On the Efficient Implementation of Group Lasso

James Yang

May 2, 2023

1 Group Lasso Problem

The lasso problem solves the following optimization problem:

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \; f(\beta) + \lambda \|\beta\|_1$$

where $f : \mathbb{R}^p \to \mathbb{R}$ is a convex function and $\lambda > 0$. Given data $X \in \mathbb{R}^{n \times p}$ and $y \in \mathbb{R}^n$, the most popular choices for f are the Generalized Linear Model (GLM) negative log-likelihoods under the Gaussian and Binomial family:

$$\text{Gaussian:} \quad f(\beta) := \frac{1}{2} \|y - X\beta\|_2^2 \tag{1}$$

$$\text{Binomial:} \quad f(\beta) := \sum_{i=1}^{n} \left(y_i x_i^\top \beta - \log\left(1 + e^{x_i^\top \beta}\right)\right) \tag{2}$$

The celebrated lasso problem in regression settings as in the above gained wide popularity for its interpretability as it shrinks some coefficients exactly to zero.

James: add more

The group lasso problem [Yuan and Lin, 2006] is an extension of the lasso problem that attempts to capture group structure within the features. Specifically, the group lasso attempts to zero-out *groups of coefficients* rather than a single coefficient. The interpretation is that if a group of coefficients is zero, the whole group is uninformative in predicting y [Meier et al., 2008]. This overcomes a major fallback of lasso with categorical features since lasso selects individual categorical feature (factor) instead of a whole group of features (e.g. one-hot encoded factors). Moreover, the result of lasso changes based on the encoding of factors, which is an undesirable property.

James: Cite early literature of group lasso.

In preparation of posing the group lasso problem, we begin with some notation. Let $1 \leq G \leq p$ be the number of groups among the p features. Let p_i be the group size of group i for each $1 \leq i \leq G$ such that $\sum_{i=1}^{G} p_i = p$. Define $\beta = [\beta_1 \, \beta_2 \, \ldots \, \beta_G]^\top \in \mathbb{R}^p$ where each $\beta_i \in \mathbb{R}^{p_i}$. The group lasso problem is to minimize the following objective:

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \; f(\beta) + \lambda \sum_{j=1}^{G} \|\beta_j\|_2 \tag{3}$$

James: Cite GWAS and LD stuff that it makes a lot of sense to group genes.

for some convex function $f : \mathbb{R}^p \to \mathbb{R}$ and $\lambda > 0$. Similar to the lasso, given data X and y, we may use the same Gaussian and Binomial log-likelihoods (1), (2), respectively, for f. We note in passing that group lasso can be generalized to elastic net with penalty factors just as in lasso [Zou and Hastie, 2005]:

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \; f(\beta) + \lambda \sum_{j=1}^{G} w_j \left(\alpha \|\beta_j\|_2 + \frac{1-\alpha}{2} \|\beta_j\|_2^2 \right) \tag{4}$$

where $w_j \in \mathbb{R}$ is a penalty factor for group j and $\alpha \in [0,1]$ is the elastic net weight. While our software works in this full generality, for simplicity of the discussion, we limit ourselves to the case when $\alpha = 1$ and $w_j = 1$ for all j.

James: cite

2 Optimizer Algorithm

In this section, we discuss the algorithm to solve the optimization problem (3). Let $X \in \mathbb{R}^{n \times p}$, $y \in \mathbb{R}^n$ be the feature matrix and response vector, respectively. Let G be the number of groups and p_i be the size of group i for $1 \leq i \leq G$. Further let $X = [X_1\, X_2\, \ldots\, X_G]$ where $X_i \in \mathbb{R}^{n \times p_i}$ is the feature matrix corresponding to group i features only.

In general, if the loss function f is of the form $f(\beta) \equiv f(X\beta)$, the optimization problem (3) can be reparametrized in terms of $\tilde{\beta} \in \mathbb{R}^p$ defined by $\tilde{\beta}_i := V_i^\top \beta_i$, where $V_i \in \mathbb{R}^{p_i \times p_i}$ is the orthogonal matrix from Singular Value Decomposition (SVD) of X_i. Specifically, for each $i = 1, \ldots, G$, we decompose $X_i = U_i D_i V_i^\top$ where $U_i \in \mathbb{R}^{n \times m_i}$, $D_i \in \mathbb{R}^{m_i \times p_i}$, and $V_i \in \mathbb{R}^{p_i \times p_i}$ with $m_i := \min(n, p_i)$. We will shortly see that this parametrization leads to a much simpler optimizing algorithm. Concretely, it suffices to solve

James: ref sec

$$\underset{\tilde{\beta} \in \mathbb{R}^p}{\text{minimize}}\ f(\tilde{X}\tilde{\beta}) + \lambda \sum_{j=1}^{G} \left\| \tilde{\beta}_j \right\|_2 \tag{5}$$

where $\tilde{X} = [X_1 V_1\, X_2 V_2\, \ldots\, X_G V_G]$. Note that this was only possible because the group lasso penalty is invariant under the transformation from $\beta \mapsto \tilde{\beta}$. Thus, given a solution $\tilde{\beta}^*$ to the new problem (5), we have the solution to the original problem β^* given by $\beta_i^* := V_i \tilde{\beta}_i^*$. Hence, without loss of generality, we assume that our feature matrix X is of the form $\tilde{X}$. In particular, we assume that $X_i^\top X_i = D_i^\top D_i =: \Sigma_i$ is diagonal for all $1 \leq i \leq G$. Henceforth, we drop all tilde for notational simplicity.

Computationally, it may seem daunting to construct $\tilde{X}$ from X before solving (5). However, the time taken for this pre-processing is relatively cheap compared to the full optimizing algorithm. Given the data matrix X, consider the SVD of block $X_i \in \mathbb{R}^{n \times p_i}$. If $n \geq p_i$, then $\tilde{X}_i$ can be computed in $O(np_i)$ time, noting that D_i is diagonal. Otherwise if $n < p_i$, then $D_i = [D_{i0}\ \mathbf{0}_{n \times (p-n)}]$ where $D_{i0} \in \mathbb{R}^{n \times n}$ is a diagonal matrix. Then, $\tilde{X}_i = [U_i D_{i0}\ \mathbf{0}_{n \times (p-n)}]$ so that the only product to be computed takes $O(n^2)$ time. Hence, the total complexity of the procedure is

$$O\left(\sum_{i=1}^{G}\left(\underbrace{\min(n,p_i)np_i}_{\text{SVD}} + \underbrace{\min(n,p_i)n}_{\tilde{X}_i}\right)\right) = O\left(n\sum_{i=1}^{G}\min(n,p_i)p_i\right)$$

Moreover, since this algorithm is parallelizable across groups, we can reduce the time by the number of parallel threads. From numerical experimentation, this pre-processing takes less than 1% of the total fitting time. Thus, we view this pre-processing worthwhile since its cost is cheap while inducing immense speed-ups in the optimizing algorithm.

In comparison to other proposed fitting procedures, we believe that the extra cost to transform our X matrix is comparable to the cost of the other procedures. Further, our transformation of X is more general than existing methods. For example, Yuan and Lin [2006], Meier et al. [2008] rely on the fact that each group feature matrix X_i is full-rank and therefore admits a QR decomposition with an invertible R. Under this assumption, they solve (5) with $\tilde{X} = [Q_1\, Q_2\, \ldots\, Q_G]$ and transform the solution back to the original scale by applying R_i^{-1} to the ith group of coefficients. The benefit

of this approach is that the algorithm is aided by a simple closed-form expression as part of the blockwise coordinate descent algorithm. However, there are two major downsides to this approach. The first is that, while the aforementioned authors primarily focused on the application of group lasso on multi-level categorical data that easily satisfy the full-rank assumption, this assumption may easily be violated in other applications [Simon and Tibshirani, 2012]. The second is that this method only gives an approximation to the original problem. That is, given the solution to (5), $\tilde{\beta}^*$, the candidate $\beta^\star$ defined by $\beta_i^\star := R_i^{-1}\tilde{\beta}^\star{}_i$ for $i = 1, \ldots, G$ is *not* necessarily a solution to (3) with the original data. Indeed, for this approximation to be exact, rather than solving (5), one should solve

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \; f(\tilde{X}\tilde{\beta}) + \lambda \sum_{j=1}^{G} \left\| R_i^{-1} \tilde{\beta}_j \right\|_2$$

which introduces a Tikhonov regularization. Although this remains a convex optimization problem and can be solved in theory, it does not simplify any computation for the optimizer.

Similar to other approaches [Yuan and Lin, 2006, Meier et al., 2008, Tseng, 2001, Liang et al., 2022], we use blockwise coordinate descent for its excellent speed and simplicity. However, we take a completely different approach to fitting each block coordinate update. The essence of implementing a highly performant optimizer lies in solving each block update (7) as fast as possible. Aside from the orthonormalization method described above as in Yuan and Lin [2006], Meier et al. [2008], many works in literature that describe group lasso fitting procedures use Proximal Gradient Descent (PGD), or Iterative Soft-Thresholding Algorithm (ISTA) in this context, to solve (7) [Liang et al., 2022, Beck and Teboulle, 2009, Klosa et al., 2020, Wright et al., 2009, Loris, 2009, Hastie et al., 2016, O'Donoghue and Candés, 2015]. A Nesterov acceleration can be applied on ISTA to get Fast-ISTA (FISTA) [Beck and Teboulle, 2009]. Further, O'Donoghue and Candés [2015] showed that mild improvements can be made with an adaptive restarting strategy. Section 2.1 introduces the blockwise coordinate descent algorithm in the context of solving the group lasso problem. Section 2.2 describes the PGD (ISTA) procedure and its variations to solve each block coordinate update. In contrast to PGD, Sections 2.3 and 2.4 describe our method to solve (7).

2.1 Blockwise Coordinate Descent

The convex optimization problem (5) has a separable regularization, so blockwise coordinate descent gives us guaranteed convergence [Tseng, 2001]. Hence, we iterate through each block $j = 1, \ldots, G$, keeping all other blocks fixed, and solve the following problem:

$$\underset{\beta_j \in \mathbb{R}^{p_j}}{\text{minimize}} \; f(X\beta) + \lambda \left\| \beta_j \right\|_2 \tag{6}$$

until convergence is reached. The final solution vector β then solves (5).

For the remainder of the paper, we only discuss the Gaussian loss (1). For a general f, we may apply Iterative Reweighted Least Squares (IRLS) by iteratively reweighting the data and solving (5) with the Gaussian loss. The Gaussian loss yields the optimization problem:

James: Cite IRLS. Yuan and Meier?

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \; \frac{1}{2} \left\| y - X\beta \right\|_2^2 + \lambda \sum_{j=1}^{G} \left\| \beta_j \right\|_2$$

with the block update for group j as:

$$\underset{\beta_j \in \mathbb{R}^{p_j}}{\text{minimize}} \; \frac{1}{2} \beta_j^\top \Sigma_j \beta_j - (X_j^\top y - \sum_{i \neq j} X_j^\top X_i \beta_i)^\top \beta_j + \lambda \left\| \beta_j \right\|_2$$

Algorithm 1: ISTA

Data: Σ, v, λ, $\beta^{(0)}$

$\nu \leftarrow \left(\max_i \Sigma_{ii}\right)^{-1}$;

while *not converged* **do**

  $w \leftarrow \beta^{(k-1)} + \nu\left(v - \Sigma\beta^{(k-1)}\right)$;

  $\beta^{(k)} \leftarrow \left(1 - \frac{\nu\lambda}{\|w\|_2}\right)_+ w$;

For notational simplicity, define $v_j := X_j^\top \left(y - \sum_{i \neq j} X_i \beta_i\right)$. Then, dropping the subscript j, each block update is of the form:

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \ \frac{1}{2}\beta^\top \Sigma \beta - v^\top \beta + \lambda \|\beta\|_2 \tag{7}$$

where $\Sigma \in \mathbb{R}^{p \times p}$ is diagonal with non-negative entries, $v \in \mathbb{R}^p$ is any vector of the form $D^\top u$, which follows from the fact that X_j has the SVD of the form UD based on the discussion in Section 2, and $\lambda > 0$.

2.2 Proximal Gradient Descent (PGD)

In this section, we describe the PGD method, or ISTA, to solve the block update (7). The progression is similar to that of Hastie et al. [2016]. Define

$$f(\beta) = \frac{1}{2}\beta^\top \Sigma \beta - v^\top \beta$$

as the convex (Gaussian) loss function in (7). ISTA has guaranteed convergence so long as the step-size is chosen properly. Indeed, any step-size ν can be chosen such that $\nu \leq \frac{1}{L}$ where L is the Lipschitz constant for ∇f. Since

$$\begin{aligned} \nabla f(\beta) &= \Sigma\beta - v \\ \implies \left\|\nabla f(\beta) - \nabla f(\tilde{\beta})\right\|_2 &= \left\|\Sigma(\beta - \tilde{\beta})\right\|_2 \leq \|\Sigma\|_{\text{op}} \left\|\beta - \tilde{\beta}\right\|_2 \end{aligned} \tag{8}$$

we have $L \equiv \|\Sigma\|_{\text{op}} \equiv \lambda_{\max}(\Sigma)$ as the largest eigenvalue of Σ. In particular, since Σ is diagonal, $L \equiv \max_i \Sigma_{ii}$.

The ISTA procedure is given in Algorithm 1. By applying a standard Nesterov acceleration [Beck and Teboulle, 2009], we have the FISTA procedure. Additionally, we may perform adaptive restarts based on the proximal gradient updates for even faster convergence [O'Donoghue and Candés, 2015].

Note that the computational cost of ISTA (Algorithm 1) and its variants are all $O(pk)$ where k is the number of iterations until convergence. Since we must output a vector in $\mathbb{R}^p$, any optimizer must necessarily have $O(p)$ operations. Hence, the optimal optimizer of complexity $O(pk)$ must prioritize lowering the number of iterations until convergence and the constant factor that goes into the $O(p)$ operations. While each algorithm is an improvement from the previous, there is still much improvement left. Section 2.3 shows a novel approach to solving (7) with a significant decrease in iterations while maintaining $O(p)$ operations per iteration.

Algorithm 2: Newton's Method

Data: Σ, v, λ, ε, $h^{(0)}$

$h \leftarrow h^{(0)}$;

while $|\varphi(h)| > \varepsilon$ **do**

  $h \leftarrow h - \frac{\varphi(h)}{\varphi'(h)}$;

return h;

2.3 Newton's Method

In this section, we propose a simple and powerful method to solve (7). By solving the sub-gradient problem of (7), we see that the solution $\beta^\star$ must satisfy

James: our main?

$$\beta = \begin{cases} \left(\Sigma + \frac{\lambda}{\|\beta\|_2} I\right)^{-1} v, & \|v\|_2 > \lambda \\ 0, & \|v\|_2 \leq \lambda \end{cases} \tag{9}$$

Without loss of generality, we consider the first case when $\beta^\star \neq 0$. Without knowing $\|\beta^\star\|_2$, there is no closed-form solution for $\beta^\star$. In an endeavor to find an expression for $\|\beta^\star\|_2$, we take the ℓ_2 norm on both sides to get that $\|\beta^\star\|_2$ must satisfy

$$\sum_{i=1}^{p} \frac{v_i^2}{(\Sigma_{ii} \|\beta\|_2 + \lambda)^2} = 1 \tag{10}$$

Unfortunately, there is no closed-form solution for (10) either. However, we will shortly see that a simple Newton's Method can numerically solve (10) efficiently with guaranteed convergence. In turn, once we find $\|\beta^\star\|_2$, we may substitute it into (9) to get the full solution $\beta^\star$ in $O(p)$ time. We note in passing that (9) and (10) were previously discovered [Hastie et al., 2016], however, there is no mention of how to quickly solve for $\beta^\star$ or $\|\beta^\star\|_2$.

We now discuss how to solve (10). First, define $\varphi : [0, \infty) \to \mathbb{R}$ by

$$\varphi(h) := \sum_{i=1}^{p} \frac{v_i^2}{(\Sigma_{ii} h + \lambda)^2} - 1 \tag{11}$$

so that (10) is now a root-finding problem for φ. In Appendix A.1, we show that φ is convex and is, without loss of generality, strictly decreasing with a unique root. By standard convex analysis, Newton's Method guarantees convergence to the root of φ for any initial point $h^{(0)}$ (see Appendix A.2). Algorithm 2 summarizes Newton's Method.

Since φ is strongly convex with Lipschitz second derivative, we also have quadratic convergence and each iteration costs $O(p)$ [Boyd and Vandenberghe, 2004]. In principle, our setting is one of the best settings for Newton's Method to succeed. While FISTA and the adaptive restarted version also give the same convergence rate [Beck and Teboulle, 2009, O'Donoghue and Candés, 2015], Figure 1 shows a clear improvement for Newton's Method in practice. Here, accuracy was measured as

$$\max_{i=1,\ldots,p} \left|\hat{\beta}_i - \beta_i\right| \tag{12}$$

$$\hat{\beta} := \text{ solution from algorithm}$$

$$\beta := \left(\Sigma + \frac{\lambda}{\left\|\hat{\beta}\right\|_2} I\right)^{-1} v$$

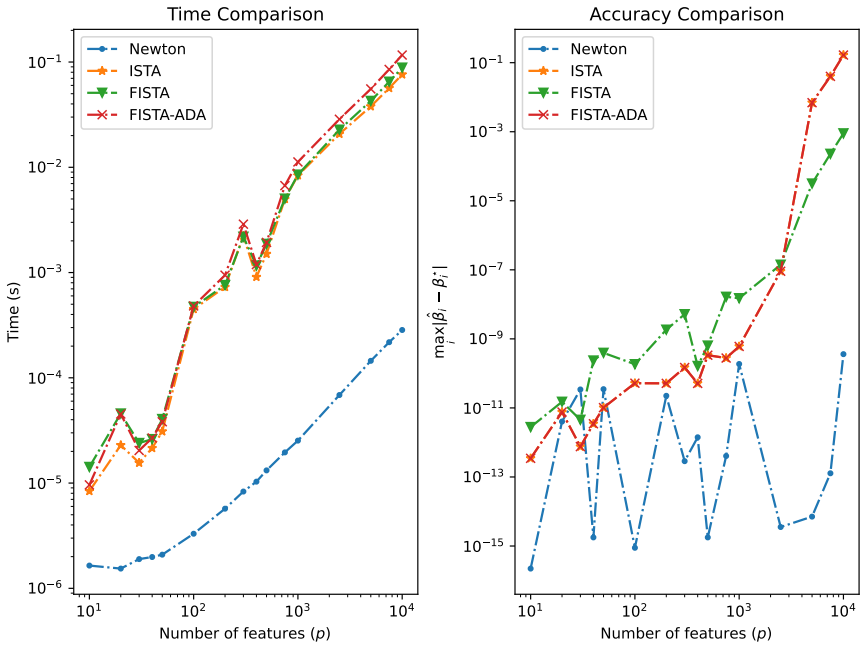

Figure 1: Time and accuracy comparison between the PGD methods in Section 2.2 and Newton's Method. Overall, the PGD methods suffer from large computational overhead due to large number of iterations, and fails to converge as the dimension p increases. Contrastingly, Newton's Method is quite stable, properly converges, and runs 10 to 1000 times faster.

Note that the PGD methods fail to converge as p increases, and even when they converge, Newton's Method performs 10 to 1000 times faster. One very important benefit of Newton's Method is that the optimization domain remains one-dimensional *regardless of the input dimension* p. However, the PGD methods directly optimize in $\mathbb{R}^p$, so their instability grows with increasing p due to the curse of dimensionality. As a result, the speed and convergence stability in Newton's Method makes it the more desirable algorithm.

It is widely known that descent methods including Newton's Method is highly sensitive to the initialization. A poor initialization may lead to slow convergence while an initialization close to the root will be rewarded with incredible convergence speed (see Figure 2). The question that remains is: how to choose the initial starting point $h^{(0)}$? We recommend selecting any $h^{(0)}$ such that $\varphi(h^{(0)}) \geq 0$. For example, $h^{(0)} \equiv 0$ is sufficient since by definition $\varphi(0) > 0$. If $h^{(0)}$ is such that $\varphi(h^{(0)}) < 0$, then it is possible that $h^{(1)} < 0$. In fact, due to the flat tail of φ, it is extremely common for $h^{(1)}$ to be negative. In this case, after projecting back to $[0, \infty)$ (i.e. setting $h^{(1)} = 0$), we arrive at the same sequence as if we had started with $h^{(0)} = 0$. For this reason, it is almost always better to set the initial point such that $\varphi(h^{(0)}) \geq 0$. Although $h^{(0)} = 0$ is a valid choice, it may also lead to large number of iterations, especially if λ is small (see Figure 3). In Section 2.4, we describe our most performant and robust modification to the vanilla Newton's Method that reduces iterations in nearly all cases.

2.4 Newton with Adaptive Bisection Starts (Newton-ABS)

In this section, we improve Newton's Method described in Section 2.3. Specifically, we first discuss how to find lower and upper bounds $h_\star$, $h^\star \in [0, \infty)$, respectively, such that the root lies in $[h_\star, h^\star]$. We then discuss an adaptive bisection method that preceeds Newton's Method to find a clever

initial point $h^{(0)}$.

We first begin with the discussion of $h_\star$ and $h^\star$. Note that the root-finding problem for φ is equivalent to finding the largest value $h > 0$ such that $\varphi(h) \geq 0$, or

$$\sup \left\{ h > 0 : \sum_{i=1}^{p} \frac{v_i^2}{(\Sigma_{ii} h + \lambda)^2} \geq 1 \right\} \tag{13}$$

In Appendix B.1, we show that we can relax this problem and solve an approximate problem using Cauchy-Schwarz to get $h_\star$ such that $\varphi(h_\star) \geq 0$. Similarly, to derive $h^\star$, we begin with the problem of finding the smallest h such that $\varphi(h) \leq 0$, or

$$\inf \left\{ h > 0 : \sum_{i=1}^{p} \frac{v_i^2}{(\Sigma_{ii} h + \lambda)^2} \leq 1 \right\} \tag{14}$$

Appendix B.1 also shows that through a different approximation, we can derive $h^\star$ such that $\varphi(h^\star) \leq 0$. This shows that the root must lie in $[h_\star, h^\star]$.

In practice, φ may decay incredibly rapidly near $h \approx 0$ and have an extremely flat tail (see Figure 2). To protect against slow convergence of Newton's Method in this case, we may use $h_\star$ and $h^\star$ to first perform a bisection method to find the initial starting point. The key idea is to use bisection first for a few iterations to avoid the region of fast decay before applying Newton's Method. Once bisection gives a sufficiently close value to the root, we apply Newton's Method for fast, guaranteed convergence. Although one may use *any* bisection method to split the interval $[h_\star, h^\star]$, e.g. the simple bisection that splits the interval in halves, we propose an *adaptive bisection method* that has been most effective.

We now describe our proposed adaptive bisection method. Ideally, we would like to know if the root is closer to $h_\star$ or $h^\star$. If we believe that the root is much closer to $h_\star$, we do not have to bisect $[h_\star, h^\star]$ at the mid-point, but perhaps at a point closer to $h_\star$. Likewise, if the root were much closer to $h^\star$, we would like to bisect at a point closer to $h^\star$. Since we do not know the root, we would like to quantify a *prior* of the root being closer to $h_\star$. With this in mind, we note that in the derivation of $h^\star$ in Appendix B.1, we approximated the problem of finding the smallest h such that $\varphi(h) \leq 0$ by solving the problem for an upper bound of φ (see (16)). Then, the more accurate the approximation, the closer the approximate solution $h^\star$ is to the root. The approximation essentially came from using the trivial fact that $\Sigma_{ii} h^\star \leq \Sigma_{ii} h^\star + \lambda$. This motivates us to consider the worst approximation error (rate)

$$w := \max_{i:\Sigma_{ii}>0} \frac{\lambda}{\Sigma_{ii} h^\star + \lambda} = \frac{\lambda}{\Sigma_\star h^\star + \lambda} \in (0, 1)$$

where $\Sigma_\star := \min_{i:\Sigma_{ii}>0} \Sigma_{ii}$. If w is small, the approximation in (16) is tight, which implies that the root is close to $h^\star$. Hence, $1 - w$ represents the prior that the root is close to $h^\star$. We bisect at the new point $h := w h_\star + (1 - w) h^\star$. If $\varphi(h) \geq 0$, we use $h^{(0)} := h$ as the initial point for Newton's Method. Otherwise, we set $h^\star := h$ and repeat the argument. Algorithm 3 summarizes this procedure. Combining Algorithm 3 with Newton's method, we have Algorithm 4, which we call *Newton with Adaptive Bisection Starts* (Newton-ABS).

Algorithm 4 can be further optimized. For example, if $h^\star - h_\star$ is below some threshold (e.g. 0.1), then we may skip bisection entirely and start Newton's Method at $h^{(0)} = h_\star$. This is because the adaptive bisection may move too slowly if the range is too small. From experimentation, this happens relatively often. On a similar note, one may also enforce enough movement towards $h_\star$ by

Algorithm 3: Adaptive Bisection

Data: Σ, v, λ, ε

compute $h_\star$ that solves $\sum_{i=1}^{p}(\Sigma_{ii}h+\lambda)^2 \le \|v\|_1^2$ and take the positive part;

$h^\star \leftarrow \sqrt{\sum_{i:\Sigma_{ii}>0} \frac{v_i^2}{\Sigma_{ii}^2}}$;

$\Sigma_\star \leftarrow \min_{i:\Sigma_{ii}>0} \Sigma_{ii}$;

$w \leftarrow \frac{\lambda}{\Sigma_\star h^\star + \lambda}$;

$h \leftarrow w h_\star + (1-w)h^\star$;

while $\varphi(h) < 0$ *and* $|\varphi(h)| > \varepsilon$ **do**

  $h^\star \leftarrow h$;

  $w \leftarrow \frac{\lambda}{\Sigma_\star h^\star + \lambda}$;

  $h \leftarrow w h_\star + (1-w)h^\star$;

return h;

Algorithm 4: Newton with Adaptive Bisection Starts (Newton-ABS)

Data: Σ, v, λ, ε

$h \leftarrow$ result of Algorithm 3;

if $|\varphi(h)| > \varepsilon$ **then**

  $h \leftarrow$ result of Newton's Method starting at $h^{(0)} = h$;

$\beta^\star \leftarrow (\Sigma + \lambda h^{-1} I)^{-1} v$;

return $\beta^\star$;

taking the max of w with a minimal probability (e.g. 0.05). This will ensure that at least some proportion of $h_\star$ is taken if the prior is too strongly suggestive that the root is close to $h^\star$. The idea is that it is always better to overshoot towards $h_\star$ such that φ is non-negative, so that Newton's Method can quickly converge down to the root, than to slowly bisect to the root.

Figure 2 shows a plot of the Newton iterations for the vanilla Newton's Method from Section 2.3 and our proposed Newton-ABS. It is clear from the right panel that Newton's Method struggles where there is a sharp decay near the origin, since the Newton iterations slowly exit the kink. However, Newton-ABS gets around this problem by moving from the upper bound $h^\star$ towards the origin until φ is non-negative. Hence, Newton-ABS first travels on the tail of φ to find an initial point, which is very likely to lie on the tail as well.

Algorithm 4 is an effective way to leverage the power of both Newton Method and bisection. Similarly, there are other methods such as Dekker's Method and Brent's Method that combine many root-finding methods such as inverse quadratic interpolation, secant method, and bisection method to leverage their different merits [Brent, 2013, Dekker, 1969]. In Figure 3, we also include benchmark results of Brent's Method and find that Newton-ABS is superior. We refer to the full benchmark comparison to Section 3.1.

3 Benchmark

In this section, we study a wide array of benchmarks.

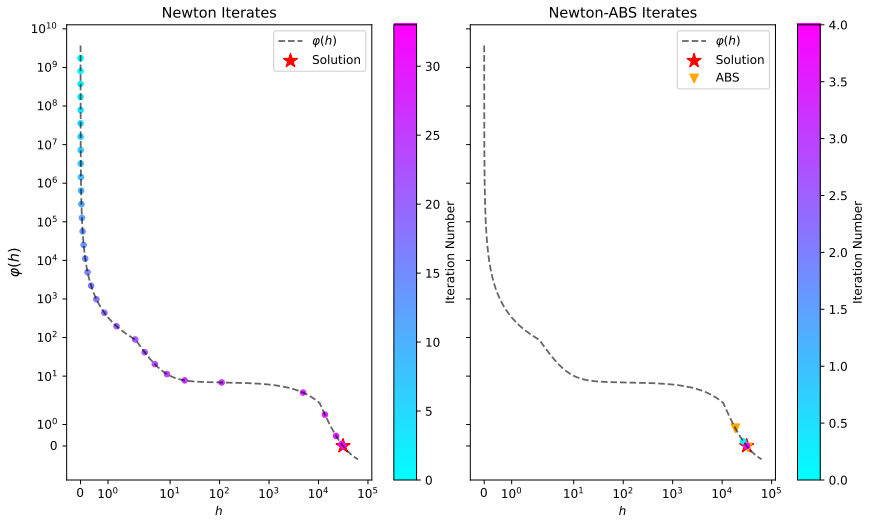

Figure 2: Plot of Newton iterations for Newton's Method and Newton-ABS. It is clear from the right panel that Newton's Method struggles when there is a sharp decay near the origin, since the Newton iterations slowly exit the kink. Newton-ABS goes around this problem by finding a good initial point sufficiently away from the origin.

3.1 Block Update Comparison

In this section, we compare the algorithms presented in Sections 2.2 to 2.4 that solve the block update (7). Recall the objective to solve is

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \ \frac{1}{2}\beta^\top \Sigma \beta - v^\top \beta + \lambda \|\beta\|_2$$

Figure 3 shows a set of comparisons Newton, Brent, and Newton-ABS (see Sections 2.3 and 2.4). Although we also ran ISTA, FISTA, and FISTA with adaptive restart (FISTA-ADA) (see Section 2.2), they failed to converge, so we omit their benchmark results. All Newton methods properly converged. For all scenarios, the diagonal of Σ was generated uniformly from $[0, 1]$. In addition,

(b) **Almost PSD:** 1% of the entries were regenerated uniformly from $[1 \times 10^{-14}, 1 \times 10^{-8}]$.

(c) **Very PSD:** 20% of the entries were set exactly to zero and 10% were regenerated from $[1 \times 10^{-14}, 1 \times 10^{-8}]$.

The vector v was generated from $\mathcal{N}(0, \Sigma)$. Figures (a) to (c) used $\lambda = 1 \times 10^{-1}$ while (d) used $\lambda = 1 \times 10^{-4}$. Both Newton-ABS and Brent methods are asymptotically faster than Newton, however, Brent has the poorest performance for small to moderate p (≤ 100). Although, Brent performs slightly better than Newton-ABS when D is positive definite, we see that Newton-ABS is faster than Brent uniformly over p when D is PSD by a larger margin. Newton-ABS is up to 7 times faster than Newton and up to 4 times faster than Brent. Given the robustness of Newton-ABS in speed and convergence across the different configurations, we expect significant improvement to the overall optimizer using Newton-ABS.

3.2 Benchmark against Other Packages

Figure 4 shows a comparison of our package against other packages: `grpreg`, `grplasso`, `sparsegl`, and `gglasso`. We generate $X \in \mathbb{R}^{n \times p}$ where $n = 100$ and $p \in \{2^4, \ldots, 2^{20}\}$ and each entry is drawn

James: name?

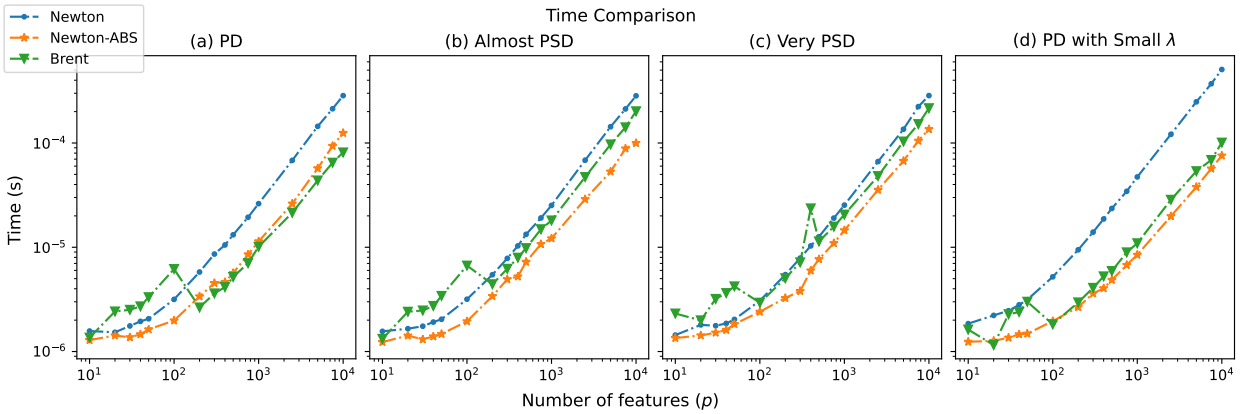

Figure 3: Plot of runtimes for Newton, Newton-ABS, and Brent's Method for various input configurations. Figures (a) to (c) modifies the singularity of D from positive definite to positive semi-definite. Figure (d) changes the regularization level to a smaller value. Overall, Newton-ABS is generally faster than Newton and Brent. Newton-ABS is up to 7 times faster than Newton and up to 4 times faster than Brent.

from $\mathcal{N}(0,1)$. The true coefficient vector β is generated from Unif$(-1,1)$ with 95% of the entries set to zero randomly. Finally, we generate y from the linear model: $y = X\beta + \epsilon$ where $\epsilon \sim \mathcal{N}(0, I_n)$. The groups are generated with a group size of 10 by grouping consecutive coefficients with the possible exception of the last group, which may contain any remaining coefficients. We center both X and y, and scale each column of X such that its norm is 1. The regularization path was kept the same across all packages for each data generation. `grpreg` produced an error for $p \geq 2^{19}$ and `grplasso` took too long for large p, so we omit these results. All packages converged, however, `grpreg` generally had a larger objective. We clearly see that our package performs significantly faster, especially with large p. With $p = 2^{20}$ (approximately 1 million features), our package fits a path of about 55 regularization values in **3 seconds**, while the next fastest package, `sparsegl`, runs in **13 seconds**, and the next fastest, `gglasso`, runs in **27 seconds**.

James: name?

James: name?

References

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. doi: 10.1137/080716542. URL https://doi.org/10.1137/080716542.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Richard P Brent. *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

Theodorus Jozef Dekker. Finding a zero by means of successive linear interpolation. *Constructive aspects of the fundamental theorem of algebra*, pages 37–51, 1969.

Trevor Hastie, Rob Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity*. Chapman & Hall, 2016.

J. Klosa, N. Simon, P.O. Westermark, V. Liebscher, and D. Wittenburg. Seagull: lasso, group lasso and sparse-group lasso regularization for linear regression models via proximal gradient descent. *BMC Bioinformatics*, 21(407), 2020. doi: 10.1186/s12859-020-03725-w.

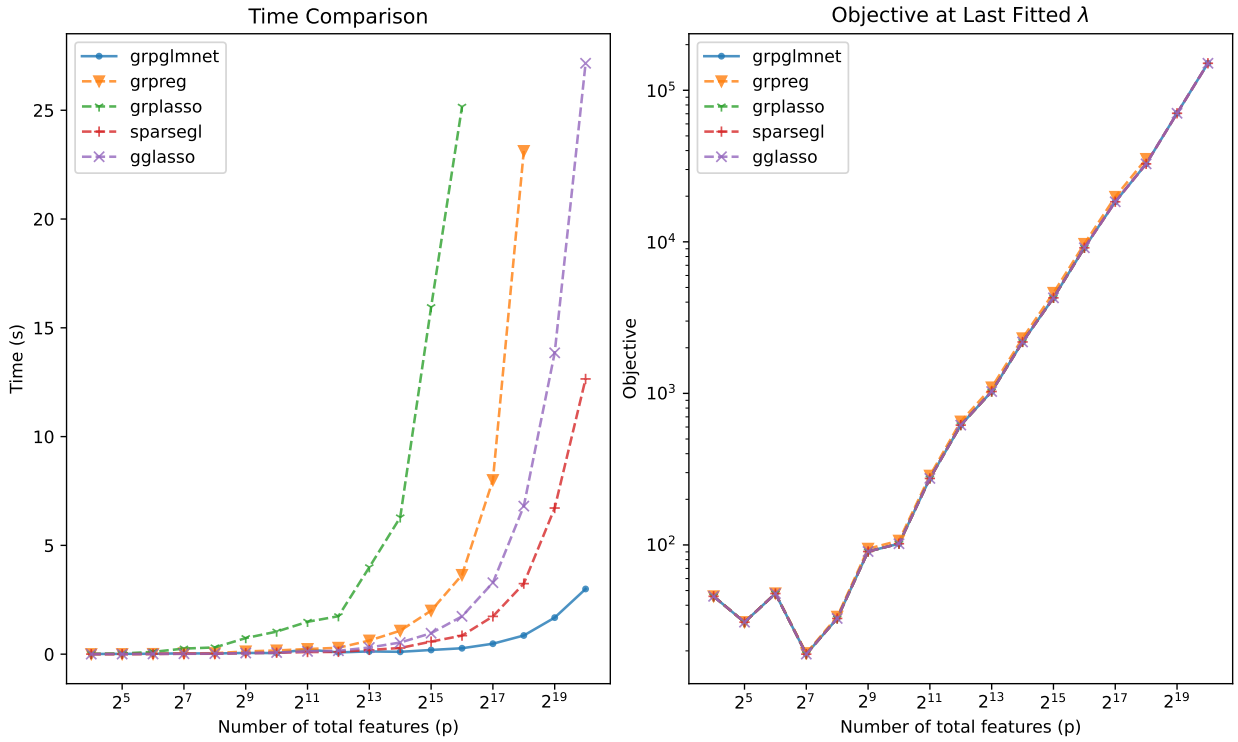

Figure 4: Plot of runtimes and objective values for our package against other packages. The left plot shows that all packages perform well for small values of p, however, we quickly see a noticeable difference as p grows. All packages converged as shown on the right plot.

Xiaoxuan Liang, Aaron Cohen, Anibal Solón Heinsfeld, Franco Pestilli, and Daniel J. McDonald. sparsegl: An r package for estimating sparse group lasso, 2022. URL https://arxiv.org/abs/2208.02942.

Ignace Loris. On the performance of algorithms for the minimization of ℓ_1-penalized functionals. *Inverse Problems*, 25(3):035008, jan 2009. doi: 10.1088/0266-5611/25/3/035008. URL https://dx.doi.org/10.1088/0266-5611/25/3/035008.

Lukas Meier, Sara Van De Geer, and Peter Böhlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society B*, 70(1):53–71, 2008. doi: 10.1111/j.1467-9868.2007.00627.x.

Brendan O'Donoghue and Emmanuel Candés. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15:715–732, 2015. doi: https://doi-org.stanford.idm.oclc.org/10.1007/s10208-013-9150-3.

Noah Simon and Robert Tibshirani. Standardization and the group lasso penalty. *Statistica Sinica*, 22(3):983–1001, 2012. ISSN 10170405, 19968507. URL http://www.jstor.org/stable/24309971.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:475–494, 2001. doi: https://doi-org.stanford.idm.oclc.org/10.1023/A:1017501703105.

Jie Wang, Peter Wonka, and Jieping Ye. Lasso screening rules via dual polytope projection. *Journal of Machine Learning Research*, 16(32):1063–1101, 2015. URL http://jmlr.org/papers/v16/wang15a.html.

Stephen J. Wright, Robert D. Nowak, and MÁrio A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009. doi: 10.1109/TSP.2009.2016892.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society B*, 68(1):49–67, 2006. doi: 10.1111/j.1467-9868.2005.00532.x.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005. ISSN 13697412, 14679868. URL http://www.jstor.org/stable/3647580.

A Newton Method Results

A.1 Properties of Root-Finding Function

Let the function to root-find, φ, be as in (11). Upon differentiating φ,

$$\frac{d\varphi(h)}{dh} = -2\sum_{i=1}^{p} \frac{v_i^2 \Sigma_{ii}}{(\Sigma_{ii}h + \lambda)^3} \leq 0$$

where the inequality is strict if and only if not all $v_i^2\Sigma_{ii} = 0$. Note that if all $v_i^2\Sigma_{ii} = 0$, then we must have that $0 = \|v\|_2 \leq \lambda$. This is because by assumption $v = D^\top u$ for some $u \in \mathbb{R}^p$ (see (7)), so that if $v_i^2\Sigma_{ii} = 0$, either $v_i = 0$ or $\Sigma_{ii} = 0 \implies v_i = 0$. Hence, without loss of generality, we may assume that, in particular, not all $\Sigma_{ii} = 0$ and φ' is strictly negative so that φ is strictly decreasing. Further, we have that

$$\varphi(h) = \sum_{i:\Sigma_{ii}>0} \frac{v_i^2}{(\Sigma_{ii}h + \lambda)^2} - 1$$

Consequently, since $\varphi(0) > 0$ by hypothesis and $\lim_{h\to\infty} \varphi(h) = -1$, there exists a (unique) root. Further, it is easy to see that φ is convex since it is a sum of convex functions.

A.2 Convergence of Newton Method

Since φ is convex, this suggests solving (10) via a one-dimensional Newton's Method. Specifically, with $h^{(0)}$ as the initial starting point, for $k \geq 1$,

$$h^{(k)} = h^{(k-1)} - \frac{\varphi(h^{(k-1)})}{\varphi'(h^{(k-1)})} \tag{15}$$

We claim that Newton's Method is guaranteed to converge for any initial point $h^{(0)}$. Indeed, for every $k \geq 1$, by convexity of φ, assuming $h^{(k)} \in [0, \infty)$,

$$\varphi(h^{(k)}) \geq \varphi(h^{(k-1)}) + \varphi'(h^{(k-1)})(h^{(k)} - h^{(k-1)}) = 0$$

Along with (15), this shows that $h^{(k)}$ is an increasing sequence for $k \geq 1$ and bounded above by $h^\star$, the root of φ, by monotonicity. Hence, $h^{(k)}$ converges to some limit $h^{(\infty)}$. From (15), taking limit as $k \to \infty$ and using that φ' is non-zero,

$$h^{(\infty)} = h^{(\infty)} - \frac{\varphi(h^{(\infty)})}{\varphi'(h^{(\infty)})} \implies \varphi(h^{(\infty)}) = 0$$

which shows that $h^{(\infty)}$ is the root of φ.

B Newton-ABS Results

B.1 Lower and Upper Bounds on the Root

In this section, we derive the lower and upper bounds on the root, $h_\star$, $h^\star \in [0, \infty)$, as discussed in Section 2.4. Specifically, we show that $\varphi(h_\star) \geq 0 \geq \varphi(h^\star)$, which implies that the root lies in $[h_\star, h^\star]$, since φ is (without loss of generality) strictly decreasing (Appendix A.1).

We first begin with the lower bound $h_\star$. Recall that we wish to relax the problem (13). Let $a(h), b(h) \in \mathbb{R}^p$ be defined by $a_k(h) := \Sigma_{kk} h + \lambda$ and $b_k(h) := \frac{|v_k|}{a_k(h)}$ for each $k = 1, \ldots, p$. Then, by Cauchy-Schwarz,

$$\|v\|_1 := \sum_k |v_k| = \sum_k a_k(h) b_k(h) \leq \|a(h)\|_2 \, \|b(h)\|_2$$

Hence, if $\|a(h)\|_2 \leq \|v\|_1$, then $\varphi(h) \geq 0$. We see that $\|a(h)\|_2 \leq \|v\|_1$ if and only if

$$\sum_{i=1}^p (\Sigma_{ii} h + \lambda)^2 \leq \|v\|_1^2$$

This inequality can be solved for h using the quadratic formula. Let $\tilde{h}$ be the solution. Then, letting $h_\star := \max(\tilde{h}, 0)$, we have that $\varphi(h_\star) \geq 0$.

Next, we derive the upper bound $h^\star$. Similar to the lower bound, we approximate (14). Since

$$\sum_{i=1}^p \frac{v_i^2}{(\Sigma_{ii} h + \lambda)^2} = \sum_{i:\Sigma_{ii}>0} \frac{v_i^2}{(\Sigma_{ii} h + \lambda)^2} \leq h^{-2} \sum_{i:\Sigma_{ii}>0} \frac{v_i^2}{\Sigma_{ii}^2} \tag{16}$$

by setting

$$h^\star := \sqrt{\sum_{i:\Sigma_{ii}>0} \frac{v_i^2}{\Sigma_{ii}^2}}$$

we have that $\varphi(h^\star) \leq 0$.

C EDPP Derivations

Recall that the group lasso problem in full generality is given by

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \; \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \sum_{i=1}^G w_i \left(\alpha \|\beta_i\|_2 + \frac{1-\alpha}{2} \|\beta_i\|_2^2 \right) \tag{17}$$

We assume that $\alpha \in (0,1)$ and all $w_i > 0$. If $\alpha = 0$, then there is no point of discussing ways to remove variables since all variables must be active. If $\alpha = 1$, then the EDPP rule is given by Wang et al. [2015, Corollary 24]. If $w_i = 0$ for some i, then we may first optimize over such β_i, which is an unpenalized optimization problem whose solutions do not depend on λ. Afterwards, we may replace y by the residual and X by the sub-matrix containing only the group features whose corresponding $w_i > 0$.

Let $z = y - X\beta$ so that (17) becomes

$$\underset{\beta \in \mathbb{R}^p, z \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2} \|z\|_2^2 + \lambda \sum_{i=1}^G w_i \left(\alpha \|\beta_i\|_2 + \frac{1-\alpha}{2} \|\beta_i\|_2^2 \right) \tag{18}$$

$$\text{subject to} \quad z = y - X\beta \tag{19}$$

We introduce the dual variables $\eta \in \mathbb{R}^n$ such that the Lagrangian of (19) is given by

$$L(\beta, z, \eta) := \frac{1}{2} \|z\|_2^2 + \lambda \sum_{i=1}^G w_i \left(\alpha \|\beta_i\|_2 + \frac{1-\alpha}{2} \|\beta_i\|_2^2 \right) + \eta^\top (y - X\beta - z)$$

Thus, the dual function $g(\eta)$ is given by

$$\begin{aligned} g(\eta) &:= \inf_{\beta, z} L(\beta, \eta) \\ &= \eta^\top y + \inf_{\beta} \left(-\eta^\top X\beta + \lambda \sum_{i=1}^G w_i \left(\alpha \|\beta_i\|_2 + \frac{1-\alpha}{2} \|\beta_i\|_2^2 \right) \right) + \inf_z \left(\frac{1}{2} \|z\|_2^2 - \eta^\top z \right) \end{aligned}$$

We now solve for the two minimization problems. First, define

$$\begin{aligned} \varphi(\beta) &:= -\eta^\top X\beta + \lambda \sum_{i=1}^G w_i \left(\alpha \|\beta_i\|_2 + \frac{1-\alpha}{2} \|\beta_i\|_2^2 \right) \\ &= \sum_{i=1}^G \left(-\eta^\top X_i \beta_i + \lambda w_i \left(\alpha \|\beta_i\|_2 + \frac{1-\alpha}{2} \|\beta_i\|_2^2 \right) \right) =: \sum_{i=1}^G \varphi_i(\beta_i) \end{aligned}$$

Since φ is separable, we minimize each φ_i separately. For any fixed i, the optimal point $\beta_i^\star$ for minimizing φ_i satisfies

$$-X_i^\top \eta + \lambda \alpha w_i u_i + \lambda (1-\alpha) w_i \beta_i = 0 \tag{20}$$

where

$$u_i \in \begin{cases} \left\{ \frac{\beta_i}{\|\beta_i\|_2} \right\}, & \beta_i \neq 0 \\ \{ u \in \mathbb{R}^{p_i} : \|u\|_2 \leq 1 \}, & \beta_i = 0 \end{cases}$$

If $\beta_i^\star = 0$, note that $\varphi_i(\beta_i^\star) = 0$. Otherwise,

$$\begin{aligned} \varphi_i(\beta_i) &= \beta_i^\top \left(-X_i^\top \eta + \lambda \alpha w_i \frac{\beta_i}{\|\beta_i\|_2} + \lambda \frac{1-\alpha}{2} w_i \beta_i \right) \\ \implies \varphi_i(\beta_i^\star) &= -\lambda \frac{1-\alpha}{2} w_i \|\beta_i^\star\|_2^2 \end{aligned}$$

Now, recall that we also have from (20) that

$$\begin{aligned}\lambda w_i \left(\frac{\alpha}{\|\beta_i^\star\|_2} + (1-\alpha) \right) \beta_i^\star &= X_i^\top \eta \\ \implies \|\beta_i^\star\|_2 &= \frac{\left\|X_i^\top \eta\right\|_2}{\lambda(1-\alpha)w_i} - \frac{\alpha}{1-\alpha} \\ &= \frac{\left\|X_i^\top \eta\right\|_2 - \lambda\alpha w_i}{\lambda(1-\alpha)w_i}\end{aligned}$$

Note that this also gives us

$$\beta_i^\star = \frac{1}{\lambda(1-\alpha)w_i} \left(1 - \frac{\lambda\alpha w_i}{\left\|X_i^\top \eta\right\|_2} \right) X_i^\top \eta$$

Hence, in either case, we have that

$$\varphi_i(\beta_i^\star) = -\frac{\left(\left\|X_i^\top \eta\right\|_2 - \lambda\alpha w_i\right)_+^2}{2\lambda(1-\alpha)w_i}$$

Note that unlike when $\alpha = 1$, we do not have any constraints on η when $\alpha \in (0, 1)$. This shows that

$$\inf_\beta \varphi(\beta) = -\frac{1}{2\lambda(1-\alpha)} \sum_{i=1}^G \frac{\left(\left\|X_i^\top \eta\right\|_2 - \lambda\alpha w_i\right)_+^2}{w_i}$$

Next,

$$\inf_z \frac{1}{2} \|z\|_2^2 - \eta^\top z = -\frac{1}{2} \|\eta\|_2^2$$

Combining, the dual function simplifies to

$$g(\eta) = \eta^\top y - \frac{1}{2} \|\eta\|_2^2 - \frac{1}{2\lambda(1-\alpha)} \sum_{i=1}^G \frac{\left(\left\|X_i^\top \eta\right\|_2 - \lambda\alpha w_i\right)_+^2}{w_i}$$

So, the dual formulation is

$$\underset{\eta}{\text{maximize}} \ \eta^\top y - \frac{1}{2} \|\eta\|_2^2 - \frac{1}{2\lambda(1-\alpha)} \sum_{i=1}^G \frac{\left(\left\|X_i^\top \eta\right\|_2 - \lambda\alpha w_i\right)_+^2}{w_i}$$

which is equivalent to

$$\underset{\eta}{\text{maximize}} \ \frac{1}{2} \|y\|_2^2 - \frac{1}{2} \|\eta - y\|_2^2 - \frac{1}{2\lambda(1-\alpha)} \sum_{i=1}^G \frac{\left(\left\|X_i^\top \eta\right\|_2 - \lambda\alpha w_i\right)_+^2}{w_i}$$

By a rescaling of the dual variables η such that $\theta = \lambda^{-1}\eta$, the dual formulation can be written as

$$\underset{\theta}{\text{maximize}} \ \frac{1}{2} \|y\|_2^2 - \frac{\lambda^2}{2} \left\|\theta - \frac{y}{\lambda}\right\|_2^2 - \frac{\lambda}{2(1-\alpha)} \sum_{i=1}^G \frac{\left(\left\|X_i^\top \theta\right\|_2 - \alpha w_i\right)_+^2}{w_i} \tag{21}$$

Algorithm 5: Parallel Block Coordinate Descent

Data: $x^{(0)}$

for $k = 0, 1, \ldots$ **do**
  for $i = 1, \ldots, G$ *in parallel* **do**
    $y_i^{(k)} \leftarrow \underset{x_i}{\operatorname{argmin}} f(x_1^{(k)}, \ldots, x_i, \ldots, x_G^{(k)})$;
  for $i = 1, \ldots, G$ *in parallel* **do**
    compute $\alpha_i^{(k)}$;
    $x_i^{(k+1)} \leftarrow (1 - \alpha_i^{(k)}) x_i^{(k)} + \alpha_i^{(k)} y_i^{(k)}$;
return $x^\star$;

Following the proof in Wang et al. [2015, Appendix B.2], the KKT conditions are given by

$$y = X\beta^\star(\lambda) + \lambda\theta^\star(\lambda)$$
$$X_i^\top \theta^\star(\lambda) - (1-\alpha) w_i \beta_i^\star(\lambda) \in \begin{cases} \left\{ \alpha w_i \frac{\beta_i^\star(\lambda)}{\|\beta_i^\star(\lambda)\|_2} \right\}, & \beta_i^\star(\lambda) \neq 0 \\ \{\alpha w_i u : \|u\|_2 \leq 1\}, & \beta_i^\star(\lambda) = 0 \end{cases}$$

It is difficult to devise a safe screening rule because of the penalty in (21). Specifically, if $\alpha < 1$, then we have an unconstrained problem that tends to pull more coefficients to be non-zero. The penalty becomes a bona fide barrier function when $\alpha \uparrow 1$. This is precisely the case that Wang et al. [2015] studies. When $\alpha = 1$, the maximization can only occur non-trivially if $\left\|X_i^\top \theta\right\| \leq \lambda w_i$. This constraint also agrees with the derivation in Wang et al. [2015]. In such a case, the maximization problem is simply a projection onto a closed, non-empty convex set, which was the critical observation. This cannot generalize when $\alpha < 1$.

D Parallel Block Coordinate Descent

In this section, we describe a new algorithm for minimizing a loss function $f : \mathbb{R}^p \to \overline{\mathbb{R}}$ of the form

$$f(x) = f_0(x) + \sum_{i=1}^{G} f_i(x_i) \tag{22}$$

where $f_0 : \mathbb{R}^p \to \overline{\mathbb{R}}$ is a smooth function, $f_i : \mathbb{R}^{p_i} \to \overline{\mathbb{R}}$ is any arbitrary (measurable) function that only depends on a block x_i, and $x = (x_1, \ldots, x_G) \in \mathbb{R}^p$ where $x_i \in \mathbb{R}^{p_i}$. Note that we are in the same setup as in Tseng [2001].

We describe our algorithm for parallel block coordinate descent given in Algorithm 5. Suppose at step $k \geq 0$, we are given a point $x^{(k)}$. The usual cyclic coordinate descent iterates through each block of coefficients *sequentially* in the following manner:

$$x_i^{(k+1)} = \underset{x_i}{\operatorname{argmin}} f(x_1^{(k+1)}, \ldots, x_i, \ldots, x_G^{(k)})$$

where the current block i is optimized while fixing other blocks at their current updates. This dependence across the block updates makes this algorithm impossible to parallelize. However, with a slight modification to the algorithm, it becomes easily parallelizable.

Consider the following new update scheme:

$$y_i^{(k)} = \underset{x_i}{\operatorname{argmin}}\, f(x_1^{(k)}, \ldots, x_i, \ldots, x_G^{(k)}) \tag{23}$$

$$x_i^{(k+1)} = (1 - \alpha_i^{(k)}) x_i^{(k)} + \alpha_i^{(k)} y_i^{(k)} \tag{24}$$

where we let $\alpha_i^{(k)} \in [0, 1]$ be any set of weights for now. Note the key difference that $y_i^{(k)}$ as in (23) can be computed *in parallel*, since we fix all other coordinates at $x^{(k)}$ rather than at their new updates. To give convergence guarantees, we must impose some conditions on f and $\alpha_i^{(k)}$, however, these conditions hold for the group lasso objective (4). More generally, Theorem D.1 shows that every cluster point of $\{x^{(k)}\}_{k=0}^{\infty}$ converges to a coordinate-wise minimum point under some mild conditions.

Theorem D.1 (Properties of Parallel BCD)**.** *Consider the sequences* $\{x^{(k)}\}_{k=0}^{\infty}$ *and* $\{y^{(k)}\}_{k=0}^{\infty}$ *defined by* (23) *and* (24) *where* $\alpha_i^{(k)} \in [0, 1]$*. Assume that* f *is convex. Suppose also that* $\sum_{i=1}^{G} \alpha_i^{(k)} = 1$ *for each* $k \geq 0$*. Then, we have the following guarantees:*

(i) $x^{(k)}$ *is a bona fide descent, i.e.* $f(x^{(k+1)}) \leq f(x^{(k)})$*.*

(ii) Suppose that for each i, $\liminf_{k\to\infty} \alpha_i^{(k)} > 0$*. Assume that the level sets* $\mathcal{L}_a := \{x : f(x) \leq f(a)\}$ *are compact. Then, every cluster point of* $\{x^{(k)}\}_{k=0}^{\infty}$ *is a coordinate-wise minimum point. That is, if* $x^{(n_k)} \to x^\star$ *along a subsequence* n_k*, then for any* $i = 1, \ldots, G$ *and* $d_i \in \mathbb{R}^{p_i}$*,*

$$f(x^\star) \leq f(x^\star + \pi_i(d_i))$$

where $\pi_i(d_i) := (0, \ldots, d_i, \ldots, 0)$*.*

Proof. Let $\pi_i : \mathbb{R}^{p_i} \to \mathbb{R}^p$ such that $\pi_i(d) = (0, \ldots, d, \ldots, 0)$ so that $d \in \mathbb{R}^{p_i}$ is embedded in the ith block.

We first prove (i). Fix any $k \geq 0$ and consider $x^{(k)}$. By definition of $y^{(k)}$, we have for each $i = 1, \ldots, G$ that

$$f(x^{(k)} + \pi_i(y_i^{(k)} - x_i^{(k)})) \leq f(x^{(k)})$$

Hence, by hypothesis that $\sum_{i=1}^{G} \alpha_i^{(k)} = 1$,

$$\sum_{i=1}^{G} \alpha_i^{(k)} f(x^{(k)} + \pi_i(y_i^{(k)} - x_i^{(k)})) \leq f(x^{(k)})$$

Next, by convexity of f,

$$f\left(x^{(k)} + \sum_{i=1}^{G} \alpha_i^{(k)} \pi_i(y_i^{(k)} - x_i^{(k)})\right) \leq \sum_{i=1}^{G} \alpha_i^{(k)} f(x^{(k)} + \pi_i(y_i^{(k)} - x_i^{(k)})) \leq f(x^{(k)})$$

Upon inspection, we see that

$$x^{(k+1)} = x^{(k)} + \sum_{i=1}^{G} \alpha_i^{(k)} \pi_i(y_i^{(k)} - x_i^{(k)})$$

Hence, we have shown that $f(x^{(k+1)}) \leq f(x^{(k)})$, that is, we have a descent method.

We now prove (ii). Note that since the level sets are compact and (i) has been established, $x^{(k)}$ is bounded. Consider any convergent subsequence $x^{(n_k)} \to x^\star$, which must be finite.

We first begin with an inequality that holds for any $k \geq 0$. For any $j = 1, \ldots, G$ and $d_j \in \mathbb{R}^{p_j}$,

$$\begin{aligned} f(x^{(k+1)}) &= f\left(x^{(k)} + \sum_{i=1}^G \alpha_i^{(k)} \pi_i(y_i^{(k)} - x_i^{(k)})\right) \\ &\leq \sum_{i=1}^G \alpha_i^{(k)} f(x^{(k)} + \pi_i(y_i^{(k)} - x_i^{(k)})) \\ &= \sum_{i \neq j} \alpha_i^{(k)} \underbrace{f(x^{(k)} + \pi_i(y_i^{(k)} - x_i^{(k)}))}_{\leq f(x^{(k)})} + \alpha_j^{(k)} \underbrace{f(x^{(k)} + \pi_j(y_j^{(k)} - x_j^{(k)}))}_{\leq f(x^{(k)} + \pi_j(d_j))} \\ &\leq (1 - \alpha_j^{(k)}) f(x^{(k)}) + \alpha_j^{(k)} f(x^{(k)} + \pi_j(d_j)) \end{aligned} \tag{25}$$

where we used convexity and that $y_i^{(k)}$ minimizes along the ith coordinate with all other coordinates fixed at $x^{(k)}$.

Now, since $f(x^{(k)})$ is monotonically decreasing by (i) and $n_{k+1} \geq n_k + 1$, we have that

$$f(x^{(n_{k+1})}) \leq f(x^{(n_k+1)}) \leq f(x^{(n_k)})$$

Since $k \mapsto f(x^{n_k+1})$ is also monotonically decreasing, taking limit as $k \to \infty$ shows that

$$f(x^\star) \leq \lim_{k \to \infty} f(x^{(n_k+1)}) \leq f(x^\star)$$

by continuity of f, which is implied by convexity.

By possibly passing through a further subsequence of n_k, we may assume without loss of generality that $\alpha_i^{(n_k)}$ converges to a finite limit $\alpha_i^\star$ for every $i = 1, \ldots, G$. Since $\liminf_{k \to \infty} \alpha_i^{(k)} > 0$, we have that $\alpha_i^\star > 0$ for every $i = 1, \ldots, G$. Then, taking limit as $k \to \infty$ in (25) restricted to the subsequence n_k,

$$\begin{aligned} & f(x^\star) \leq (1 - \alpha_j^\star) f(x^\star) + \alpha_j^\star f(x^\star + \pi_j(d_j)) \\ \implies & f(x^\star) \leq f(x^\star + \pi_j(d_j)) \end{aligned}$$

Since this is true for any j and d_j, by definition, we have that $x^\star$ is a coordinate-wise minimum point. □

We emphasize that in Theorem D.1, one can choose $\alpha_i^{(k)}$ in *any fashion* so long as the conditions are satisfied. In particular, one may choose $\alpha_i^{(k)}$ *adaptively* based on the current iterate $x^{(k)}$. It is quite easy to satisfy the condition that $\liminf_{k \to \infty} \alpha_i^{(k)} > 0$. For example, one could arbitrarily set $\alpha_i^{(k)}$ uniformly bounded away from 0.

Turning to our original discussion of the group lasso objective (4), we give a convergence guarantee using Theorem D.1 in Corollary D.1.1.

Corollary D.1.1 (Convergence of Parallel BCD for Group Lasso)**.** *Let $\alpha_i^{(k)} \in [0, 1]$ be such that $\liminf_{k \to \infty} \alpha_i^{(k)} > 0$ for every $i = 1, \ldots, G$ and $\sum_{i=1}^G \alpha_i^{(k)}$ for all $k \geq 0$. If the group lasso objective (4) admits a unique minimizer, then the parallel BCD iterates $\{x^{(k)}\}$ given by (23) and (24) converges to the global minimium.*

Proof. Note that the group lasso objective (4) is indeed convex and has compact level sets. Theorem D.1 then shows that every cluster point of $\{x^{(k)}\}_{k=0}^{\infty}$ is a coordinate-wise minimum point. Following Tseng [2001, Lemma 3.1], since the objective also satisfies the separability condition (22) with a differentiable component f_0, the objective is *regular*. Hence, every coordinate-wise minimum point z is a stationary point, that is, $f'(z;d) \geq 0$ for all $d \in \mathbb{R}^p$. By convexity, we have that z is a global minimum. As soon as the group lasso objective admits a unique minimizer, the sequence $\{x^{(k)}\}_{k=0}^{\infty}$ then converges to the global minimum.

□