this convention, we assume $\omega_k = \beta$ for $k = 1, 2, \ldots, N$. Theorem 1 below presents the optimal choice of hyperparameters under mild conditions.

Theorem 1 *Assume conditions (8) and (9) hold. Let $\delta_1 = \omega_k = \omega$ and $h = 0$. Then the complexity upper bound (10) is minimized by:*

$$\bar{\alpha}_{k+1} = \frac{2}{1+\sqrt{1+\frac{4}{\bar{\alpha}_k^2}}},\ \bar{\alpha}_1 = 1, \tag{13}$$

$$\bar{\delta}_{k+1} = \frac{\bar{\omega}}{\bar{\alpha}_{k+1}}, \tag{14}$$

$$\bar{\omega} = \frac{2}{3L_\Psi}. \tag{15}$$

Proof See "Appendix A.1". □

As illustrated by the proof of the above theorem, the optimization hyperparameter settings (13), (14), and (15) allow for the greatest values of $\{\alpha_k\}$ under the constant gradient-correction step size and maximum initial update assumptions; i.e., condition 1. Such settings allow the most acceleration for the convex component. Although a greater momentum will result in a much faster convergence at the initial stage of the algorithm, it will also result in oscillations of larger magnitudes near the minimizer. Therefore, in the following theorem, we will show that the complexity upper bound will always maintain $O(1/N)$ rate of convergence. This observation implies that the accelerated gradient method's worst-case scenario is at least as good as $O(1/N)$ for gradient descent in terms of the rate of convergence.

Theorem 2 *Assume conditions (8) and (9) hold. Then under the assumptions of Theorem 1, the complexity upper bound is $O(1/N)$.*

Proof See "Appendix A.2". □

The recursive formula for optimal momentum hyperparameter, $\{\alpha_k\}$, as presented in (13), is of a rather complicated structure. The next theorem illustrates the vanishing rate of $\{\alpha_k\}$.

Theorem 3 *Let $\bar{\alpha}_1 = 1$ and (13) holds. Then*

$$\frac{2}{\left(1 + a \cdot k^{-b}\right) k + 1} < \bar{\alpha}_k \le \frac{2}{k+1}, \quad k = 1, \ldots, N, \tag{16}$$

for any $a > 0$, $0 < b < 1$, such that

$$a(1-b) \cdot 2^{2-b} - ab(1-b) \cdot 2^{-b} - 1 \ge 0. \tag{17}$$

Proof See "Appendix A.3". □

The following corollary establishes a tight bound for the damping sequence, hence providing the speed of convergence of our proposed optimal damping sequence $\{\bar{\alpha}_k\}$ to $\frac{2}{k+1}$.

Corollary 1 *The lower bound in (16) is maximized at*

$$\bar{a}_k = \frac{2^{\bar{b}_k}}{(1-\bar{b}_k)(4-\bar{b}_k)} \quad \text{and}$$

$$\bar{b}_k = \frac{2+5\left(\log\frac{2}{k}\right)+\sqrt{9\left(\log\frac{2}{k}\right)^2+4}}{2\left(\log\frac{2}{k}\right)} \quad \text{for} \quad k \ge 8.$$

The lower bound in (16) therefore becomes

$$\frac{k+1}{2} - \bar{\alpha}_k^{-1} = O(\log k) \tag{18}$$

Proof See "Appendix A.4". □

To better illustrate Corollary 1, we plot the value of $\log\left(\bar{a}_k k^{-\bar{b}_k}\right)$ v.s. (k, b) in Fig. 1. The plot shows that as k grows large, the optimizer $\bar{b}_k$ converges to 1 at a very slow rate. It also reflects on the speed of $1 + \bar{a}_k \cdot k^{-\bar{b}_k}$, the coefficient of k in the denominator of the lower bound in (16), goes to 1 as k increases.

5 Simulations studies

In this section, we conduct two sets of simulation studies for nonconvex penalized linear and logistic models. We first visualize the convergence rates and signal recovery performance for each set of simulation studies using a single simulation replicate. Second, we compare the convergence rates across the first-order methods with varying q/n ratios and covariate correlations for 100 simulation replications. Lastly, we compare the signal recovery performance using our method to the state-of-the-art method, `ncvreg` (Breheny and Huang 2011), with varying covariate correlations and signal-to-noise ratios (SNRs) for 100 simulation replications. Since the iterative complexity differs for the first-order methods and coordinate descent methods, the convergence rates in terms of the number of iterations are not directly

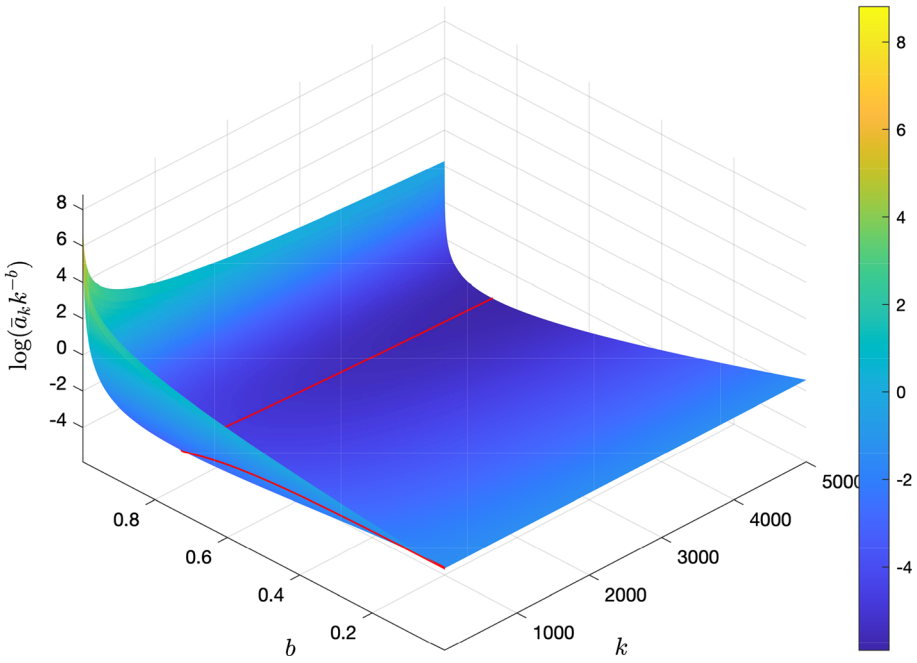

Fig. 1 Numerical plots for Corollary 1. The figure plots $\log\left(\bar{a}_k k^{-b}\right)$ v.s. k and b; the red line plots its minimizer $\bar{b}_k = \frac{2+5\left(\log\frac{2}{k}\right)+\sqrt{9\left(\log\frac{2}{k}\right)^2+4}}{2\left(\log\frac{2}{k}\right)}$ for each k. The plot reflects on the speed for the coefficient of k in the denominator of the lower bound in (16) converges to 1. The red line shows that $\bar{b}_k$ converges to 1 at an extremely slow rate

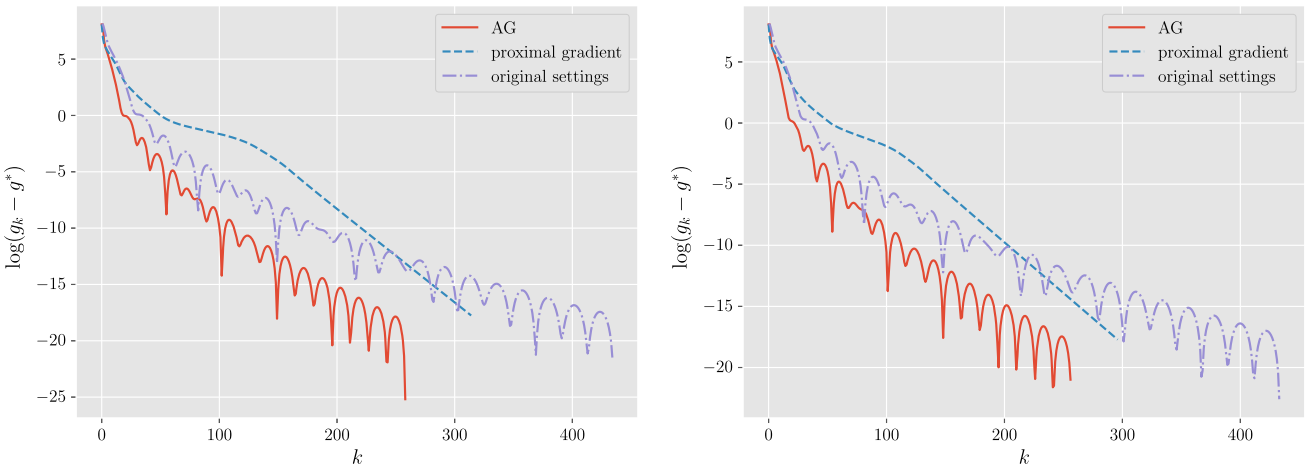

Fig. 2 Convergence rate performance of first-order methods on SCAD (left) and MCP (right) penalized linear model for a single simulation replicate. k represents the number of iterations, g_k represents the iterative objective function value, and g^* represents the minimum found by the three methods considered

comparable. Thus, we choose to compare the computing time between AG, proximal gradient descent, and coordinate descent.

5.1 Simulation setup

Linear models with the OLS loss function is a popular method for modelling a continuous response. We aim to achieve signal recovery by solving the following problem for penalized linear models:

$$\arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{q+1}} \frac{1}{2n}\|\mathbf{X}\boldsymbol{\beta}-\mathbf{y}\|_2^2 + \sum_{j=1}^{q} p_\lambda\left(\beta_j\right),$$

where $p_\lambda : \mathbb{R} \mapsto \mathbb{R}_{\geq 0}$ is the SCAD or MCP penalty function. To compare the convergence rates across the first-order methods, we choose different q/n ratios and the strength of correlation, τ, between the covariates. These two parameters are most likely to impact the convergence rates. Median and corresponding 95% bootstrap confidence intervals from 1000 bootstrap replications for the number of iterations required for the iterative objective values to make a fixed amount of descent are reported. To compare the signal recovery performance between our AG method and the state-of-the-art package `ncvreg`, we performed 100 simulation replications with varying SNRs and covariate correlations, as they directly

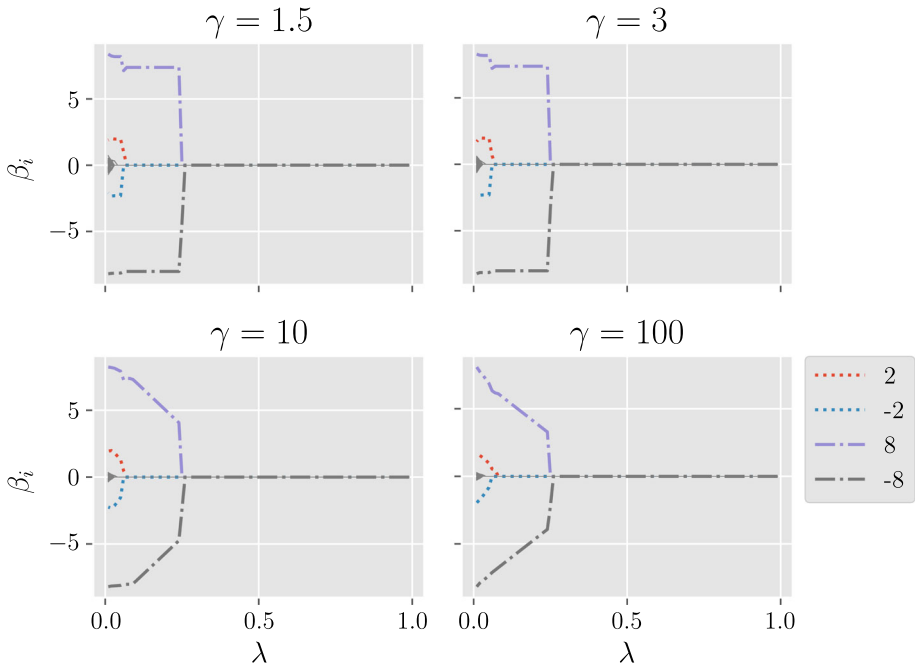

Fig. 3 Solution paths obtained using the proposed AG method for MCP-penalized linear model with different values of γ for a single simulation replicate. The behaviors of the solution path match the expected from the MCP penalized problems. The solution path behaves similarly to hard-thresholding for a small γ. As γ increases, the solution path will behave more similarly to soft-thresholding

impact the signal recovery performance. The simulation studies we performed adapt the following setups:

- The total number of observations $n = 1000$ for visualization plots and signal recovery performance comparison, and $n = 200, 500, 1000, 3000$ for convergence rate and computing time comparisons.
- For visualization purposes, we perform one simulation replicate with the number of covariates $q = 2004$, with 4 nonzero signals being $2, -2, 8, -8$. We perform 100 simulation replications with the number of covariates $q = 2050$, with 5 blocks of "true" signals equal-spaced with 500 zeros in-between for convergence rate and computing time comparison, as well as signal recovery performance comparison. For each simulation replicate, the blocks of the "true" signals are simulated from $N_{10}(0.5, 1)$, $N_{10}(5, 2)$, $N_{10}(10, 3)$, $N_{10}(20, 4)$, $N_{10}(50, 5)$, respectively.
- The design matrix, $\mathbf{X}$, is simulated from a multivariate Gaussian distribution with mean 0. The covariance matrix $\mathbf{\Sigma}$ is a $\tau-$Toeplitz matrix, where $\tau = 0.5$ for the visualization plots and $\tau = 0.1, 0.5, 0.9$ for the convergence rate and computing time comparison, as well as signal recovery performance comparison. All covariates are standardized; i.e., centered by the sample mean and scaled by the sample standard deviation.
- The signal-to-noise ratio is set as $\text{SNR} = \frac{\sqrt{\boldsymbol{\beta}_{\text{true}}^T \mathbf{\Sigma} \boldsymbol{\beta}_{\text{true}}}}{\sigma}$, where $\boldsymbol{\beta}_{\text{true}}$ are the "true" coefficient values, and σ is used as the residual standard deviation. $\text{SNR} = 5$ for visualization plots, $\text{SNR} = 3$ for convergence rate comparison, and $\text{SNR} = 1, 3, 7, 10$ for signal recovery performance comparison.
- For visualization plots, convergence rate and computing time comparisons, we take $\lambda = 0.5, a = 3.7$ for SCAD and $\lambda = 0.5, \gamma = 3$ for MCP, unless otherwise specified. For signal recovery rate comparison, λ sequence consists of 50 values equal-spaced from λ_{max}[1] to 0. The tuning parameter λ is chosen to minimize the (non-penalized) loss function value on a validation set of the same size as the training set.
- For signal recovery performance comparison, we use the same objective function as `ncvreg` to ensure that the same value of penalty tuning parameters results in the same degree of penalization. We also adapt the same strong rule setup as `ncvreg` (Lee and Breheny 2015).

To compare the gradient-based methods and the coordinate descent method, we compare the computing time when both coded in `Python/CuPy`. The coordinate descent method was coded based on the state-of-the-art pseudo-code (Breheny and Huang 2011). All of the computing was carried out on a NVIDIA A100 GPU with CUDA compute capability of 8.0 on the Narval computing cluster from Calcul Quèbec/Compute Canada. Furthermore, we also excluded the computation of the L-smoothness parameter for the coordinate descent method in our simulations.

The simulation setups for penalized logistic models are similar to those above for penalized linear models, except

[1] λ_{max} is the minimal value for λ such that all penalized coefficients are estimated as 0.

Table 1 Signal recovery performance (sample mean and standard error of $\left\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\right\|_2^2/\|\boldsymbol{\beta}_{\text{true}}\|_2^2$, positive/negative predictive values (PPV, NPV) for signal detection, and active set cardinality $|\hat{\mathcal{A}}|$) for `ncvreg` and AG with our proposed hyperparameter settings on SCAD-penalized linear model over 100 simulation replications, across varying values of SNRs and covariates correlations (τ)

$\left\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\right\|_2^2/\|\boldsymbol{\beta}_{\text{true}}\|_2^2$	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.128(0.021)	0.521(0.114)	2.839(0.497)
SNR = 1, `ncvreg`	0.131(0.02)	0.485(0.102)	2.929(0.525)
SNR = 3, AG	0.05(0.009)	0.156(0.035)	2.075(0.339)
SNR = 3, `ncvreg`	0.052(0.009)	0.156(0.028)	2.087(0.357)
SNR = 7, AG	0.022(0.004)	0.085(0.014)	1.278(0.262)
SNR = 7, `ncvreg`	0.021(0.004)	0.083(0.015)	1.3(0.262)
SNR = 10, AG	0.016(0.003)	0.065(0.011)	1.163(0.207)
SNR = 10, `ncvreg`	0.015(0.003)	0.063(0.013)	1.167(0.22)
PPV	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.747(0.134)	0.622(0.188)	0.488(0.25)
SNR = 1, `ncvreg`	0.255(0.061)	0.287(0.132)	0.286(0.19)
SNR = 3, AG	0.681(0.162)	0.551(0.206)	0.327(0.234)
SNR = 3, `ncvreg`	0.282(0.079)	0.307(0.098)	0.275(0.148)
SNR = 7, AG	0.58(0.138)	0.42(0.257)	0.197(0.141)
SNR = 7, `ncvreg`	0.32(0.065)	0.344(0.152)	0.175(0.101)
SNR = 10, AG	0.528(0.272)	0.437(0.09)	0.211(0.081)
SNR = 10, `ncvreg`	0.349(0.127)	0.409(0.1)	0.206(0.047)
NPV	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.984(0.001)	0.984(0.001)	0.979(0.001)
SNR = 1, `ncvreg`	0.987(0.001)	0.986(0.001)	0.98(0.001)
SNR = 3, AG	0.989(0.001)	0.988(0.002)	0.98(0.001)
SNR = 3, `ncvreg`	0.99(0.001)	0.989(0.001)	0.98(0.001)
SNR = 7, AG	0.992(0.001)	0.991(0.001)	0.981(0.001)
SNR = 7, `ncvreg`	0.993(0.001)	0.991(0.001)	0.981(0.001)
SNR = 10, AG	0.993(0.001)	0.992(0.001)	0.982(0.001)
SNR = 10, `ncvreg`	0.993(0.001)	0.992(0.001)	0.982(0.001)
$\lvert\hat{\mathcal{A}}\rvert$	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	25.82(8.08)	31.58(17.056)	23.11(15.166)
SNR = 1, `ncvreg`	100.88(25.582)	94.32(41.572)	42.01(20.592)
SNR = 3, AG	42.78(14.003)	55.48(20.653)	42.83(16.308)
SNR = 3, `ncvreg`	120.17(33.554)	101.75(29.498)	46.72(16.252)
SNR = 7, AG	61.89(21.881)	97.88(36.736)	86.71(26.567)
SNR = 7, `ncvreg`	115.4(23.845)	107.19(31.445)	89.74(23.1)
SNR = 10, AG	101.21(66.968)	81.17(25.325)	70.8(11.642)
SNR = 10, `ncvreg`	123.5(52.077)	90.58(40.419)	71.47(10.954)

that the active coefficients are set differently to account for the exponential scale inherent to the logistic regression. For the single-replicate visualization simulations, we let the 4 nonzero signals be 0.5, −0.5, 0.8, −0.8. For the simulations with 100 replications to compare the convergence rate and signal recovery performance, we simulate the 5 blocks of the "true" signals from N_{10} (0.5, 1), N_{10} (0.5, 1), N_{10} (−0.5, 1), N_{10} (−0.5, 1), N_{10} (1, 1), respectively. The SNR for logistic regression has the same definition as linear models, with Gaussian noise added to the generated continuous predictor $\mathbf{X}\boldsymbol{\beta}_{true}$. The binary outcomes are independent Bernoulli realizations, with probabilities being the logistic transforms of the continuous response.

5.2 Simulation results

5.2.1 Penalized linear regression

Figure 2 shows the log differences of iterative objective values for a single replicate. This figure visualizes the accelerating effect of the AG method using our proposed hyperparameter settings. Median with the corresponding 95% bootstrap CI of the number of iterations required for the

Table 2 Signal recovery performance (sample mean and standard error of $\left\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\right\|_2^2/\|\boldsymbol{\beta}_{\text{true}}\|_2^2$, positive/negative predictive values (PPV, NPV), and active set cardinality $|\hat{\mathcal{A}}|$ for signal detection) for `ncvreg` and AG with our proposed hyperparameter settings on MCP-penalized linear model over 100 simulation replications, across varying values of SNRs and covariates correlations (τ)

$\left\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\right\|_2^2/\|\boldsymbol{\beta}_{\text{true}}\|_2^2$	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.133(0.022)	0.563(0.124)	2.839(0.39)
SNR = 1, `ncvreg`	0.126(0.019)	0.494(0.112)	2.86(0.427)
SNR = 3, AG	0.049(0.01)	0.169(0.034)	1.997(0.329)
SNR = 3, `ncvreg`	0.048(0.009)	0.161(0.032)	1.92(0.34)
SNR = 7, AG	0.021(0.004)	0.088(0.016)	1.503(0.329)
SNR = 7, `ncvreg`	0.02(0.004)	0.086(0.017)	1.416(0.302)
SNR = 10, AG	0.014(0.003)	0.059(0.011)	1.084(0.272)
SNR = 10, `ncvreg`	0.014(0.003)	0.059(0.013)	1.134(0.248)
PPV	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.85(0.081)	0.744(0.161)	0.616(0.208)
SNR = 1, `ncvreg`	0.435(0.085)	0.407(0.135)	0.387(0.154)
SNR = 3, AG	0.842(0.119)	0.732(0.21)	0.506(0.286)
SNR = 3, `ncvreg`	0.505(0.112)	0.514(0.121)	0.366(0.18)
SNR = 7, AG	0.761(0.175)	0.646(0.293)	0.505(0.218)
SNR = 7, `ncvreg`	0.541(0.128)	0.547(0.173)	0.483(0.201)
SNR = 10, AG	0.801(0.099)	0.489(0.134)	0.375(0.225)
SNR = 10, `ncvreg`	0.559(0.107)	0.476(0.135)	0.377(0.225)
NPV	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.983(0.001)	0.982(0.001)	0.979(0.001)
SNR = 1, `ncvreg`	0.986(0.001)	0.984(0.001)	0.979(0.0)
SNR = 3, AG	0.988(0.001)	0.986(0.001)	0.98(0.001)
SNR = 3, `ncvreg`	0.989(0.001)	0.987(0.001)	0.98(0.001)
SNR = 7, AG	0.991(0.001)	0.989(0.001)	0.981(0.001)
SNR = 7, `ncvreg`	0.992(0.001)	0.989(0.001)	0.981(0.001)
SNR = 10, AG	0.992(0.001)	0.99(0.001)	0.982(0.001)
SNR = 10, `ncvreg`	0.993(0.001)	0.99(0.001)	0.982(0.001)
$\lvert\hat{\mathcal{A}}\rvert$	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	19.7(4.584)	20.6(9.45)	12.5(8.163)
SNR = 1, `ncvreg`	51.61(13.612)	47.32(16.093)	20.25(11.411)
SNR = 3, AG	30.55(8.437)	34.52(16.44)	25.37(14.373)
SNR = 3, `ncvreg`	60.14(15.873)	48.08(13.783)	31.0(13.981)
SNR = 7, AG	44.45(14.273)	56.95(32.804)	31.96(25.048)
SNR = 7, `ncvreg`	66.7(20.364)	58.36(24.633)	33.38(25.617)
SNR = 10, AG	43.23(11.26)	64.65(12.923)	46.58(18.186)
SNR = 10, `ncvreg`	65.36(13.06)	67.16(15.483)	46.07(19.223)

iterative objective function values to make a fixed amount of descent for 100 simulation replications are reported in Figs. 8 and 9 in "Appendix B.1". The lack of bars in the reported barplots indicates that the median of 100 replications breaks down; i.e., the corresponding proximal gradient algorithm fails to converge to the minimizer found by the three algorithms within 2000 iterations. The AG method using our hyperparameter settings converges much faster than proximal gradient and AG using the original hyperparameter settings proposed by Ghadimi and Lan for both SCAD and MCP-penalized models discussed here, as reflected in Figs. 2, 8 and 9. It can also be observed that momentum methods such as AG are much less likely to be stuck at saddle points or local minimizers than proximal gradient—this property is consistent with previous findings (Jin et al. 2017). Since the proposed AG methods belong to the class of momentum methods, the AG algorithms do not possess a descent property. As suggested by a previous study (Su et al. 2014), oscillation will occur at the end of the trajectory; the descent property will therefore vanish. This is also reflected in Figs. 2, 5—as the trajectory moves close to the optimizer, the oscillation will start to occur for the AG methods. Among all the first-order methods, the AG method with our proposed hyperparameter settings tends to converge the fastest in

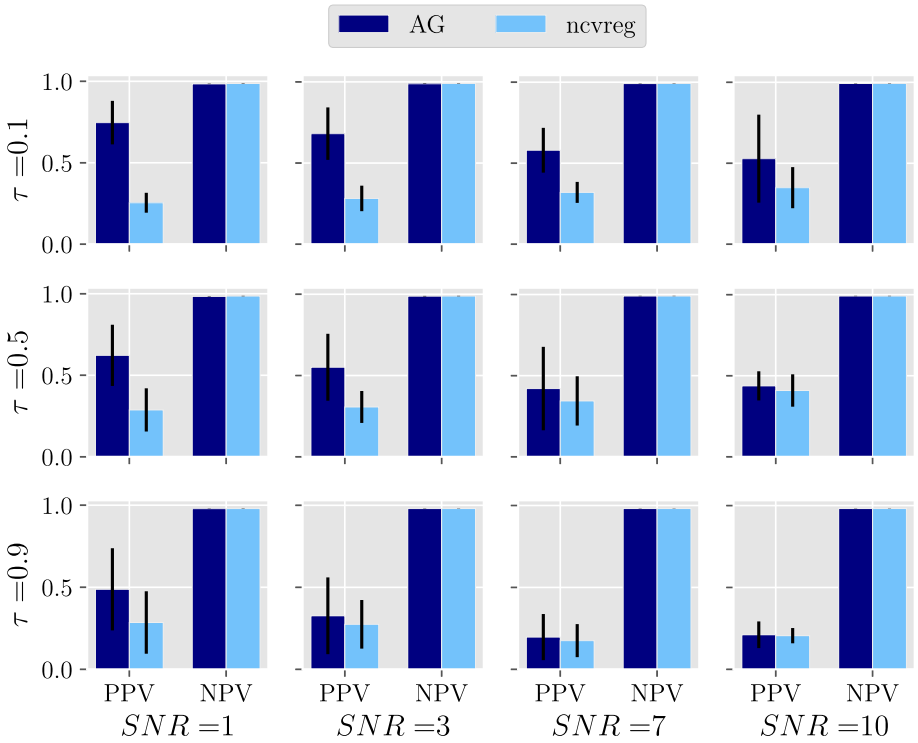

Fig. 4 Sample means for positive/negative predictive values (PPV, NPV) of signal detection across different values of covariates correlation (τ) and SNRs for AG with our proposed hyperparameter settings and `ncvreg` on SCAD-penalized linear model over 100 simulation replications. The error bars represent the standard errors

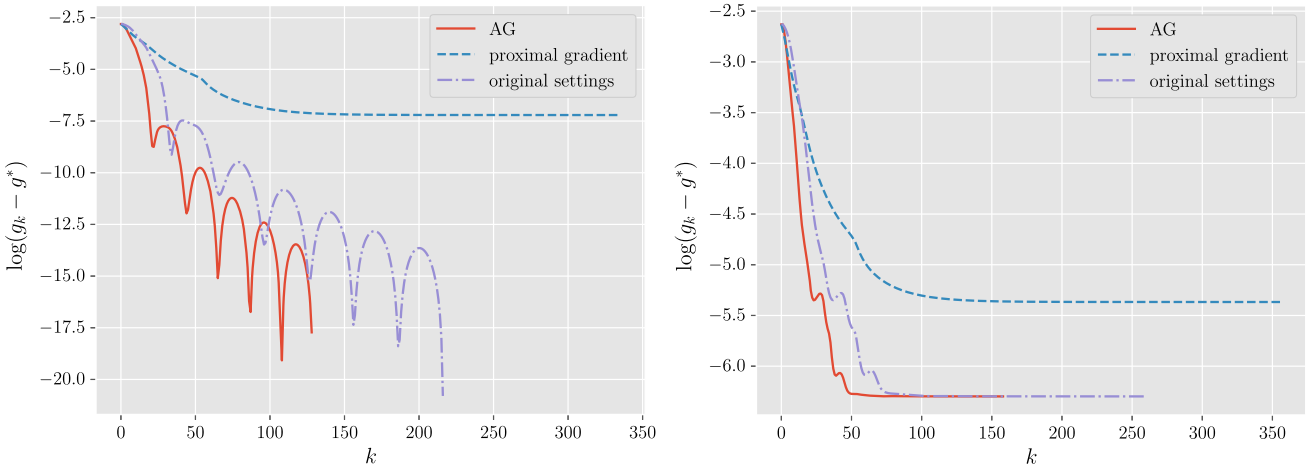

Fig. 5 Convergence rate performance of first-order methods on SCAD (left) and MCP (right) penalized logistic regression for a single simulation replicate. k represents the number of iterations, g_k represents the iterative objective function value, and g^* represent the minimum found by the three methods considered

all scenarios considered, as illustrated by Figs. 8 and 9 in "Appendix B.1". The observed standard errors among 100 simulation replications are rather small, suggesting that the halting time retains predictable for high-dimensional models, which agrees with the recent findings (Paquette et al. 2020).

Figures 10 and 11 report median with the corresponding 95% bootstrap CI of the computing time (in seconds) required for the infinity norm of the two consecutive iterations $\left\|\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)}\right\|_\infty$ to fall below 10^{-4} for 100 simulation replications. It can be observed that the computing time for AG with suggested settings is much shorter than the computing time for coordinate descent.

To visualize the signal recovery performance using our proposed method, Fig. 3 plots the solution paths for the MCP-penalized linear model with different values of γ. The grey lines in Fig. 3 represent the recovered values for the noise variables. AG method performs very well when applied to signal recovery problems for nonconvex-penalized linear models. Figure 3 serves as an arbitrary instance that the recovered signals using our method exhibit the expected pattern with MCP—as λ decreases, the degree of penalization

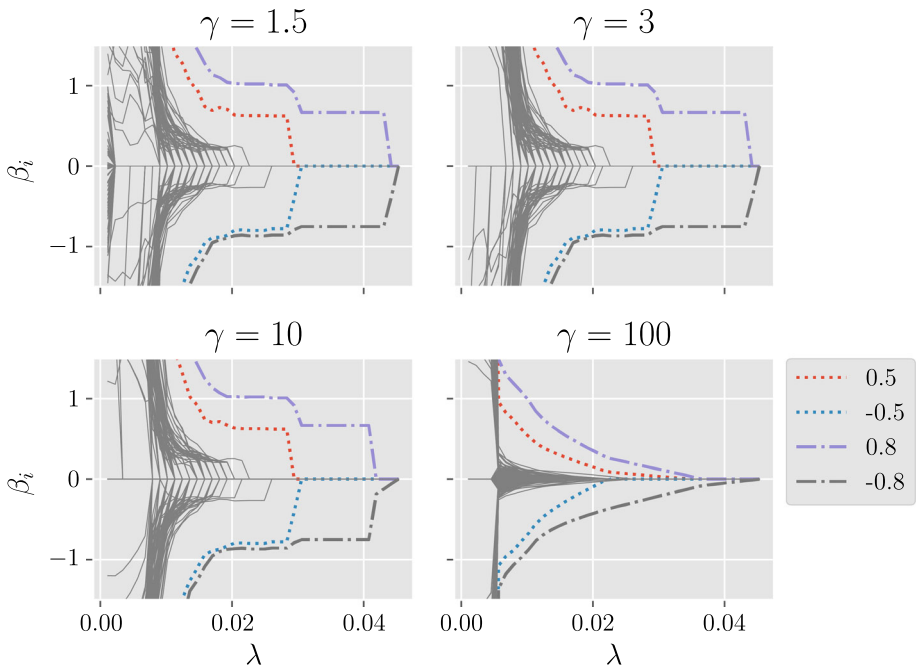

Fig. 6 Solution paths obtained using the proposed AG method for MCP-penalized logistic regression with different values of γ for a single simulation replicate. The behaviors of the solution path match the expected from the MCP penalized problems. The solution path behaves similarly to hard-thresholding for a small γ. As γ increases, the solution path will behave more similarly to soft-thresholding

decreases, and more false-positive signals will be selected. The stable solution path for the recovered signals suggests that the algorithm does not converge to a point far away from the "true" coefficients.

To further illustrate the signal recovery performance, the means and standard errors for the scaled estimation error $\frac{\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\|_2^2}{\|\boldsymbol{\beta}_{\text{true}}\|_2^2}$, positive/negative predictive values (PPV, NPV), and active set cardinality across 100 replications are reported in Tables 1 and 2 in "Appendix B.1". In what follows, $\mathcal{A}$ denotes the set of nonzero "true" coefficients and $\hat{\mathcal{A}}$ denotes the set of nonzero coefficients selected by the model. PPV and NPV use the following definitions:

$$\text{PPV} := \frac{|\mathcal{A} \cap \hat{\mathcal{A}}|}{|\hat{\mathcal{A}}|}, \quad \text{NPV} := \frac{|\mathcal{A}^C \cap \hat{\mathcal{A}}^C|}{|\hat{\mathcal{A}}^C|}.$$

Sample means and standard errors for PPV and NPV from Table 1 are further visualized in Fig. 4. When applied to sparse learning problems, the signal recovery performance of our proposed method often outperforms `ncvreg`, the current state-of-the-art method (Breheny and Huang 2011), particularly in terms of the positive predictive values (PPV). This can be observed from Fig. 4 and Tables 1, 2 from "Appendix B.1". This observation is especially evident when the signal-to-noise ratios are low. At the same time, $\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\|_2^2/\|\boldsymbol{\beta}_{\text{true}}\|_2^2$ for both methods are close. As the SNR increases, the validation set becomes more similar to the training set, causing the chosen model to have a smaller λ. The model size will therefore increase, which will decrease the value of PPV.

5.2.2 Penalized logistic regression

The simulation results reflected in Figs. 5 and 6, as well as Figs. 12, 13 and Tables 3, 4 in "Appendix B.2" suggest similar findings for penalized logistic models to our findings for penalized linear models as discussed in Sect. 5.2.1. We further note that when applied to penalized logistic models, the coordinate descent method often fails to converge, resulting in overall poor performance in positive predictive values as reflected in Fig. 7 and Tables 3, 4 in "Appendix B.2". When it does converge, the coordinate descent method does so at a very slow rate. In comparison, our proposed method has a convergence guarantee in theory and converges within a reasonable number of iterations in our simulation studies, as shown in Figs. 8, 9 in "Appendix B.2". In our computing time comparison, we used identical simulation setups and convergence standard for both the AG method and coordinate descent method, running both on a NVIDIA A100 GPU with CUDA compute capability of 8.0 from Compute Canada; the submitted simulation job finished well within 20 minutes for both SCAD and MCP-penalized logistic models when using the AG method, but exceeded the 7-day computing time limit imposed on the Narval cluster when using the coordinate descent method.

6 Discussion

We considered a recently developed generalization of Nesterov's accelerated gradient method for nonconvex optimization, and we have discussed its potential in sparse statistical

Table 3 Signal recovery performance (sample mean and standard error of $\left\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\right\|_2^2/\|\boldsymbol{\beta}_{\text{true}}\|_2^2$, positive/negative predictive values (PPV, NPV), and active set cardinality $|\hat{\mathcal{A}}|$ for signal detection) for `ncvreg` and AG with our proposed hyperparameter settings on SCAD-penalized logistic model over 100 simulation replications, across varying values of SNRs and covariates correlations (τ)

$\left\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\right\|_2^2/\|\boldsymbol{\beta}_{\text{true}}\|_2^2$	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.768(0.047)	0.81(0.041)	0.896(0.04)
SNR = 1, `ncvreg`	0.803(0.033)	0.84(0.033)	0.903(0.037)
SNR = 3, AG	0.556(0.057)	0.656(0.054)	0.839(0.056)
SNR = 3, `ncvreg`	0.603(0.053)	0.682(0.055)	0.813(0.053)
SNR = 7, AG	0.377(0.076)	0.521(0.073)	0.779(0.072)
SNR = 7, `ncvreg`	0.438(0.054)	0.537(0.074)	0.735(0.074)
SNR = 10, AG	0.311(0.077)	0.474(0.073)	0.757(0.079)
SNR = 10, `ncvreg`	0.377(0.064)	0.481(0.079)	0.712(0.078)
PPV	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.8(0.079)	0.779(0.1)	0.697(0.126)
SNR = 1, `ncvreg`	0.221(0.045)	0.265(0.079)	0.309(0.169)
SNR = 3, AG	0.875(0.054)	0.859(0.065)	0.765(0.096)
SNR = 3, `ncvreg`	0.244(0.052)	0.273(0.072)	0.273(0.133)
SNR = 7, AG	0.901(0.052)	0.881(0.057)	0.788(0.098)
SNR = 7, `ncvreg`	0.27(0.04)	0.271(0.079)	0.267(0.136)
SNR = 10, AG	0.915(0.048)	0.899(0.054)	0.789(0.097)
SNR = 10, `ncvreg`	0.29(0.05)	0.279(0.072)	0.26(0.123)
NPV	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.982(0.001)	0.98(0.001)	0.978(0.001)
SNR = 1, `ncvreg`	0.987(0.002)	0.985(0.002)	0.98(0.001)
SNR = 3, AG	0.985(0.002)	0.982(0.001)	0.979(0.001)
SNR = 3, `ncvreg`	0.99(0.002)	0.987(0.002)	0.98(0.001)
SNR = 7, AG	0.987(0.002)	0.984(0.001)	0.979(0.001)
SNR = 7, `ncvreg`	0.992(0.001)	0.988(0.001)	0.98(0.001)
SNR = 10, AG	0.988(0.002)	0.984(0.001)	0.979(0.001)
SNR = 10, `ncvreg`	0.992(0.001)	0.988(0.001)	0.98(0.001)
$\lvert\hat{\mathcal{A}}\rvert$	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	17.07(3.91)	13.4(3.365)	7.62(2.134)
SNR = 1, `ncvreg`	120.14(28.882)	86.49(24.421)	39.41(19.448)
SNR = 3, AG	23.34(4.203)	16.59(3.459)	8.69(2.082)
SNR = 3, `ncvreg`	134.85(29.96)	98.48(28.434)	42.47(15.014)
SNR = 7, AG	26.98(4.58)	19.46(3.659)	9.79(2.246)
SNR = 7, `ncvreg`	130.33(22.255)	105.03(28.123)	48.81(19.059)
SNR = 10, AG	27.95(4.462)	19.57(3.141)	10.24(2.346)
SNR = 10, `ncvreg`	124.58(23.016)	103.49(27.66)	50.64(21.138)

learning with nonconvex penalties. An important issue concerning this algorithm is the selection of its sequences of hyperparameters. We present an explicit solution to this problem by minimizing the algorithm's complexity upper bound, hence accelerating convergence of the algorithm. Our simulation studies indicate that among first-order methods, the AG method using our proposed hyperparameter settings achieves a convergence rate considerably faster than other first-order methods such as the AG method using the original proposed hyperparameter settings or proximal gradient. Our simulations also show that signal recovery using our proposed method generally outperforms `ncvreg`, the current state-of-the-art method. This performance gain is much more pronounced for penalized linear models when the signal-to-noise ratios are low. For penalized logistic regression, the performance gain observed is consistent across various covariates correlation and signal-to-noise ratio settings. Compared to coordinate-wise minimization methods, our proposed method is less challenged by low signal-to-noise ratios and is feasible to implement in parallel. Given today's computing facilities, parallel computing is particularly meaningful for large datasets (Parnell et al. 2020). We also show this gain in parallel computing performance by comparing computing time on a GPU. Furthermore, our

Table 4 Signal recovery performance (sample mean and standard error of $\left\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\right\|_2^2/\|\boldsymbol{\beta}_{\text{true}}\|_2^2$, positive/negative predictive values (PPV, NPV), and active set cardinality $|\hat{\mathcal{A}}|$ for signal detection) for `ncvreg` and AG with our proposed hyperparameter settings on MCP-penalized logistic model over 100 simulation replications, across varying values of SNRs and covariates correlations (τ)

$\left\|\boldsymbol{\beta}_{\text{true}}-\hat{\boldsymbol{\beta}}\right\|_2^2/\|\boldsymbol{\beta}_{\text{true}}\|_2^2$	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.769(0.044)	0.808(0.041)	0.897(0.043)
SNR = 1, `ncvreg`	0.795(0.036)	0.829(0.032)	0.903(0.038)
SNR = 3, AG	0.555(0.058)	0.654(0.053)	0.834(0.054)
SNR = 3, `ncvreg`	0.605(0.049)	0.674(0.054)	0.825(0.057)
SNR = 7, AG	0.383(0.08)	0.521(0.069)	0.779(0.07)
SNR = 7, `ncvreg`	0.438(0.057)	0.533(0.07)	0.761(0.071)
SNR = 10, AG	0.31(0.079)	0.469(0.073)	0.753(0.076)
SNR = 10, `ncvreg`	0.381(0.061)	0.48(0.082)	0.737(0.077)
PPV	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.879(0.06)	0.859(0.058)	0.779(0.087)
SNR = 1, `ncvreg`	0.372(0.068)	0.401(0.106)	0.375(0.157)
SNR = 3, AG	0.906(0.05)	0.889(0.05)	0.805(0.086)
SNR = 3, `ncvreg`	0.43(0.065)	0.445(0.106)	0.395(0.126)
SNR = 7, AG	0.919(0.044)	0.903(0.05)	0.809(0.102)
SNR = 7, `ncvreg`	0.463(0.063)	0.45(0.104)	0.417(0.145)
SNR = 10, AG	0.918(0.045)	0.911(0.038)	0.804(0.111)
SNR = 10, `ncvreg`	0.502(0.069)	0.468(0.095)	0.412(0.137)
NPV	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	0.981(0.001)	0.98(0.001)	0.978(0.001)
SNR = 1, `ncvreg`	0.986(0.002)	0.983(0.001)	0.978(0.001)
SNR = 3, AG	0.985(0.002)	0.982(0.001)	0.979(0.001)
SNR = 3, `ncvreg`	0.989(0.002)	0.985(0.001)	0.979(0.001)
SNR = 7, AG	0.987(0.002)	0.984(0.001)	0.98(0.001)
SNR = 7, `ncvreg`	0.991(0.002)	0.986(0.001)	0.98(0.001)
SNR = 10, AG	0.988(0.002)	0.984(0.001)	0.98(0.001)
SNR = 10, `ncvreg`	0.991(0.001)	0.987(0.001)	0.98(0.001)
$\lvert\hat{\mathcal{A}}\rvert$	$\tau = 0.1$	0.5	0.9
SNR = 1, AG	13.86(3.082)	11.42(2.776)	6.72(1.744)
SNR = 1, `ncvreg`	59.83(14.138)	42.1(12.546)	19.72(8.393)
SNR = 3, AG	21.86(4.313)	15.84(3.036)	8.84(1.938)
SNR = 3, `ncvreg`	66.57(13.203)	48.28(14.5)	22.81(9.784)
SNR = 7, AG	25.75(4.776)	18.78(3.189)	10.33(2.565)
SNR = 7, `ncvreg`	69.44(11.876)	52.54(13.638)	24.63(8.741)
SNR = 10, AG	27.53(4.649)	19.55(3.093)	11.06(2.877)
SNR = 10, `ncvreg`	65.38(10.776)	51.66(12.785)	25.59(9.428)

proposed method has weaker convergence conditions and can be applied to a class of problems that do not have an explicit solution to the coordinate-wise objective function. For example, linear mixed models for grouped or longitudinal data involve the inverse of a large covariance matrix. Decomposition of this covariance matrix is necessary to apply the coordinate descent method. However, such decomposition can be computationally costly and numerically unstable (Quarteron 2000). On the other hand, matrix decomposition is not needed for first-order methods, as numerically stable yet computationally efficient approaches such as conjugate gradient can be adapted when applying our proposed method. The proposed nonconvex AG method can be applied to a wide range of statistical learning problems, opening various future research opportunities in statistical machine learning and statistical genetics.

7 Disclaimer

All codes to reproduce the simulation results of this paper and outputs from Calcul Quebec/Compute Canada can be found on the following GitHub repository:

https://github.com/Kaiyangshi-Ito/nonconvexAG

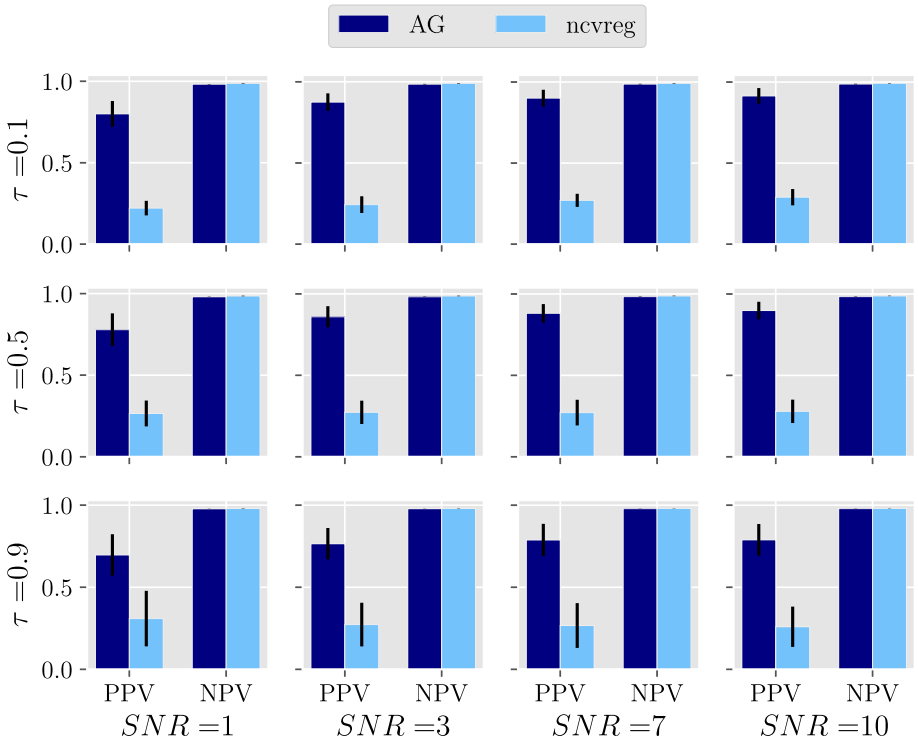

Fig. 7 Sample means for Positive/Negative Predictive Values (PPV, NPV) of signal detection across different values of covariates correlation (τ) and SNRs for AG with our proposed hyperparameter settings and `ncvreg` on SCAD-penalized logistic model over 100 simulation replications. The error bars represent the standard error

Appendix A Proofs

We first establish the following Lemma needed for the proof of Theorem 1.

A.1 Proof of Theorem 1

The following lemma is needed in the proof of Theorem 1.

Lemma 1 *Assume that* $\forall k = 1, 2, \ldots, N$, *the convergence conditions* (8) *and* (9) *hold, then we have the following recursive relation:*

$$\alpha_{k+1} \leq \frac{1}{1 + \frac{\delta_k/\delta_{k+1}}{\alpha_k}}. \tag{A1}$$

Proof The convergence conditions (8) and (9) gives that $\forall k = 1, 2, \ldots, N-1$,

$$\begin{aligned}
&\alpha_{k+1}\delta_{k+1} \leq \omega_{k+1} \Leftrightarrow \alpha_{k+1} \leq \frac{\omega_{k+1}}{\delta_{k+1}}, \text{ and} \\
&\frac{\alpha_k}{\delta_k \Gamma_k} \geq \frac{\alpha_{k+1}}{\delta_{k+1}\Gamma_{k+1}} \Leftrightarrow \frac{\alpha_k}{\delta_k} \\
&\geq \frac{\alpha_{k+1}}{\delta_{k+1}(1-\alpha_{k+1})} \Leftrightarrow \alpha_{k+1} \leq \frac{\alpha_k \delta_{k+1}}{\alpha_k \delta_{k+1} + \delta_k}.
\end{aligned}$$

Following above two inequalities, we have that

$$\alpha_{k+1} \leq \min\left\{\frac{\omega_{k+1}}{\delta_{k+1}}, \frac{\alpha_k \delta_{k+1}}{\alpha_k \delta_{k+1} + \delta_k}\right\}. \tag{A2}$$

We observe that in (A2), $\frac{\omega_{k+1}}{\delta_{k+1}}$ is monotonically decreasing with respect to δ_{k+1} on $\mathbb{R}_+$; while $\frac{\alpha_k \delta_{k+1}}{\alpha_k \delta_{k+1} + \delta_k}$ is monotonically increasing with respect to δ_{k+1} on $\mathbb{R}_+$. This suggests:

$$\begin{aligned}
&\arg\max_{\delta_{k+1}>0}\left(\min\left\{\frac{\omega_{k+1}}{\delta_{k+1}}, \frac{\alpha_k \delta_{k+1}}{\alpha_k \delta_{k+1} + \delta_k}\right\}\right) \\
&= \left\{\frac{\omega_{k+1} + \sqrt{\omega_{k+1}^2 + \frac{4\omega_{k+1}\delta_k}{\alpha_k}}}{2}\right\}.
\end{aligned} \tag{A3}$$

That is, the inequality constraints conditions (8) and (9) for convergence are merely a lower bound on the *vanishing rate* of $\{\alpha_k\}$. Therefore it follows from (8) and the (necessary) optimality condition for (A3) that

$$\begin{aligned}
\alpha_{k+1} &\leq \frac{2\omega_{k+1}}{\omega_{k+1} + \sqrt{\omega_{k+1}^2 + \frac{4\omega_{k+1}\delta_k}{\alpha_k}}} \leq \frac{2}{1 + \sqrt{1 + \frac{4\delta_k}{\alpha_k \omega_{k+1}}}} \\
&= \frac{2}{1 + \sqrt{1 + \frac{4\delta_k/\delta_{k+1}}{\alpha_k \alpha_{k+1}}}}.
\end{aligned} \tag{A4}$$

By simplifying (A1), we have:

$$\alpha_{k+1} \leq \frac{1}{1 + \frac{\delta_k/\delta_{k+1}}{\alpha_k}}.$$

□

We now proceed with the proof of Theorem 1.

Proof The complexity upper bound (10) under the given conditions can be simplified as:

$$\begin{aligned}
&\left[\sum_{k=1}^{N}\Gamma_k^{-1}\omega_k\left(1-L_\Psi\omega_k\right)\right]^{-1}\\
&\left[\frac{\|x_0-x^*\|^2}{\delta_1}+\frac{2L_f}{\Gamma_N}\left(\|x^*\|^2+M^2\right)\right]\\
=&\left[\sum_{k=1}^{N}\Gamma_k^{-1}\omega_k\left(1-L_\Psi\omega_k\right)\right]^{-1}\cdot\frac{\|x_0-x^*\|^2}{\delta_1}\\
=&\frac{1}{\omega\left(1-L_\Psi\omega\right)}\left(\sum_{k=1}^{N}\Gamma_k^{-1}\right)^{-1}\cdot\frac{\|x_0-x^*\|^2}{\omega}\\
=&\left(\sum_{k=1}^{N}\Gamma_k^{-1}\right)^{-1}\cdot\frac{\|x_0-x^*\|^2}{\omega^2\left(1-L_\Psi\omega\right)}. \qquad \text{(A5)}
\end{aligned}$$

Observe that $\left(\sum_{k=1}^{N}\Gamma_k^{-1}\right)^{-1}$ is monotonically decreasing with respect to α_k for all $k=1,2,\ldots,N$. This property implies that (A5) is minimized when α_k attains its greatest value for $k=1,2,\ldots,N$.

Condition $\delta_1=\omega_k=\omega$ gives that

$$\omega_1=\delta_1=\alpha_1\delta_1.$$

Since the upper bound for α_{k+1} presented in (A1) is monotonically increasing with respect to α_k, it then follows inductively from the (necessary) optimality condition of (A2) that

$$\alpha_{k+1}\le\frac{1}{1+\frac{\delta_k/\delta_{k+1}}{\alpha_k}}=\frac{1}{1+\frac{\alpha_{k+1}}{\alpha_k^2}},$$

which simplifies to

$$\alpha_{k+1}\le\frac{2}{1+\sqrt{1+\frac{4}{\alpha_k^2}}}.$$

While $\omega^2\left(1-L_\Psi\omega\right)$ should be maximized to minimize the value of (A5), which implies the minimizer for ω is

$$\bar{\omega}=\frac{2}{3L_\Psi}.$$

And $\bar{\lambda}_{k+1}=\frac{\bar{\omega}}{\bar{\alpha}_{k+1}}$ follows directly form the necessary optimality condition for (A2). It is trivial to check that $\left(\{\bar{\alpha}_k\},\{\bar{\delta}_k\},\bar{\omega}\right)$ is feasible under given constraints (8) and (9). □

A.2 Proof of Theorem 2

Proof Consider arbitrary $k=2,\ldots,N$, then $\alpha_k\in(0,1)$ by definition. In the convergence conditions (8) and (9), this gives us that

$$\frac{\alpha_{k+1}}{\alpha_k}\le\frac{2}{\alpha_k+\sqrt{\alpha_k^2+4}}\in\left(\frac{\sqrt{5}-1}{2},1\right).$$

Thus, $\{\alpha_k\}$ is a bounded monotonically decreasing sequence, and $\alpha_2\le\frac{2}{1+\sqrt{1+\frac{4}{1^2}}}=\frac{\sqrt{5}-1}{2}$ further implies that $\forall k\ge 2,\ \alpha_k\in(0,\frac{\sqrt{5}-1}{2}]$.

For all $k\ge 2$, $\alpha_k\in(0,1)$ implies that $1-\alpha_k\in(0,1)$. Therefore, $\Gamma_k^{-1}=\frac{1}{(1-\alpha_2)(1-\alpha_3)\cdots(1-\alpha_k)}$ is monotonically increasing with respect to k. Thus, $\sum_{k=1}^{N}\Gamma_k^{-1}=O(N)$, which implies that $\left(\sum_{k=1}^{N}\Gamma_k^{-1}\right)^{-1}\cdot C_1=O(1/N)$.

Observe that

$$\begin{aligned}
0<&\left(\Gamma_N\sum_{k=1}^{N}\frac{1}{\Gamma_k}\right)^{-1}=\frac{1}{N\cdot\Gamma_N}\cdot\frac{N}{\sum_{k=1}^{N}\frac{1}{\Gamma_k}}\\
\le&\frac{1}{N\cdot\Gamma_N}\cdot\left(\prod_{k=1}^{N}\Gamma_k\right)^{\frac{1}{N}}=\frac{1}{N}\cdot\left(\prod_{k=1}^{N}\frac{\Gamma_k}{\Gamma_N}\right)^{\frac{1}{N}}\\
=&\frac{1}{N}\cdot\left(\prod_{k=1}^{N}\frac{\Gamma_N}{\Gamma_k}\right)^{-\frac{1}{N}}=\frac{1}{N}\cdot\left(\prod_{k=2}^{N}(1-\alpha_k)^k\right)^{-\frac{1}{N}}\\
=&\frac{1}{N}\cdot\prod_{k=2}^{N}(1-\alpha_k)^{-\frac{k}{N}}, \qquad \text{(A6)}
\end{aligned}$$

where the inequality in (A6) follows from the harmonic mean-geometric mean inequality.

Consider arbitrary $N\in\mathbb{N}$, now we are to prove that $\forall k=1,2,\ldots,N,\ \alpha_k\le\frac{2}{k+1}$. By definition, $\alpha_1=1\le 1$. Assume that $\alpha_k\le\frac{2}{k+1}$, then by the convergence conditions,

$$\begin{aligned}
\alpha_{k+1}\le&\frac{2}{1+\sqrt{1+\frac{4}{\alpha_k^2}}}\\
\le&\frac{2}{1+\sqrt{1+4/\left(\frac{2}{k+1}\right)^2}}\\
=&\frac{2}{1+\sqrt{2+2k+k^2}}\\
<&\frac{2}{k+2}.
\end{aligned}$$

Thus, by mathematical induction, $\forall k=1,2,\ldots,N,\ \alpha_k\le\frac{2}{k+1}$. Hence, $\sum_{k=1}^{N}\frac{k}{N}\alpha_k<\sum_{k=1}^{N}\frac{k}{N}\cdot\frac{2}{k}=\sum_{k=1}^{N}\frac{2}{N}=2<\infty$ as $N\to\infty$.

Furthermore, we have that $\forall x \in (0, \frac{\sqrt{5}-1}{2}]$, $-\log(1-x) < x$. Combined with the fact that $\forall k \geq 2$, $\alpha_k \in (0, \frac{\sqrt{5}-1}{2}]$, we have that $\forall k \geq 2$, $-\log(1-\alpha_k) < \alpha_k$. Thus,

$$\log\left(\prod_{k=2}^{N}(1-\alpha_k)^{-\frac{k}{N}}\right) = -\sum_{k=2}^{N}\frac{k}{N}\log(1-\alpha_k) < \sum_{k=2}^{N}\frac{k}{N}\alpha_k \leq 2 < \infty.$$

Therefore, $\prod_{k=2}^{N}(1-\alpha_k)^{-\frac{k}{N}}$ is also upper bounded as $N \to \infty$, which implies that

$$\left(\sum_{k=1}^{N}\frac{\Gamma_N}{\Gamma_k}\right)^{-1} \leq \frac{1}{N}\cdot\prod_{k=2}^{N}(1-\alpha_k)^{-\frac{k}{N}} = O(1/N).$$

Hence, $\left(\sum_{k=1}^{N}\frac{\Gamma_N}{\Gamma_k}\right)^{-1}\cdot C_2 = O(1/N)$. Therefore, $\left(\sum_{k=1}^{N}\Gamma_k^{-1}\right)^{-1}\cdot C_1 + \left(\sum_{k=1}^{N}\frac{\Gamma_N}{\Gamma_k}\right)^{-1}\cdot C_2 = O(1/N)$. □

A.3 Proof of Theorem 3

Proof $\bar{\alpha}_k \leq \frac{2}{k+1}$ for $k = 1, 2, \ldots, N$ has already been proved in the proof of Theorem 2. For the left inequality, note that $\bar{\alpha}_1 = 1 \geq \frac{2}{2+a}$ for $a > 0$; for $k \geq 2$, we are to prove a stronger inequality:

$$\bar{\alpha}_k \geq \frac{2}{\sqrt{\left(1+a\cdot k^{-b}\right)k\left[\left(1+a\cdot k^{-b}\right)k+2\right]}}. \tag{A7}$$

For $k = 2$, condition (17) implies that

$$a\cdot 2^{-b} \geq \frac{1}{(1-b)(4-b)} > \frac{1}{4} > \sqrt{5}-2 \text{ for } 0 < b < 1, \tag{A8}$$

which suggests $\bar{\alpha}_2 = \frac{2}{1+\sqrt{5}} \geq \frac{2}{\sqrt{(1+a\cdot 2^{-b})\cdot 2[(1+a\cdot 2^{-b})\cdot 2+2]}}$ by simple algebra. Assume (A7) holds for $k = t$, then

$$\begin{aligned}\bar{\alpha}_{t+1} &= \frac{2}{1+\sqrt{1+\frac{4}{\bar{\alpha}_t^2}}}\\ &\geq \frac{2}{1+\sqrt{1+4/\left(2/\sqrt{\left(1+a\cdot t^{-b}\right)t\left[\left(1+a\cdot t^{-b}\right)t+2\right]}\right)^2}}\\ &= \frac{2}{1+\sqrt{1+\left(1+a\cdot t^{-b}\right)t\left[\left(1+a\cdot t^{-b}\right)t+2\right]}}\\ &= \frac{2}{\left(1+a\cdot t^{-b}\right)t+2}\\ &\geq \frac{2}{\sqrt{\left(1+a\cdot(t+1)^{-b}\right)(t+1)\left[\left(1+a\cdot(t+1)^{-b}\right)(t+1)+2\right]}};\end{aligned} \tag{A9}$$

and (A9) follows from

$$\begin{aligned}&\left(1+a\cdot(t+1)^{-b}\right)(t+1)\left[\left(1+a\cdot(t+1)^{-b}\right)(t+1)+2\right]\\ &-\left[\left(1+a\cdot t^{-b}\right)t+2\right]^2\\ =&a^2\left[(t+1)^{2-2b}-t^{2-2b}\right]+2at\left[(t+1)^{1-b}-t^{1-b}\right]\\ &+4a\left[(t+1)^{1-b}-t^{1-b}\right]-1\\ \geq&2at\left[(t+1)^{1-b}-t^{1-b}\right]-1\\ =&2at^{2-b}\left[\left(1+\frac{1}{t}\right)^{1-b}-1\right]-1\\ \geq&2at^{2-b}\left[1+(1+b)t^{-1}-\frac{1}{2}b(1-b)t^{-2}-1\right]-1 \qquad \text{(A10)}\\ =&2a(1-b)t^{1-b}-ab(1-b)t^{-b}-1 \geq 0. \qquad \text{(A11)}\end{aligned}$$

(A10) follows from binomial approximation inequality; $a > 0$ and $0 < b < 1$ suggest that $2a(1-b)k^{1-b} - ab(1-b)k^{-b} - 1$ is monotonically increasing with respect to k for $k > 0$, condition (17) therefore implies that $2a(1-b)k^{1-b} - ab(1-b)k^{-b} - 1 \geq 0$ for all $k \geq 2$, which is (A11).

And proof of the left inequality for $k \geq 2$ proceeds as the following:

$$\begin{aligned}\bar{\alpha}_k &\geq \frac{2}{\sqrt{\left(1+a\cdot k^{-b}\right)k\left[\left(1+a\cdot k^{-b}\right)k+2\right]}}\\ &> \frac{2}{\sqrt{\left(1+a\cdot k^{-b}\right)k\left[\left(1+a\cdot k^{-b}\right)k+2\right]+1}}\\ &= \frac{2}{\left(1+a\cdot k^{-b}\right)k+1}.\end{aligned}$$

□

A.4 Proof of Corollary 1

Proof Observe that the lower bound of (16) is monotonically decreasing with respect to a under given conditions. Constraint (17) implies (A8), which further suggests that

$$a \geq \frac{2^b}{(1-b)(4-b)} > 0 \text{ for } 0 < b < 1;$$

i.e., $\bar{a}_k = \frac{(2/k)^{\bar{b}_k}}{(1-\bar{b}_k)(4-\bar{b}_k)}$. Thus, maximizing the lower bound of (16) is equivalent to minimize the convex function

$\log \frac{(2/k)^b}{(1-b)(4-b)}$ with respect to b over a open set $(0, 1)$. First-order sufficient optimality condition gives the unique optimizer

$$\bar{b}_k = \frac{2 + 5\left(\log \frac{2}{k}\right) + \sqrt{9\left(\log \frac{2}{k}\right)^2 + 4}}{2\left(\log \frac{2}{k}\right)} \in (0, 1)$$

for $k \geq 8$. Simple algebra shows that $\lim_{k\to\infty} \frac{\bar{a}_k k^{1-\bar{b}_k}}{\log k} = \frac{2}{3}e$. Thus, the lower bound in Theorem 3 becomes $\frac{k+1}{2} - \bar{\alpha}_k^{-1} = O(\log k)$. □

Appendix B Further simulations

B.1 Penalized linear model

In Fig. 8 and 9, the red bar represents AG using our proposed hyperparameter settings, blue bar represents proximal gradient, and the purple bar represents AG using the original hyperparameter settings (Ghadimi and Lan 2015). It is evident that for penalized linear models, AG using our hyperparameter settings outperforms proximal gradient or AG using the original proposed hyperparameter settings considerably.

In Fig. 10 and 11, the red bar represents AG using our proposed hyperparameter settings, blue bar represents proximal gradient, and the purple bar represents coordinate descent.

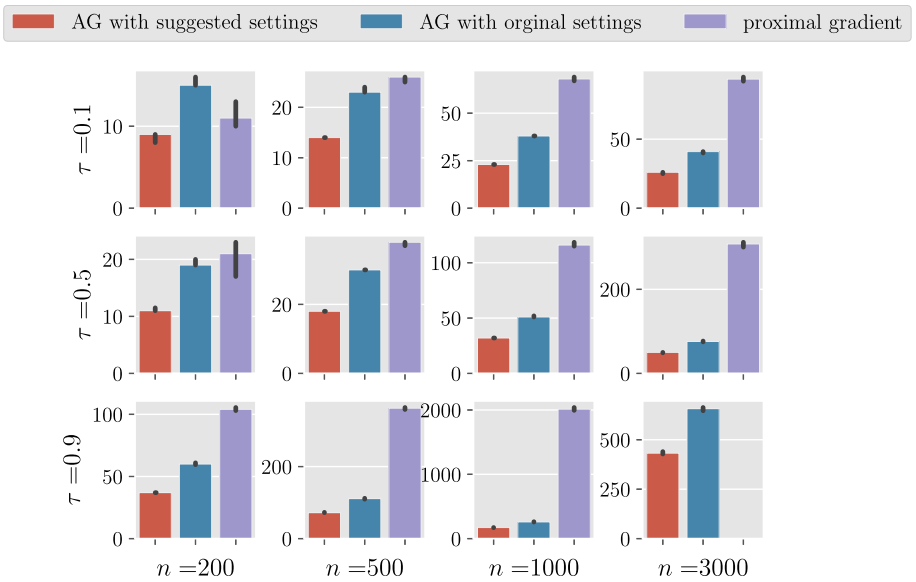

Fig. 8 Median for the number of iterations required for the iterative objective value to reach $g^* + e^3$ on SCAD-penalized linear model for AG with our proposed hyperparameter settings, AG with original settings, and proximal gradient over 100 simulation replications, across varying covariates correlation (τ) and q/n values. The error bars represent the 95% CIs from 1000 bootstrap replications, g^* represents the minimum per iterate found by the three methods considered

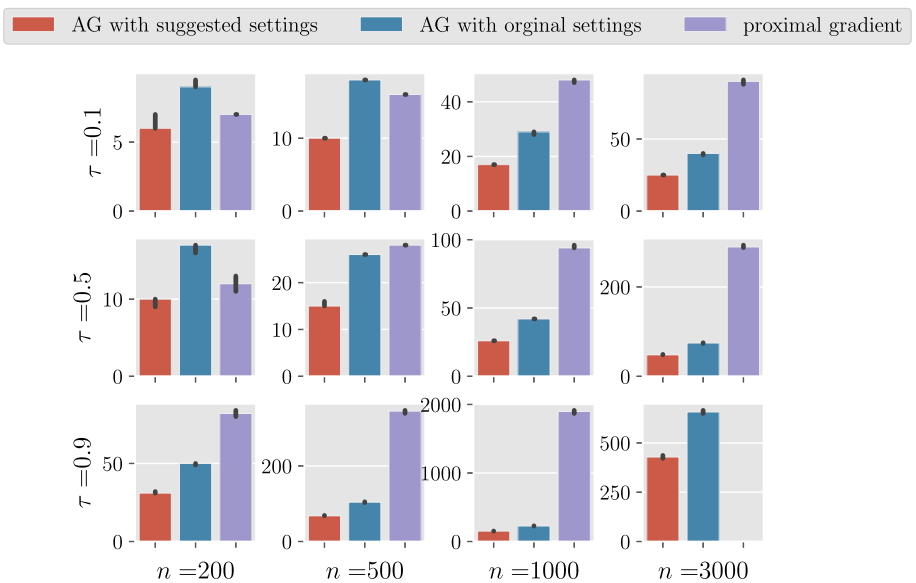

Fig. 9 Median for the number of iterations required for iterative objective values to reach $g^* + e^3$ on MCP-penalized linear model for AG with our proposed hyperparameter settings, AG with original settings, and proximal gradient over 100 simulation replications, across varying covariates correlation (τ) and q/n values. The error bars represent the 95% CIs from 1000 bootstrap replications, g^* represents the minimum per iterate found by the three methods considered

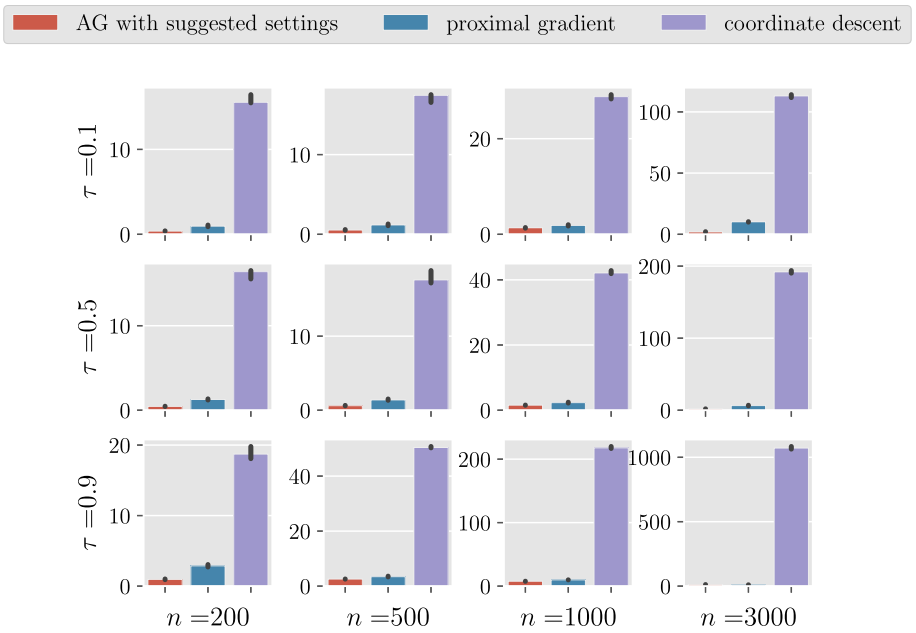

Fig. 10 Median for the computing time (in seconds) required for $\left\|\boldsymbol{\beta}^{(k+1)}-\boldsymbol{\beta}^{(k)}\right\|_{\infty}$ to fall below 10^{-4} on SCAD-penalized linear model for AG with our proposed hyperparameter settings, proximal gradient, and coordinate descent over 100 simulation replications, across varying covariates correlation (τ) and q/n values. The error bars represent the 95% CIs from 1000 bootstrap replications, g^* represents the minimum per iterate found by the three methods considered

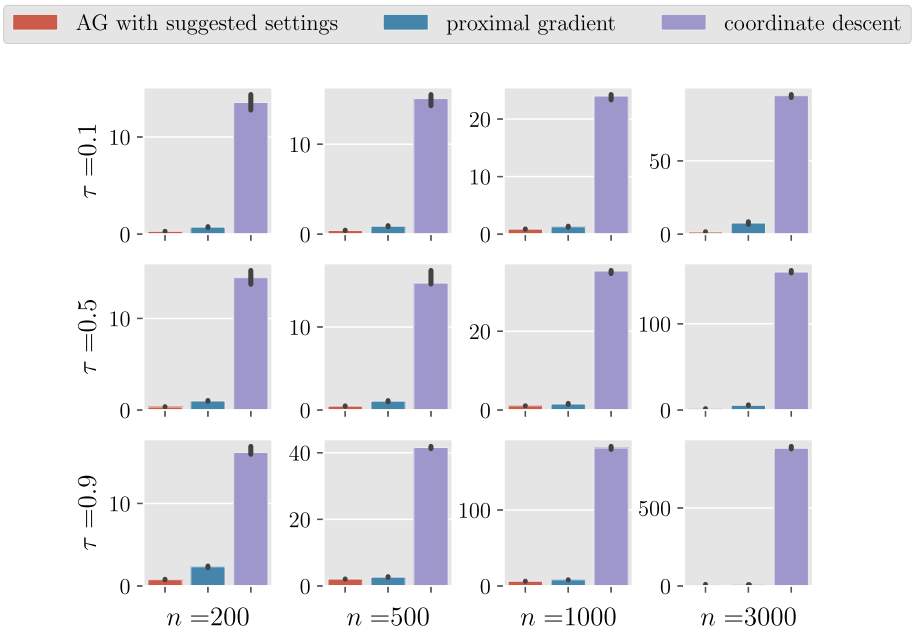

Fig. 11 Median for the computing time (in seconds) required for $\left\|\boldsymbol{\beta}^{(k+1)}-\boldsymbol{\beta}^{(k)}\right\|_{\infty}$ to fall below 10^{-4} on MCP-penalized linear model for AG with our proposed hyperparameter settings, proximal gradient, and coordinate descent over 100 simulation replications, across varying covariates correlation (τ) and q/n values. The error bars represent the 95% CIs from 1000 bootstrap replications, g^* represents the minimum per iterate found by the three methods considered

It is evident that for penalized linear models, AG using our hyperparameter settings outperforms coordinate descent significantly in terms of computing time.

B.2 Penalized logistic regression

Figure (12) and (13) suggest that much less iterations are needed for our method to achieve the same amount of descent in comparison of AG with original proposed settings for penalized logistic models.

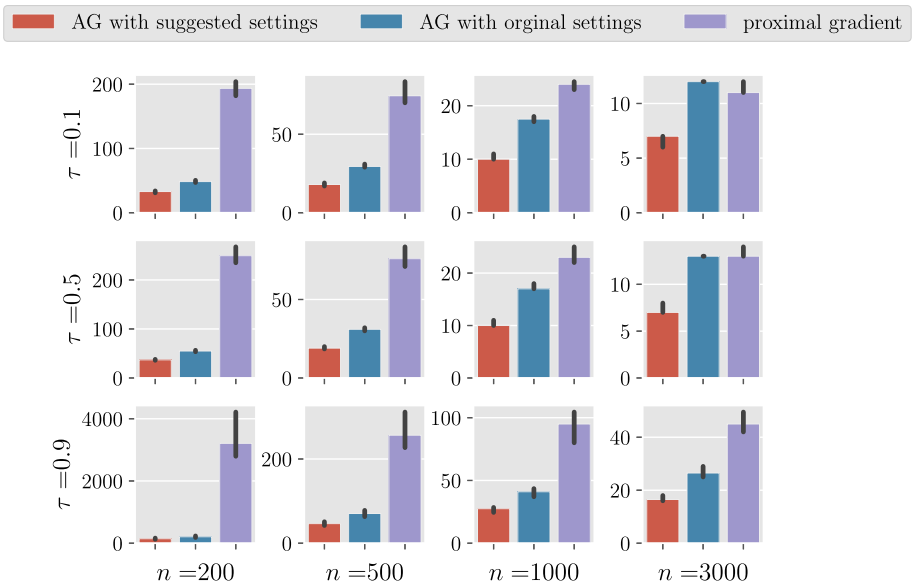

Fig. 12 Median for the number of iterations required for the iterative objective values to reach $g^* + e^2$ on SCAD-penalized logistic regression for AG with our proposed hyperparameter settings, AG with original settings, and proximal gradient over 100 simulation replications, across varying covariates correlation (τ) and q/n values. The error bars represent the 95% CIs from 1000 bootstrap replications, g^* represents the minimum per iterate found by the three methods considered

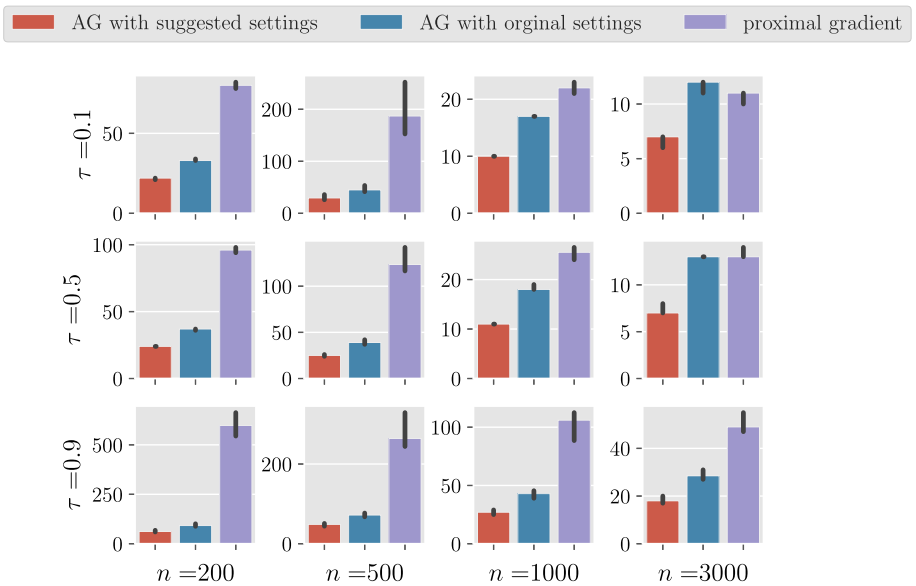

Fig. 13 Median for the number of iterations required for iterative objective values to reach $g^* + e^2$ on MCP-penalized logistic regression for AG with our proposed hyperparameter settings, AG with original settings, and proximal gradient over 100 simulation replications, across varying covariates correlation (τ) and q/n values. The error bars represent the 95% CIs from 1000 bootstrap replications, g^* represents the minimum per iterate found by the three methods considered

References

Akyildiz, Ö.D., Míguez, J.: Convergence rates for optimised adaptive importance samplers. Stat. Comput. (2021). https://doi.org/10.1007/s11222-020-09983-1

Beck, A.: First-Order Methods in Optimization. SIAM-Society for Industrial and Applied Mathematics. Philadelphia, PA, USA (2017). https://doi.org/10.5555/3204879

Breheny, P., Huang, J.: Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. Ann. Appl. Stat. **5**(1), 232–253 (2011). arxiv:1104.2748v1. https://doi.org/10.1214/10-AOAS388

Edelman, A.: Eigenvalues and condition numbers of random matrices. SIAM J. Matrix Anal. Appl. **9**(4), 543–560 (1988). https://doi.org/10.1137/0609045

Fan, J., Li, R.: Variable selection via nonconcave penalized likelihood and its oracle properties. J. Am. Stat. Assoc. **96**(456), 1348–1360 (2001)

Friedman, J., Hastie, T., Höfling, H., Tibshirani, R.: Pathwise coordinate optimization. Ann. Appl. Stat. **1**(2), 302–332 (2007). arxiv:0708.1485. https://doi.org/10.1214/07-AOAS131

Ghadimi, S., Lan, G.: Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Math. Program. **156**(1–2), 59–99 (2015). https://doi.org/10.1007/s10107-015-0871-8

Ghosh, A., Thoresen, M.: Non-concave penalization in linear mixed-effects models and regularized selection of fixed effects. AStA Adv. Stat. Anal. **102**(2), 179–210 (2018). arxiv:1607.02883v1. https://doi.org/10.1007/s10182-017-0298-z

Ibrahim, S.Z., Abdallah, M., N'Guessan, A.: A mixture of local and quadratic approximation variable selection algorithm in nonconcave penalized regression. ARIMA **15**, 18 (2012)

Jin, C., Netrapalli, P., Jordan, M.I.: Accelerated gradient descent escapes saddle points faster than gradient descent (2017) arxiv:1711.10456

Kim, Y., Choi, H., Oh, H.-S.: Smoothly clipped absolute deviation on high dimensions. J. Am. Stat. Assoc. **103**(484), 1665–1673 (2008)

Lan, G.: An optimal method for stochastic composite optimization. Math. Program. **133**(1–2), 365–397 (2011). https://doi.org/10.1007/s10107-010-0434-y

Lee, S., Breheny, P.: Strong rules for nonconvex penalties and their implications for efficient algorithms in high-dimensional regression. J. Comput. Graph. Stat. **24**(4), 1074–1091 (2015). https://doi.org/10.1080/10618600.2014.975231

Lee, S., Kwon, S., Kim, Y.: A modified local quadratic approximation algorithm for penalized optimization problems. Comput. Stat. Data Anal. **94**(C), 275–286 (2016). https://doi.org/10.1016/j.csda.2015.08.019

Mazumder, R., Friedman, J.H., Hastie, T.: Sparsenet: coordinate descent with nonconvex penalties. J. Am. Stat. Assoc. **106**(495), 1125–1138 (2011)

Meckes, E.: The eigenvalues of random matrices, IMAGE, the Bulletin of the International Linear Algebra Society, no. 65, pp. 9–22 (2020). arxiv:2101.02928

Nesterov, Y.E.: A method for solving the convex programming problem with convergence rate $o(1/k^2)$. Dokl. Akad. Nauk SSSR **269**, 543–547 (1983)

Nesterov, Y.: Introductory Lectures on Convex Optimization. Springer, New York (2004). https://doi.org/10.1007/978-1-4419-8853-9

Nesterov, Y.: Gradient methods for minimizing composite functions. Math. Program. **140**(1), 125–161 (2012). https://doi.org/10.1007/s10107-012-0629-5

Paquette, C., van Merriënboer, B., Paquette, E., Pedregosa, F.: Halting time is predictable for large models: a universality property and average-case analysis (2020). arxiv:2006.04299

Parnell, T., Dünner, C., Atasu, K., Sifalakis, M., Pozidis, H.: Tera-scale coordinate descent on GPUs. Futur. Gener. Comput. Syst. **108**, 1173–1191 (2020). https://doi.org/10.1016/j.future.2018.04.072

Quarteroni, A.: Numerical Mathematics. Springer, New York (2000)

Simon, N., Friedman, J., Hastie, T., Tibshirani, R.: A sparse-group lasso. J. Comput. Graph. Stat. **22**(2), 231–245 (2013). https://doi.org/10.1080/10618600.2012.681250

Spall, J.C.: Cyclic seesaw process for optimization and identification. J. Optim. Theory Appl. **154**(1), 187–208 (2012). https://doi.org/10.1007/s10957-012-0001-1

Su, W., Boyd, S., Candès, E.J.: A differential equation for modeling Nesterov's accelerated gradient method: theory and insights. In: Proceedings of the 27th International Conference on Neural Information Processing Systems—Volume 2. NIPS'14, pp. 2510–2518. MIT Press, Cambridge (2014)

Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B (Methodol.) **58**(1), 267–288 (1996)

Wang, L., Kim, Y., Li, R.: Calibrating nonconvex penalized regression in ultra-high dimension. Ann. Stat. **41**(5), 2505–2536 (2013), arxiv:1311.4981v1. https://doi.org/10.1214/13-AOS1159

Yang, Y., Zou, H.: A fast unified algorithm for solving group-lasso penalize learning problems. Stat. Comput. **25**(6), 1129–1141 (2014). https://doi.org/10.1007/s11222-014-9498-5

Yu, D., Won, J.-H., Lee, T., Lim, J., Yoon, S.: High-dimensional fused lasso regression using majorization-minimization and parallel processing. J. Comput. Graph. Stat. **24**(1), 121–153 (2015). https://doi.org/10.1080/10618600.2013.878662

Zhang, C.-H.: Nearly unbiased variable selection under minimax concave penalty. Ann. Stat. **38**(2), 894–942 (2010) arxiv:1002.4734v1. https://doi.org/10.1214/09-AOS729

Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **67**(2), 301–320 (2005)

Zou, H., Li, R.: One-step sparse estimates in nonconcave penalized likelihood models. Ann. Stat. **36**(4), 1509–1533 (2008). arxiv:0808.1012v1