The Strong Screening Rule for SLOPE

Johan Larsson[1,*], Małgorzata Bogdan[1,2], and Jonas Wallin[1]

[1] *Department of Statistics, Lund University*
[2] *Department of Mathematics, University of Wroclaw*
[*] *Corresponding author: johan.larsson@stat.lu.se*

May 11, 2020

Abstract

Extracting relevant features from data sets where the number of observations (n) is much smaller then the number of predictors (p) is a major challenge in modern statistics. Sorted L-One Penalized Estimation (SLOPE)—a generalization of the lasso—is a promising method within this setting. Current numerical procedures for SLOPE, however, lack the efficiency that respective tools for the lasso enjoy, particularly in the context of estimating a complete regularization path. A key component in the efficiency of the lasso is predictor screening rules: rules that allow predictors to be discarded before estimating the model. This is the first paper to establish such a rule for SLOPE.

We develop a screening rule for SLOPE by examining its subdifferential and show that this rule is a generalization of the strong rule for the lasso. Our rule is heuristic, which means that it may discard predictors erroneously. We present conditions under which this may happen and show that such situations are rare and easily safeguarded against by a simple check of the optimality conditions.

Our numerical experiments show that the rule performs well in practice, leading to improvements by orders of magnitude for data in the $p \gg n$ domain, as well as incurring no additional computational overhead when $n \gg p$. We also examine the effect of correlation structures in the design matrix on the rule and discuss algorithmic strategies for employing the rule. Finally, we provide an efficient implementation of the rule in our R package SLOPE.

1 Introduction

Extracting relevant features from data sets where the number of observations (n) is much smaller then the number of predictors (p) is one of the major challenges in modern statistics. The dominating method for this problem, in a regression setting, is the lasso [29]. Recently, however, an alternative known as Sorted L-One Penalized Estimation (SLOPE) has been proposed [3].

SLOPE[1] is a regularization method where one uses the sorted ℓ_1 norm, instead

[1] We recognize that lasso and SLOPE specifically refer to the regularized ordinary least squares problem. We will not, hower, make this distinction here and let the lasso and SLOPE be synonymous with the general form of the ℓ_1-regularized and sorted ℓ_1-regularized problem respectively.

of the regular ℓ_1 norm, which is used in the lasso. SLOPE is characterized by several interesting theoretical properties, such as control of the false discovery rate [3], asymptotic minimaxity [27], and clustering of regression coefficients in the presence of strong dependence between predictors [36].

Like the lasso, however, SLOPE depends on hyper-parameters that often need to be selected using cross-validation, which results in the need for solving a large number of optimization problems. As a consequence, there is ample demand for algorithms with which this problem can be tackled efficiently—this is the issue we address in this paper.

In more detail, SLOPE solves the convex optimization problem

$$\text{minimize}_{\beta \in \mathbb{R}^p} \left\{ f(\beta) + J(\beta; \lambda) \right\}, \tag{1}$$

where $f(\beta)$ is smooth and convex and $J(\beta; \lambda) = \sum_{j=1}^{p} \lambda_j |\beta|_{(j)}$ is the convex but non-smooth sorted ℓ_1 norm [3, 36], where $|\beta|_{(1)} \geq |\beta|_{(2)} \geq \cdots \geq |\beta|_{(p)}$ and $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$.

This problem was introduced by Bondell and Reich [4] in the form of OSCAR (octagonal shrinkage and clustering algorithm for regression), in which λ is chosen to be a linearly decaying sequence. Bogdan et al. [2], Bogdan et al. [3] and Zeng and Figueiredo [35] generalized the OSCAR formulation and developed methods to efficiently solve the problem, referring to them as SLOPE (Sorted L-One Penalized Estimation) [2, 3] and OWL (ordered weighted L1) regression [35] respectively.

It is easy to see that the lasso is a specific instance of SLOPE, which can be obtained by setting all elements of λ to the same value. But in constrast to SLOPE, the lasso suffers from unpredictable behavior in the presence of highly correlated predictors, resulting in solutions wherein only a subset among a group of correlated predictors is selected. Nonetheless, Yuan and Lin [33] and Jia and Yu [18] showed that the elastic net (a mix of ℓ_1 and ℓ_2 regularization) remedies this issue. More specifically, Yuan and Lin [33] and Jia and Yu [18] proved that the elastic net consistently selects the true model provided that the *elastic irrepresentability condition* holds. This condition, however, is weaker than the respective irrepresentability condition for the lasso and is satisfied even when the true predictors are linearly dependent.

SLOPE too turns out to be robust to correlated designs, which it accomplishes via clustering: setting coefficients of correlated predictors to the same value [36]. Figueiredo and Nowak [12] revealed conditions under which this holds and remarkably provided finite-sample bounds for this result. Additional results on the clustering properties of SLOPE are provided in Kremer et al. [21] and Schneider and Tardivel [26], where it is explained that the clustering of SLOPE coefficients is driven by the similarity of the influence of respective variables on the likelihood function, which may happen due to the strong correlation but also due to the similarity of true regression coefficients.

The choice of λ sequence in (1) typically needs to be based on cross-validation or similar schemes. Most algorithms for fitting sparse regression, such as as the one implemented for lasso in the glmnet package for R [13], accomplish this by constructing a path of decreasing λ—starting from the choice that results in a completely sparse model. This design is appealing since it makes effective use of warm starts and because over-saturated fits can be avoided by prematurely stopping the regularization path.

For the SLOPE problem, we begin the path with $\lambda^{(1)}$ and finish at $\lambda^{(l)}$, where $\lambda_j^{(m)} \geq \lambda_j^{(m+1)}$ for all $j = 1, 2, \dots, p$ and $m = 1, 2, \dots, l$. For any point along this path, we let $\hat{\beta}(\lambda^{(m)})$ be the respective SLOPE estimate, such that

$$\hat{\beta}(\lambda^{(m)}) = \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p} \left\{ f(\beta) + J(\beta; \lambda^{(m)}) \right\}.$$

Fitting the path repeatedly by cross-validation undeniably introduces a heavy computational burden. Consider, for instance, that we are interested in tuning our λ sequence with K times repeated k-fold cross validation for a path with length l. This clearly means fitting Kkl models (of varying complexity). Combine this with the requirement that our "optimal" λ is identified with a reasonable precision—most implementations of the lasso use a default path length of roughly 100—and it is easy to see that the computational load is heavy for large p or n.

For the lasso, an important remedy for this problem arose with the advent of screening rules, which provide criteria for setting part of the solution vector to zero—effectively discarding the respective predictors—before attempting to fit the model.

Screening rules can be broken down into two categories: *safe* and *heuristic* screening rules. The former of these guarantee that any predictors screened as inactive (determined to be zero by the rule) are in fact zero at the solution. This is not the case for the latter category, heuristic rules, which may lead to *violations:* incorrectly discarding predictors. This latter fact means that heuristic rules must be supplemented with a check of the Karush–Kuhn–Tucker (KKT) conditions. For any predictors failing the test, the model must be refit with these predictors added back in order to ensure optimality. There has also been attempts to combine safe and heuristic screening rules into so called *hybrid* screening rules [38].

Safe screening rules include the safe feature elimination rule (SAFE [10]), the dome test [32], Enhanced Dual-Polytope Projection (EDPP [31]), and Gap Safe rule [25]. In essence, all of the safe screening rules attempt to bound the dual solution to a region and then use the KKT criteria to determine whether a given predictor might potentially be active at the solution.

Heuristic rules include Sure Independence Screening (SIS [11]) and the strong rule [30], as well as *working set* methods such as Blitz [20] and the working set algorithm from Tibshirani et al. [30] that is implemented in glmnet.

There have been no systematic comparisons between the various screening rules, but many papers suggest that the strong rule (and its working set algorithm), Blitz, Gap Safe, and EDPP rules [24, 25, 31] may be among the most effective methods. It may be instructive to note that the presence of violations of a given rule is not excessively detrimental to its performance. The working set heuristic for the strong rule, for instance, uses the set of predictors that have at least once been active previously on the path as a working set [30], which causes numerous violations, yet still manages to be competitive[2].

The implications of screening rules have been striking, allowing models in the $p \gg n$ domain to be solved in a fraction of the time required otherwise—Tibshirani et al. [30] for instance showed improvements in speed of up to 40 times.

[2] The algorithm checks the strong set against the KKT conditions first and refits if there are any failures in that set. Only when the strong set is free from violations does the algorithm check the KKT conditions for the entire set of predictors.

In addition, datasets that are otherwise too large to even fit in memory can be screened offline and the lasso optimization problem can be solved completely via a smaller subset of the original dataset (that does fit in memory).

Despite the evident impact of and considerable interest in screening rules for the class of lasso-type problems, there has so far been no attempts to derive screening rules for SLOPE. In addition, SLOPE involves a proximal operator that is computationally more demanding than that of the lasso[3], which indicates that the potential benefits of screening rules may be even larger for SLOPE. In this paper we will tackle this issue, by presenting a heuristic screening rule for SLOPE based on the strong rule for the lasso.

1.1 Outline of the Paper

In section 2 we derive strong rules for SLOPE and provide a linear-time algorithm for implementing them in practice. In section 3 we then test our rules on a number of simulated and real data sets, showing performance gains comparable to those of the original strong rules.

1.2 Notation

Throughout the paper, we use uppercase letters for matrices and lowercase letters for vectors and scalars. $\mathbf{1}$ and $\mathbf{0}$ denote vectors with all elements equal to 1 and zero respectively, with dimension inferred from context. We use $\prec$ and $\succ$ to denote element-wise relational operators. We also let $\operatorname{card}\mathcal{A}$ denote the cardinality of set $\mathcal{A}$ and define $\operatorname{sign} x$ to be the signum function with range $\{-1, 0, 1\}$. Furthermore, we define $x_\downarrow$ to refer to a version of x sorted in decreasing order and the cumulative sum function for a vector $x \in \mathbb{R}^n$ as

$$\operatorname{cumsum}(x) = \begin{bmatrix} x_1 & x_1 + x_2 & \cdots & \sum_{i=1}^n x_i \end{bmatrix}^T.$$

We also let $|i|$ be the index operator of $y \in \mathbb{R}^p$ so that $|y_{|i|}| = |y|_{(i)}$ for all $i = 1, \dots, p$. Finally, we allow a vector to be subset with elements from an integer-valued set by including elements from that vector in order if its index value is an element of that set. For instance, if $\mathcal{A} = \{3, 1\}$ and $v = [v_1, v_2, v_3]^T$, then $v_\mathcal{A} = [v_1, v_3]^T$.

2 Theory

2.1 The Subdifferential for SLOPE

The basis of the strong rule for ℓ_1-regularized models is the subdifferential. By the same argument, we now turn to the subdifferential of SLOPE. The subdifferential for SLOPE has been derived previously [6], but here (Theorem 1) we offer a version better tailored to our purposes. First, however, let $\mathcal{A}_i(\beta) \subseteq \{1, \dots, p\}$ denote a set of indices for $\beta \in \mathbb{R}^p$ such that

$$\mathcal{A}_i(\beta) = \{j \in \{1, \dots, p\} \mid |\beta_i| = |\beta_j|\} \tag{2}$$

[3] Current state-of-the art algorithms for the proximal operator of the sorted ℓ_1 norm has an average complexity of $\mathcal{O}(p \log p)$ [3, 37] compared to $\mathcal{O}(p)$ for the lasso.

where $\mathcal{A}_i(\beta) \cap \mathcal{A}_l(\beta) = \varnothing$ if $l \notin \mathcal{A}_i(\beta)$. To keep notation to a minimum, we let $\mathcal{A}_i$ serve as a shorthand for $\mathcal{A}_i(\beta)$.

In addition, we define the operator $O : \mathbb{R}^p \to \mathbb{N}^p$, which returns a permutation that rearranges its argument in descending order by its absolute values and $R : \mathbb{R}^p \to \mathbb{N}^p$, which returns the ranks of the absolute values in its argument.

Example 1. *Suppose that we have* $\beta = \{-3, 5, 3, 6\}$*. Then* $\mathcal{A}_1 = \{1, 3\}$*,* $O(\beta) = \{4, 2, 1, 3\}$*, and* $R(\beta) = \{3, 2, 4, 1\}$*.*

Theorem 1. *The subdifferential* $\partial J(\beta; \lambda) \in \mathbb{R}^p$ *is the set of all* $g \in \mathbb{R}^p$ *such that*

$$g_{\mathcal{A}_i} = \left\{ s \in \mathbb{R}^{\operatorname{card} \mathcal{A}_i} \;\middle|\; \begin{cases} \operatorname{cumsum}(|s|_{\downarrow} - \lambda_{R(s)_{\mathcal{A}_i}}) \preceq \mathbf{0} & \text{if } \beta_{\mathcal{A}_i} = \mathbf{0}, \\ \operatorname{cumsum}(|s|_{\downarrow} - \lambda_{R(s)_{\mathcal{A}_i}}) \preceq \mathbf{0} & \\ \quad \wedge \sum_{j \in \mathcal{A}_i} \left(|s_j| - \lambda_{R(s)_j}\right) = 0 & \text{otherwise.} \end{cases} \right\}$$

Proof. By definition, the subdifferential $\partial J(\beta; \lambda)$ is the set of all $g \in \mathbb{R}^p$ such that

$$J(y; \lambda) \geq J(\beta; \lambda) + g^T(y - \beta) = \sum_{j=1}^{p} |\beta|_{(j)} \lambda_j + g^T(y - \beta), \tag{3}$$

for all $y \in \mathbb{R}^p$.

Assume that we have K clusters $\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_K$ (as defined per (2)) and that $\beta = |\beta|_{\downarrow}$, which means we can rewrite (3) as

$$\begin{aligned} 0 &\geq J(\beta; \lambda) - J(y; \lambda) + g^T(y - \beta) \\ &= \sum_{i \in \mathcal{A}_1} (\lambda_i |\beta|_{(i)} - g_i \beta_i - \lambda_i |y|_{(i)} + g_i y_i) + \dots \\ &\quad + \sum_{i \in \mathcal{A}_K} (\lambda_i |\beta|_{(i)} - g_i \beta_i - \lambda_i |y|_{(i)} + g_i y_i). \end{aligned}$$

Notice that we must have $\sum_{i \in \mathcal{A}_j} (\lambda_i |\beta|_{(i)} - g_i \beta_i - \lambda_i |y|_{(i)} + g_i y_i) \leq 0$ for all $j \in \{1, 2, \dots, K\}$ since otherwise the inequality breaks by selecting $y_i = \beta_i$ for $i \in \mathcal{A}_j^c$. This means that it is sufficient to restrict attention to a single set as well as take this to be the set $\mathcal{A}_i = \{1, \dots, p\}$.

Case 1 ($\beta = \mathbf{0}$). In this case (3) reduces to $J(y; \lambda) \geq g^T y$. Now take a $c \in \mathcal{Z}$ where

$$\mathcal{Z} = \left\{ s \in \mathbb{R}^p \;\middle|\; \operatorname{cumsum}(|s|_{\downarrow} - \lambda) \preceq \mathbf{0} \right\} \tag{4}$$

and assume that $|c_1| \geq \dots \geq |c_p|$ without loss of generality.

Clearly, $J(y; \lambda) \geq c^T y$ holds if and only if $J(y; \lambda) - c^T y^* \geq 0$ where

$$y^* = \operatorname*{arg\,min}_y \left\{ J(y; \lambda) - c^T y \right\}.$$

Now, since $J(y; \lambda)$ is invariant to changes in signs and permutation of y, it follows from the rearrangement inequality (see e.g. Theorem 368 in Hardy, Littlewood, and Pólya [16]) that $|y|_1^* \geq \dots \geq |y|_p^*$. This permits us to formulate the following equivalent problem:

$$\begin{aligned} &\text{minimize} && y^T(\operatorname{sign}(y) \odot \lambda - c) \\ &\text{subject to} && \operatorname{sign}(y) = \operatorname{sign}(c), \\ &&& |y_1| \geq \dots \geq |y_p|. \end{aligned}$$

To minimize the objective $y^T(\operatorname{sign}(y) \odot \lambda - c) = |y|^T(\lambda - c)$, recognize first that we must have $y_1^* = y_2^*$ since $\lambda_1 - c_1 \geq 0$. Likewise, $y_2^*(\lambda_1 - c_1) + y_2^*(\lambda_2 - c_2) \geq 0$ since $\lambda_1 + \lambda_2 - (c_1 + c_2) \geq 0$, which leads us to conclude that $y_2^* = y_3^*$. Then, proceeding inductively, it is easy to see that $y_p^* \sum_{i=1}^p (\lambda_i - c_i) \geq 0$, which implies $y_1^* = \cdots = y_p^* = 0$. At this point, we have shown that $c \in \mathcal{Z} \implies c \in \partial J(\beta; \lambda)$.

For the next part note that (4) is equivalent to requiring $|g|_{(1)} \leq \lambda_1$ and

$$|g|_{(i)} \leq \sum_{j=1}^{i} \lambda_j - \sum_{j=2}^{i} |g|_{(j)}, \qquad i = 1, \dots, p. \tag{5}$$

Now assume that there is a c such that $c \in \partial J(\beta; \lambda)$ and $c \notin \mathcal{Z}$. Then there exists an $\varepsilon > 0$ and $i \in \{1, 2, \dots, p\}$ such that

$$|c|_{(i)} \leq \sum_{j=1}^{i} \lambda_j - \sum_{j=2}^{i} |c|_{(j)} + \varepsilon, \qquad i = 1, \dots, p.$$

Yet if $c = [\lambda_1, \dots, \lambda_{i-1}, \lambda_i + \varepsilon, \lambda_{i+1}, \dots, \lambda_p]^T$ then (3) breaks for $y = \mathbf{1}$, which implies that $c \notin \mathcal{Z} \implies c \notin \partial J(\beta; \lambda)$.

Case 2 ($\beta \neq \mathbf{0}$). Now let $|\beta_i| \coloneqq \alpha$ for all $i = 1, \dots, p$, since by construction all β are equal in absolute value. Now (3) reduces to

$$\begin{aligned} J(y; \lambda) &\geq J(\beta; \lambda) - g^T \beta + g^T y \\ &= \sum_{i=1}^{p} \lambda_i \alpha - \sum_{i=1}^{p} g_i \operatorname{sign}(\beta_i) \alpha + g^T y \\ &= \alpha \sum_{i=1}^{p} (\lambda_i - g_i \operatorname{sign}(\beta_i)) + g^T y. \end{aligned} \tag{6}$$

The first term on the right-hand side of the last equality must be zero since otherwise the inequality breaks for $y = \mathbf{0}$. In addition, it must also hold that $\operatorname{sign}(\beta_i) = \operatorname{sign}(g_i)$ for all i such that $|\beta_i| > 0$. To show this, suppose the opposite is true, that is, there exists at least one j such that $\operatorname{sign}(g_j) \neq \operatorname{sign}(\beta_j)$. But then if we take $y_j = \alpha \operatorname{sign}(g_j)$ and $y_i = -\alpha \operatorname{sign}(g_i)$, (6) is violated, which proves the statement by contradiction.

Taken together, this means that we have $g \in \mathcal{H}$ where

$$\mathcal{H} = \left\{ s \in \mathbb{R}^p \mid \sum_{j=1}^{p} (|s_j| - \lambda_j) = 0. \right\}$$

We are now left with $J(y; \lambda) \geq g^T y$, but this is exactly the setting from case one. Direct application of the reasoning from that part shows that we must have $g \in \mathcal{Z}$. Connecting the dots, we finally conclude that $c \in \mathcal{Z} \cap \mathcal{H} \implies c \in \partial J(\beta; \lambda)$.

□

2.2 Screening Rule for SLOPE

2.2.1 Sparsity Pattern

Recall that we are attempting to solve the following problem: we know $\hat{\beta}(\lambda^{(m)})$ and want to predict the support of $\hat{\beta}(\lambda^{(m+1)})$, where $\lambda^{(m+1)} \preceq \lambda^{(m)}$. The KKT stationarity criterion for SLOPE is

$$\mathbf{0} \in \nabla f(\beta) + \partial J(\beta; \lambda), \tag{7}$$

where $\partial J(\beta; \lambda)$ is the subdifferential for SLOPE (Theorem 1). This means that if $\nabla f(\hat{\beta}(\lambda^{(m+1)}))$ was available to us, we could identify the support exactly. In Algorithm 1, we present an algorithm to accomplish this in practice.

Algorithm 1

```
Require: $c \in \mathbb{R}^p$, $\lambda \in \mathbb{R}^p$, where $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$.
$\mathcal{S}, \mathcal{B} \leftarrow \varnothing$
for $i \leftarrow 1, \ldots, p$ do
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{i\}$
    if $\sum_{j \in \mathcal{B}} \left(c_j - \lambda_j\right) \geq 0$ then
        $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{B}$
        $\mathcal{B} \leftarrow \varnothing$
    end if
end for
Return $\mathcal{S}$
```

In Proposition 1, we show that the result of Algorithm 1 with $c := |\nabla f(\hat{\beta}(\lambda^{(m+1)}))|_{\downarrow}$ and $\lambda := \lambda^{(m+1)}$ as input is certified to contain the true support set of $\hat{\beta}(\lambda^{(m+1)})$.

Proposition 1. *Taking $c := |\nabla f(\hat{\beta}(\lambda^{(m+1)}))|_{\downarrow}$ and $\lambda := \lambda^{(m+1)}$ as input to Algorithm 1 returns a superset of the the true support set of $\hat{\beta}(\lambda^{(m+1)})$.*

Proof. Suppose that we have $\mathcal{B} \neq \varnothing$ after running Algorithm 1. In this case we have

$$\operatorname{cumsum}(c_{\mathcal{B}} - \lambda_{\mathcal{B}}) = \operatorname{cumsum}\left(\left(\left|\nabla f(\hat{\beta}(\lambda^{(m+1)}))\right|_{\downarrow}\right)_{\mathcal{B}} - \lambda_{\mathcal{B}}^{(m+1)}\right) \prec \mathbf{0},$$

which implies via Theorem 1 and (7) that all predictors in $\mathcal{B}$ must be inactive and that $\mathcal{S}$ contains the true support set. □

Remark 1. *In Algorithm 1, we implicitly make use of the fact that the results are invariant to permutation changes within each cluster $\mathcal{A}_i$ (as defined in (2))—a fact that follows directly from the definition of the subdifferential (Theorem 1). In particular, this means that the indices for the set of inactive predictors will be ordered last in both $|\hat{\beta}|_{\downarrow}$ and $|\nabla f(\hat{\beta})|_{\downarrow}$; that is, for all $i, j \in \{1, 2, \ldots, p\}$ such that $\hat{\beta}_i = 0$, $\hat{\beta}_j \neq 0$,*

$$O(\nabla f(\hat{\beta}))_i > O(\nabla f(\hat{\beta}))_j \implies O(\hat{\beta})_i > O(\hat{\beta})_j,$$

which allows us to determine the sparsity in $\hat{\beta}$ via $\nabla f(\hat{\beta})$.

Proposition 1 implies that Algorithm 1 may lead to a conservative decision by potentially including some of the support of inactive predictors in the result, i.e. indices for which the corresponding coefficients are in fact zero. To see this, let $\mathcal{U} = \{l, l+1, \ldots, p\}$ be a set of inactive predictors and take $c := |\nabla f(\hat{\beta}(\lambda^{(m+1)}))|_{\downarrow}$. For every $k \in \mathcal{U}$, $k \geq l$ for which $\sum_{i=l}^{k}(c_i - \lambda_i) = 0$, $\{l, l+1, \ldots, k\}$ will be in the result of Algorithm 1 in spite of being inactive. This situation, however, occurs only when c is the true gradient at the solution and for this reason is of little practical importance.

Since the check in Algorithm 1 hinges only on the last element of the cumulative sum at any given time, we need only to store and update a single scalar instead of the full cumulative sum vector. Using this fact, we can derive a fast version of the rule (Algorithm 2), which returns k: the predicted number of active predictors at the solution[4].

Algorithm 2

```
Require: $c \in \mathbb{R}^p$, $\lambda \in \mathbb{R}^p$, where $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$
  $i \leftarrow 1$, $k \leftarrow 0$, $s \leftarrow 0$
  while $i + k \leq p$ do
    $s \leftarrow s + c_{i+k} - \lambda_{i+k}$
    if $s \geq 0$ then
      $k \leftarrow k + i$
      $i \leftarrow 1$
      $s \leftarrow 0$
    else
      $i \leftarrow i + 1$
    end if
  end while
  return $k$
```

Since we only have to take a single pass over the predictors, the cost of the algorithm is linear in p. To use the algorithm in practice, however, we first need to compute the gradient at the previous solution and sort it. Using ordinary least squares (OLS) regression as an example, this results in a complexity of $\mathcal{O}(np + p \log p)$. To put this into perspective, this is lower than the cost of a single gradient step if a first-order method is used to compute the SLOPE solution.

2.2.2 Gradient Approximation

The validity of Algorithm 1 requires $\nabla f(\hat{\beta}(\lambda^{(m+1)}))$ to be available, which of course is not the case. Assume, however, that we are given a reasonably accurate surrogate of the gradient vector and suppose that we substitute this estimate for $\nabla f(\hat{\beta}(\lambda^{(m+1)}))$ in Algorithm 1. Intuitively, this should yield us an estimate of the active set—the better the approximation, the more closely this screened set should resemble the active set. For the sequel, let $\mathcal{S}$ and $\mathcal{T}$ be the screened and active set respectively.

An obvious consequence of using our approximation is that we run the risk of picking $\mathcal{S} \not\subset \mathcal{T}$, which we then naturally must safeguard against. Fortunately, doing so requires only a KKT stationarity check—whenever the check fails, we relax $\mathcal{S}$ and refit. If such failures are rare, it is not hard to imagine that the benefits of tackling the reduced problem might outweigh the costs of these occasional failures.

[4] The active set is then retrieved by sub-setting the first k elements of the ordering permutation.

Based on this argument, we are now ready to state the strong rule for SLOPE, which is a natural extension of the strong rule for the lasso [30]. Let $\mathcal{S}$ be the output from running Algorithm 1 with $c := \big|\nabla f(\hat{\beta}(\lambda^{(m)})) + \lambda^{(m)} - \lambda^{(m+1)}\big|_{\downarrow}$ and $\lambda := \lambda^{(m+1)}$ as input. The strong rule for SLOPE then discards all predictors corresponding to $\mathcal{S}^{\mathsf{c}}$.

Proposition 2. *Let $c_j(\lambda) = (\nabla f(\hat{\beta}(\lambda)))_{|j|}$. If $|c_j'(\lambda)| \leq 1$ for all $j = 1, 2, \dots, p$ and $O(c(\lambda^{(m+1)})) = O(c(\lambda^{(m)}))$ (see Section 2.1 for the definition of O), the strong rule for SLOPE cannot produce any violations.*

Proof. We need to show that the strong rule approximation does not violate the inequality on the fourth line in Algorithm 1. Since $\operatorname{cumsum}(y) \succeq \operatorname{cumsum}(x)$ for all $x, y \in \mathbb{R}^p$ if and only iff $y \succeq x$, it suffices to show that

$$|c_j(\lambda^{(m)})| + \lambda_j^{(m)} - \lambda_j^{(m+1)} \geq |c_j(\lambda^{(m+1)})|$$

for all $j = 1, 2, \dots, p$, which in turn means that Algorithm 1 with $|c_j(\lambda^{(m)})| + \lambda_j^{(m)} - \lambda_j^{(m+1)}$ as input cannot result in any violations.

From our assumptions we have

$$|c_j(\lambda^{(m+1)}) - c_j(\lambda^{(m)})| \leq |\lambda_j^{(m+1)} - \lambda_j^{(m)}|.$$

Using this fact, observe that

$$\begin{aligned} |c_j(\lambda^{(m+1)})| &\leq |c_j(\lambda^{(m+1)}) - c_j(\lambda^{(m)})| + |c_j(\lambda^{(m)})| \\ &\leq \lambda_j^{(m)} - \lambda_j^{(m+1)} + |c_j(\lambda^{(m)})|. \end{aligned}$$

□

Except for the assumption on fixed ordering permutation, the proof for Proposition 2 is in fact exactly analogous to the proof of the strong rule for the lasso [30]. The authors referred to this bound as the *unit slope bound*, which results in the following rule for the lasso: discard the jth predictor if

$$\nabla f(\beta(\lambda^{(m)}))_j \leq 2\lambda^{(m+1)} - \lambda^{(m)}.$$

In Proposition 3, we formalize the connection between the strong rule for SLOPE and lasso.

Proposition 3. *The strong rule for SLOPE is a generalization of the strong rule for the lasso; that is, when $\lambda_j = \lambda_i$ for all $i, j \in \{1, \dots, p\}$, the two rules always produce the same screened set.*

Proof. Let $c = (\nabla f(\hat{\beta}(\lambda)))$ and $\lambda_1 = \lambda_2$ and assume without loss of generality that $p = 2$ and $c_1 \geq c_2 \geq 0$. Recall that the strong rule for lasso discards the jth predictor whenever $c_j < \lambda_1$. There are three cases to consider.

Case 1 ($c_2 \leq c_1 < \lambda_1$). $\operatorname{cumsum}(c - \lambda) \prec 0$, which means both predictors are discarded.

Case 2 ($c_1 \geq \lambda_1 > c_2$). The first predictor is retained since $\operatorname{cumsum}(c - \lambda)_1 > 0$; the second is discarded because $c_2 \leq \lambda$.

Case 3 ($c_1 \geq c_2 \geq \lambda_1$). Both predictors are retained since $\operatorname{cumsum}(c - \lambda) \succeq 0$.

The two results are equivalent for the lasso and thus the strong rule for SLOPE is a generalization of the strong rule for the lasso. □

2.2.3 Violations of the Rule

Violations of the strong rule for SLOPE occur only when the unit slope bound fails, which is equivalent to breaking the assumption that the gradient vector is a piece-wise linear function of the regularization sequence. To our knowledge, there are three situations by which this might happen:

- changes in the active set,
- changes in the ordering permutation, and
- clustering of coefficients.

The first kind affects the strong rule for the lasso too and has been described thoroughly elsewhere [30]. The second and third types are, as far as we know, specific to SLOPE.

In Section 3.2.2, we will study the prevalence of violations in simulated experiments.

2.2.4 Algorithms

Tibshirani et al. [30] considered two algorithms using the strong rule for the lasso. In this paper, we consider two algorithms that are analogous except in one regard. First, however, let $\mathcal{S}(\lambda)$ be the screened set, i.e. the set obtained by application of the strong rule for SLOPE, and $\mathcal{T}(\lambda)$ the active set. Both algorithms begin with a set $\mathcal{E}$ of predictors, fit the model to this set, and then either expand this set, refit and repeat, or stop.

The *strong set* algorithm is outlined in Algorithm 3. Here, we initialize $\mathcal{E}$ with the union of the strong set and the set of predictors active at the previous step on the regularization path. We then fit the model and check for KKT violations in the full set of predictors, expanding $\mathcal{E}$ to include any predictors for which violations occur and repeat until there are no violations.

Algorithm 3 Strong set algorithm

```
$\mathcal{V} \gets \varnothing$
$\mathcal{E} \gets \mathcal{S}(\lambda^{(m+1)}) \cup \mathcal{T}(\lambda^{(m)})$
do
    compute $\hat{\beta}_\mathcal{E}(\lambda^{(m+1)})$
    $\mathcal{V} \gets$ KKT violations in full set
    $\mathcal{E} \gets \mathcal{E} \cup \mathcal{V}$
while $\mathcal{V} \neq \varnothing$
return $\hat{\beta}_\mathcal{E}(\lambda^{(m+1)})$
```

The *previous set* algorithm is outlined in Algorithm 4. Here, we initialize $\mathcal{E}$ with only the set of previously active predictors, fit, and check the KKT conditions against the strong rule set. If there are violations in the strong set, the corresponding predictors are added to $\mathcal{E}$ and the model is refit. Only when there are no violations in the strong set do we check the KKT conditions in the full set, once again refitting until there are no violations.

These two algorithms differ from the strong and working set algorithms from Tibshirani et al. [30] in that we use only the set of previously active predictors

Algorithm 4 Previous set algorithm

```
$\mathcal{V} \leftarrow \varnothing$
$\mathcal{E} \leftarrow \mathcal{T}(\lambda^{(m)})$
do
    compute $\hat{\beta}_{\mathcal{E}}(\lambda^{(m+1)})$
    $\mathcal{V} \leftarrow$ KKT violations in $\mathcal{S}(\lambda^{(m+1)})$
    if $\mathcal{V} = \varnothing$ then
        $\mathcal{V} \leftarrow$ KKT violations in full set
    end if
    $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{V}$
while $\mathcal{V} \neq \varnothing$
return $\hat{\beta}_{\mathcal{E}}(\lambda^{(m+1)})$
```

rather than the set of ever-active predictors: predictors that have been active at least one previously on the path. The motivation for this difference is that SLOPE, in comparison with the lasso, behaves differently in the presence of strong correlation and "sufficiently" large gaps in the λ sequence. In this setting, the regularization path for SLOPE often starts with a relatively large set of predictors clustered in a few sets of predictors equal in absolute value[5]. For the first stretch of the regularization path, the cardinality of this set often decreases and even surpasses the cardinality of the set at termination of the path[6]. In these situations, the use of the ever-active set would clearly prove detrimental to performance, which is why we choose to use the set of previously active predictors instead. In Section 3.3.2 we compare the performance of the strong and previous set strategies.

3 Experiments

In this section we conduct simulations to examine the effects of applying the screening rules. The problems here reflect our focus on problems in the $p \gg n$ domain, but we will also briefly consider the reverse in order to examine the potential overhead of the rules when $n > p$.

We begin in Section 3.2 by studying simulated data and look at efficiency[7] (Section 3.2.1), violations (Section 3.2.2), and performance[8] (Section 3.2.3). In Section 3.3, we turn to real data, conducting experiments on efficiency (Section 3.3.1) and performance (Section 3.3.2).

3.1 Setup

Unless stated otherwise, we will use the strong set algorithm (Algorithm 3) with the strong set computed using the fast version of the strong rule for SLOPE (Algorithm 2). Unless stated otherwise, predictors were normalized such that

[5] In Section 3.2.1, we analyze this behavior in simulations.

[6] See Section 3.1.2 for a specification of the path termination criteria we use in our simulations.

[7] By efficiency we mean the size of the screened set relative to the active set.

[8] By performance we mean the computational costs of running our implementation with and without the rule.

$\bar{x}_j = 0$ and $\|x_j\|_2 = 1$ for $j = 1, \ldots, p$. In addition, we center the response vector y for ordinary least squares regression.

Throughout the paper we use version 0.2.1 of the R package SLOPE [22], in which we employ the accelerated proximal graident algorithm FISTA [1] to estimate all models. All simulations were run on a dedicated high-performance computing cluster and the code for the simulations is available at github.com/jolars/strong.SLOPE.simulations.

3.1.1 Penalty Sequence

There is a variety of ways to construct the penalty sequence λ for SLOPE. The OSCAR sequence [4] is a linear sequence, which can be written as

$$\lambda_i = q(p - i) + 1, \qquad i = 1, 2, \ldots, p.$$

Note that we refrain from using the double-parameter parameterization from Bondell and Reich [4] here to facilitate comparisons between the various paths and because we are interested only in the *shape* of the sequence—not its *scale*—assuming that the sequence is scaled so as to yield the all-zero solution at $\lambda^{(1)}$.

Bogdan et al. [3] report two methods for computing the sequence: the *Benjamini–Hochberg* (BH) and *Gaussian* methods. The BH method sets

$$\lambda_i^{\text{BH}} = \Phi^{-1}\left(1 - \frac{qi}{2p}\right), \qquad i = 1, 2, \ldots, p,$$

where Φ^{-1} is the probit function. The Gaussian method is a modification of the BH type, which sets

$$\lambda_1^{\text{G}} = \lambda_1^{\text{BH}}, \qquad \lambda_i^{\text{G}} = \lambda_i^{\text{BH}} \sqrt{1 + \frac{1}{n-i} \sum_{j<i} \left(\lambda_j^{\text{G}}\right)^2} \qquad i = 2, 3, \ldots, p,$$

with the restriction that λ_i^{G} is set to the previous value in the sequence if and when the sequence begins to increase. Note, however, that λ_i^{G} is undefined whenever $i = n$, which occurs exactly once whenever $p \geq n$. Moreover, for q/p small "enough", the sequence will in fact be *non-increasing*, which means that it will effectively reduce to the constant function and, consequently, the standard ℓ_1 penalty. For a given q/p, the value of n required for this to happen corresponds to

$$n \leq \frac{\left(\lambda_1^{\text{BH}} \lambda_2^{\text{BH}}\right)^2}{(\lambda_2^{\text{BH}})^2 - (\lambda_2^{\text{BH}})^2}.$$

If, for instance, we let $p = 100$ and $q = 0.1$, the Gaussian sequence reduces to a constant sequence whenever $n \leq 82$ and, consequently, our rule reduces to the standard strong rule for ℓ_1-regularized problems, which is a topic that has been covered in depth previously; we therefore will not consider the Gaussian sequence any further in this paper.

3.1.2 Regularization Path

To construct the regularization path, we parameterize the sorted ℓ_1 penalty as

$$J(\beta; \lambda, \sigma) = \sigma \sum_{j=1}^{p} |\beta|_{(j)} \lambda_j,$$

with $\sigma^{(1)} > \sigma^{(2)} > \cdots > \sigma^{(l)} > 0$. We pick $\sigma^{(1)}$ corresponding to the point at which the first predictor enters the model, which corresponds to maximizing $\sigma \in \mathbb{R}$ subject to $\operatorname{cumsum}(\nabla f(\mathbf{0})_\downarrow - \sigma\lambda) \preceq 0$, which is given explicitly as

$$\sigma^{(1)} = \max(\operatorname{cumsum}(\nabla f(\mathbf{0})_\downarrow) \oslash \operatorname{cumsum}(\lambda)),$$

where $\oslash$ is the Hadamard (element-wise) division operator. We choose $\sigma^{(l)}$ to be $t\sigma^{(1)}$ with $t = 10^{-2}$ if $n < p$ and 10^{-4} otherwise.

Unless stated otherwise, we employ a regularization path of $l = 100$ λ sequences computed using the BH method from Bogdan et al. [3], but stop this path prematurely if

1. the number of unique coefficient magnitudes exceed the number of observations,
2. the fractional change in deviance from one step to another is less than 10^{-5}, or
3. if the fraction of deviance explained exceeds 0.995.

3.2 Simulated Data

Let $X \in \mathbb{R}^{n\times p}$, $\beta \in \mathbb{R}^{p\times m}$, and $y \in \mathbb{R}^n$. We take

$$y_i = x_i^T\beta + \varepsilon_i, \qquad i = 1, 2, \ldots, n,$$

where ε_i are sampled from independently and identically distributed standard normal variables. X is generated such that each row is sampled independently and identically from a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \Sigma)$. From here on out, we also let k denote the cardinality of the non-zero support set of the true coefficients, that is, $k = \operatorname{card}\{i \in \mathbb{N}^p \mid \beta_i \neq 0\}$.

3.2.1 Efficiency

In this section we let $n = 200$, $p = 5000$, and

$$\Sigma_{ij} = \begin{cases} 1 & i = j, \\ \rho & i \neq j. \end{cases}$$

We begin by studying the strong rule for SLOPE on problems with varying levels of correlation ρ. We take $k = p/4$ and generate β_i for $i = 1, \ldots, k$ from $\mathcal{N}(0, 1)$. We then fit an OLS regression model regularized with the sorted ℓ_1 norm to this data and screen the predictors with the strong rule for SLOPE (Figure 1). Here we set $q = 0.005$ in the construction of the BH sequence (Section 3.1.1).

The size of the screened set is clearly small next to the full set. No violations of the rule were observed in any instance. The presence, however, of strong correlation among the predictors weaken the strong rule at the start of the path.

Next, we consider the effects of various types of penalty sequences on the strong rule. We use the BH and OSCAR methods (see Section 3.1.1) along with a lasso penalty. We now take $n = 200$ and $p = 10000$ and fit for various levels of correlation ρ. We let $k = 10$ and sample β_i for $i = 1, \ldots, k$ from $\{-2, 2\}$. Finally, we let $q = n/(10p)$ in the construction of the BH sequence.

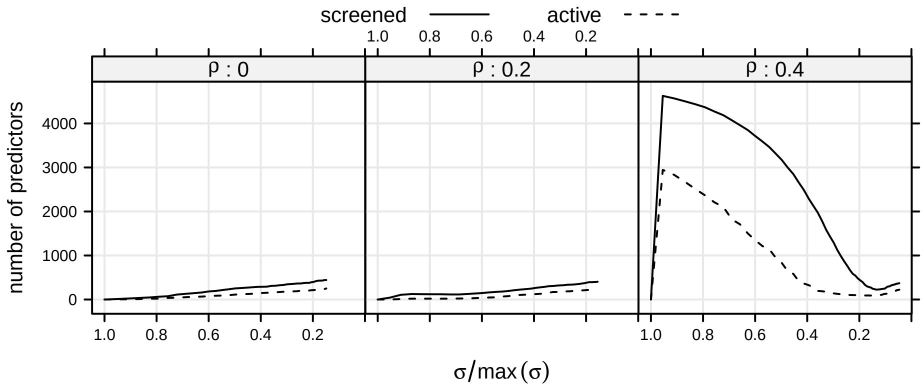

Figure 1: Ordinary Least Squares regression with the SLOPE norm using either the strong or SAFE rule for SLOPE. The predictor matrix $X \in \mathbb{R}^{200\times 5000}$ was generated from a multivariate normal distribution distribution such that all predictors had variance 1 and pairwise correlation with one another of ρ. Please see the text for further information regarding the setup. There were no violations in this example.

The type of sequence affects the efficiency of the rule (Figure 2), which is most effective for the lasso-type sequence and least so for BH. Across the board, the efficiency of the rule appears to deteriorate as the model becomes more and more saturated.

3.2.2 Violations

The usefulness of the screening rule depends on the frequency with which it is violated. To examine this, we generate a number of data sets with $n = 100$, $p \in \{20, 50, 100, 500, 1000\}$, and $\rho = 0.5$. We then fit a full path of 100 λ sequences. (Here we disable the rules for prematurely aborting the path described at the start of this section.) We once again sample the first fourth of the elements of β from $\{-2, 2\}$ and set the rest to zero.

Violations appear to be rare in this setting and occur only for the lower range of p values (Figure 3). For $p = 100$, for instance, we would at an average need to fit 25 paths of regularization sequences for this type of design to encounter a single violation. Which, given that a complete path consists of 100 steps and we expect the warm start after the violation to be a good initialization, can be considered to be a marginal cost.

3.2.3 Performance

In this section, we study the performance of the screening rules for sorted ℓ_1-penalized OLS, logistic, multinomial, and Poisson regression.

We now take $p = 20,000$, $n = 200$, and $k = 20$. To construct X, we let $X_1, X_2, \dots, X_p$ be random variables distributed according to

$$X_1 \sim \mathcal{N}(\mathbf{0}, I), \qquad X_j \sim \mathcal{N}(\rho X_{j-1}, I) \quad \text{for } j = 2, 3, \dots, p,$$

and sample the jth column in X from X_j for $j = 1, 2, \dots, p$.

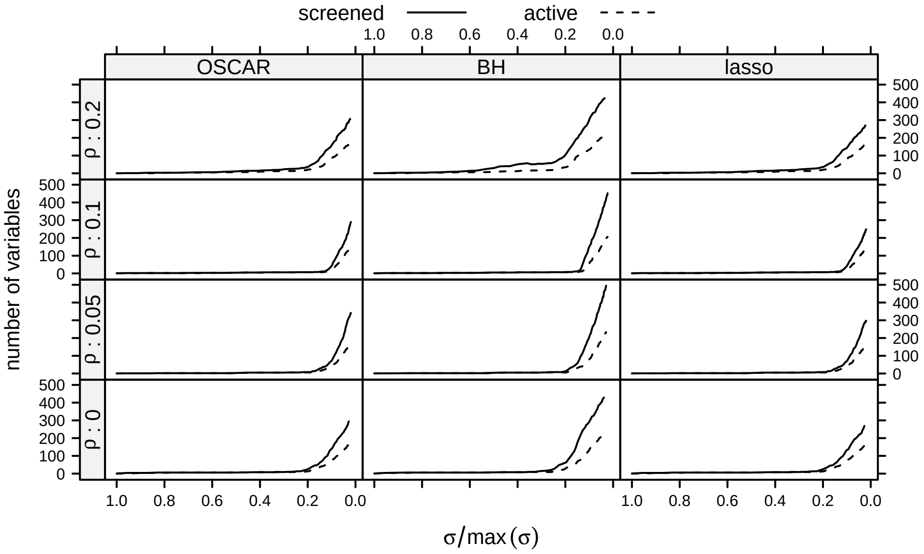

Figure 2: Size of screened set versus active set for sorted ℓ_1-penalized OLS regression with three types of regularization sequences.

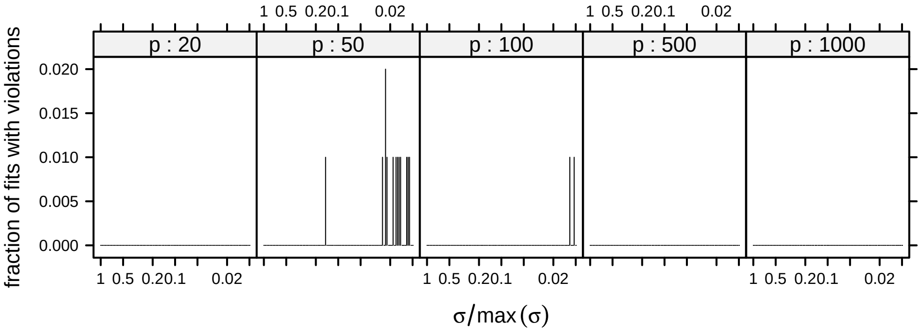

Figure 3: Prevalence of violations for the strong rule when running SLOPE on OLS regression with $n = 100$ and $p \in \{20, 50, 100, 500, 1000\}$ for a full path of 100 λ sequences. Each average was based on 100 repetitions. See the text for full specifications on how the data was generated.

For OLS and logistic regression data we sample the first $k = 20$ elements of β without replacement from $\{1, 2, \ldots, 20\}$. Then we let $y = X\beta + \varepsilon$ for OLS regression and $y = \operatorname{sign}(X\beta + \varepsilon)$ for logistic regression, in both cases taking $\varepsilon \sim \mathcal{N}(\mathbf{0}, 20I)$.

For Poisson regression, we generate β by taking random samples without replacement from $\{\frac{1}{40}, \frac{2}{40}, \ldots, \frac{20}{40}\}$ for its first 20 elements. Then we sample y_i from $\operatorname{Poisson}\big(\exp((X\beta)_i)\big)$ for $i = 1, 2, \ldots, n$,

For multinomial regression, we start by taking $\beta \in \mathbb{R}^{p \times 3}$, initializing all elements to zero. Then, for each row in β we take a random sample from $\{1, 2, \ldots, 20\}$ without replacement and insert it at random into one of the elements of that row. Then we sample y_i randomly from $\operatorname{Categorical}(3, p_i)$ for $i = 1, 2, \ldots, n$, where

$$p_{i,l} = \frac{\exp\big((X\beta)_{i,l}\big)}{\sum_{l=1}^{m} \exp\big((X\beta)_{i,l}\big)}.$$

The benchmarks reveal a strong effect on account of the screening rule through the range of models used (Figure 4), leading to a substantial reduction in run time. As an example, the run time for fitting logistic regression when $\rho = 0.5$ decreases from roughly 70 to 5 seconds when the screening rule is used.

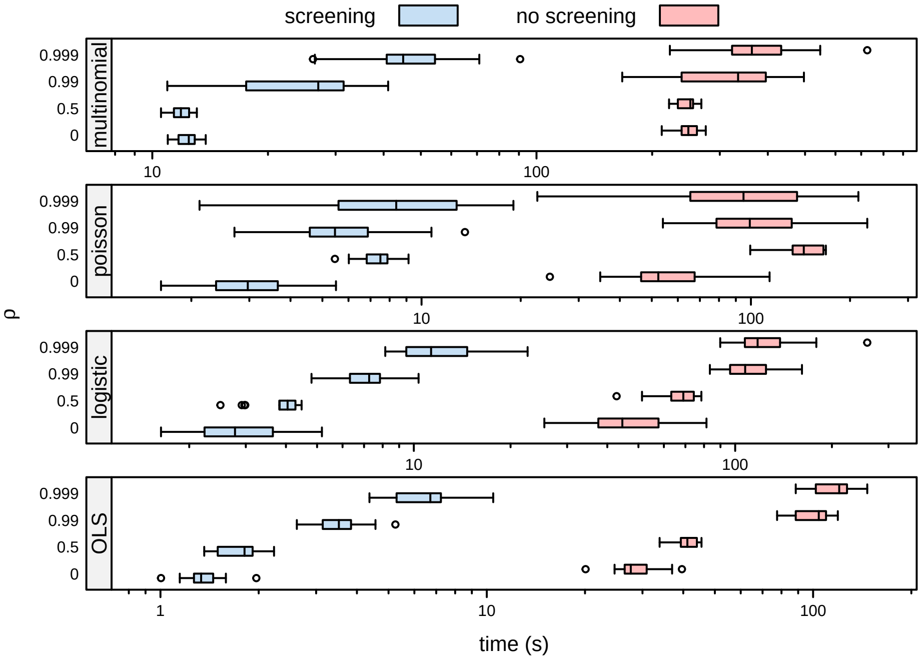

Figure 4: Box plots of wall-clock time for fitting SLOPE with or without the strong screening rule for randomly generated datasets with $p = 20000$, $n = 200$, $k = p/10$, and various levels of correlation ρ. Please see the text for details on how the data was generated.

In Table 1, we see that the relative speed-ups of using the rule (compared to not using it) are sizeable, first most of the cases surpassing 10 times. The improvement is most noticeable for OLS.

Table 1: Relative speed-up of using the screening rule with $n = 200$, $k = 20$, and $p = 20000$ for OLS, logistic, Poisson, and multinomial regression models. See the text for details regarding the generation of data.

	ρ			
model	0	0.5	0.99	0.999
OLS	21.3	23.5	28.5	17.62
logistic	16.2	17.3	15.2	10.46
poisson	18.4	19.6	18.2	10.73
multinomial	20.1	20.7	12.8	7.97

The utility of the rule depends directly on the relationship between n and p. To probe this property, we fit sorted ℓ_1-penalized OLS regression to the simple case of orthonormal predictors and standard normal error term, using $n = 1000$, varying p, $k = p/10$, and nonzero β elements sampled uniformly from $\{-2, 2\}$. It is both clear that the screening rule imposes no penalty on run time at any point and starts to improve performance at approximately $p = 2n$ (Figure 5).

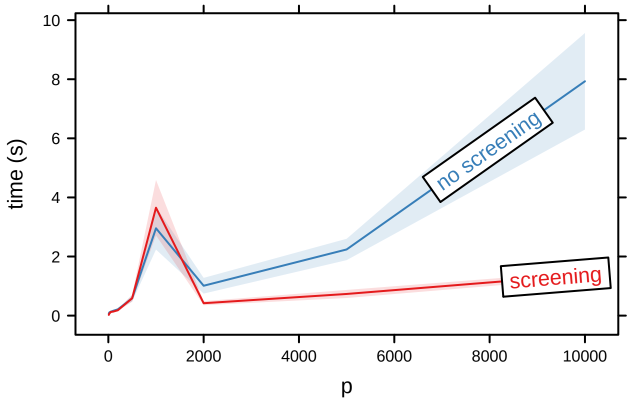

Figure 5: Time taken to fit a path of lambda sequences for $n = 1000$ and varying values of p. Shaded bands indicate 95% confidence intervals. The results are based on 100 repetitions of fitting ordinary least squares regression using a predictor matrix generated with identically and independently distributed columns.

We finish this section with an examination of two types of algorithms outlined in Section 2.2.4: the strong set (Algorithm 3) and previous set algorithm (Algorithm 4). In Figure 1 we observed that the strong rule is excessively conservative when correlation is high among predictors, which suggests that the previous set algorithm might yield an improvement over the strong set algorithm.

In order to examine this, we conduct a simulation in which we vary the strength of correlation between predictors as well as the parameter q in the construction of the BH regularization sequence. Motivation for varying the latter comes from the relationship between coefficient clustering and the intervals in the regularization sequence—higher values of q cause larger gaps in the sequence, which in turn leads to more clustering among predictors. This clustering, in turn, is strongest at the start of the path when regularization is strong.

For large enough q and ρ, this behavior in fact occasionally causes almost all predictors to enter the model at the second step on the path. As an example, using when $\rho = 0.6$ and fitting with $q = 10^{-2}$ and 10^{-4} leads to 2215 and 8 coefficients respectively at the second step for a single example.

Here, we let $n = 200$, $p = 5000$, $k = 50$, and $\rho \in \{0, 0.1, 0.2, \ldots, 0.8\}$. The data generation process corresponds to the setup at the start of this section and the covariance structure of X is equal to that in Section 3.2.1. We sample the non-zero entries in β independently from a random variable $U \sim \mathcal{N}(0, 1)$.

The two algorithms perform similarly for $\rho \leq 0.6$ (Figure 6). For larger ρ, the previous set strategy evidently outperforms the strong set strategy. This result is not surprising: consider Figure 1, for instance, where the behavior of the regularization path under strong correlation makes the previous set strategy much more effective than the strong set strategy for all but the second and last few steps on the path.

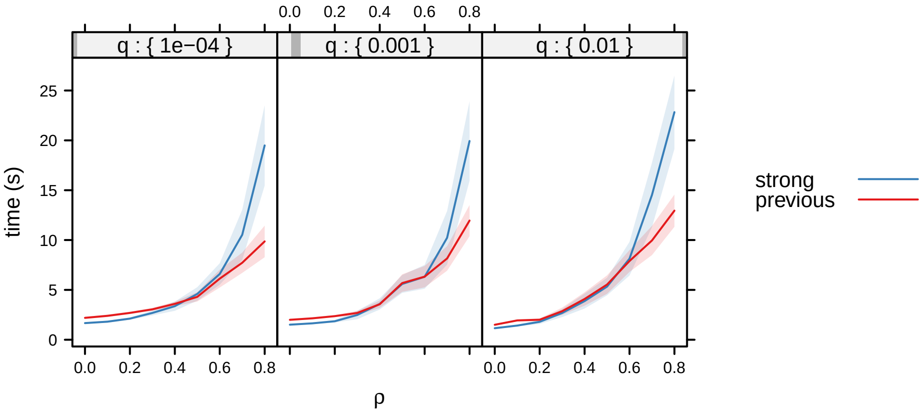

Figure 6: Time taken to fit a regularization path of SLOPE for OLS using either the strong or previous set algorithm, with $n = 200$, $p = 5000$, $k = 50$, and pairwise correlation between predictors of ρ. The data are based on 100 repetitions.

3.3 Real Data

3.3.1 Efficiency and Violations

We examine efficiency and violations for four real data sets: *arcene*, *dorothea*, *gisette*, and *golub*, which are the same data sets that were tested in the original strong rule article. The first three originate from Guyon et al. [15] and were originally collected from the UCI (University of California Irvine) Machine Learning Repository [9], whereas the last data set, *golub*, was originally published in Golub et al. [14]. All of the data sets were collected from http://statweb.stanford.edu/~tibs/strong/realdata/.

All of the datasets feature a response vector $y \in \{0, 1\}$. We fit both OLS and logistic regression models to the data sets.

We first note that there were no violations in any of the fits. The efficiency is moreover excellent for each of the data sets (Figure 7), with the size of the screened set of predictors ranging from roughly 1.5–4 times the size of the active set (Table 2).

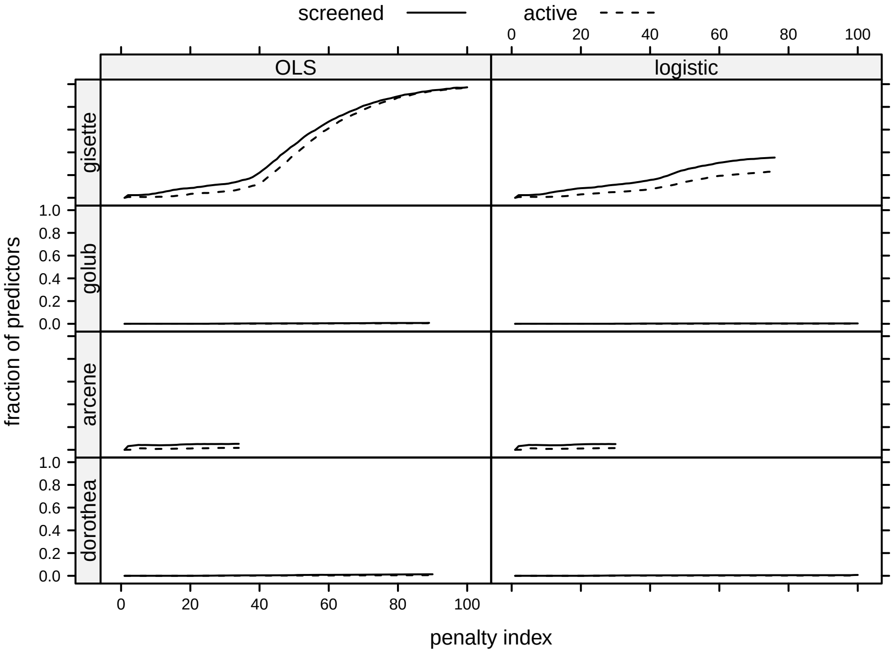

Figure 7: Proportion of predictors included in the model by the strong screening rule as a proportion of the total number of active predictors in the model for a path of λ sequences. There were no violations of the screening rule in any of the examples.

3.3.2 Performance

In this section, we introduce three new data sets: *cpusmall*, *physician*, and *zipcode*. *cpusmall* was collected from csie.ntu.edu.tw/~cjlin/libsvmtools/datasets [7] and was originally published by The University of Toronto [28]. *physician* was collected from jstatsoft.org/article/view/v027i08 [34] and is credited to Deb and Trivedi [8]. *zipcode*, meanwhile, was collected from web.stanford.edu/~hastie/ElemStatLearn [17] and is credited to Le Cun et al. [23].

In Table 3, we summarize the results from fitting sorted ℓ_1-regularized OLS, logistic, Poisson, and multinomial regression to the four data sets. Once again, we see that the screening rule improves performance in the high-dimensional regime and presents no noticeable drawback even when $n \gg p$.

Table 2: Average number of predictors left after screening (screened) and active variables in the model (active) for four datasets modelled using either sorted ℓ_1-penalized OLS or logistic regression. There were no violations in any of the examples.

dataset	n	p	model	screened	active
arcene	100	9920	OLS	441.1	112.88
			logistic	426.6	102.57
dorothea	800	88119	OLS	499.3	191.72
			logistic	317.3	122.10
gisette	6000	4955	OLS	2376.1	2152.33
			logistic	879.1	489.03
golub	38	7129	OLS	25.2	14.93
			logistic	16.0	8.04

Table 3: Benchmarks measuring wall-clock time for four data sets fit with different models using either the strong screening rule or non rule.

				time (s)	
dataset	model	n	p	no screening	screening
cpusmall	OLS	8192	12	8.11	8.444
golub	logistic	38	7129	10.24	0.357
physician	poisson	4406	25	36.04	36.077
zipcode	multinomial	200	256	14.63	12.439

4 Software

The latest version of the R-package SLOPE [22], which we have used in our simulations for this paper, features the screening rules as well as the two algorithms examined in this paper.

5 Conclusions

In this paper, we have developed a heuristic predictor screening rule for SLOPE and shown that it is a generalization of the strong rule for the lasso. We have demonstrated that it offers dramatic improvements in the $p \gg n$ regime, often reducing the time required to fit the full regularization path for SLOPE by orders of magnitude, as well as imposing little-to-no cost when $p \lessapprox n$. Jointly with the publication of this paper, we have also made an efficient implementation of the screening rule available in the R package SLOPE [22].

Analogous with the strong rule for the lasso, the strong rule for SLOPE turns out to be excessively conservative when predictors in the design matrix are heavily correlated. For the lasso, this problem can be remedied with the use of previous-set strategies such as the one implemented in glmnet [13], which we have also examined in this paper. The benefits of that strategy, however, are evidently limited here due to the clustering behavior that SLOPE exhibits: large portions of the total number of predictors often enter the model in a few clusters when regularization is strong (at the start of the path); optimization strategies targeting this setting would be welcome.

Screening rules have been of crucial importance to the field of sparse statistical methods, bringing high-dimensional problems within reach of even modest computational resources. As far as we know, however, this is the first publication to develop screening rules for SLOPE. And, given the widepread interest in screening rules for the lasso and attractive properties of SLOPE, we believe there is much to be gained from researching this problem further.

Acknowledgments

We would like to thank Patrick Tardivel (Mathematics Institute of Wrocław, Wrocław University) and Yvette Baurne (Department of Statistics, Lund University) for their insightful comments on the paper.

References

[1] A. Beck and M. Teboulle. “A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems”. In: *SIAM Journal on Imaging Sciences* 2.1 (Jan. 1, 2009), pp. 183–202. DOI: 10.1137/080716542.

[2] Malgorzata Bogdan et al. “Statistical Estimation and Testing via the Sorted L1 Norm”. In: (Oct. 29, 2013). arXiv: 1310.1969 [math, stat].

[3] Małgorzata Bogdan et al. “SLOPE - Adaptive Variable Selection via Convex Optimization”. In: *The annals of applied statistics* 9.3 (2015), pp. 1103–1140. ISSN: 1932-6157. DOI: 10.1214/15-AOAS842. pmid: 26709357.

[4] Howard D. Bondell and Brian J. Reich. “Simultaneous Regression Shrinkage, Variable Selection, and Supervised Clustering of Predictors with OSCAR”. In: *Biometrics* 64.1 (2008), pp. 115–123. ISSN: 0006-341X. DOI: 10.1111/j.1541-0420.2007.00843.x.

[5] Damian Brzyski et al. “Group SLOPE – Adaptive Selection of Groups of Predictors”. In: *Journal of the American Statistical Association* (Jan. 15, 2018), pp. 1–15. ISSN: 0162-1459. DOI: 10/gfrd93.

[6] Zhiqi Bu et al. “Algorithmic Analysis and Statistical Estimation of SLOPE via Approximate Message Passing”. In: *Advances in Neural Information Processing Systems 32*. NeurIPS. Ed. by H. Wallach et al. Vancouver: Curran Associates, Inc., 2019, pp. 9361–9371.

[7] Chih-chung Chang and Chih-jen Lin. “LIBSVM: A Library for Support Vector Machines”. In: *ACM Transactions on Intelligent Systems and Technology* 2.3 (May 2011), 27:1–27:27. DOI: 10.1145/1961189.1961199.

[8] Partha Deb and Pravin K. Trivedi. “Demand for Medical Care by the Elderly: A Finite Mixture Approach”. In: *Journal of Applied Econometrics* 12.3 (1997), pp. 313–336. ISSN: 0883-7252.

[9] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2019. URL: http://archive.ics.uci.edu/ml.

[10] Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. *Safe Feature Elimination in Sparse Supervised Learning*. UCB/EECS-2010-126. Berkeley: EECS Department, University of California, Sept. 21, 2010.

[11] Jianqing Fan and Jinchi Lv. “Sure Independence Screening for Ultrahigh Dimensional Feature Space”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.5 (2008), pp. 849–911. ISSN: 1467-9868. DOI: 10.1111/j.1467-9868.2008.00674.x.

[12] Mario Figueiredo and Robert Nowak. “Ordered Weighted L1 Regularized Regression with Strongly Correlated Covariates: Theoretical Aspects”. In: *Artificial Intelligence and Statistics*. Artificial Intelligence and Statistics. May 2, 2016, pp. 930–938.

[13] Jerome Friedman et al. “Pathwise Coordinate Optimization”. In: *The Annals of Applied Statistics* 1.2 (Dec. 2007), pp. 302–332. ISSN: 1932-6157, 1941-7330. DOI: 10/d88g8c.

[14] T. R. Golub et al. “Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring”. In: *Science (New York, N.Y.)* 286.5439 (Oct. 15, 1999), pp. 531–537. ISSN: 0036-8075. DOI: 10.1126/science.286.5439.531. pmid: 10521349.

[15] Isabelle Guyon et al. “Result Analysis of the NIPS 2003 Feature Selection Challenge”. In: *Advances in Neural Information Processing Systems 17*. Advances in Neural Information Processing Systems 17. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. Vancouver, Canada: MIT Press, May 2005, pp. 545–552. ISBN: 978-0-262-19534-8.

[16] G.H. Hardy, J.E. Littlewood, and G. Pólya. *Inequalities, Cambridge Mathematical Library*. 2nd ed. Cambridge University Press, 1952. ISBN: 0-521-05206-8.

[17] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. 2nd ed. Springer Series in Statistics. New York: Springer-Verlag, 2009. ISBN: 978-0-387-84857-0.

[18] J. Jia and B. Yu. “On Model Selection Consistency of the Elastic Net When p >> n”. In: *Statistica Sinica* 20.2 (2010), pp. 595–611.

[19] W. Jiang et al. “Adaptive Bayesian SLOPE–High-dimensional Model Selection with Missing Values”. In: (Nov. 6, 2019). arXiv: 1909.06631 [stat].

[20] Tyler B Johnson and Carlos Guestrin. “Blitz: A Principled Meta-Algorithm for Scaling Sparse Optimization”. In: *Proceedings of the 32nd International Conference on Machine Learning*. International Conference on Machine Learning. Vol. 37. Lille, France: JMLR: W&CP, 2015, p. 9.

[21] Philipp Kremer et al. *Sparse Index Clones via the Sorted L1-Norm*. en. SSRN Scholarly Paper ID 3412061. Rochester, NY: Social Science Research Network, June 2019, pp. 1–30. DOI: 10.2139/ssrn.3412061.

[22] Johan Larsson et al. *SLOPE: Sorted L1 Penalized Estimation*. Version 0.2.1. Apr. 16, 2020.

[23] Yann Le Cun et al. “Handwritten Digit Recognition with a Back-Propagation Network”. In: *Advances in Neural Information Processing Systems 2 (NIPS 1989)*. Neural Information Processing Systems 1989. Vol. 2. IEEE. Denver, Colorado, USA: Morgan Kaufman, Nov. 27–30, 1989, pp. 35–40. ISBN: 1-55860-100-7.

[24] Mathurin Massias, Alexandre Gramfort, and Joseph Salmon. “From Safe Screening Rules to Working Sets for Faster Lasso-Type Solvers”. In: (Mar. 21, 2017). arXiv: 1703.07285 [cs, math, stat].

[25] Eugene Ndiaye et al. “Gap Safe Screening Rules for Sparsity Enforcing Penalties”. In: *Journal of Machine Learning Research* 18.128 (2017), pp. 1–33.

[26] Ulrike Schneider and Patrick Tardivel. “The Geometry of Uniqueness and Model Selection of Penalized Estimators including SLOPE, LASSO, and Basis Pursuit”. In: (Apr. 20, 2020). arXiv: 2004.09106 [math, stat].

[27] Weijie Su and Emmanuel Candès. “SLOPE Is Adaptive to Unknown Sparsity and Asymptotically Minimax”. In: *The Annals of Statistics* 44.3 (June 2016), pp. 1038–1068. ISSN: 0090-5364. DOI: 10.1214/15-AOS1397.

[28] The University of Toronto. *Delve Datasets*. May 27, 1997. URL: http://www.cs.toronto.edu/~delve/data/datasets.html (visited on 04/06/2020).

[29] Robert Tibshirani. “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 0035-9246. JSTOR: 2346178.

[30] Robert Tibshirani et al. “Strong Rules for Discarding Predictors in Lasso-Type Problems”. In: *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 74.2 (Mar. 2012), pp. 245–266. ISSN: 1369-7412. DOI: 10/c4bb85.

[31] Jie Wang, Peter Wonka, and Jieping Ye. "Lasso Screening Rules via Dual Polytope Projection". In: *The Journal of Machine Learning Research* 16.1 (Jan. 1, 2015), pp. 1063–1101. ISSN: 1532-4435.

[32] Zhen James Xiang and Peter J. Ramadge. "Fast Lasso Screening Tests Based on Correlations". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Mar. 2012, pp. 2137–2140. DOI: 10.1109/ICASSP.2012.6288334.

[33] Ming Yuan and Yi Lin. "On the Non-Negative Garrotte Estimator". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 69.2 (2007), pp. 143–161. ISSN: 1369-7412.

[34] Achim Zeileis, Christian Kleiber, and Simon Jackman. "Regression Models for Count Data in R". In: *Journal of Statistical Software* 27.1 (1 July 29, 2008), pp. 1–25. ISSN: 1548-7660. DOI: 10.18637/jss.v027.i08.

[35] Xiangrong Zeng and Mário A. T. Figueiredo. "Decreasing Weighted Sorted L1 Regularization". In: *IEEE Signal Processing Letters* 21.10 (Oct. 2014), pp. 1240–1244. ISSN: 1070-9908, 1558-2361. DOI: 10.1109/LSP.2014.2331977.

[36] Xiangrong Zeng and Mário A. T. Figueiredo. "The Atomic Norm Formulation of OSCAR Regularization with Application to the Frank-Wolfe Algorithm". In: *2014 22nd European Signal Processing Conference (EUSIPCO)*. 2014 22nd European Signal Processing Conference (EUSIPCO). Lisbon, Portugal: IEEE, Sept. 2014, pp. 780–784. ISBN: 978-0-9928626-1-9.

[37] Xiangrong Zeng and Mário A. T. Figueiredo. "The Ordered Weighted L1 Norm: Atomic Formulation, Projections, and Algorithms". In: (Apr. 10, 2015). arXiv: 1409.4271 `[cs, math]`.

[38] Yaohui Zeng, Tianbao Yang, and Patrick Breheny. "Efficient Feature Screening for Lasso-Type Problems via Hybrid Safe-Strong Rules". In: (Nov. 21, 2017). arXiv: 1704.08742 `[stat]`.