Variable Selection With Second-Generation *P*-Values

Yi Zuo, Thomas G. Stewart & Jeffrey D. Blume

To cite this article: Yi Zuo, Thomas G. Stewart & Jeffrey D. Blume (2021): Variable Selection With Second-Generation *P*-Values, The American Statistician, DOI: 10.1080/00031305.2021.1946150

To link to this article: https://doi.org/10.1080/00031305.2021.1946150

THE AMERICAN STATISTICIAN
2021, VOL. 00, NO. 0, 1–11: General
https://doi.org/10.1080/00031305.2021.1946150

Variable Selection With Second-Generation *P*-Values

Yi Zuo, Thomas G. Stewart, and Jeffrey D. Blume

Department of Biostatistics, Vanderbilt University, Nashville, TN

ABSTRACT
Many statistical methods have been proposed for variable selection in the past century, but few balance inference and prediction tasks well. Here, we report on a novel variable selection approach called penalized regression with second-generation *p*-values (ProSGPV). It captures the true model at the best rate achieved by current standards, is easy to implement in practice, and often yields the smallest parameter estimation error. The idea is to use an ℓ_0 penalization scheme with second-generation *p*-values (SGPV), instead of traditional ones, to determine which variables remain in a model. The approach yields tangible advantages for balancing support recovery, parameter estimation, and prediction tasks. The ProSGPV algorithm can maintain its good performance even when there is strong collinearity among features or when a high-dimensional feature space with $p > n$ is considered. We present extensive simulations and a real-world application comparing the ProSGPV approach with smoothly clipped absolute deviation (SCAD), adaptive lasso (AL), and minimax concave penalty with penalized linear unbiased selection (MC+). While the last three algorithms are among the current standards for variable selection, ProSGPV has superior inference performance and comparable prediction performance in certain scenarios.

ARTICLE HISTORY
Received March 2021
Accepted June 2021

KEYWORDS
Lasso; Penalized regression; Second-generation *p*-values; Variable selection

1. Introduction

Data are typically composed of an outcome and features (predictors or covariates). A common scientific task is to separate important features (signals) from unrelated features (statistical noise) to facilitate modeling, learning, clinical diagnosis, and decision-making. Statistical models are selected for a variety of reasons: predictive ability, interpretability, ability to perform parameter inference, and ease of computation. A model's set of features is called its "support" and the task of recovering the model's true support from observed data is called "support recovery." A desirable variable selection method will tend to return the set of true predictors—that is, those features with truly nonzero coefficients—with high probability. Support recovery aids inference, because knowing the model's true support benefits parameter estimation by reducing bias and improving efficiency. While an incorrectly specified model can sometimes have better predictive performance than a correctly specified model (Shmueli et al. 2010), having the correct support is essential for achieving optimal statistical inference (Zhang et al. 2009; Shortreed and Ertefaie 2017).

Penalized likelihood procedures, originally optimized for prediction tasks, are widely used for variable selection. The lasso, an ℓ_1 penalization method, produces models with strong predictive ability (Tibshirani 1996). However, the lasso solution that maximizes predictive ability does not always lead to consistent support recovery (Leng, Lin, and Wahba 2006; Meinshausen et al. 2006; Shmueli et al. 2010; Bogdan et al. 2015). This is because noise variables are often included in the lasso solution that maximizes predictive ability (Meinshausen et al. 2006). The adaptive lasso (AL), which introduces weights in the ℓ_1 penalty, was proposed to resolve the issue that lasso solutions can be variable selection inconsistent (Zou 2006). With clever choice of tuning parameters, and in large samples, the AL can recover the true support with high probability and yield parameter estimates that converge properly (Zou 2006). Smoothly clipped absolute deviation (SCAD) (Fan and Li 2001) and minimax concave penalty (MCP) with penalized linear unbiased selection (MC+) (Zhang et al. 2010) make use of distinctive piecewise linear thresholding functions to bridge the gap between the ℓ_0 and ℓ_1 algorithms. Both SCAD and MC+ seek to preserve large coefficients, like the ℓ_0 penalty does, and shrink small coefficients, like the ℓ_1 penalty does. While their variable selection properties have been well established, these methods are still not widely used in routine practice.

All of the above approaches place a strong emphasis on predictive ability, at the cost of subsequent inference tasks. Because inference is an essential component of scientific investigations, a variable selection approach that balances prediction and inference tasks is highly desirable. Since traditional *p*-values do not reflect whether a variable is scientifically relevant or not (Heinze, Wallisch, and Dunkler 2018), we investigated whether using second-generation *p*-values (SGPV) (Blume et al. 2018, 2019) would lead to good support recovery and subsequent parameter estimation and prediction. SGPVs emphasize scientific

CONTACT Yi Zuo yi.zuo@vanderbilt.edu Department of Biostatistics, Vanderbilt University, Nashville, TN 37235.
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/TAS.

relevance in addition to statistical significance, and thus they are a good tool for screening out noise features and identifying the true signals in a set of candidate variables.

Following this idea, we propose a variable selection algorithm based on an ℓ_0-penalized regression with SGPVs (ProSGPV). The ProSGPV algorithm has a high support recovery rate and low parameter estimation bias, while maintaining good prediction performance even in the high-dimensional setting where $p > n$. In a series of comprehensive simulations and a real-world application, the ProSGPV algorithm is shown to be a viable alternative to, and often a noticeable improvement on, current variable selection standards such as AL, SCAD, and MC+. While only linear models are discussed in this article, forthcoming work will show that the ProSGPV approach generalizes to models of other classes, including logistic regression, Poisson regression, Cox proportional hazards model, etc.

The structure of this article is as follows. Section 2 provides a brief background. Section 3 describes the proposed ProSGPV algorithm. Section 4 presents simulation studies comparing ProSGPV to AL, SCAD, and MC+ under various feature correlation structures and signal-to-noise ratios (SNRs). Section 5 illustrates the ProSGPV algorithm using a real-world data application. Section 6 discusses the practical implications of the simulation results and some limitations of ProSGPV, and summarizes key findings in the article.

2. Background Material

We review some fundamental ideas related to shrinkage, thresholding, inference, and prediction in the variable selection context to facilitate subsequent discussions about ProSGPV. Readers familiar with standard variable selection notation, lasso (Section 2.1), AL (Section 2.2), SCAD and MC+ (Section 2.3), and SGPV (Section 2.4) may skip to Section 3 for the development of the ProSGPV algorithm.

2.1. Lasso

The lasso is an ℓ_1 penalization procedure and one of the most widely used regularization methods for prediction modeling (Tibshirani 1996). It reduces the feature space and identifies a subset of features that maximize predictive accuracy subject to a sparsity condition induced by the ℓ_1 penalty. The set of features selected by lasso is called the active set.

Let $\boldsymbol{Y} = (Y_1, Y_2, \ldots Y_n)$ denote the response vector, $\boldsymbol{X}$ denote the $n \times p$ design matrix, and $\boldsymbol{\beta} \in \mathbb{R}^p$ denote the coefficient vector. $\lambda > 0$ is a regularization parameter. $||\cdot||_2^2$ is the squared ℓ_2-norm and $||\cdot||_1$ is the ℓ_1-norm. Formally, the lasso solution is written as

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \{\frac{1}{2}||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1\}. \qquad (1)$$

The lasso is often used for variable selection because its solution encourages sparsity in the active set. However, even in the classical setting of a fixed p and a growing n, the lasso active set tends to be different from the set of true signals. An exception to this is when true feature columns are roughly orthogonal to noise feature columns (Knight and Fu 2000), which unfortunately, is seldom seen in practice. Wainwright (2009b) improved the ability of the lasso solution to recover the true support under random Gaussian designs and showed that lasso can recover the true support when the effect size is sufficiently large and when no noise variables are highly correlated with true features. However, these conditions are strong and hard to apply in practice. In addition, even when they are met, there is no explicit way to implement the procedure because the shrinkage factor λ that yields the correct support recovery is unknown (Wang, Kim, and Li 2013). Last, the soft thresholding function in lasso shrinks large effects and results in biased parameter estimates that are ideal for prediction tasks, but not necessarily optimal for inference tasks.

2.2. Adaptive Lasso

The AL uses weights in the ℓ_1 penalty to address the inconsistent variable selection property of the lasso (Zou 2006). With the right shrinkage parameter, initial weights, and weight moments, the AL can recover the true support with high probability while preserving prediction performance. Formally, the solution to the AL is

$$\hat{\boldsymbol{\beta}}^n = \arg\min_{\boldsymbol{\beta}} \{\frac{1}{2}||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda_n||\hat{\boldsymbol{\omega}}\boldsymbol{\beta}||_1\}, \qquad (2)$$

where $\hat{\boldsymbol{\omega}} = 1/|\hat{\boldsymbol{\beta}}^*|^\gamma$. Here $\gamma > 0$ is a tuning parameter and $\hat{\boldsymbol{\beta}}^*$ is any root-n-consistent estimator of the parameter $\boldsymbol{\beta}$, for example, an OLS estimator, or a lasso estimator.

Zou (2006) showed that AL has large-sample oracle (optimal) properties for support recovery and parameter estimation as $\lambda_n/\sqrt{n} \to 0$ and $\lambda_n n^{(\gamma-1)/2} \to \infty$. However, when the sample size is finite, it can be hard to find a combination of $\hat{\boldsymbol{\beta}}^*$, γ, and λ_n such that the resulting active set matches the true support and the estimated coefficients have low bias.

2.3. SCAD and MC+

SCAD and MC+ were designed to bridge ℓ_0 and ℓ_1 penalization schemes. As a result, both algorithms use nonconvex penalties. There are considerable advantages that come with using nonconvex penalization, such as a sparse solution and reduced parameter estimation bias, see Fan and Lv (2011), Fan and Lv (2013), Zheng, Fan, and Lv (2014), and Loh and Wainwright (2015).

The penalty function in the SCAD corresponds to a quadratic spline function with knots at λ and $\gamma\lambda$ (Fan and Li 2001). With proper choice of regularization parameters, SCAD can yield consistent variable selection in large samples (Fan and Li 2001). MC+ has two components: an MCP and a penalized linear unbiased selection (PLUS) algorithm (Zhang et al. 2010). MC+ returns a continuous piecewise linear path for each coefficient as the penalty increases from zero (least squares) to infinity (null model). When the penalty level is set to $\lambda = \sigma\sqrt{(2/n)\log(p)}$, the MC+ algorithm has a high probability of support recovery and does not need to assume the strong irrepresentable condition (Wainwright 2009b) that is required by lasso for support recovery (Zhang et al. 2010). For visualization, Figure 1 displays the thresholding functions of ℓ_0 and ℓ_1 penalties, SCAD, and MC+ when the feature columns are orthogonal.

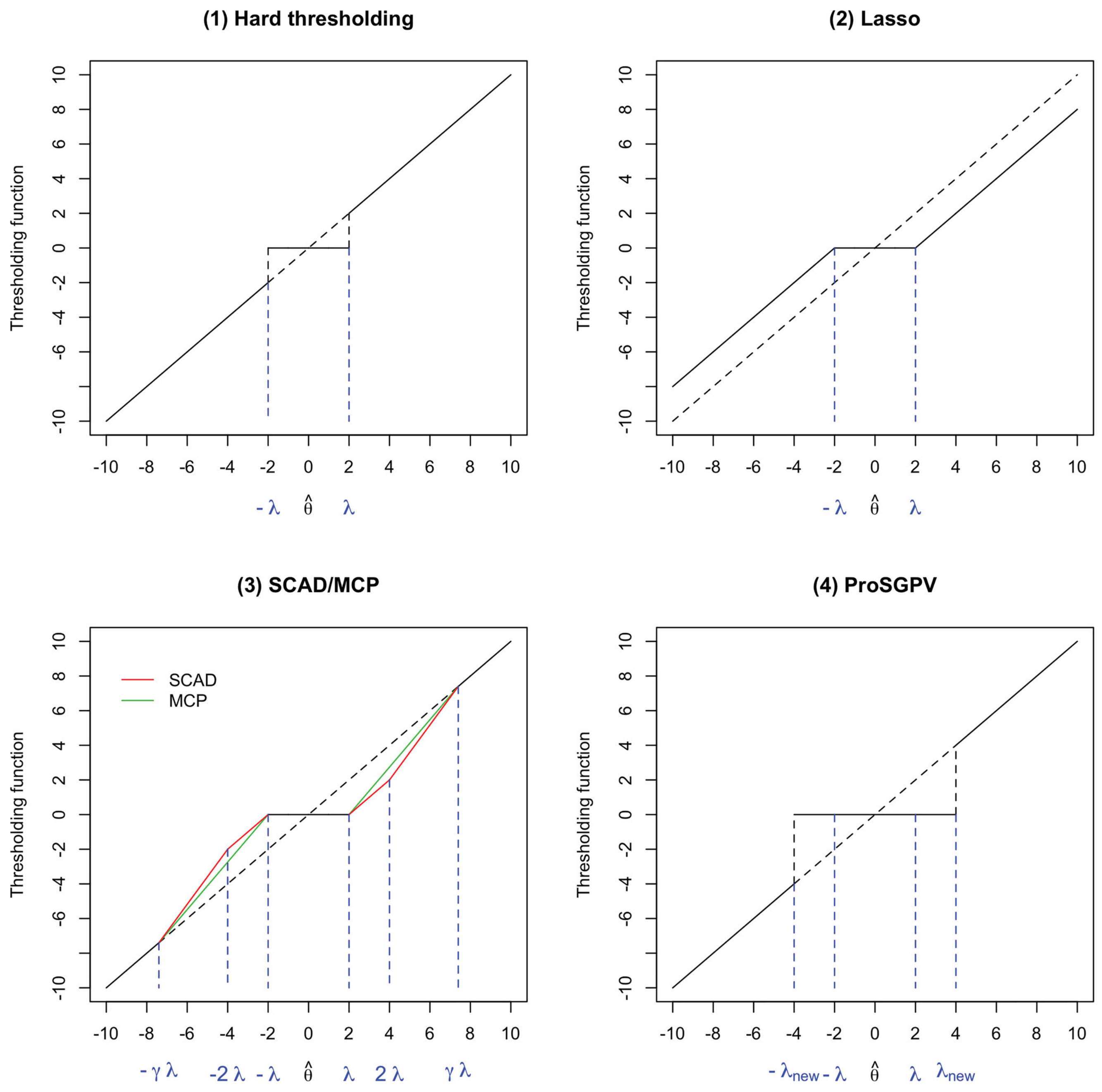

Figure 1. Thresholding functions from five algorithms when features are orthogonal.

2.4. Second-Generation p-Values

Second-generation p-values (SGPV), denoted as p_δ, were proposed for use in high-dimensional multiple testing contexts (Blume et al. 2018, 2019). SGPVs attempt to resolve some of the deficiencies of traditional p-values by replacing the point null hypothesis with a prespecified interval null $H_0 = [-\delta, \delta]$. The idea is to use the interval as a buffer region between "null" and "nonnull" effects. The interval represents the set of effects that are scientifically indistinguishable or immeasurable from the point null due to limited precision or practicality. SGPVs are essentially the fraction of data-supported hypotheses that are null, or nearly null, hypotheses.

Formally, let θ be a parameter of interest, and let $I = [\theta_l, \theta_u]$ be an interval estimate of θ whose length is given by $|I| = \theta_u - \theta_l$. In this article, we will use a 95% CI for I, but any type of the uncertainty interval can be used. If we denote the length of the interval null by $|H_0|$, then the SGPV p_δ is defined as

$$p_\delta = \frac{|I \cap H_0|}{|I|} \times \max\left\{\frac{|I|}{2|H_0|}, 1\right\}, \tag{3}$$

where $I \cap H_0$ is the intersection of two intervals. The correction term $\max\{|I|/(2|H_0|), 1\}$ applies when the interval estimate is very wide, that is, when $|I| > 2|H_0|$. In that case, the data are often inconclusive and the correction term shrinks the SGPV back to 1/2. As such, SGPVs indicate when data are compatible with null hypotheses ($p_\delta = 1$), or with alternative hypotheses ($p_\delta = 0$), or when data are inconclusive ($0 < p_\delta < 1$).

By design, SGPVs emphasize effects that are scientifically meaningful as defined by exceeding a prespecified effect size

δ. Empirical studies have shown that SGPVs have the potential for identifying feature importance in high-dimensional settings (Blume et al. 2018, 2019). This idea dovetails well with the natural tendency in variable selection to keep variables whose effects are above some threshold, say δ. One extension here is that we will let the null bound δ shrink to zero at a prespecified rate. This slight modification of the basic SGPV idea makes variable selection by SPGVs much more effective.

3. The ProSGPV Algorithm

The ProSGPV algorithm is a two-stage algorithm. In the first stage, a candidate set of variables is acquired. In the second stage, an SGPV-based thresholding is applied to select variables from the candidate set that are meaningfully associated with the outcome.

3.1. Steps

The steps of the ProSGPV algorithm are shown in Algorithm 1.

Algorithm 1 ProSGPV

```
 1: procedure ProSGPV($\boldsymbol{X}$, $\boldsymbol{Y}$)
 2:     Stage one: Find a candidate set
 3:         Standardize all inputs (the outcome and features)
 4:         Fit a lasso and find $\lambda_{\text{gic}}$ using generalized information criterion
 5:         Fit an OLS model on the lasso active set
 6:     Stage two: SGPV screening
 7:         Extract the confidence intervals of all variables from the previous OLS model
 8:         Calculate the mean coefficient standard error $\overline{\text{SE}}$ of standardized features
 9:         Get the SGPV for each variable $k$ with $I_k = \hat{\beta}_k \pm 1.96 \times \text{SE}_k$ and $H_0 = [-\overline{\text{SE}}, \overline{\text{SE}}]$
10:         Keep variables with SGPV of zero
11:         Re-run the OLS model with selected variables on the original scale
12: end procedure
```

Note that the outcome and features are standardized except for the final step. Generalized information criterion (GIC) (Fan and Tang 2013) is used to find the shrinkage parameter λ_{gic} that leads to a fully relaxed lasso (Meinshausen 2007) in the first stage. λ could also be found through cross-validation, as there is evidence that a range of λs will lead to the true support (Fan and Li 2001; Zou 2006; Wang, Kim, and Li 2013; Sun et al. 2019). That adds to the flexibility of the algorithm. In the second stage, SGPVs are used to screen variables in the candidate set, where the null bound δ is derived from coefficient standard errors. Sensitivity to the choice of the null bound δ is assessed in Section 3.3. We have implemented the ProSGPV algorithm in the *ProSGPV* R package, which is available from the Comprehensive R Archive Network (CRAN) at *https://CRAN.R-project.org/package=ProSGPV*.

3.2. Solution

The solution to the ProSGPV algorithm $\hat{\boldsymbol{\beta}}^{\text{pro}}$ is

$$\hat{\boldsymbol{\beta}}^{\text{pro}} = \hat{\boldsymbol{\beta}}^{\text{ols}}_{|S} \in \mathbb{R}^p, \text{ where}$$
$$S = \{k \in C : |\hat{\beta}^{\text{ols}}_k| > \lambda_k\}, C = \{j \in \{1, 2, \dots, p\} : |\hat{\beta}^{\text{lasso}}_j| > 0\} \quad (4)$$

where $\hat{\boldsymbol{\beta}}^{\text{ols}}_{|S}$ is a vector of length p with nonzero elements being the OLS coefficient estimates from the model with variables only in the set S, the final selection set. C is the candidate set from the first-stage screening. $\hat{\beta}^{\text{lasso}}_j$ is the jth lasso solution evaluated at λ_{gic} in the first stage. In the second stage, the cutoff is $\lambda_k = 1.96 \times \text{SE}_k + \overline{\text{SE}}$. λ_k is constant over k when the features are orthogonal. In that case, the coefficient standard errors are identical with centered and standardized feature columns.

The ProSGPV algorithm is effectively a hard thresholding function. In the first stage, variables not selected by lasso are shrunk to zero. In the second stage, ProSGPV relaxes the coefficients and shrinks effects smaller than $\overline{\text{SE}}$ to zero while preserving large effects. Because this is a two-stage algorithm, there does not appear to be a simple closed-form solution for the implied thresholding without conditioning on the first stage. However, this would-be threshold, call it λ_{new}, tends to be larger than λ_{gic} from lasso. The only routine exception to this is when data have weak signals or high correlation. But in that case, no algorithm can fully recover the true support (Zhao and Yu 2006; Wainwright 2009a).

A visualization of thresholding functions for several penalization methods (assuming orthogonal features) is displayed in Figure 1. The hard thresholding function in the panel (1) shrinks the coefficient estimates to zero when the effects are less than λ and preserves them otherwise. The lasso in (2) shrinks small effects to zero and shrinks large effects by λ. SCAD and MCP in (3) bridge the gap between a hard thresholding function seen in (1) and a soft thresholding function in (2). When the coefficient is small ($|\hat{\theta}| \leq \lambda$), both methods have the same behavior as the lasso because the coefficient is shrunk to zero in all cases. When the coefficient is large ($|\hat{\theta}| \geq \gamma\lambda$), SCAD and MCP have the same behavior as the hard thresholding (no shrinkage is applied). What distinguishes SCAD and MCP is the shape of its thresholding function between λ and $\gamma\lambda$. As mentioned earlier, ProSGPV amounts to a hard thresholding function whose cutoff is usually larger than λ.

3.3. Null Bound

The null bound in ProSGPV is set to be the average coefficient standard error, say $\overline{\text{SE}}$, from the OLS model on the lasso candidate set. Because of the scaling, this is equivalent to hard-thresholding variables whose absolute coefficients are below $1.96 \times \text{SE}_k + \overline{\text{SE}} \approx 3 \times \overline{\text{SE}}$. This is in line with variable selection ideas from literature. Fan and Li (2006) argued that in order to achieve optimal properties of variable selection, the amount of lasso shrinkage must be proportional to the standard error of the maximum likelihood estimates of coefficients. Intuitively, the interval null acts as a buffer zone to screen out effects that are likely false discoveries. By definition, the sampling distribution of false discoveries will be near the point null (since they are "false" discoveries) and the variance of this distribution shrinks at a rate proportional to the information in the sample. Hence, using the SE to delineate the smallest effect size of interest is natural. It is possible that a constant multiplier of the SE might yield a better Type I–Type II error tradeoff, but after trying some obvious variations we did not find anything better.

A sensitivity analysis on the choice of the null bound was conducted and is summarized in Figure 1 (supplementary material). We compared the support recovery performance of ProSGPV using different null bounds when SNR is medium or high and when $n > p$. Choices of null bounds include the original bound $\overline{\text{SE}}$, $\overline{\text{SE}} \times \sqrt{\log(n/p)}$, $\overline{\text{SE}}/\sqrt{\log(n/p)}$, $\hat{\sigma}/12$, and 0. When the null bound is constant, for example, $\hat{\sigma}/12$, the support recovery performance is poor. When the null bound is scaled by $\sqrt{\log(n/p)}$, performance appears to be slightly improved in the high correlation case, but, importantly, is inferior in all other cases. When the null bound is set at 0, ProSGPV amounts to selecting variables using traditional p-values. In this case, the support recovery performance is expectedly poor even when SNR is high, because the null bound of 0 leads to many false positives (Kaufman and Rosset 2014; Janson, Fithian, and Hastie 2015). When $p > n$, the above observations hold because the null bound is calculated from a model with a reduced number of features (same order as $s << p$, where s is the number of true signals). This sparsity assumption is necessary for successful high-dimensional support recovery (Meinshausen et al. 2006; Zhao and Yu 2006; Wainwright 2009a). Hence, allowing the null bound, which acts as a thresholding function, to shrink at a $\sqrt{n}$-rate, appears to offer the best performance across the widest range of scenarios.

3.4. Example

Figure 2 shows the effect of the ProSGPV algorithm on the regression coefficients in our simulated setting. Suppose that the true data-generating model is $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}$ where $\boldsymbol{y}$ is a vector of length 400. The design matrix $\boldsymbol{X}$ has five columns with mean zero and covariance matrix $\Sigma_{i,j} = 0.5^{|i-j|}$. The coefficient vector $\boldsymbol{\beta}$ is zero everywhere except $\beta_3 = 0.28$. The errors are iid $N(0, 1)$. We see in Figure 2 that the ProSGPV algorithm succeeds by selecting V3, whereas the lasso and relaxed lasso select V3 and V5 at λ_{gic}.

3.5. Similar Algorithms From the Literature

Other two-stage algorithms have been proposed for prescreening features (Meinshausen et al. 2009; Zhang et al. 2009; Wasserman and Roeder 2009; Zhou 2009, 2010; Sun et al. 2019; Weng, Feng, and Qiao 2019; Wang et al. 2020). Meinshausen et al. (2009) proposed a two-stage thresholded lasso, where a lasso model is fit and features are kept if they pass a data-dependent coefficient threshold. Because of this, the resulting coefficient estimates are biased even when the correct support is recovered. Wasserman and Roeder (2009) proposed using variable selection methods (lasso, marginal regression, forward stepwise regression, etc.) with cross-validation to prescreen candidate variables before using Bonferroni corrected t-tests to identify and remove noise features. Wasserman's method controls the Type I error rate across all features, but pays a higher price in false negatives. ProSGPV, however, allows the Type I error rate to shrink toward zero and yields fewer false positives (see supplementary material, Figure 3). Zhang et al. (2009) identified relevant and irrelevant features from lasso in the first stage and fit another ℓ_1-penalized regression using only irrelevant features afterwards. However, Zhang et al. (2009) emphasized parameter estimation and neglects support recovery. In addition, their algorithm needs to run multiple cross-validations, while ProSGPV uses GIC to tune λ and is therefore much faster to compute. Sun et al. (2019) proposed the hard thresholding regression (HRS). When lasso is used to derive initial weights, the HRS reduces to the fully relaxed lasso, which is the first stage of our two-stage ProSGPV algorithm. Unlike our algorithm, HRS keeps all variables that survive the first stage. Last, Zhou (2009, 2010) used lasso or the Dantzig selector to pre-screen and then used a fully relaxed model on thresholded coefficients with a data-driven bound; Weng, Feng, and Qiao (2019) selected important variables and penalized only the unselected variables for the final variable selection; Wang et al. (2020) used a bridge regression in the first stage and thresholded variables in the second stage.

3.6. Special Case: One-Stage ProSGPV Algorithm

When λ_{gic} is replaced with zero in the first stage of lasso, ProSGPV reduces to a one-stage algorithm. That amounts to calculating the SGPV for each variable in the full OLS model and selecting ones that are above the threshold. The one-stage ProSGPV is faster to compute, as no lasso solution path is required. However, it does not appear to be variable selection consistent in the limit, and its inferential performance is inferior to that of the two-stage ProSGPV when data do not contain strong signals or features are highly correlated. Moreover, it is not applicable when $p > n$, that is, when the OLS model is not identifiable. For completeness, the support recovery performance of the one-stage algorithm can be found in Figure 2 (supplementary material). Its performance is very close to the two-stage algorithm when explanatory variables are independent.

3.7. Summary

The ideas behind the ProSGPV algorithm are intuitive: exclude small effects using a data-dependent threshold for noise and keep large effects. ProSGPV is essentially an ℓ_0-penalized regression. Unlike the ℓ_1 penalty, ℓ_0 optimization is nonconvex, so it is harder to compute and less popular in practice. However, our algorithm avoids enumerating all possible combinations of variables by leveraging the lasso solution in the first stage and threshold effects with an explicit bound afterwards. That translates into less computational cost than other convex optimization algorithms (as seen in the supplementary material, Figure 6). ProSGPV can also be thought of as a variation of the thresholded lasso with refitting. van de Geer et al. (2011) showed that the thresholded lasso with refitting requires less severe minimal signal conditions for successful support recovery than AL. While lasso is used in the first stage screening, other variable selection methods, such as Sure Independence Screening (SIS) (Fan and Lv 2008), can be used there. This adds the/flexibility

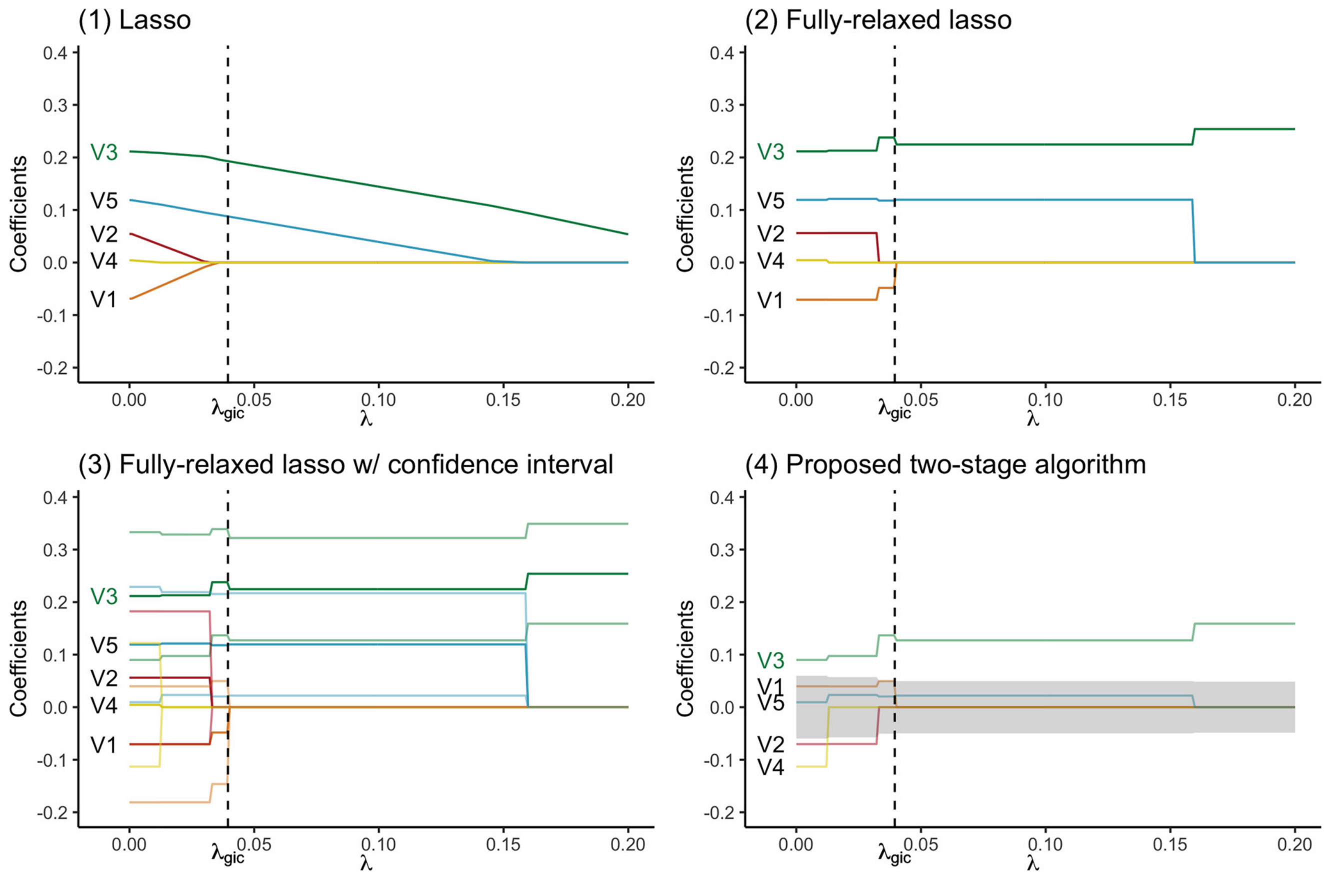

Figure 2. Illustration of the ProSGPV algorithm. Panel (1) presents the colored lasso solution path where the vertical dotted line is the λ_{gic}. Panel (2) shows the fully-relaxed lasso path with point estimates only. Panel (3) shows the same path plus 95% confidence intervals in light colors. Panel (4) is the proposed two-stage algorithm's selection path. The shaded area is the null region and only the 95% confidence bound that is closer to zero is shown for each variable.

to our algorithm. Last, in terms of post-selection inference, the point estimates and corresponding confidence intervals derived from our algorithm are best when the selected model matches the true underlying model. Even when ProSGPV misses true signals, those missed variables often have small effects, which results in minimal impact on the inference of the other larger effects.

4. Simulation Studies

Extensive simulation studies were conducted to evaluate the inferential and prediction performance of the ProSGPV algorithm and compare it to existing methods. We investigated both traditional $n > p$ and high-dimensional $p > n$ settings.

4.1. Design

The simulation setup is motivated by similar investigations such as Hastie et al. (2020). We set sample size n, dimension of explanatory variables p, sparsity level s (number of true signals), true coefficient vector $\boldsymbol{\beta}_0 \in \mathbb{R}^p$, autocorrelation level ρ within explanatory variables, and SNR ν.

In the traditional $n > p$ setting, p is fixed at 50 and n ranges from 100 to 2000 with an increment of 50. The number of true signals s is fixed at 10. In the high-dimensional setting, n is fixed at 200 and p ranges from 200 to 2000 with an increment of 20. Here, the number of true signals is fixed at 4. $\boldsymbol{\beta}_0$ has s nonzero values equally spaced between one and five, at random positions, and the rest are zero. The coefficients are half positive and half negative. ρ can take the value of 0 (independent), 0.35 (medium autocorrelation), and 0.7 (high autocorrelation). SNR is defined as SNR $=$ var$(f(x))/$var(ϵ), where data are generated from a probabilistic distribution. SNR take the value of 0.7 (moderate SNR), and 2 (high SNR) (Hastie et al. 2020).

We evaluated the performance of each algorithm using standard metrics: support recovery rate, Type I error rate, power, false discovery rate, false nondiscovery rate, along with the mean absolute error (MAE defined below) for parameter estimation, prediction accuracy in a separate test set, and running time. See supplementary material (Table 1) for detailed definitions of the metrics for inference.

Step 1: Draw n rows of the matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ iid from $N_p(0, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$ has entry (i, j) equal to $\rho^{|i-j|}$.

Step 2: Generate the response vector $\boldsymbol{Y} \in \mathbb{R}^n$ from $N_n(\boldsymbol{X}\boldsymbol{\beta}_0, \sigma^2\boldsymbol{I})$, with σ^2 defined to meet the desired SNR level ν, that is, $\sigma^2 = \boldsymbol{\beta}_0^T\boldsymbol{\Sigma}\boldsymbol{\beta}_0/\nu$.

Step 3: Run SCAD, MC+, AL, and ProSGPV on the training set with n observations; record the active set from each algorithm; compute evaluation metrics in Table 1 (supplementary material) plus capture rate of the exact true model, absolute bias in parameter estimation, and running time; use a separate test set to compute prediction accuracy. Note that the test set was generated in Step 1, and set aside for later use by inflating the target sample size n.

Step 4: Repeat the previous steps 1000 times and aggregate the results.

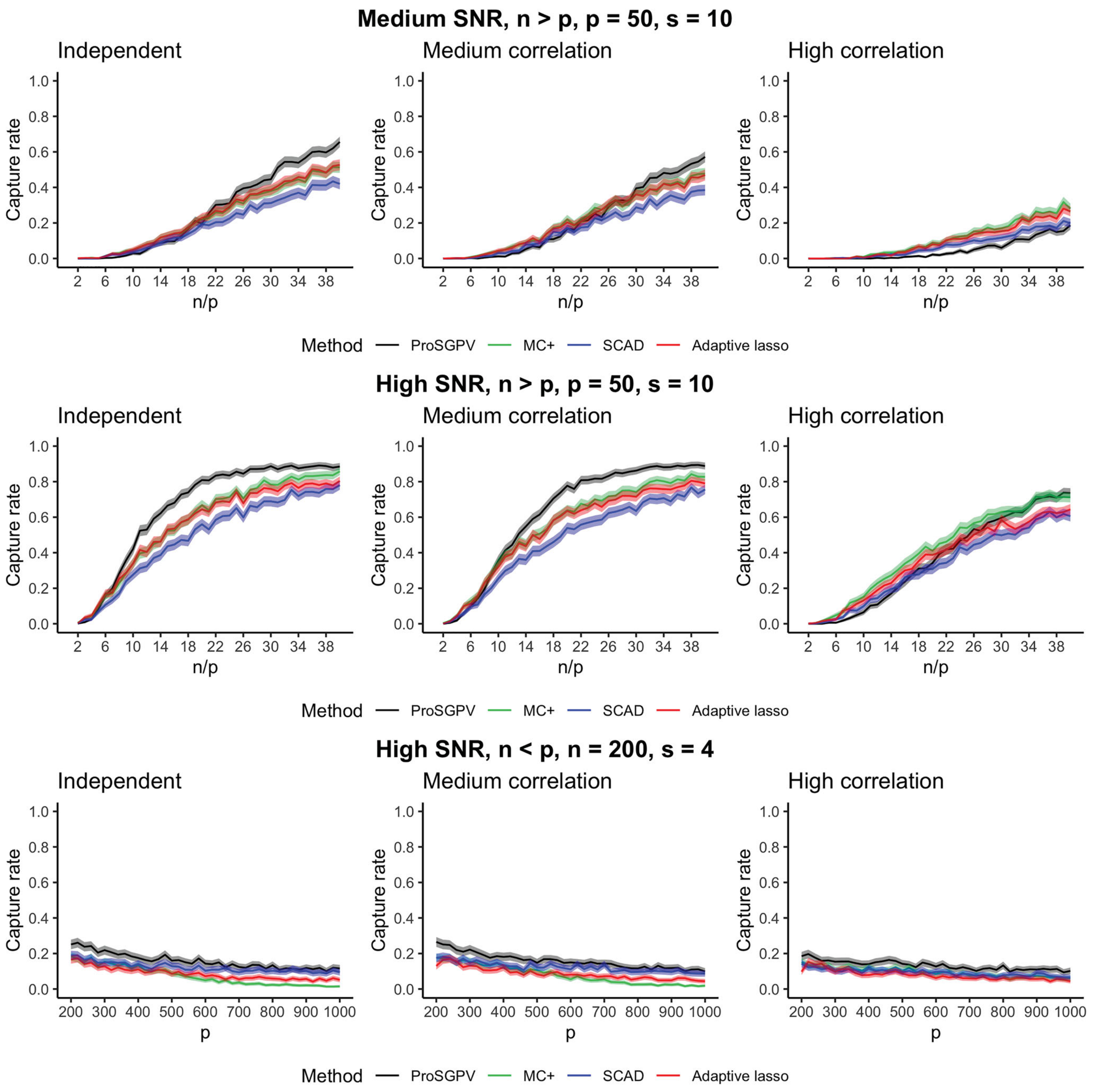

Figure 3. Capture rate of the exact true model under combinations of autocorrelation level, signal-noise-ratios, and (n, p, s). In each panel, one algorithm has a colored solid line representing the average capture rate surrounded by the shaded 95% Wald interval over 1000 simulations.

SCAD was implemented using the *ncvreg* package in R and γ was fixed at 3.7, MC+ was implemented using the *plus* package. AL was implemented using the *glmnet* package and the initial weights are the inverse of absolute value of lasso estimates. For a fair comparison, GIC was used to select λ in all algorithms. The ProSGPV algorithm was implemented using the *ProSGPV* package. The R code to replicate simulation results can be found at *https://github.com/zuoyi93/r-code-prosgpv-linear*.

4.2. Results and Findings

We recorded whether or not each algorithm captured the exact true model in each iteration and compared the average capture rates over 1000 iterations in Figure 3. We also compared the MAE of all coefficient estimates, defined as $\frac{1}{p}\sum_{j=1}^{p}|\hat{\beta}_j - \beta_{0,j}|$, in Figure 4, where $\beta_{0,j}$ is the jth true coefficient. We compared the prediction accuracy of each algorithm, as measured by root mean square error (RMSE) in an independent test set in Figure 5. Power and Type I error rates are presented in Figure 3 (supplementary material). False discovery proportions (pFDR) and false non-discovery proportions (pFNR) are presented in Figure 4 (supplementary material). The effect of different parameter tuning methods on MC+ is illustrated in Figure 5 (supplementary material). The comparison of computation time is shown in Figure 6 (supplementary material).

In Figure 3, capture rates of the exact true model are compared under combinations of SNR and autocorrelation levels within the design matrix, when both $n > p$ and $n < p$.

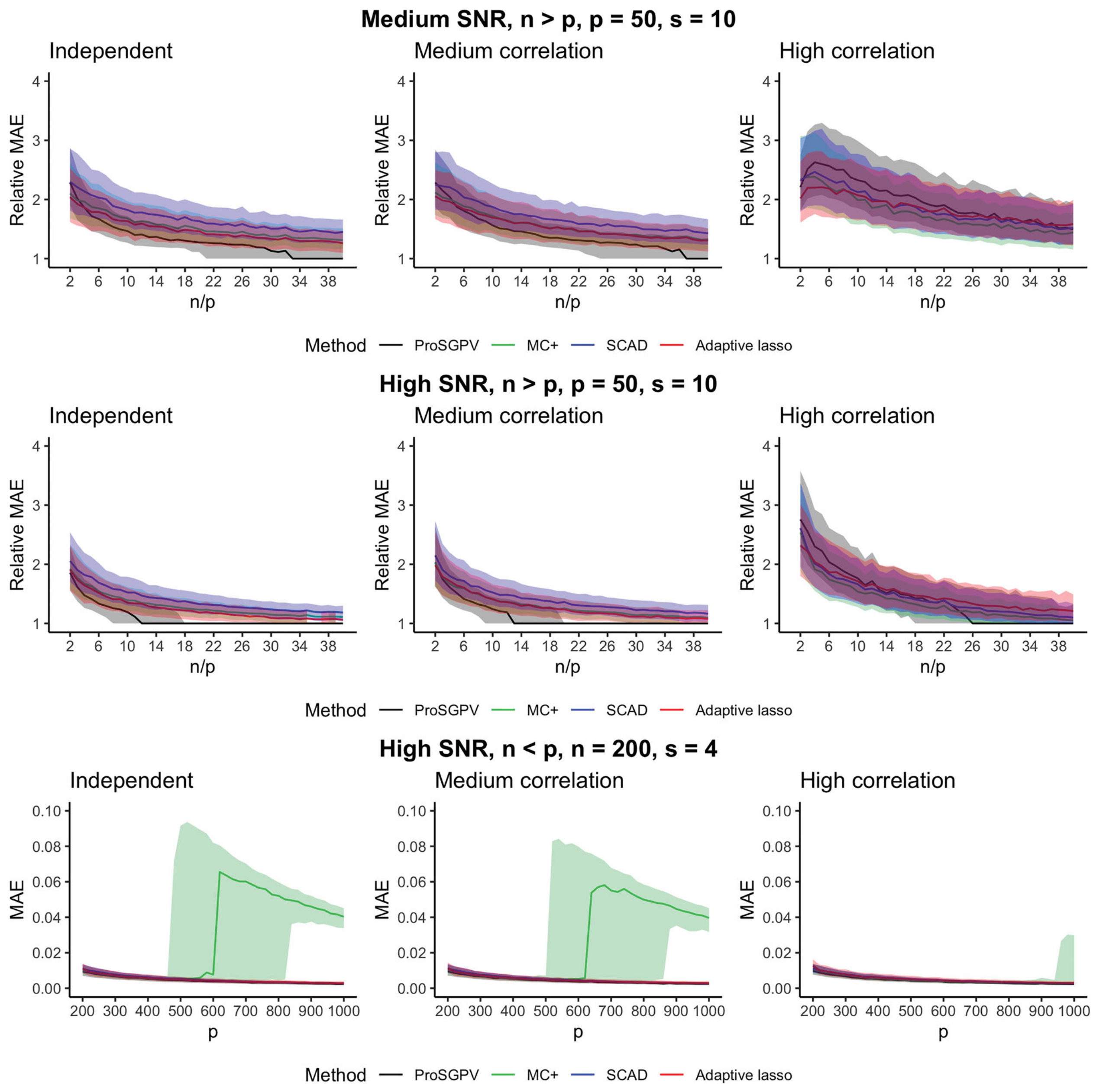

Figure 4. Parameter estimation error of all algorithms under combinations of autocorrelation level, SNR, and (n, p, s). In each panel, one algorithm has a colored solid line representing the median (relative) MAEs surrounded by the shaded first and third quartiles over 1000 simulations.

When $n > p$, ProSGPV's support recovery rate increases as n grows. It generally has the highest support recovery rate except when the SNR is medium and correlation is high. MC+ and AL have similar capture rates, while SCAD is the worst among the four. When $p > n$, support recovery rates are low for all methods and decrease as p increases in the data. ProSGPV again is the highest, followed by SCAD, AL, and MC+. We investigated factors driving the support recovery performance in Figures 3 and 4 (supplementary material). When $n > p$, we see that all algorithms have decreasing Type I error rates, pFDR, pFNR, and increasing power. When $p > n$ and data are not highly correlated, GIC-based MC+ has notably higher pFDR than the others, indicating that it overfits the training data and includes many noise variables.

Mean absolute error (MAE) is used to assess the parameter estimation error. When $n > p$, we used relative MAE which is defined as the ratio of an algorithm's MAE to that of the OLS model with only true features. A good estimator would have an asymptotic relative MAE of one. When $p > n$, absolute MAE is used because no OLS fit is possible. Figure 4 displays the median (relative) MAE of four algorithms under various scenarios. The shading shows the first and third quartiles of the empirical (relative) MAE distribution.

In both $n > p$ and $n < p$ cases, ProSGPV has the lowest parameter estimation error. This should not be surprising for sparse settings with well-defined signals, as ProSGPV is effectively an ℓ_0 penalization derivative and ℓ_0 penalization drops small effects while keeping large ones. Johnson et al. (2015) showed that the parameter estimation risk of ℓ_0-penalized regression can be infinitely better than that of the ℓ_1-penalized regression under certain conditions and this is a practical example. The shape of the relative MAE from ProSGPV generally follows what would be expected from a rate of $\sqrt{\log(n)/n}$. This rate matches the ideal rate of parameter estimation in the optimal

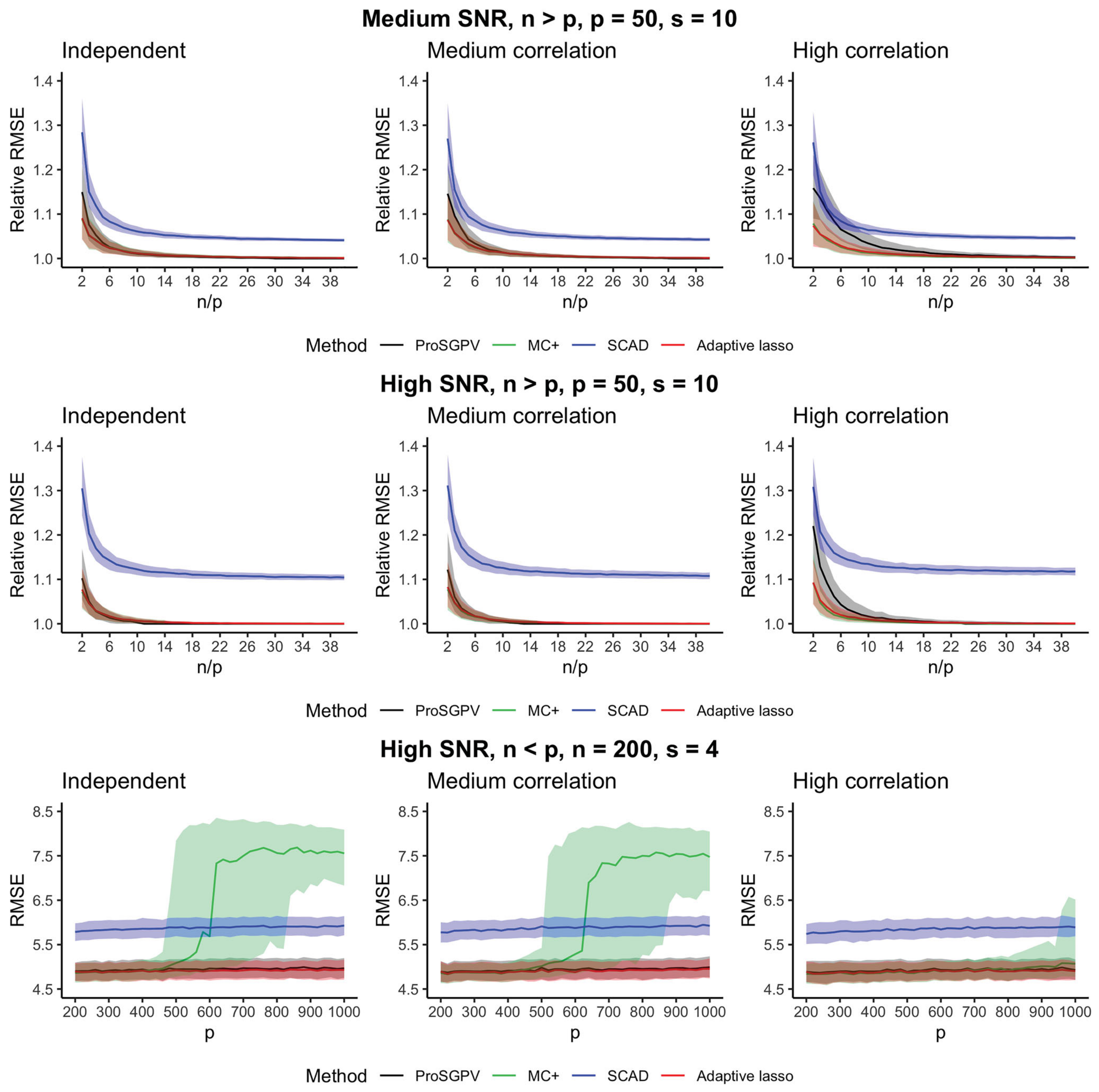

Figure 5. Comparison of prediction accuracy of all algorithms under combinations of autocorrelation level, SNR, and (n, p, s). Median (relative) root mean square errors are surrounded by their first and third quartiles over 1000 simulations.

model from any hard thresholding function, as suggested by (Zheng, Fan, and Lv 2014, theor. 1). When $n > p$, AL and MC+ have very close performance and SCAD has the slowest rate of convergence. When $n < p$, the order stays the same for all except MC+. As p passes 600, MC+ with GIC-based tunning selects more noise variables in the model and the parameter estimation performance is compromised. This can be remedied by using a universal $\lambda = \sigma\sqrt{(2/n)\log p}$ in MC+ (see supplementary material, Figure 5). Fan and Tang (2013) argued that MC+ has the same performance as SCAD when GIC is used. However, in their setting, s, p, and n are allowed to grow together. In our case, s is fixed at 4, n is fixed at 200, and only p grows.

In Figure 5, the prediction RMSE is calculated in an independent test set (40%) using models built with a training set (60%). Again, when $n > p$ the relative RMSE is used while when $n < p$ the absolute RMSE is used. Relative RMSE is defined as the ratio of the prediction RMSE from one algorithm to that from the OLS model with true signals only.

When $n > p$, all algorithms have worse prediction performance than the true OLS model unless n is really large. But their prediction RMSEs converge to the true OLS RMSE from above as n increases. While ProSGPV is not optimized for prediction tasks, its predictive ability quickly catches up with other algorithms when $n/p > 6$. When $n < p$, ProSGPV and AL have the best performance followed by SCAD. SCAD can have better prediction performance when λ is selected by cross-validation. However, in that case, its support recovery is worse than that from the GIC-based SCAD. MC+ has much higher prediction error than the others when $p > 600$. That is because λ selected by GIC leads to a dense model which includes many noise variables. The overfitted model has poor prediction performance in an external dataset. However, this can be remedied by using a universal λ in MC+, as shown in Figure 5 (supplementary material).

In Figure 6 (supplementary material), the running time in seconds from all algorithms are compared. The computing environment was 2.6 GHz Dual-Core Intel Core i7 processor and 32 GB memory. ProSGPV and AL have the shortest computation time, followed by SCAD. MC+ is more time-consuming when data are highly correlated, or when $n < p$.

5. Real-World Example

We illustrate our approach using the Tehran housing data (Rafiei and Adeli (2016)), which was high SNR ($R^2 = 0.98$) in the OLS model with all variables. We also explored the medium SNR case by removing potentially redundant variables until $R^2 = 0.4$. The Tehran housing data are available as a data object t.housing in the *ProSGPV* package. The dataset contains 26 features and 372 records (see supplementary material, Table 2 for the variable description). The goal is to predict the sale price (variable 9 or V9). The explanatory variables consist of seven project physical and financial variables, 19 economic variables, all at baseline. Clustering and correlation patterns are displayed in Figure 7 (supplementary material). We see that several explanatory variables form prominent clusters and that there is high pairwise correlation among the features. In particular, the price per square meter of the unit at the beginning of the project (V8) has high correlation ($\rho = 0.98$) with the sale price (V9).

We repeatedly split the data into a training set (70%) and a test set (30%). We applied AL, SCAD, MC+, and ProSGPV algorithms on the training set (n=260) with all the covariates. Prediction RMSE was calculated on the test set (n=112). We summarized the sparsity of the solutions (supplementary material, Figure 8) and prediction accuracy (supplementary material, Figure 9) over 1000 training-test split repetitions. SCAD overfits the training data and has the largest selection set. AL yields the sparsest model followed by ProSGPV. GIC-based MC+ yields a constant model size. Regarding the prediction performance, ProSGPV has the lowest median prediction error closely followed by AL and MC+, while SCAD has the largest test error because of overfitting. All algorithms select duration of construction (V7) and initial price per square meter (V8) with high frequency, and there is no consensus as for which other variables to include because of high correlation and clustering.

To refine the analysis, we removed variables that had an absolute correlation with the outcome of 0.45 or greater. The remaining covariates explain 40% of the variability in the response, which represents medium SNR. Of the remaining nine variables, MC+ always selects zero variables. ProSGPV selects four or five variables with high frequency. AL selects six or seven variables with high frequency. SCAD selects more variables than ProSGPV and AL. ProSGPV, SCAD and AL have similar prediction performance, while MC+ has worse performance, due to the null model it selects. The common selected variables include total floor area of the building (V2), lot area (V3), the price per square meter of the unit at the beginning of the project (V8), and the number of building permits issued (V11). The R code to replicate the results is available at *https://github.com/zuoyi93/r-code-prosgpv-linear*.

6. Practical Implications, Limitations, and Comments

A naïve way to perform variable selection is to screen variables by p-values. Such methods include forward selection, backward selection, and stepwise selection (Efroymson 1966). However, these methods have serious drawbacks. They have poor capture rates of the true underlying model (Wang 2009; Kozbur 2018) and larger effective degrees of freedom (Kaufman and Rosset 2014; Janson, Fithian, and Hastie 2015). In addition, the standard errors of the coefficient estimates are too small, which leads to over-optimistic discoveries (Harrell Jr 2015). Better approaches do exist, but they are more complex, require specialized software, and are not fully adopted in routine applied practice. However, ProSGPV offers a simple and effective option. It exhibits excellent statistical properties in both inference and prediction tasks without increased computation time.

Our simulation studies reinforce the notion that a model with good prediction ability does not necessarily lead to good inference. Comparing Figure 3 with Figure 5, we see that models optimized for prediction tend not to be optimized for inferential tasks, even when we use a different parameter tuning approach for each algorithm. This corroborates findings in the literature (Leng, Lin, and Wahba 2006; Meinshausen et al. 2006; Wasserman and Roeder 2009; Zheng, Fan, and Lv 2014; Giacobino et al. 2017; Shortreed and Ertefaie 2017). This statement is important and bears repeating: models optimized for prediction tasks do not necessarily support good inference. Similar observations can be found by comparing the parameter estimation in Figure 4 with prediction performance in Figure 5. ProSGPV does a better job by yielding a model that is primed for inference and also has good prediction properties.

There is a link between SNR and the proportion of variance explained (PVE).

$$\mathrm{PVE}(f) = 1 - \frac{\mathbb{E}(y - f(x)^2)}{\mathrm{var}(y)} = 1 - \frac{\mathrm{var}(\epsilon)}{\mathrm{var}(y)} = \frac{\mathrm{SNR}}{1 + \mathrm{SNR}}, \quad (5)$$

where f is the mean function, and x is independent from ϵ. Practically, when the R^2 is around 0.40 in the full model, which is equivalent to a medium SNR in our simulation, ProSGPV has a comparable performance in support recovery and slightly better parameter estimation in large n; when the R^2 is above 0.66, which corresponds to a high SNR, ProSGPV has superior inference properties than the other algorithms.

There are some limitations. Sensitivity to tuning parameter specification is an issue for both implementation and generalizability of results. However, we found the findings to be fairly robust to tuning parameter specification. In results not shown here, we repeated the experiment using each algorithm's preferred method for choosing a tuning parameter and the general ordering of results remained stable. Another limitation is when the design matrix has high within-correlation, which is a challenging problem for any algorithm. Not unexpectedly, ProSGPV does not do well in support recovery and its parameter estimation and prediction performance suffers. We also note that the exact threshold function of the two-stage ProSGPV algorithm is difficult to conceptualize, as the null bound in the fully relaxed lasso has a different feature space than the full feature space. We are actively working on formulating solutions for the two-stage algorithm.

Despite these relatively minor limitations, the ProSGPV algorithm looks very promising. It gives up little in terms of prediction, and offers improved support recovery and parameter estimation properties compared to the class of standard procedures currently in use today. Also, the ProSGPV algorithm does not depend on tuning parameters that are hard to specify. It is fair to say that unlike traditional p-values, SGPVs can be used for variable selection and subsequent statistical inference.

Supplementary Materials

Supplementary materials include all supplementary tables and figures, as well as the R code to replicate the simulation studies and the real-world example.

References

Blume, J. D., McGowan, Lucy D'Agostino, A., Dupont, W. D., and Greevy, R. A. Jr. (2018), "Second-Generation p-Values: Improved Rigor, Reproducibility, & Transparency in Statistical Analyses," *PLoS One*, 13, e0188299. [1,3,4]

Blume, J. D., Greevy, R. A., Welty, V. F., Smith, J. R., and Dupont, W. D. (2019), "An Introduction to Second-Generation p-Values," *The American Statistician*, 73, 157–167. [1,3,4]

Bogdan, M., Van Den Berg, E., Sabatti, C., Su, W., and Candès, E. J. (2015), "Slope—Adaptive Variable Selection Via Convex Optimization," *The Annals of Applied Statistics*, 9, 1103. [1]

Efroymson, M. (1966), "Stepwise Regression–A Backward and Forward Look," in *Proceedings Eastern Regional Meetings of the Institute of Mathematical Statistics*, pp. 27–29. [10]

Fan, J., and Li, R. (2001), "Variable Selection Via Nonconcave Penalized Likelihood and Its Oracle Properties," *Journal of the American Statistical Association*, 96, 1348–1360. [1,2,4]

——— (2006), "Statistical Challenges With High Dimensionality: Feature Selection in Knowledge Discovery," arXiv: preprint math/0602133. [4]

Fan, J., and Lv, J. (2008), "Sure Independence Screening for Ultrahigh Dimensional Feature Space," *Journal of the Royal Statistical Society*, Series B, 70, 849–911. [5]

——— (2011), "Nonconcave Penalized Likelihood With np-Dimensionality," *IEEE Transactions on Information Theory*, 57, 5467–5484. [2]

——— (2013), "Asymptotic Equivalence of Regularization Methods in Thresholded Parameter Space," *Journal of the American Statistical Association*, 108, 1044–1061. [2]

Fan, Y., and Tang, C. Y. (2013), "Tuning Parameter Selection in High Dimensional Penalized Likelihood," *Journal of the Royal Statistical Society*, Series B, 531–552. [4,9]

Giacobino, C., Sardy, S., Diaz-Rodriguez, J., and Hengartner, N. (2017), "Quantile Universal Threshold," *Electronic Journal of Statistics*, 11, 4701–4722. [10]

Harrell, F. E. Jr. (2015), *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*, New York: Springer. [10]

Hastie, T., Tibshirani, R., and Tibshirani, R. (2020), "Best Subset, Forward Stepwise or Lasso? Analysis and Recommendations Based on Extensive Comparisons," *Statistical Science*, 35, 579–592. [6]

Heinze, G., Wallisch, C., and Dunkler, D. (2018), "Variable Selection–A Review and Recommendations for the Practicing Statistician," *Biometrical Journal*, 60, 431–449. [1]

Janson, L., Fithian, W., and Hastie, T. J. (2015), "Effective Degrees of Freedom: A Flawed Metaphor," *Biometrika*, 102, 479–485. [5,10]

Johnson, K. D., Lin, D., Ungar, L. H., Foster, D. P., and Stine, R. A. (2015), "A Risk Ratio Comparison of *l*_0 and *l*_1 Penalized Regression," arXiv: 1510.06319. [8]

Kaufman, S., and Rosset, S. (2014), "When Does More Regularization Imply Fewer Degrees of Freedom? Sufficient Conditions and Counterexamples," *Biometrika*, 101, 771–784. [5,10]

Knight, K., and Fu, W. (2000), "Asymptotics for Lasso-Type Estimators," *Annals of Statistics*, 1356–1378. [2]

Kozbur, D. (2018), "Sharp Convergence Rates for Forward Regression in High-Dimensional Sparse Linear Models," University of Zurich, Department of Economics, Working Paper, 1(253). [10]

Leng, C., Lin, Y., and Wahba, G. (2006), "A Note on the Lasso and Related Procedures in Model Selection," *Statistica Sinica*, 16, 1273–1284. [1,10]

Loh, P.-L., and Wainwright, M. J. (2015), "Regularized m-Estimators With Nonconvexity: Statistical and Algorithmic Theory for Local Optima," *The Journal of Machine Learning Research*, 16, 559–616. [2]

Meinshausen, N. (2007), "Relaxed Lasso," *Computational Statistics & Data Analysis*, 52, 374–393. [4]

Meinshausen, N., and Bühlmann, P. (2006), "High-Dimensional Graphs and Variable Selection With the Lasso," *The Annals of Statistics*, 34, 1436–1462. [1,5,10]

Meinshausen, N., and Yu, B. (2009), " Lasso-Type Recovery of Sparse Representations for High-Dimensional Data," *The Annals of Statistics*, 37, 246–270. [5]

Hossein Rafiei, M., and Adeli, H. (2016), "A Novel Machine Learning Model for Estimation of Sale Prices of Real Estate Units," *Journal of Construction Engineering and Management*, 142, 04015066. [10]

Shmueli, G. (2010), "To Explain or to Predict?" *Statistical Science*, 25, 289–310. [1]

Shortreed, S. M., and Ertefaie, A. (2017), "Outcome-Adaptive Lasso: Variable Selection for Causal Inference," *Biometrics*, 73, 1111–1122. [1,10]

Sun, Q., Jiang, B., Zhu, H., and Ibrahim, J. G. (2019), "Hard Thresholding Regression," *Scandinavian Journal of Statistics*, 46, 314–328. [4,5]

Tibshirani, R. (1996), "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society*, Series B, 58, 267–288. [1,2]

van de Geer, S., Bühlmann, P., and Zhou, S. (2011), "The Adaptive and the Thresholded Lasso for Potentially Misspecified Models (and a Lower Bound for the Lasso)," *Electronic Journal of Statistics*, 5, 688–749. [5]

Wainwright, M. J. (2009a), "Information-Theoretic Limits on Sparsity Recovery in the High-Dimensional and Noisy Setting," *IEEE Transactions on Information Theory*, 55, 5728–5741. [4,5]

——— (2009b), "Sharp Thresholds for High-Dimensional and Noisy Sparsity Recovery Using ℓ_1-Constrained Quadratic Programming (Lasso)," *IEEE Transactions on Information Theory*, 55, 2183–2202. [2]

Wang, H. (2009), "Forward Regression for Ultra-High Dimensional Variable Screening," *Journal of the American Statistical Association*, 104, 1512–1524. [10]

Wang, L., Kim, Y., and Li, R. (2013), "Calibrating Non-Convex Penalized Regression in Ultra-High Dimension," *Annals of Statistics*, 41, 2505. [2,4]

Wang, S., Weng, H., and Maleki, A. (2020), "Which Bridge Estimator is the Best for Variable Selection?" *Annals of Statistics*, 48, 2791–2823. [5]

Wasserman, L., and Roeder, K. (2009), "High Dimensional Variable Selection," *Annals of Statistics*, 37, 2178. [5,10]

Weng, H., Feng, Y., and Qiao, X. (2019), "Regularization After Retention in Ultrahigh Dimensional Linear Regression Models," *Statistica Sinica*, 29, 387–407. [5]

Zhang, C.-H. (2010), "Nearly Unbiased Variable Selection Under Minimax Concave Penalty," *The Annals of Statistics*, 38, 894–942. [1,2]

Zhang, T. (2009), "Some Sharp Performance Bounds for Least Squares Regression With l1 Regularization," *The Annals of Statistics*, 37, 2109–2144. [1,5]

Zhao, P., and Yu, B. (2006), "On Model Selection Consistency of Lasso," *Journal of Machine Learning Research*, 7, 2541–2563. [4,5]

Zheng, Z., Fan, Y., and Lv, J. (2014), "High Dimensional Thresholded Regression and Shrinkage Effect," *Journal of the Royal Statistical Society*, Series B, 627–649. [2,9,10]

Zhou, S. (2009), "Thresholding Procedures for High Dimensional Variable Selection and Statistical Estimation," *Advances in Neural Information Processing Systems*, 22, 2304–2312. [5]

——— (2010), "Thresholded Lasso for High Dimensional Variable Selection and Statistical Estimation," arXiv: 1002.1583. [5]

Zou, H. (2006), "The Adaptive Lasso and Its Oracle Properties," *Journal of the American Statistical Association*, 101, 1418–1429. [1,2,4]