Coordinate descent for SLOPE

Anonymous Author
Anonymous Institution

Abstract

1 Introduction

In this paper we present a novel numerical algorithm for Sorted L-One Penalized Estimation (SLOPE) (Bogdan, E. v. d. Berg, Su, et al. 2013; Bogdan, E. v. d. Berg, Chiara Sabatti, et al. 2015; Zeng and M. Figueiredo 2014), defined as

$$\min_{\beta \in \mathbb{R}^p} P(\beta) = \ell(\beta) + J(\beta) \tag{1}$$

where we take ℓ to be smooth and twice differentiable and

$$J(\beta) = \sum_{j=1}^{p} \lambda_j |\beta_{(j)}| \tag{2}$$

is the *sorted* ℓ_1 *norm*, defined through

$$|\beta_{(1)}| \geq |\beta_{(2)}| \geq \cdots \geq |\beta_{(p)}| \ ,$$

with λ being a fixed non-increasing and non-negative sequence.

The sorted ℓ_1 norm is a sparsity-enforcing penalty that has become increasingly popular due to several appealing properties, such as its ability to control false discovery rate (Bogdan, E. v. d. Berg, Chiara Sabatti, et al. 2015; Kos and Bogdan 2020), cluster coefficients (Mario Figueiredo and Nowak 2016; Schneider and P. Tardivel 2020), and recover sparsity and ordering patterns in the solution (Bogdan, Dupuis, et al. 2022). Unlike other competing sparse regularization methods such as MCP (C.-H. Zhang 2010) and SCAD (Fan and Li 2001), SLOPE is also a convex problem (Bogdan, E. v. d. Berg, Chiara Sabatti, et al. 2015).

In spite of the availability of predictor screening rules (Larsson, Bogdan, and Wallin 2020; Elvira and Herzet 2022), which help speed up SLOPE in the high-dimensional regime, current state-of-the-art algorithms for SLOPE perform poorly in comparison to those of more established penalization methods such as the lasso (ℓ_1-norm regularization) and ridge regression (ℓ_2-norm regularization). As a small illustration of this issue, we compared the speed at which the SLOPE and glmnet packages fit a complete regularization path for the bcTCGA data set. SLOPE takes x seconds to fit the full path, whilst glmnet requires only y seconds.

Mathurin: put result here and details in experiment section

This lackluster performance has hampered the applicability of SLOPE to many real-world applications. A major reason for why algorithms for solving ℓ_1-, MCP-, or SCAD-regularized problems enjoy better performance is that they use coordinate descent (Tseng 2001; Friedman, Hastie, and Tibshirani 2010; Breheny and Huang 2011). Current SLOPE solvers, on the other hand, rely on proximal gradient descent algorithms such as FISTA (Beck and Teboulle 2009) and the alternating direction method of multipliers method (ADMM) (Boyd et al. 2010), which have proven to be less inefficient than coordinate descent in empirical benchmarks on related problems, such as the lasso (Moreau et al. 2022). Applying coordinate descent to SLOPE is not, however, straightforward since convergence guarantees for coordinate descent require the objective to be separable, which is not the case for SLOPE. As a result, naive coordinate descent schemes can get stuck (figure 1).

In this article we address this problem by introducing a new, highly effective algorithm for SLOPE based on a hybrid proximal gradient and coordinate descent scheme. Our method features convergence guarantees and reduces the time required to fit SLOPE by orders of magnitude in our empirical experiments.

Notation Let $(i)^-$ be the inverse of (i) such that $\left((i)^-\right)^- = (i)$. See table 1 for an example of this operator for a particular β. This means that

$$J(\beta) = \sum_{j=1}^{p} \lambda_j |\beta_{(j)}| = \sum_{j=1}^{p} \lambda_{(j)^-} |\beta_j|.$$

Preliminary work. Under review by AISTATS 2023. Do not distribute.

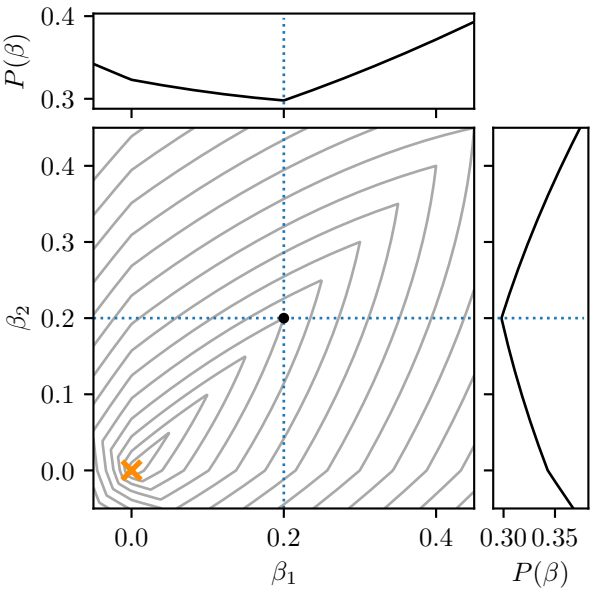

Figure 1: An example of (standard) coordinate descent getting stuck on a two-dimensional SLOPE problem. The main plot shows level curves for the primal objective (1), with the optimum $\beta^* = [0, 0]^T$ indicated by the orange cross. The marginal plots displays objective values at $\beta_1 = 0.2$ when optimizing over β_2 and vice versa. At $\beta = [0.2, 0.2]^T$, a naive coordinate descent algorithm can only move in the directions indicated by the dashed lines—neither of which are descent directions for the objective. As a result, the algorithm is stuck.

Sorted ℓ_1 norm penalization produces vectors with clustered coefficients. To this end, for a fixed β such that $|\beta_j|$ takes m values, we introduce $\mathcal{C}_1, \mathcal{C}_2 \ldots, \mathcal{C}_m$ and $c_1, c_2, \ldots, c_m$ for the indices and coefficients respectively of the m clusters of β, such that $\mathcal{C}_i = \{j : |\beta_j| = c_i\}$ and $c_1 > c_2 > \cdots > c_m \geq 0$. We also let $\bar{\mathcal{C}}$ denote the complement of $\mathcal{C}$. Let $(e_i)_{i\in[d]}$ denote the canonical basis of $\mathbb{R}^d$. Finally, let $\operatorname{sign}(x) = x/|x|$ (with the convention $0/0 = 1$) be the scalar sign, that acts entrywise on vectors.

2 Theory

Table 1: Example of the permutation operator (i) and its inverse $(i)^-$

i	β_i	(i)	$(i)^-$
1	0.5	2	3
2	-5	3	1
3	4	1	2

2.1 Coordinate Descent for SLOPE

Proximal coordinate descent cannot be applied to Problem (1) because it is not separable. However, if the clusters $\mathcal{C}_1^*, \ldots, \mathcal{C}_{m^*}^*$ of the solution β^* were known together with their signs, then the values $c_1^*, \ldots, c_{\hat{m}}^*$ taken by β^* on the clusters could be equivalently computed by solving:

Mathurin: TODO MM: put in prop and write proof?

$$\min_{z\in\mathbb{R}^{m^*}} \ell\left(\sum_{i=1}^{m^*}\sum_{j\in\mathcal{C}_i} z_i \operatorname{sign}(\beta_j^*) e_j\right) + \sum_{i=1}^{m^*} |z_i| \sum_{\lambda\in\lambda^{\mathcal{C}_i^*}} \lambda\,, \tag{3}$$

Conditionally on the knowledge of the clusters and the signs of the coefficients, the penalty becomes separable and coordinate descent could be used. Hence, the idea of our algorithm is to intertwine steps that identify the clusters, and large coordinate-wise steps on these clusters.

Based on this observation, we derive a coordinate descent algorithm for minimizing the SLOPE problem (1) with respect to the coefficients of a single cluster at a time.

In all the sequel, β is fixed, with m clusters $\mathcal{C}_1, \ldots, \mathcal{C}_m$ corresponding to values $c_1, \ldots, c_m$. Let also $k \in [m]$ be fixed, let $s = \operatorname{sign}\beta_{\mathcal{C}_k}$. We are interested in updating β by changing only the value taken on the k-th cluster. To this end, we let

$$\beta_i(z) = \begin{cases} s_k z\,, & \text{if } i \in \mathcal{C}_k\,, \\ \beta_i\,, & \text{otherwise}\,. \end{cases} \tag{4}$$

Minimizing the objective in this directions amounts to solving the following one-dimensional problem:

$$\min_{z\in\mathbb{R}} \Big(G(z) = P(\beta(z)) = F(\beta(z)) + \phi(z)\Big)\,, \tag{5}$$

where

$$\phi(z) = |z| \sum_{j\in\mathcal{C}_k} \lambda_{(j)_z^-} + \sum_{j\notin\mathcal{C}_k} |\beta_j| \lambda_{(j)_z^-}$$

is the *partial sorted* ℓ_1 *norm* with respect to the k-th cluster and where we write $\lambda_{(j)_z^-}$ to indicate that the inverse sorting permutation $(j)_z^-$ is defined with respect to $\beta(z)$. The optimality condition for Problem (5) is

$$G'(z;\delta) \geq 0,$$

where $G'(z;\delta)$ is the directional derivative in the direction δ. Since F is differentiable we have

$$G'(z;\delta) = \delta \nabla F(\beta(z)) + \phi'(z;\delta)\,,$$

where $\phi'(z;\delta)$ is the directional derivative of ϕ.

Throughout the rest of this section we derive the solution to (5). To do so, we will introduce the directional derivative for the sorted ℓ_1 norm with respect to the coefficient of the k-th cluster. First, let $C(z)$ be a function that returns the cluster corresponding to z, that is

$$C(z) = \{j : |\beta(z)_j| = z\}.$$

Next, let ε_c be an arbitrary positive value such that

$$\varepsilon_c < \left|c_i - c_j\right|, \quad \forall\, i \neq j \text{ and } \varepsilon_c < c_m \text{ if } c_m \neq 0\,. \tag{6}$$

Then for $\delta \in \{-1, 1\}$ we have

$$\begin{aligned} \lim_{h \downarrow 0} C(z + h\delta) &= C(z + \varepsilon_c \delta), \\ \lim_{h \downarrow 0} \lambda_{(i)^-_{z+h\delta}} &= \lambda_{(i)^-_{z+\varepsilon_c\delta}}. \end{aligned} \tag{7}$$

In other words, the order permutation corresponding to $\beta(z + h\delta)$ depends only on δ as h tends to 0.

Remark 2.1. As consequence of the definition of ε_c, the order permutations for $\beta(z)$ and $\beta(z + \varepsilon_c\delta)$ differ only for a subset of the permutation vectors. The permutation for $\beta(h\delta)$ corresponds to

$$\begin{cases} \mathcal{C}_1, \ldots, \mathcal{C}_{k-1}, \mathcal{C}_{k+1}, \ldots, \mathcal{C}_m, C(\varepsilon_c\delta) & \text{if } c_m > 0\,, \\ \mathcal{C}_1, \ldots, \mathcal{C}_{k-1}, \mathcal{C}_{k+1}, \ldots, C(\varepsilon_c\delta), \mathcal{C}_m & \text{if } c_m = 0\,. \end{cases}$$

If $z \neq 0$, the permutation for $\beta(z + \varepsilon_c\delta)$ corresponds to

$$\begin{cases} \mathcal{C}_1, \ldots, \mathcal{C}_{k-1}, \mathcal{C}_{k+1}, \ldots, & \\ \quad C(c_i + \varepsilon_c\delta), \mathcal{C}_k, \ldots, \mathcal{C}_m & \text{if } \delta = 1\,, \\ \mathcal{C}_1, \ldots, \mathcal{C}_{k-1}, \mathcal{C}_{k+1}, \ldots, & \\ \quad \mathcal{C}_k, C(c_k + \varepsilon_c\delta), \ldots, \mathcal{C}_m & \text{if } \delta = -1\,. \end{cases}$$

We are now ready to state the directional derivative of ϕ.

Theorem 2.2. *Let $c^{\setminus k}$ be the $m-1$ length version of c where the k-th coordinate has been removed: $c^{\setminus k} = (c_1, \ldots c_{k-1}, c_{k+1}, \ldots, c_m)$. Let ε_c defined as in (6). The directional derivative of the partial sorted ℓ_1 norm with respect to the k-th cluster ϕ, in the direction δ is*

$$\phi'(z;\delta) = \begin{cases} \displaystyle\sum_{j \in C(\varepsilon_c)} \lambda_{(j)^-_{\varepsilon_c}} & \text{if } z = 0\,, \\ \operatorname{sign}(z)\delta \displaystyle\sum_{j \in C(z+\varepsilon_c\delta)} \lambda_{(j)^-_{z+\varepsilon_c\delta}} & \text{if } |z| = c_i^{(k)} > 0\,, \\ \operatorname{sign}(z)\delta \displaystyle\sum_{j \in C(z)} \lambda_{(j)^-_z} & \text{otherwise}\,. \end{cases}$$

In figure 2, we show an example of the directional derivative and the objective function.

We are now ready to present the SLOPE thresholding operator.

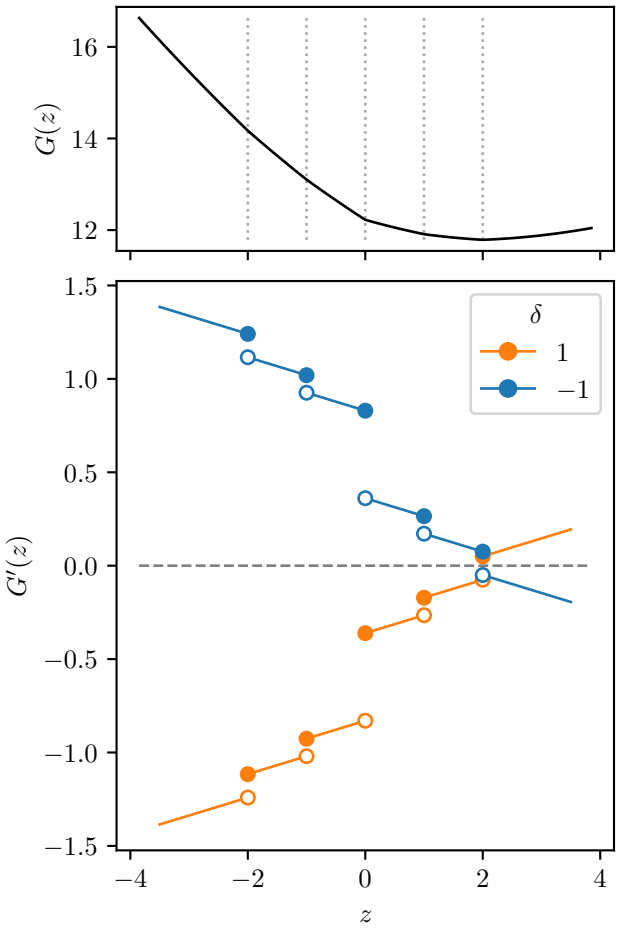

Figure 2: The function $P_k(z)$ and its directional derivative $P'_k(z;\delta)$ for an example with $\beta = [-3.0, 1.0, 3.0, 2.0]^T$, $k = 1$, and consequently $c^{\setminus k} = [2, 1]^T$. The solution corresponds to the value of z for which $P'_k(z;\delta) \geq 0$ for $\delta \in \{-1, 1\}$, which holds only at $z = 2$, which must therefore be the solution.

Theorem 2.3 (The SLOPE Thresholding Operator)**.** *Define $S(x) = \sum_{j \in C(x)} \lambda_{(j)^-_x}$ and let*

$$T_k(\gamma;\omega,c,\lambda) = \begin{cases} 0 & \text{if } |\gamma| \leq S(\varepsilon_c), \\ \operatorname{sign}(\gamma)c_i & \text{if } \omega c_i + S(c_i - \varepsilon_c) \\ & \quad \leq |\gamma| \leq \\ & \quad \omega c_i + S(c_i + \varepsilon_c), \\ \frac{\operatorname{sign}(\gamma)}{\omega}\left(|\gamma| - S(c_i + \varepsilon_c)\right) & \text{if } \omega c_i + S(c_i + \varepsilon_c) \\ & \quad < |\gamma| < \\ & \quad \omega c_{i-1} + S(c_{i-1} - \varepsilon_c), \\ \frac{\operatorname{sign}(\gamma)}{\omega}\left(|\gamma| - S(c_1 + \varepsilon_c)\right) & \text{if } |\gamma| \geq \\ & \quad \omega c_1 + S(c_1 + \varepsilon_c). \end{cases}$$

with ε_c defined as in (6) and let $\gamma = \tilde{r}^T x$, $\omega = \tilde{x}^T\tilde{x}$. Then $T(\gamma;\omega,c^{(k)},\lambda) \in \arg\min_{z \in \mathbb{R}} P_k(z)$.

JL: Consider adding a remark showing that our operator generalizes the soft thresholding operator.

In figure 3, we visualize the SLOPE thresholding operator.

2.1.1 Naive Updates

As in Friedman, Hastie, and Tibshirani (2010), we can improve the efficiency of updates by observing that

$$\begin{aligned}\tilde{r}_k &= y - \tilde{y}_k \\ &= y - X_{\bar{\mathcal{C}}_k}\beta_{\bar{\mathcal{C}}_k} - \tilde{x}c_k + \tilde{x}c_k \\ &= r + \tilde{x}c_k\end{aligned}$$

and therefore that

$$\tilde{x}_k^T(y - \tilde{y}_k) = \tilde{x}_k^T r + \tilde{x}_k^T\tilde{x}_k c_k. \tag{8}$$

2.1.2 Caching Reductions

Observe that $\tilde{x}_k$ only changes between subsequent coordinate updates provided that the members of the cluster k change, for instance if two clusters are merged, a predictor leaves a cluster, or the signs flip (through an update of α_k). As a result, it is possible to obtain computational gains by caching $\tilde{x}_k$ and $\tilde{x}_k^T\tilde{x}_k$ for each cluster (except the zero cluster, which we do not consider in our coordinate descent step). When there is no change in the clusters, there is no need to recompute these quantities. And even when there are changes, we can still reduce the costs involved since $\tilde{x}_k$ can be updated in place. If a large cluster is joined by few new predictors, then the cost of updating may be much lower than recomputing the quantities for the entire cluster. Also note that, for single-member clusters we only need to store $\tilde{x}_k^T\tilde{x}_k$ since $\tilde{x}_k$ is just a column in X times the corresponding sign.

Letting $\tilde{x}_k^{\text{old}}$ correspond to the value of $\tilde{x}_k$ before the update, we note that $\tilde{x}_k \leftarrow \tilde{x}_k^{\text{old}} + x_j \operatorname{sign}(\beta_j)$ for each $j \in \mathcal{C}_k^{\text{new}} \setminus \mathcal{C}_k^{\text{old}}$ and $\tilde{x}_k \leftarrow \tilde{x}_k^{\text{old}} - x_j \operatorname{sign}(\beta_j)$ for each $j \in \mathcal{C}_k^{\text{old}} \setminus \mathcal{C}_k^{\text{new}}$. If only the signs flip, we simply have to also flip the signs in $\tilde{x}_k$.

2.1.3 Covariance Updates

Notice that we can rewrite the first term in (8) as

$$\begin{aligned}\tilde{x}_k^T r &= \tilde{x}_k^T y - \sum_{j:\beta_j\neq 0} \tilde{x}_k^T x_j \beta_j \\ &= \tilde{x}_k^T y - \sum_{j:c_j\neq 0} \tilde{x}_k^T \tilde{x}_j c_j \\ &= s_{\mathcal{C}_k}^T X_{\mathcal{C}_k}^T y - \sum_{j:\beta_j\neq 0} s_{\mathcal{C}_k}^T X_{\mathcal{C}_k}^T x_j \beta_j \\ &= s_{\mathcal{C}_k}^T \left(X^T y\right)_{\mathcal{C}_k} - \sum_{j:\beta_j\neq 0} s_{\mathcal{C}_k}^T X_{\mathcal{C}_k}^T x_j \beta_j \\ &= \sum_{j\in\mathcal{C}_k}\left(s_j x_j^T y - \sum_{t:\beta_t\neq 0} s_j x_j^T x_t \beta_t\right)\end{aligned} \tag{9}$$

As in Friedman, Hastie, and Tibshirani (2010), this formulation can be used to achieve so-called *covariance updates*. We compute $X^T y$ once at the start. Then, each time a new predictor becomes non-zero, we compute its inner product with all other predictors, caching these products.

2.2 Hybrid proximal coordinate descent strategy

We propose an iterative solver that alternates between proximal gradient step and proximal coordinate descent. Since the regularization term for SLOPE is not separable, applying PCD does not guarantee convergence. However, Dupuis and P. J. Tardivel 2022 showed that once the clusters are known, the subdifferential of J can be written as the cartesian product of the subdifferential of J restricted to the clusters. Hence, if one knew the clusters, PCD updates could be applied on each cluster.

The notion of clusters for SLOPE extends the notion of sparsity coming from the LASSO. Identification of the sparsity pattern throughout the iterative algorithm have largely been studied. Talk about Partly Smooth functions, related Manifold and that support transpose to cluster for SLOPE regularization. DO the maths.

Then identification of this underlying structure occurs when applying PGD. Hence the idea to alternate, PGD and PCD step to take advantage of the speed of PCD and ensure convergence via the identification of the right structure with PGD steps.

In figure 4, we show how algorithm 1 works in practice on a two-dimensional SLOPE problem.

Lemma 2.4. *Let $\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(k)}$ be a sequence of iterates generated by algorithm 1, $1/v$ the frequency of proximal gradient descent iterates in algorithm 1, and*

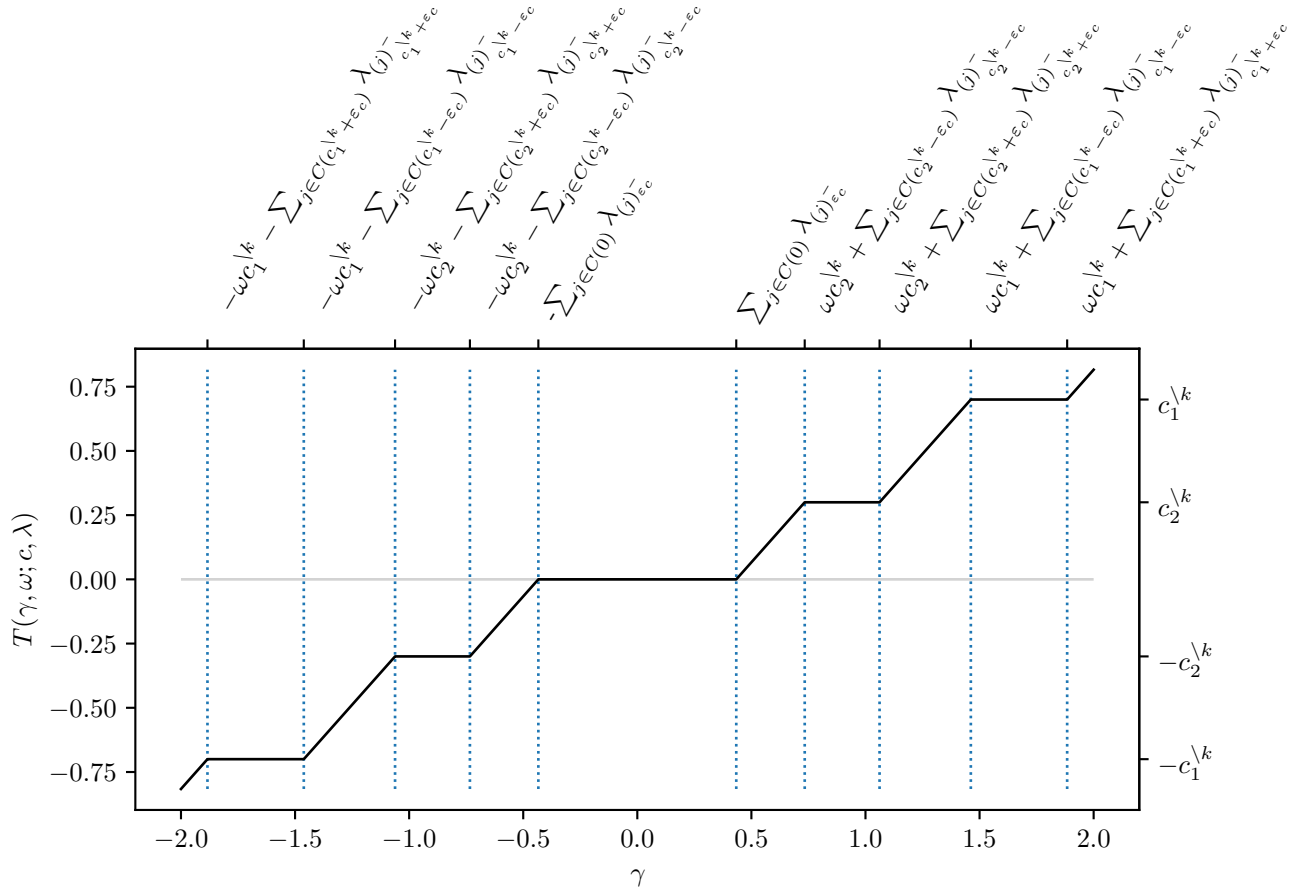

Figure 3: An example of the SLOPE thresholding operator. The result corresponds to an example for $\beta = [0.5, -0.5, 0.3, 0.7]^T$, $c = [0.7, 0.5, 0.3]^T$ with an update for the second cluster ($k = 2$), such that $c^{\setminus k} = [0.5, 0.3]^T$. Across regions where the function is constant, the operator sets the result to be either exactly 0 or to the value of one of the elements of $c^{\setminus k}$.

L the Lipschitz constant of $\nabla \ell$. Then

$$P(\beta^{(k)}) - P(\beta^*) \leq \frac{L \|\beta^{(0)} - \beta^*\|_2^2}{2\lfloor k/v \rfloor},$$

where $\beta^* \in \arg\min_{\beta \in \mathbb{R}^p} P$.

3 Experiments

The proposed algorithm is part of a python package relying on numpy and numba (Harris et al. 2020; Lam, Pitrou, and Seibert 2015). It will be made open-source upon publication. To compare its efficiency, we used several public datasets described in section 3. We performed an extensive benchmark with the following competitors:

- Alternating Direction Method of Multipliers (`admm`) (Boyd et al. 2010)
- Anderson acceleration for proximal gradient descent (`anderson`) (J. Zhang, O'Donoghue, and Boyd 2020)
- Proximal Gradient Descent (`pgd`) Combettes and Wajs 2005
- Fast Iterative Shrinkage-Thresholding Algorithm (`fista`) (Beck and Teboulle 2009)
- Semismooth Newton-Based Augmented Lagrangian (`newt-alt`) (Luo et al. 2019)
- The Oracle solver (`oracle`) uses the clusters obtained via another solver to compute coordinate descent updates from the known solver.
- The Hybrid (ours) (`hybrid`) solver (see algorithm 1) combines proximal gradient descent and coordinate descent to overcome the non-separability of the SLOPE problem.

We used the `benchopt` (Moreau et al. 2022) tool to obtain the convergence curves for the different solvers. `Benchopt` launches each solver several times increasing the number of iterations and store the objective value, dual gap and time to reach it. The repository to reproduce the benchmark is available at XXX.

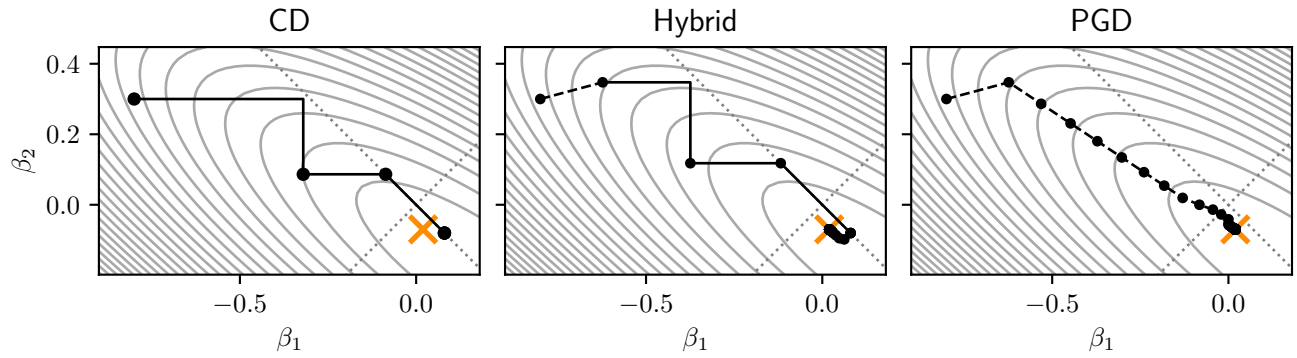

Figure 4: Illustration of the proposed solver. The figures show progress until convergence for the coordinate descent (CD) solver that we use as part of the hybrid method, our hybrid method, and proximal gradient descent (PGD). The orange cross marks the optimum. Dotted lines indicate where the coefficients are equal in absolute value. The dashed lines indicate PGD steps and solid lines CD steps. Each dot marks a complete epoch, which may correspond to only a single coefficient update for the CD and hybrid solvers if the coefficients flip order. Each solver was run until the duality gap was smaller than 10^{-10}. Note that the CD algorithm cannot split clusters and is therefore stuck after the third epoch. The hybrid and PGD algorithms, meanwhile, reach convergence after 67 and 156 epochs respectively.

Algorithm 1 Hybrid coordinate descent and proximal gradient descent algorithm for SLOPE

input: $X \in \mathbb{R}^{n\times p}, y \in \mathbb{R}^n, \lambda \in \{\mathbb{R}^p : \lambda_1 \geq \lambda_2 \geq \cdots > 0\}$, $v \in \mathbb{N}$
init : $t \leftarrow 0$, $\beta \leftarrow 0$, $L \leftarrow \|X\|_2^2$
repeat
  $t \leftarrow t + 1$
  if $t \bmod v = 0$ **then**
    $\beta \leftarrow \operatorname{prox}_J(\beta - \frac{1}{L}\nabla f(\beta); \lambda/L)$
    Update c, $\mathcal{C}$
  else
    $k \leftarrow 0$
    while $k \leq |\mathcal{C}|$ **do**
      $k \leftarrow k + 1$
      $s \leftarrow \operatorname{sign}(\beta_{\mathcal{C}_k})$
      $\tilde{x}_k \leftarrow X_{\mathcal{C}_k} s$
      $\tilde{r}_k \leftarrow y - X_{\bar{\mathcal{C}}_k}\beta_{\bar{\mathcal{C}}_k}$
      $\beta \leftarrow T(\tilde{r}^T\tilde{x}_k; \tilde{x}_k^T\tilde{x}_k, c^{\setminus k}, \lambda)$
      Update c, $\mathcal{C}$
until *convergence*;
return β

3.1 Simulated data

The design matrix X is simulated with correlation between features j and j' equal to $\rho^{|j-j'|}$ where ρ is a parameter that can be chosen in $[0, 1[$. The true regression vector $\beta^\star$ contains a certain number of non-zero entries that are obtained by simulating a gaussian distribution with zero mean and unit variance. The observations are equal to $y = X\beta^\star + \varepsilon$ where ε is a centered gaussian distribution with variance such that $\|X\beta^\star\|/\|\varepsilon\| = 3$.

Datasets	n	p	Density
Simulated 1	200	20 000	1
Simulated 2	20 000	200	1
Simulated 3	200	2 000 000	0.001
Rhee2006	842	361	?
bcTCGA	536	17 322	1
Scheetz2006	120	18 975	?

3.2 Real data

Klopfe: Do we keep the three different values for q that changes the sequence of lambas?

4 Discussion

References

Beck, A. and M. Teboulle (Jan. 1, 2009). "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". In: *SIAM Journal on Imaging Sciences* 2.1, pp. 183–202. DOI: 10.1137/080716542. URL: https://epubs.siam.org/doi/abs/10.1137/080716542 (visited on 02/10/2019).

Bogdan, Małgorzata, Ewout van den Berg, Chiara Sabatti, et al. (Sept. 2015). "SLOPE – Adaptive Variable Selection via Convex Optimization". In: *The annals of applied statistics* 9.3, pp. 1103–1140. ISSN: 1932-6157. DOI: 10.1214/15-AOAS842. pmid: 26709357. URL: https://projecteuclid.org/euclid.aoas/1446488733 (visited on 12/17/2018).

Bogdan, Małgorzata, Ewout van den Berg, Weijie Su, et al. (Oct. 29, 2013). "Statistical Estimation and

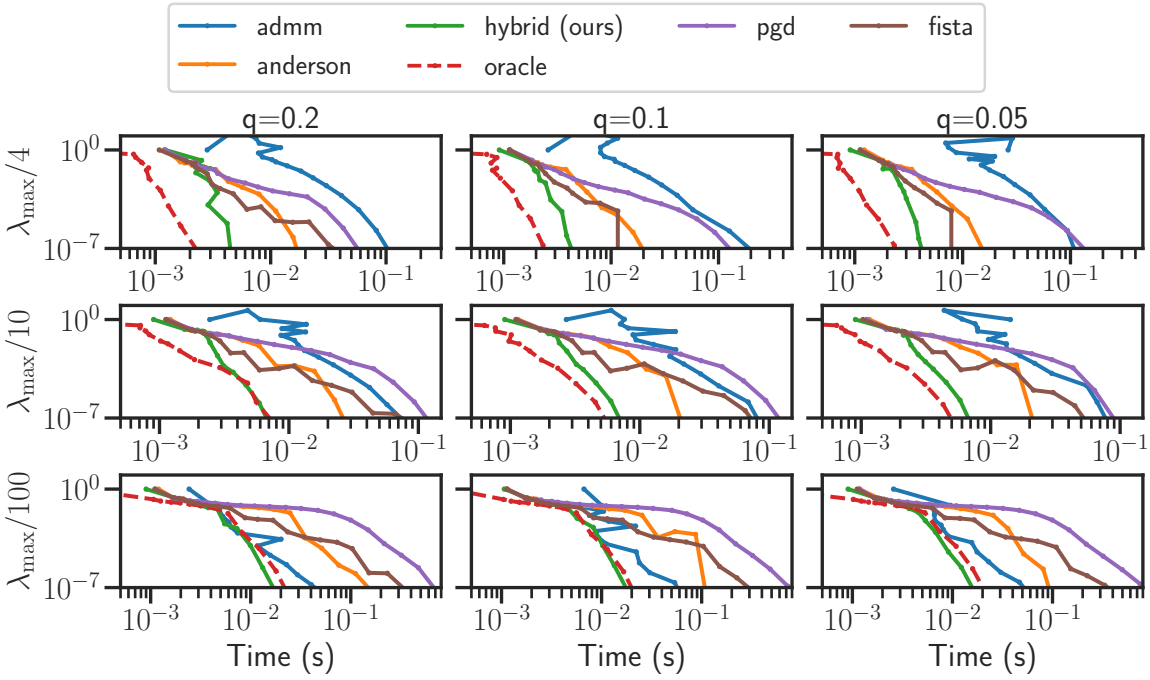

Figure 5: Benchmark on the Rhee2006 dataset.

Testing via the Sorted L1 Norm". arXiv: 1310.1969 [math, stat]. URL: http://arxiv.org/abs/1310.1969 (visited on 04/16/2020).

Bogdan, Małgorzata, Xavier Dupuis, et al. (May 17, 2022). "Pattern Recovery by SLOPE". DOI: 10.48550/arXiv.2203.12086. arXiv: 2203.12086 [math, stat]. URL: http://arxiv.org/abs/2203.12086 (visited on 06/03/2022).

Boyd, Stephen et al. (2010). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends® in Machine Learning* 3.1, pp. 1–122. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000016. URL: http://www.nowpublishers.com/article/Details/MAL-016 (visited on 02/07/2020).

Breheny, Patrick and Jian Huang (Mar. 2011). "Coordinate Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection". In: *The Annals of Applied Statistics* 5.1, pp. 232–253. ISSN: 1932-6157, 1941-7330. DOI: 10/dxfzfz. URL: https://projecteuclid.org/euclid.aoas/1300715189 (visited on 03/12/2018).

Combettes, Patrick L and Valérie R Wajs (2005). "Signal recovery by proximal forward-backward splitting". In: *Multiscale modeling & simulation* 4.4, pp. 1168–1200.

Dupuis, Xavier and Patrick J.C. Tardivel (2022). "Proximal operator for the sorted l1 norm: Application to testing procedures based on SLOPE". In: *Journal of Statistical Planning and Inference* 221, pp. 1–8. ISSN: 0378-3758. DOI: https://doi.org/10.1016/j.jspi.2022.02.005. URL: https://www.sciencedirect.com/science/article/pii/S0378375822000179.

Elvira, Clément and Cédric Herzet (Apr. 18, 2022). "Safe Rules for the Identification of Zeros in the Solutions of the SLOPE Problem". DOI: 10.48550/arXiv.2110.11784. arXiv: 2110.11784 [cs]. URL: http://arxiv.org/abs/2110.11784 (visited on 06/03/2022).

Fan, Jianqing and Runze Li (Dec. 1, 2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties". In: *Journal of the American Statistical Association* 96.456, pp. 1348–1360. ISSN: 0162-1459. DOI: 10/fd7bfs. URL: https://doi.org/10.1198/016214501753382273 (visited on 03/14/2018).

Figueiredo, Mario and Robert Nowak (May 2, 2016). "Ordered Weighted L1 Regularized Regression with Strongly Correlated Covariates: Theoretical Aspects". In: *Artificial Intelligence and Statistics*. Artificial Intelligence and Statistics, pp. 930–938. URL: http://proceedings.mlr.press/v51/figueiredo16.html (visited on 11/05/2019).

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (Jan. 2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1, pp. 1–22. DOI: 10.18637/jss.v033.i01. URL: http://www.jstatsoft.org/v33/i01/.

Harris, Charles R et al. (2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362.

Kos, Michał and Małgorzata Bogdan (Aug. 11, 2020). "On the Asymptotic Properties of SLOPE". In: *Sankhya A* 82.2, pp. 499–532. ISSN: 0976-8378. DOI: 10.1007/s13171-020-00212-5. URL: https://doi.org/10.1007/s13171-020-00212-5 (visited on 06/03/2022).

Lam, Siu Kwan, Antoine Pitrou, and Stanley Seibert (2015). "Numba: A llvm-based python jit compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6.

Larsson, Johan, Małgorzata Bogdan, and Jonas Wallin (Dec. 6, 2020). "The Strong Screening Rule for SLOPE". In: *Advances in Neural Information Processing Systems 33*. Neural Information Procession Systems 2020. Ed. by H. Larochelle et al. Vol. 33. Virtual: Curran Associates, Inc., pp. 14592–14603. URL: https://proceedings.neurips.cc/paper/2020/file/a7d8ae4569120b5bec12e7b6e9648b86-Paper.pdf.

Luo, Ziyan et al. (2019). "Solving the OSCAR and SLOPE Models Using a Semismooth Newton-Based Augmented Lagrangian Method". In: *Journal of Machine Learning Research* 20.106, pp. 1–25. URL: http://jmlr.org/papers/v20/18-172.html.

Moreau, Thomas et al. (2022). "Benchopt: Reproducible, efficient and collaborative optimization benchmarks". In: *arXiv preprint arXiv:2206.13424*.

Schneider, Ulrike and Patrick Tardivel (Aug. 18, 2020). "The Geometry of Uniqueness, Sparsity and Clustering in Penalized Estimation". DOI: 10.48550/arXiv.2004.09106. arXiv: 2004.09106 [math, stat]. URL: http://arxiv.org/abs/2004.09106 (visited on 06/03/2022).

Tseng, Paul (2001). "Convergence of a block coordinate descent method for nondifferentiable minimization". In: *Journal of optimization theory and applications* 109.3, pp. 475–494.

Zeng, X. and M. Figueiredo (2014). "The Ordered Weighted ℓ_1 Norm: Atomic Formulation, Projections, and Algorithms". In: *arXiv preprint arXiv:1409.4271*.

Zhang, Cun-Hui (Apr. 2010). "Nearly Unbiased Variable Selection under Minimax Concave Penalty". In: *The Annals of Statistics* 38.2, pp. 894–942. ISSN: 0090-5364, 2168-8966. DOI: 10/bp22zz. URL: https://projecteuclid.org/euclid.aos/1266586618 (visited on 03/14/2018).

Zhang, Junzi, Brendan O'Donoghue, and Stephen Boyd (2020). "Globally convergent type-I Anderson acceleration for nonsmooth fixed-point iterations". In: *SIAM Journal on Optimization* 30.4, pp. 3170–3197.

A Proofs

A.1 Proof of Theorem 2.2

Proof. By the definition of the directional derivative,

$$\begin{aligned}\phi'(z;\delta) =& \lim_{h\downarrow 0}\frac{\phi(z+h\delta)-\phi(z)}{h}\\ =& \lim_{h\downarrow 0}\frac{1}{h}\left(|z+h\delta|\sum_{j\in C(z+h\delta)}\lambda_{(j)^-_{z+h\delta}}+\sum_{j\notin C(z+h\delta)}|\beta_j|\lambda_{(j)^-_{z+h\delta}}-|z|\sum_{j\in C(z)}\lambda_{(j)^-_z}-\sum_{j\notin C(z)}|\beta_j|\lambda_{(j)^-_z}\right)\\ =& \lim_{h\downarrow 0}\frac{1}{h}\left(|z+h\delta|\sum_{j\in C(z+\varepsilon_c\delta)}\lambda_{(j)^-_{z+\varepsilon_c\delta}}+\sum_{j\notin C(z+\varepsilon_c\delta)}|\beta_j|\lambda_{(j)^-_{z+\varepsilon_c\delta}}-|z|\sum_{j\in C(z)}\lambda_{(j)^-_z}-\sum_{j\notin C(z)}|\beta_j|\lambda_{(j)^-_z}\right)\end{aligned} \tag{10}$$

We have the following cases to consider: $|z|\in\{c_i^{(k)}\}_{i=1}^{m-1}$, $|z|=0$, and $|z|\notin\{0,\{c_i^{(k)}\}_{i=1}^{m-1}\}$.

Starting with$|z|=c_i^{(k)}$, note that we have, as a result of the definition of ε_c, the following identities:

$$\begin{aligned}C(c_i^{(k)}+\varepsilon_c\delta) &\subseteq C(c_i^{(k)}),\\ \tilde{\mathcal{C}}_i &= \overline{C(c_i^{(k)}+\varepsilon_c\delta)}\cap C(c_i^{(k)}),\\ C(c_i^{(k)}) &= \tilde{C}_i\cup\left(C(c_i^{(k)}+\varepsilon_c\delta)\cap C(c_i^{(k)})\right)=\tilde{\mathcal{C}}_i\cup C(c_i^{(k)}+\varepsilon_c\delta),\\ \overline{C(c_i^{(k)}+\varepsilon_c\delta)} &= \overline{C(c_i^{(k)})}\cup\tilde{\mathcal{C}}_i.\end{aligned}$$

Using this, we can rewrite (10) as

$$J_k'(z;\delta)=\lim_{h\downarrow 0}\frac{1}{h}\left(\begin{array}{r}|z+h\delta|\sum_{j\in C(z+\varepsilon_c\delta)}\lambda_{(j)^-_{z+\varepsilon_c\delta}}+|c_i^{(k)}|\sum_{j\in\tilde{C}_i}\lambda_{(j)^-_{z+\varepsilon_c\delta}}+\sum_{j\in\overline{C(z)}}|\beta_j|\lambda_{(j)^-_{z+\varepsilon_c\delta}}\\ -|z|\sum_{j\in\tilde{C}_i}\lambda_{(j)^-_z}-|c_i^{(k)}|\sum_{j\in C(z+\varepsilon_c\delta)}\lambda_{(j)^-_z}-\sum_{j\in\overline{C(z)}}|\beta_j|\lambda_{(j)^-_z}\end{array}\right).$$

Next, observe that $\lambda_{(j)^-_{z+\varepsilon_c\delta}}=\lambda_{(j)^-_z}$ for all $j\in\overline{C(z)}$ and consequently

$$\sum_{j\in\overline{C(z)}}|\beta_j|\lambda_{(j)^-_{z+\varepsilon_c\delta}}=\sum_{j\in\overline{C(z)}}|\beta_j|\lambda_{(j)^-_z}.$$

Moreover, note that, since $z=\pm c_i$, there exists a permutation corresponding to $\lambda_{(j)^-_z}$ such that

$$|z|\sum_{j\in\tilde{C}_i}\lambda_{(j)^-_{z+\varepsilon_c\delta}}=|c_i^{(k)}|\sum_{j\in\tilde{C}_i}\lambda_{(j)^-_z}$$

and consequently

$$J_k'(z;\delta)=\lim_{h\downarrow 0}\frac{1}{h}\left(|z+h\delta|\sum_{j\in C(z+\varepsilon_c\delta)}\lambda_{(j)^-_{z+\varepsilon_c\delta}}-|c_i^{(k)}|\sum_{j\in C(z+\varepsilon_c\delta)}\lambda_{(j)^-_z}.\right). \tag{11}$$

Now, since $c_i^{(k)}+h\delta>0$ and $-c_i^{(k)}+h\delta<0$ in the limit as h goes to 0 for $c_i\neq 0$, we have

$$\lim_{h\downarrow 0}|-c_i+h\delta|=\lim_{h\downarrow 0}\left(|c_i^{(k)}|-h\delta\right)\quad\text{and}\quad\lim_{h\downarrow 0}|c_i^{(k)}+h\delta|=\lim_{h\downarrow 0}(|c_i^{(k)}|+h\delta)$$

which means that

$$\begin{aligned} J_k'(z;\delta) &= \lim_{h\downarrow 0}\frac{1}{h}\left(\left(|z|+\operatorname{sign}(z)h\delta\right)\sum_{j\in C(z+\varepsilon_c\delta)}\lambda_{(j)^-_{z+\varepsilon_c\delta}} - |c_i|\sum_{j\in C(z+\varepsilon_c\delta)}\lambda_{(j)^-_z}.\right) \\ &= \lim_{h\downarrow 0}\frac{1}{h}\operatorname{sign}(z)h\delta\sum_{j\in C(z+\varepsilon_c\delta)}\lambda_{(j)^-_{z+\varepsilon_c\delta}} \\ &= \operatorname{sign}(z)\delta\sum_{j\in C(z+\varepsilon_c\delta)}\lambda_{(j)^-_{z+\varepsilon_c\delta}} \end{aligned}$$

For the case when $z=0$ there are two sub-cases to consider. First, when $c_m^{(k)}=0$ then (11) is simply

$$J_k'(0;\delta) = \lim_{h\downarrow 0}\frac{1}{h}|h\delta|\sum_{j\in C(\varepsilon_c\delta)}\lambda_{(j)^-_{\varepsilon_c\delta}} = \sum_{j\in C(\varepsilon_c\delta)}\lambda_{(j)^-_{\varepsilon_c\delta}}$$

since $|\delta|=1$ by definition.

The other sub-case is $c_m^{(k)}\neq 0$. Here (10) reduces to

$$J_k'(0;\delta) = \lim_{h\downarrow 0}\frac{1}{h}\left(|h\delta|\sum_{j\in C(\varepsilon_c\delta)}\lambda_{(j)^-_{\varepsilon_c\delta}} + \sum_{j\notin C(\varepsilon_c\delta)}|\beta_j|\lambda_{(j)^-_{\varepsilon_c\delta}} - \sum_{j\notin C(0)}|\beta_j|\lambda_{(j)^-_0}\right).$$

In this case, we have $C(0)=C(\varepsilon_c\delta)$ since $c_m^{(k)}\neq 0$, and therefore

$$\sum_{j\notin C(\varepsilon_c\delta)}|\beta_j|\lambda_{(j)^-_{\varepsilon_c\delta}} = \sum_{j\notin C(0)}|\beta_j|\lambda_{(j)^-_0},$$

which means that

$$J_k'(0;\delta) = \lim_{h\downarrow 0}\frac{1}{h}|h\delta|\sum_{j\in C(\varepsilon_c\delta)}\lambda_{(j)^-_{\varepsilon_c\delta}} = |\delta|\sum_{j\in C(\varepsilon_c\delta)}\lambda_{(j)^-_{\varepsilon_c\delta}} = \sum_{j\in C(0)}\lambda_{(j)^-_0}.$$

Finally, note that J_k is differentiable in every other case, i.e. $|z|\notin\{0,\{c_i^{(k)}\}_{i=1}^{m-1}\}$. Here since $C(z+\varepsilon_c\delta)=C(z)$, has the directional derivative

$$J_k'(z;\delta) = \delta\operatorname{sign}(z)\sum_{j\in C(z)}\lambda_{(j)^-_z}.$$

□

A.2 Proof of Theorem 2.3

Proof. Recall that $P_k(z):\mathbb{R}\mapsto\mathbb{R}$ is a convex, continuous piecewise-differentiable function with kinks whenever $|z|=c_i^{(k)}$ or $z=0$. Let $\gamma=\tilde{r}^T\tilde{x}$ and $\omega=\tilde{x}^T\tilde{x}$ and note that the optimality criterion for (5) is

$$\delta(\omega z-\gamma)+J_k'(z;\delta)\geq 0,\quad \forall\delta\in\{-1,1\},$$

which is equivalent to

$$\omega z - J_k'(z;-1)\leq\gamma\leq\omega z+J_k'(z;1). \tag{12}$$

We now proceed to show that there is a solution $z^*\in\arg\min_{z\in\mathbb{R}}J_k(z)$ for every interval over $\gamma\in\mathbb{R}$.

First, assume that the first case in the definition of T_k holds and note that this is equivalent to (12) with $z=0$ since $C(\varepsilon_c)=C(-\varepsilon_c)$ and $\lambda_{(j)^-_{-\varepsilon_c}}=\lambda_{(j)^-_{\varepsilon_c}}$. This is sufficient for $z^*=0$.

Next, assume that the second case holds and observe that this is equivalent to (12) with $z=c_i^{(k)}$, since $C(c_i+\varepsilon_c)=C(-c_i-\varepsilon_c)$ and $C(-c_i+\varepsilon_c)=C(c_i-\varepsilon_c)$. Thus $z^*=\operatorname{sign}(\gamma)c_i^{(k)}$.

For the third case, we have

$$\sum_{j \in C(c_i + \varepsilon_c)} \lambda_{(j)^-_{c_i+\varepsilon_c}} = \sum_{j \in C(c_{i-1} - \varepsilon_c)} \lambda_{(j)^-_{c_{i-1}-\varepsilon_c}}$$

and therefore (12) is equivalent to

$$c_i < \frac{1}{\omega}\left(|\gamma| - \sum_{j \in C(c_i+\varepsilon_c)} \lambda_{(j)^-_{c_i+\varepsilon_c}}\right) < c_{i-1}.$$

Now let

$$z^* = \frac{\operatorname{sign}(\gamma)}{\omega}\left(|\gamma| - \sum_{j \in C(c_i+\varepsilon_c)} \lambda_{(j)^-_{c_i+\varepsilon_c}}\right) \tag{13}$$

and note that $|z^*| \in \left(c_i^{(k)}, c_{i-1}^{(k)}\right)$ and hence

$$\frac{1}{\omega}\left(|\gamma| - \sum_{j \in C(c_i+\varepsilon_c)} \lambda_{(j)^-_{c_i+\varepsilon_c}}\right) = \frac{1}{\omega}\left(|\gamma| - \sum_{j \in C(z^*)} \lambda_{(j)^-_{z^*}}\right).$$

Furthermore, since P_k is differentiable in $\left(c_i^{(k)}, c_{i-1}^{(k)}\right)$, we have

$$\frac{\partial}{\partial z} P_k(z)\Big|_{z=z^*} = \omega z^* - \gamma + \operatorname{sign}(z^*) \sum_{j \in C(z^*)} \lambda_{(j)^-_{z^*}} = 0,$$

and therefore (13) must be the solution.

The solution for the last case follows using reasoning analogous to that of the third case. □

A.3 Proof of Lemma 2.4

Proof. From theorem 2.3, we know that lines 9–14 in algorithm 1 correspond to minimizing $G(z)$ for a given $\beta := \beta^{(t+k/|\mathcal{C}|)}$, and therefore that

$$G\left(\beta^{(t+(k-1)/|\mathcal{C}|)}\right) \leq G\left(\beta^{(t+k/|\mathcal{C}|)}\right)$$

since $G(z) = P\big(\beta(z)\big)$.

Moreover, for a sequence of iterates of the proximal gradient descent step, $\beta^{(v)}, \beta^{(2v)}, \ldots, \beta^{(\lfloor k/v \rfloor)}$, we know from Beck and Teboulle (2009, Theorem 3.1) that it holds that

$$P(\beta^{(\lfloor k/v \rfloor)}) - P(\beta^*) \leq \frac{L\|\beta^{(0)} - \beta^*\|_2^2}{2\lfloor k/v \rfloor}.$$

Combining this and the result from the previous paragraph yields the desired result. □