Solar: stable and ultrafast variable selection for high dimensional and large-scale data

Ning Xu[*]

School of Economics, University of Sydney, Australia

Timothy C.G. Fisher[**]

School of Economics, University of Sydney, Australia

Jian Hong

School of Economics, University of Sydney, Australia

Abstract

We propose a new algorithm for variable selection in high-dimensional, large-scale data, called *subsample-ordered least-angle regression (solar)* and a coordinate-descent generalization *(solar-cd)*. By averaging and re-ordering lasso paths using the L_0 norm, solar can easily be modified to replace many lasso variants. Solar alleviates several high-dimensional issues with lasso, such as the strong/safe rules, variable screening, and subsampling variable selection. Using examples, simulations, and real-world data, we illustrate: (i) with the same computation load, solar yields substantial improvements over lasso in terms of the sparsity (37-64% reduction in false variable selection), stability, and accuracy of variable selection; (ii) compared with the lasso strong/safe rule and variable screening, solar largely rectifies incorrect variable purging and sparsity loss with complicated dependence structures and harsh settings of the irrepresentable condition; (iii) solar supplemented with the hold-out average test (an adaptation of data-splitting hypothesis testing) dramatically improves the efficiency and accuracy of post-selection inference; (iv) replacing lasso with solar in subsampling variable selection (e.g., the bootstrap lasso or stability selection) produces a novel *multi-layer* variable ranking scheme, resulting in improvements in selection sparsity, ranking accuracy, and huge computation load reductions (a saving of at least 96% in runtime, substantially beyond the theoretical maximum speedup for parallelizing the lasso-type algorithm.) Using a Python interface and an Intel MKL Fortran/C++ compiler, we provide a parallel computing package for solar (`solarpy`) in the supplementary file and at the dedicated Github page.

Keywords: Sparsity, complicated dependence structure, irrepresentable condition, lasso rule, bolasso, subsampling variable selection, variable screening.

1. Introduction

In the last decade, lasso-related algorithms have been widely applied to high dimensional, large-scale applications (Efron et al., 2004; Friedman et al., 2007, 2010). Various improvements have been

[*]Principal corresponding author.
[**]Corresponding author.
Email addresses: `n.xu@sydney.edu.au` (Ning Xu), `tim.fisher@sydney.edu.au` (Timothy C.G. Fisher), `jian.hong@sydney.edu.au` (Jian Hong)

proposed for lasso to address known issues of computation speed, selection sparsity, stability, and accuracy (Weisberg, 2004; Lim and Yu, 2016). Subsampling variable selection (e.g., Bach (2008); Meinshausen and Bühlmann (2010)) has emerged as one of the most effective improvements to the variable-selection accuracy and stability of lasso. Given that optimization of the lasso-type tuning parameter is often combined with cross validation (CV), computing CV-lasso on each bootstrap subsample sharply increases the computation load, making any subsampling-based lasso variants less attractive in large-scale applications such as computer vision and natural language processing (where p may be in the millions and data in the GBs) (Xu et al., 2012). Thus, in a world of ever-expanding data dimensionality, it is critical that improvements in variable selection are accompanied with significant computation load reductions.

Post-lasso selection rules (e.g., the 'safe rule' (Ghaoui et al., 2010) and the 'strong rule' (Tibshirani et al., 2012)) are capable of improving lasso without exponentially increasing the computation runtime. However, the rules may falsely purge informative variables or propose repeated modifications (Wang et al., 2014; Zeng et al., 2017). Alternatively, Wasserman and Roeder (2009), Meinshausen et al. (2009) and Barber and Candès (2019) propose data splitting to conduct classical hypothesis tests after subsample selection. The original data are split into two: one part for variable selection, the other for testing. However, to improve test power, data-splitting must be repeated on each bootstrap sample, raising similar computational concerns as subsampling variable selection (Bach, 2008; Meinshausen and Bühlmann, 2010). Romano and DiCiccio (2019) also conclude that because data splitting reserves a portion of the data for variable selection, it reduces the degrees of freedom for testing on the remaining data, which may be a problem when the original sample size is limited. Lastly, Fan and Lv (2008) propose the variable screening algorithm, which ranks the absolute values of unconditional correlations between each covariate and the response variable, selecting only the top-ranked variables. However, Fan and Lv (2008) and Barut et al. (2016) (and Section 3.3 below) show that variable screening methods also suffer from invalid variable selection and purging when the dependence structure is complicated.

1.1. Main results

To address these issues, we propose a new algorithm for variable selection in high-dimensional, large-scale data, called *subsample-ordered least-angle regression* (*solar*), and a coordinate-descent generalization (*solar-cd*). Because solar is based on averaging and re-ordering lasso paths via the L_0 norm, it is easily adaptable to many lasso variants. Solar alleviates the high-dimensional issues with lasso, such as the strong/safe rules, variable screening, and subsampling variable selection while having the same computation load as lasso.

Using simulations, examples, and real-world data, we demonstrate the following advantages of solar: (i) with the same computation load, solar yields substantial improvements over lasso in terms of the sparsity (37-64% reduction in false variable selection), stability, and accuracy of variable selection; (ii) compared with the lasso strong/safe rule and variable screening, solar largely rectifies incorrect variable purging and sparsity loss in the presence of complicated dependence structures and harsh settings of the irrepresentable condition; (iii) solar supplemented with the hold-out average test (an adaptation of data-splitting hypothesis testing) substantially improves the efficiency and accuracy of post-selection inference; (iv) replacing lasso with solar in subsampling variable selection (e.g., the bootstrap lasso or stability selection) produces a novel *multi-layer* variable ranking scheme that brings improvements in selection sparsity, ranking accuracy, and computation resources (a saving of at least 96% in runtime, substantially beyond the theoretical maximum speedup for parallelizing the lasso-type algorithm.). Using a Python interface and an Intel MKL Fortran/C++ compiler, we provide a parallel computing package for solar (solarpy) in the supplementary file and at the dedicated Github page.

The paper is organized as follows. In section 2, we introduce solar and its coordinate descent generalization. In section 3 and 4, we use examples and simulations to demonstrate solar's improvements over lasso, the strong/safe rules, variable screening, forward regression and the lasso-related, subsampling variable selection. In section 5, we use real-world data that exhibits complicated dependency structures to show that the improvements from solar are feasible, while lasso and elastic net lose sparsity.

2. Solar algorithm

The solar improvements over lasso and lasso-related subsampling variable selection are due to a reallocation of computation. Lasso and its variants allocate most of the computation to optimizing the shrinkage parameter t and λ using cross-validation. By contrast, with the same computation load, solar allocates computation to stabilizing the solution path, which is key to the variable selection procedure. Zhang (2009, Theorem 2) implies that the earlier a variable enters the solution path, the more likely it is informative. As a result, a stable and accurate ordering of variables in the solution path may help identify informative variables. Since we focus on variable selection accuracy, the only relevant feature of the regression coefficients in the solution path is whether $\beta_i = 0$ at each stage. Thus, for least-angle regression and pathwise cooridinate descent, we reparametrize the lasso path via the L_0 norm.

Definition 2.1 (L_0 solution path)**.** Define the L_0 **solution path** on (Y, X) to be the order that least-angle regression includes variables across all stages. For example, if the least angle regression includes $\mathbf{x}_3$ at stage 1, $\mathbf{x}_2$ at stage 2 and $\mathbf{x}_1$ at stage 3, the corresponding L_0 path is the ordered set $\{\{\mathbf{x}_3\}, \{\mathbf{x}_3, \mathbf{x}_2\}, \{\mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1\}\}$.

2.1. Solar solved by least angle regression

The solution path is the foundation of a L_p-penalized regression. To stabilize the regression performance in high-dimensional spaces, we first reduce the sensitivity of the solution path when $p > n$. The *average L_0 solution path estimation* algorithm (summarized in Algorithm 1 and illustrated in Figure 1) accomplishes this task by estimating the *average stage $\mathbf{x}_i$ enters the solution path* of least-angle regression.

Algorithm 1: average L_0 path estimation via least angle regression

```
  input : $(Y, X)$.
1 generate $K$ subsamples $\left\{\left(Y^k, X^k\right)\right\}_{k=1}^{K}$ by randomly remove $1/K$ of observations in $(Y, X)$;
2 set $\widetilde{p} = \min\{n_{\text{sub}}, p\}$;
3 for $k := 1$ to $K$, stepsize $= 1$ do
4     run an unrestricted least-angle regression on $\left(Y^k, X^k\right)$ and record the order of variable inclusion at
        each stage;
5     define $\widehat{q}^k = \mathbf{0} \in \mathbb{R}^p$;
6     $\forall i, l \in \mathbf{N}^+$, if $\mathbf{x}_i$ is included at stage $l$ and excluded at $l-1$, set $\widehat{q}_i^k = (\widetilde{p} + 1 - l)/\widetilde{p}$, where $\widehat{q}_i^k$ is the
        $i^{\text{th}}$ entry of $\widehat{q}^k$;
7 end
8 $\widehat{q} := \frac{1}{K}\sum_{k=1}^{K}\widehat{q}^k$;
9 return $\widehat{q}$
```

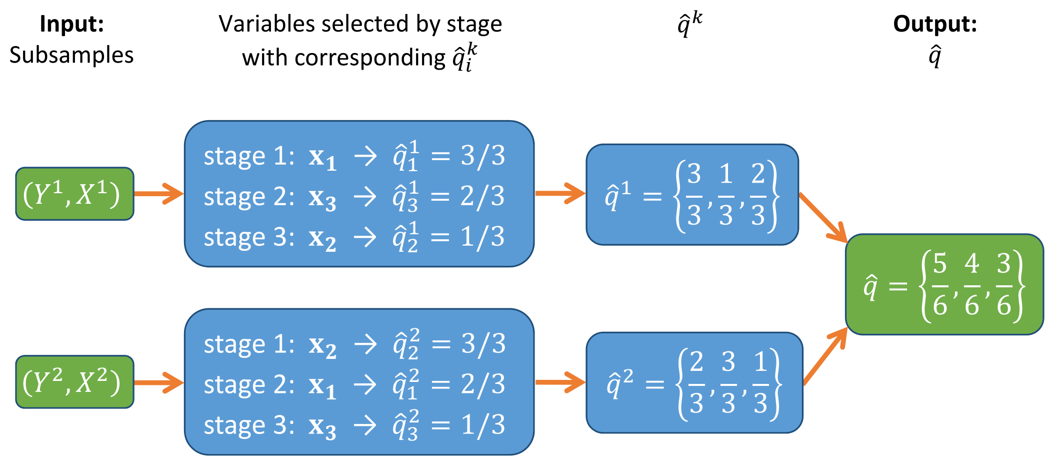

Input:
Subsamples
Variables selected by stage with corresponding $\hat{q}_i^k$
$\hat{q}^k$
Output:
$\hat{q}$
(Y^1, X^1)
stage 1: $\mathbf{x}_1 \rightarrow \hat{q}_1^1 = 3/3$
stage 2: $\mathbf{x}_3 \rightarrow \hat{q}_3^1 = 2/3$
stage 3: $\mathbf{x}_2 \rightarrow \hat{q}_2^1 = 1/3$
$\hat{q}^1 = \left\{\frac{3}{3}, \frac{1}{3}, \frac{2}{3}\right\}$
(Y^2, X^2)
stage 1: $\mathbf{x}_2 \rightarrow \hat{q}_2^2 = 3/3$
stage 2: $\mathbf{x}_1 \rightarrow \hat{q}_1^2 = 2/3$
stage 3: $\mathbf{x}_3 \rightarrow \hat{q}_3^2 = 1/3$
$\hat{q}^2 = \left\{\frac{2}{3}, \frac{3}{3}, \frac{1}{3}\right\}$
$\hat{q} = \left\{\frac{5}{6}, \frac{4}{6}, \frac{3}{6}\right\}$

Figure 1: Computation of $\widehat{q}$ on 2 subsamples where $\{\mathbf{x}_1, \mathbf{x}_2\}$ are informative and $\mathbf{x}_3$ redundant.

After subsampling, the second part of Algorithm 1 (lines 4-6) computes $\widehat{q}^k$, which summarizes the order that least angle regression includes each $\mathbf{x}_i$ across all stages (as in the Figure 1 example). The unrestricted least-angle regression only ranks variables by the stage $\mathbf{x}_i$ enters the solution path. As shown in line 6 of Algorithm 1 and Figure 1, variables included at earlier stages have larger corresponding $\widehat{q}_i^k$ values: the first variable included is assigned 1, the last is assigned $1/\widetilde{p}$ and the dropped variables are assigned 0 (which occurs only when $p > n$). Thus, by ranking (decreasingly) the $\mathbf{x}_i$ according to their corresponding $\widehat{q}_i^k$ values, we obtain the L_0 solution path. The Zhang (2009, Theorem 2) result implies that the variables with the largest $\widehat{q}_i^k$ values, on average, are more likely to be informative variables. The $\widehat{q}_i^k$ may be sensitive in high-dimensional spaces to multicollinearity, sampling randomness and noise. A consequence is that a redundant variable may be included at an early stage in some subsample $\left(Y^k, X^k\right)$. Hence, Algorithm 1 reduces the impact of sensitivity in the $\widehat{q}_i^k$ values by computing $\widehat{q} := \frac{1}{K}\sum_{k=1}^{K}\widehat{q}^k$ and ranking the $\mathbf{x}_i$ (decreasingly) based on the corresponding value of $\widehat{q}_i$ (the i^{th} entry of $\widehat{q}$), to get the average L_0 solution path. The average L_0 solution path is formally defined as follows.

Definition 2.2 (average L_0 solution path)**.** Define the **average L_0 solution path** of least-angle regression on $\left\{\left(Y^k, X^k\right)\right\}_{k=1}^{K}$ to be the decreasing rank order of variables based on their corresponding $\widehat{q}_i$ values. For example, in Figure 1, the $\widehat{q}_i$ values for $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are, respectively, $\widehat{q}_1 = 5/6$, $\widehat{q}_2 = 4/6$ and $\widehat{q}_3 = 3/6$. As a result, the average L_0 solution path can be represented as an ordered set $\left\{\left\{\mathbf{x}_1\right\}, \left\{\mathbf{x}_1, \mathbf{x}_2\right\}, \left\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\right\}\right\}$.

Algorithm 2: Subsample-ordered least-angle regression

```
1 Randomly select 20% of the sample points as the validation set; denote the remaining points as the
    training set;
2 Estimate $\widehat{q}$ using Algorithm 1 on the training set and compute $Q(c) = \{\mathbf{x}_j \mid \widehat{q}_j \geqslant c, \forall j\}$ for all
    $c \in \{1, 0.98, \ldots, 0.02, 0\}$.
3 Run an OLS regression of each $Q(c)$ to $Y$ on the training set and find $c^*$, the value of $c$ that minimizes
    the validation error;
4 Compute the OLS coefficients of $Q(c^*)$ to $Y$ on the whole sample.
```

Built on Algorithm 1, the solar algorithm is summarized in Algorithm 2. Instead of the equiangular, partial-correlation search in least-angle regression, variables are included into forward regression according to their rank order in the average L_0 solution path, represented by $\{Q(c)|c = 1, 0.98, \ldots, 0\}$ in Algorithm 2. We use $\widehat{q}$ from Algorithm 1 to generate a list of variables $Q\left(c\right) = \{\mathbf{x}_j \mid \widehat{q}_j \geqslant c, \forall j \leqslant p\}$. For any $c_1 > c_2$, $Q\left(c_1\right) \subset Q\left(c_2\right)$, implying a sequence of nested sets $\{Q(c)|c = 1, 0.98, \ldots, 0\}$. Each c denotes a stage of forward or least-angle regression. For a given value of c, $Q(c)$ denotes the set of variables with $\left\|\beta_i\right\|_0 = 1$ on average and $Q(c) - Q(c - 0.02)$ is the set of variables with $\left\|\beta_i\right\|_0$ just turning to 1 at c. Therefore, $\{Q(c)|c = 1, 0.98, \ldots, 0\}$ is the average L_0 solution path of Definition 2.2. Variables that are more likely to be informative are included in $Q(c)$ with larger c values and will be selected first by the solar algorithm.

2.2. Solar solved by coordinate descent

The solar algorithm can easily be generalized to use coordinate descent. Both least-angle regression and coordinate descent generate a solution path for lasso, parametrized with β_i on the vertical axis and tuning parameter (t or λ) on the horizontal axis. Thus, to reprogram solar to use coordinate descent, we simply replace Algorithm 1 with Algorithm 3, which records the order of variable activation along the coordinate descent solution path.

Algorithm 3: average L_0 path estimation via coordinate descent

```
   input : $(Y, X)$.
 1 generate $K$ subsamples $\left\{\left(Y^k, X^k\right)\right\}_{k=1}^K$ by randomly remove $1/K$ of observations in $(Y, X)$;
 2 set $\widetilde{p} = \min\left\{n_{\text{sub}}, p\right\}$ ;
 3 for $k := 1$ to $K$, stepsize $= 1$ do
 4     denote $\lambda_s$ as the $\lambda$ value that coordinate descent lasso includes $s$ variables, $\forall s \in [0, \widetilde{p}]$;
 5     run a pathwise coordinate descent for lasso on $\left(Y^k, X^k\right)$, $\forall \lambda \in \{\lambda_0, \lambda_1, \ldots, \lambda_{\widetilde{p}},\}$
 6     record the order of variable inclusion at each $\lambda \in \{\lambda_0, \lambda_1, \ldots, \lambda_{\widetilde{p}},\}$;
 7     define $\widehat{q}^k = \mathbf{0} \in \mathbb{R}^p$;
 8     $\forall i, s \in \mathbf{N}^+$, if $\mathbf{x}_i$ is included at $\lambda = \lambda_s$ and excluded at $\lambda_{s-1}$, set $\widehat{q}_i^k = (\widetilde{p} + 1 - s)/\widetilde{p}$, where $\widehat{q}_i^k$ is the
        $i^{\text{th}}$ entry of $\widehat{q}^k$;
 9 end
10 $\widehat{q} := \frac{1}{K}\sum_{k=1}^K \widehat{q}^k$;
11 return $\widehat{q}$
```

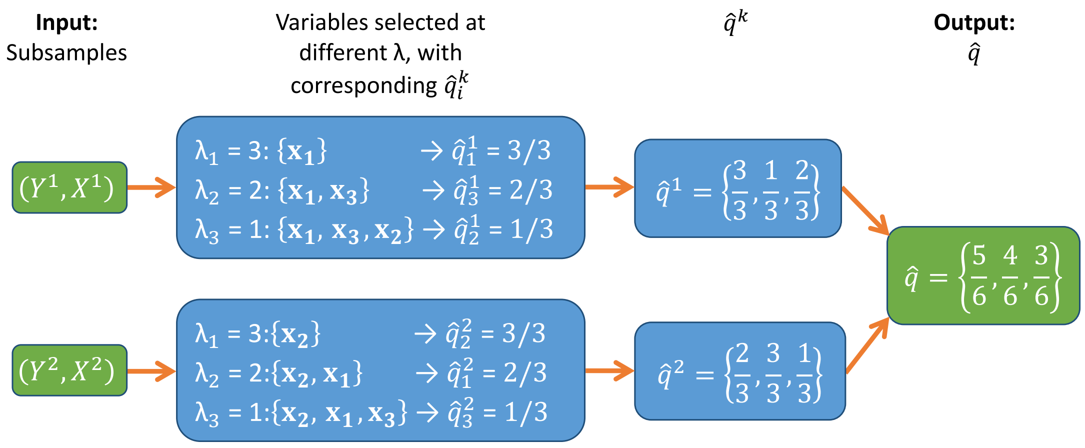

Input:
Subsamples
Variables selected at different λ, with corresponding $\hat{q}_i^k$
$\hat{q}^k$
Output:
$\hat{q}$
(Y^1, X^1)
$\lambda_1 = 3: \{\mathbf{x}_1\} \rightarrow \hat{q}_1^1 = 3/3$
$\lambda_2 = 2: \{\mathbf{x}_1, \mathbf{x}_3\} \rightarrow \hat{q}_3^1 = 2/3$
$\lambda_3 = 1: \{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_2\} \rightarrow \hat{q}_2^1 = 1/3$
$\hat{q}^1 = \left\{\frac{3}{3}, \frac{1}{3}, \frac{2}{3}\right\}$
(Y^2, X^2)
$\lambda_1 = 3: \{\mathbf{x}_2\} \rightarrow \hat{q}_2^2 = 3/3$
$\lambda_2 = 2: \{\mathbf{x}_2, \mathbf{x}_1\} \rightarrow \hat{q}_1^2 = 2/3$
$\lambda_3 = 1: \{\mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_3\} \rightarrow \hat{q}_3^2 = 1/3$
$\hat{q}^2 = \left\{\frac{2}{3}, \frac{3}{3}, \frac{1}{3}\right\}$
$\hat{q} = \left\{\frac{5}{6}, \frac{4}{6}, \frac{3}{6}\right\}$

Figure 2: Computation of $\widehat{q}$ on 2 subsamples using coordinate descent.

Algorithm 3 estimates the average L_0 path via coordinate descent, serving the same purpose as Algorithm 1. Algorithm 3 uses λ to record the order that each variable enters the path. Consider the example in Figure 2. To reparameterize the solution path, we denote λ_s to be the λ value that coordinate descent lasso includes s variables, $\forall s \in (0, \min\{n/2, p\}]$, giving a sequence of λ for grid search. In each subsample $\left(Y^k, X^k\right)$, we train a standard pathwise coordinate descent for lasso, allowing λ to increase stepwise within the grid $\left\{\lambda_1, \ldots, \lambda_{\min\{n/2,p\}}\right\}$, where $\lambda_1 \geqslant \ldots \geqslant \lambda_{\min\{n/2,p\}}$. In Figure 2, when $\lambda \leqslant \lambda_3$ at subsample $\left(Y^1, X^1\right)$, all three variables are activated in the solution path, implying that $\widehat{q}_i^1 \geqslant 1/3$ for all variables. When λ increases to λ_2, only $\{\mathbf{x}_3, \mathbf{x}_1\}$ survive the harsher shrinkage, implying that they should be ranked higher than $\mathbf{x}_2$. As a result, $\widehat{q}_1^1, \widehat{q}_3^1 \geqslant 2/3$ and $\widehat{q}_2^1 = 1/3$. When λ hits λ_3, only $\{\mathbf{x}_1\}$ remains activated, leaving $\widehat{q}_1^1 = 3/3$ and $\widehat{q}_3^1 = 2/3$. Subsequently, we apply the same method to each subsample, resulting in the same $\widehat{q}$ as Algorithm 1.

3. Solar improvements over lasso variants, lasso rules, and variable screening

3.1. Comparison and generalization to lasso variants

As shown above, solar averages and re-ranks lasso paths using the corresponding L_0 norm. Compared with lasso, solar has the following unique features.

- Solar abandons shrinkage parameter optimization, determining a new L_0 path by discretizing the original lasso path. Compared with the L_1 path, the L_0 path is concentrated on dimensionality rather than parameter value and, hence, greatly boosts variable selection performance. More importantly, to avoid the NP-hard computation inherent in the traditional L_0 method (e.g., the best subset method via AIC/BIC), the L_0 path is computed from L_1-penalized optimization, improving speed even in large-scale and repetitive learning tasks (see Section 4).

- Because solar no longer needs to optimize λ or t, it avoids the cross-validation conundrum. While unbiased and more generalizable than AIC/BIC, cross-validation faces a *bias-variance trade-off* (Kearns and Ron, 1999) and a *stability issue* (Lim and Yu, 2016). Both issues significantly impact hyperparameter tuning and, hence, lasso selection performance (sparsity, stability, etc). Without a shrinkage parameter, solar relies on a jackknife- or bootstrap-like algorithm—training for K rounds, removing $1/K$ of data each round—to stabilize the path.

Because it is trained by least-angle regression or coordinate descent, solar can easily be generalized to many lasso-type problems. For example,

- 'Grouped solar' is invoked by requiring some variables to be simultaneously selected into the solution path;

- 'Adaptive solar' is obtained by weighting variable rankings in the average L_0 path according to their OLS coefficients;

- 'Solar elastic net' or 'fused solar' is derived by replacing the coordinate descent loss function in Algorithm 3 with the L_1-L_2 loss

$$\|Y - X\beta\|_2^2 + \lambda^{(1)} \|\beta\|_1 + \lambda^{(2)} \|\beta\|_2^2 \tag{3.1}$$

  or fused loss

$$\|Y - X\beta\|_2^2 + \lambda^{(1)} \|\beta\|_1 + \lambda^{(2)} \sum_{j=2}^{p} |\beta_j - \beta_{j-1}|_1 . \tag{3.2}$$

Most importantly, because they use the same optimization methods, many lasso enhancements (e.g., lasso rules, post-lasso hypothesis testing) can be applied to solar. Rather than competing with the lasso enhancements, solar supplements them, further improving variable selection efficiency and performance in large-scale applications.

3.2. Solar advantages on post-selection hypothesis testing

As we shall see in Section 4, solar exhibites a substantial superiority over lasso-type algorithms in terms of selection sparsity, accuracy and efficiency. In addition, solar is amenable to post-selection testing. Because the lasso tests (Lockhart et al., 2014; Taylor et al., 2014) are based on least-angle regression, they may be adapted to solar. More interestingly, it is straightforward to adapt the data-splitting tests (Wasserman and Roeder, 2009; Meinshausen et al., 2009) to solar for a further performance boost. We illustrate this point using Example 1.

Example 1. Suppose the DGP is

$$Y = 2\mathbf{x}_0 + 3\mathbf{x}_1 + 4\mathbf{x}_2 + 5\mathbf{x}_3 + 6\mathbf{x}_4 + \sum_{j=5}^{p} 0 \cdot \mathbf{x}_j + e, \tag{3.3}$$

where all $\mathbf{x}_i$ are standard Gaussian with pairwise correlations of 0.5 and e is a standard Gaussian noise. Post-selection tests are typically applied with sufficiently large n, so $p/n = 50/100$. In this setting, forward regression (FR) selects $[\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0, \mathbf{x}_{45}, \mathbf{x}_8, \mathbf{x}_{40}, \mathbf{x}_6, \mathbf{x}_{39}, \mathbf{x}_{11}, \mathbf{x}_{34}, \mathbf{x}_5]$ using BIC score minimization. By contrast, solar selects only $[\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0, \mathbf{x}_{45}, \mathbf{x}_{40}]$. We then conduct post-FR and post-solar t-tests and compare the results to the OLS regression of Y on the $\mathbf{x}_i$.

After standardizing the $\mathbf{x}_i$ and $\mathbf{Y}$, the sample regression coefficient is identical to the corresponding conditional correlation. Both the t-test results and FR selection decisions are largely based on the sample regression coefficient absolute values; keeping variables with large values and purging the rest. Hence, the variables selected by FR also have low p-values on the current sample, implying that direct, post-FR t-tests overfit the sampling randomness that also affects the variable selection algorithm (referred to as 'p-value overfitting'). The main problem here is that there is no sample change between the FR sample and the post-FR test sample.

To solve this issue, we improve the data-splitting tests by merging the Bousquet and Elisseeff (2002) method—a jackknife-type algorithm—and the Romano and DiCiccio (2019) theoretical results, as follows: after solar selection on the whole sample, we randomly divide the sample into K folds, remove each fold in turn, and compute the t- and p-values of post-solar OLS on the remaining $K-1$ folds. We calculate the average t-, p-values, and standard errors in K rounds resulting in the *K-fold average*. In this example we focus on $K=2$, which we refer to as the *hold-out average*. Our approach has the following advantages over the Wasserman and Roeder (2009) and Meinshausen et al. (2009) methods:

- The Meinshausen et al. (2009) method requires a computationally costly, bootstrap-like repetition: each bootstrap subsample is split, one portion is used for selection, the other for testing. By contrast, the hold-out average only uses one realization of variable selection and two post-selection tests. This is useful when the dimensionality or data size is very large (see the simulation runtime comparison at section 4).

- Rather than using a portion of the data, the hold-out average uses the whole sample for variable selection, improving variable selection accuracy.

- The hold-out average only relies on the concentration inequalities for averaging testing performance. Hence, the hold-out average is consistent with the Romano and DiCiccio (2019) framework and inherits a number of elegant theoretical properties. Most importantly, the hold-out averaging test with $K=2$ splits the data into two disjoint portions, which removes the autocorrelation of t- and p-values across rounds and further simplifies the Romano and DiCiccio (2019) theoretical derivation.[1]

As shown in Table 1, the hold-out average t-values for $[\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0]$ are very similar to their OLS counterparts while the hold-out average t-values for $[\mathbf{x}_{40}, \mathbf{x}_{45}]$ are lower, which leads to purging $\mathbf{x}_{40}$ and $\mathbf{x}_{45}$. We repeat the simulation 200 times and the results, summarized in Figure 3, confirm that improvements in the hold-out average are not simply due to chance. It is important to note that OLS uses all of the sample and has 50 residual degrees of freedom whereas the hold-out average uses only 50% of the sample in each round and consequently has 30-40 degrees of freedom. Thus, the hold-out average naturally produces slightly lower t-values for $[\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0]$ and the t-value boxplots for the hold-out average are slightly less concentrated than the OLS boxplots. Simulations in Section 4.4, show that solar supplemented with the hold-out average purges almost all of the redundant variables while retaining all of the informative variables. ■

[1] Due to the similarity between K-fold CV and K-fold averaging, $K \leqslant 10$ (Friedman et al., 2001) is also recommended for controlling the autocorrelation among t- or p-values across different rounds.

Table 1: Comparison of post-selection t tests between FR, hold-out average, and OLS.

	post-FR			hold-out average			classical OLS		
	se	t	$P > \|t\|$	se	t	$P > \|t\|$	se	t	$P > \|t\|$
$\mathbf{x}_0$	0.12	15.68	0	0.18	11.34	0	0.19	9.85	0
$\mathbf{x}_1$	0.14	20.06	0	0.22	12.76	0	0.23	12.05	0
$\mathbf{x}_2$	0.14	28.11	0	0.21	18.22	0	0.20	19.45	0
$\mathbf{x}_3$	0.13	39.19	0	0.20	24.48	0	0.20	25.08	0
$\mathbf{x}_4$	0.13	42.34	0	0.21	27.36	0	0.20	28.15	0
$\mathbf{x}_{40}$	0.12	2.59	0.01	0.19	1.90	0.18	0.18	2.36	0.02
$\mathbf{x}_{45}$	0.13	2.30	0.02	0.21	1.19	0.25	0.19	1.60	0.12

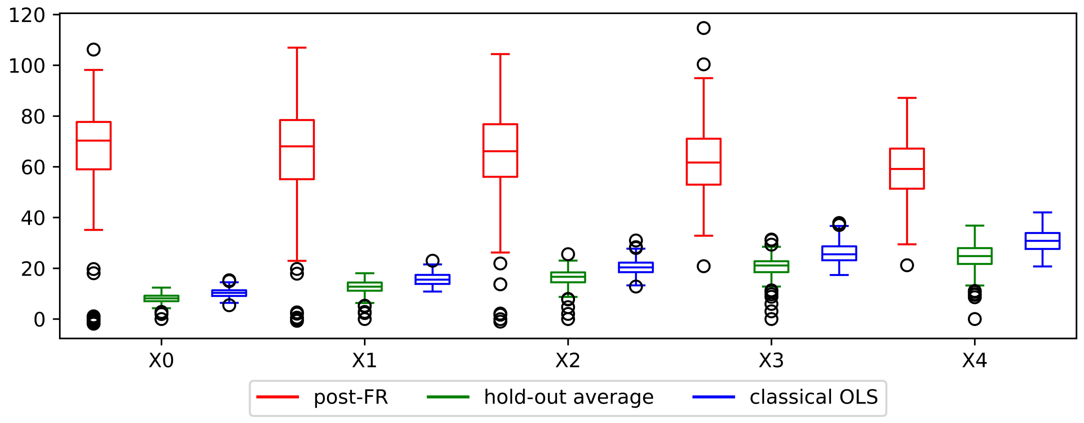


Figure 3: Boxplots of post-selection t values from FR, hold-out average, and OLS.

While the hold-out average procedure can be applied to other methods, solar has several intrinsic advantages.

- Firstly, to ensure accurate tests with only half the sample, it is important to retain residual degrees of freedom, implying that variable selection must be as sparse and accurate as possible. As shown in Sections 4 and 5, sparse and accurate selection is one of the benefits of solar.

- Secondly, the residual degree of freedom is less than p in high dimensional data; hence, when applying post-selection tests to rectify redundant variable selections or informative variable omissions, we need decide which β_i to test. Zhang (2009, Theorem 2) suggests that the earlier a variable enters the path, the more likely it is informative. As a result, we should testing variables with higher ranks in priority. Compared to FR and lasso paths, the ranking in average L_0 path of solar is robust to settings of the irrepresentable condition, sampling noise, multicollinearity, and other issues. Hence, solar also provides a stable and accurate guidance on β_i testing priority.

To conclude, Example 1 illustrates that the hold-out average may significantly reduce the severity of ‘p-value overfitting’ while utilizing as many data points as possible for selection and testing. This method helps reduce the severity of informative variable omission or redundant variable inclusion.

3.3. Solar advantages under complicated dependence structures

Indeed, solar has another advantage: the average L_0 solution path offers additional robustness against outliers, multicollinearity, and noise in high-dimensional spaces. Thus, solar is likely to be more reliable than other variable selection methods under complicated dependence structures. We illustrate the point with the following two Bayesian network examples.

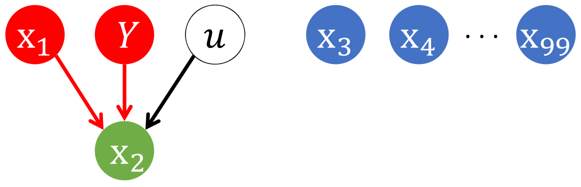


Figure 4: Y is unconditionally uncorrelated to an informative $\mathbf{x}_1$.

Consider the non-standard regression case in Figure 4, which depicts a very common empirical regression problem in biostatistics and finance: *informative variables* may be *unconditionally uncorrelated to* Y in the DGP. For instance, either concussion ($\mathbf{x}_1$) or brain tumor (Y) may cause headache ($\mathbf{x}_2$), implying that we must consider the concussion history when diagnosing a brain tumor. Hence, $\mathbf{x}_1$ and $\mathbf{x}_2$ are informative variables for Y, even though $\mathbf{x}_1$ and Y are independent. In such scenario, Example 4 demonstrates that solar is more reliable than post-lasso rules and variable screening.

Example 2. In Figure 4, there are 100 variables and $\mathbf{x}_2$ is (causally) generated by its parents $\{\mathbf{x}_1, Y\}$ as follows,

$$\mathbf{x}_2 = \alpha_1 \mathbf{x}_1 + \alpha_2 Y + u, \tag{3.4}$$

where $\mathbf{x}_1$ is unconditionally uncorrelated with Y, $\mathbf{x}_1$ and Y are both unconditionally and conditionally uncorrelated with the redundant variables, $\{\alpha_1, \alpha_2\}$ are population regression coefficients, and u is a Gaussian noise term. If Y is chosen to be the response variable, we have the population regression equation

$$Y = -\frac{\alpha_1}{\alpha_2}\mathbf{x}_1 + \frac{1}{\alpha_2}\mathbf{x}_2 - \frac{1}{\alpha_2}u. \tag{3.5}$$

Note that $\mathbf{x}_1$ and $\mathbf{x}_2$ are both informative variables for Y. However, since $\mathbf{x}_1$ is unconditionally uncorrelated with Y in the population, some post-lasso rules [such as the strong rule (Tibshirani et al., 2012) and the safe rule (Ghaoui et al., 2010)] may be prone to purging $\mathbf{x}_1$ incorrectly. Given the value of shrinkage parameter λ in a grid search, the base strong rule and the safe rule for lasso to purge any selected variable, respectively, satisfies (3.6) and (3.7):

$$\left|\mathbf{x}_i^T Y\right| < \quad \lambda - \left\|\mathbf{x}_i\right\|_2 \left\|Y\right\|_2 \tfrac{\lambda_{max}-\lambda}{\lambda_{max}}; \tag{3.6}$$

$$\left|\mathbf{x}_i^T Y\right| < \quad 2\lambda - \lambda_{max}, \tag{3.7}$$

where the $\mathbf{x}_i$ are standardized and λ_{max} is the value of λ that purges all the variables. Both rules are based on the unconditional covariance between $\mathbf{x}_i$ and Y. For a given value of λ (typically selected by CV), lasso likely will select $\mathbf{x}_1$ and $\mathbf{x}_2$ along with some redundant variables from $\{\mathbf{x}_3, \ldots, \mathbf{x}_{99}\}$ (since the DGP does not violate any IRC). Since $\text{corr}\left(\mathbf{x}_1, Y\right) = \text{corr}\left(\mathbf{x}_3, Y\right) = \cdots = \text{corr}\left(\mathbf{x}_{99}, Y\right) = 0$ in the population, the sample value of $\left|\mathbf{x}_1^T Y\right|$ will be approximately as small as the $\left|\mathbf{x}_i^T Y\right|$ of any redundant variable. Put another way, $\mathbf{x}_1$ cannot be distinguished from the redundant variables by the value of $\left|\mathbf{x}_i^T Y\right|$. To ensure $\mathbf{x}_1$ is not purged by (3.6) or (3.7), both $\lambda - \|\mathbf{x}_1\|_2 \|Y\|_2 \frac{\lambda_{max}-\lambda}{\lambda_{max}}$ and $2\lambda - \lambda_{max}$ must be smaller than $\left|\mathbf{x}_1^T Y\right|$. However, this will lead to two problems. First, decreasing the right-hand side of (3.6) and (3.7) will reduce the value of λ, implying that lasso will select more redundant variables. Second, since $\left|\mathbf{x}_1^T Y\right|$ will be approximately as small as the $\left|\mathbf{x}_i^T Y\right|$ of any redundant variable selected by lasso, not purging $\mathbf{x}_1$ (by reducing both right-hand side terms) may result in (3.6) and (3.7) retaining redundant variables.

Variable screening methods (Fan and Lv, 2008) may also be prone to selecting redundant variables. Screening ranks variables decreasingly based on the absolute values of their unconditional correlations to Y, selecting the top w variables (with w selected by CV, bootstrap or BIC). Since $\text{corr}\left(\mathbf{x}_2, Y\right) \neq 0$ in the population, screening will rank $\mathbf{x}_2$ highly. However, it may not rank $\mathbf{x}_1$ highly because $\text{corr}\left(\mathbf{x}_1, Y\right) = 0$ in the population. Thus, some redundant variables may be ranked between $\mathbf{x}_2$ and $\mathbf{x}_1$, implying that if both $\mathbf{x}_1$ and $\mathbf{x}_2$ are selected, screening will select redundant variables.

The average L_0 solution path will not suffer the same problem. For convenience, assume $-\alpha_1/\alpha_2 > 0$ and $p/n = 100/200$ or smaller. For lars, as we increase $\|\beta_2\|_1$ at stage 1 (i.e., as we 'partial' $\mathbf{x}_2$ out of Y), the unconditional correlation between $Y - \beta_2\mathbf{x}_2$ and $\mathbf{x}_1$ will increase above 0 significantly while the marginal correlation between $Y - \beta_2\mathbf{x}_2$ and any redundant variable will remain approximately 0. Thus, in the L_0 solution path and, hence, the average L_0 solution path, $\mathbf{x}_1$ will be included immediately after $\mathbf{x}_2$ is included. ■

Fan and Lv (2008) and Barut et al. (2016) propose three solutions for the Example 2 variable screening issue. However,

- the first approach (Barut et al., 2016, Section 2.2 and 3) assumes the identity of $\mathbf{x}_2$ in Example 2 and Figure 4 is known (also called a *collider* in Bayesian network or probabilistic graph modelling, Barut et al. (2016) call it a *hidden signature* variable and denote it by X_c), which is unlikely to be realistic in practical applications;

- the second approach (Barut et al., 2016, Section 1 and 2.2) suggests that randomly trying out several variables as colliders may be beneficial. The logic is straightforward: randomly trying out the wrong variable to be a collider ($\mathbf{x}_2$ in Example 2 and Figure 4) is harmless because conditioning on that variable will not make $corr(Y, \mathbf{x}_1) \neq 0$, nor will it cause the selection of a redundant variable. Moreover, by repeatedly randomly trying out variables, there is a non-zero probability the correct collider is eventually uncovered, producing a statistically significant $corr(Y, \mathbf{x}_1) \neq 0$. However, this method crucially relies on multiple trials, which may be inefficient and computationally expensive, especially with high-dimensional data. Even though $corr(Y, \mathbf{x}_1) \neq 0$ after simultaneously trying out multiple variables (say $\{\mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{99}\}$), we still need to additionaly check whether there are redundant variales among $\{\mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{99}\}$ in case of selecting redundant variables, leaving variable selection a possible loose end.

The next example illustrate another common probelm in empirical regressions: *redundant variables* may be *unconditionally correlated to* Y in the DGP. As shown in Figure 5, such problem often occurs when $\mathbf{x}_3$ and Y are determined by common variables. For instance, house rent (Y) and food expenditure ($\mathbf{x}_3$) are both determined by income and saving ($\mathbf{x}_1$ and $\mathbf{x}_2$); however, $\mathbf{x}_3$ is redundant if you have already predict Y with $\{\mathbf{x}_1, \mathbf{x}_2\}$. We carefully choose parameter values to demonstrate that, even when IRC is satisfied, the strong rule, base rule, and variable screening methods may have difficulty purging the redundant $\mathbf{x}_3$. By contrast, solar is more reliable.

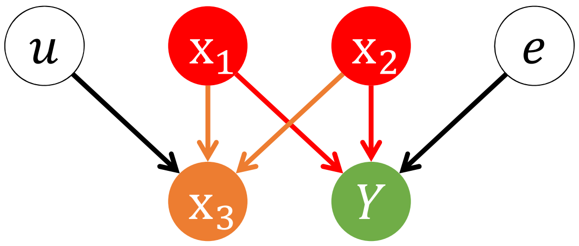


Figure 5: Y is unconditionally correlated with redundant $\mathbf{x}_3$.

Example 3. Consider the following confounding structure,

$$\begin{cases} \mathbf{x}_3 = \frac{1}{3}\mathbf{x}_1 + \frac{1}{3}\mathbf{x}_2 + \frac{\sqrt{7}}{3}u, \\ Y = \frac{7}{10}\mathbf{x}_1 + \frac{2}{10}\mathbf{x}_2 + \frac{\sqrt{47}}{10}e. \end{cases} \tag{3.8}$$

As shown in Figure 5, $\mathbf{x}_1$ and $\mathbf{x}_2$ cause both Y and $\mathbf{x}_3$, implying that $\mathbf{x}_3$ is unconditionally correlated to Y; $\mathbf{x}_1$, $\mathbf{x}_2$, u and e are independent; $\mathbf{x}_3$ is independent from e; Y is independent from u; and all variables are standardized. When n is large and the sample correlations close to their population values, the sample marginal correlations to Y can be ranked in decreasing order,

$$\begin{aligned} &\text{corr}(\mathbf{x}_1, Y) = 0.7 \\ &\text{corr}(\mathbf{x}_3, Y) = \text{corr}\left(\frac{1}{3}\mathbf{x}_1 + \frac{1}{3}\mathbf{x}_2, \frac{7}{10}\mathbf{x}_1 + \frac{2}{10}\mathbf{x}_2\right) = 0.3. \\ &\text{corr}(\mathbf{x}_2, Y) = 0.2 \end{aligned} \tag{3.9}$$

Because $\mathbf{x}_2$ ranks below $\mathbf{x}_1$ and $\mathbf{x}_3$ in terms of marginal correlations to Y, the variable screening method will have to select all 3 variables, including the redundant $\mathbf{x}_3$, to avoid omitting $\mathbf{x}_2$. Similarly, the base strong rule and safe rule may also have difficulty purging $\mathbf{x}_3$. Because $\operatorname{corr}(\mathbf{x}_3, Y)$ is larger than $\operatorname{corr}(\mathbf{x}_2, Y)$, if lasso selects both $\mathbf{x}_3$ and $\mathbf{x}_2$ and we use the base strong rule or the safe rule to purge $\mathbf{x}_3$, we will also have to purge $\mathbf{x}_2$.

Forward regression, solar and lasso will not make the same errors. Because (3.8) does not violate the IRC, variable-selection consistency of forward regression, lasso and lars is assured by the theoretical results of Zhang (2009) and Zhao and Yu (2006). Specifically in forward regression, $\mathbf{x}_1$ will be included at the first stage. After controlling for $\mathbf{x}_1$, the partial correlations to Y of both $\mathbf{x}_2$ and $\mathbf{x}_3$ can be ranked in decreasing order as follows (when n is large),

$$\begin{aligned} \operatorname{corr}(\mathbf{x}_2, Y|\mathbf{x}_1) &= \operatorname{corr}\left(\mathbf{x}_2, \frac{2}{10}\mathbf{x}_2\right) = 0.2. \\ \operatorname{corr}(\mathbf{x}_3, Y|\mathbf{x}_1) &= \operatorname{corr}\left(\frac{1}{3}\mathbf{x}_1 + \frac{1}{3}\mathbf{x}_2, \frac{2}{10}\mathbf{x}_2\right) = 0.0667. \end{aligned} \tag{3.10}$$

Thus, at the second stage, forward regression will include $\mathbf{x}_2$, not $\mathbf{x}_3$. After controlling for both $\mathbf{x}_1$ and $\mathbf{x}_2$, the remaining variation in Y comes from e, which $\mathbf{x}_3$ cannot explain. As a result, CV or BIC will terminate forward regression after the second stage and $\mathbf{x}_3$ will not be selected. Similarly, because solar relies on the average L_0 path, it will include $\mathbf{x}_1$ and $\mathbf{x}_2$ but not $\mathbf{x}_3$. $\blacksquare$

Essentially, the strong rule, safe rule, and variable screening struggle in these two examples because they rely on unconditional correlations to Y, whereas informative variables in regression analysis are defined in terms of conditional correlations. In many scenarios, unconditional and conditional correlations are aligned. However, when they are not, variable selection based conditional correlation is better placed to select the informative variables.

Fan and Lv (2008) propose the following solutions. That approach requires repeating the following steps: running variable screening on Y, selecting variables having high unconditional correlations with Y, and running a lasso of the residuals on the dropped variables. In contrast, by using conditional correlation ranking, solar completes variable selection in a single realization, reducing the computational cost in large-scale applications. Moreover, the Fan and Lv (2008) approach does not solve the Example 3 problem. In their first step, variables with high unconditional correlations to Y (including the redundant $\mathbf{x}_3$) will be selected, implying the inclusion of invalid variables. The issue will be exacerbated if Y has multiple $\mathbf{x}_3$-like siblings within a complicated dependence structure: the multicollinearity will produce inaccurate estimations of the regression coefficients and standard errors in finite samples. In short, solar is likely to be more computationally efficient and better at variable selection under complicated dependence structures.

3.4. Solar robustness to different settings of the IRC

Here we illustrate that, compared to lasso, solar has superior robustness to different settings of the *irrepresentable condition (IRC)*, which is considered to be sufficient and almost neccessary for accurate lasso variable selection (Zhang, 2009). We skip lasso rules and variable screening here since, as discussed above, their selection accuracy may be affected by their reliance on unconditional correlation to Y. Following Zhang (2009), we define IRC as follows,

Definition 3.1 (IRC)**.** Given $F \subset \{1, \ldots, p\}$, define X_F to be the $n \times |F|$ matrix with only the full set of informative variables. Define

$$\mu(F) = \max\left\{ \left\| \left((X_F)^T X_F \right)^{-1} (X_F)^T \mathbf{x}_j \right\|_1 \mid \forall j \notin F \right\}.$$

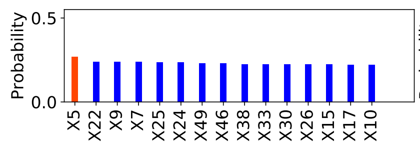
Given a constant $1 \geqslant \eta > 0$, the *strong* irrepresentable condition is satisfied if $\mu(F) \leqslant 1 - \eta$ and the *weak* irrepresentable condition is satisfied if $\mu(F) < 1$.■

We follow the simulation of Zhao and Yu (2006) and slightly modify the Example 3 DGP. Here, $n = 200$, $p = 50$, and $[\mathbf{x}_0, \ldots, \mathbf{x}_4, \mathbf{x}_6, \ldots, \mathbf{x}_{50}]$ is generated from a zero-mean, unit-variance multivariate Gaussian distribution, where all the correlation coefficients are 0.5. The DGP of Y and $\mathbf{x}_5$ is

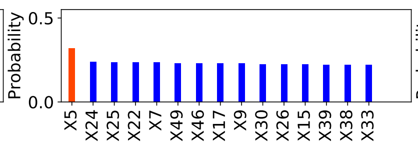
$$\begin{cases} \mathbf{x}_5 = \omega \mathbf{x}_0 + \omega \mathbf{x}_1 + \gamma \cdot \sqrt{1 - 2\omega^2} \\ Y = 2\mathbf{x}_0 + 3\mathbf{x}_1 + 4\mathbf{x}_2 + 5\mathbf{x}_3 + 6\mathbf{x}_4 + e \end{cases} \tag{3.11}$$

where $\omega \in \mathbb{R}$, while γ and e are both standard Gaussian noise terms, independent from each other and all the other variables. Compared with Example 3, this DGP has more redundant signals, increasing the challenge of purging redundant variables. The example also makes it straightforward to control the value of $\mu(F)$ with ω.

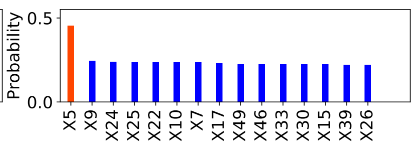
In (3.11), IRC only affects the redundant $\mathbf{x}_5$. Hence, we focus on the probability of wrongly selecting $\mathbf{x}_5$ over 200 repetitions. By setting ω to either 1/4, 1/3, or 1/2, the population value of $\mu(F)$ changes, respectively, to 1/2, 2/3, or 1, gradually increasing the difficulty of purging the redundant $\mathbf{x}_5$. Figure 6 displays the simulation results. When $\mu(F) = 1/2$, lasso wrongly includes $\mathbf{x}_5$ with probability 0.25. By contrast, $\mathbf{x}_5$ does not crack the top-10 list for solar, implying that the corresponding probability remains below 0.1. When $\mu(F)$ increases to 2/3, the probability that lasso includes $\mathbf{x}_5$ increases to around 0.3, standing out from the other 9 variables. When $\mu(F)$ jumps to 1 in the population and strong IRC is violated, the probability that lasso includes $\mathbf{x}_5$ rises to almost 0.5. Despite the increase in $\mu(F)$, the probability that solar includes $\mathbf{x}_5$ is always below 0.1. The results illustrate that solar is more robust to different settings of the IRC.



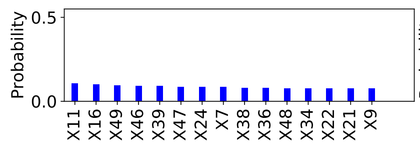
(a) $\omega = 1/4,\ \mu(F) = 1/2$, lasso



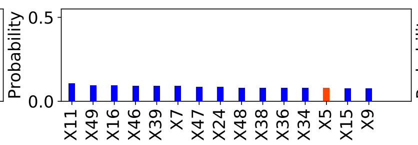
(b) $\omega = 1/3,\ \mu(F) = 2/3$, lasso



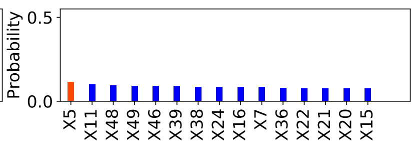
(c) $\omega = 1/2,\ \mu(F) = 1$, lasso



(d) $\omega = 1/4,\ \mu(F) = 1/2$, solar



(e) $\omega = 1/3,\ \mu(F) = 2/3$, solar



(f) $\omega = 1/2,\ \mu(F) = 1$, solar

Figure 6: Probability of including redundant variables (top 15) in simulation 2 ($\mathbf{x}_5$ in orange).

4. Solar improvements over lasso-related, subsampling variable selection

In this section, we show that: (i) solar and hold-out averaging offer significant improvements over lasso-type algorithms in terms of variable selection sparsity, stability, accuracy, and; (ii) replacing lasso with solar in subsampling variable selection drastically reduces the computational load (both runtime and subsample repetition) while improving sparsity and accuracy.

4.1. Simulation competitors

We do not consider all lasso-type algorithms for comparison. Firstly, some lasso modifications (e.g., fused lasso, grouped lasso) are designed to solve specific empirical problems that are not relevant to our paper. Secondly, it may be difficult to determine how much some variants outperform lasso.[2] Since both solar and lasso may be evaluated via least-angle regression and coordinate descent, many other lasso modifications can be directly applied to solar, as discussed in section 3.1. We also do not consider information criteria for shrinkage parameter tuning. Pedregosa et al. (2011) points out that information criteria are over-optimistic and require a proper estimation of the degrees of freedom of the solution. Moreover, information criteria are derived asymptotically and tend to breakdown when the problem is badly conditioned (e.g., $p > n$).[3]

Least-angle regression and coordinate descent yield nearly identical selection results for solar and lasso. Hence, we combine the lars results and coordinate descent results for solar, ignore the coordinate

[2]For example, while Jia and Yu (2010) show numerically that elastic net has slightly better variable-selection accuracy than lasso, they also find that "when the lasso does not select the true model, it is more likely that the elastic net does not select the true model either" (a point we verify in Section 5). While simulations in Zou (2006) show that adaptive lasso outperforms lasso when $p/n < 1$, it requires first computing the OLS estimates of all $\mathbf{x}_i$ coefficients, which is difficult when $p/n > 1$.

[3]See `https://scikit-learn.org/stable/modules/linear_model#lasso.html` for details.

descent lasso (which is included in the supplementary files), and report only the runtime comparison between least-angle regression and coordinate descent.

Along with lasso, we include subsampling variable selection algorithms (aka subsampling ensembles). A lasso subsampling ensemble repeats lasso multiple times across subsamples, producing the averaged (or accumulated) selection result of the repetitions. In our simulation, we compare the lasso and solar subsampling ensembles and demonstrate that the solar ensemble improves variable selection performance and substantially reduces computation load. Due to the similarity across lasso subsampling ensembles, we choose the bootstrap lasso (Bach, 2008) to be the competitor. Bach (2008) proposes two bolasso algorithms: bolasso-H and bolasso-S. Bolasso-H selects only variables that are selected in all subsamples, i.e., the subsample selection frequency threshold, $f = 1$. Bolasso-S selects variables that are selected in 90% of the subsamples ($f = 0.9$). In our simulation, we use both as the competitors. Bach (2008) finds that it is always beneficial, in terms of selection and prediction, to generate more subsamples in bolasso. Hence, to compare solar with state-of-art bolasso performance, we set the number of subsamples in bolasso to 256, the maximum in the Bach (2008) simulations.

In a similar vein, we introduce the bootstrap solar ensemble *(bsolar)*, which runs solar on each subsample and computes the selection frequency for each variable across subsamples. To ensure that any performance improvement is due to replacing lasso with solar in the ensemble system, bsolar and bolasso use the same f. Thus, we evaluate 2 versions of bsolar: bsolar-H ($f = 1$) and bsolar-S ($f = 0.9$). To denote the number of subsamples clearly, we use the notation bsolar-mH and bsolar-mS, where m is the number of subsamples used to compute the selection frequency.

4.2. Simulation settings

The DGP is as follows. All p covariates in $X \in \mathbb{R}^{n\times p}$ are generated by a zero-mean, multivariate Gaussian, with covariance matrix having 1 on the main diagonal and 0.5 for the off-diagonal elements. The first 5 are informative variales, generating the response variable $Y \in \mathbb{R}^{n\times 1}$ as

$$Y = 2\mathbf{x}_0 + 3\mathbf{x}_1 + 4\mathbf{x}_2 + 5\mathbf{x}_3 + 6\mathbf{x}_4 + e, \tag{4.1}$$

where $e \in \mathbb{R}^{n\times 1}$ is a standard Gaussian noise. All data points are independently and identically distributed. Each $\mathbf{x}_j$ is independent from the noise term e, which is standard Gaussian. Solar competes with K-fold, cross-validated lasso (denoted 'lasso' for short). We choose the number of CV folds and the number of subsamples generated in Algorithm 1 to be 10, following the Friedman et al. (2001) simulations that show $K = 10$ balances the bias-variance trade-off in CV error minimization.

We use three criteria to compare the variable-selection performance of solar and lasso: sparsity, accuracy, and runtime. Sparsity is measured by the mean of the number of variables selected. Accuracy is measured by the average number of selected informative variables. Runtime is the average CPU time for one realization.

In this simulation, we generate data with various p/n (each with 200 repeats and fixed Python random generators) as follows. In scenario 1, p/n approaches 0 from above, corresponding to the classical $p < n$ setting. In scenario 2, p/n approaches 1 from above, corresponding to high-dimensional settings. In scenario 3, $p/n = 2$ as $\log(p)/n$ slowly approaches 0, corresponding to ultrahigh-dimensional settings, i.e., where $(p - n) \to \infty$. The raw simulation results are also in the supplementary file.

4.3. Programming languages, parallelization and hardware for simulations

We carefully design the simulation settings so that our comparisons are fair, representative, and generalizable and, in particular, to enable a like-for-like comparison to state-of-the-art lasso algorithms.

First, to maximize computation speed, we use `Numpy`, `Scipy`, and `Cython`—all well-known for performance and speed—to outsource all numerical and matrix operations to `Intel MKL`, currently the fastest and most accurate C++/Fortran library for CPU numerical operations.

Second, to reduce the possibility of CPU and RAM bottlenecks in parallel computing of the lasso and lasso-related subsampling algorithms, we choose Python rather than R as the programming interface. Research shows [e.g., Donoho (2017)] that "R has the well-known reputation of being less scalable than Python to large problem sizes", which is due to its memory-CPU management. Given the simulations repeat solar, lasso, and lasso-related subsampling algorithms many times to arrive at accurate average performance measures, choosing Python over R greatly reduces any potential hardware limitations. Moreover, computations are carried out with an Intel Xeon W-3245 CPU with 3.2GHz frequency, 10-processor parallelization, and 64GB RAM, further reducing the possibility of CPU-RAM bottlenecks.

Third, to guarantee the programming quality of the lasso package, we implement lasso and lasso-related subsampling algorithms from the Sci-kit learn library (Pedregosa et al., 2011), one of the most efficient machine-learning packages and one that is widely used in research and industry.[4] Sci-kit learn also use `Numpy`, `Scipy` and `Cython` to source all numerical/matrix operations to the Fortran/C++. Again, the idea is to maximize the lasso computation speed.

Lastly, to optimize computation and avoid large overheads, we implement 10-processor parallelization. Because each realization of solar and CV-lasso requires 10 repetitions of lars or pathwise coordinate descent, optimization must be carried out sequentially, meaning that each realization of solar or lasso must wait for the preceding realization to finish. Thus, we design the parallel architecture to assign one realization per CPU core. The design is optimized for a CPU with at least 10 cores and may generate a parallel overhead with fewer cores. For example, in each realization, an 8-core CPU would assign two cores to training the second repetition of coordinate descent with the other 6

[4]More detail is available at `https://scikit-learn.org/stable/`.

remaining idle until all 10 repetitions are finished.

4.4. Accuracy and sparsity comparison with varying p/n

Table 2: Simulation result.

		$p/n \to 0$			$p/n \to 1$			$\log(p)/n \to 0$		
		$\frac{100}{100}$	$\frac{100}{150}$	$\frac{100}{200}$	$\frac{150}{100}$	$\frac{200}{150}$	$\frac{250}{200}$	$\frac{400}{200}$	$\frac{800}{400}$	$\frac{1200}{600}$
average number of selected variables	lasso	19.73	19.84	19.54	22.30	23.57	26.56	28.92	33.88	37.96
	solar	9.86	8.66	8.50	11.34	9.8	8.2	10.54	13.28	15.52
	solar + hold out	5.02	5.12	5.17	4.99	5.16	5.13	5.12	5.24	5.26
	bsolar-3S/3H	5.44	5.18	5.22	5.44	5.18	5.22	5.25	5.86	6.09
	bsolar-5S/5H	5.14	5.07	5.1	5.14	5.07	5.1	5.08	5.28	5.46
	bsolar-10S	5.12	5.04	5.04	5.12	5.04	5.04	5.05	5.24	5.39
	bsolar-10H	5.06	5.01	5	5.06	5.01	5	5.01	5.09	5.17
	bolasso-S	5.46	6.09	6.60	5.46	6.09	6.6	5.58	6.63	7.67
	bolasso-H	5	5.02	5.01	5	5.02	5.01	5	5.01	5.02
average number of selected informative variables	lasso	5	5	5	5	5	5	5	5	5
	solar	5	5	5	5	5	5	5	5	5
	solar + hold out	4.95	5	5	4.91	5	5	5	5	5
	bsolar-3S/3H/5S/5H/10S/10H	5	5	5	5	5	5	5	5	5
	bolasso-S/H	5	5	5	5	5	5	5	5	5
average runtime	solar + hold out	0.07	0.08	0.08	0.08	0.10	0.13	0.19	0.49	0.83
	bsolar-3	0.11	0.16	0.15	0.10	0.15	0.22	0.41	0.74	1.48
	bolasso (lars, 256 SR)	9.52	12.49	10.61	10.01	13.92	19.72	23.10	184.59	502.56
	bolasso (cd, 256 SR)	13.49	60.51	60.35	13.92	16.85	20.17	27.73	100.58	308.12

Table 2 summarizes the average selection performance.[5] While solar and lasso always include all 5 informative variables, solar clearly outperforms lasso in terms of sparsity in every p/n scenario, implying a much better ability to control redunant variable inclusion. More interestingly, the lasso sparsity deteriorates as p/n approaches 1, while solar sparsity continues to improve. As $\log(p)/n \to 0$, while the sparsity of each competitor deteriorates, solar maintains a clear advantage over lasso.

Table 2 also confirms a solar advantage over lasso in subsampling ensembles. All subsampling ensembles select the 5 informative variable with probability 1. Among all subsampling ensembles, bolasso-S sparsity is always the worst while the others perform nearly identically. However, bolasso requires 256 subsample lasso repetitions while bsolar-3, -5 and -10 only take 3, 5, and 10 subsample solar repetitions, respectively. Hence, bsolar reduces subsample repetitions by 96%. As we will show in section 4.6, solar and lasso have identical computation loads. As a result, the 96% reduction in subsample repetitions implies at least a 96% reduction in computation time, assuming time complexity increases linearly.

Most importantly, the performance of *solar + hold out* (the combination of solar and the hold-out average) in Table 2 illustrates that solar performance can be further boosted by hold-out average test.

[5]Detailed histograms are in the supplementary file.

Provided n is not very small (a necessary requirement for hypothesis testing), *solar + hold out* sparsity and accuracy are on par with any subsampling variable selection method. It is worth noting that, with limited degrees of freedom (fewer than 50 when $n = 100$), *solar + hold out* occasionally omits informative variables. Under such a situation, bsolar, which has a similar computation time, would be a better choice. Overall, these results confirm and supplement the Example 1 findings.

4.5. Bolasso-bsolar efficiency discrepency and explanations

Most of lasso-related subsampling selection algorithm requires completing CV-lasso on each subsample, directly multiplying the computation load and ranking all variables only once. However, with only 4% of subsample repetitions, bsolar performance comes on par with bolasso, which requires 256 repetitions. The bsolar efficency are quite expected due to its unique *multi-layer variable ranking scheme*.

- Algorithms 1 and 3 show that solar internalizes a variable ranking scheme (the *internal ranking*) without increasing the computational burden above lasso (see section 4.6). Internal ranking significantly improves variable selection performance and avoids the issues with lasso rules and variable screening.
- Bsolar further embeds solar into a bootstrap ranking framework. That is, once solar has ranked and selected variables on each subsample, bsolar collects the highly ranked variables across the subsamples and produces a ranking based on new test data, selecting the most stable of the strong signals (the *external ranking*).

Research (Fan and Lv (2008); Hall et al. (2009); Hall and Miller (2009); Li et al. (2012a,b)) confirms the efficiency of one-layer ranking. As a result, the efficiency of multi-layer ranking is not surprising.

Table 3: Subsample selection frequency for bolasso and bsolar-10.

(a) bolasso

frequency	variables
$\geqslant 1.00$	$\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0$
$\geqslant 0.88$	$\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0, \mathbf{x}_{28}$
$\geqslant 0.84$	$\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0, \mathbf{x}_{28}, \mathbf{x}_{71}$
$\geqslant 0.76$	$\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0, \mathbf{x}_{28}, \mathbf{x}_{71}, \mathbf{x}_{91}$
$\geqslant 0.70$	$\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0, \mathbf{x}_{28}, \mathbf{x}_{71}, \mathbf{x}_{91}, \mathbf{x}_{94}$
$\geqslant 0.69$	$\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0, \mathbf{x}_{28}, \mathbf{x}_{71}, \mathbf{x}_{91}, \mathbf{x}_{94}, \mathbf{x}_{70}, \mathbf{x}_{40}$
$\vdots$	$\vdots$

(b) bsolar-10

frequency	variables
$\geqslant 1.00$	$\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0$
$\geqslant 0.10$	$\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_0, \mathbf{x}_{91}, \mathbf{x}_{71}$

Table 3 shows numerically how efficient the mutli-layer ranking is. Table 3a breaks down the subsample selection frequency list from 256 subsamples for one bolasso realization with $p/n = 100/200$.

Due to the length of the list, we report only subsample selection frequencies ≥ 0.69. Equipped with only one layer of ranking, bolasso is not able clearly to separate informative from redundant variables even with 256 subsample repetitions. The frequency discrepancy between the highest-ranking redundant ($\mathbf{x}_{28}$) and the lowest-ranking informative variable ($\mathbf{x}_0$) is only 0.12. By contrast, bsolar-10 returns a much shorter list, the frequency discrepancy between the highest-ranking redundant ($\mathbf{x}_{91}$) and the lowest-ranking informative variable ($\mathbf{x}_0$) is 0.9.

As a possible fix, Bach (2008) suggests that increasing the number of subsamples in bolasso may increase the discrepancy between the lowest ranked informative and the highest ranked redundant variables. However, increasing the subsample repetitions substantially raises the computation load of bolasso in high-dimensional spaces, making bsolar even more computationally preferable.

4.6. Computation load comparison

Since the computation load of lars or coordinate descent on a given sample is fixed, we may use the number of lars or coordinate descent to approximate the computation loads of solar and lasso. For comparison purposes, we compute solar with K subsamples and lasso with K-fold cross-validation. As shown in Algorithm 1 and 3, we need to compute one lars or coordinate descent for solar on each subsample (X^k, Y^k), implying we need to estimate K lars or coordinate descent to compute $\widehat{q}$ and one to compute c^* for variable selection. Lasso also requires computing K lars or coordinate descent to optimize the tuning parameter and, given the optimal tuning parameter, one extra on the full sample to select variables. Thus, with the same optimizer, solar and lasso have the same computation load.

Due to the equality of the computataion load between solar and lasso, the computation load differences between bolasso and bsolar are due primarily to subsample repetition. So we measure computation load by *the number of subsample repetitions (SR) required by the competitors*. Thus, solar and lasso have a computation load of 1 SR, bolasso has a load of 256 SR, and bsolar-3/5/10 a load of 3/5/10 SR.

Since bsolar and bolasso are very similar in terms of their computation procedures, they are coded similarly to make the comparison fair. The only code difference lies in the computation of subsample selection frequency.[6] Thus, runtime differences below are due to the reduction in subsample repetitions from the solar ensemble. To push bolasso to its maximum computation speed, we compute bolasso using the built-in `Sci-kit learn` parallel scheme (denoted as *built-in parallel*), which is fully optimized for lasso. Besides, we follow the recommendations of the `Sci-kit learn` and design a high-performance parallel architecture for *solar + hold out* and bsolar. Table 2 summarizes average runtime in 10 repetitions on Intel Xeon W-3245 CPU with 3.2GHz frequency, 10-processor parallelization and

[6]See README.html in the supplementary file for more detail.

64GB RAM. Both *solar + hold out* and bsolar-3 have much shorter runtimes than bolasso. The runtime differences become even more pronounced as p and n increase. The huge repetition number drastically render the bootstrap variable selection algorithm computational infeasible even with moderate p and n. By contrast, bsolar3 only require 30 realization of lars or coordinate descent. Due to very light computational load and CPU usage, the CPU can work at a much higher frequency than it does for bolasso, making runtime for each lars realization much shorter.

4.6.1. Comparison with previous lasso computation research

Our finding on bolasso actually aligns with the previous research on lasso. With same number of folds for CV ($K = 10$), number of λ in the grid search (100), and convergence condition, the time complexity of lasso is mostly relevent to n, p and pairwise correlation among covariates ($corr$). Thus, we conduct the following comparison at $p/n = 1000/100, corr = 0.5$.

- Friedman et al. (2010, Table 1) shows that, on a 2-core Intel Xeon CPU with 2.8GHz frequency, the average runtime of one pathwise coordinate descent realiztion in Fortran is 0.07 seconds with covariance updating. The Friedman et al. (2010) result only record the time for coordinate descent computation.

- Using Intel Xeon W-3245 CPU with 3.2GHz frequency and 10-processor parallelization, the average runtime for our coordinate descent bolasso package is 41.92 seconds with covariance pre-computed automatically, accounting for 256 realizations of 10-fold, cross-validated lasso. Hence, the total 41.92 seconds is for 2816 coordinate descent realizations, resulting in 0.014 seconds per realization. With a slightly faster CPU frequency, our average time also account for generating data, ranking varaibles in bolasso and a final-stage variable selection.

Thus, comparing our result to Friedman et al. (2010), our average speedup for one pathwise coordinate descent is roughly 5. The theoretical maximum speedup can be accurately calculated via Amdahl's law. Both lars and coordinate descent requre 10 repetitions to optimize λ and one extra to compute β based on the previous 10, implying that 1 of the 11 repetitions (11% of the computations, including data generation, variable ranking, matrix manipulation, I/O, etc.) is not parallelizable. Hence, with a given computation load (n and p), Amdahl's law implies that the 16-core maximum speedup over 2-core is roughly

$$\frac{1}{\rho + (1-\rho)/s} = \frac{1}{0.11 + (1-0.11)/(16/2)} \approx 4.52. \tag{4.2}$$

where ρ is the computation proportion that is not parallelizable, s is the computation speedup for the parallelizable proportion (e.g., the core number multiple). Note that our CPU frequency is also higher than the Friedman et al. (2010, Table 1) (3.2GHz over 2.8); hence, we need to adjust the maximum speedup above by multiplying with the frequency multiple (3.2/2.8), resulting the final maximum speedup as $4.52 \times 3.2/2.8 \approx 5.17$.

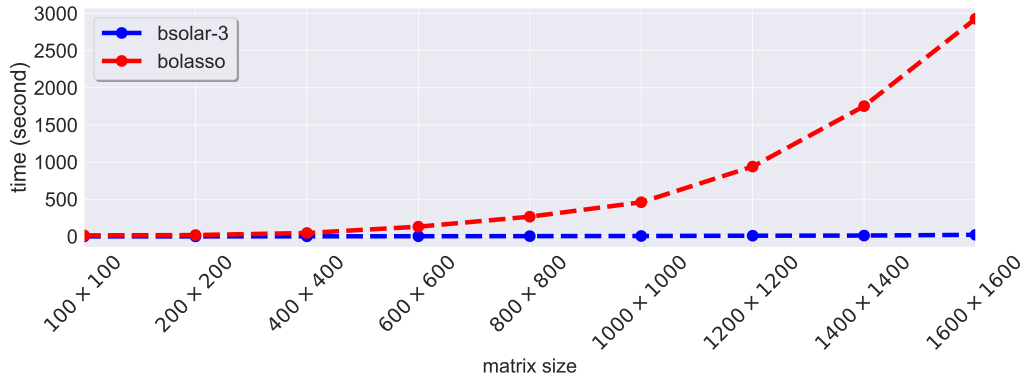


Figure 7: Average runtime comparison with different input matrix sizes (by pathwise coordinate descent).

It is hard to make a perfectly fair comparison of runtimes with different hardwares. However, by comparing our lasso speedup with Amdahl's law, we can somehow illustrate that our coordinate descent bolasso package is very competitive in terms of parallel computation efficiency. Nonetheless, bolasso still requires a large computation time with $p/n = 1200/600$ due to the 256 CV-lasso repetitions, once again confirming the bsolar computation advantage. Furthermore, Figure 7 illustrates how the average runtime progresses when the input matrix size increases. The matrix size in the figure emulates how the computation load increases for the regression algorihtms at the reproducing kernel Hilbert space (e.g., the kernel ridge and kernel lasso regression). Bsolar-3 runtime remains quite affordable while bolasso runtime increases exponentially, further confirming that bolasso is not efficient for a large input matrix.

Due to the lean parallelization design, the solar parallelization can be further improved. However, for a fair bsolar-bolasso comparison, we design the 10-processor parallelization algorithm to be the same for both. Due to the small number of subsample solar repetitions (3), our 10-processor parallelization is more than sufficient for bsolar-3. Thus, unlike bolasso, the performance of bsolar will not be compromised by the number of CPU cores, implying a similar efficiency on 4- and 8-core CPUs (such as i7-8500u and i9-9900k). By contrast, bolasso will suffer a large parallel overhead with a smaller number of CPU cores due to its large computation load.

The runtime analysis may be extended to a comparison between the hold-out average and the bootstrap-based data-splitting tests, as in Meinshausen et al. (2009). Bootstrap-based data-splitting tests require computationally costly, bootstrap-like repetitions: each bootstrap subsample is split, with one portion used for selection and the other for testing. Thus, with lasso used for variable selection, the runtime for the bootstrap-based data-splitting test with 256 bootstrap subsamples will be close to that of bolasso in Table 2. By contrast, the hold-out average only uses one realization of variable selection and two t-tests, offering a faster runtime.

5. Real-world data: Sydney house price prediction

To demonstrate that the improvements from solar are empirically feasible, we apply solar to real-world data. In the context of the simulations above, the real-world data reflect both the $p/n \to 0$ scenarios of simulation 1 and the high multicollinearity and complicated dependence structures of simulation 2. As such, the data reflect the challenging IRC settings, complicated dependence structures, and grouping effects typical of data in the social sciences.

The sample is assembled from more than 10 different databases: Sydney, Australia, real estate market transaction data for 11,974 houses sold in 2010, including price and house attribute information (property address, bedrooms, bathrooms, car spaces, etc.), matched to 2011 census data by Statistical Area Level 1 (the smallest census area in Australia, comprising at most 200 people or 60 households), 2010 and 2011 crime data by suburb, 2010 GIS data (extracted from a geo-spatial topological database, a climate database, a pollution database, and the Google Maps database), 2009 primary and secondary school data, and 2010 Sydney traffic and public transport data (bus routes, train stations, and ferry wharfs). We predict house price with a linear model.

Using an ensemble of Bayes network learning algorithms for data clearning, we purge variables with both very low conditional and unconditional correlations to house price. The remaining variables are listed in the first column of Table 4.[7] The 57 variables fall into 5 categories: house attributes, distance to key locations (public transport, shopping, etc.), neighbourhood socio-economic data, localized administrative and crime data, and local school quality. Pairwise correlations among all 57 covariates indicate, not surprisingly, severe multicollinearity and grouping effects, implying a harsh IRC setting.[8] Thus, heuristically increasing the value of the tuning parameter in lasso-type estimators (e.g., using the one-sd or the 'elbow' rule) is unlikely to be useful since it may trigger further grouping effects and the random dropping of variables.

Table 4 shows the selection comparison across the elastic net, lasso, and solar. With all variables in linear form, both lasso and elastic net lose sparsity due to the complicated dependence structures and severe multicollinearity in the data, consistent with Jia and Yu (2010). By contrast, solar returns a much sparser model, with only 9 variables selected from 57. A very similar result can be found with the variables in log form, hinting that solar possesses superior selection sparsity and robustness to a change in functional form. More importantly, solar variable selection outperforms the lasso-type estimators in terms of the balance between sparsity and prediction power. While pruning most of

[7]Due to 200GB size of the database, we only include the data for these variables in the supplementary file.

[8]Correlations and IRC are also reported in supplementary files.

the variables from the elastic net and lasso selections, solar reduces the post-selection regression R^2 by only 3-5%. In the supplementary file, we also perform a hold-out average method to avoid the possibility that the 'p-value overfitting' issue also affects the direct post-selection OLS R^2. However, there is no concern in this sample because $p/n = 57/11,974$. This confirms that, from the perspective of prediction, solar successfully identifies the most important variables in the database.

6. Conclusion

In this paper we offer a new least-angle regression algorithm for high-dimensional data called solar (for subsample-ordered least-angle regression). We show that solar yields substantial improvements over lasso in terms of the sparsity, stability, accuracy, and robustness of variable selection. We also illustrate the improvements relative to lasso ensembles from solar ensembles in terms of variable selection accuracy, sparsity and computation load.

Detection of weak signals is a small weakness evident in solar, although relative to the lasso competitor the difference is very slight. Nonetheless, we are working on an extension to solar, the double-bootstrap solar (DBsolar), which, if early results are any indication, promises to enable solar accurately to detect variables with weak signals.

References

Bach, F.R., 2008. Bolasso: model consistent lasso estimation through the bootstrap, in: Proceedings of the 25th international conference on Machine learning, ACM. pp. 33–40.

Barber, R.F., Candès, E.J., 2019. A knockoff filter for high-dimensional selective inference. The Annals of Statistics 47, 2504–2537.

Barut, E., Fan, J., Verhasselt, A., 2016. Conditional sure independence screening. Journal of the American Statistical Association 111, 1266–1277.

Bousquet, O., Elisseeff, A., 2002. Stability and generalization. Journal of Machine Learning Research 2, 499–526.

Donoho, D., 2017. 50 years of data science. Journal of Computational and Graphical Statistics 26, 745–766.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., 2004. Least angle regression. Annals of Statistics 32, 407–499.

Fan, J., Lv, J., 2008. Sure independence screening for ultrahigh dimensional feature space. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 70, 849–911.

Friedman, J., Hastie, T., Höfling, H., Tibshirani, R., et al., 2007. Pathwise coordinate optimization. Annals of applied statistics 1, 302–332.

Friedman, J., Hastie, T., Tibshirani, R., 2001. The elements of statistical learning. volume 1 of *Springer Series in Statistics*. Springer-Verlag New York.

Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. Journal of statistical software 33, 1.

Ghaoui, L.E., Viallon, V., Rabbani, T., 2010. Safe feature elimination for the lasso and sparse supervised learning problems. arXiv preprint arXiv:1009.4219 .

Hall, P., Miller, H., 2009. Using generalized correlation to effect variable selection in very high dimensional problems. Journal of Computational and Graphical Statistics 18, 533–550.

Hall, P., Miller, H., et al., 2009. Using the bootstrap to quantify the authority of an empirical ranking. The Annals of Statistics 37, 3929–3959.

Jia, J., Yu, B., 2010. On model selection consistency of the elastic net when $p >> n$. Statistica Sinica , 595–611.

Kearns, M., Ron, D., 1999. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. Neural Computation 11, 1427–1453.

Li, G., Peng, H., Zhang, J., Zhu, L., et al., 2012a. Robust rank correlation based screening. The Annals of Statistics 40, 1846–1877.

Li, R., Zhong, W., Zhu, L., 2012b. Feature screening via distance correlation learning. Journal of the American Statistical Association 107, 1129–1139.

Lim, C., Yu, B., 2016. Estimation stability with cross-validation (ESCV). Journal of Computational and Graphical Statistics 25, 464–492.

Lockhart, R., Taylor, J., Tibshirani, R.J., Tibshirani, R., 2014. A significance test for the lasso. Annals of Statistics 42, 413–468.

Meinshausen, N., Bühlmann, P., 2010. Stability selection. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 72, 417–473.

Meinshausen, N., Meier, L., Bühlmann, P., 2009. P-values for high-dimensional regression. Journal of the American Statistical Association 104, 1671–1681.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

Romano, J.P., DiCiccio, C., 2019. Multiple data splitting for testing. Department of Statistics, Stanford University.

Taylor, J., Lockhart, R., Tibshirani, R.J., Tibshirani, R., 2014. Exact post-selection inference for forward stepwise and least angle regression. arXiv preprint arXiv:1401.3889 7, 2.

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., Tibshirani, R.J., 2012. Strong rules for discarding predictors in lasso-type problems. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 74, 245–266.

Wang, J., Zhou, J., Liu, J., Wonka, P., Ye, J., 2014. A safe screening rule for sparse logistic regression, in: Advances in neural information processing systems, pp. 1053–1061.

Wasserman, L., Roeder, K., 2009. High dimensional variable selection. Annals of statistics 37, 2178.

Weisberg, S., 2004. Discussion following "Least angle regression," by B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Annals of Statistics 32, 490–494.

Xu, G., Huang, J.Z., et al., 2012. Asymptotic optimality and efficient computation of the leave-subject-out cross-validation. The Annals of Statistics 40, 3003–3030.

Zeng, Y., Yang, T., Breheny, P., 2017. Efficient feature screening for lasso-type problems via hybrid safe-strong rules. arXiv preprint arXiv:1704.08742 .

Zhang, T., 2009. On the consistency of feature selection using greedy least squares regression. Journal of Machine Learning Research 10, 555–568.

Zhao, P., Yu, B., 2006. On model selection consistency of Lasso. Journal of Machine Learning Research 7, 2541–2563.

Zou, H., 2006. The adaptive lasso and its oracle properties. Journal of the American statistical association 101, 1418–1429.

Table 4: Variable selection results for linear and log house price models.

		elastic net		lasso		solar	
Variable	Description	linear	log	linear	log	linear	log
Bedrooms	property, number of bedrooms	✓	✓	✓	✓	✓	✓
Baths	property, number of bathrooms	✓	✓	✓	✓	✓	✓
Parking	property, number of parking spaces	✓	✓	✓	✓	✓	✓
AreaSize	property, land size	✓	✓	✓	✓		
Airport	distance, nearest airport	✓	✓	✓	✓		
Beach	distance, nearest beach	✓	✓	✓	✓	✓	✓
Boundary	distance, nearest suburb boundary	✓	✓	✓	✓		
Cemetery	distance, nearest cemetery	✓		✓			
Child care	distance, nearest child-care centre	✓	✓	✓	✓		✓
Club	distance, nearest club	✓	✓	✓	✓		
Community facility	distance, nearest community facility	✓	✓				
Gaol	distance, nearest gaol	✓	✓			✓	✓
Golf course	distance, nearest golf course	✓	✓	✓	✓		
High	distance, nearest high school	✓	✓	✓	✓		
Hospital	distance, nearest general hospital	✓	✓		✓		
Library	distance, nearest library	✓		✓			
Medical	distance, nearest medical centre	✓	✓		✓		
Museum	distance, nearest museum	✓	✓	✓	✓		
Park	distance, nearest park	✓	✓	✓			
PO	distance, nearest post office	✓	✓		✓		
Police	distance, nearest police station	✓	✓	✓	✓		
Pre-school	distance, nearest preschool	✓	✓	✓	✓		
Primary	distance, nearest primary school	✓	✓	✓	✓		
Primary High	distance, nearest primary-high school	✓	✓	✓	✓		
Rubbish	distance, nearest rubbish incinerator	✓	✓	✓			
Sewage	distance, nearest sewage treatment	✓					
SportsCenter	distance, nearest sports centre	✓	✓	✓	✓		
SportsCourtField	distance, nearest sports court/field	✓		✓	✓		
Station	distance, nearest train station	✓		✓			
Swimming	distance, nearest swimming pool	✓	✓	✓	✓		
Tertiary	distance, nearest tertiary school	✓	✓	✓	✓		
Mortgage	SA1, mean mortgage repayment (log)	✓	✓	✓	✓	✓	✓
Rent	SA1, mean rent (log)	✓	✓	✓	✓	✓	✓
Income	SA1, mean family income (log)	✓	✓	✓	✓	✓	✓
Income (personal)	SA1, mean personal income (log)	✓					
Household size	SA1, mean household size	✓	✓	✓	✓		
Household density	SA1, mean persons to bedroom ratio	✓	✓	✓	✓		
Age	SA1, mean age	✓	✓	✓	✓		✓
English spoken	SA1, percent English at home	✓		✓			
Australian born	SA1, percent Australian-born	✓		✓			
Suburb area	suburb area	✓		✓	✓		
Population	suburb population	✓	✓		✓		
TVO2010	suburb total violent offences, 2010	✓					
TPO2010	suburb total property offences, 2010	✓	✓		✓		
TVO2009	suburb total violent offences, 2009	✓	✓	✓			
TPO2009	suburb total property offences, 2009	✓	✓				
ICSEA	local school, socio-educational advantage	✓	✓	✓	✓	✓	✓
ReadingY3	local school, year 3 mean reading score	✓	✓	✓	✓		
WritingY3	local school, year 3 mean writing score	✓	✓	✓	✓		
SpellingY3	local school, year 3 mean spelling score	✓	✓	✓			
GrammarY3	local school, year 3 mean grammar score	✓		✓			
NumeracyY3	local school, year 3 mean numeracy score	✓	✓	✓	✓		
ReadingY5	local school, year 5 mean reading score	✓					
WritingY5	local school, year 5 mean writing score	✓	✓	✓			
SpellingY5	local school, year 5 mean spelling score	✓	✓	✓			
GrammarY5	local school, year 5 mean grammar score	✓	✓	✓			
NumeracyY5	local school, year 5 mean numeracy score	✓					
	Number of variables selected	57	45	44	36	9	11
	post-selection OLS R^2	0.55	0.76	0.55	0.76	0.50	0.73
	Sample size			11,974			