FAST AND SCALABLE SOLVERS FOR PENALIZED REGRESSION WITH SPARSITY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF STATISTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

James Yang
March 2025

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Trevor Hastie, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dominik Rothenhaeusler

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jonathan Taylor

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format.

Preface

In recent years, statistical estimation and inference methods have grown increasingly complex with the advent of large data, relying on large computations and solving non-trivial optimization problems. Such methods arise in a wide array of applications such as genomics, healthcare, neural network theory, and signal processing. It is often imperative that these methods are fast and scalable to ultimately become useful tools for practitioners. In this thesis, we focus on a particular aspect of estimation methods, namely, sparsity-inducing penalized regression problems with a heavy emphasis on the group lasso problem, primarily geared towards developing algorithms that achieve high-performance speed. Speed can come in different flavors such as algorithmic design, software engineering, heuristics, and theoretical results. We discuss several different works that contain a mix of these strategies that coalesce into reliable algorithms that achieve stunning speed in practice. Chapter 2 introduces the most foundational work of the thesis that establishes a novel algorithm for solving the group lasso and elastic net. Chapter 3 develops a heuristic for discarding features in the group lasso solver that is more robust than the state-of-the-art method. Chapter 4 extends the group lasso solver by allowing group-wise linear inequality constraints, which requires a considerable change to the original algorithm. Finally, Chapter 5 develops a theoretical framework for studying the existence of a sparse solution to a general polyhedron constrained least squares problem. This leads us to a fast high-dimensional non-negative least squares solver, which is a fundamental tool in our constrained group lasso solver.

Acknowledgments

I am immensely grateful and deeply indebted to my advisor, Trevor Hastie, for his priceless wisdom, patience, and guidance. I came into this Ph.D. wanting to build a thesis comprised of a mix of theoretical and computational works, and in this regard, I can say with certainty that I have fulfilled my desire thanks to his perfect mentorship. As someone who also cares deeply about the computational aspect of research, Trevor always encouraged me to highlight my strengths in this area. However, he never ceased to push me to improve in other areas such as developing refined statistical (and sometimes mathematical) intuitions and creating narratives with convincing applications. Through his guidance over the years, he has shaped me to become a well-rounded and, most importantly, a practical researcher. Trevor has always been an incredibly supportive, dedicated, and caring advisor, and I will never forget the lasting impact he made in my life both academically and personally. Thank you again, Trevor, for making the last five years the most memorable and cherished years of my life.

Next, I thank my committee members Rob Tibshirani, Jonathan Taylor, Dominik Rothenhäusler, and Mert Pilanci. I also thank other collaborators from my time at Stanford: Zihuai He, Alexander Ioannidis, Naras Balasubramanian, Michael Sklar, and Ben Thompson. I have worked with every one of them to some capacity and I am extremely grateful for those opportunities as well as their invaluable inputs on my work throughout the years.

My Ph.D. journey at Stanford would not have been possible without my mentors from my undergraduate at Columbia University. I thank Professor Philip Protter, Arian Maleki, and Cynthia Rush for instilling a love for statistics early on. I thank Xiang Liu whom I worked closely with in the summer of 2018 on my first statistics research project and gave me my first exposure to developing a methodology, proving theoretical guarantees, and writing serious software for usability, which solidified my interest in the type of work I wished to pursue in my Ph.D.. I also thank Muye Wang, Steven Yin, and Seungki Min, who were Ph.D. students at the time, for their career advice and helping me prepare and navigate through the Ph.D. application process.

I thank my peers in the statistics department at Sequoia Hall for the wonderful discussions about statistics that certainly helped me become a more rigorous thinker, the random trips to the gym to show-off our other prowesses, and the banters that kept the fun and laughter running in the

department. Some of these people include Anav Sood, Kevin Fry, Tim Morrison, John Cherian, Tim Sudijono, Michael Howes, Yash Nair, Aditya Ghosh, Etaash Katiyar, Rex Shen, Jack Krew, Joon Lee, Henry Smith, Ginnie Ma, and Valérie Ho. I especially thank Anav Sood and Kevin Fry for helping me both in statistics and in life. They are some of the most genuine, honest, intelligent, and humorous people and I aspire to be like them in many ways.

A special shoutout to Susie Ementon, our student services officer. With probability 1, I would have incorrectly filled out some important form or missed a deadline had it not been for her meticulousness and attentiveness. Sequoia would absolutely not function without her.

I thank my partner Irene Kyoung for making my life so much more colorful and complete. She is one of the greatest things that happened during my Ph.D., and I am truly fortunate to have such a special person beside me. The numerous times she took care of me, especially when work became difficult, and showed extreme patience with my shortcomings will certainly be remembered as a testament to her pure heart and unconditional love for me. She inspires me to be a better person and partner, and I hope I bring the same happiness and love that she has gifted me with.

Lastly, I thank my mom, dad, and sister for their unending support and love. Throughout my life, I shared so many memorable moments with my sister playing video games, watching shows, eating Mr. Jinn's ice cream, and finding new hobbies to try. These moments are forever blended with my time at Stanford, and I am truly grateful for them. For most of my childhood, my parents invested a lot of their time and money into developing my musical career with no hesitation. As a glimpse into my past, my dad even changed his job (and industry) to financially support this decision; my mom drove me to lessons every week, cooked Michelin-star-level food every day, and tirelessly researched ways to help me succeed as a solo violinist. Despite these efforts, my musical career was unrealized and I ultimately decided step into the world of academics. Fast-forwarding 9 years, I will now work in the same industry as my dad, waving my new badge of honor; I can now drive my mom to Michelin-star restaurants where she will now be the one chauffeured and served divine food. I am forever grateful for everything they have given me and I dedicate all my successes to them. I hope I made you proud, mom and dad.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the last few decades, the landscape of machine learning changed drastically with the exponential growth of data in both the number of samples n and features p. Most regression models – the prototypical one being ordinary least squares (OLS) – favor the increase in n as they tend to generalize better. Moreover, the computational cost of large n can be easily ameliorated via distributed computing. However, as p grows, they typically suffer from overfitting, large memory consumptions, and increased computational complexity. The standard approach to tackle the overfitting issue is regularization. Many different regularizations exist today and some notable ones include the ridge, lasso [Friedman et al., 2010], elastic net [Zou and Hastie, 2005], and group lasso [Yuan and Lin, 2006]. The ridge adds the ℓ_2 penalty $\|\beta\|_2^2$ to the regression objective while the lasso adds the ℓ_1 penalty $\|\beta\|_1$. The ridge solution typically remains dense, however, the lasso often enjoys sparsity in the solution due to the soft-thresholding property. For this reason, the lasso has gained wide popularity in high-dimensional regression settings where p can be extremely large (sometimes on the order of millions) as the state-of-the-art solvers bypass the aforementioned computational challenges by leveraging this sparsity property. The elastic net is a convex combination of the ridge and the lasso, allowing for a more flexible model. The group lasso adds the (sum of) ℓ_2-norm penalties $\sum_{g=1}^{G} \|\beta_g\|_2$ where groups of coefficients β_g are separately penalized. This model tends to have *group-wise* sparse solutions (but typically dense within each non-zero group of coefficients).

While the ridge, lasso, and elastic net have been studied carefully both theoretically and empirically, the same cannot be said about the group lasso. We believe a large reason for this phenomenon is the inexistence of a gold-standard package like `glmnet` [Friedman et al., 2010] for solving the group lasso. In Chapter 2, we propose a novel algorithm to solve the group lasso extremely efficiently and provide an industry-grade package, `adelie`, available in both `Python` and `R`. Our main contribution lies in solving the block update in the block-coordinate descent algorithm, which achieves a quadratic rate of convergence. Moreover, we solve the group lasso problem in full generality, making no assumptions about the structure of the feature matrix X that other solvers impose. Many of the

tricks employed in `glmnet` are naturally carried over to our solver such as warm-starts, active set strategy, and extensions to general convex losses via the proximal quasi-Newton method. In Chapter 3, we propose a heuristic called the *pivot rule* for discarding features in our algorithm. We often observe extremely fast convergence with the pivot rule especially when p is large, comparable to the state-of-the-art *strong rule* and significantly faster when there exists high pairwise correlation in the features. The pivot rule is largely motivated by the strong rule and only uses the KKT conditions to derive a robust ranking system of the features to guess the next active features. We use a simple change-point detection algorithm via OLS to automatically make our selection. In Chapter 4, we introduce the (group-wise) polyhedron constrained group lasso problem, which adds linear inequality constraints for each group of coefficients in the unconstrained group lasso problem. To the best of our knowledge, there does not exist any specialized solver for this problem. Our main motivation for this project is to optimally train 2-layer ReLU neural networks [Pilanci and Ergen, 2020]. However, our solver also extends to more standard statistical applications such as isotonic regressions. Most of the moving parts of the unconstrained solver can be salvaged, but the block update requires a substantial amount of extra work. One crucial component of our constrained group lasso solver is a fast high-dimensional non-negative least squares solver (NNLS), which we study as a separate topic in Chapter 5. In fact, we study the more general polyhedron constrained least squares problem in Chapter 5 and first derive theoretical results around the existence of a sparse solution and its local uniqueness property. We use these results to derive our desired fast NNLS solver.

Chapter 2

Group lasso and elastic net

2.1 Introduction

In this chapter, we discuss one of the core results of the thesis following Yang and Hastie [2024a] that establishes fast algorithms for solving group lasso and elastic net penalized regression problems.

The group lasso [Yuan and Lin, 2006] is an extension of the celebrated lasso [Tibshirani, 1996] that aims to additionally capture group structure within the features to induce group-wise sparsity in the solutions. Specifically, the group lasso tends to zero-out *groups of coefficients* rather than individual coefficients like the lasso. The group lasso is a bona fide extension in that if all groups are of size one, then we recover the lasso. Despite the wide popularity of the lasso for its interpretability of its sparse solutions, there is a drawback when multiple columns of a feature matrix represent one entity. For example, a categorical feature, or a factor, is often represented as indicators for each category, yielding many columns. In such cases, the group lasso will select all or none of these columns whereas the lasso may select any subset of them, leaving the user in an ambiguous situation to decide whether the factor is informative or not [Meier et al., 2008]. Moreover, the result of lasso changes based on the encoding scheme of factors, which is an undesirable property [Yuan and Lin, 2006]. For these reasons, the group lasso has gained popularity over the years.

While the group lasso seems very close in spirit to the lasso (as both models induce sparsity in their solutions), their computational aspects are vastly different. A popular method for solving the lasso is coordinate-descent [Friedman et al., 2010]. The authors demonstrate that coordinate-descent can be made extremely efficient with a closed-form solution to each coordinate update. As we shall see in Section 2.3, the group lasso does not enjoy such properties as soon as a group size is greater than one. Still, inspired by the success of coordinate-descent with the lasso, many authors pursue (block) coordinate-descent as the primary workhorse of their algorithm [Yuan and Lin, 2006, Meier et al., 2008, Tseng, 2001, Liang et al., 2022]. To solve each (block) coordinate update, people have gravitated towards proximal gradient methods such as ISTA, FISTA, and FISTA with

adaptive-restarts [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]. These methods are attractive for problems like the group lasso because the block-coordinate updates can be solved iteratively where each iteration has a simple closed-form solution, now reminiscient of the lasso.

In this paper, we develop fast and scalable algorithms for solving the group lasso under the usual least squares loss (Gaussian loss) as well as any twice-continuously differentiable convex, proper, and closed losses. In particular, we can solve generalized linear models such as logistic regression, poisson regression, and multinomial regression with the group lasso penalty. Following the work of Zou and Hastie [2005], which extends the lasso to the elastic net, we work with an analogous elastic net penalty for the group lasso. Our work largely follows the work of Friedman et al. [2010] and our main result lies in the block-coordinate update. Specifically, we propose a novel algorithm to solve each block-coordinate update via Newton's method, which can be shown to have a convergence guarantee and a quadratic convergence rate. We provide a publicly available `Python` package `adelie` with brief installation directions in Section 2.8.

In Section 2.3, we present our main result for solving the group elastic net under the Gaussian loss using block-coordinate descent with a special emphasis on the block-coordinate update. Section 2.4 extends this result to general convex losses using proximal quasi-Newton method. Section 2.5 demonstrates that we may then use our results to easily fit multi-response data as well with no special changes to the solver. Finally, Section 2.6 shows our benchmark results comparing our package to existing `R` packages for solving the group lasso as well as the lasso.

2.2 Preliminaries and notations

Let $\mathbb{R}$, $\mathbb{R}^n$, and $\mathbb{R}^{m\times n}$ denote the space of real numbers, real vectors of size n, and real matrices of shape $m\times n$, respectively. Similarly, we define the analogous spaces for the non-negative real numbers $\mathbb{R}_+$ and the positive real numbers $\mathbb{R}_{++}$. We denote ℓ_p-norms on $\mathbb{R}^n$ as $\|x\|_p := (\sum_{i=1}^n |x_i|^p)^{1/p}$. We write $\|x\|_W := \sqrt{x^\top W x}$ to be the ℓ_2-norm scaled by (diagonal) weights $W \in \mathbb{R}_+^{n\times n}$. We denote $\mathbf{0}$ and $\mathbf{1}$ to be the vector of zeros and ones, respectively. The identity matrix is denoted as $I_n \in \mathbb{R}^{n\times n}$ and I if the dimensions are implicitly defined based on context. We define $[n] := \{1, \dots, n\}$ to be the set of integers ranging from 1 to n. For any vector $v \in \mathbb{R}^n$ and a set of indices $S \subseteq [n]$, let v_S be the subset of v along indices in S. If S is a subset of G groups, that is, $S \subseteq [G]$ and each group $g \in [G]$ has size p_g such that $\sum_{g=1}^G p_g = p$, then v_S is the subset of v along each group so that $v_S = \{v_g \in \mathbb{R}^{p_g} : g \in S\}$. In either context, we denote $v_{-S} \equiv v_{S^c}$ as the subset of v along indices (groups) not in S. For a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$, we denote its gradient as ∇f and the hessian as $\nabla^2 f$. For two square matrices $A, B \in \mathbb{R}^{n\times n}$, we denote $A \preceq B$ if $B - A$ is positive semi-definite and $A \prec B$ if $B - A$ is positive definite. We denote $A \otimes B$ to be the Kronecker product of any two arbitrary matrices A and B.

2.3 Algorithms for group lasso/elastic net under Gaussian loss

Consider the usual regression setup where $X \in \mathbb{R}^{n \times p}$ and $y \in \mathbb{R}^n$ are the feature matrix and response vector, respectively. Let G be the number of groups and p_g be the size of group $g \in [G]$ so that X may be decomposed as $[X_1\, X_2\, \ldots\, X_G]$ where $X_g \in \mathbb{R}^{n \times p_g}$ is the feature matrix corresponding to group g features only. In this section, we discuss our main algorithm to solve the group elastic net problem under the Gaussian loss given by

$$\underset{(\beta_0,\beta)\in\mathbb{R}^{1+p}}{\text{minimize}} \quad \frac{1}{2}\left\|y - X\beta - \beta_0\mathbf{1}\right\|_W^2 + \lambda P_{\alpha,\omega}(\beta) \tag{2.1}$$

and

$$P_{\alpha,\omega}(\beta) := \sum_{g=1}^{G} \omega_g \left(\alpha \left\|\beta_g\right\|_2 + \frac{1-\alpha}{2}\left\|\beta_g\right\|_2^2\right) \tag{2.2}$$

where $\beta_g \in \mathbb{R}^{p_g}$ are the coefficients corresponding to group g. Here, $W \in \mathbb{R}_+^{n \times n}$ is assumed to be a diagonal matrix where the diagonal specifies the observation weights. Moreover, we assume that the sum of the diagonal $\mathbf{1}^\top W \mathbf{1}$ is exactly 1 so that the weights are normalized. Typically, we use the uniform weights, which results in $W \equiv n^{-1} I$. The penalty factor $\omega \in \mathbb{R}_+^G$ allows us to control the *relative* penalty across the groups and $\lambda \geq 0$ to control the *overall* penalty. A common strategy is to set $\omega_g \equiv \sqrt{p_g}$ so that the groups are penalized relative to their size [Yuan and Lin, 2006]. The elastic net parameter $\alpha \in [0, 1]$ dictates the proportion of the group lasso penalty ($\|\cdot\|_2$) and the ridge penalty ($\|\cdot\|_2^2$) so that α closer to 1 induces more sparsity in the solutions. Note that the intercept β_0 is unpenalized. Hence, without loss of generality, we assume for simplicity that X is column-centered and y is centered (where the center is a weighted average using the diagonal of W as weights) so that we may replace $\beta_0 \equiv 0$. This simplification is specific to the Gaussian loss and we will soon remove this assumption when we move to general convex losses in Section 2.4.

Although the groups typically have less correlation *between* groups than *within* groups, as most applications tend to group correlated features together, it is nonetheless still possible to have strong correlations between groups. In this case, the group lasso can still suffer from the same erratic behavior observed with the lasso where the group lasso will tend to pick one of the correlated groups and ignore the rest. For this reason, the group elastic net penalty can help ameliorate this extreme behavior by applying some ridge penalty to shrink every group towards each other.

Following [Friedman et al., 2010], we show in Figure 2.1 a comparison of the solution paths for the group lasso, group elastic net, and ridge on the Leukemia dataset from Golub et al. [1999][1]. This dataset contains 72 samples and 7129 gene expression probes that are measured with DNA

[1] https://www.openintro.org/data/index.php?data=golub

microarrays. To introduce group structure, we take each column of gene probes $Z_{\cdot,j}$ and create two more features $Z^2_{\cdot,j}$, $Z^3_{\cdot,j}$, that is, the square and cube of the original column (element-wise). We then group every three such features together. This results in a feature matrix $X \in \mathbb{R}^{72\times 21387}$. The response y is a binary variable indicating cancer or not for each observation. However, for the sake of illustration, we treat it as a continuous variable. We run the group lasso ($\alpha = 1$), group elastic net ($\alpha = 0.2$), and ridge ($\alpha = 0$) for 100 values of λ, and plot the coefficient profiles for the first 10 values. The following code snippet shows an example of using our package `adelie` to fit group lasso on the Leukemia dataset (assuming X and y have already been loaded):

```
import adelie as ad
import numpy as np

p = X.shape[1]

state = ad.grpnet(
    X=X,
    glm=ad.glm.gaussian(y),
    groups=3 * np.arange(p),
    alpha=1.0,
    early_exit=False,
)
```

The total time to run all three models for all 100 values of λ was 0.5 seconds on a standard M1 Macbook Pro. It is clear that group lasso tends to pick sparse models with few large coefficients whereas the ridge includes all the features but shrinks them towards zero. The group elastic net is somewhere in between where we observe more non-zero coefficients than the group lasso but fewer than the ridge.

Our method to solve (2.1) is based on the block-coordinate descent (BCD) algorithm [Tseng, 2001]. Since the objective is composed of a smooth, convex, closed, and proper component and a separable non-differentiable component, BCD provably attains the minimizer. The BCD algorithm cycles through each group $g = 1, \dots, G$ and solves (2.1) only for β_g while fixing all other coefficients. In other words, we solve the following optimization problem:

$$\underset{\beta_g \in \mathbb{R}^{p_g}}{\text{minimize}} \quad \frac{1}{2}\beta_g^\top X_g^\top W X_g \beta_g - (y - \sum_{g' \neq g} X_{g'}\beta_{g'})^\top W X_g \beta_g + \lambda \omega_g \left(\alpha \left\|\beta_g\right\|_2 + \frac{1-\alpha}{2}\left\|\beta_g\right\|_2^2\right) \tag{2.3}$$

As we show in Section 2.3.1, it is advantageous to first perform an eigen-decomposition of $X_g^\top W X_g = Q_g \Lambda_g Q_g^\top$ and transform $\beta_g \mapsto Q_g^\top \beta_g$. Here, $Q_g \in \mathbb{R}^{p_g \times p_g}$ is a square orthogonal matrix and $\Lambda_g \in \mathbb{R}_+^{p_g \times p_g}$ is a non-negative diagonal matrix, which may contain some zeros. Using the fact that the

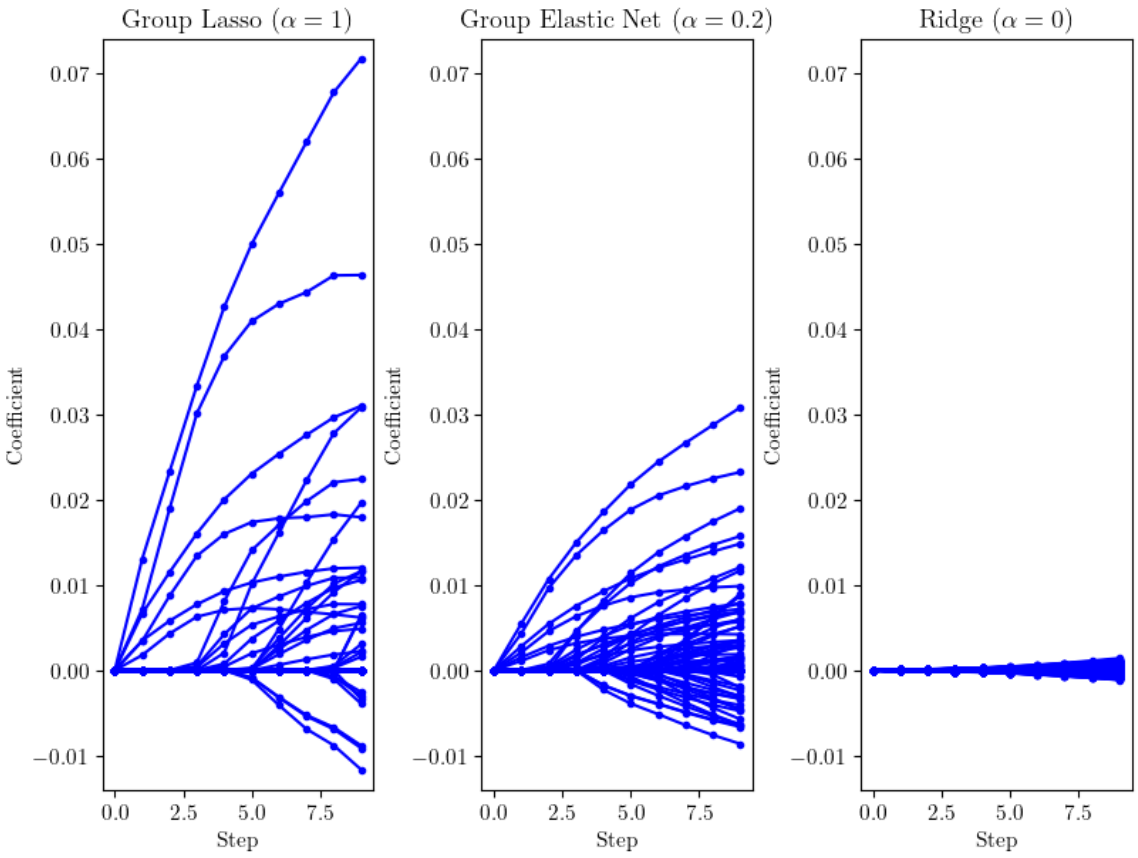


Figure 2.1: A comparison of the coefficient profile for the group lasso, group elastic net, and the ridge on the Leukemia dataset by [Golub et al., 1999]. From left to right, we gradually see more features entering the model with an overall shrinkage on the coefficients towards zero until the ridge includes every feature.

penalty is invariant under rotation and letting $\tilde{\beta}_g := Q_g^\top \beta_g$, (2.3) changes to

$$\underset{\tilde{\beta}_g \in \mathbb{R}^{p_g}}{\text{minimize}} \quad \frac{1}{2}\tilde{\beta}_g^\top \Lambda_g \tilde{\beta}_g - (y - \sum_{g' \neq g} X_{g'}\beta_{g'})^\top W X_g Q_g \tilde{\beta}_g + \lambda\omega_g \left(\alpha \left\| \tilde{\beta}_g \right\|_2 + \frac{1-\alpha}{2} \left\| \tilde{\beta}_g \right\|_2^2 \right) \tag{2.4}$$

We abstract this problem into the following general problem

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \quad \frac{1}{2}x^\top \Sigma x - v^\top x + \lambda \left\| x \right\|_2 \tag{2.5}$$

where $\Sigma \in \mathbb{R}_+^{d \times d}$ is a non-negative diagonal matrix and $\lambda \geq 0$. Relating back to (2.4), we have $\Lambda_g + \lambda\omega_g(1-\alpha)I \mapsto \Sigma$, $Q_g^\top X_g^\top W(y - \sum_{g' \neq g} X_{g'}\beta_{g'}) \mapsto v$, and $\lambda\omega_g\alpha \mapsto \lambda$ (with a slight abuse of notation). The rest of the section focuses on solving (2.5). Since $\lambda > 0$ is the only interesting case, we assume without loss of generality that $\lambda > 0$.

We make a few remarks on (2.5). Suppose for the moment that the observation weights $W = n^{-1}I$ are equal weights to be consistent with the other references. In the literature, we have found that most, if not all, methods only consider isotropic Σ, that is, $\Sigma = tI$ for some scalar $t > 0$. Yuan and Lin [2006] directly assume that X_g in (2.3) has orthonormal columns so that $\Sigma \equiv I$. In the general case, they at least assume X_g to be full-rank so that it admits a QR decomposition with an invertible R. Then, they solve (2.3) with X_g replaced with the Q matrix and naively rescale the result by R^{-1}. This strategy has drawbacks because it assumes full-rankness and, more fundamentally, the resulting coefficient is *not* the solution to the original problem (2.3). This was also observed and noted in Simon and Tibshirani [2012] and the authors proposed to fix the issue by amending the group lasso penalty from $\left\|\beta_g\right\|_2$ to $\left\|X_g\beta_g\right\|_2$. While this allows the derivation to work out nicely, we otherwise view this change merely as a mathematical convenience and not a satisfactory solution to the original problem (2.3). Other authors such as Liang et al. [2022], Beck and Teboulle [2009], Meier et al. [2008] majorize $X_g^\top X_g$ using an isotropic matrix. They are able to show convergence and a convergence rate of $O(1/k^2)$ where k is the number of iterations. However, we noticed in practice that their methods converge very slowly either when $X_g^\top X_g$ is near singular or the dimension of the block increases. As the block-coordinate update is the most critical component of the BCD algorithm, we desire a more reliable and a faster algorithm. Our method directly solves (2.5) with no workarounds or approximations and works in full generality for any non-negative diagonal Σ, or equivalently, any *arbitary* X_g. Moreover, we attain a quadratic convergence rate and a stable algorithm in practice.

It was previously noted that we must perform an eigen-decomposition for $X_g^\top W X_g$. This decomposition only needs to be done *at most once per group*. Moreover, we will see later in Section 2.3.6 that the decomposition only needs to be done on a small subset of the groups.

We conclude this section with the statement of Theorem 2.3.1 (see Appendix A.1 for the proof). We state this theorem for two reasons. First, the problem in (2.5) is indeed more general than (2.4).

If Σ is not positive definite, then for a specific choice of v, (2.5) may not have a solution due to an unbounded objective. Since we will now restrict our attention to (2.5), we provide a sufficient condition for the existence of a solution for completion. It is easy to check via the SVD of X_g that the conditions of Theorem 2.3.1 hold for our original problem (2.3). Secondly, condition (2.6) will become crucial in Section 2.3.2 where we develop our adaptive bisection method.

Theorem 2.3.1 (Sufficient Condition for the Existence of the Minimizer of (2.5))**.** *Consider the optimization problem of* (2.5) *with* $\lambda > 0$*. If the following condition holds*

$$v_S = \mathbf{0}, \quad S := \{i \in [d] : \Sigma_{ii} = 0\} \tag{2.6}$$

then a finite minimizer exists. Moreover, the minimizer is given by

$$x^\star = \begin{cases} \left(\Sigma + \frac{\lambda}{\|x^\star\|_2} I\right)^{-1} v, & \|v\|_2 > \lambda \\ 0, & \|v\|_2 \leq \lambda \end{cases} \tag{2.7}$$

2.3.1 Vanilla Newton's method-based algorithm

We now discuss how to solve (2.8) via Newton's method. Assume (2.6) holds so that a solution exists by Theorem 2.3.1. Without loss of generality, we only consider the first case of (2.7) when $\|v\|_2 > \lambda$. Without knowing $\|x^\star\|_2$, there is no closed-form solution for $x^\star$. In an endeavor to find an expression for $\|x^\star\|_2$, we take the ℓ_2-norm on both sides to get that $\|x^\star\|_2$ must satisfy

$$\sum_{i=1}^{d} \frac{v_i^2}{(\Sigma_{ii} \|x^\star\|_2 + \lambda)^2} = 1 \tag{2.8}$$

Unfortunately, there is no closed-form solution for $\|x^\star\|_2$ from (2.8) either. We see this as the main difference from performing coordinate-descent for lasso. Namely, the coordinate update in lasso has a simple closed-form solution (soft-thresholding function) [Friedman et al., 2010]. However, as soon as a group has size greater than 1, there is no closed-form solution to its block-coordinate update in the group lasso. In spite of this difficulty, we shall see shortly that a simple Newton's method can numerically solve (2.8) efficiently with guaranteed convergence. In turn, once we find $\|x^\star\|_2$, we may substitute it into (2.7) to get the full solution $x^\star$ in $O(d)$ time. Note that we can achieve this complexity because Σ is diagonal. Indeed, the subsequent arguments will also heavily rely on this fact to achieve $O(dk)$ time complexity, where k is the number of Newton iterations, which would otherwise be $\Omega(d^2 k)$ if Σ were not diagonal. This computation reduction was the primary reason for transforming our problem from (2.3) to (2.4). We also note in passing that Hastie et al. [2016, Ex. 4.7] also discovered the relationship between (2.7) and (2.8) from which they proposed to solve for $\|x^\star\|_2$ via the golden-section search.

Algorithm 1: 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;

Now, define $\varphi : [0, \infty) \to \mathbb{R}$ by

$$\varphi(h) := \sum_{i=1}^{d} \frac{v_i^2}{(\Sigma_{ii} h + \lambda)^2} - 1 \tag{2.9}$$

so that (2.8) is now a one-dimensional root-finding problem for φ. In Appendix A.3, we show that φ is strictly decreasing, strictly convex, and $\varphi(0) > 0 > \lim_{h\to\infty} \varphi(h)$, so that a root (uniquely) exists. Our strategy is then to apply Newton's method to search for the root of φ. Newton's method starts with an initial point $h^{(0)}$ and iteratively updates

$$h^{(k+1)} = h^{(k)} - \varphi'(h^{(k)})^{-1} \varphi(h^{(k)}) \tag{2.10}$$

Algorithm 1 summarizes Newton's method. By Theorem 2.3.2, Newton's method has guaranteed convergence to the root of φ for any initial point $h^{(0)} \geq 0$ such that $\varphi(h^{(0)}) \geq 0$ (e.g. $h^{(0)} \equiv 0$). Moreover, the Newton iterates are increasing towards the root. For a proof of Theorem 2.3.2, see Appendix A.2. Since φ' is bounded away from 0 and φ'' is continuous on the interval $[0, r]$ where r is the root of φ, standard theory shows that we also have a quadratic convergence rate [Boyd and Vandenberghe, 2004].

Theorem 2.3.2 (Convergence Result of Newton's Method)**.** *Suppose $f : \mathbb{R} \mapsto \mathbb{R}$ is convex, continuously differentiable with a non-vanishing derivative, and decreasing. Suppose further that there exists a root $r \in \mathbb{R}$ such that $f(r) = 0$. Let $x^{(0)}$ be any initial point such that $f(x^{(0)}) \geq 0$. Then, the Newton iterates $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ defined by*

$$x^{(k+1)} = x^{(k)} - f'(x^{(k)})^{-1} f(x^{(k)}) \tag{2.11}$$

is an increasing sequence that converges to a root of f.

It is clear from the discussion so far why our method, in principle, is faster and more stable than the aforementioned proximal gradient methods such as ISTA, FISTA, and FISTA with adaptive restarts. First, none of these methods are able to give any convergence rate comparable to our quadratic convergence rate largely because they are first-order methods. Secondly, while these

proximal gradient methods search the entire $\mathbb{R}^d$ space, we have reduced our problem to a *one-dimensional* root finding problem. Lastly, we have established a well-behaving sequence of iterates (e.g. increasing and bounded).

Figure 2.2 shows a comparison of the proximal gradient methods and our Newton's method, clearly demonstrating the superiority of our method. We implemented all these methods as optimized as possible in `C++`. For FISTA and FISTA with adaptive restarts, we follow the details laid out by the authors of Beck and Teboulle [2009], O'Donoghue and Candés [2015]. We generate the diagonal of $\Sigma \in \mathbb{R}^{d\times d}_{++}$ from the uniform distribution on the interval $(0, 1)$, the linear term $v = \Sigma^{1/2}v'$ where $v' \sim \mathcal{N}(0, I_d)$, and $\lambda = 0.1$. We measure accuracy as

$$\max_{i=1,\dots,d} |\hat{x}_i - \tilde{x}_i| \tag{2.12}$$

where $\hat{x}$ is the solution returned by a solver and $\tilde{x}$ is defined by (2.7) where the $\|x\|_2$ is replaced with $\|\hat{x}\|_2$. Note that the proximal methods fail to converge within 10000 iterations as d increases, and even when they converge, Newton's method performs 10 to 1000 times faster. Furthermore, the accuracy of the Newton's method remains steady regardless of the dimension d, while the proximal methods have trouble searching the space efficiently, suffering from the lack of guidance from the curvature information.

Despite the success of our Newton's method approach, there is still room for improvement. It is widely known that descent methods (including Newton's method) is highly sensitive to the initialization. A poor initialization may lead to significantly slower convergence in practice while an initialization close to the root will be rewarded with incredible speed. It then remains to show how to properly choose the initial starting point $h^{(0)}$. First, it is always advantageous to select $h^{(0)}$ such that $\varphi(h^{(0)}) \geq 0$ to preserve the increasing property of the iterates and the convergence guarantees. We have chosen $h^{(0)} \equiv 0$ thus far, but it may not be the best initial value as we may observe relatively larger number of iterations, especially if Σ is singular and λ is small. This is due to the fact that we may require many iterations before achieving the quadratic convergence rate. The left plot in Figure 2.3 shows an example of such a case. We see that φ decays rapidly near $h \approx 0$ and has an extremely flat tail. The iterates slowly move from the origin until $h \approx 100$ and the speed picks up from there. One possible remedy is to simply start Newton's method from a large value of h rather than 0. However, as Figure 2.3 suggests, because the function usually has a very flat tail, Newton's method will likely shoot the iterate to the left with a large magnitude. As a result, we may end up with an iterate that is negative. Upon projecting back to $[0, \infty)$, we return to the situation where we start Newton's method at the origin. For these reasons, we develop in Section 2.3.2 a special modification of our Newton-based approach, which describes our most performant and robust method that reduces iterations in nearly all configurations of the inputs.

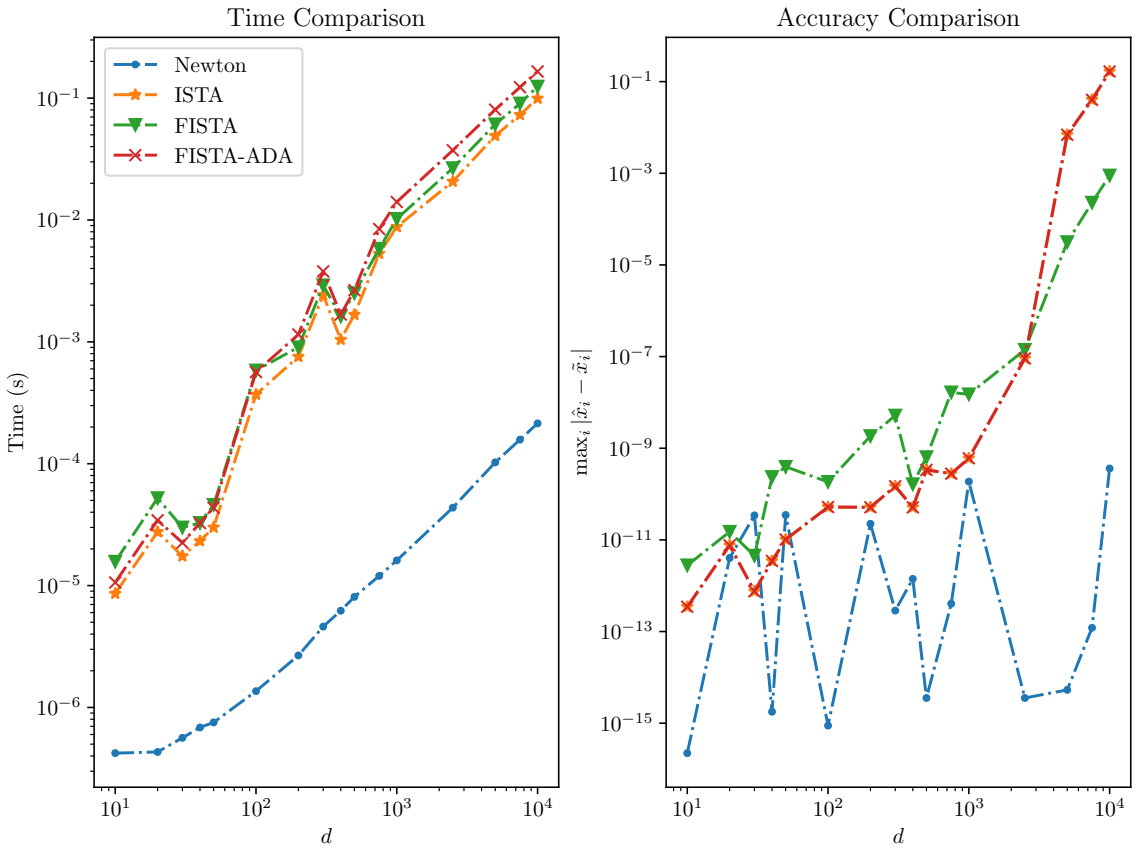


Figure 2.2: Time and accuracy comparison between the proximal gradient methods and our Newton's method. Overall, the proximal gradient methods suffer from large computational overhead due to large number of iterations, and fails to converge in time as the dimension increases. Contrastingly, Newton's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

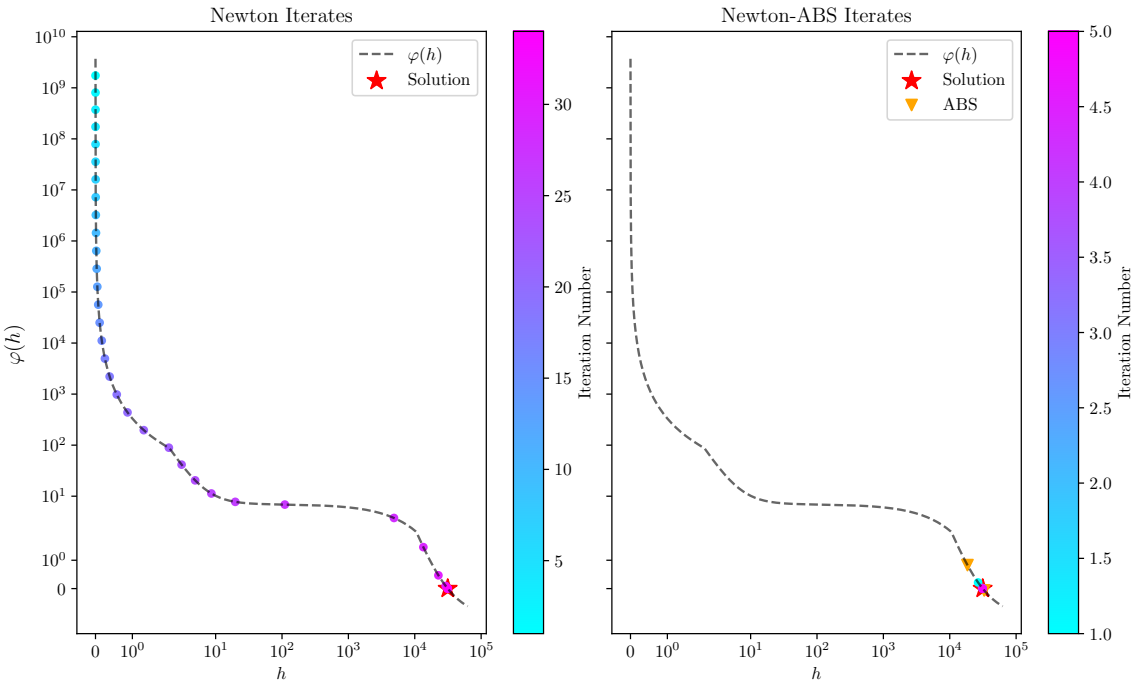


Figure 2.3: Plot of the iterates for the vanilla Newton's method (left) and Newton-ABS (right). In both plots, we display the iterates by their value of φ and describe their path by the color gradient scheme. The iterates are guaranteed to move from left to right. The solution is marked with a star. For visualization purposes, the axes were scaled using the symmetrical log transformation, which deceptively makes φ look non-convex. It is clear that Newton's method struggles where there is a sharp decay near the origin as the Newton iterates slowly exit the kink. Newton-ABS gets around this problem by finding a good initial point sufficiently away from the origin using our adaptive bisection strategy.

2.3.2 Newton's method with Adaptive Bisection Starts algorithm

In this section, we improve our vanilla Newton's method-based algorithm described in Section 2.3.1. The key idea is to first apply a bisection method to search for a good initial point. In particular, based on Figure 2.3, it is important to avoid the region of fast decay when h is close to 0, as this is the region with many Newton iterations. Once we find a point sufficiently close to the root, we apply Newton's method as a refinement procedure, recovering all the convergence guarantees. Although one could use any bisection method in theory, we propose an *adaptive bisection method* that has been most effective in our experience. We note in passing that we have experimented with other bisection methods such as the naive one that bisects evenly and other sophisticated methods such as Dekker's method and Brent's method [Brent, 2013, Dekker, 1969]. Brent's method, in particular, was the only competitive method to our approach, however the runtime was quite unpredictable as it was too dependent on the input values.

We first discuss how to find lower and upper bounds $h_\star,\ h^\star \in [0, \infty)$, respectively, such that the root lies in $[h_\star, h^\star]$. To motivate the derivation, note that the root-finding problem for φ is equivalent to finding the largest value $h > 0$ such that $\varphi(h) \geq 0$, or

$$\sup \{h > 0 : \varphi(h) \geq 0\} \tag{2.13}$$

The idea is to find a different function $\varphi_\star(\cdot)$ such that

$$h_\star := \sup \{h > 0 : \varphi_\star(h) \geq 0\} \implies \varphi(h_\star) \geq 0$$

and $h_\star$ is easily computable. Similarly, we find a function $\varphi^\star(\cdot)$ such that

$$h^\star := \inf \{h > 0 : \varphi^\star(h) \leq 0\} \implies \varphi(h^\star) \leq 0$$

This shows that the root must lie in $[h_\star, h^\star]$. In Appendix A.4, we find suitable functions $\varphi_\star$ and $\varphi^\star$, and derive the following expressions for $h_\star$ and $h^\star$:

$$h_\star = \left(\frac{-\lambda \mathbf{1}^\top \Sigma \mathbf{1} + \sqrt{(\mathbf{1}^\top \Sigma \mathbf{1})^2 - (\mathbf{1}^\top \Sigma^2 \mathbf{1})(\lambda^2 d - \|v\|_1^2)}}{\mathbf{1}^\top \Sigma \mathbf{1}} \right)_+ \tag{2.14}$$

$$h^\star = \sqrt{\sum_{i:\Sigma_{ii}>0} v_i^2 \Sigma_{ii}^{-2}} \tag{2.15}$$

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 closer to $h^\star$. Since we do not know the root a priori, it is not possible

Algorithm 2: Adaptive Bisection

Data: Σ, v, λ, ε

Compute $h_\star$ using (2.14) and $h^\star$ using (2.15);

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

$h \leftarrow h^\star$;

while $\varphi(h) < -\varepsilon$ **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;

to determine which side it is closer to. However, we can make an informed guess based on a *prior* of how likely the root is closer to $h_\star$. With this in mind, we construct our prior based on the derivation of $h^\star$ in Appendix A.4. For completeness, we provide the relevant part of the proof:

$$\sum_{i=1}^{d} \frac{v_i^2}{(\Sigma_{ii} h + \lambda)^2} = \sum_{i:\Sigma_{ii}>0} \frac{v_i^2}{(\Sigma_{ii} h + \lambda)^2} \leq h^{-2} \sum_{i:\Sigma_{ii}>0} \frac{v_i^2}{\Sigma_{ii}^2} \tag{2.16}$$

It is easy to see that our $h^\star$ makes the right-side of (2.16) equal to 1. We note that the inequality above becomes tighter as $h^\star$ approaches the root. This motivates us to consider the worst approximation error (rate)

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

If $w \approx 0$, the inequality in (2.16) is tight, which suggests that the root is close to $h^\star$. Hence, 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 $\varphi(h) > \epsilon$ (sufficiently positive), we use $h^{(0)} := h$ as the initial point for our Newton's method. Otherwise if $\varphi(h) < -\epsilon$ (sufficiently negative), we set $h^\star := h$ and repeat the bisection argument. Finally, if $|\varphi(h)| \leq \epsilon$ (sufficiently close to 0), we declare h to be the root. Algorithms 2 and 3 summarize this procedure, which we call *Newton's Method with Adaptive Bisection Starts* (Newton-ABS). We emphasize that we still have a convergence guarantee since the bisection method always shrinks the interval containing the root and we eventually apply Newton's method.

Algorithm 3 can be further optimized. 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. One may also enforce enough movement towards $h_\star$ by 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 suggests too strongly that the root is close to $h^\star$. The idea is that it is better to overshoot towards $h_\star$ such that φ is non-negative so that we can

Algorithm 3: Newton's Method with Adaptive Bisection Starts (Newton-ABS)

Data: Σ, v, λ, ε
$h \leftarrow$ result of Algorithm 2;
if $|\varphi(h)| > \varepsilon$ **then**
  $h \leftarrow$ result of Algorithm 1 starting at $h^{(0)} = h$;
$x^\star \leftarrow (\Sigma + \lambda h^{-1} I)^{-1} v$;
return $x^\star$;

apply Newton's method sooner.

Figure 2.3 shows a plot of the Newton iterations for the vanilla Newton's method from Section 2.3.1 and our proposed Newton-ABS method. It is clear that Newton's method struggles where there is a sharp decay near the origin as the Newton iterates slowly exit the kink. However, Newton-ABS gets around this problem due to the fact that the adaptive bisection first moves in the opposite direction, starting at the upper bound $h^\star$ and moving towards $h_\star$ until φ is non-negative. In this example, only 2 adaptive bisections were required and 5 subsequent Newton iterations, which is a noticeable improvement from the 34 vanilla Newton iterations. Empirically, we have never observed a case where the initial value remains at the kink after performing the adaptive bisection.

2.3.3 Naive and covariance updates

As in the coordinate descent algorithm for lasso, we have a "naive" and "covariance" method to update the residual or gradient vector, respectively [Friedman et al., 2010]. Denote β as the current coefficient vector and suppose in the BCD algorithm, we are at group g.

The "naive" method keeps track of the residual $r := y - X\beta$. To solve the BCD update (2.3), we must prepare the inputs to (2.4). Recall that we can cache the eigen-decomposition of $X_g^\top W X_g$ so that we have access to its eigenvalues Λ_g and eigenvectors Q_g. Then, we may compute the linear term in the following way

$$Q_g^\top X_g^\top W(y - \sum_{g' \neq g} X_{g'} \beta_{g'}) = Q_g^\top X_g^\top W r + \Lambda_g Q_g^\top \beta_g$$

In other words, we simply need to compute $\gamma_g := X_g^\top W r$, express γ_g and β_g in the eigen-basis Q_g, and perform simple elementwise operations. Altogether, the cost is $O((n + p_g)p_g)$. Once the BCD update, $\tilde{\beta}_g$, is found by solving (2.3), we must update the residual to reflect the change in β_g. Given the current residual r, the change in residual, Δr, can then be expressed as

$$\Delta r = -X_g \left(\tilde{\beta}_g - \beta_g \right)$$

So, the residual update requires another $O(np_g)$ operations. Sometimes, we can avoid performing the

residual update entirely if we observe that $\tilde{\beta}_g \equiv \beta_g$, i.e. the BCD update did nothing. Nevertheless, the "naive" method has a worst-case cost of $O((n + p_g + k)p_g)$ per BCD update where k is the number of iterations for the Newton-ABS method.

The "covariance" method keeps track of the gradient $\gamma := X^\top W(y - X\beta)$. This simplifies the input preparation for the BCD update (2.3) since the linear term does not require the $O(np_g)$ operations to compute $X_g^\top Wr$ as in the "naive" method so that the complexity is $O(p_g^2)$. However, it could potentially make the gradient update more costly. The change in the gradient $\Delta\gamma$ can be expressed as

$$\Delta\gamma = -X^\top W X_g(\tilde{\beta}_g - \beta_g)$$

where we only have access to $X^\top W X_g$ as a whole. Hence, the gradient update requires $O(p \cdot p_g)$ operations. Therefore, the "covariance" method has a worst-case cost of $O((p + p_g + k)p_g)$, where k is the number of iterations for the Newton-ABS method.

Depending on the use-case, the user may find that one method is faster than the other. Specifically, when $n \ll p$, the "naive" method is more efficient, and when $p \ll n$, the "covariance" method is more efficient. However, in terms of memory, the "naive" method is almost always preferred (unless $n \gg p \cdot n_{\mathcal{A}}$ where $n_{\mathcal{A}}$ is the number of active coefficients). This is because the "covariance" method requires saving $X^\top W X_g \in \mathbb{R}^{p \times p_g}$ at the very least for the active groups g (with non-zero coefficients). With large p, this may be unmanageable. While we could technically recompute $X^\top W X_g$ on-the-fly if we have insufficient memory, this would significantly hurt the performance and will fail to be competitive against the "naive" method. On the other hand, the "naive" method only requires $O(n)$ extra memory for the residual.

2.3.4 Matrix abstraction

In this section, we restrict ourselves to the "naive" method for brevity, however, we note that similar arguments can be made for the "covariance" method. The reader may notice that the BCD algorithm only ever interacts with the matrix X through a few operations. Most notably, in Section 2.3.3, we demonstrate that X is only used in the BCD update to compute the gradient γ_g and update the residual r. Moreover, it is easily seen that the bulk of the computation in each BCD update lies in these two operations. Hence, any effort into speeding up these operations will yield large performance gains for the solver. Our software `adelie` takes advantage of this fact and abstracts out the matrix class. That is, any matrix-like class supporting methods to compute $X_{\cdot,j:k}^\top v$ (gradient) and $r \mathrel{-}= X_{\cdot,j:k} w$ (residual update) can be used by our solver in the block update. This gives the user the opportunity to exploit any structure in X when implementing these methods to potentially simplify the computation.

This abstraction has been proven to be quite useful for many applications. Simple examples

include dense and sparse matrices. A more motivating example is in genome-wide association studies (GWAS) where the main feature matrix of interest is the genotype matrix of SNP information taking on values in $\{0, 1, 2, \text{NA}\}$. There is a plethora of references in this area, so we only mention a few [Uffelmann et al., 2021, Witte, 2010, Mills and Rahal, 2019, He et al., 2022]. These genotype matrices can be of daunting size. The UK biobank dataset [Sudlow et al., 2015], for example, contains 500,000 samples and about 1.7 million SNPs (features). However, both the sparsity of this matrix as well as the restricted range of values allow us to efficiently store this matrix with far more structure than a dense or a sparse matrix. Indeed, there has been some work to solve lasso on such large-scale data [Qian et al., 2020, Li et al., 2021, He et al., 2024, Zeng and Breheny, 2018], all implementing their own solvers by exploiting the structure of the genotype matrix. Another useful matrix is a column-wise concatenation of a list of matrices. In the GWAS setting, we also have a small set of dense covariates such as age, sex, and 10 principal components of the genotype matrix. These covariates are often included in the final feature matrix along with the genotype matrix.

We conclude this section with a brief discussion of the implementation details of the gradient computation and the residual update for the aforementioned matrix classes. For an in-depth discussion of the full API of the abstract matrix class, we refer the readers to our documentation linked in Section 2.8. For examples on using various types of matrices, we refer the readers to our user guide[2]. First, dense and sparse matrices can be wrapped to expose the standard implementations of the two operations (e.g. BLAS). Next, for the genotype matrix discussed above, the gradient computation can be implemented in the following way:

$$X_{\cdot,\ell}^{\top} v = \sum_{i:X_{i\ell}=1} v_i + 2 \sum_{i:X_{i\ell}=2} v_i + \mu_\ell \sum_{i:X_{i\ell}=\text{NA}} v_i, \quad \forall \ell = j, \ldots, k$$

where μ_ℓ is any imputed value (typically the imputed mean) for column ℓ. That is, we may skip over rows when $X_{i\ell} = 0$ and otherwise compute a partial sum of v for all other categories; afterwards, the partial sums are multiplied by their respective category values and summed. Similarly, the residual update can be done in the following way: for each $\ell = j, \ldots, k$ and category $c \in \{1, 2, \text{NA}\}$,

$$r_i \mathrel{-}= \rho_\ell(c) w_\ell, \quad \text{if} \quad X_{i\ell} = c$$

where $\rho_\ell(1) = 1$, $\rho_\ell(2) = 2$, $\rho_\ell(\text{NA}) = \mu_\ell$.

2.3.5 Pathwise block-coordinate descent

We compute solutions along a decreasing sequence of the regularization parameter λ. We start with the smallest λ, say $\lambda_{\max}$, for which all groups with non-zero group lasso penalty have coefficients identically $\mathbf{0}$. This value can be directly computed once we fit all unpenalized variables. That is, if

[2] https://jamesyang007.github.io/adelie/notebooks/examples.html

$U := \{g \in [G] : \omega_g \alpha = 0\}$ denotes the set of groups that have vanishing group lasso penalty, we first solve (2.1) only for groups in U to get their corresponding coefficients $\hat{\beta}_U$. Then, we compute $\lambda_{\max}$ given by

$$\lambda_{\max} := \max_{g: g \notin U} \frac{\left\| X_g^\top W (y - X_U \beta_U) \right\|_2}{\alpha \omega_g} \tag{2.17}$$

By the KKT conditions, we have our desired property that the zero solution on groups not in U is optimal at $\lambda_{\max}$. Note that if U is empty (which occurs if $\alpha = 0$ or every $\omega_g = 0$) so that every group is included in the model, then we may arbitrarily take any value for $\lambda_{\max}$. From $\lambda_{\max}$, we generate a decreasing sequence of K number of λ values down to $\epsilon \lambda_{\max}$ evenly-spaced on the log scale, where $\epsilon \in [0, 1]$ is a user-specified value that determines the depth of the path. Following Friedman et al. [2010], we typically set $\epsilon \in [0.0001, 0.01]$ and $K = 100$.

The advantage of a pathwise solver is that the solver is warm-started at each λ (including $\lambda_{\max}$). This helps tremendously for smaller values of λ where a cold-start generally makes the solver wander longer before reaching the optimum. This was also observed by Friedman et al. [2010]. Similar ideas exist in the convex literature, in particular, with interior-point methods and barrier methods [Boyd and Vandenberghe, 2004].

2.3.6 Screen and active sets

In the BCD algorithm, we can layer two types of optimizations that both reduce computation by cycling over only a subset of the groups. The first optimization is based on a *screening rule*, which tries to guess the set of groups that will likely contain the true active set (groups with non-zero coefficients). We call such a set the *screen set* and denote it as $\mathcal{S}$. It is possible to modify the BCD algorithm to only ever iterate over the screen set. Therefore, as mentioned in Section 2.3, we may only cache the eigen-decomposition of $X_g^\top W X_g$ for $g \in \mathcal{S}$. However, this optimization comes at a (small) price of checking the KKT conditions. The second optimization further reduces most of the computation to a subset of active groups in the screen set.

In the first optimization, the screen rule can be a heuristic or a guarantee. The literature contains many types of such rules including SAFE rule, strong rule, and EDPP rule [Ghaoui et al., 2011, Tibshirani et al., 2012, Wang et al., 2015b]. In particular, the SAFE rule and the EDPP rule are examples of guarantees in the sense that at a fixed λ if these rules discard groups, then they guarantee that these groups will truly have zero coefficients. On the other hand, the strong rule is an example of a heuristic, which does not always guarantee that the groups that it discards will be inactive. Despite the lack of a guarantee, we find that among these rules the strong rule was the most effective one in practice as it includes much fewer groups and is almost always correct. For this reason, we implement the strong rule as part of our software.

We briefly describe the strong rule for the group elastic net problem. Suppose we have a solution

$\tilde{\beta}$ at $\tilde{\lambda}$ and the current screen set is $\tilde{\mathcal{S}}$. We must now solve for $\lambda < \tilde{\lambda}$ to obtain the next solution β. For the group elastic net, the strong rule searches for all remaining groups $g \notin \tilde{\mathcal{S}}$ such that

$$\left\| X_g^\top W(y - X\tilde{\beta}) \right\|_2 < 2\lambda - \tilde{\lambda} \tag{2.18}$$

If this condition is satisfied, group g is discarded from the screen set. Note that $2\lambda - \tilde{\lambda} = \lambda - (\tilde{\lambda} - \lambda)$ so that (2.18) looks at the target λ and a band below equal to the path gap at λ. Once all such groups are discarded to obtain the next screen set $\mathcal{S}$, we proceed to solve the group elastic net only over the groups in $\mathcal{S}$, effectively solving the (constrained) group elastic net problem with the constraint that $\beta_{-\mathcal{S}} = \mathbf{0}$; at the start of the path, the screen set only contains unpenalized groups so that the screen set is exactly the active set at $\lambda_{\max}$. It is easy to imagine that the more accurate the screening rule is, the smaller the screen set $\mathcal{S}$, and therefore, the less computation needed in the BCD algorithm. Since we guess that groups outside the screen set are inactive, we must verify the KKT conditions to confirm our guess. This amounts to checking that

$$\left\| X_g^\top W(y - X\beta) \right\|_2 \leq \lambda, \quad \forall g \notin \mathcal{S}$$

If the condition holds, then we may declare β as the solution at λ.

The second optimization reduces the computation further by restricting most of the BCD iterations to the current active set within the screen set. Namely, we first iterate over the screen set once to determine the active groups. Afterwards, if we have not converged, we iterate only over the active groups until convergence. We then repeat the entire process until we reach convergence on the screen set. This active-set strategy is also mentioned in Friedman et al. [2010], Meier et al. [2008], Krishnapuram et al. [2005].

2.3.7 Convergence criterion

In the BCD algorithm, we measure convergence by measuring the average amount in which each block-update results in a change in the linear prediction and take the largest such value. Concretely, suppose $\tilde{\beta}$ is the current coefficient vector and we have just updated group g so that the new coefficient β only differs from $\tilde{\beta}$ in the entries defined by group g. Then, we compute $p_g^{-1}\|X_g(\beta_g - \tilde{\beta}_g)\|_W^2$. Note that since we have already cached $X_g^\top W X_g = Q_g \Lambda_g Q_g^\top$, it is equivalent to computing $p_g^{-1}\|Q_g^\top(\beta_g - \tilde{\beta}_g)\|_{\Lambda_g}^2$. Since we must apply the rotation $Q_g^\top$ on $\tilde{\beta}_g$ and β_g anyways to solve (2.4), computing the convergence measure is an easy task. After one full cycle of the groups, we check if

$$\max_g \quad p_g^{-1} \left\| X_g(\beta_g - \tilde{\beta}_g) \right\|_W^2 \leq \epsilon \tag{2.19}$$

where $\epsilon \geq 0$ is a user-specified tolerance level. If the condition holds, we declare convergence.

2.3.8 Lasso optimization

As mentioned before, groups of size 1 have the same coordinate update as in the lasso, which admits a closed-form solution. Hence, at every coordinate update, our software checks for the group sizes and falls back to the lasso update in this special case. Otherwise, we use our Newton-ABS method to solve the update.

2.4 Algorithms for regularized generalized linear models

In Section 2.3, we discussed our main algorithm for solving the group elastic net under the Gaussian loss. In this section, we extend our algorithm for general convex losses.

Suppose $\ell(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is a convex, proper, and closed function that is twice-continuously differentiable. We typically consider $\ell(\cdot)$ as the negative log-likelihood of some generalized linear model (GLM) so that it is of the form

$$\ell(\eta) := \sum_{i=1}^{n} w_i \left(-y_i \eta_i + A(\eta_i)\right)$$

where $A(\cdot)$ is the log-partition function of an exponential family. However, for the current discussion, we need not restrict ourselves to such form. Our goal is to solve the more general optimization problem

$$\begin{aligned} \underset{(\beta_0,\beta)\in\mathbb{R}^{1+p}}{\text{minimize}} \quad & \ell(\eta) + \lambda P_{\alpha,\omega}(\beta) \\ \text{subject to} \quad & \eta = X\beta + \beta_0 \mathbf{1} + \eta^0 \end{aligned} \tag{2.20}$$

where the group elastic net penalty $P_{\alpha,\omega}(\cdot)$ is defined by (2.2). The only new term is $\eta^0 \in \mathbb{R}^n$, which we call the *offset* vector. The offset is a fixed vector that defines possibly different baseline linear predictions for each observation so that $X\beta + \beta_0 \mathbf{1}$ is interpreted as the change in the linear prediction from the given baseline. For the Gaussian loss

$$\ell(\eta) := \sum_{i=1}^{n} w_i \left(-y_i \eta_i + \frac{\eta_i^2}{2}\right) \equiv \frac{1}{2}\left(\|y-\eta\|_W^2 - \|y\|_W^2\right)$$

the offset can be managed in a trivial manner by replacing y with $y - \eta^0$ and running the algorithm described in Section 2.3 with no other changes. However, with general loss functions, the offset can interact with the objective in a non-trivial manner.

We use proximal quasi-Newton method to solve (2.20). This method is also mentioned in Friedman et al. [2010], Meier et al. [2008]. Moreover, Lee et al. [2014] proves a superlinear convergence rate for proximal quasi-Newton methods (with line search). However, as in Friedman et al. [2010],

Algorithm 4: Proximal Quasi-Newton Method for General Smooth, Convex Losses

Data: X, y, $\beta_0^{(0)}$, $\beta^{(0)}$, η^0
$k \leftarrow 0$;
while *not converged based on* (2.23) **do**
  $k \leftarrow k + 1$;
  $\eta^{(k-1)} = X\beta^{(k-1)} + \beta_0^{(k-1)}\mathbf{1} + \eta^0$;
  Compute $\nabla\ell(\eta^{(k-1)})$;
  Compute $W^{(k-1)}$ appropriate for the given loss $\ell(\cdot)$;
  Compute $(\beta_0^{(k)}, \beta^{(k)})$ by solving (2.21) using our method in Section 2.3;
return $(\beta_0^{(k)}, \beta^{(k)})$;

we do not perform the line search since it will significantly impact the overall runtime and it does not seem necessary in practice thanks to the warm-starts in our pathwise solver. We observe fast convergence, usually only requiring 1 to 5 iterations.

We are now in position to describe the proximal quasi-Newton method for our problem. We begin with an initial value $(\beta_0^{(0)}, \beta^{(0)}) \in \mathbb{R}^{1+p}$. For each iteration $k \geq 0$ until convergence, we compute our current linear prediction $\eta^{(k)} = X\beta^{(k)} + \beta_0^{(k)}\mathbf{1} + \eta^0$. We then perform a second-order expansion of $\ell(\cdot)$ around $\eta^{(k)}$ and solve for the proximal operator to get the next iterate $(\beta_0^{(k+1)}, \beta^{(k+1)})$. That is,

$$(\beta_0^{(k+1)}, \beta^{(k+1)}) = \underset{(\beta_0,\beta)\in\mathbb{R}^{1+p}}{\operatorname{argmin}} \quad \nabla\ell(\eta^{(k)})^\top\eta + \frac{1}{2}(\eta - \eta^{(k)})^\top W^{(k)}(\eta - \eta^{(k)}) + \lambda P_{\alpha,\omega}(\beta) \qquad (2.21)$$
$$\text{subject to} \quad \eta = X\beta + \beta_0\mathbf{1} + \eta^0$$

Note that the linear term is indeed the gradient of $\ell(\cdot)$ at $\eta^{(k)}$; however, $W^{(k)}$ need not be $\nabla^2\ell(\eta^{(k)})$. In general, our strategy is to find a positive diagonal matrix $W^{(k)}$ that majorizes $\nabla^2\ell(\eta^{(k)})$, i.e. $0 \prec W^{(k)}$ diagonal and $\nabla^2\ell(\eta^{(k)}) \preceq W^{(k)}$. Of course, if $\nabla^2\ell(\eta^{(k)})$ is already positive and diagonal, we may simply take $W^{(k)} \equiv \nabla^2\ell(\eta^{(k)})$. The proximal operator (2.21) is simply an instance of the group elastic net problem under the Gaussian loss given by (2.1). Hence, we may use our algorithm in Section 2.3 to solve (2.21). We summarize our algorithm in Algorithm 4. This algorithm is referred to as *iteratively reweighted least squares* (IRLS) in Friedman et al. [2010], McCullagh and Nelder [1989].

2.4.1 Application on generalized linear models

Our primary application of Algorithm 4 is when $\ell(\cdot)$ represents the negative log-likelihood of a GLM. Some examples include the Binomial family with the logit or probit link function, Poisson family with the log link function, and the Cox family. In terms of the solver, we do not make any special changes based on the loss function. We note that this is one of the more notable deviations from the approach of Friedman et al. [2010], as the authors made amends to the solver specialized for each

GLM, which we found unnecessary.

The only manner in which the loss function plays a role in Algorithm 4 is in computing the gradient $\nabla\ell(\eta^{(k)})$ and the hessian-like quantity $W^{(k)}$. As a result, we have structured our software to abstract out the loss function to a class that provides the gradient and the hessian, similar to how we abstracted out the matrix classes (see Section 2.3.4). As an example, for the Binomial family with the logit link function [McCullagh and Nelder, 1989] where the loss is given by

$$\ell(\eta) := \sum_{i=1}^{n} w_i \left(-y_i\eta_i + \log(1+\exp(\eta_i))\right)$$

we implement a class to compute the gradient and hessian, respectively, given by

$$\nabla\ell(\eta)_i = -w_i(y_i - p(\eta_i)), \quad \nabla^2\ell(\eta)_{ij} = \begin{cases} w_i p(\eta_i)(1-p(\eta_i)), & i=j \\ 0, & i \neq j \end{cases} \tag{2.22}$$

where $p(t) := 1/(1+\exp(-t))$ is the sigmoid function.

2.4.2 Screen sets and the KKT check

We may reuse the strong rule mentioned in Section 2.3.6 when solving (2.20). We follow the notation as in Section 2.3.6. The only change we make is in computing the scores so that we discard group $g \notin \tilde{\mathcal{S}}$ if

$$\left\| X_g^\top \nabla\ell(X\tilde{\beta} + \tilde{\beta}_0\mathbf{1} + \eta^0) \right\|_2 < 2\lambda - \tilde{\lambda},$$

Similarly, we check the KKT condition by verifying that

$$\left\| X_g^\top \nabla\ell(X\beta + \beta_0\mathbf{1} + \eta^0) \right\|_2 \leq \lambda, \quad \forall g \notin \mathcal{S}$$

2.4.3 Numerical stability issues

In (2.21), it is possible that $W^{(k)}$ is nearly semi-definite, which may lead to divergence issues. For example, in the case of a Binomial family (logistic regression), if the estimated probabilities $p_i^{(k)} := 1/(1+\exp(-\eta_i^{(k)}))$ are near 0 or 1, then $W_{ii}^{(k)}$ will be near 0 (see (2.22)). Hence, we majorize $W^{(k)}$ one more time by taking the maximum of each diagonal entry with a small positive number (e.g. 10^{-12}) to guarantee positiveness of $W^{(k)}$. The authors of Friedman et al. [2010] only make this change for the Binomial family, but we see no reason why this should not be done for any other loss functions.

2.4.4 Convergence criterion

We discuss our convergence criterion. At iteration $k \geq 0$, upon solving the proximal operator (2.21), we check that

$$\left| \left(\eta^{(k+1)} - \eta^{(k)} \right)^\top \left(\nabla \ell(\eta^{(k+1)}) - \nabla \ell(\eta^{(k)}) \right) \right| \leq \epsilon \cdot n_{\mathcal{A}} \tag{2.23}$$

and declare convergence if the condition holds. Here, $n_{\mathcal{A}}$ denotes the current number of coefficients (not groups) that were updated and $\epsilon \geq 0$ is a user-specified tolerance level. We explain our intuition behind this criterion below.

First, suppose we perform a first-order Taylor expansion of $\nabla \ell(\cdot)$ around $\eta^{(k)}$,

$$\nabla \ell(\eta) \approx \nabla \ell(\eta^{(k)}) + \nabla^2 \ell(\eta^{(k)})(\eta - \eta^{(k)})$$

Then, we see that

$$\left(\eta^{(k+1)} - \eta^{(k)} \right)^\top \left(\nabla \ell(\eta^{(k+1)}) - \nabla \ell(\eta^{(k)}) \right) \approx \left(\eta^{(k+1)} - \eta^{(k)} \right)^\top \nabla^2 \ell(\eta^{(k)}) \left(\eta^{(k+1)} - \eta^{(k)} \right) \tag{2.24}$$

which measures the change in linear predictions weighted accordingly based on the hessian at $\eta^{(k)}$. This way, a change in the prediction along a direction for which the hessian is small is downweighted, as this suggests the loss $\ell(\cdot)$ is only mildly affected by such a change. Conversely, a change along a direction for which the hessian is large is upweighted, as this suggests the loss $\ell(\cdot)$ may be largely affected.

Next, we argue that (2.23) is a generalization of the convergence metric in Section 2.3.7. To see this, first replace $\eta = X\beta$, where we ignore the intercept and the offset momentarily for simplicity. Then, from (2.24) we have that

$$\left(\eta^{(k+1)} - \eta^{(k)} \right)^\top \nabla^2 \ell(\eta^{(k)}) \left(\eta^{(k+1)} - \eta^{(k)} \right) = \Delta\beta^\top X^\top \nabla^2 \ell(\eta^{(k)}) X \Delta\beta \tag{2.25}$$

where $\Delta\beta := \beta^{(k+1)} - \beta^{(k)}$. Under the Gaussian loss where $\nabla^2 \ell(\eta^{(k)}) \equiv W$, the right side of (2.25) becomes $\|X\Delta\beta\|_W^2$. Now, suppose (as in the BCD algorithm) that $\beta^{(k+1)}$ only differs in one group from $\beta^{(k)}$. That is, for some $g \in [G]$, $\beta_g^{(k+1)} \neq \beta_g^{(k)}$ and $\beta_{g'}^{(k+1)} \equiv \beta_{g'}^{(k)}$ for all $g' \neq g$. Then, $\|X\Delta\beta\|_W^2 = \|X_g \Delta\beta_g\|_W^2$. In (2.19), we scale this quantity by p_g, which is the total number of coefficients that were updated (since we have fixed all other coordinates). Hence, we scale by the analogous quantity in (2.23).

2.5 Algorithms for multi-response data

In this section, we discuss methods to fit multi-response data. In multi-response data, we have a response *matrix* $y \in \mathbb{R}^{n \times c}$ rather than a response vector. As a result, we change the group elastic net objective (2.20) to

$$\begin{aligned} \underset{\beta_0 \in \mathbb{R}^c, B \in \mathbb{R}^{p \times c}}{\text{minimize}} \quad & \ell(\text{vec}(\eta^\top)) + \lambda P_{\alpha,\omega}(\text{vec}(B^\top)) \\ \text{subject to} \quad & \eta = XB + \mathbf{1}\beta_0^\top + \eta^0 \end{aligned} \tag{2.26}$$

where $\text{vec}(\cdot)$ is the operator that flattens a matrix into a one-dimensional array by stacking the columns. Note that we must now fit a matrix of coefficients along with an intercept vector $\beta_0 \in \mathbb{R}^c$ (one for each response).

While it is possible to specify any arbitrary group structure on B, we focus only on two types: *grouped* and *ungrouped*. The grouped method defines each *row* of B as a group whereas the ungrouped method defines each *coefficient* of B as a group. Hence, the grouped method yields a non-trivial group lasso problem and the ungrouped method reduces to a lasso problem. The penalty $P_{\alpha,\omega}$ simplifies to the following respective forms

$$\begin{aligned} P_{\alpha,\omega}^{\text{grouped}}(\text{vec}(B^\top)) &= \sum_{g=1}^{p} \omega_g \left(\alpha \left\| B_{g\cdot} \right\|_2 + \frac{1-\alpha}{2} \left\| B_{g\cdot} \right\|_2^2 \right) \\ P_{\alpha,\omega}^{\text{ungrouped}}(\text{vec}(B^\top)) &= \sum_{g=1}^{p} \omega_g \left(\alpha \left\| B_{g\cdot} \right\|_1 + \frac{1-\alpha}{2} \left\| B_{g\cdot} \right\|_2^2 \right) \end{aligned}$$

Although it may seem like (2.26) requires a different solver, we now demonstrate that it can be easily solved with the tools built in Sections 2.3 and 2.4. It is not difficult to show that the constraint in (2.26) can be rewritten as

$$\text{vec}(\eta^\top) = (X \otimes I_c)\,\text{vec}(B^\top) + (\mathbf{1} \otimes I_c)\beta_0 + \text{vec}(\eta^{0\top}) \tag{2.27}$$

Written this way, with the data matrix $[(\mathbf{1} \otimes I_c) \quad (X \otimes I_c)] \in \mathbb{R}^{nc \times (1+p)c}$ and the coefficient vector $(\beta_0, \text{vec}(B^\top)) \in \mathbb{R}^{(1+p)c}$, we return to the setting of (2.20) (with no intercept). As discussed in Section 2.3.4, as soon as we implement a matrix to represent the Kronecker product with an identity matrix, we are able to solve (2.26) with our existing tools. Such a matrix can be implemented to compute the gradient and residual update efficiently due to its structure. For the Multigaussian family where the loss function is given by

$$\ell(\eta) = \sum_{i=1}^{n} w_i \left(-\sum_{j=1}^{c} y_{ij}\eta_{ij} + \frac{\left\| \eta_{i\cdot} \right\|_2^2}{2} \right)$$

we may simply rely on our algorithm that solves the group elastic net under the Gaussian loss (see (2.1)). Otherwise, we must rely on our algorithm that handles general convex losses (see (2.20)).

We make a special remark on the Multinomial family where the loss is given by

$$\ell(\eta) := \sum_{i=1}^{n} w_i \left(-\sum_{j=1}^{c} y_{ij}\eta_{ij} + \log\left(\sum_{j=1}^{c} \exp(\eta_{ij})\right)\right)$$

Unlike the GLMs discussed previously, this marks the first example where the hessian $\nabla^2\ell(\eta)$ is not exactly positive and diagonal. Observe that the model is not identifiable in the sense that for any fixed $\{\beta_{0,j}, \beta_{\cdot,j}\}_{j=1}^{c}$, we have that $\{\beta_{0,j} - d_0, \beta_{\cdot,j} - d\}_{j=1}^{c}$ yields the same loss for any $d_0 \in \mathbb{R}$ and $d \in \mathbb{R}^p$. As a result, the hessian is never positive definite. Moreover, the hessian is only block-diagonal. Consequently, Friedman et al. [2010] approach the Multinomial family quite differently from the other GLMs. First, they tweak the solver to cycle over the classes $j = 1, \ldots, c$ while fixing all other class coefficients, so that the gradient and hessian in (2.21) is only with respect to the linear predictors for class j, namely, $\eta_{\cdot,j}$. Because of this modification, given a candidate solution (β_0, β), they found it necessary to then further minimize only the lasso penalty with respect to d_0 and d to improve the convergence speed. Despite these efforts, we have found a real dataset on which their software `glmnet` fails to converge. We pursue a different and much simpler approach. We simply need to construct a positive, diagonal matrix that majorizes $\nabla^2\ell(\eta)$. Simon et al. [2013b] show that $2\,\mathrm{diag}(\nabla^2\ell(\eta))$ is a valid choice for dominating $\nabla^2\ell(\eta)$. Hence, we set $W^{(k)} := 2\,\mathrm{diag}(\nabla^2\ell(\eta^{(k)}))$ in (2.21) and reuse our work from Section 2.4. With this approach, our software is able to converge. This method implicitly settles the identifiability issue precisely because $W^{(k)}$ is now positive definite (or can be made positive definite as in Section 2.4.3). Out of curiosity, we also experimented with minimizing the penalty with respect to d_0 and d, but to our surprise, this resulted in a less stable algorithm often diverging.

2.6 Benchmark

In this section, we show an assortment of benchmarks to demonstrate the power of our method. We developed a highly efficient implementation of our algorithms discussed in Sections 2.3 to 2.5 in our package `adelie` where most of the code is written in `C++`. Although we intend `adelie` to be a `Python`-first package, we are currently developing a `R` package as well[3]. We exported enough functionalities to perform the benchmark comparisons in this section. We note that both the `Python` and `R` packages share the same `C++` code, which performs most of the computation, and the only difference comes from the input preparation for the `C++` solver. Hence, there is only a negligible difference in their runtimes. All benchmarks were run on an M1 Macbook Pro.

[3] https://github.com/JamesYang007/adelie-r

Name	Version	Path	C-GLM	C-Matrix	Algorithm	Strong rule	Backend	Include
`adelie`	1.1.38	✓	✓	✓	BCD	✓	C++	✓
`gglasso`	1.5.1	✓			GMD	✓	Fortran	✓
`grpnet`	0.3	✓			GMD	✓	Fortran	✓
`sparsegl`	1.0.2	✓			GMD	✓	Fortran	✓
`grplasso`	0.4-7	✓	✓		GMD		R	
`SGL`	1.3	✓			AGMD		C++	
`grpreg`	3.4.0	✓			OBCD	✓	C	

Table 2.1: A table of key features of `adelie` and related `R` packages. Every package considered is a path-solver. Few have customizable GLM or matrix objects. Our package uniquely solves the group elastic net via block-coordinate descent (BCD) with an exact solution to the block update, whereas other packages use group-wise majorization descent (GMD) based methods or orthogonalization block-coordinate descent (OBCD). Many packages support strong rules and a low-level backend for the solver. We exclude `grplasso` and `SGL` due to their significantly slower performance; we exclude `grpreg` since OBCD does not solve the group elastic net problem exactly.

The packages we consider are `sparsegl` [Liang et al., 2022, McDonald et al., 2023], `gglasso` [Yang and Zou, 2015, Yang et al., 2024], `grpnet` [Helwig, 2024], `grplasso` [Meier et al., 2008, Meier, 2020], `SGL` [Simon et al., 2013a, 2019], and `grpreg` [Breheny and Huang, 2015]. We summarize in Table 2.1 the key features of each package. We only list the features that we believe are most relevant to the performance. First, every package we considered are path-solvers. Hence, we found it fair to compare the solver times on a (common) path of regularization values λ. `adelie` and `grplasso` are the only packages that support customizable GLM objects (C-GLM). Moreover, `adelie` is the only package that supports customizable matrix objects (C-Matrix). All other packages only support dense matrices and a few GLMs such as Binomial and Cox families. As a result, their implementations should, in theory, have more advantage than `adelie` as they can potentially apply more optimizations given these restrictions. `adelie` is the only package that supports a bona fide block-coordinate descent (BCD) algorithm in that each block update is solved exactly. `gglasso`, `grpnet`, `sparsegl`, and `grplasso` use a slight modification to BCD called *group-wise majorization descent* (GMD), which majorizes $X_g^\top W X_g$ by an isotropic matrix ηI (typically, η is the largest eigenvalue of $X_g^\top W X_g$) and minimizes the majorized loss. GMD can be seen as applying one iteration of proximal gradient descent at each block update. It can be shown that GMD strictly decreases the objective and converges to an optimum [Meier et al., 2008], however, each block update remains only an approximation. `SGL` further applies a Nesterov acceleration to GMD (AGMD). Finally, `grpreg` applies an orthogonalization prior to fitting group lasso via BCD (OBCD) and rescales the coefficients back to the original scale. As discussed in Section 2.3, this does not solve the original problem. Indeed, this was evident from observing that the objectives were clearly far from that of the other packages. For this reason, we do not include `grpreg` in our analysis. All packages except `grplasso` and `SGL`

Name	n	G	Task
Leukemia	72	7129	classification
UJIIndoorLoc	19937	465	regression
Prostate	102	6033	classification

Table 2.2: The number of samples (n) and groups (G) as well as the task for each of the real datasets.

use strong rule as discussed in Section 2.3.6. Most packages use a low-level language such as `C`, `C++`, or `Fortran` as the backend for the solvers except `grplasso`, which is fully written in `R`. We believe the lack of a low-level backend for `grplasso` and strong rule for `SGL` are the primary reasons for their significantly slower performance that ultimately led us to omit them from the analysis.

For our real data analyses, we consider the following datasets:

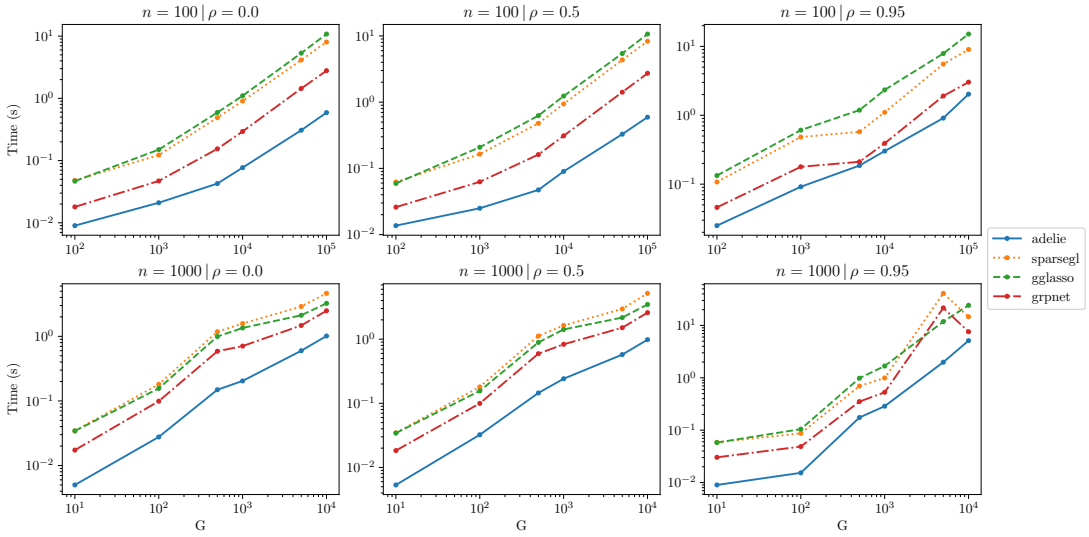
- Leukemia dataset [Golub et al., 1999]: the Leukemia dataset has been previously described in Section 2.3. However, in this section, we will use it to classify whether an individual has Leukemia.
- UJIIndoorLoc dataset [Joaqun Torres-Sospedra, Raul Montoliu, Adolfo Martnez-Us, Tomar Arnau, Joan Avariento, 2014]: the UJIIndoorLoc dataset contains WiFi fingerprints of Received Signal Strength Intensity (RSSI) from location trackers. We remove any redundant features from the dataset. Our task is to predict the longitude coordinate of the sample.
- Prostate dataset [Singh et al., 2002]: the Prostate dataset gene expressions from a microarray study of prostate cancer. Our task is to classify whether an individual has prostate cancer.

We summarize the datasets in Table 2.2.

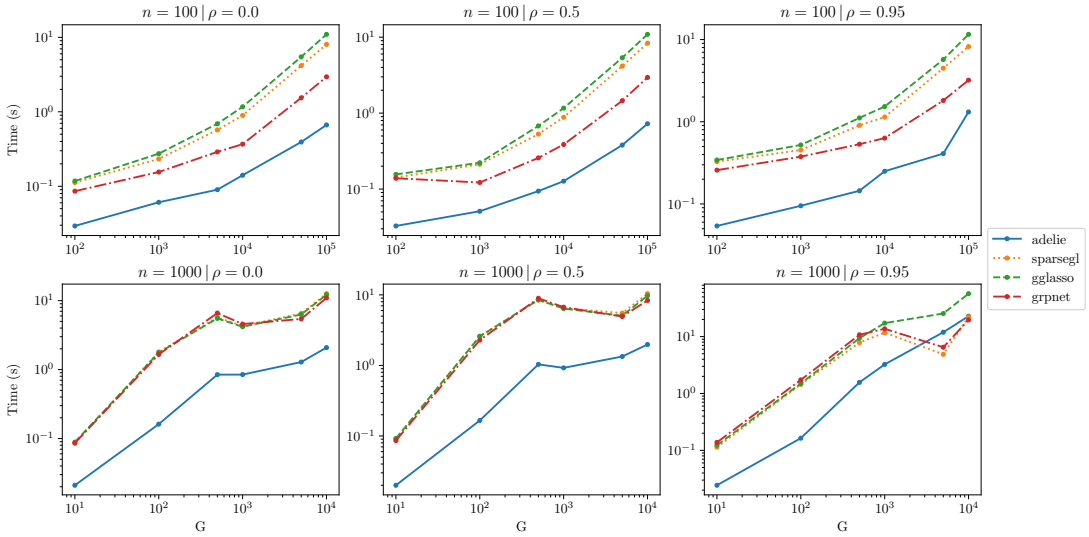
2.6.1 Group lasso comparisons

We show a comparison of solving the group lasso under the Gaussian and Binomial losses using both simulated and real data.

We first describe our simulation setup. Our setup is similar to the one used by Yuan and Lin [2006]. Let n be the number of samples, G be the number of groups, and $\rho \in [0, 1]$ be the equi-correlation between features. We generate $Z_i \sim \mathcal{N}(0, I_G)$ for $i \in [n]$ and $W \sim \mathcal{N}(0, I_n)$ all independent. We then define a sample $Y_i = \sqrt{\rho} W_i \mathbf{1} + \sqrt{1-\rho} Z_i$ so that the features have variance 1 and equi-correlation ρ. Finally, we construct our design matrix X with a third-degree polynomial expansion of each $Y_{i,g}$ so that $X_{i,3g-2} = Y_{i,g}$, $X_{i,3g-1} = Y_{i,g}^2$, and $X_{i,3g} = Y_{i,g}^3$ for all $i \in [n]$ and $g \in [G]$, so that $X \in \mathbb{R}^{n \times 3G}$. Naturally, we define every 3 columns of X as a group. The natural parameter η is generated from a linear model $\eta = X\beta + \sigma\varepsilon$. The first 6 components of β are independently generated from a standard normal and the rest are set to 0. The noise $\varepsilon \sim \mathcal{N}(0, I_n)$ and σ is such that the signal-to-noise ratio is 3. For the Gaussian loss, our response y is identical to



(a) Gaussian Loss



(b) Binomial Loss

Figure 2.4: Timing comparisons for solving the group lasso under the Gaussian loss (Figure 2.4a) and the Binomial loss (Figure 2.4b) against existing `R` packages. We study a small ($n = 100$) and large ($n = 1000$) sample size case, and for each case we vary the (equi-)correlation of the features (ρ).

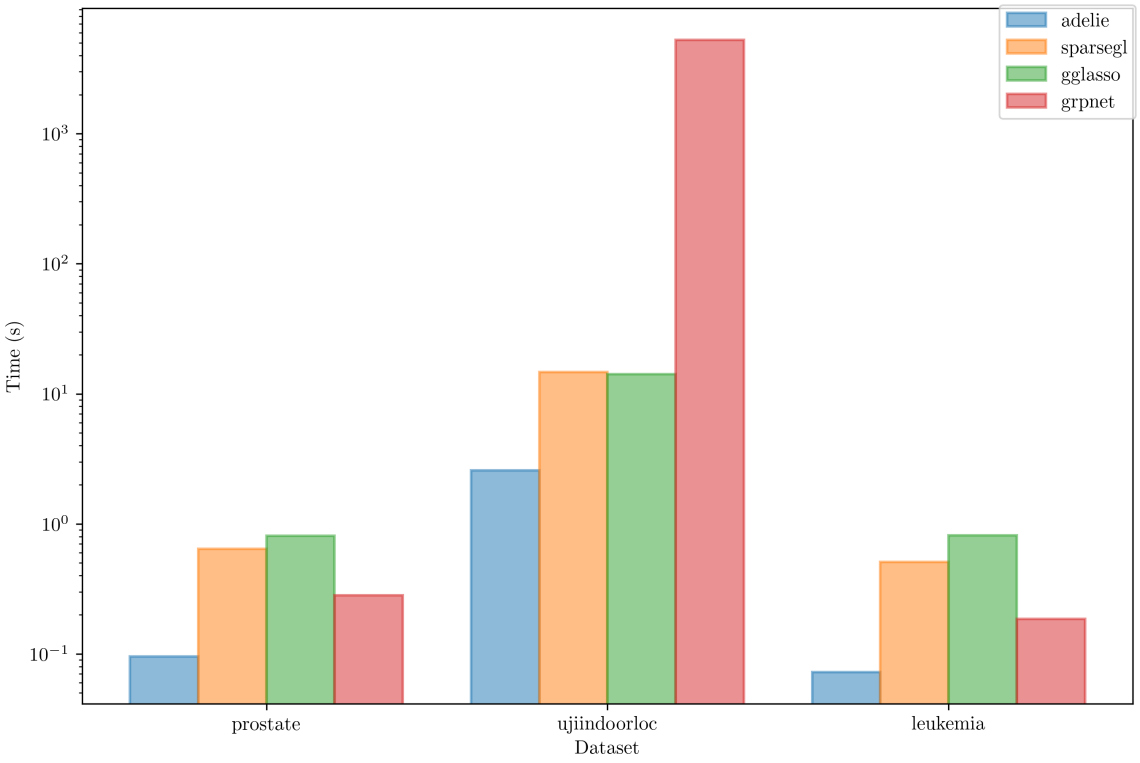


Figure 2.5: Timing comparisons for solving the group lasso for the real datasets against existing `R` packages.

η, and for the Binomial loss, we generate y as Bernoulli random variables where the logit parameters are given by η. Finally, prior to fitting, we center and standardize the columns of X and y.

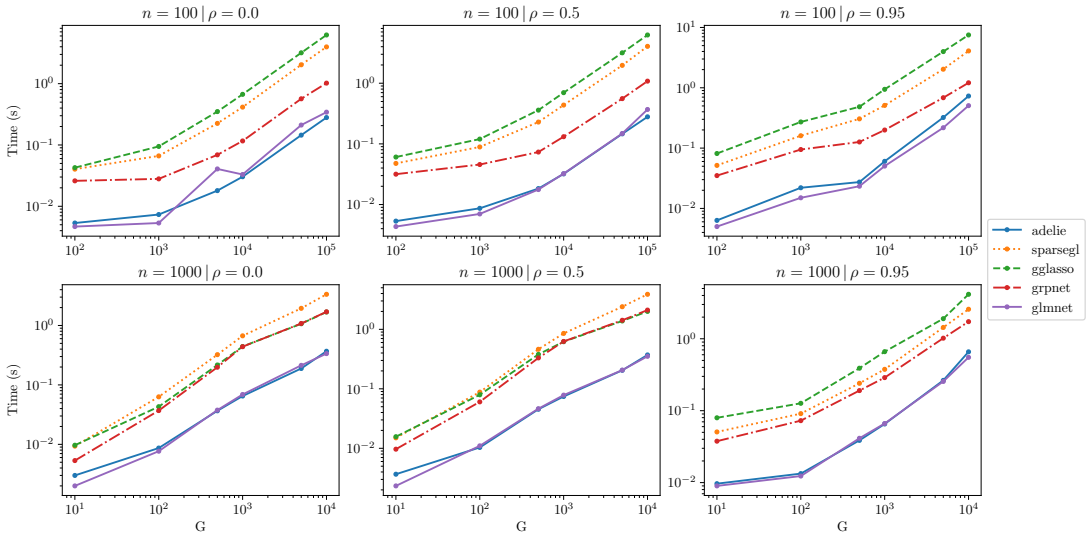
In Figure 2.4, we show the timings for solving the group lasso under the Gaussian loss (Figure 2.4a) and the Binomial loss (Figure 2.4b). We study a small ($n = 100$) and large ($n = 1000$) sample size case, and for each case we vary the (equi-)correlation of the features (ρ). Each plot shows the average times across 3 trials to solve the same path of 100 regularization values evenly-spaced on the log-scale starting at $\lambda_{\max}$, the smallest λ for which $\mathbf{0}$ is an optimal solution, until $0.01\lambda_{\max}$. In all cases except one ($n = 1000$, $\rho = 0.95$, $G = 5000$, Binomial loss), `adelie` runs uniformly faster than the other packages. `adelie` is up to 10 times faster than the next fastest package and, more importantly, this difference factor remains to be the case as the data scales both in n and G. In general, `grpnet` is faster than `gglasso` and `sparsegl`. Upon inspecting the code, we believe this is because `grpnet`'s implementation incurs much less memory allocation cost. In the one case where `adelie` was not the fastest, we observed that the screen set via the strong rule pulled in a large number of groups from the very beginning. Combined with the fact that our convergence criterion is different from that of the other packages, `adelie` ended up requiring a larger number of coordinate descent iterations and therefore a relatively longer runtime for this particular example. We are currently working to improve the strong rule to ameliorate such cases.

We briefly mention that we tried the following setting with a larger group size of $s = 100$ rather than 3. In this setting, we generate $Y \in \mathbb{R}^{n \times sG}$ as before such that every feature is Gaussian with mean 0, variance 1, and equi-correlated with ρ. We then group every s features so that we have G number of groups. The response vector y is generated in the same manner as before. Every package except `adelie` struggled to converge as their objective values were unacceptably far from that of `adelie`; we verified from the KKT conditions that `adelie` converged properly. For this reason, we omit the timing comparison for this setting. This highlights the value in solving each block update exactly rather than minimizing a majorized loss; our method remains stable even with larger group sizes.

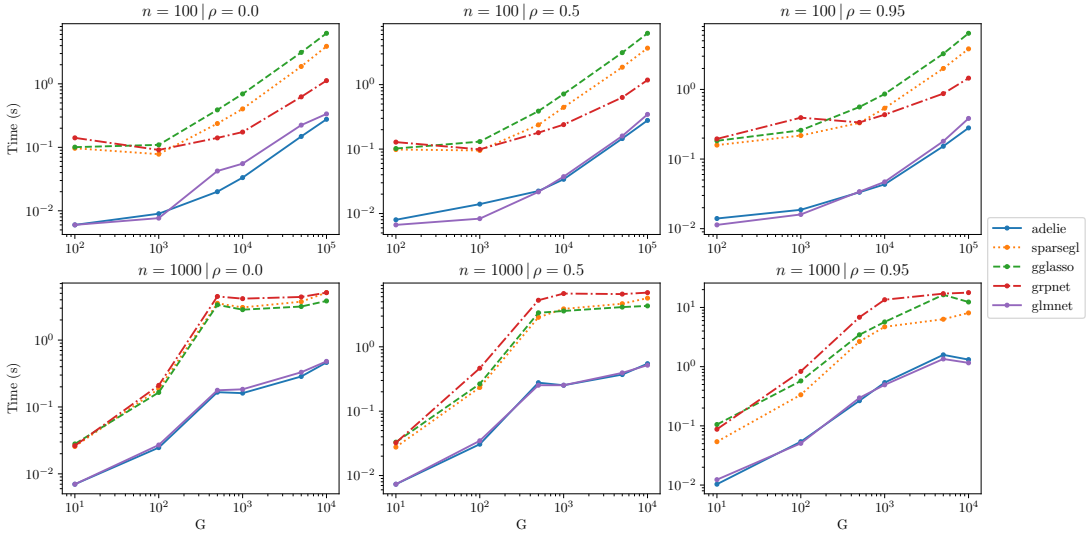
For the real datasets, we first expand each feature (column) x into three polynomial terms x, x^2, and x^3 just as with the Leukemia dataset in Section 2.3 so that our feature matrix $X \in \mathbb{R}^{n \times 3G}$. Prior to fitting, we center and standardize the columns of X and y. In Figure 2.5, we show the average timings to solve group lasso over 3 trials. Similar to the simulation analysis, we solve for the same path of 100 regularization values evenly-spaced on the log-scale for all packages. Once again, we observe a dramatic speed improvement with `adelie`.

2.6.2 Lasso comparisons

In this section, we demonstrate that `adelie` is a competitive lasso solver. We also include `glmnet` [Friedman et al., 2010] in the list of packages for comparison. As in Section 2.6.1, we show a timings comparison of solving the lasso under the Gaussian and Binomial losses using simulated and real



(a) Gaussian Loss



(b) Binomial Loss

Figure 2.6: Timing comparisons for solving the lasso under the Gaussian loss (Figure 2.6a) and the Binomial loss (Figure 2.6b) against existing `R` packages. We study a small ($n = 100$) and large ($n = 1000$) sample size case, and for each case we vary the (equi-)correlation of the features (ρ).

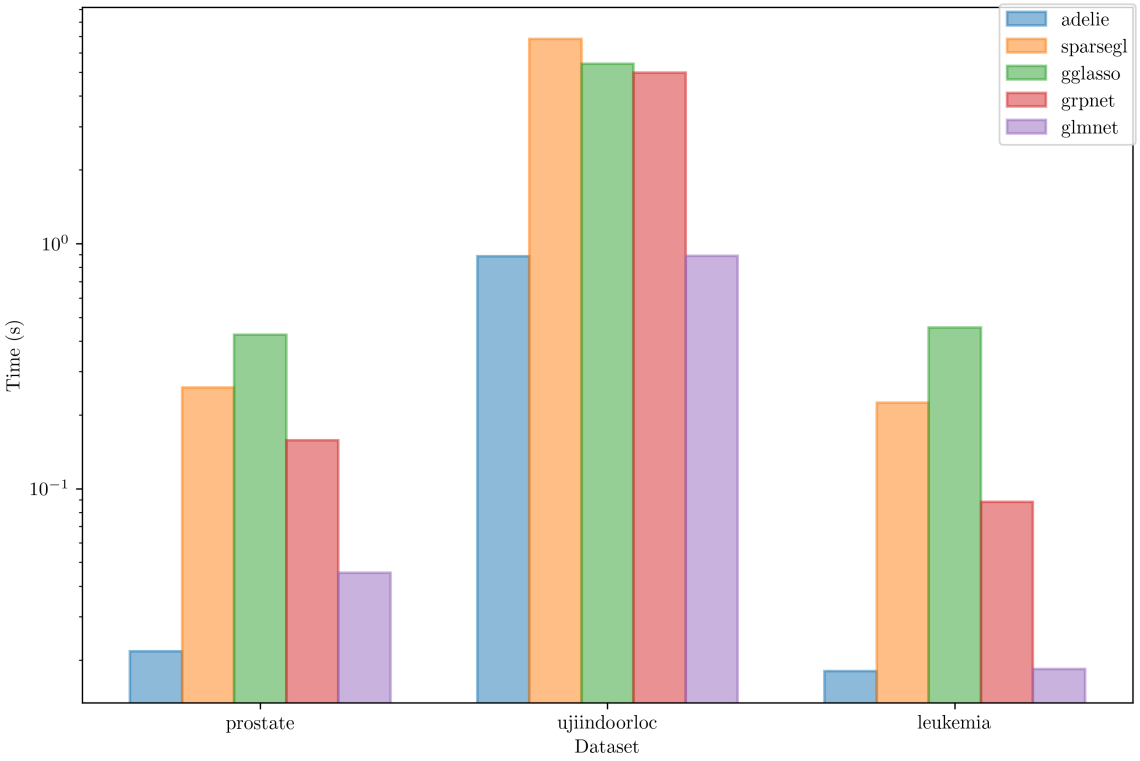


Figure 2.7: Timing comparisons for solving the lasso for the real datasets against existing `R` packages.

data.

Our simulation setup is taken from Friedman et al. [2010]. For completion, we describe it next. We begin by generating n independent samples with p features from a Gaussian distribution where every feature is equi-correlated with ρ. The natural parameter $\eta = X\beta + \sigma\epsilon$ where $\beta_j = (-1)^j \exp(-2(j-1)/20)$, $\epsilon \sim \mathcal{N}(0,1)$, and σ is chosen such that the signal-to-noise ratio is 3. Note that the coefficients have alternating signs and decrease exponentially in magnitude. For the Gaussian loss, we let our response $y \equiv \eta$ and for the Binomial loss, we generate y from the Bernoulli distribution where the logit parameter is given by η. Prior to fitting, we center and standardize the columns of X and y.

For the real datasets, we do not perform any special modifications to the data as in the group lasso setting except for centering and standardizing the columns of X and y.

In Figures 2.6 and 2.7, we show the timings for solving the lasso under the Gaussian loss (Figure 2.6a) and the Binomial loss (Figure 2.6b) for the simulated and real data setting, respectively. We interpret the figures in the same way as in Figures 2.4 and 2.5 for the group lasso comparisons in Section 2.6.1. We note that `adelie` and `glmnet` have nearly identical timings and they clearly outperform the other packages. This shows that `adelie` is not only a fast group lasso solver, but also a fast lasso solver, suffering no loss of performance compared to the highly optimized and specialized lasso solver `glmnet`.

2.7 Polygenic risk scores with local ancestry information

In this section, we discuss our motivating application of group lasso in genomics. Specifically, we use group lasso to construct polygenic risk scores (PRS) using both genotype data as well as local ancestry information (LAI). As the current state-of-the-art methods for constructing PRS only use the genotype data, we study whether LAI provides meaningful improvements to the prediction power of PRS specifically to admixed populations, i.e., populations with a diverse set of LAI. Although this part of the work is still under development, we show preliminary results that seem promising.

We are given phased genotype data $Z^1, Z^2 \in \{0,1\}^{n \times s}$ where the number of individuals $n \approx 250,000$ and the number of single nucleotide polymorphisms (SNPs) $s \approx 1,000,000$ (exact numbers depending on the data source e.g., UK biobank [Sudlow et al., 2015], Mexico biobank [Sohail et al., 2023], and All-of-Us [Bick et al., 2024]). Moreover, we often observe some sparsity in the genotype data of around 30% non-zero values in each Z^h $(h = 1, 2)$. Here, $Z^h_{ij} = 1$ indicates a mutation in the jth SNP for individual i in haplotype h. We assume no missing data, as we often impute such entries using `Beagle` [Browning et al., 2021]. We are also given a corresponding LAI data $L^1, L^2 \in [A]^{n \times s}$ where A is the number of ancestries and L^h_{ij} indicates the ancestry of the jth SNP for individual i and haplotype h. These ancestries indicate the group of population that most attribute to the origin of the SNP. Examples of ancestries include European (EUR), African (AFR), East Asian (EAS),

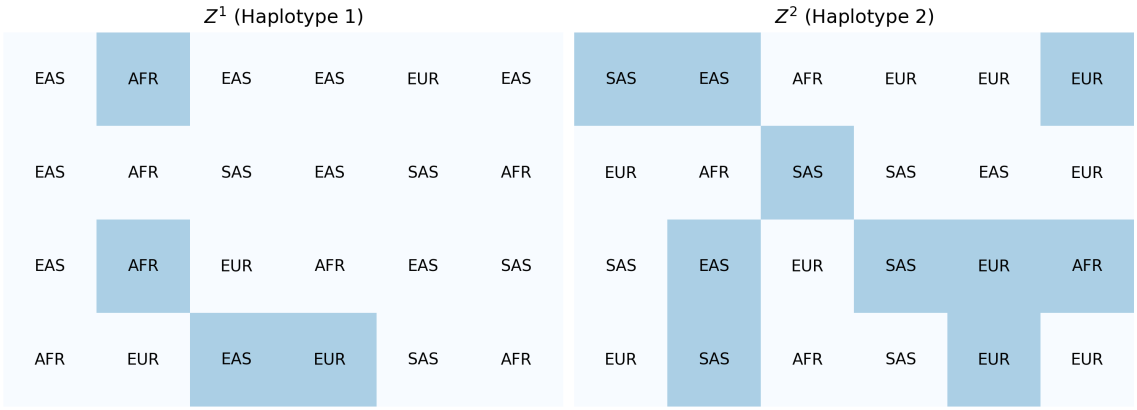


Figure 2.8: Visualization of the phased genotype and LAI data. For each haplotype $h = 1, 2$, we show a few columns of the data where a darker cell indicates $Z_{ij}^h = 1$ and each ancestry L_{ij}^h is labeled in the center of the cells.

and South Asian (SAS). The exact set of ancestries is determined by the source of the data. We also assume that LAI data contains no missing data as imputation methods such as `MendelImpute` [Chu et al., 2021] are often used in practice. Figure 2.8 shows a visualization of the phased genotype and LAI data.

We now discuss our encoding scheme of the genotype and LAI data. At a high level, for each SNP, we construct auxiliary columns corresponding to each ancestry label. We then count the number of mutations in each ancestry. Figure 2.9 demonstrates the encoding scheme using four individuals on SNP 57 as an example. The first individual contains no mutations in SNP 57 for either haplotypes, so the encoded vector is $\mathbf{0}$. The second individual contains a mutation in haplotype 2 labeled AFR, so the corresponding encoded entry is incremented to 1 while all other entries are kept at 0. The third individual contains a mutation in both haplotypes labeled EUR and EAS, respectively, so the corresponding encoded entries are incremented to 1. The last individual contains a mutation in both haplotypes labeled SAS, so the corresponding encoded entry is incremented to 2. This encoding is performed for each SNP, resulting in a fully encoded matrix $\tilde{Z} \in \mathbb{R}^{n \times As}$. Naturally, we group every A columns of $\tilde{Z}$ as they are artificially constructed columns corresponding to a physical entity (SNP). We also include dense covariates $D \in \mathbb{R}^{n \times 12}$ consisting of age, sex, and 10 principal components of $Z^1 + Z^2$, which is a common practice when constructing PRS [Qian et al., 2020, Li et al., 2020, 2021]. The concatenated matrix $X = [D\ \tilde{Z}]$ is the full feature matrix we consider. For any phenotype $y \in \mathbb{R}^n$, we solve the group lasso problem using the Gaussian loss for quantitative phenotypes and the Binomial loss for binary phenotypes to construct the PRS. Note that the dense covariates are unpenalized so that the group lasso penalty only affects coefficients corresponding to $\tilde{Z}$.

As mentioned in Section 2.3.4, we implement a custom matrix class to represent our encoded

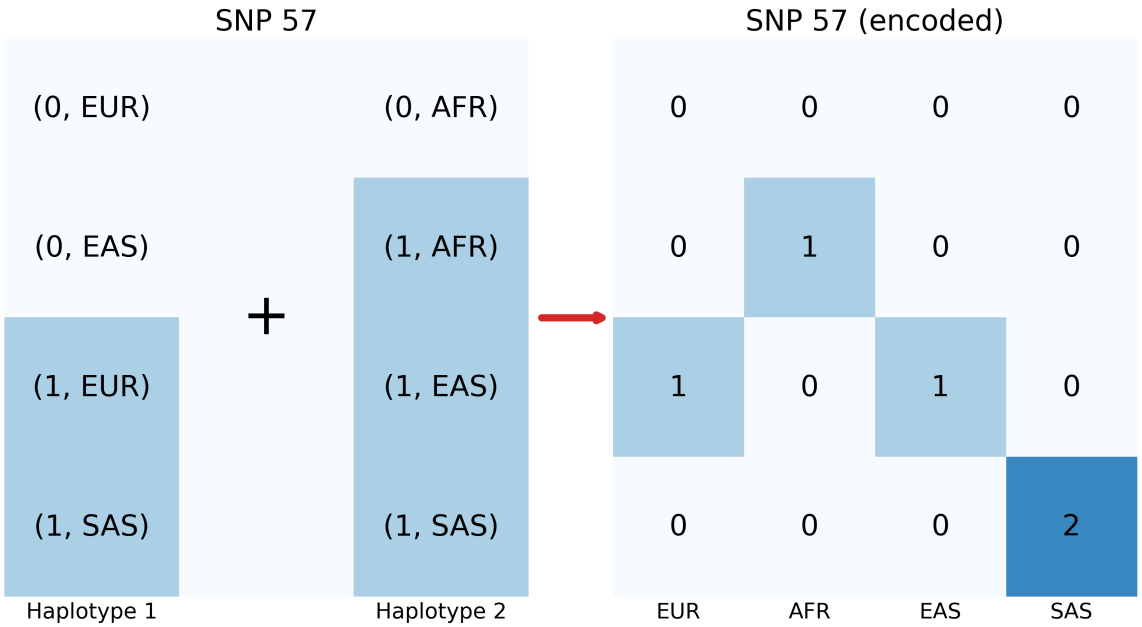


Figure 2.9: Visualization of the encoding scheme of the phased genotype and LAI data. For each SNP, we construct auxiliary columns corresponding to each ancestry label. In this example, we focus on the four ancestry labels EUR, AFR, EAS, and SAS. We then count the number of mutations in each ancestry.

matrix $\tilde{Z}$. We do not discuss the full detail of the memory representation of $\tilde{Z}$ for brevity, however, we mention some key points. First, $\tilde{Z} = \tilde{Z}^1 + \tilde{Z}^2$ where each $\tilde{Z}^h \in \{0,1\}^{n \times As}$ is a binary encoding matrix for haplotype $h = 1, 2$. Hence, we only store the $\tilde{Z}^h$ and perform the addition on-the-fly. Since $\tilde{Z}^h$ is binary, we only need to store the non-zero row indices for each column. The row indices are blocked in chunks of size 256 and each block is indexed using a 64-bit unsigned integer. For example, a row index $i = 256b + r$ where b is the block index and $r \in \{0, \ldots, 255\}$ is the relative index within the block. The relative indices are stored using an 8-bit unsigned integer. If a block contains no non-zero indices, then it can be omitted in the serialization. Hence, this representation is ideal in the setting where non-zero indices are clustered together so that there are few blocks to store, each block containing many non-zero relative indices. In practice, we observe only about a 10-20% increase in memory compared to the raw data, suggesting we are close to the ideal setting. We implement optimized matrix operations with parallelization using this memory representation.

Our method closely models the work of Qian et al. [2020], which solves the lasso problem on only the genotype data. In detail, the authors only consider $Z := Z^1 + Z^2$ and use the full feature matrix $X = [D\ Z]$ with the same dense covariates D (unpenalized). They then solve the lasso with X and the phenotype y using the appropriate GLM depending on the nature of y. They provide a custom lasso solver, `snpnet`, designed to exploit the structure of Z for faster computation. We believe that

	adelie			snpnet		
Phenotype	All	Admx	Time (m)	All	Admx	Time (m)
log(WBC) (Q)	.0563	**.0857**	20	.0570	.0510	10
CRP (Q)	.0225	**.0636**	8	.0231	.0207	2
COR (B)	.7967	.8177	13	.7869	.8190	11
T2D (B)	.7037	.7363	21	.6916	.7482	36
PRO (B)	.9300	.9647	7	.9252	.9655	19

Table 2.3: Comparison of prediction power and runtime between `adelie` and `snpnet` for various phenotypes. Each phenotype is either quantitative (Q) or binary (B). For quantitative phenotypes, we use R^2 as the goodness-of-fit metric; for binary phenotypes, we use AUC-ROC. For both methods, we fit the corresponding model using all the training data. The "Time" column corresponds to the training time in minutes. The "All" column corresponds to the best metric evaluated on all test individuals. The "Admx" column corresponds to the best metric evaluated on *only* admixed latino test individuals.

our methodology is a bona fide extension of this result since the methods coincide if $A = 1$, i.e., all individuals are labeled the same ancestry, thereby effectively removing LAI from the data.

In Table 2.3, we show our preliminary results comparing our method using `adelie` to `snpnet`. We use the All-of-Us dataset as it offers over $400,000$ individuals, millions of SNPs, and a diverse group of individuals. The UK biobank offers a predominantly white-British population and, therefore, does not provide enough admixed individuals for a meaningful analysis; the Mexico biobank, while highly admixed, only offers a small fraction of individuals. We consider five phenotypes: (log of) white blood counts (WBC), C-reactive protein (CRP), coronary artery disease (COR), type 2 diabetes (T2D), and prostate cancer (PRO). We use R^2 to measure goodness-of-fit for quantitative traits and AUC-ROC for binary traits. For each method, we fit the corresponding model using all the training data. The training time in minutes is given by the "Time" column. Note that both methods were run on the same machine with 64 cores. Finally, we compute our metrics on two sets of test individuals: 1) all test individuals ("All") and 2) only admixed latino test individuals ("Admx"). Both methods perform similarly when predicting on all test individuals. Our method seems to improve the predictions for the quantitative traits when restricted to the admixed latino population. Moreover, this improvement seems significant as `snpnet` is unable to achieve the same improvement and, in fact, achieves a slightly lower metric in "Admx" compared to that of "All". In all other cases, there does not seem to be a significant difference in the performance between `adelie` and `snpnet`, which suggests that these traits are not sensitive to LAI. While we plan to perform a more comprehensive analysis across all the phenotypes, we conclude that incorporating LAI at least does not degrade and sometimes significantly improves the PRS for some phenotypes on the admixed latino population.

2.8 Discussion

We demonstrate that block-coordinate descent is a powerful and effective method for solving the group elastic net problem. We derive a novel algorithm to solve the block-coordinate update for the Gaussian loss, outperforming existing popular proximal gradient descent based methods in both performance and accuracy. Moreover, our method naturally generalizes to other convex losses via the proximal (quasi)-Newton method. Through simulated and real data examples, we demonstrate that our method is remarkably fast for both group lasso and lasso problems.

Our `Python` package `adelie` is available under the MIT license from PyPI (https://pypi.org/project/adelie) and GitHub (https://github.com/JamesYang007/adelie). Our documentation is hosted on GitHub Pages (https://jamesyang007.github.io/adelie). The user can install the latest version by running `pip install adelie` in the terminal.

Chapter 3

Pivot rule for discarding predictors for group lasso

3.1 Introduction

In this chapter, we develop a heuristic to discard predictors in the group lasso solver from Chapter 2. We focus on the penalized regression with group lasso penalty

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \; f(\beta) + \lambda \sum_{g=1}^{G} \omega_g \left\| \beta_g \right\|_2 \tag{3.1}$$

for some convex loss $f(\cdot)$. As a reminder, $\lambda \in \mathbb{R}_+$ is the regularization parameter, G is the total number of groups, and $\omega_g \in \mathbb{R}_+$ is the penalty factor for group g. In particular, we primarily focus on the Gaussian loss given by

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \; \frac{1}{2} \left\| y - X\beta \right\|_2^2 + \lambda \sum_{g=1}^{G} \omega_g \left\| \beta_g \right\|_2 \tag{3.2}$$

where $X \in \mathbb{R}^{n \times p}$ is the feature matrix, $y \in \mathbb{R}^n$ is the response vector, and $\beta = (\beta_1, \ldots, \beta_G) \in \mathbb{R}^p$ is the coefficient vector. Without loss of generality, we assume that the columns of X and the vector y are centered so that we may omit the intercept term from the model. We further absorb the observation weights into X and y for simplicity (see (2.1)). We note that when $G \equiv p$, i.e. there are p groups each of size 1, the group lasso becomes equivalent to the lasso problem [Tibshirani, 1996]. Hence, for brevity, we discuss our results in the general setting of the group lasso and note that all results apply for the lasso problem as well.

A powerful consequence of the group lasso penalty is that the solutions are typically sparse.

That is, if $\beta^\star(\lambda)$ is a solution to (3.1), we oftentimes observe many groups g such that $\beta_g^\star(\lambda) = \mathbf{0}$ [Tibshirani, 1996, Yuan and Lin, 2006, Meier et al., 2008]. This characteristic alone has driven a lot of research into deriving fast algorithms for solving these problems at large scales [Qian et al., 2020, Tibshirani et al., 2012, Friedman et al., 2010]. In fact, there is a large body of work in the prediction of the set of non-zero coefficients called the *active set*. Concretely, we define the active set as

$$\mathcal{A}(\lambda) := \left\{ g : \beta_g^\star(\lambda) \neq \mathbf{0} \right\},$$

the set of groups with non-zero coefficients at the solution $\beta^\star(\lambda)$. We wish to construct the smallest prediction set $\mathcal{S}(\lambda)$ containing $\mathcal{A}(\lambda)$. The motivation is as follows: if $\mathcal{S}(\lambda)$ correctly contains all of $\mathcal{A}(\lambda)$, then the solution to

$$\begin{aligned} \underset{\beta \in \mathbb{R}^p}{\text{minimize}} \quad & f(\beta) + \lambda \sum_{g=1}^{G} \omega_g \left\| \beta_g \right\|_2 \\ \text{subject to} \quad & \beta_g = 0 \quad \forall g \notin \mathcal{S}(\lambda) \end{aligned} \tag{3.3}$$

will also be optimal for (3.1). Moreover, we only need to solve (3.3) for the subset of coefficients $\beta_{\mathcal{S}(\lambda)}^\star(\lambda)$. This can greatly save computation cost since we only iterate over the subset of groups in $\mathcal{S}(\lambda)$ rather than the set of all groups. Hence, from a computation standpoint, it is ideal for $\mathcal{S}(\lambda)$ to be as small as possible.

While we have not found any results that perfectly predict the true active set, we discuss some previous works that nonetheless provide reasonable methods to perform these predictions. Suppose we wish to solve (3.2) with regularization λ and penalty $\omega_g \equiv 1$. In the context of the lasso problem (i.e. $G \equiv p$), Ghaoui et al. [2011] describes their "SAFE" rule which discards feature j if

$$\left| x_j^\top y \right| < \lambda - \left\| x_j \right\|_2 \left\| y \right\|_2 \frac{\lambda_{\max} - \lambda}{\lambda_{\max}}$$

where $\lambda_{\max} := \max_j \left| x_j^\top y \right|$ is the smallest regularization such that $\beta^\star(\lambda) = \mathbf{0}$. A remarkable result is that the SAFE rule gives a guarantee in that discarded features truly have zero coefficients in the optimal solution. That is, if feature j is discarded, then $\beta_j^\star(\lambda) = 0$. Although this guarantee seems attractive at first, we pay the price of over-conservativeness; in practice, we often see a large *bloating* of the prediction set as λ decreases.

Tibshirani et al. [2012] noticed this behavior and proposed a more aggressive screening rule called the "strong" (or "sequential strong") rule. For the lasso problem, if $\beta^\star(\lambda_0)$ is known and we wish to solve the lasso at $\lambda < \lambda_0$, the strong rule discards the jth predictor if

$$\left| x_j^\top (y - X\beta^\star(\lambda_0)) \right| < 2\lambda - \lambda_0$$

Similarly, in the group lasso setting, the gth group is discarded if

$$\left\|X_g^\top(y - X\beta^\star(\lambda_0))\right\|_2 < 2\lambda - \lambda_0$$

The authors also note that extensions are possible for elastic net [Zou and Hastie, 2005] and other objectives such as the logistic regression loss function. Unlike the SAFE rule, the strong rule does not offer any guarantees for the discarded groups in general. However, in practice, the strong rule performs surprisingly well for many examples, often creating much smaller prediction sets than the SAFE rule. Indeed, the strong rule is employed in the `R` package `glmnet` [Friedman et al., 2010] for solving the lasso.

Wang et al. [2015a] later showed that the SAFE rule can be further improved using their "EDPP" rule, which produces tighter prediction sets while still providing guarantees. The EDPP rule is based on dual-polytope projection (DPP), which heavily relies on the fact that the dual problem to (3.2) is a projection problem. Aside from the EDPP rule, there are several works revolving around DPP [Fercoq et al., 2015, Xiang et al., 2011, Xiang and Ramadge, 2012, Dai and Pelckmans, 2012, Lee et al., 2018]. For the more general group lasso problem, if $\beta^\star(\lambda_0)$ is known and we wish to solve the lasso at λ, the "EDPP" rule discards the gth group if

$$\left\|X_g^\top\left(\frac{y - X\beta^\star(\lambda_0)}{\lambda_0} + \frac{1}{2}v_2^\perp(\lambda, \lambda_0)\right)\right\|_2 < \omega_g - \frac{1}{2}\left\|v_2^\perp(\lambda, \lambda_0)\right\|_2 \|X_g\|_2$$

where we refer the readers to Wang et al. [2015a] for the definitions of the above quantities. We only point out that the EDPP rule is similar to the other rules in that it also discards groups based on the gradient quantity $X_g^\top(y - X\beta^\star(\lambda_0))$ with a correction using $v_2^\perp(\lambda, \lambda_0)$. We have not found any extensions of the EDPP rule to the elastic net or general objectives. The authors show that the EDPP rule is comparable to the strong rule for large values of λ and discards much more aggressively than the SAFE rule. However, we note in passing that in our experiments, even the EDPP rule suffers from the same bloating issue as with the SAFE rule for smaller λ values and often bloats more than the strong sets.

The literature around these screening rules is quite rich. For brevity, we do not discuss the entire literature, but only mention other related works [Bonnefoy et al., 2014, Ndiaye et al., 2017, Fan and Lv, 2008, Fan et al., 2009, Kong et al., 2016, Żogała-Siudem and Jaroszewicz, 2023].

Our main contribution is a novel screening rule called the *pivot rule*. The pivot rule is largely inspired by the strong rule and our main objective is to demonstrate that the pivot rule is much more robust than the strong rule. That is, in the cases where the strong rule is accurate, the pivot rule yields similar prediction sets, and, therefore, similar runtimes for the optimizer; on the other hand, when the strong rule is inaccurate, the pivot rule manages to still produce prediction sets with unwavering accuracy. The general principle is to first order the *active scores*, $\omega_g^{-1}\left\|X_g^\top(y - X\beta^\star(\lambda_0))\right\|_2$, at the solution $\beta^\star(\lambda_0)$ across the groups $g = 1, \ldots, G$. Then, we select a subset of the groups with

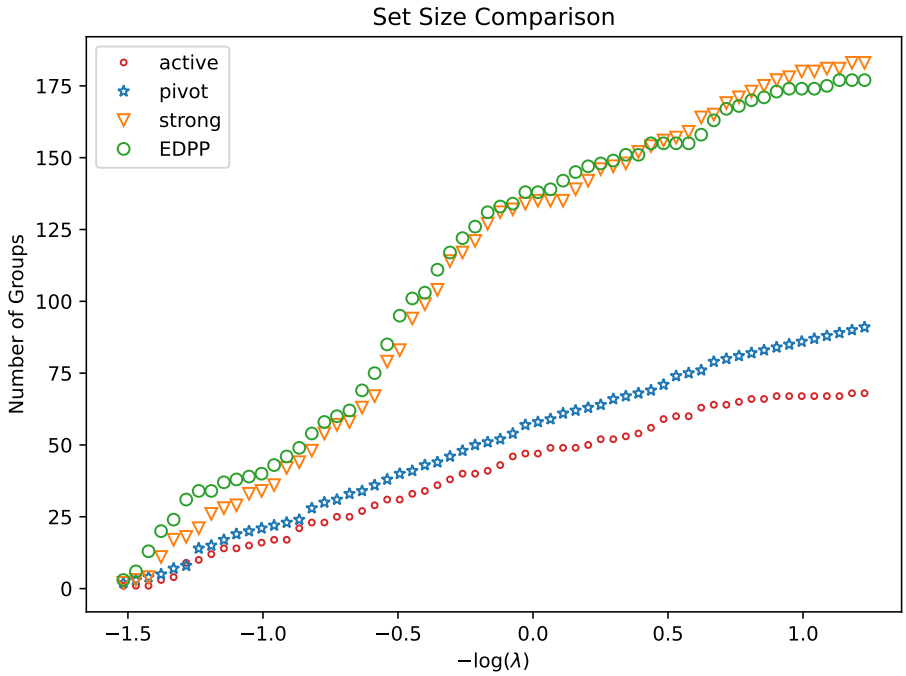


Figure 3.1: A demonstration of the effectiveness of the pivot rule for simulated data. We simulate data as described in Section 3.4.1 and fit group lasso with $n = 100$, $p = 10000$, $G = 2000$, and equicorrelation $\rho = 0.3$ until we reach 99% training R^2. We plot the ever-prediction set sizes for the pivot rule, strong rule, and EDPP rule compared against the ever-active set sizes. The pivot rule yields a prediction set that is extremely tight to the active set compared to the strong rule or the EDPP rule. See Section 3.4.1 for the exact setup of the simulation.

"large enough" scores. This subset is determined by searching for a pivot in a piecewise-linear function, which is where the pivot rule derives its name. In our experiments, we find that it is robust to the correlation structure in X as well as the signal strength in the data. As a quick demonstration, we show in Figure 3.1 the power of our pivot rule. We direct the reader to the figure caption for all technical details of the setup. The figure shows that the pivot rule achieves uniformly smaller ever-prediction sets that lie tight to the ever-active sets with no bloating for this example (for a more in-depth comparison, see Section 3.4). Note that the ever-prediction set is the set of groups that were predicted to be active for some earlier $\lambda > \lambda_0$ in the regularization path. Similarly, ever-active set is the set of groups that were active for some earlier $\lambda > \lambda_0$. Since our primary goal is to compare the pivot rule with the strong rule and `glmnet` implements ever-strong sets, we exclusively focus on ever-prediction sets for all comparisons in this paper.

In Section 3.2, we describe our pivot rule in full detail: Section 3.2.1 begins with our motivation and the methodology, followed by an efficient algorithm to implement the pivot rule in Section 3.2.2; Section 3.2.3 remarks on the hybrid approach of combining the pivot rule with a safe rule; Section 3.2.4 shows a concrete example of when the pivot rule significantly outperforms the strong rule; Section 3.2.5 makes a connection to the LARS algorithm to give further justification for the pivot rule. Section 3.3 shows that the pivot rule can be easily extended to general differentiable, convex losses and the elastic net framework. Finally in Section 3.4, we perform a comparison between the pivot rule, strong rule, and EDPP rule using both simulated and real data. Specifically, we study the difference in the ever-prediction set sizes relative to the ever-active set sizes and their effect on the overall runtime of the optimizer.

3.2 Pivot rule

3.2.1 Motivation and methodology

We begin with the Karesh-Kuhn-Tucker (KKT) conditions for solving the group lasso problem (3.2). According to the KKT conditions, β is a solution for the group lasso problem with λ if and only if

$$\left\|X_g^\top (y - X\beta)\right\|_2 \leq \lambda \omega_g, \quad \forall g = 1, \ldots, G \tag{3.4}$$

Moreover, if $\left\|X_g^\top (y - X\beta)\right\|_2 < \lambda\omega_g$, then β_g must be identically zero. For each group $g = 1, \ldots, G$, define the *active score*

$$s_g(\beta) := \begin{cases} \omega_g^{-1} \left\|X_g^\top (y - X\beta)\right\|_2, & \omega_g > 0 \\ \lambda, & \omega_g = 0 \end{cases} \tag{3.5}$$

and let $s(\beta) := (s_1(\beta), \ldots, s_G(\beta)) \in \mathbb{R}_+^G$ be the concatenation of the active scores. Note that if $\omega_g = 0$, then group g must be active, which gives justification for setting $s_g(\beta)$ to be λ.

The score $s_g(\beta^\star(\lambda))$ can be thought of as the likelihood of group g to be next active, where a larger score indicates higher likelihood. The intuition is the following: suppose we know the solution $\beta^\star(\lambda_0)$ and the corresponding active scores $s(\beta^\star(\lambda_0))$. Then for a sufficiently close $\lambda < \lambda_0$, we expect $s(\beta^\star(\lambda))$ to be close to $s(\beta^\star(\lambda_0))$ simply by continuity of $\lambda \mapsto s(\beta^\star(\lambda))$. Note also that $s_g(\beta^\star(\lambda))$ attains the largest value whenever the group g is active, which follows from the KKT conditions. Combining the two ideas, if $s_g(\beta^\star(\lambda_0))$ is close to the largest value (i.e. λ_0), we should expect $s_g(\beta^\star(\lambda))$ to be close to λ. Hence, the ordering of $s_g(\beta^\star(\lambda_0))$ should be highly predictive of the next active groups at λ.

Indeed, our intuition helps explain the success of the (sequential) strong rule proposed by Tibshirani et al. [2012]. Note that the strong rule determines the next likely active groups at λ by considering all groups g that satisfy

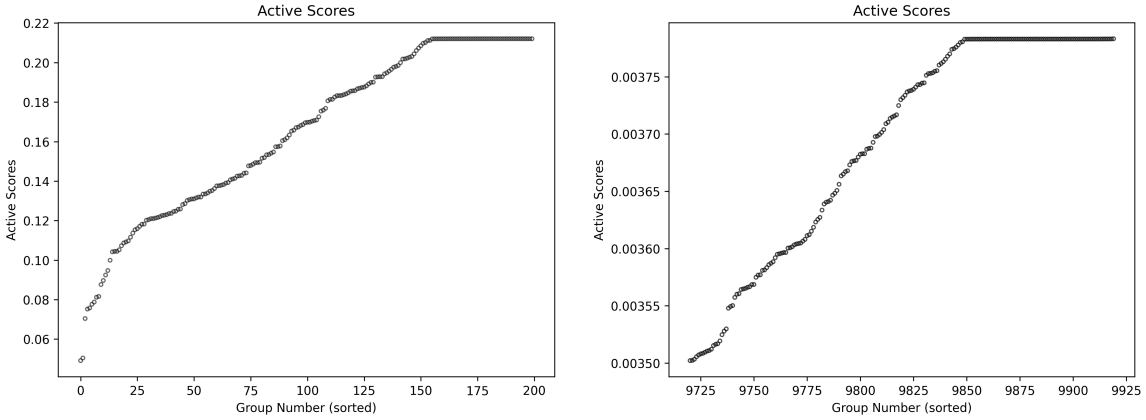
$$s_g(\beta^\star(\lambda_0)) \geq 2\lambda - \lambda_0 \tag{3.6}$$

where $\lambda_0 > \lambda$. That is, we consider the M largest active scores to be included in the next prediction set where M is determined by the cutoff $2\lambda - \lambda_0$. The authors justify this choice of cutoff by assuming the case when the active score is Lipschitz continuous in λ with Lipschitz constant 1, in which case, the strong rule guarantees to discard bona-fide non-active groups. In any case, the intuition for the strong rule is similar to ours in that the goal is to determine groups with "sufficiently large" active scores as a guess for the newly active groups at λ. Naturally, this raises the following question: how do we construct a cutoff that is more aggressive than the strong rule cutoff while not inadvertently discarding the true active groups at λ?

Before presenting our method, we first motivate the discussion with a typical plot of the active scores in Figure 3.2. In Figure 3.2a, we generate $X \in \mathbb{R}^{n \times p}$ from a standard normal distribution with $n = 100$, $p = 1000$, and group sizes of 5. The response vector y is generated from a linear model $y = X\beta + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ such that the signal-to-noise ratio (SNR) is 1 where SNR is defined to be

$$\text{SNR} := \frac{\beta^\top \operatorname{Var}(X)\, \beta}{\sigma^2} \tag{3.7}$$

In Figure 3.2b, we use the Arcene dataset in Section 3.4.2. Under both cases, we plot the active scores $s_g(\beta^\star(\lambda))$ in sorted order. We observe the following pattern in the active scores: the active groups all have a score of λ, which yield the flat component on the right, and the next largest scores decay (approximately) linearly as we read from right to left. Indeed, we observed this pattern for every example we studied. This motivates us to first identify the *pivot* at which the active score first decays significantly from the largest value. Due to numerical precision as well as the inevitable



(a) Simulated data

(b) Real data

Figure 3.2: A typical curve of the active scores (sorted) at a solution $\beta^\star(\lambda)$. Figure 3.2a is using simulated data as described in Section 3.4.1. Figure 3.2b is using real data (Arcene) as described in Section 3.4.2 (zoomed in on the largest active scores to visualize the pivot). For both figures, the active groups all have a score of λ, which yields the flat component on the right. The next largest scores decay (approximately) linearly as we read to the left.

convergence error of the optimizer, the active groups may have scores that slightly deviate from the true value λ, so it is not sufficient to simply look for the first time a decay occurs.

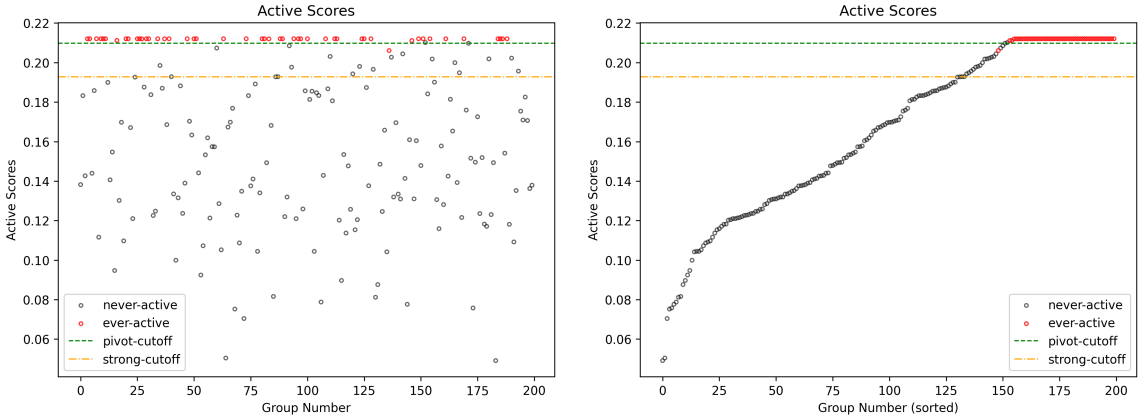
We now present our method to identify the next active groups. We view our current problem as simply an application of linear regression. Suppose we are able to first isolate the K largest active scores such that the pivot is contained in this set of points. With a slight abuse of notation, let $s \in \mathbb{R}_+^K$ denote the active scores for these K points and $t \in \{1, \dots, K\}$ be the indices of s. For each pivot candidate t_j for $j = 2, \dots, K$[1], we model

$$s_i = \beta_0 + \beta_1 (t_j - t_i) \mathbb{1}_{t_i \le t_j} + \text{noise}_i, \quad i = 1, \dots, K \tag{3.8}$$

and perform linear regression to obtain the mean-squared error (MSE), L_j. We then select j that minimizes $\{L_j\}_{j=2}^K$ as the pivot point and include all groups that have active scores higher than that of the pivot point. Since the pivot point is the point at which the active scores *begin* to decay, we apply some slack to include groups whose active scores are slightly smaller than that of the pivot point. In practice, we determine this slack dynamically. If Δ denotes the number of new active groups most recently added, the slack is some proportion γ of $\max(\Delta, 1)$. We typically take $\gamma \in [1, 2]$ and $\gamma = 1.25$ is an effective default value.

The previous discussion crucially relies on the knowledge of K such that the set of K points contains the pivot point. Of course, we may set $K \equiv G$, i.e. the entire set of active scores, however, as shown in Figure 3.2, the decay in the active score is only *locally linear* near the pivot point.

[1]We start from $j = 2$ since we need at least 2 points in each linear piece.



(a) Unsorted active scores.

(b) Sorted active scores.

Figure 3.3: Plot of the active scores with the pivot rule and strong rule cutoffs. The black circles denote the groups that were never active, and the red circles denote the groups that are currently active as well as those that became active at the next λ. The green horizontal line denotes the active score cutoff given by the pivot rule such that all groups whose active scores are above the line are included in the prediction set. The orange horizontal line denotes the active score cutoff given by the strong rule such that all groups whoe active scores are above the line are included in the prediction set.

Indeed, our experiments show that if $K \equiv G$, then the pivot point estimate can be quite poor. As a heuristic, we determine K sequentially. At λ_i, if the pivot rule yields the prediction set $\mathcal{S}_i$, we simply set K to be $(1+\epsilon)\,|\mathcal{S}_i|$ for some $\epsilon \in (0,1)$. Intuitively, we expect that the next set of active groups are contained in the current predicted set $\mathcal{S}_i$ and some number of new groups proportional to the size of $\mathcal{S}_i$. If the next active groups are fully within $\mathcal{S}_{i+1}$, then the pivot must surely exist in $\mathcal{S}_{i+1}$ as well. In practice, $\epsilon = 0.1$ works well in many examples.

Figure 3.3 demonstrates the pivot rule for the same example as in Figure 3.2. The black circles denote the groups that were never active, and the red circles denote the groups that were ever active as well as those that became active at the next λ. Hence, some groups are colored red even if their active scores are not maximal since they were either active previously or are right about to become active. The green horizontal line denotes the active score cutoff given by the pivot rule such that all groups whose active scores are above the line are included in the prediction set. Note that the pivot rule discards many groups and is quite tight to the ever-active set. Moreover, despite the tightness, we do not see any mistakes (the red circle below the green line was previously active). Figure 3.3b clearly shows the effectiveness of the pivot search as it correctly identifies the pivot.

The pivot rule is only a heuristic like the strong rule, so a fallback method must be applied in the event that the rule fails to capture the next active set. We use the same method as in the strong rule. If the prediction set $\mathcal{S}(\lambda)$ fails to capture the next active set $\mathcal{A}(\lambda)$, i.e. a KKT violation occurs for some group outside $\mathcal{S}(\lambda)$, then we include all groups that failed the KKT condition into $\mathcal{S}(\lambda)$

and refit at λ. This process is repeatedly done until the KKT condition passes for all groups outside $\mathcal{S}(\lambda)$. This method guarantees that the prediction set will eventually contain the next active set.

3.2.2 Pivot search algorithm

In Section 3.2, we introduced our pivot search method by repeatedly applying linear regression in (3.8). A priori, it seems we must run $K-1$ separate linear regressions to obtain our MSEs $\{L_j\}_{j=2}^K$. This naive method has a computation complexity of $O(K^2)$. However in this section, we present a simple sequential algorithm where we can compute $\{L_j\}_{j=2}^K$ in $O(K)$ time and $O(1)$ extra space. This demonstrates that the power of our method comes with negligible cost compared to the other parts of the group lasso solver.

We discuss the pivot search in a more general context and briefly drop the notation used in the previous sections. For the remainder of this section, let $x \in \mathbb{R}^n$ be a position vector (sorted) and $y \in \mathbb{R}^n$ be a response vector where

$$y_i = \beta_0 + \beta_1(p - x_i)\mathbb{1}_{x_i \le p} + \text{noise}_i, \quad i = 1, \dots, n \tag{3.9}$$

for some unknown coefficients $\beta_0, \beta_1 \in \mathbb{R}$ and an unknown pivot $p \in \mathbb{R}$. We approximate the pivot to be one of the positions x_j for $j = 2, \dots, n$, and only run linear regression for each of the choices x_j. In this case, we show in Appendix B.1 that there are only four quantities, each represented as a cumulative sum, that determine the full set of MSEs $\{L_j\}_{j=2}^n$ as we iterate over the pivot x_j. Moreover, it requires $O(1)$ time and (extra) space to update the cumulative sums and compute the MSE at each iteration. This simplification is due to the sorted nature of x. We present the algorithm in Algorithm 5.

3.2.3 Pivot-safe hybrid rule

We make a small remark that similar to the work of Zeng et al. [2021], we may combine the pivot rule with a safe rule. Specifically, we may first discard groups via any safe rule such as in Ghaoui et al. [2011], Wang et al. [2015a], then further discard groups using the pivot rule. This in general yields a smaller prediction set. However, in our experience, the use of the safe rule only adds extra non-negligible computation cost and does not differ much from the prediction sets given by the pivot rule on its own. Oftentimes, the strong rule combined with the EDPP rule performs slightly worse than the strong rule on its own as well. For this reason, we do not consider the hybrid approach.

3.2.4 Robustness and performance tradeoff

There is a natural tradeoff between creating a tighter prediction set to the active set and the performance of the optimizer. With smaller prediction sets, we increase the chance of discarding truly

Algorithm 5: Pivot Search

Data: $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$, $n \geq 2$

$\overline{y} = n^{-1} \sum_{i=1}^{n} y_i$

/* Initialize cumulative sums */

$q_1 = x_1$ // Cumulative sum of x: $\sum_{i=1}^{j} x_i$

$q_2 = x_1^2$ // Cumulative sum of x^2: $\sum_{i=1}^{j} x_i^2$

$q_3 = y_1$ // Cumulative sum of y: $\sum_{i=1}^{j} y_i$

$q_4 = y_1 x_1$ // Cumulative sum of yx: $\sum_{i=1}^{j} y_i x_i$

for $j = 2, \ldots, n$ **do**

  /* Update cumulative sums */

  $q_1 = q_1 + x_j$

  $q_2 = q_2 + x_j^2$

  $q_3 = q_3 + y_j$

  $q_4 = q_4 + y_j x_j$

  /* Compute linear regression quantities */

  $\overline{t} = n^{-1}(j x_j - q_1)$

  $v_t = j x_j^2 - 2 x_j q_1 + q_2 - n\overline{t}^2$

  $c_{ty} = x_j(q_3 - j\overline{y}) - (q_4 - \overline{y} q_1)$

  $\hat{\beta}_1 = c_{ty} / v_t$

  $L_j = -\hat{\beta}_1^2 v_t$

return $\operatorname{argmin}_{2 \leq j \leq n} L_j$;

active groups, which may result in frequent refitting and therefore more compute time. However, with larger prediction sets, we increase the overall compute time simply because we must optimize over a larger set of groups. Not only does that increase the complexity of the algorithm, but it may also affect the convergence speed, especially if largely correlated features are included in the prediction set. Generally, we have found that the pivot rule using our default configurations in Section 3.2.1 creates smaller prediction sets than the strong rule. In most cases, the mild bloating cost for the strong rule has a similar impact on the optimization runtime as the refit cost for the pivot rule (see Table 3.1 as an example). However, there are cases when the pivot rule absolutely shines.

In Figure 3.4, we show an example of the active scores at $\lambda_{\max}$, the smallest λ such that the solution is $\beta^\star(\lambda_{\max}) = \mathbf{0}$, for a simulated dataset described in Section 3.4.1 with equicorrelation $\rho = 0.5$ in the feature matrix X. This example highlights cases when the active scores are so close to each other that the strong rule must bring in a large set of groups into its prediction set because it uses a *fixed* threshold that is *data-independent*. This phenomenon occurs when there are correlated features because stronger feature correlation implies that the correlation between the features and the residual are less distinguishable; in turn, the active scores are less distinguishable as well (see (3.5)). However, the pivot rule is completely unaffected by this phenomenon. This is due to the fact that the pivot rule is much more adaptive to the structure of the active scores; the

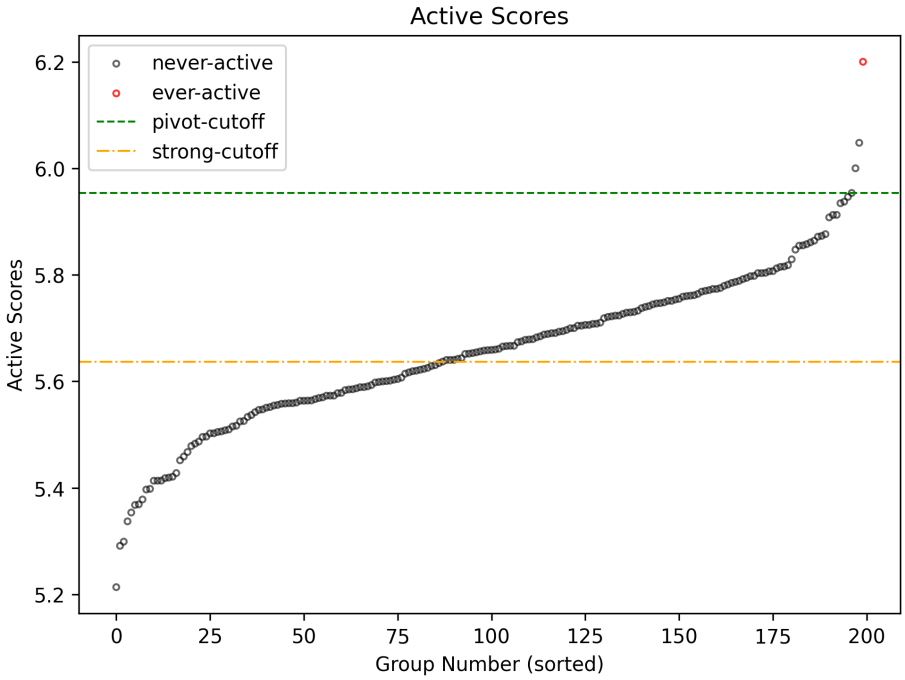


Figure 3.4: A plot of the active scores at $\lambda_{\max}$ for a simulated dataset described in Section 3.4.1 with equicorrelation $\rho = 0.5$ in the feature matrix X. This example shows that the strong rule can be too conservative when the active scores are not sufficiently spread out. However, the pivot rule still manages to construct reliably small prediction sets.

pivot rule only relies on the general pattern of the scores no matter the scale or the relative closeness of the scores. Especially in these cases where the active scores are relatively close, the pivot rule benefits even more from its robustness as the number of groups increases, since the strong rule will suffer from pulling more groups into its prediction sets.

3.2.5 Connection to LARS

The LARS algorithm [Efron et al., 2004] sheds more light on the effectiveness of the pivot rule. To parallel the exposition in Efron et al. [2004], we consider the lasso problem, i.e. (3.2) with $G \equiv p$ and penalty factors $\omega_g \equiv 1$. At a high level, the LARS procedure begins with an empty predictor set with all coefficients equal to zero, and finds the predictor x_{j_1} most correlated with the current residual. Along x_{j_1}, we take the largest step possible until another predictor x_{j_2} has equicorrelation with the residual as x_{j_1}. In this way, LARS continues to move along a direction that is equiangular with the current active predictors until a new predictor becomes equiangular.

Formally, let $\mathcal{A}(\lambda_0)$ be the current active set and $\hat{c}_g := X_g^\top(y - X\beta^\star(\lambda_0))$ the current gradient. Note that $\hat{c}_g$ has the interpretation of a correlation with the current residual and the predictor X_g (assuming they are both centered and standardized). Finally, let $u_{\mathcal{A}(\lambda_0)}$ denote the unit direction that is equiangular with every predictor in $\mathcal{A}(\lambda_0)$. Suppose $\lambda < \lambda_0$ is the largest λ such that a new predictor becomes active. Then, the LARS algorithm takes a step such that

$$X\beta^\star(\lambda) = X\beta^\star(\lambda_0) + \hat{\gamma} u_{\mathcal{A}}$$

where

$$\begin{aligned}
\hat{\gamma} &= \min_{g \in \mathcal{A}(\lambda_0)^c} \left\{ \frac{\lambda_0 - \hat{c}_g}{A_{\mathcal{A}(\lambda_0)} - a_g}, \frac{\lambda_0 + \hat{c}_g}{A_{\mathcal{A}(\lambda_0)} + a_g} \right\} \\
a_g &= X_g^\top u_{\mathcal{A}(\lambda_0)} \\
A_{\mathcal{A}(\lambda_0)} &= \left(\mathbf{1}_{\mathcal{A}(\lambda_0)}^\top \left(X_{\mathcal{A}(\lambda_0)}^\top X_{\mathcal{A}(\lambda_0)} \right)^{-1} \mathbf{1}_{\mathcal{A}(\lambda_0)}^\top \right)^{-\frac{1}{2}}
\end{aligned}$$

and the minimum in $\hat{\gamma}$ is taken only over positive components for each choice g.

The connection between LARS and the pivot rule lies in the definition of the step size $\hat{\gamma}$. Ignoring the denominator for the moment, we immediately see that $\hat{\gamma}$ is achieved by a group $g \notin \mathcal{A}(\lambda_0)$ when $\lambda_0 \pm \hat{c}_g$ is smallest (and positive), in other words, when the active score $s_g \equiv |\hat{c}_g|$ is closest to λ_0. Hence, now considering the denominator as well, we expect that $\hat{\gamma}$ is achieved by a group g that has, perhaps not the largest, but *one of the largest* active scores. The pivot rule is simply one mechanism to determine the set of these "largest active scores" that is then most likely to contain the next active groups.

3.3 Generalized pivot rule

3.3.1 Extension to general loss functions

While the primary focus of the paper is on the group lasso problem with squared-loss objective (3.2), our method naturally extends to general convex, differentiable loss functions such as in logistic regression. We begin by extending the definition of the active scores in (3.5), which arose from studying the KKT conditions. For the general problem (3.1), the KKT conditions show that β is a solution if for every $g = 1, \dots, G$, there exists u_g a subgradient of $\beta_g \mapsto \|\beta_g\|_2$ such that

$$\nabla_{\beta_g} f(\beta) + \lambda \omega_g u_g = 0 \iff \left\|\nabla_{\beta_g} f(\beta)\right\|_2 \leq \lambda \omega_g$$

Hence, we simply define the active scores to be

$$s_g(\beta) := \begin{cases} \omega_g^{-1} \left\|\nabla_{\beta_g} f(\beta)\right\|_2, & \omega_g > 0 \\ \lambda, & \omega_g = 0 \end{cases}$$

Once the active scores are defined, we may reuse the rest of the method laid out in Section 3.2.

As a concrete example, in the logistic regression case where

$$f(\beta) = \sum_{i=1}^{n} \left(-y_i x_i^\top \beta + \log\left(1 + e^{x_i^\top \beta}\right)\right)$$

the active scores are

$$s_g(\beta) = \begin{cases} \omega_g^{-1} \left\|X_g^\top (y - p(\beta))\right\|_2, & \omega_g > 0 \\ \lambda, & \omega_g = 0 \end{cases}$$

where $p_i(\beta) := \left(1 + e^{-x_i^\top \beta}\right)^{-1}$. We then find the pivot in these scores (sorted) and include all groups with scores higher than that of the pivot and a slack of groups with scores just below that of the pivot.

3.3.2 Extension to elastic net

The elastic net [Zou and Hastie, 2005] is given by

$$\underset{\beta}{\text{minimize}} \ \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \sum_{j=1}^{p} \left(\alpha |\beta_j| + \frac{1-\alpha}{2} \beta_j^2 \right)$$

Extending to the group lasso form with a general convex, differentiable loss f, we have

$$\underset{\beta}{\text{minimize}} \; f(\beta) + \lambda \sum_{g=1}^{G} \omega_g \left(\alpha \left\| \beta_g \right\|_2 + \frac{1-\alpha}{2} \left\| \beta_g \right\|_2^2 \right) \tag{3.10}$$

As in Section 3.3.1, we only discuss how to define active scores since the rest of our method follows immediately. Following the KKT conditions, β is a solution to (3.10) if for every $g = 1, \dots, G$, there exists u_g a subgradient of $\beta_g \mapsto \left\| \beta_g \right\|_2$ such that

$$\nabla_{\beta_g} f(\beta) + \lambda \omega_g \left(\alpha u_g + (1-\alpha)\beta_g \right) = 0 \iff \left\| \nabla_{\beta_g} f(\beta) + \lambda \omega_g (1-\alpha)\beta_g \right\|_2 \leq \lambda \omega_g \alpha$$

Hence, we define the active scores

$$s_g(\beta) = \begin{cases} (\omega_g \alpha)^{-1} \left\| \nabla_{\beta_g} f(\beta) + \lambda \omega_g (1-\alpha)\beta_g \right\|_2, & \omega_g \alpha > 0 \\ \lambda, & \omega_g \alpha = 0 \end{cases}$$

In particular, if $\alpha = 0$ (ridge regression) so that no sparsity is induced, every active score is identical to λ since every group will be active. Otherwise when $\alpha > 0$, we get the usual form,

$$s_g(\beta) = \begin{cases} (\omega_g \alpha)^{-1} \left\| \nabla_{\beta_g} f(\beta) + \lambda \omega_g (1-\alpha)\beta_g \right\|_2, & \omega_g > 0 \\ \lambda, & \omega_g = 0 \end{cases}$$

3.4 Comparisons to other methods

In this section, we provide comparisons of our pivot rule against the strong rule and EDPP rule. We use our Python package `adelie` to solve (group) lasso (available on PyPI[2]), which implements the screening rules we consider in this paper. The experiments were done on a local machine with AMD Ryzen 7 5800X 8-core processor and 128GB of RAM, which was sufficient for the scale of our data.

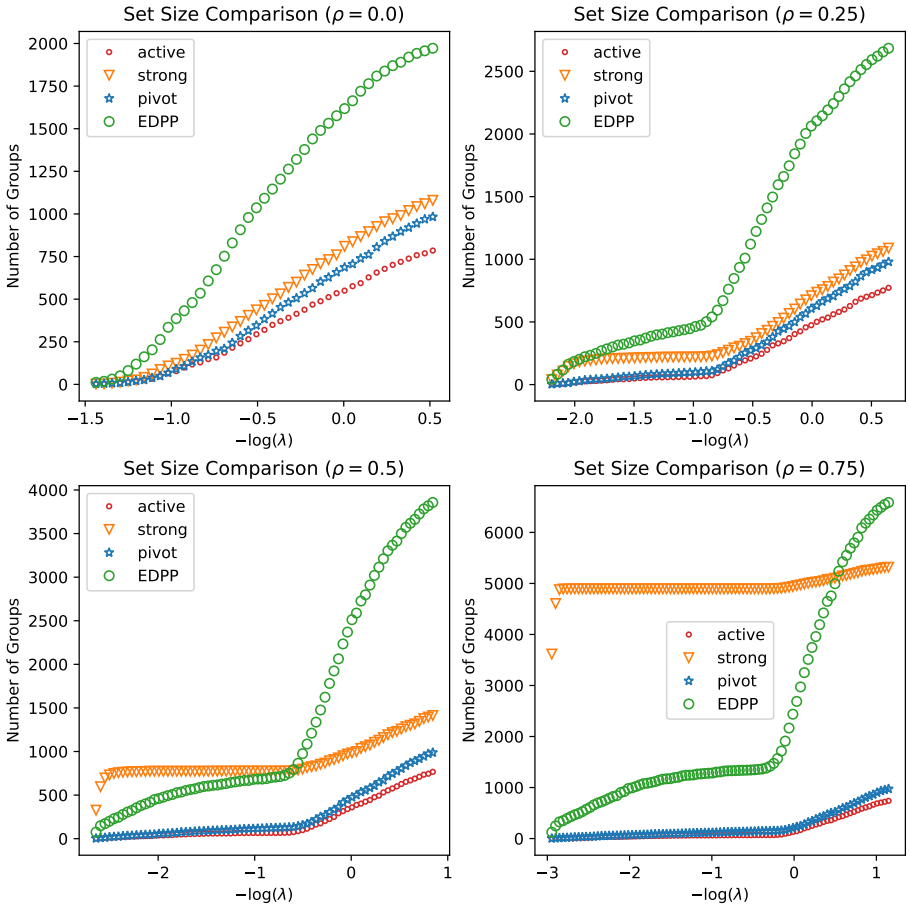


Figure 3.5: Plot of the set size comparison across the active set, strong set, pivot set, and EDPP set where $n = 1000$, $p = 10000$, and $G = 10000$ with group sizes of 1 (lasso).

3.4.1 Simulated data

We first consider the following simulation setting. Let $X \in \mathbb{R}^{n \times p}$ be the feature matrix where each row $X_{i\cdot}$ is sampled from $\mathcal{N}(0, \Sigma)$ where $\Sigma \in \mathbb{R}^{p \times p}$ is given by the equicorrelation matrix

$$\Sigma = \begin{bmatrix} 1 & \rho & \dots & \rho \\ \rho & 1 & \dots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \dots & 1 \end{bmatrix}$$

[2]https://pypi.org/project/adelie/1.1.4/

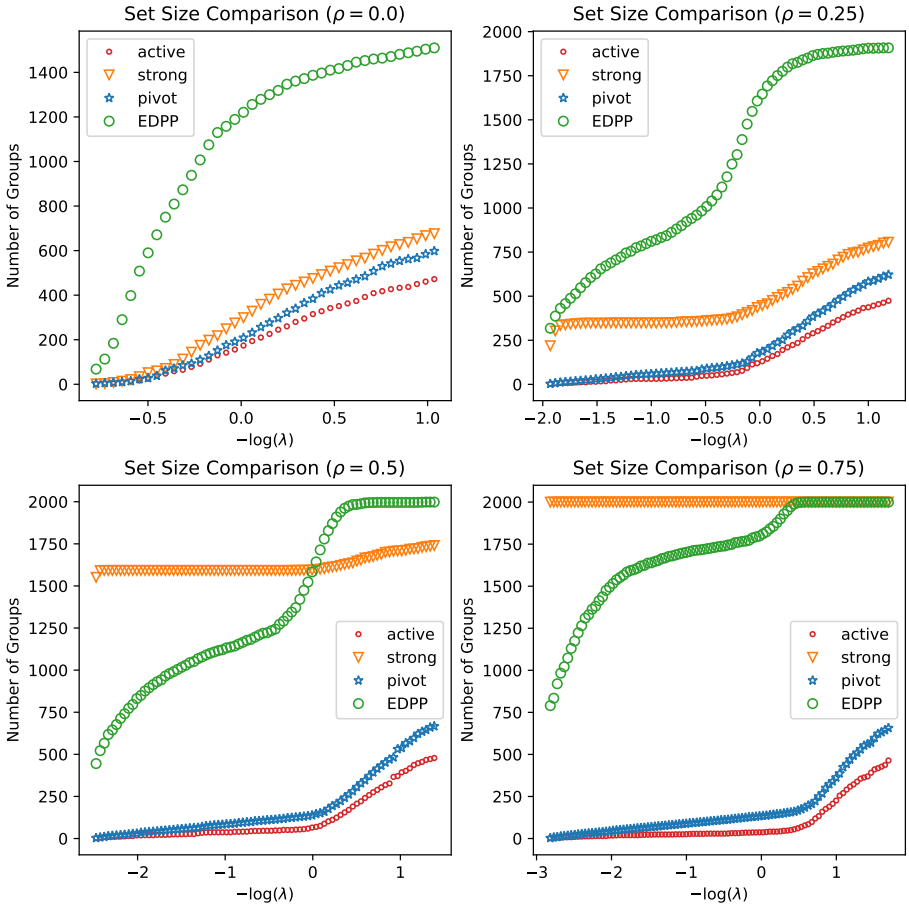


Figure 3.6: Plot of the set size comparison across the active set, strong set, pivot set, and EDPP (safe) set where $n = 1000$, $p = 10000$, and $G = 2000$ with group sizes of 5 (group lasso).

with equicorrelation $\rho \in [0, 1]$. Note that when $\rho = 0$, we have independent columns of X. We generate our response $y \in \mathbb{R}^n$ from the linear model

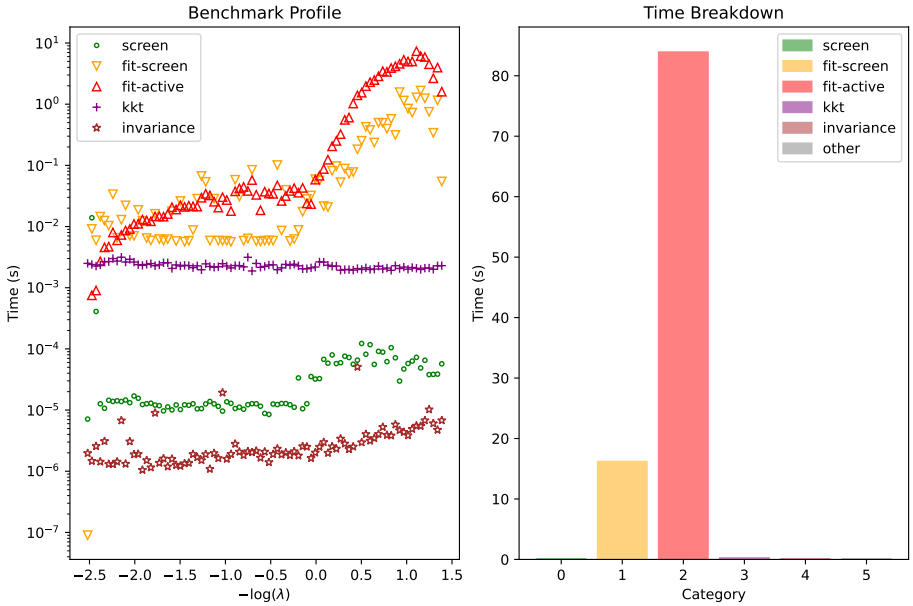
$$y = X\beta + \varepsilon$$

The true coefficient vector $\beta \in \mathbb{R}^p$ is sampled from $\mathcal{N}(0, I_p)$ and 95% of the entries are set to zero. Although we randomly generate β, we work in the "fixed-β" regime. The noise $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ is sampled such that SNR $= 1$ (see (3.7) for the definition). Finally, both X and y are normalized by $\sqrt{n}$.

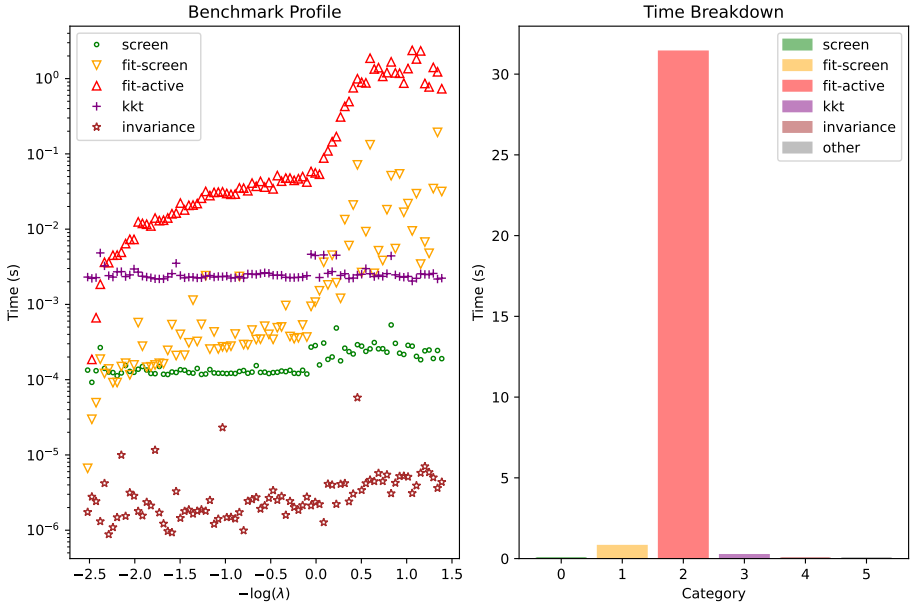
We show in Figure 3.5 a comparison of the ever-prediction set sizes for the strong rule, EDPP rule, and our pivot rule against the ground truth for the ever-active set in the lasso setting. We let $n = 1000$, $p = 10000$, and $G = 10000$ so that every group has size 1. This comparison is done for $\rho \in \{0, 0.25, 0.5, 0.75\}$. For each plot, we run the lasso solver for 100 λ values until 99% of the training R^2 is reached. The overall trend is the same across all examples. First, EDPP rule is generally the most conservative rule, which is unsurprising as it must theoretically guarantee that the discarded variables must be inactive. However, it performs better than the strong rule for large values of λ especially as ρ increases, which is consistent with the findings in Wang et al. [2015a]. Next, the strong rule generally discards more than the EDPP rule, however, it quickly becomes less effective as ρ increases due to an initial bloating. Finally, our pivot rule clearly outperforms the other methods as it stays extremely close to the ever-active set and completely robust to changes in ρ and λ. The number of KKT failures with the pivot rule were 5 when $\rho = 0$; 2 when $\rho = 0.25$; and 0 otherwise. Hence, for this lasso example, the pivot rule made few mistakes while staying tight to the ever-active set.

We provide an example of the group lasso in Figure 3.6. We let $n = 1000$, $p = 10000$, and $G = 2000$ so that each group has size 5. Again, the comparison is done for $\rho \in \{0, 0.25, 0.5, 0.75\}$. For each plot, we run the group lasso solver for 100 λ values until 99% of the training R^2 is reached. Similar to Figure 3.5, the strong rule is quite accurate when $\rho = 0$, however it behaves more erratically as ρ increases. In particular, the strong rule pulls in too many groups early on. The number of KKT failures with the pivot rule were 1 when $\rho = 0$; 2 when $\rho = 0.25$; and 0 otherwise. Hence, as in the lasso example, the pivot rule is extremely accurate for group lasso as well.

We end our discussion with some performance analysis. Due to the early bloating of the strong rule, the authors of Tibshirani et al. [2012] keep track of both an ever-strong set and an ever-active set in their `R` package, `glmnet`, where most of the optimization is done on the ever-active set and the optimization over the ever-strong set is done relatively infrequently. However, when the ever-strong set contains a significant portion of the groups early on, the computational cost significantly increases and also degrades the convergence speed. In the simulation setup of Figure 3.6 with $\rho = 0.5$, the strong rule yielded a total solving time of 88.6 seconds whereas the pivot rule yielded 28.1 seconds, which is about a 3x speed-up using the pivot rule. In Figure 3.7, we show a more detailed benchmark



(a) Strong rule



(b) Pivot rule

Figure 3.7: Benchmark results of solving group lasso in the setting of Figure 3.6 using the strong rule and the pivot rule. We note that while most of the time is spent on iterating over the active set, the iteration over the prediction set consumes a considerable amount of time. The pivot rule clearly shows a significant decrease in the proportion of time spent on iterating over the prediction set.

Dataset	n	p	G
arcene	100	9920	9920
arcene_gl	100	79360	9920
dorothea	800	88119	88119
dorothea_gl	800	704952	88119
gisette	6000	4955	4955
gisette_gl	6000	39640	4955
mnist	784	59999	59999
mnist_gl	784	479992	59999
electricity	370	140255	140255
electricity_gl	370	1122040	140255
gene	801	20264	20264
gene_gl	801	162112	20264

Table 3.1: The number of observations, features, and groups used in each dataset. All dataset names of the form "xxx_gl" refer to using the dataset "xxx" with 5 B-spline basis for each feature as groups and fitting with group lasso. Otherwise, the dataset "xxx" is used directly to fit lasso.

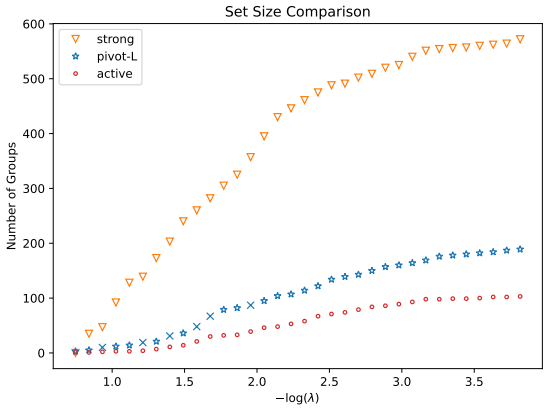
result. We note that while most of the time is spent on iterating over the active set, the iteration over the prediction set consumes a considerable amount of time. The pivot rule clearly shows a significant decrease in the proportion of time spent on iterating over the prediction set. Interestingly, the pivot rule seems to slightly decrease the active set iteration time as well. This provides some evidence to our earlier hypothesis that larger prediction set yields longer convergence time.

3.4.2 Real data

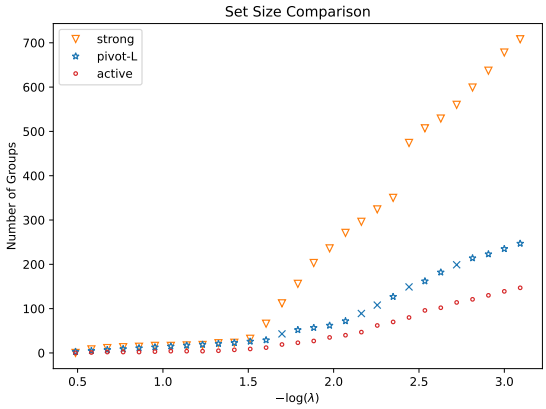
For our real data analysis, we consider the datasets listed in Table 3.1 from the University of California, Irvine, machine learning repository[3]. For each dataset, we standardize the features and the response vector to have unit variance. Moreover, we further divide X and y by $\sqrt{n}$ to account for the scaling in the number of observations. For the group lasso, we create 5 B-Spline bases for each feature and treat them as a group (so that each group has size 5). We fit a path of 100 λ values until either the training R^2 reaches 90% or all 100 λ values are processed. For some examples, we increase the R^2 tolerance to 99% for better visualization. The pivot rule is applied using the default configuration specified in Section 3.2.1.

In Figure 3.8, we show a comparison of ever-prediction set sizes using the strong rule and the pivot rule against the ever-active set sizes for each λ for fitting lasso. Figure 3.9 shows a similar comparison for fitting group lasso. For each rule, we mark with an "x" to denote a KKT failure. In all cases, the pivot rule yields similar or smaller prediction set sizes at the cost of a few more KKT failures. We remind the readers that the user has the flexibility to be less aggressive than the default configuration of the pivot rule by specifying a larger slack proportion parameter, for example. In

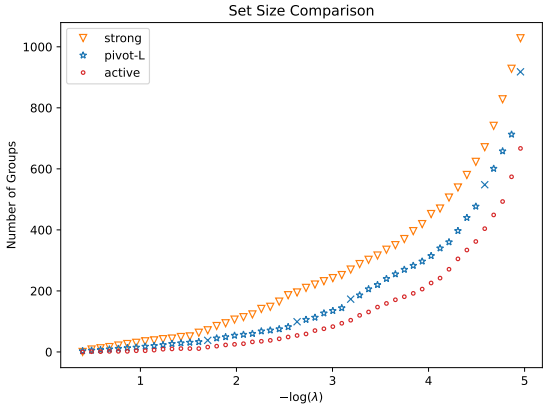
[3]https://archive.ics.uci.edu/



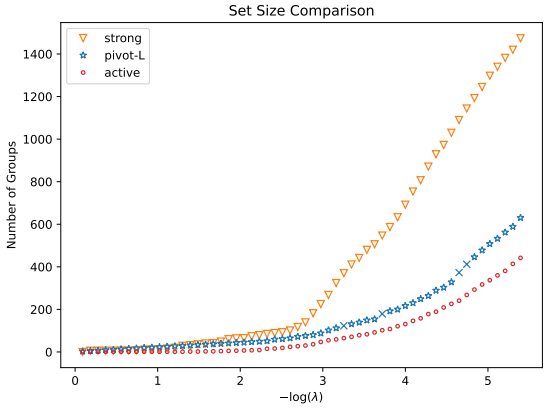
(a) Arcene



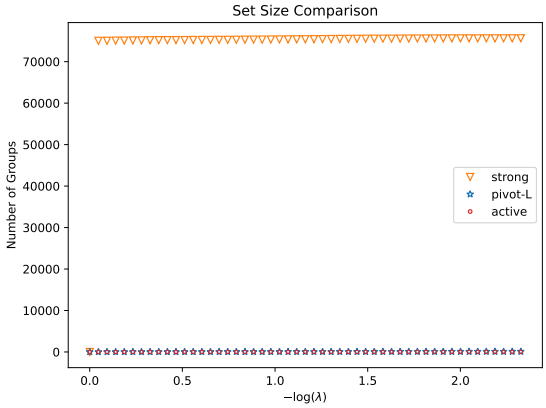
(b) Dorothea



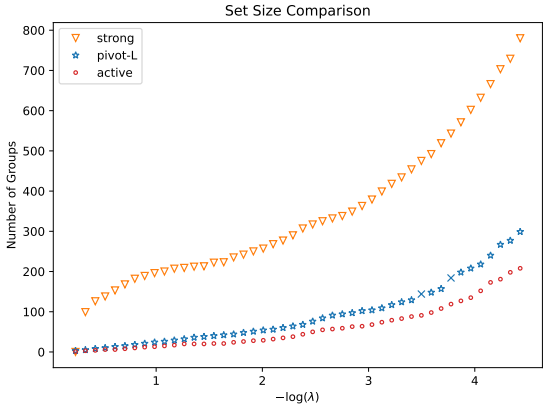
(c) Gisette



(d) MNIST

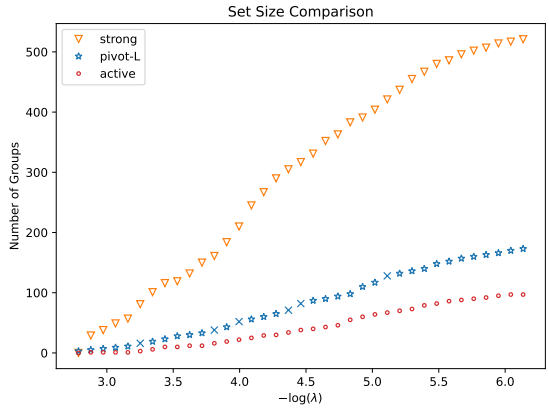


(e) Electricity

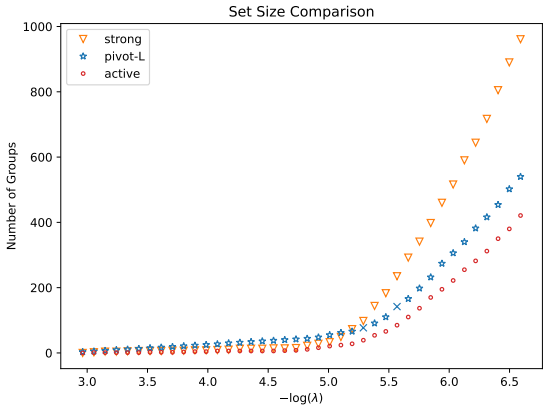


(f) Gene

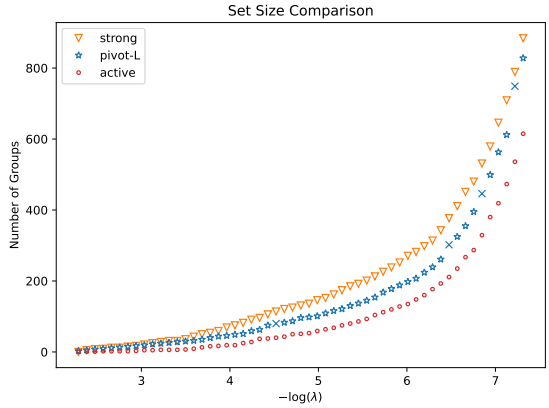
Figure 3.8: Plot of prediction set sizes using the strong rule and the pivot rule against the true active set sizes for fitting lasso on real datasets. Overall, the pivot rule yields smaller prediction set sizes but with more violations. However, it is much more robust to the cases of stronger correlation in X as shown in Figure 3.8e.



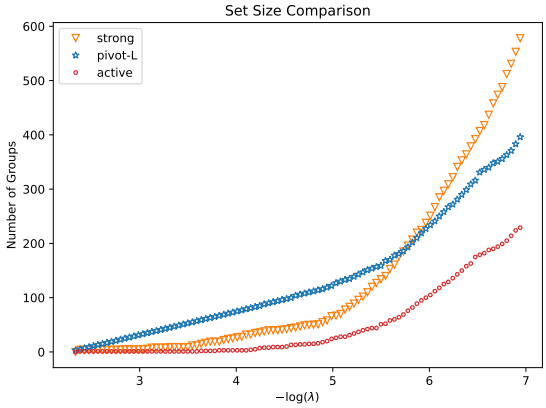
(a) Arcene



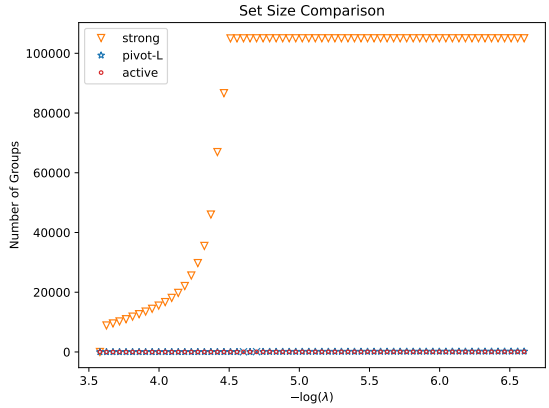
(b) Dorothea



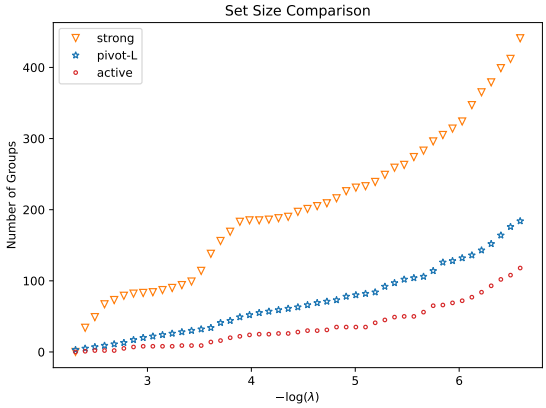
(c) Gisette



(d) MNIST



(e) Electricity



(f) Gene

Figure 3.9: Plot of prediction set sizes for using the strong rule and the pivot rule against the true active set sizes for fitting group lasso on real datasets with 5 B-spline bases for each feature. Overall, the pivot rule yields smaller prediction set sizes but with more violations. However, it is much more robust to the cases of stronger correlation in X as shown in Figure 3.8e.

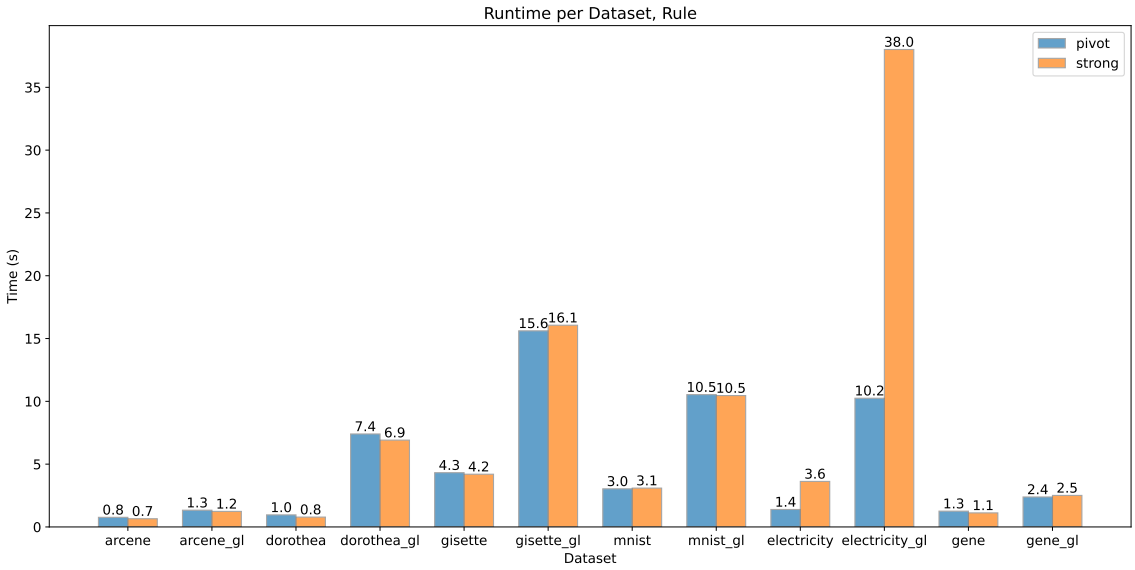
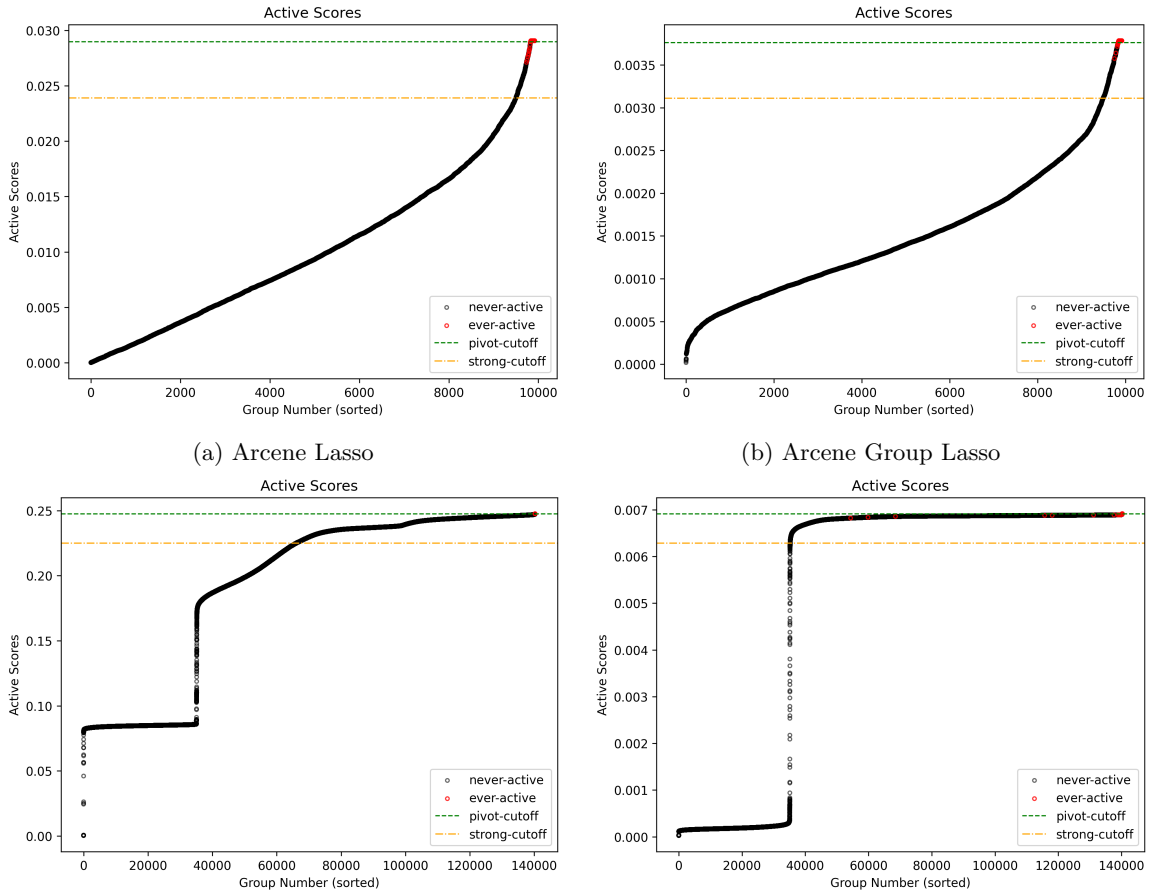


Figure 3.10: Runtime comparison of the optimizer using the strong rule vs. the pivot rule. In almost all cases, the runtime difference is marginal and does not seem to indicate any meaningful differences between the two methods. However, the Electricity dataset shows that under more feature correlation, the strong rule suffers from being too conservative.

any case, this cost seems negligible based on the overall runtime comparison of our solver using the two rules as shown in Figure 3.10. In almost all cases, the runtime difference is marginal and does not seem to indicate any meaningful differences between the two methods. This is not surprising because in these cases the feature matrices were highly uncorrelated, which is the regime where we should expect the strong rule to be most accurate. Hence, since the strong rule is already an effective rule for these cases, we should expect minor, if any, improvements using the pivot rule. Conversely, in the cases where the strong rule performs poorly, the pivot rule remains robust and dramatically improves the runtime. The most stark difference between the strong rule and the pivot rule occurs in Figures 3.8e and 3.9e where the pivot rule is faster by a factor of 3. We note that this dataset contains stronger pairwise correlation compared to the other datasets. Hence, we observe a similar behavior as in Figures 3.5 and 3.6 where the strong rule initially pulls too many groups as the pairwise correlation increases, but the pivot rule is completely robust to these changes and only pulls a small subset of groups.

As further motivation for the pivot method, we show in Figure 3.11 the (sorted) active scores for two of the datasets, Arcene and Electricity, after fitting lasso and group lasso. Arcene is an example of a dataset where the strong rule performed well and Electricity is an example where the strong rule performed quite poorly. In all cases, we see that the active scores indeed have a locally linear behavior near the pivot. Moreover, for Electricity, many scores are clumped together, which makes the strong rule cutoff quite conservative. However, the pivot rule is still able to learn a more



(a) Arcene Lasso

(b) Arcene Group Lasso

(c) Electricity Lasso

(d) Electricity Group Lasso

Figure 3.11: A plot of active scores for Arcene and Electricity datasets for both lasso and group lasso. In all cases, we observe a locally linear trend near the pivot. The Electricity dataset has a much narrower spread of the active scores than in the Arcene dataset so that the strong rule cutoff is much more conservative in the former. However, in all cases, the pivot rule cutoff is aggressive to a similar degree.

aggressive cutoff despite the flatness of the scores.

3.5 Discussion

We propose a new rule called the pivot rule for discarding groups in group lasso problems. The primary motivation for the pivot rule is to discard more predictors in cases where the strong rule performs poorly, while also performing on par with the strong rule in its advantageous settings. We then describe the motivation, methodology, and algorithm to employ the pivot rule. The simplicity of the pivot rule also allows a natural extension to general convex, differentiable loss functions and the elastic net framework. Through simulations and real data analyses, we show that the pivot rule is much more robust to the correlation structure in the feature matrix and the signal strength in the response vector than the strong rule, significantly improving the overall runtime in some cases by several orders of magnitude and otherwise having nearly identical runtimes. We believe that the pivot rule is a much more reliable general-purpose screening method as its behavior remains consistent across various inputs.

Chapter 4

Polyhedron constrained group lasso and elastic net

4.1 Introduction

In this chapter, we discuss the polyhedron constrained group lasso problem, the most technically involved and daunting problem in this thesis. This work is still under development and our results are preliminary, however, the methodology is largely finished.

We consider the group lasso problem with (separable) polyhedron constraints given by

$$\begin{aligned} \underset{\beta_0,\beta\in\mathbb{R}^p}{\text{minimize}} \quad & \frac{1}{2}\left\|y-\beta_0\mathbf{1}-X\beta\right\|_W^2+\lambda P_{\alpha,\omega}(\beta) \\ \text{subject to} \quad & -\ell_g\le A_g\beta_g\le u_g, \quad \forall g=1,\dots,G, \end{aligned} \tag{4.1}$$

where $X\in\mathbb{R}^{n\times p}$ is the feature matrix, $y\in\mathbb{R}^n$ is the response vector, $\beta=(\beta_1,\dots,\beta_G)\in\mathbb{R}^p$ is the coefficient vector with group-wise coefficients $\beta_g\in\mathbb{R}^{p_g}$ for every $g=1,\dots,G$, $\beta_0\in\mathbb{R}$ is the intercept, $W\in\mathbb{R}_+^{n\times n}$ is the diagonal matrix of observation weights, $\lambda\ge 0$ is the group lasso regularization parameter, $P_{\alpha,\omega}(\cdot)$ is the group elastic net penalty

$$P_{\alpha,\omega}(\beta)=\sum_{g=1}^{G}\omega_g\left(\alpha\left\|\beta_g\right\|_2+\frac{1-\alpha}{2}\left\|\beta_g\right\|_2^2\right),$$

and $A_g\in\mathbb{R}^{m_g\times p_g}$ and $\ell_g,u_g\in\mathbb{R}_+^{m_g}$ define the group-wise polyhedron constraints. Note that we assume $\mathbf{0}$ is feasible, or equivalently, $\ell_g,u_g\ge\mathbf{0}$. This is a crucial assumption that preserves sparsity of the solution, which simplifies our algorithm.

We see this constrained problem potentially useful in typical regression settings with box-constraints. For example, a natural extension to isotonic regression is to include a group lasso penalty as soon as there is a natural group structure to the feature matrix and feature selection is desired; isotonic regression is typically recasted as a higher-dimensional regression problem with non-negativity constraints. A more concrete application is in convex 2-layer ReLU neural network training. It is a proven fact that a 2-layer ReLU neural network can be optimally trained by solving a polyhedron constrained group lasso problem (see Section 4.4) [Pilanci and Ergen, 2020, Bach, 2017, Ergen and Pilanci, 2023].

In Section 4.3, we develop our novel algorithm to solve (4.1). We discuss various aspects of the solver such as backtracking, warm-starts, screening, and extensions to general convex losses that make it extremely flexible and powerful. In Section 4.4, we apply our algorithm to train 2-layer ReLU networks using the convex reformulation. Moreover, we show that this can be done stunningly fast, and the resulting neural networks generally predict better than ADAM-trained networks.

4.2 Preliminaries and notations

We denote $\mathbb{R}$ as the space of real numbers and $\mathbb{R}_+$ as the space of non-negative real numbers. Define $\mathbb{R}^n$ to be the n-dimensional vector of real numbers and $\mathbb{R}^{n\times p}$ as the space of all $n \times p$ matrices of real numbers. Similarly, we define $\mathbb{R}^n_+$ and $\mathbb{R}^{n\times p}_+$ to be the space of vectors and matrices of non-negative real numbers. We use $\|x\|_p = (\sum_{i=1}^p |x_i|^p)^{1/p}$ to denote the ℓ_p-norm of a vector. For two vectors $u, v \in \mathbb{R}^n$, we interpret $u \leq v$ as an elementwise comparison. The convex indicator function $\mathbb{1}_A(x)$ is defined to be zero whenever $x \in A$ and otherwise $+\infty$. We use the notation $[n] := \{1, \dots, n\}$ to denote the set of integers between 1 and n. For a vector $v = (v_1, \dots, v_G) \in \mathbb{R}^p$ where each $v_g \in \mathbb{R}^{p_g}$ and a set of indices $S \subseteq [G]$, we denote $v_S := \{v_g : g \in S\}$ and $v_{-S} := \{v_g : g \notin S\}$. Similarly, for a matrix $X = [X_1, \dots, X_G] \in \mathbb{R}^{n\times p}$ where each $X_g \in \mathbb{R}^{n\times p_g}$, we denote $X_S := \{X_g : g \in S\}$ and $X_{-S} := \{X_g : g \notin S\}$.

4.3 Algorithms for constrained group lasso/elastic net under Gaussian loss

We propose a block-coordinate descent based algorithm to solve (4.1). Since the added polyhedron constraints are separable across the penalized groups β_g, block-coordinate descent provably attains the optimum [Tseng, 2001]. It remains to discuss the algorithm for solving a single block update.

Minimizing with respect to a single group β_g, the block update is given by

$$\begin{aligned} \underset{\beta_g \in \mathbb{R}^{p_g}}{\text{minimize}} \quad & \frac{1}{2} \left\| r_g - X_g \beta_g \right\|_W^2 + \lambda \omega_g \left(\alpha \left\| \beta_g \right\|_2 + \frac{1-\alpha}{2} \left\| \beta_g \right\|_2^2 \right) \\ \text{subject to} \quad & -\ell_g \leq A_g \beta_g \leq u_g, \end{aligned}$$

where $r_g := y - X_{-g}\beta_{-g}$ is the current residual from residualizing out all other blocks. Rewriting the objective in the quadratic form, the block update is equivalent to

$$\begin{aligned} \underset{\beta_g \in \mathbb{R}^{p_g}}{\text{minimize}} \quad & \frac{1}{2} \beta_g^\top X_g^\top W X_g \beta_g - r_g^\top W X_g \beta_g + \lambda \omega_g \left(\alpha \left\| \beta_g \right\|_2 + \frac{1-\alpha}{2} \left\| \beta_g \right\|_2^2 \right) \\ \text{subject to} \quad & -\ell_g \leq A_g \beta_g \leq u_g. \end{aligned}$$

As in the unconstrained case (see Section 2.3), it is advantageous to first rotate β_g to the eigenbasis of $X_g^\top W X_g$. In that endeavor, let $X_g^\top W X_g = Q_g \Lambda_g Q_g^\top$ be the eigen-decomposition where $Q_g \in \mathbb{R}^{p_g \times p_g}$ is an orthogonal matrix and $\Lambda_g \in \mathbb{R}_+^{p_g \times p_g}$ is a non-negative diagonal matrix. Then, changing variables to $\tilde{\beta}_g := Q_g^\top \beta_g$, the block update becomes

$$\begin{aligned} \underset{\tilde{\beta}_g \in \mathbb{R}^{p_g}}{\text{minimize}} \quad & \frac{1}{2} \tilde{\beta}_g^\top \Lambda_g \tilde{\beta}_g - \tilde{r}_g^\top \tilde{\beta}_g + \lambda \omega_g \left(\alpha \left\| \tilde{\beta}_g \right\|_2 + \frac{1-\alpha}{2} \left\| \tilde{\beta}_g \right\|_2^2 \right) \\ \text{subject to} \quad & -\ell_g \leq A_g Q_g \tilde{\beta}_g \leq u_g, \end{aligned}$$

where $\tilde{r}_g = Q_g^\top X_g^\top W r_g$ and we used the rotational invariance of the ℓ_2-norm. Note that since we only made a linear transformation, the constraint remains linear in the transformed problem. Finally, abstracting out the main problem, the block update (with a slight abuse of notation) is of the form

$$\begin{aligned} \underset{x \in \mathbb{R}^d}{\text{minimize}} \quad & \frac{1}{2} x^\top \Sigma x - v^\top x + \lambda \left\| x \right\|_2 \\ \text{subject to} \quad & -\ell \leq A x \leq u \end{aligned}$$

where $\Sigma \in \mathbb{R}_+^{d \times d}$ is a non-negative diagonal matrix, $A \in \mathbb{R}^{m \times d}$, and $\ell, u \in \mathbb{R}_+^d$ such that $\mathbf{0}$ is feasible. For simplicity of the derivations, we further assume that Σ is a *positive* diagonal matrix. In practice, this assumption seldom matters since we may artificially add a small positive value to the diagonal of Σ to remove its singularity. Finally, we assume that $\lambda > 0$ since otherwise, the problem becomes a well-studied quadratic program with a linear inequality constraint, which can be solved using an interior-point method.

Since the block update objective is strictly convex, it is bounded below. By the relaxed Slater's condition, strong duality then holds [Ben-Tal and Nemirovski, 2023]. Hence, we may first form the

(negative) dual function

$$g(\nu) := -\min_x \mathcal{L}(x, \nu)$$

where $\mathcal{L}$ is the Lagrangian given by

$$\mathcal{L}(x, \nu) := \frac{1}{2}x^\top \Sigma x - v^\top x + \lambda \|x\|_2 - \nu_\ell^\top (Ax + \ell) + \nu_u^\top (Ax - u).$$

an $\nu := (\nu_\ell, \nu_u) \in \mathbb{R}^{2m}$ are the dual variables. Then, we solve the dual problem

$$\underset{\nu \geq \mathbf{0}}{\text{minimize}} \; g(\nu)$$

to obtain the optimal dual variable $\nu^\star$. Letting $x^\star(\nu)$ be the (unique) minimizer of the Lagrangian for a fixed ν, we must have that $(x^\star(\nu^\star), \nu^\star)$ is an optimal primal-dual pair. The remainder of the section discusses our method to solve this dual problem.

4.3.1 Proximal Newton method-based algorithm

To solve the dual problem, we propose to use proximal Newton method. The proximal Newton method begins with an initial point $\nu^{(0)}$. At every iteration $k \geq 0$, it forms a quadratic approximation to $g(\cdot)$ and solves the proximal operator for the next update, that is,

$$\nu^{(k+1)} := \underset{\nu \geq \mathbf{0}}{\operatorname{argmin}} \frac{1}{2}(\nu - \nu^{(k)})^\top \nabla^2 g(\nu^{(k)})(\nu - \nu^{(k)}) + \nabla g(\nu^{(k)})^\top (\nu - \nu^{(k)}).$$

We first discuss how to compute the gradient and hessian of g. Note that since the primal solution is always unique, it is a standard convex analysis result [Bazaraa et al., 2006] that the (negative) dual function $g(\cdot)$ is differentiable and

$$\nabla g(\nu) = \begin{bmatrix} \ell \\ u \end{bmatrix} - \begin{bmatrix} -I \\ I \end{bmatrix} Ax^\star(\nu).$$

Moreover, $x^\star(\nu)$ is computed by solving the unconstrained problem:

$$x^\star(\nu) := \underset{x}{\operatorname{argmin}} \mathcal{L}(x, \nu) = \underset{x}{\operatorname{argmin}} \frac{1}{2}x^\top \Sigma x - (v - A^\top(\nu_u - \nu_\ell))^\top x + \lambda \|x\|_2 .$$

It is not difficult to see that this is the same problem as the *unconstrained* block update as in Section 2.3. Hence, we may reuse the Newton-ABS method to compute $x^\star(\nu)$, which readily gives the gradient $\nabla g(\nu)$. The hessian requires a substantial amount of work. In Appendix C.1, we show that the hessian is $\mathbf{0}$ in the interior of the ellipsoid $\{\nu \in \mathbb{R}^{2m} : \left\|v - A^\top(\nu_u - \nu_\ell)\right\|_2 \leq \lambda\}$ and may

not exist at the boundary. Outside the ellipse, we have that

$$\nabla^2 g(\nu) = \begin{bmatrix} -I \\ I \end{bmatrix} AS(\nu)A^\top \begin{bmatrix} -I & I \end{bmatrix}$$

where

$$S(\nu) := \|x^\star(\nu)\|_2 \left(\lambda\kappa(\nu)\alpha(\nu)\alpha(\nu)^\top + (\Sigma\|x^\star(\nu)\|_2 + \lambda I)^{-1}\right) \tag{4.2}$$

$$\kappa(\nu) := \left(1 - \lambda u^\star(\nu)^\top(\Sigma\|x^\star(\nu)\|_2 + \lambda I)^{-1}u^\star(\nu)\right)^{-1} \tag{4.3}$$

$$\alpha(\nu) := (\Sigma\|x^\star(\nu)\|_2 + \lambda I)^{-1}u^\star(\nu). \tag{4.4}$$

Suppose for the moment that $\nabla^2 g$ exists. Let $\mu := \nu_u - \nu_\ell \in \mathbb{R}^m$. Rewriting the proximal operator in terms of μ, we have

$$\mu^{(k+1)} := \underset{\mu\in\mathbb{R}^m}{\operatorname{argmin}} \frac{1}{2}(\mu - \mu^{(k)})^\top AS(\mu^{(k)})A^\top(\mu - \mu^{(k)}) - x^\star(\mu^{(k)})^\top A^\top\mu + \ell^\top\mu_- + u^\top\mu_+. \tag{4.5}$$

Note that the gradient and hessian of $g(\nu)$ only depend on ν through μ since it is true for $x^\star(\nu)$. Without loss of generality, we may then assume the explicit mapping $\mu_+ \equiv \nu_u$ and $\mu_- \equiv \nu_\ell$. It is worth emphasizing that (4.5) is an unconstrained least squares problem with a pinball penalty. Moreover, while the original proximal operator dealt with $2m$ variables, we have reduced the problem to solving for only m variables. Since the penalty is separable, we may use coordinate descent to solve (4.5).

4.3.2 Backtracking for feasibility

Since $\nabla^2 g$ may not always exist, we must take care in our implementation. Before applying the proximal Newton method, we first check whether $\mathbf{0}$ is an optimal primal solution. In other words, we check if there exists any $\mu \in \mathbb{R}^m$ such that $x^\star(\mu) = \mathbf{0}$ and $\mu_- \odot \ell = \mu_+ \odot u = \mathbf{0}$ (complementary slackness). Equivalently, letting $S_- := \{i \in [m] : \ell_i > 0\}$ and $S_+ := \{i \in [m] : u_i > 0\}$, we solve

$$\underset{\mu_{S_+} \le \mathbf{0} \le \mu_{S_-}}{\text{minimize}} \left\|v - A^\top\mu\right\|_2 \tag{4.6}$$

and check whether the minimum objective is less than λ. We note that (4.6) only needs to be solved until the objective is less than λ, in which case we may terminate early and simply return the current iterate for μ and zero for the primal solution. We may solve (4.6), which is a variant of non-negative least squares (NNLS), efficiently using coordinate descent as well. Indeed, we provide an in-depth treatment of implementing a fast NNLS solver in Chapter 5. If no μ exists such that zero is an optimal primal solution, then $\mathbf{0}$ cannot be the primal solution to the block update. This

guarantees that the optimal dual solution must lie outside the ellipsoid where the hessian is well-defined, which suggests that proximal Newton should find the solution. It is, however, still possible that the proximal Newton algorithm yields iterates that fall back into the ellipsoid. In this case, we employ a backtracking algorithm that moves in the direction $\mu^{(k+1)} - \mu^{(k)}$ but with a step-size that guarantees that the new iterate lies outside the ellipsoid.

4.3.3 Warm-starts

Our full solver can be summarized as a nested coordinate descent algorithm. At the outer-most level, we cycle through the blocks of coefficients β_g and each block update is solved via the proximal Newton algorithm, which iteratively evaluates the proximal operator, solved via coordinate descent. Although this algorithm seems convoluted due to this nested structure, it works quite effectively largely thanks to warm-starts. Indeed, since the primal and dual variables are all warm-started, each nested optimization algorithm terminates after a few iterations, making this algorithm tractable in practice.

4.3.4 Screening

As in the unconstrained case, we can extend screening methods such as the strong rule or the pivot rule to the constrained setting. Suppose we determine some subset of groups $\mathcal{S}(\lambda) \subseteq [G]$ that contains the *active set* $\mathcal{A}(\beta^\star(\lambda))$ at the solution $\beta^\star(\lambda)$ where $\mathcal{A}(\beta) := \{g \in [G] : \beta_g \neq \mathbf{0}\}$. Then, we may solve (4.1) only over the groups in $\mathcal{S}(\lambda)$. In practice, we do not know a priori that $\mathcal{S}(\lambda)$ contains the true active set $\mathcal{A}(\beta^\star(\lambda))$, so we must check the KKT conditions for groups not in $\mathcal{S}(\lambda)$ to verify that $\beta^\star(\lambda)_{-\mathcal{S}(\lambda)} \equiv \mathbf{0}$. Similar to the computations in Section 4.3.1, the KKT conditions gives the following equivalent set of conditions:

$$\beta^\star(\lambda)_g = \mathbf{0} \iff \left\|X_g^\top (y - X\beta) - A^\top \mu_g\right\|_2 \leq \lambda\alpha\omega_g, \quad \forall g \notin \mathcal{S}(\lambda), \tag{4.7}$$

for some $\mu_g \in \mathbb{R}^{p_g}$ such that $(\mu_g)_- \odot \ell = (\mu_g)_+ \odot u = \mathbf{0}$. Note that (4.7) can be solved using a fast NNLS solver (see Chapter 5). For the strong or pivot rule, which rely on ranking the active scores, we must use the dual-corrected active scores

$$s_g := \begin{cases} (\alpha\omega_g)^{-1} \left\|X_g^\top (y - X\beta) - A^\top \mu_g^\star\right\|_2, & \alpha\omega_g > 0 \\ \lambda, & \alpha\omega_g = 0 \end{cases}$$

where

$$\mu_g^\star := \operatorname*{argmin}_{(\mu_g)_- \odot \ell = (\mu_g)_+ \odot u = \mathbf{0}} \left\|X_g^\top (y - X\beta) - A^\top \mu_g\right\|_2.$$

We note that it is imperative that the NNLS solver can efficiently solve (4.7). Since these problems are only solved at the end of each outer-most block-coordinate descent loop, they are the least warm-started part of the algorithm, especially at the start of the solver. This is the primary reason for why we dedicate a separate chapter for studying (high-dimensional) NNLS problems in great detail.

4.3.5 Extension to general convex losses

The extension to general convex losses from the Gaussian loss follows via the proximal quasi-Newton method as in Section 2.4. The only change is in the dual-corrected active scores in Section 4.3.4 where $y - X\beta$ must be replaced with $-\nabla\ell(\eta)$ with $\eta = X\beta$ being the current linear prediction.

4.4 Convex 2-layer ReLU neural network

In this section, we discuss our main application of solving polyhedron constrained group lasso problems. Consider a feature matrix $X \in \mathbb{R}^{n\times d}$ and a response vector $y \in \mathbb{R}^n$. We typically train 2-layer ReLU neural network models by solving the optimization problem given by

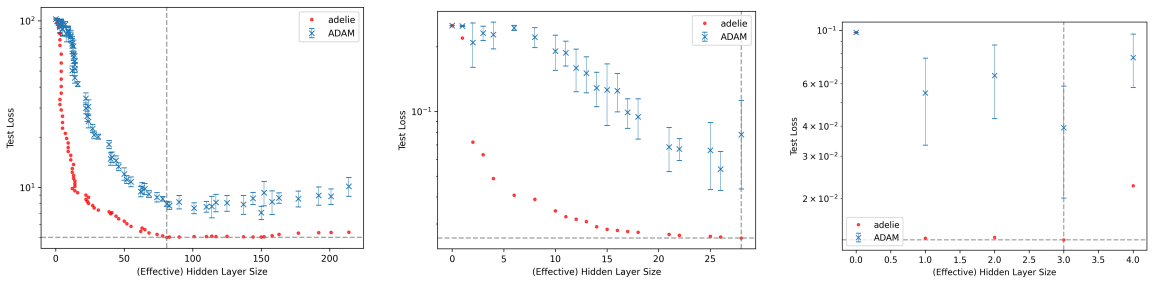
$$\underset{\{\alpha_j,u_j\}_{j=1}^m}{\text{minimize}} \ \frac{1}{2}\left\| y - \sum_{j=1}^m (Xu_j)_+\alpha_j \right\|_2^2 + \frac{\beta}{2}\sum_{j=1}^m \left(\|u_j\|_2^2 + \alpha_j^2\right) \tag{4.8}$$

where $\beta > 0$ is a regularization parameter, $\{u_j \in \mathbb{R}^d\}_{j=1}^m$ are the hidden-layer weights, and $\{\alpha_j \in \mathbb{R}\}_{j=1}^m$ are the output-layer weights. Namely, we minimize the (training) mean-squared-error (MSE) for the 2-layer ReLU network with a ridge regularization on the weights for better generalizability. It is worth noting that (4.8) is highly non-convex due to the non-linear ReLU activations. The best algorithm to solve this problem is a brute-force search that is exponential in m [Pilanci and Ergen, 2020].

Pilanci and Ergen [2020] shows a remarkable result that (4.8) admits a convex reformulation, which can identify the *global* optimum of the non-convex problem. Moreover, the reformulation is a polyhedron constrained group lasso problem given by

$$\begin{aligned} \underset{\{v_i,w_i\}_{i=1}^P}{\text{minimize}} \quad & \frac{1}{2}\left\| y - \sum_{i=1}^P D_i X(v_i - w_i) \right\|_2^2 + \beta\sum_{i=1}^P \left(\|v_i\|_2 + \|w_i\|_2\right) \\ \text{subject to} \quad & (2D_i - I)Xv_i \geq \mathbf{0}, \quad (2D_i - I)Xw_i \geq \mathbf{0}, \quad \forall i \in [P]. \end{aligned} \tag{4.9}$$

Furthermore, there exists a simple and explicit mapping from the solutions $\{v_i^\star, w_i^\star\}_{i=1}^P$ to the original network weights $\{u_j, \alpha_j\}_{j=1}^m$ (see Pilanci and Ergen [2020]). In particular, any non-zero coefficients $v_i^\star$ or $w_i^\star$ get mapped to non-zero weights for some pair (u_j, α_j) and all other (u_j, α_j) are set to zero.



(a) Boston Housing (b) Satellite (c) MNIST

Figure 4.1: Comparison of the test MSE between `adelie` for solving (4.9) and ADAM for solving (4.8) across multiple datasets. These plots show that solving the convex reformulation does provide sparse models that predict better than any model trained via ADAM.

As a result, any solution to (4.9) readily yields a bona fide 2-layer ReLU neural network and preserves sparsity. Here, $D_i \in \{0,1\}^{n\times n}$ are diagonal binary matrices that enumerate all possible values of $\text{diag}(\mathbb{1}_{Xu\geq \mathbf{0}})$ over all $u \in \mathbb{R}^d$. Since P grows exponentialy in $\text{rank}(X)$, enumerating all possible D_i's becomes an intractable task. In practice, we must resort to methods that sub-sample the D_i's. Pilanci and Ergen [2020] suggests that approximating the set of D_i's by sampling $u \sim \mathcal{N}(0, I)$ performs extremely well in practice. In our experiments, we sample 1000 to 2000 u_i's, which gives excellent performance. We note that solving (4.9) for a sub-sample of D_i's leads to a *local minimum* for (4.8). Hence, it is advantageous to implement a sub-sampling strategy that can capture as many active D_i's (i.e., D_i such that $v_i^\star \neq w_i^\star$ from solving the full problem (4.9)) as possible so that the local minimum is as close to the global minimum. One advantage of the convex reformulation is that it decouples the problem of finding predictive D_i's and the problem of obtaining the optimal weights that obey these *gates* in the network (i.e., satisfying the linear constraints); we can view the non-convex problem as trying to solve both problems simultaneously.

There are several interesting consequences of solving (4.9) efficiently. Firstly, we have far fewer hyperparameters to tune compared to the usual SGD methods (e.g., ADAM) that have number of epochs, learning rates, running average coefficients, and numerically stabilizing parameters. The only hyperparameter is the regularization parameter β, which we can cleverly grid [Yang and Hastie, 2024b, Friedman et al., 2010] and automatically tune via validation. Secondly, due to the pathwise-solving property of our constrained group lasso solver, we now have *principled model selection* for 2-layer ReLU neural networks, which has never been formalized to the best of our knowledge. Lastly, we typically obtain sparse networks, which is desirable in practice as sparser networks tend to generalize better and carry less memory footprint.

In Figure 4.1, we show a performance comparison of training a 2-layer ReLU network using ADAM to solve the non-convex problem (4.8) and `adelie` to solve the convex reformulation (4.9). We consider the following datasets: Boston Housing [Harrison and Rubinfeld, 1978], 294 Satellite

Data	n	d	Time	Cores
Boston Housing	404	13	30s	8
Satellite	5148	36	45s	8
MNIST	60000	784	12m	64

Table 4.1: Real data description.

Image from PMLB [Romano et al., 2021], and MNIST [Deng, 2012]. For ADAM, we use the default configurations in `PyTorch` and perform epoch tuning by picking the best model (based on validation error) across a wide range of epochs. For each hidden layer size m, we train a network using ADAM over 10 random initializations. The "x" marks the average test loss observed across the 10 initializations along with 2 standard error bars above and below. For `adelie`, we solve (4.9) over a grid of 100 β values and use the number of active groups of coefficients (for each solution) to determine the (effective) hidden layer size of the induced network; we then plot the corresponding test loss in red dots. For all datasets, we sub-sampled 2000 D_i's via the Gaussian-sampling scheme described earlier except for MNIST where we sub-sampled to 1000. In Table 4.1, we show the data sizes for the feature matrix X as well as the training time and number of cores across the different datasets. Boston Housing and Satellite datasets ran on a standard M1 Macbook Pro with 8 cores in less than a minute while the MNIST dataset was so large that we ran on a cluster with 64 cores. We note that as the number of samples n grows, the convex problem becomes *extremely non-trivial* as we have n linear inequality constraints for *each* coefficient v_i and w_i, $i = 1, \dots, m$, where m is the number of D_i's. For MNIST, we then have $2nm \equiv 120$ million linear inequality constraints! Based on the results, we conclude that the convex reformulation indeed finds sparse networks that predict better than any ADAM-trained network across the same grid of hidden layer sizes. In particular, the best model chosen via `adelie` performs significantly better than the best model chosen via ADAM. One peculiar phenomenon is that as the dataset grows, we observe less of a difference between the two curves. We believe that this occurs specifically due to the increase in n, which naturally increases the number of D_i's to consider. Consequently, our sub-sampling method seems to become less effective in finding the active D_i's, and the same must be true for ADAM simply because it cannot escape the curse of dimensionality either. While both methods seem bottlenecked by finding the active gates, the convex reformulation has an advantage in that sub-sampling the gates can be separated out as a stand-alone problem. We believe that a more refined sub-sampling scheme will substantially improve the convex reformulation method and we leave this as a future work.

4.5 Discussion

We develop a novel algorithm to solve polyhedron constrained group lasso and elastic net penalized regression problems via block-coordinate descent. We show that a dual formulation of the block

update naturally leads to a proximal Newton algorithm where the proximal operators can be easily solved using another coordinate descent solver. To demonstrate the effectiveness of our method, we solve highly non-trivial problems in convex 2-layer ReLU neural network training within minutes using a wide range of data sizes. Our method not only solves the problem efficiently, but also obtains better prediction power than networks trained via the state-of-the-art ADAM.

Chapter 5

Polyhedron constrained least squares

5.1 Introduction

In this final chapter, we discuss a largely theoretical work that establishes the existence of a locally unique sparse solution to polyhedron constrained least squares problems. This result unifies many empirical observations across various domains where practitioners have noted sparsity in their solutions to seemingly different optimization problems. While theoretically interesting on its own, this result also demonstrates to be a powerful tool for devising a heuristic to solve high-dimensional non-negative least squares problems quickly.

Sparse regression methods have grown increasingly popular recently with the prevalance of high-dimensional data and large computing resource. Notably, the lasso [Tibshirani, 1996] is widely used for feature selection as it often sets many coefficients exactly to zero. It is particularly useful in ultra high-dimensional settings such as in genomics [He et al., 2022, Qian et al., 2020, Li et al., 2020] where feature selection is especially meaningful for interpretability. The non-negative least squares (NNLS) [Lawson and Hanson, 1995] is also a reliable tool for many practitioners across diverse fields as it often provides a physically meaningful and interpretable solution due to its empirical sparsity [Li and Speed, 2000, Slawski and Hein, 2013, Chou et al., 2022]. Some theoretical work exists around understanding this behavior [Bruckstein et al., 2008, Wang and Tang, 2009, Wang et al., 2011]. A natural extension of NNLS is the bounded variable least squares (BVLS) [Stark and Parker, 1995], which extends the non-negativity constraint to any box constraint. Similar sparsity occurs in high dimensions where many coefficients lie at their respective boundaries. The simplex constrained least squares (SLS) is at the heart of synthetic controls in causal inference [Abadie, 2021] where the potential outcome of no intervention for the treatment group is estimated using a weighted average

of the donor pool's outcomes. Abadie [2021] notes that we often observe sparsity in the weighted average, which is desirable for interpretability among practitioners especially when the donor pool size is large. Indeed, we see in Lei and Sudijono [2024][Table 2] that many real data examples have more donors than the pre-intervention time periods, which guarantees the existence of a sparse solution per our results in Section 5.3. Finally, the inequality simplex constrained least squares (ISLS) is commonly seen in portfolio optimization, e.g., a long-only portfolio under a Markowitz model with a leverage constraint so that the asset weights w satisfy $w \geq 0$ and $\sum_i w_i \leq C$ for some $C > 0$. The model may be in a high-dimensional setting if the covariance matrix of the assets $\Sigma \in \mathbb{R}^{p\times p}$ admits a low-rank representation $\Sigma = FF^\top$ where $F \in \mathbb{R}^{p\times k}$ and $k < p$. In this case, we often observe near zero weights for many assets using a standard interior point solver.

While these methods may seem disparate at first glance, they share a deep connection through the more general problem of polyhedron constrained least squares. Our first contribution is a theoretical result in Section 5.3 that establishes the existence of a "sparse" solution to any polyhedron constrained least squares problem, thereby unifying our understanding of sparse solutions to the aforementioned methods. The notion of sparsity is extended to mean that the solution *binds* many linear inequality constraints as equality constraints in the polyhedron (see Definition 5.3.1). Moreover, we quantify the number of such binding constraints and show the *local uniqueness property*, which states that such sparse solutions are unique within the affine subspace defined by the binding constraints. We apply our results to NNLS, BVLS, SLS, ISLS, lasso, and box constrained lasso to show that the promised sparsity is at least $p - \text{rank}(X) + O(1)$ where $X \in \mathbb{R}^{n\times p}$ is the feature matrix. Lastly, we argue that the local uniqueness property helps explain the success of many active-set optimizers for these problems. Our second contribution is a computational result in Section 5.4 that uses the previous result as motivation to derive a coordinate descent based solver for BVLS and NNLS with screening and an active-set strategy. We show in Section 5.5 that our solver performs competitively in terms of accuracy and speed in high-dimensional settings.

5.2 Preliminaries and notations

Let $\mathbb{R}$, $\mathbb{R}^n$, and $\mathbb{R}^{m\times n}$ denote the space of real numbers, real vectors of size n, and real matrices of shape $m \times n$, respectively. Denote $\overline{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$ as the extended real line. We denote $\mathbf{0}$ and $\mathbf{1}$ to be the vector or matrix of zeros and ones, respectively, depending on the context. We define $[n] := \{1, \ldots, n\}$ to be the set of integers ranging from 1 to n. For any vector $v \in \mathbb{R}^n$ and a set of indices $S \subseteq [n]$, let v_S be the subset of v along indices in S. We denote $v_{-S} \equiv v_{S^c}$ as the subset of v along indices not in S. Similarly, for any matrix A, we denote A_S as the rows of A in S and $A_{-S} \equiv A_{S^c}$. For any two vectors $x, y \in \mathbb{R}^n$, we define $x \leq y$ to be the element-wise less-than comparison operator and similarly for other comparison operators. Let $x \odot y$ denote the Hadamard product of two vectors. The ℓ_p-norm of a vector x is given by $\|x\|_p := (\sum_{i=1}^n |x_i|^p)^{1/p}$.

For a matrix $X \in \mathbb{R}^{m\times n}$, $\text{span}(X)$ denotes the linear span of the columns of X. For a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$, we denote its gradient as ∇f and the ith partial derivative as $\partial_i f$. Given a set $C \subseteq \mathbb{R}^n$, $\mathbb{1}_C(x)$ is defined to be 1 whenever $x \in C$ and 0 otherwise, unless stated otherwise. The positive and negative parts of a value $x \in \mathbb{R}$ are given by $x_+ := \max(x, 0)$ and $x_- := \max(-x, 0)$, respectively. We denote $\Pi_C(\cdot)$ as the projection operator onto the set C whenever defined. For a matrix X, we may abuse the notation and denote $\Pi_X(\cdot) \equiv \Pi_{\text{span}(X)}(\cdot)$.

5.3 Theoretical results

In this section, we first provide some background on the theory of polyhedrons for completeness in Section 5.3.1. Then, we discuss our main theoretical results in Section 5.3.2 and apply these results in Section 5.3.3 to establish the existence of a sparse solution to various sparse regression models.

5.3.1 Background

We first begin with the definition of a polyhedron. Note that some authors define a polyhedron more generally and refer to Definition 5.3.1 specifically as a *convex polyhedron*. Since we always work with convex polyhedra in this paper, we simplify the terminology.

Definition 5.3.1 (Polyhedron)**.** *$\mathcal{P} \subseteq \mathbb{R}^p$ is a* polyhedron *if there exists $A \in \mathbb{R}^{m\times p}$ and $b \in \mathbb{R}^m$ for some m such that $\mathcal{P} \equiv \{x \in \mathbb{R}^p : Ax \leq b\}$. We say that the polyhedron $\mathcal{P}$ is* represented by *A and b. An element $x \in \mathcal{P}$* binds *the ith constraint if $a_i^\top x = b_i$. Moreover, we say $x \in \mathcal{P}$* binds *k constraints if there exists $S \subseteq [m]$ of size k such that x binds the ith constraint for every $i \in S$. We say a constraint a_i is non-zero if $a_i \neq \mathbf{0}$.*

It is sometimes useful to identify a polyhedron with a set of "non-trivial" inequality constraints and a set of equality constraints that are always satisfied by its elements. Concretely, it is useful to represent a polyhedron $\mathcal{P}$ as

$$\mathcal{P} = \{x \in \mathbb{R}^p : A_{-S}x \leq b_{-S},\ A_S x = b_S\} \tag{5.1}$$

where $S := \left\{i \in [m] : a_i^\top x = b_i,\ \forall x \in \mathcal{P}\right\}$ is the set of equality constraints. Lemma D.1.2 shows that $\{x \in \mathbb{R}^p : A_{-S}x \leq b_{-S}\}$ has a non-empty interior, which is a way to express that this set of inequality constraints is "non-trivial." Moreover, the affine space $\{x \in \mathbb{R}^p : A_S x = b_S\}$ is uniquely determined. This motivates our definition of a dimension of a polyhedron.

Definition 5.3.2 (Dimension of a polyhedron)**.** *Let $\mathcal{P} \subseteq \mathbb{R}^p$ be a non-empty polyhedron represented by $A \in \mathbb{R}^{m\times p}$ and $b \in \mathbb{R}^m$. Let $S := \left\{i \in [m] : a_i^\top x = b_i,\ \forall x \in \mathcal{P}\right\}$. We call S the* maximal affine constraints *of $\mathcal{P}$. We define the* dimension of a polyhedron $\mathcal{P}$ *denoted as* $\dim(\mathcal{P})$ *to be $p - \text{rank}(A_S)$.*

Intuitively, $p - \text{rank}(A_S)$ is the degrees of freedom in representing an element $x \in \mathcal{P}$, and hence, the effective dimension of the polyhedron. Since the affine space determined by S is uniquely determined, $\dim(\mathcal{P})$ is well-defined.

Next, we extend the notion of a *hull* in Definition 5.3.3.

Definition 5.3.3 (Hull)**.** *Let $Q \in \mathbb{R}^{n\times p}$ be any matrix and $\mathcal{P} \subseteq \mathbb{R}^p$ be any set. The* $\mathcal{P}$-hull *of Q is defined to be*

$$\text{Hull}(Q, \mathcal{P}) := \{x \in \mathbb{R}^n : x = Qw,\, w \in \mathcal{P}\}. \tag{5.2}$$

If $\mathcal{P}$ is empty, we take $\text{Hull}(Q, \mathcal{P}) \equiv \mathbb{R}^n$*. An element of* $\text{Hull}(Q, \mathcal{P})$ *is called a* $\mathcal{P}$-combination of Q*. We say that $x \in \text{Hull}(Q, \mathcal{P})$ is a* $\mathcal{P}$-combination of Q with $w \in \mathcal{P}$ *if $x = Qw$.*

Definition 5.3.3 is a bona fide extension of commonly used hulls. For example, a convex hull of a set of points Q is $\text{Hull}(Q, \mathcal{P})$ where $\mathcal{P}$ is the unit simplex; a conical hull of Q is where $\mathcal{P}$ is the positive orthant. This extension will become useful as we will soon discuss the $\mathcal{P}$-hull of Q when $\mathcal{P}$ is a polyhedron. As a first result, we show in Lemma D.1.3 that if $\mathcal{P}$ is a polyhedron, then $\text{Hull}(Q, \mathcal{P})$ is also a polyhedron.

5.3.2 Main results

In this section, we discuss our main results for showing the existence of a sparse polyhedral regression solution in a high-dimensional setting and the local uniqueness property of such solutions. We first give a high-level description of our results. Suppose $\mathcal{P}$ is a polyhedron represented by $A \in \mathbb{R}^{m\times p}$ and $b \in \mathbb{R}^m$. We will show that there always exists a solution to the polyhedral regression

$$\underset{\beta\in\mathcal{P}}{\text{minimize}}\ \frac{1}{2}\|y - X\beta\|_2^2 \tag{5.3}$$

that binds many constraints of A. Moreover, we will show the local uniqueness property that such a solution is unique when restricted to the affine subspace defined by the binding constraints.

We first restrict our attention to the simple case of a polyhedron $\mathcal{P}$ with a non-empty interior. Hence, the maximal affine constraints S is empty. Suppose that $x \in \text{Hull}(Q, \mathcal{P})$ so that $x = Qw$ for some $w \in \mathcal{P}$. Our goal is to show that we can find a (possibly different) $\tilde{w} \in \mathcal{P}$ that binds at least a certain number of constraints of A while reproducing x, i.e., $x = Q\tilde{w}$. Lemma 5.3.1 formalizes this claim and quantifies the number of binding constraints. Moreover, it also shows that the binding constraints form a full row rank sub-matrix of A, a fact that will become useful when we prove the local uniqueness property in Theorem 5.3.3.

Lemma 5.3.1. *Let $Q \in \mathbb{R}^{n\times p}$ be any matrix and $\mathcal{P} \subseteq \mathbb{R}^p$ be any polyhedron represented by $A \in \mathbb{R}^{m\times p}$ and $b \in \mathbb{R}^m$ with a non-empty interior. Suppose that* $\text{Null}(Q) \setminus \{\mathbf{0}\} \subseteq \text{Null}(A)^c$*. If x is in the $\mathcal{P}$-hull of Q, then x can be written as a $\mathcal{P}$-combination of Q with an element w that binds at least*

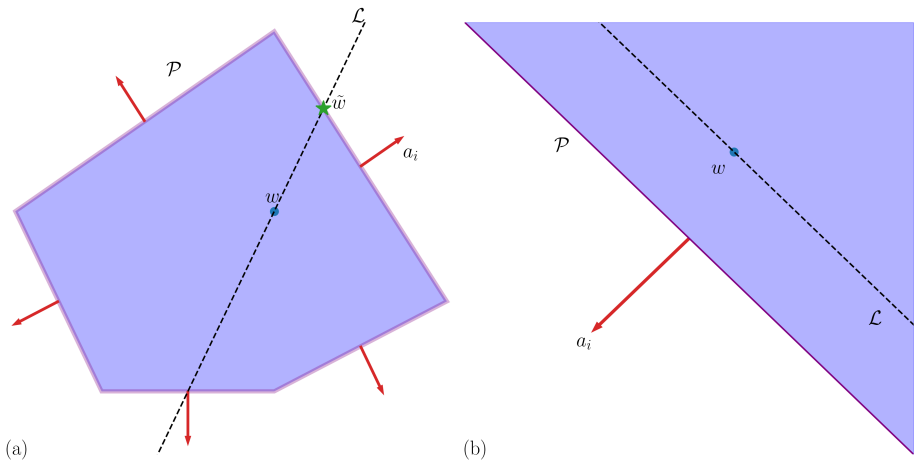


Figure 5.1: Visualization of the main proof idea of Lemma 5.3.1. The polyhedron $\mathcal{P} \subseteq \mathbb{R}^2$ (with a non-empty interior) is given by the shaded blue region defined by the constraints a_i (red arrows). Suppose $w \in \mathcal{P}$ (blue dot) is such that $x = Qw$. If $v \neq \mathbf{0} \in \text{Null}(Q)$, then any point along the line $\mathcal{L} := \{w + tv : t \in \mathbb{R}\}$ (dotted black line) reproduces x. As long as the direction v is not orthogonal to every constraint, i.e., $Av \neq \mathbf{0}$, then there always exists an intersection point $\tilde{w} \in \mathcal{L} \cap \mathcal{P}$ by Lemma D.1.4. The left plot shows the case of a polytope when $Av \neq \mathbf{0}$ with the intersection point $\tilde{w}$ denoted by a green star. On the other hand, the right plot shows the opposite case of a single half-space where no such an intersection point exists.

$\min(\text{rank}(A), (p-n)_+)$ *constraints of* A *that form a full row rank sub-matrix of* A.

We give a proof of Lemma 5.3.1 in Appendix D.2. Note that Lemma 5.3.1 is only interesting in the high-dimensional setting of $p > n$. The main idea of the proof is to consider a solution $w \in \mathcal{P}$ such that $x = Qw$ and move w along a direction in the null space of Q until a boundary of $\mathcal{P}$ is hit. Once we hit a boundary, we may restrict ourselves to that facet of the polyhedron and repeat the process in this lower-dimensional polyhedron. The reader should imagine "peeling" off the facet, discarding the rest of the polyhedron, and viewing this facet in a lower-dimensional space. Each time we restrict to the lower-dimensional space, we remove a degree of freedom from both Q and $\mathcal{P}$. Since $p > n$, we can then guarantee that at least $p - n$ degrees of freedom can be removed, so long as there are enough degrees of freedom, i.e., $\text{rank}(A)$, to remove from $\mathcal{P}$. With this intuition in mind, it is no surprise that Lemma 5.3.1 holds.

We now discuss a few technical aspects of Lemma 5.3.1. The hypothesis that $\text{Null}(Q) \setminus \{\mathbf{0}\} \subseteq \text{Null}(A)^c$ guarantees that moving along *any* direction in $\text{Null}(Q)$ will hit a boundary of $\mathcal{P}$ with the help of Lemma D.1.4. This condition may seem convoluted at first, but it has a meaningful geometrical interpretation. In words, any non-zero vector $v \in \text{Null}(Q)$ must not be orthogonal to every constraint of A, i.e., $Av \neq \mathbf{0}$. We give a visualization in Figure 5.1. We show in the left plot a case of a polytope where any direction v (not just in $\text{Null}(Q)$) satisfies $Av \neq \mathbf{0}$. On the other hand, the right plot depicts the case of a single half-space where our condition may fail (specifically along

the direction parallel to the boundary). In the same vein, the assumption that $\mathcal{P}$ has a non-empty interior is also crucial, since otherwise, the line $\mathcal{L} = \{w + tv : t \in \mathbb{R}\}$ may only cross at w while satisfying $Av \neq \mathbf{0}$ for every $v \in \text{Null}(Q)$ (imagine $\mathcal{P}$ on the xy-plane and a line piercing through w not parallel to the z-axis).

We now remove the assumption that $\mathcal{P}$ has a non-empty interior while also strengthening the conclusion of Lemma 5.3.1. We show in Theorem 5.3.2 our main theoretical result, which is an extension of Lemma 5.3.1 to the most general case.

Theorem 5.3.2 (Existence of a sparse representation)**.** *Let $Q \in \mathbb{R}^{n\times p}$ be any matrix and $\mathcal{P} \subseteq \mathbb{R}^p$ be any polyhedron represented by $A \in \mathbb{R}^{m\times p}$ and $b \in \mathbb{R}^m$ with S as its maximal affine constraints. Further let $V \in \mathbb{R}^{p\times \dim(\mathcal{P})}$ be any full column rank matrix such that $A_S V = \mathbf{0}$. Suppose that $\text{Null}(QV)\setminus\{\mathbf{0}\} \subseteq \text{Null}(A_{-S}V)^c$. If x is in the $\mathcal{P}$-hull of Q, then x can be written as a $\mathcal{P}$-combination of Q with an element w that binds at least $\min(\text{rank}(A)-\text{rank}(A_S), \dim(\mathcal{P})-\text{rank}(QV))$ constraints of A_{-S} that form a full row rank sub-matrix.*

We give a proof of Theorem 5.3.2 in Appendix D.3. We first explain the assumptions of the theorem. As explained previously, a polyhedron may have an empty interior, and therefore, Lemma 5.3.1 cannot be applied directly. The remedy is the same as in the previous discussion; we first find the subspace in which the polyhedron lies (i.e., given by S) and remove the degrees of freedom from Q and $\mathcal{P}$ to restrict to a lower-dimensional space. We then invoke Lemma 5.3.1 on the lower-dimensional space where the polyhedron is guaranteed to have a non-empty interior by Lemma D.1.5. The proof reveals that the new set of points is precisely QV and the new polyhedron is represented by the constraint matrix $A_{-S}V$. Moreover, we show in the proof that $\text{rank}(A_{-S}V) = \text{rank}(A) - \text{rank}(A_S)$. Lastly, we note that $\dim(\mathcal{P}) - \text{rank}(QV)$ has been improved from $(\dim(\mathcal{P}) - n)_+$. We argue that without loss of generality we may assume QV to be full row rank since otherwise we can perform a QR-decomposition and use the "R" matrix in place of QV.

We provide a few consequences of Theorem 5.3.2. The first consequence is Corollary 5.3.2.1, which shows that the famous Carathéodory's Theorem is an immediate corollary. Secondly, we can finally establish the existence of a "sparse" solution for polyhedral regression. This is formally stated in Corollary 5.3.2.2.

Corollary 5.3.2.1 (Carathéodory's Theorem)**.** *Let $Q \in \mathbb{R}^{n\times p}$ be any matrix. If x is a convex combination of the columns of Q, then it can be written as a convex combination of at most $n+1$ columns of Q. Equivalently, if x is a conical combination of the columns of Q, then it can be written as a conical combination of at most n columns of Q.*

Proof. Without loss of generality, we may assume $p > n$.

We first consider the case of convex combinations. Define the polyhedron $\mathcal{P} \subseteq \mathbb{R}^p$ represented

by

$$A := \begin{bmatrix} \mathbf{1}^\top \\ -I \end{bmatrix}, \quad b := \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}. \tag{5.4}$$

Then, $S := \{1\}$ forms the maximal affine constraints. It is clear that $\dim(\mathcal{P}) = \text{rank}(A) - \text{rank}(A_S) = p - 1$. Then, for the matrix V in Theorem 5.3.2, which is injective, we have that $\text{Null}(QV) \setminus \{\mathbf{0}\} \subseteq \mathbb{R}^{p-1} \setminus \{\mathbf{0}\} = \text{Null}(V)^c = \text{Null}(A_{-S}V)^c$. By definition, the convex hull of Q is equivalent to the $\mathcal{P}$-hull of Q. Then, if x is a convex combination of Q, i.e., a $\mathcal{P}$-combination of Q, then by Theorem 5.3.2, it is a $\mathcal{P}$-combination of Q with w that binds at least $p - 1 - \text{rank}(QV)$ constraints of A_{-S}. In other words, $w \in \mathcal{P}$ contains at most $\text{rank}(QV) + 1 \leq n + 1$ positive values so that $x = Qw$ is a convex combination of at most $n + 1$ columns of Q.

The second case of conical combinations is easier. Define the polyhedron $\mathcal{P} \subseteq \mathbb{R}^p$ represented by

$$A := -I, \quad b := \mathbf{0} \tag{5.5}$$

It is clear that $\dim(\mathcal{P}) = \text{rank}(A) - \text{rank}(A_S) = p$. Since the maximal affine constraints is empty and A is invertible, the conditions of Theorem 5.3.2 are satisfied. If x is a conical combination of Q, i.e., a $\mathcal{P}$-combination of Q, then by Theorem 5.3.2, it is a $\mathcal{P}$-combination of Q with w that binds at least $p - \text{rank}(Q)$ constraints of A. In other words, $w \in \mathcal{P}$ contains at most $\text{rank}(Q) \leq n$ positive values so that $x = Qw$ is a conical combination of at most n columns of Q. □

Corollary 5.3.2.2 (Polyhedral regression)**.** *Let $X \in \mathbb{R}^{n \times p}$ be a feature matrix and $y \in \mathbb{R}^n$ a response vector. Suppose $\mathcal{P} \subseteq \mathbb{R}^p$ is a polyhedron represented by $A \in \mathbb{R}^{m \times p}$ and $b \in \mathbb{R}^m$ that satisfies the conditions of Theorem 5.3.2 with $Q \equiv X$. Let S be the maximal affine constraints of $\mathcal{P}$ and $V \in \mathbb{R}^{p \times \dim(\mathcal{P})}$ such that $A_S V = \mathbf{0}$. Consider the polyhedral regression given by* (5.3)*. Then, there exists a solution $\hat{\beta} \in \mathbb{R}^p$ that binds at least* $\min(\text{rank}(A) - \text{rank}(A_S), \dim(\mathcal{P}) - \text{rank}(XV))$ *constraints of A_{-S} that form a full row rank sub-matrix.*

Proof. Note that any solution β for (5.3) must satisfy

$$\Pi_{\text{Hull}(X,\mathcal{P})}(y) = X\beta \tag{5.6}$$

where the projection exists since $\text{Hull}(X, \mathcal{P})$ is closed and convex, which follows directly from Lemma D.1.3. By Theorem 5.3.2, since $\Pi_{\text{Hull}(X,\mathcal{P})}(y) \in \text{Hull}(X, \mathcal{P})$, we have the desired claim. □

We conclude this section with a final result on *the local uniqueness property* of polyhedrons in Theorem 5.3.3. The proof is provided in Appendix D.4.

Theorem 5.3.3 (Local uniqueness property)**.** *Consider the setting of Theorem 5.3.2. Suppose that*

$$\operatorname{rank}(A) - \operatorname{rank}(A_S) \geq \dim(\mathcal{P}) - \operatorname{rank}(QV).$$

Let w be the sparse representation that binds at least $\dim(\mathcal{P}) - \operatorname{rank}(QV)$ constraints of A_{-S}. Let T be the set of indices such that $(A_{-S})_T$ denotes the binding constraints. Define

$$\mathcal{T} := \{w \in \mathbb{R}^p : (A_{-S})_T w = (b_{-S})_T\}.$$

Then, w is the unique element in $\mathcal{P} \cap \mathcal{T}$ such that $x = Qw$.

The local uniqueness property says that the sparse solution w given by Theorem 5.3.2 must be unique in a sufficiently restricted subspace. The condition for Theorem 5.3.3 can be equivalently written as

$$\operatorname{rank}(QV) + \operatorname{rank}(A) \geq p. \tag{5.7}$$

This mild condition formalizes the idea that there must be enough degrees of freedom in the set of points QV and the constraint matrix A combined for the existence of a solution in such a sufficiently restricted subspace. In many practical settings, this condition will hold true as we often have $\operatorname{rank}(A) \geq p$.

The local uniqueness property offers a way of concretely understanding active-set optimization algorithms. Active-set strategies typicially employ heuristics to first enforce many constraints and simplify an optimization problem to a lower-dimensional problem. If the binding constraints are correctly guessed, then the solution to the lower-dimensional problem is optimal for the original problem. These algorithms have found a place in the optimization literature as sometimes the state-of-the-art methods for various polyhedral regression problems such as the non-negative least squares and the lasso [Friedman et al., 2010, Yang and Hastie, 2024b, Lawson and Hanson, 1995, Myre et al., 2017, Stark and Parker, 1995]. Theorem 5.3.3 suggests that if these strategies have an accurate way of guessing the binding constraints of an optimal solution, then the solution is unambiguous in the restricted subspace. Hence, any optimizer should, in principle, quickly converge to the optimum. In Section 5.4, we will utilize this fact to derive a new coordinate descent based solver for BVLS and NNLS that works especially well in high-dimensional settings.

5.3.3 Applications on sparse regression models

In this section, we use our main results from Section 5.3.2 to unify many popular sparse regression models such as NNLS, BVLS, SLS, ISLS, lasso, and box constrained lasso under the umbrella of polyhedral regression. In all these examples, we establish the existence of a sparse solution. In the case of lasso, we rediscover a well-known result that there exists a solution with at most $\operatorname{rank}(X)$

non-zero entries [Tibshirani, 2012]. For all other models, we show an analogous result, which we believe is novel and does not exist in the literature to the best of our knowledge.

Non-negative least squares

The non-negative least squares (NNLS) problem is given by

$$\underset{\beta \geq \mathbf{0}}{\text{minimize}} \ \frac{1}{2} \|y - X\beta\|_2^2 . \tag{5.8}$$

Using Corollary 5.3.2.2, we now show in Corollary 5.3.3.1 that there *always* exists a solution with at most $\text{rank}(X)$ positive entries.

Corollary 5.3.3.1 (Existence of a sparse NNLS solution)**.** *There exists an NNLS solution with at most* $\text{rank}(X)$ *positive values.*

Proof. The polyhedron $\mathcal{P}$ is given by the positive orthant $\mathbb{R}^p_+$, which can be represented by $A = -I$ and $b = \mathbf{0}$. Since A is invertible and the maximal affine constraint set is empty, the conditions of Theorem 5.3.2 are trivially satisfied. Moreover, $\dim(\mathcal{P}) = \text{rank}(A) - \text{rank}(A_S) = p$. By Corollary 5.3.2.2, there exists a solution $\hat{\beta} \in \mathbb{R}^p$ that binds at least $p - \text{rank}(X)$ constraints of A. That is, $\hat{\beta}$ has at most $\text{rank}(X)$ positive values. □

Bounded variable least squares

The bounded variable least squares (BVLS) problem is given by

$$\underset{\ell \leq \beta \leq u}{\text{minimize}} \ \frac{1}{2} \|y - X\beta\|_2^2 , \tag{5.9}$$

which can be thought of as an extension of NNLS. Similar to NNLS, we show in Corollary 5.3.3.2 the existence of a sparse solution to BVLS.

Corollary 5.3.3.2 (Existence of a sparse BVLS solution)**.** *There exists a BVLS solution with at most* $\text{rank}(X)$ *non-boundary values.*

Proof. The polyhedron $\mathcal{P}$ is given by

$$A := \begin{bmatrix} -I \\ I \end{bmatrix}, \quad b := \begin{bmatrix} -\ell \\ u \end{bmatrix}. \tag{5.10}$$

Since A has full column rank and the maximal affine constraint set is empty, the conditions of Theorem 5.3.2 are trivially satisfied. Moreover $\dim(\mathcal{P}) = \text{rank}(A) - \text{rank}(A_S) = p$. By Corollary 5.3.2.2, there exists a solution $\hat{\beta} \in \mathbb{R}^p$ that binds at least $p - \text{rank}(X)$ constraints of A. Note that a coefficient is either non-binding or exactly one of the lower or upper bounds is binding. Hence, we indeed have that $\hat{\beta}$ has at most $\text{rank}(X)$ non-binding entries. □

Simplex constrained least squares

The simplex constrained least squares (SLS) is given by

$$\underset{\beta \geq \mathbf{0}, \mathbf{1}^\top \beta = C}{\text{minimize}} \frac{1}{2} \|y - X\beta\|_2^2 \tag{5.11}$$

and the inequality version (ISLS) is given by

$$\underset{\beta \geq \mathbf{0}, \mathbf{1}^\top \beta \leq C}{\text{minimize}} \frac{1}{2} \|y - X\beta\|_2^2 \tag{5.12}$$

for some $C \geq 0$. We show in Corollary 5.3.3.3 the existence of a sparse solution to both problems.

Corollary 5.3.3.3 (Existence of a sparse ISLS/SLS solution)**.** *There exists an ISLS solution with at most* $\text{rank}(X)$ *positive values, and an SLS solution with at most* $\text{rank}(X) + 1$ *positive values.*

Proof. We first consider the ISLS problem. The polyhedron $\mathcal{P}$ is represented by

$$A := \begin{bmatrix} -I \\ 1^\top \end{bmatrix}, \quad b := \begin{bmatrix} \mathbf{0} \\ C \end{bmatrix}. \tag{5.13}$$

Since A has full column rank and the maximal affine constraint set is empty, the conditions of Theorem 5.3.2 are trivially satisfied. Moreover, $\dim(\mathcal{P}) = \text{rank}(A) - \text{rank}(A_S) = p$. By Corollary 5.3.2.2, there exists a solution $\hat{\beta} \in \mathbb{R}^p$ that binds at least $p - \text{rank}(X)$ constraints of A. If $\mathbf{1}^\top \hat{\beta} < C$, then we have the desired claim. Otherwise, there can be at most $\text{rank}(X) + 1$ positive values in $\hat{\beta}$. Without loss of generality, suppose $\hat{\beta}$ has exactly $\text{rank}(X) + 1$ positive values. Let $\mathcal{A} := \left\{ j \in [p] : \hat{\beta}_j > 0 \right\}$. Note that $X_{\cdot \mathcal{A}}$ must have a non-trivial null space since $|\mathcal{A}| = \text{rank}(X) + 1$ by construction. Consider any $v \neq \mathbf{0} \in \text{Null}(X_{\cdot \mathcal{A}})$. Without loss of generality, we may assume $\mathbf{1}^\top v \leq 0$. Then, v must have some strictly negative entry since otherwise $\mathbf{1}^\top v \leq 0$ implies that $v \equiv \mathbf{0}$, which is a contradiction. Then, $\mathbf{1}^\top (\hat{\beta}_{\mathcal{A}} + tv) \leq C$ for all $t \geq 0$ and there must exist a smallest t (in magnitude) such that $\hat{\beta}_{\mathcal{A}} + tv$ contains a zero while remaining in $\mathcal{P}$. Moreover, $X_{\mathcal{A}}(\hat{\beta}_{\mathcal{A}} + tv) = X_{\mathcal{A}} \hat{\beta}_{\mathcal{A}}$ by construction so that the objective remains the same. We have then constructed a possibly different optimal solution with at most $\text{rank}(X)$ positive values.

For the SLS problem, the polyhedron $\mathcal{P}$ is the same as in the ISLS case but the last constraint is always binding. Since $A_{-S} \equiv -I$ is invertible, the conditions of Theorem 5.3.2 are trivially satisfied. Moreover, $\dim(\mathcal{P}) = \text{rank}(A) - \text{rank}(A_S) = p - 1$. Let $V \in \mathbb{R}^{p \times \dim(\mathcal{P})}$ such that $A_S V = \mathbf{0}$. By Corollary 5.3.2.2, there exists a solution $\hat{\beta} \in \mathbb{R}^p$ that binds at least $p - 1 - \text{rank}(XV) \geq p - 1 - \text{rank}(X)$ constraints of A_{-S}. That is, $\hat{\beta}$ has at most $\text{rank}(X) + 1$ positive values. $\square$

Lasso

The lasso problem is given by

$$\underset{\beta}{\text{minimize}} \ \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \tag{5.14}$$

for a regularization parameter $\lambda \geq 0$. A perhaps interesting fact is that despite (5.14) being an unconstrained problem, one can show that there exists some $C \geq 0$ such that (5.14) is equivalent to the constrained problem [Hastie et al., 2001]

$$\underset{\|\beta\|_1 \leq C}{\text{minimize}} \ \frac{1}{2}\|y - X\beta\|_2^2 . \tag{5.15}$$

When viewed this way, it becomes clearer why the lasso exhibits very similar behavior to other polyhedral regression models such as NNLS. Namely, the constraint $\|\beta\|_1 \leq C$ defines a polytope. In Corollary 5.3.3.4, we will use (5.15) to rediscover the result that there exists a lasso solution with at most $\text{rank}(X)$ non-zero values. As a novel extension, we show in Corollary 5.3.3.5 that a similar result holds for the box constrained lasso problem given by

$$\underset{\ell \leq \beta \leq u}{\text{minimize}} \ \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \|\beta\|_1 . \tag{5.16}$$

Corollary 5.3.3.4 (Existence of a sparse lasso solution)**.** *There exists a lasso solution with at most* $\text{rank}(X)$ *non-zero values.*

Proof. We use the constrained form (5.15). For any given lasso solution, it must lie in one of the 2^p disjoint partitions of the ℓ_1-ball given by

$$\mathcal{P}^j := \left\{\beta \in \mathbb{R}^p : AD^j\beta \leq b\right\}, \quad A := \begin{bmatrix} -I \\ \mathbf{1}^\top \end{bmatrix}, \quad b := \begin{bmatrix} \mathbf{0} \\ C \end{bmatrix} \tag{5.17}$$

where D^j enumerates all diagonal sign matrices. Hence, it suffices to show that the least squares constrained to any one of the $\mathcal{P}^j$ polyhedrons admits a solution with at most $\text{rank}(X)$ non-zero values. Now, the least squares constrained to $\mathcal{P}^j$ can be reparametrized as

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \frac{1}{2}\left\|y - XD^j\beta\right\|_2^2 \\ \text{subject to} \quad & \beta \geq \mathbf{0}, \quad \mathbf{1}^\top\beta \leq C \end{aligned}$$

which is an ISLS problem. By Corollary 5.3.3.3, there exists a solution $\hat{\beta}$ with at most $\text{rank}(XD^j)$ $\equiv \text{rank}(X)$ (since D^j is invertible) positive values. Then, the original solution is given by $D^j\hat{\beta}$, which does not change the number of non-zero values. □

Corollary 5.3.3.5 (Existence of a sparse box constrained lasso solution)**.** *There exists a box constrained lasso solution with at most* $\operatorname{rank}(X)$ *non-zero, non-boundary values.*

Proof. Consider the proof setup as in Corollary 5.3.3.4. The only change is in $\mathcal{P}^j$, which is now represented as

$$\mathcal{P}^j := \left\{\beta \in \mathbb{R}^p : AD^j\beta \leq b\right\}, \quad A := \begin{bmatrix} -I \\ I \\ \mathbf{1}^\top \end{bmatrix}, \quad b := \begin{bmatrix} \mathbf{0} \\ \tilde{u}^j \\ C \end{bmatrix} \tag{5.18}$$

where $\tilde{u}^j_k = -\ell_k$ if $D^j_{kk} = -1$ and otherwise u_k. The $\mathcal{P}^j$ constrained least squares problem can be reparametrized as

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \frac{1}{2}\left\|y - XD^j\beta\right\|_2^2 \\ \text{subject to} \quad & \mathbf{0} \leq \beta \leq \tilde{u}^j, \quad \mathbf{1}^\top\beta \leq C. \end{aligned}$$

By Corollary 5.3.2.2, there exists a solution $\hat{\beta}$ that binds at most $p - \operatorname{rank}(XD^j) \equiv p - \operatorname{rank}(X)$ constraints. Note that β_k is either non-binding or binding the lower or upper bound, exclusively. If $\mathbf{1}^\top\beta < C$, then we have our desired claim. Otherwise, we may use the same proof as in Corollary 5.3.3.3 for this case. □

5.4 Coordinate descent for BVLS and NNLS

Consider the usual regression setup where $X \in \mathbb{R}^{n\times p}$ and $y \in \mathbb{R}^n$ are the feature matrix and response vector, respectively. Let $\ell \leq u$ be vectors in $\overline{\mathbb{R}}^p$ denoting our lower and upper bounds (which may not be all finite) for the coefficients. In this section, we discuss our algorithm for solving BVLS given by (5.9). Note that BVLS is an extension of NNLS where $\ell \equiv \mathbf{0}$ and u infinite. Our proposed algorithm is specifically designed to tackle high-dimensional settings where $p > n$. Although our only use-case of our method is in solving NNLS, the algorithm extends naturally to the case of BVLS, so we present the more general algorithm. While BVLS may be an ill-posed problem if $p > n$, there nonetheless always exists a solution to (5.9) since the objective is closed, convex, and proper and the constraint is convex.

The most popular optimizer for NNLS is an active-set strategy first developed by Lawson and Hanson (LH-NNLS) [Lawson and Hanson, 1995]. They propose an algorithm that searches for the next variable to be added as a candidate non-zero variable. Then, the unconstrained least squares solution is obtained for the set of non-zero variables. A safety measure is put in place in case a variable is selected incorrectly. Moreover, the algorithm is guaranteed to converge in finite number of steps. Despite the simplicity of this algorithm, it is now considered an out-dated algorithm for the

scale of problems we see today. Since then, many similar algorithms improve upon LH-NNLS while adhering to the active-set strategy [Bro and De Jong, 1997, Van Benthem and Keenan, 2004]. For example, the FNNLS method [Bro and De Jong, 1997] is a direct modification of LH-NNLS designed to cache certain quantities for reuse during subsequent similar computations. Other methods such as the TNT-NN method [Myre et al., 2017] further improve LH-NNLS by solving the least squares solution using a left-preconditioned conjugate gradient descent normal residual routine. We found one reference that uses coordinate descent to solve NNLS [Franc et al., 2005]; however, it does not incorporate clever screening or active-set strategies, which is a large part of our method.

Our algorithm for solving (5.9) is based on the coordinate descent algorithm [Tseng, 2001]. Indeed, letting $\mathbb{1}_{\beta_i \in C}$ to be defined as 0 whenever $x \in C$ and $+\infty$ otherwise, we may rewrite (5.9) as the following unconstrained problem:

$$\underset{\beta}{\text{minimize}} \ \frac{1}{2} \|y - X\beta\|_2^2 + \sum_{i=1}^{p} \mathbb{1}_{\ell_i \leq \beta_i \leq u_i}$$

where the objective is composed of a differentiable convex loss and a non-differentiable separable convex penalty. In this setting, Tseng [2001] provides a convergence guarantee. We choose coordinate descent since it is the state-of-the-art method for fitting sparse regression models such as the lasso and group lasso [Yang and Hastie, 2024b, Friedman et al., 2010]. With the help of Corollary 5.3.3.2, we are assured that there must exist a sparse solution with few coefficients in the interior of the box constraint. Hence, we expect coordinate descent to efficiently find a sparse solution, though no such guarantees exist. Indeed, we observe in some cases, especially when y is perfectly reproducible, that the solution contains more than $\text{rank}(X)$ non-binding entries. However, in the cases when y is not perfectly reproducible, we often observe a sparse solution.

The vanilla coordinate descent algorithm simply cycles through each coefficient β_k and performs a *coordinate update* on β_k, that is, optimize over the kth coordinate while holding all others fixed. The separability and simplicity of the box constraints allow us to implement an efficient closed-form coordinate update. While it is possible in principle to solve BVLS by cycling over all p variables, it generally suffers from a long computation time when $p > n$ (underdetermined system) for two main reasons. First, due to the non-uniqueness of the solution (in general) and collinearity of X, the iterates tend to wander for a long period of time before converging to a solution if the algorithm is not well warm-started. Secondly, the sheer computational complexity increases as p grows since there are more variables to cycle over. Before addressing these issues in full detail, we give a high-level summary of our method. First, we identify a well-chosen (and small) subset of features S that most require adjustments to their coefficients. In the process, we check the Karush-Kuhn-Tucker (KKT) conditions to verify whether the current β is optimal. If the KKT check does not pass, we solve the BVLS problem only on S. This procedure is repeated until the KKT conditions are satisfied or $S \equiv [p]$. We summarize the pseudocode in Algorithm 6 and explain the steps in detail

Algorithm 6: BVLS

Data: X, y, κ

$\beta = \ell \odot \mathbb{1}_{|\ell|<|u|} + u \odot \mathbb{1}_{|\ell|\geq|u|}$;
$\mathcal{S} = \emptyset$;
$\mathcal{A} = \emptyset$;
while *not converged and KKT check not passed* **do**
  Compute and sort violations δ as in (5.20);
  $|\mathcal{S}_{\text{old}}| = |\mathcal{S}|$;
  for $j = 1, \ldots, p$ **do**
    if $\delta_{(p-j+1)} \leq 0$ *or* $(p-j+1) \in \mathcal{S}$ **then**
      continue;
    Set a flag to mark that KKT check did not pass;
    if $|\mathcal{S}| - |\mathcal{S}_{\text{old}}| == \kappa$ **then**
      break;
    Add $(p-j+1)$ into $\mathcal{S}$;
  if *KKT check passed* **then**
    break;
  while *not converged on* $\mathcal{S}$ **do**
    One cycle of coordinate descent on $\mathcal{S}$ and add all active variables to $\mathcal{A}$;
    if *converged* **then**
      Remove binding coefficients from $\mathcal{A}$;
      break;
    Coordinate descent on $\mathcal{A}$ until convergence;
    Remove binding coefficients from $\mathcal{A}$;
return β;

below. We note that this type of active-set strategy is ubiquitous not only in the BVLS and NNLS literature [Lawson and Hanson, 1995, Stark and Parker, 1995, Myre et al., 2017], but also in lasso type regression literature [Friedman et al., 2010, Zou and Hastie, 2005, Yang and Hastie, 2024b].

We first discuss how to choose the subset of features S. Let $f(\beta) = \|y - X\beta\|_2^2/2$ be the objective of BVLS. Given an initial value β, we compute the gradient $\nabla f(\beta) = -X^\top(y - X\beta)$. The KKT conditions for optimality state that β is optimal if and only if

$$\forall i \in [p], \quad \partial_i f(\beta) \in \begin{cases} [0, \infty), & \beta_i = \ell_i \\ 0, & \beta_i \in (\ell_i, u_i) \\ (-\infty, 0], & \beta_i = u_i \end{cases}. \tag{5.19}$$

Hence, it is intuitive to first order the coefficients by their violation of the KKT conditions. Namely,

we compute the *violations*

$$\forall i \in [p], \quad \delta_i := (\partial_i f(\beta))_- \mathbb{1}_{\beta_i < u_i} + (\partial_i f(\beta))_+ \mathbb{1}_{\beta_i > \ell_i} \tag{5.20}$$

and sort in decreasing order from the most violating feature to the least. We say that feature i is *violating* if $\delta_i > 0$. While this heuristic provides a reasonable way to determine the order of features to include in the set S, it is not clear thus far *how many* to pick. Fortunately, Corollary 5.3.3.2 shows that there always exists a solution with $\text{rank}(X)$ entries in the interior of their respective bounds. A simple rule that works well in practice is to batch at most $\text{rank}(X)$ most violating features at a time. If $\text{rank}(X)$ is not known, one can approximate it with $\min(n, p)$. In a sense, this batching creates a "path" of coefficients similar to a penalized regression coefficient path (e.g., lasso) as the penalization is varied.

In the process, we check the KKT conditions on S^c to ensure that we have found an optimal solution. We simply check that $\nabla f(\beta)$ satisfies (5.19), or equivalently, the violations are 0. If the check passes, we have found an optimal solution. Otherwise, we continue the algorithm with the augmented S. Since S is always strictly augmented, we have guaranteed convergence via the coordinate descent algorithm in the worst case that $S \equiv [p]$. In practice, we rarely reach the worst case especially when $p \gg n$.

Given a subset of features S, we next solve BVLS only iterating over S. We discuss a few optimizations to aid in faster convergence. We keep track of the active set, which contains the variables in S that are in the interior of their bounds. We first perform one cycle over S and collect the active set in the process. If we converged, we are done. Otherwise, we cycle over only the active set until convergence. We then remove variables in the active set that are at the boundary and repeat the entire procedure. This strategy significantly improves the performance of the algorithm as S tends to be sufficiently larger than the active set.

We set the initial value β with $\beta_i = \ell_i$ if $|\ell_i| < |u_i|$ and otherwise u_i. This way, β is always initialized at some vertex of the box and is closest to zero entry-wise. Since collinearity is inevitable in high-dimensional settings, it is beneficial to initialize β close to $\mathbf{0}$ to mimic the shrinkage in penalized regression.

5.5 Numerical experiments

In this section, we discuss numerical experiments for our algorithm in Section 5.4. We restrict our attention to NNLS for simplicity and since it is more widely used in practice than the general BVLS problem. We begin with a simulated data setting in Section 5.5.1 and then move to a positive spike deconvolution example on real data in Section 5.5.2. All experiments are run on a standard Macbook M1 Pro. We implement an efficient version of our algorithm in `C++` and include it in the `Python`

package `adelie`[1].

5.5.1 Simulated data

Our simulation setting is as follows. We let $X \in \mathbb{R}_+^{n \times p}$ be a non-negative matrix and $y \in \mathbb{R}^n$ be any vector. This ensures that the conic hull of X does not span $\mathbb{R}^n$ so that the NNLS problem is non-trivial. In practice, there are also numerous examples such as text mining and functional MRI where the data is non-negative [Diakonikolas et al., 2022]. We generate each $X_{ij} \sim \text{Unif}(0, 1)$ independently and set 80% of the entries of X to zero. This results in a matrix with pairwise nearly orthogonal vectors. We generate i.i.d. $y_i \sim \mathcal{N}(\mu, \sigma^2)$ where μ is chosen in the equally-spaced gridding of $[-3\sigma, 3\sigma]$ with 20 points and $\sigma = 1$. We solve NNLS using our method (`adelie`) and `MOSEK` [ApS, 2024]. We note that both methods guarantee primal feasibility. We also ran LH-NNLS and FNNLS [Lawson and Hanson, 1995, Bro and De Jong, 1997], but we omit their results since they took significantly longer. Figure 5.2 shows a comparison of the runtime and objective for when $n = 100$ and $n = 1000$. In all cases, $p = 10000$. In general, our method obtains a solution as accurate as `MOSEK`'s with a speed-up factor anywhere from 10-1000x. As `MOSEK` is a general-purpose solver, it is expected that a specialized solver for NNLS will be faster; however, we believe the scale of improvement is notable. Moreover, it is clear that our solution tends to be quite sparse while `MOSEK` always outputs a fully dense solution since their solver is based on interior point method. However, as y begins to lie in the cone of X, i.e., as μ increases, it is possible for our method to include more variables than necessary. This is shown in the case of $n = 100$ when μ is large since there are more than $\min(n, p)$ active variables.

5.5.2 Deconvolution of spike trains on carbon monoxide levels

We consider a positive spike-deconvolution model given in Li and Speed [2000]. Suppose we observe a response vector $y \in \mathbb{R}^n$ where each y_i are noisy observations of the mean function

$$f(t; \beta, \tau) := \sum_{k=1}^{p} \beta_k \phi(t - \tau_k).$$

The coefficients $\beta \in \mathbb{R}_+^p$ are assumed to be non-negative and $\phi(\cdot)$ is the mean-zero Gaussian density with a user-specified standard deviation. In practice, the user defines $\tau \in \mathbb{R}^p$ as a dense gridding of an interval of interest, and the goal is to identify $\beta \geq \mathbf{0}$ that best reconstructs y. The positive entries of β then correspond to the important spike points in τ. The feature matrix $X \in \mathbb{R}^{n \times p}$ is

[1] `https://github.com/JamesYang007/adelie`

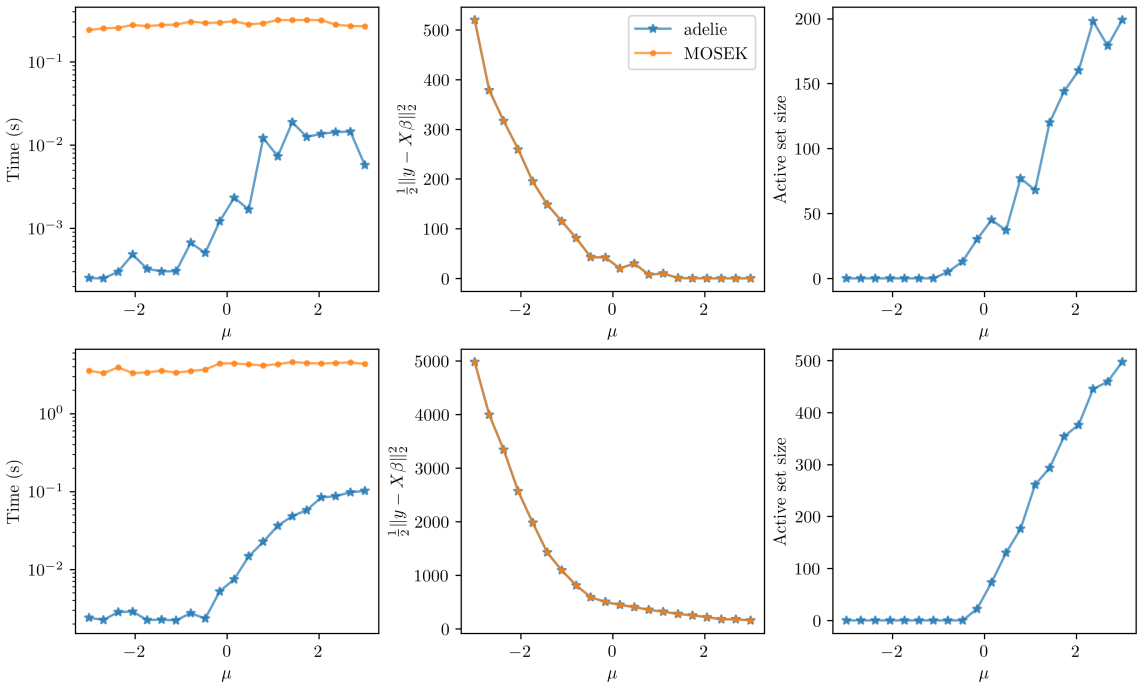


Figure 5.2: Plots of solving NNLS in the simulation setting for various values of μ using our method (`adelie`) and `MOSEK`. The top three plots correspond to $n = 100$ and the bottom three correspond to $n = 1000$. In all cases, $p = 10000$. The left column shows the time in seconds (smaller is better) and the middle column shows the objective for `adelie` and `MOSEK`. The right column shows the number of strictly positive entries in the solution obtained via our method. In general, our method obtains a solution as accurate as `MOSEK`'s with a speed-up factor anywhere from 10-1000x. Moreover, it is clear that our solution tends to be quite sparse while `MOSEK` always outputs a fully dense solution since their solver is based on interior point method. However, as y begins to lie in the cone of X, i.e., as μ increases, it is possible for our method to include more variables than necessary. This is shown in the case of $n = 100$ when μ is large since there are more than $\min(n, p)$ active variables.

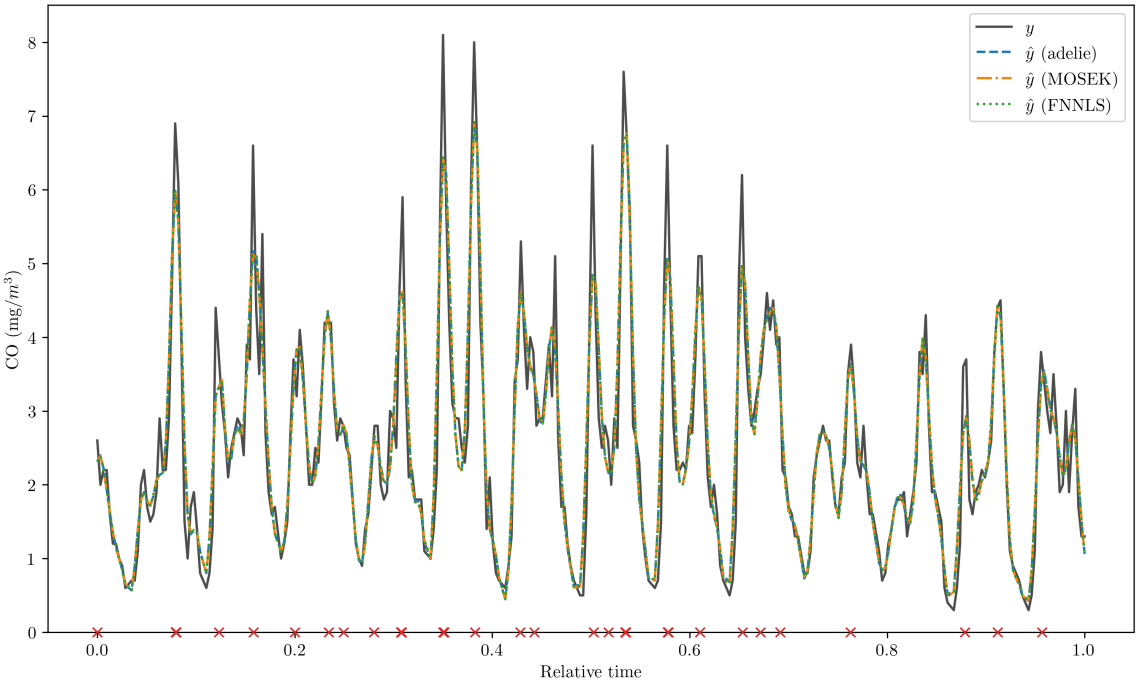


Figure 5.3: Plot of the carbon monoxide level y within the first two weeks of measurement and the NNLS predictions $\hat{y}$ using `adelie`, `MOSEK`, and FNNLS. We see that NNLS is able to reproduce y fairly well and identify the spikes. The red "x"'s mark the top 30 spike locations τ_k with the largest coefficients β_k using our method. These spike locations align well with the spikes observed in the data. All methods produce nearly identical fits.

then given by

$$X := \begin{bmatrix} \phi(t_1 - \tau_1) & \dots & \phi(t_1 - \tau_p) \\ \vdots & \ddots & \vdots \\ \phi(t_n - \tau_1) & \dots & \phi(t_1 - \tau_p) \end{bmatrix}$$

where $y_i = f(t_i; \beta, \tau) + \varepsilon_i$ for some noise ε_i. We use NNLS to solve for β by regressing y on X (given the non-negativity constraint).

We use the Air Quality data set from UCI [Vito, 2008] to apply the spike deconvolution method. This data set consists of hourly averaged responses from 5 metal oxide chemical sensors in an Air Quality Checmical Multisensor Device located in a significantly polluted area of an Italian city. We use only two weeks worth of data from March 10, 2004 to March 24, 2004. Our goal is to detect the spikes of carbon monoxide levels (mg/m^3). After removing missing data, there are $n = 304$ samples. The time stamps are put relative to the start date and normalized to the unit interval. We let $p = 1000$ and set $\tau \in \mathbb{R}^p$ equally spaced on the interval $[0, 1]$. The standard deviation in ϕ

is defined as 2 times the median (normalized) time difference in the n samples, as similarly done in Slawski and Hein [2013]. We solve the NNLS problem using our method, which takes 0.026 seconds. For a comparison, we also solve it using `MOSEK` and FNNLS [Bro and De Jong, 1997], which take 0.12 seconds and 1.17 seconds, respectively. In Figure 5.3, we show the response y along with the predictions $X\hat{\beta}$ where $\hat{\beta}$ is obtained via our method, `MOSEK`, and FNNLS. We see that the fits are nearly identical and reproduce y fairly well. Although all methods produce a feasible solution, our method and FNNLS are the only ones that output sparse solutions. FNNLS is the most accurate method, but it is the most time consuming. `MOSEK` is only slightly less accurate despite being about 10x faster, however, it outputs a dense solution, which is not as easily interpretable. We see our method as a bridge between the two methods; that is, we provide a sparse solution with a similar accuracy as `MOSEK` and at a fast speed. Using our method, we identify 225 spikes for this example.

5.6 Discussion

We develop a framework for understanding sparse solutions to any polyhedron constrained least squares problem. In particular, we establish a connection between popular regression methods such as NNLS, BVLS, SLS, ISLS, and the lasso that they all admit a locally unique sparse solution in high dimensions. We leverage this theory to derive an efficient coordinate descent based solver for NNLS, demonstrating that our solver tends to quickly find such sparse solutions in both simulated data settings and real data examples.

Our implementation is included in the Python package `adelie`, which is available under the MIT license from PyPI (`https://pypi.org/project/adelie`). The user can install the latest version by running `pip install adelie` in the terminal.

Appendix A

Group lasso and elastic net penalized regression

A.1 Proof of Theorem 2.3.1

Consider the optimization problem (2.5) with $\lambda > 0$. Suppose that $v_S = \mathbf{0}$ where S is given in (2.6). Without loss of generality, we may assume S^c is non-empty so that there exists at least one i such that $\Sigma_{ii} > 0$. Otherwise, by hypothesis, we must have $v = \mathbf{0}$ so that (2.5) reduces to minimizing $\|x\|_2$, which is uniquely minimized at $x^\star = \mathbf{0}$.

By the KKT conditions, the (finite) minimizer $x^\star$ must satisfy

$$\Sigma x - v + \lambda u = \mathbf{0}$$

where u is a sub-gradient of $\|\cdot\|_2$ at x. We first prove the identity (2.7). If $x^\star = \mathbf{0}$, then we must have that $\|v\|_2 \leq \lambda$. Otherwise, $u \equiv x/\|x\|_2$ so that

$$x = \left(\Sigma + \frac{\lambda}{\|x\|_2} I\right)^{-1} v \tag{A.1}$$

It remains to show that if $\|v\|_2 \leq \lambda$, then $x^\star = \mathbf{0}$, which will establish the identity (2.7).

In that endeavor, suppose $x^\star \neq \mathbf{0}$. Then, taking norms on both sides of (A.1), we have that $x^\star$ satisfies

$$\sum_{i=1}^{d} \frac{v_i^2}{(\Sigma_{ii}\|x\|_2 + \lambda)^2} = 1$$

Note that we then have that $\|v\|_2^2 \lambda^{-2} \geq 1$, or equivalently, $\|v\|_2 \geq \lambda$. There must exist some i

such that $v_i \neq 0$ and $\Sigma_{ii} > 0$. Otherwise, $v \equiv \mathbf{0}$ by hypothesis that $v_S = \mathbf{0}$, which contradicts $\|v\|_2 \geq \lambda > 0$. Consequently, for the element i such that $v_i \neq 0$ and $\Sigma_{ii} > 0$, we have the strict inequality

$$\frac{v_i^2}{(\Sigma_{ii}\|x\|_2 + \lambda)^2} < v_i^2 \lambda^{-2}$$

Hence, we have that $\|v\|_2 > \lambda$. This establishes the equivalence $x^\star = \mathbf{0} \iff \|v\|_2 \leq \lambda$.

A.2 Proof of Theorem 2.3.2

Consider any f satisfying the hypothesis of Theorem 2.3.2. Let $x^{(0)}$ be any initial point such that $f(x^{(0)}) \geq 0$. For a given $k \geq 0$, suppose $f(x^{(k)}) \geq 0$. Then, by convexity of f and the definition of $x^{(k+1)}$,

$$f(x^{(k+1)}) \geq f(x^{(k)}) + f'(x^{(k)})(x^{(k+1)} - x^{(k)}) = 0$$

By induction, we have that $f(x^{(k)}) \geq 0$ for every $k \geq 0$. Since f is decreasing and has a non-vanishing derivative, we must have that $f'(x) < 0$ for all x. In particular, $x^{(k+1)} \geq x^{(k)}$ so that it is an increasing sequence. Note that if $f(x^{(k)}) = 0$ for some k, then $x^{(k+j)} = x^{(k)}$ for all $j \geq 0$ so that the sequence converges in a trivial fashion to a root of f. Hence, without loss of generality, suppose $f(x^{(k)}) > 0$ for all $k \geq 0$. By the decreasing property of f, we must then have that $x^{(k)} \leq r$ for every $k \geq 0$. Since the sequence is increasing and bounded above, it converges to some value $x^{(\infty)}$. By continuity, taking $k \to \infty$ in (2.11),

$$x^{(\infty)} = x^{(\infty)} - \frac{f(x^{(\infty)})}{f'(x^{(\infty)})} \implies f(x^{(\infty)}) = 0$$

which shows that $x^{(\infty)}$ is a root of f.

A.3 Properties of the root-finding function

Consider φ as in (2.9). Recall by Theorem 2.3.1, we only consider φ when $\|v\|_2 > \lambda$. Similar to the proof of Theorem 2.3.1 in Appendix A.1, we have that there exists some i such that $v_i \neq 0$ and $\Sigma_{ii} > 0$. Then, differentiating φ, we get that

$$\varphi'(h) = -2\sum_{i=1}^{d} \frac{v_i^2 \Sigma_{ii}}{(\Sigma_{ii}h + \lambda)^3} < 0, \quad \varphi''(h) = 6\sum_{i=1}^{d} \frac{v_i^2 \Sigma_{ii}^2}{(\Sigma_{ii}h + \lambda)^4} > 0$$

This shows that φ is strictly decreasing and strictly convex.

Next, rewriting φ as

$$\varphi(h) = \sum_{i \notin S} \frac{v_i^2}{(\Sigma_{ii}h + \lambda)^2} - 1 \tag{A.2}$$

we see that $\lim_{h\to\infty} \varphi(h) < 0$. Note that $\varphi(0) = \|v\|_2^2 / \lambda^2 - 1 > 0$. Hence, a finite positive root uniquely exists for φ.

A.4 Derivation of lower and upper bounds on the root

We begin with the lower bound $h_\star$. Let $a(h), b(h) \in \mathbb{R}^d$ be defined by $a_k(h) := \Sigma_{kk}h + \lambda$ and $b_k(h) := |v_k| / a_k(h)$ for each $k = 1, \ldots, d$. Then, by Cauchy-Schwarz,

$$\|v\|_1 := \sum_{k=1}^{d} |v_k| = \sum_{k=1}^{d} a_k(h) b_k(h) \leq \|a(h)\|_2 \|b(h)\|_2$$

Hence, if $\|a(h)\|_2 \leq \|v\|_1$, then $\|b(h)\|_2 \geq 1$, or equivalently, $\varphi(h) \geq 0$. We see that $\|a(h)\|_2 \leq \|v\|_1$ if and only if

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

By the quadratic formula, we may choose the largest h that satisfies the above condition, which is given by (2.14).

The upper bound $h^\star$ is even easier. Since

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

we may set $h^\star$ to (2.15). Then, we immediately have that $\varphi(h^\star) \leq 0$.

Appendix B

Pivot rule for discarding predictors for group lasso

B.1 Pivot search derivations

We use the notation in Section 3.2.2 and consider pivot candidate x_j for some $2 \leq j \leq n$. Let $t \in \mathbb{R}^n$ be defined as $t_i := (x_j - x_i)\mathbb{1}_{x_i \leq x_j}$. Then, the coefficients for fitting linear regression with is given by

$$\hat{\beta}_0 = \bar{y} - \bar{t}\hat{\beta}_1, \quad \hat{\beta}_1 = \frac{(y - \overline{y}\mathbf{1})^\top t}{\|t - \bar{t}\mathbf{1}\|_2^2}$$

Hence, the MSE is given by

$$\begin{aligned}\left\|y - \hat{\beta}_0 - \hat{\beta}_1 t\right\|_2^2 &= C - 2\hat{\beta}_1(y - \overline{y}\mathbf{1})^\top(t - \bar{t}\mathbf{1}) + \hat{\beta}_1^2 \|t - \bar{t}\mathbf{1}\|_2^2 \\ &= C - \hat{\beta}_1^2 \|t - \bar{t}\mathbf{1}\|_2^2\end{aligned}$$

where C is a constant independent of the pivot choice x_j. Since we are only interested in comparing the different MSE values, it suffices to compute only the latter quantity.

We now expand the coefficient $\hat{\beta}_1$. First, consider the numerator $(y - \overline{y}\mathbf{1})^\top t$, which simplifies to

$$\begin{aligned}(y - \overline{y}\mathbf{1})^\top t &= \sum_{i=1}^{j}(y_i - \overline{y})(x_j - x_i) \\ &= x_j(\sum_{i=1}^{j} y_i - j\overline{y}) - \left(\sum_{i=1}^{j} x_i y_i - \overline{y}\sum_{i=1}^{j} x_i\right)\end{aligned}$$

Next, the denominator $\left\|t - \bar{t}\mathbf{1}\right\|_2^2$ simplifies to

$$\begin{aligned}
\left\|t - \bar{t}\mathbf{1}\right\|_2^2 &= \|t\|_2^2 - n\bar{t}^2 \\
&= \sum_{i=1}^{j}(x_j - x_i)^2 - n\bar{t}^2 \\
&= jx_j^2 - 2x_j\sum_{i=1}^{j}x_i + \sum_{i=1}^{j}x_i^2 - n\bar{t}^2 \\
\bar{t} &= \frac{1}{n}\sum_{i=1}^{j}(x_j - x_i) \\
&= \frac{j}{n}x_j - \frac{1}{n}\sum_{i=1}^{j}x_i
\end{aligned}$$

Hence, this shows that we only need to sequentially update the following quantities as we iterate over $j = 2, \ldots, n$:

$$\sum_{i=1}^{j}x_i, \quad \sum_{i=1}^{j}x_i^2, \quad \sum_{i=1}^{j}y_i, \quad \sum_{i=1}^{j}x_iy_i$$

and the MSE can otherwise be computed in $O(1)$ time per iteration with $O(1)$ space.

Appendix C

Polyhedron constrained group lasso and elastic net

C.1 Derivation of the hessian of the dual function

To derive the hessian $\nabla^2 g(\nu)$, it suffices to determine $Dx^\star(\nu)$, the Jacobian of $x^\star(\nu)$, since

$$\nabla^2 g(\nu) = -\begin{bmatrix} -I \\ I \end{bmatrix} ADx^\star(\nu).$$

By the KKT conditions, we have that

$$x^\star(\nu) \equiv \mathbf{0} \iff \left\| v - A^\top(\nu_u - \nu_\ell) \right\|_2 \leq \lambda.$$

Hence, on the boundary of the ellipsoid $\{\nu \in \mathbb{R}^{2m} : \left\| v - A^\top(\nu_u - \nu_\ell) \right\|_2 = \lambda\}$, $x^\star(\nu)$ may not be differentiable. However, in the interior of the ellipsoid, the hessian is exactly 0. We now derive the hessian outside the ellipsoid. We begin with the identity that for all ν outside the ellipse,

$$x^\star(\nu) = \|x^\star(\nu)\|_2 \left(\Sigma \|x^\star(\nu)\|_2 + \lambda I\right)^{-1} (v - A^\top(\nu_u - \nu_\ell)). \tag{C.1}$$

Now,

$$x^\star(\nu)_i = \left(1 - \frac{\lambda}{\Sigma_{ii} \|x^\star(\nu)\|_2 + \lambda}\right) \frac{v_i - A_{\cdot i}^\top(\nu_u - \nu_\ell)}{\Sigma_{ii}}.$$

Then, by the product rule and using (C.1),

$$\begin{aligned}\partial_j x_i^\star(\nu) &= \lambda \partial_j \left\|x^\star(\nu)\right\|_2 \frac{(v_i - A_{\cdot i}^\top(\nu_u - \nu_\ell))}{(\Sigma_{ii}\left\|x^\star(\nu)\right\|_2 + \lambda)^2} - \left\|x^\star(\nu)\right\|_2 \frac{(-1)^{\mathbb{1}_{j>m}} A_{ji}}{\Sigma_{ii}\left\|x^\star(\nu)\right\|_2 + \lambda} \\ &= \lambda \partial_j \left\|x^\star(\nu)\right\|_2 \frac{u_i^\star(\nu)}{\Sigma_{ii}\left\|x^\star(\nu)\right\|_2 + \lambda} - \left\|x^\star(\nu)\right\|_2 \frac{(-1)^{\mathbb{1}_{j>m}} A_{ji}}{\Sigma_{ii}\left\|x^\star(\nu)\right\|_2 + \lambda}\end{aligned}$$

where $u^\star(\nu) := x^\star(\nu)/\left\|x^\star(\nu)\right\|_2$. It remains to simplify $\partial_j \left\|x^\star(\nu)\right\|_2$. From (C.1), we also have the identity that

$$\left\|(\Sigma\left\|x^\star(\nu)\right\|_2 + \lambda I)^{-1}(v - A^\top(\nu_u - \nu_\ell))\right\|_2^2 = \|u^\star(\nu)\|_2^2 = 1.$$

Rewriting,

$$\sum_{i=1}^d \frac{(v_i - A_{\cdot i}^\top(\nu_u - \nu_\ell))^2}{(\Sigma_{ii}\left\|x^\star(\nu)\right\|_2 + \lambda)^2} = 1.$$

Differentiating, we get that

$$2\sum_{i=1}^d \left[\frac{(v_i - A_{\cdot i}^\top(\nu_u - \nu_\ell))(-1)^{\mathbb{1}_{j>m}} A_{ji}}{(\Sigma_{ii}\left\|x^\star(\nu)\right\|_2 + \lambda)^2} - \frac{(v_i - A_{\cdot i}^\top(\nu_u - \nu_\ell))^2 \Sigma_{ii}\partial_j \left\|x^\star(\nu)\right\|_2}{(\Sigma_{ii}\left\|x^\star(\nu)\right\|_2 + \lambda)^3}\right] = 0,$$

which readily gives us a formula for $\partial_j \left\|x^\star(\nu)\right\|_2$. Indeed,

$$\nabla \left\|x^\star(\nu)\right\|_2 = -\left\|x^\star(\nu)\right\|_2 \kappa(\nu) \begin{bmatrix} -I \\ I \end{bmatrix} A\alpha(\nu).$$

where $\kappa(\nu)$ and $\alpha(\nu)$ are given by (4.3) and (4.4). Substituting into $\nabla^2 g(\nu)$, we immediately get the desired form.

Appendix D

Polyhedron constrained least squares

D.1 Background

Lemma D.1.1 (Interior of a polyhedron)**.** *Let $\mathcal{P} \subseteq \mathbb{R}^p$ be a polyhedron represented by A and b. Then, the interior of $\mathcal{P}$ is given by* $\operatorname{int}(\mathcal{P}) = \{x \in \mathcal{P} : Ax < b\}$.

Proof. Without loss of generality, assume $\mathcal{P}$ is non-empty. Suppose $x \in \operatorname{int}(\mathcal{P})$. Then, there exists $B_r(x)$, a ball of radius r centered at x, for some $r > 0$ such that $B_r(x) \subseteq \operatorname{int}(\mathcal{P})$. Suppose x binds a constraint a_i for some $i \in [m]$. Without loss of generality, we may assume $a_i \neq \mathbf{0}$ since it may be removed from the constraints without changing $\mathcal{P}$. Consider $w := x + \epsilon a_i$ for any $\epsilon > 0$. Then, $a_i^\top w = a_i^\top x + \epsilon \|a_i\|_2^2 = b_i + \epsilon \|a_i\|_2^2 > b_i$ so that $w \notin \mathcal{P}$. In particular, if $\epsilon < r/\|a_i\|_2$, then $w \in B_r(x)$, which is a contradiction.

Conversely, suppose $x \in \mathbb{R}^p$ such that $Ax < b$. Consider φ defined by

$$\varphi(t) := \max_{i \in [m]} \max_{v : \|v\|_2 = 1} \left\{ a_i^\top (x + tv) - b_i \right\} \tag{D.1}$$

$$= \max_{i \in [m]} \left\{ a_i^\top x + t \|a_i\|_2 - b_i \right\} \tag{D.2}$$

which identifies the constraint closest to the boundary on the ball centered at x with radius t. Note that φ is continuous, $\varphi(0) < 0$, and $\varphi(t) \to \infty$ as $t \to \infty$. Hence, there exists $t^\star > 0$ such that $\varphi(t^\star) = 0$. Note also that φ is strictly increasing, so this implies that for any $t \in [0, t^\star)$, $\varphi(t) < 0$. In other words, $B_t(x) \subseteq \mathcal{P}$ so that x is an interior point. □

Lemma D.1.2. *Let $\mathcal{P} \subseteq \mathbb{R}^p$ be a non-empty polyhedron represented by $A \in \mathbb{R}^{m \times p}$ and $b \in \mathbb{R}^m$. Define $S := \left\{ i \in [m] : a_i^\top x = b_i, \forall x \in \mathcal{P} \right\}$ as the set of constraints that are always binding. Then,*

the polyhedron represented by A_{-S} and b_{-S} has a non-empty interior and the affine space $\{x \in \mathbb{R}^p : A_S x = b_S\}$ is unique. Consequently, any non-empty polyhedron can be decomposed as an intersection of a polyhedron with a non-empty interior and a unique affine space.

Proof. We first show that the polyhedron represented by A_{-S} and b_{-S} has a non-empty interior. If $\mathcal{P}$ has a non-empty interior, we are done. Suppose $\mathcal{P}$ has an empty interior. We claim that there exists a constraint $a_{i^\star}$ such that $a_{i^\star}^\top x = b_{i^\star}$ for all $x \in \mathcal{P}$. Suppose not so that for every constraint a_i, there exists $x_i \in \mathcal{P}$ such that $a_i^\top x_i < b_i$. We show that there exists a convex combination of $\{x_k\}_{k=1}^m$ such that $Ax < b$, which will establish a contradiction by Lemma D.1.1. Indeed, for any $\theta \in \mathbb{R}^m_{++}$ strictly positive and $\sum_k \theta_k = 1$,

$$a_i^\top \sum_k \theta_k x_k < b_i, \quad \forall i \in [m] \tag{D.3}$$

since for every $i \in [m]$, $a_i^\top x_i < b_i$ while $a_i^\top x_k \leq b_i$ for all $k \neq i$. This completes the proof for showing there exists a constraint $a_{i^\star}$ such that $a_{i^\star}^\top x = b_{i^\star}$ for all $x \in \mathcal{P}$. Once we identify the constraint $a_{i^\star}$, we remove it and repeat the procedure on the polyhedron $\mathcal{P}' := \{x \in \mathbb{R}^p : A_{-i^\star} x \leq b_{-i^\star}\}$ until $\mathcal{P}'$ has a non-empty interior. Clearly, this process must terminate in finite steps.

We now show that the affine space determined by S is unique. Suppose there exists a different affine space that contains $\mathcal{P}$. Then, $\mathcal{P}$ must lie in a strictly lower-dimensional subspace, which implies that $\mathcal{P}$ has an empty interior. This contradicts the fact that we have terminated our procedure when we reached $\mathcal{P}'$ with a non-empty interior. □

Lemma D.1.3. *Let $Q \in \mathbb{R}^{n \times p}$ be any matrix and $\mathcal{P} \subseteq \mathbb{R}^p$ be a polyhedron. Then,* $\mathrm{Hull}(Q, \mathcal{P})$ *is a polyhedron.*

Proof. Without loss of generality, assume $\mathcal{P}$ is non-empty. By the polyhedron decomposition theorem, $\mathcal{P}$ can be represented as a Minkowski sum of a cone and a polytope. Let $\mathcal{C} = \mathrm{Cone}(B_\mathcal{C})$ and $\mathcal{V} = \mathrm{Conv}(B_\mathcal{V})$ denote the cone and the polytope generated by $B_\mathcal{C}$ and $B_\mathcal{V}$, respectively. Then, $Q(\mathcal{P}) = \mathrm{Cone}(QB_\mathcal{C}) + \mathrm{Conv}(QB_\mathcal{V})$. By the polyhedron decomposition theorem again, $Q(\mathcal{P})$ is a polyhedron. □

Lemma D.1.4. *Let $\mathcal{P} \subseteq \mathbb{R}^p$ be a polyhedron represented by $A \in \mathbb{R}^{m \times p}$ and $b \in \mathbb{R}^m$ with a non-empty interior. Suppose y is any element in the interior of $\mathcal{P}$ and $v \in \mathbb{R}^p$ any vector. Then, $v \notin \mathrm{Null}(A)$ if and only if there exists a point $w = y + tv$ for some $t \neq 0$ such that w lies in $\mathcal{P}$ and binds at least one constraint of A.*

Proof. Since y is in the interior of $\mathcal{P}$, we must have that $Ay < b$ by Lemma D.1.1.

Suppose $v \notin \mathrm{Null}(A)$ so that $Av \neq \mathbf{0}$. Then, there exists $i \in [m]$ such that $a_i^\top v \neq 0$. For any

constraint a_i not orthogonal to v,

$$a_i^\top (y + tv) = b_i \iff t = \frac{b_i - a_i^\top y}{a_i^\top v}. \tag{D.4}$$

Label the solution for t in (D.4) as $t_i^\star$ for each i such that $a_i^\top v \neq 0$. Then, we simply choose the $t_i^\star$ that is smallest in magnitude, that is,

$$i^\star := \underset{i: a_i^\top v \neq 0}{\operatorname{argmin}} |t_i^\star|, \quad t^\star := t_{i^\star}^\star. \tag{D.5}$$

Note that since each $t_i^\star \neq 0$, we also have that $t^\star \neq 0$. With this choice of $t^\star$, we have the guarantee that $w := y + t^\star v$ binds the $i^\star$th constraint and is the first binding constraint so that w remains in $\mathcal{P}$.

Conversely, suppose $v \in \text{Null}(A)$ so that $Av = \mathbf{0}$. Then, for any point $w = y + tv$ for any $t \in \mathbb{R}$, we have that $Aw = Ay < b$. Hence, every such point w does not bind any constraint. □

Lemma D.1.5. *Let $\mathcal{P} \subseteq \mathbb{R}^p$ be a non-empty polyhedron represented by $A \in \mathbb{R}^{m \times p}$ and $b \in \mathbb{R}^m$ with S as its maximal affine constraints. Let $\xi \in \mathbb{R}^p$ be any vector satisfying $A_S \xi = b_S$ and $V \in \mathbb{R}^{p \times \dim(\mathcal{P})}$ be any full column rank matrix such that $A_S V \equiv \mathbf{0}$. Define the following quantities*

$$A' := A_{-S} V \in \mathbb{R}^{(m-|S|) \times \dim(\mathcal{P})}, \quad b' := b_{-S} - A_{-S} \xi \in \mathbb{R}^{m-|S|} \tag{D.6}$$

and let $\mathcal{P}' \subseteq \mathbb{R}^{\dim(\mathcal{P})}$ be the polyhedron represented by (D.6)*. Then, every $x \in \mathcal{P}$ can be represented uniquely as $x = \xi + Vx'$ with $x' \in \mathcal{P}'$. Moreover, $\mathcal{P}'$ has a non-empty interior.*

Proof. By hypothesis, $\mathcal{P} \equiv \{x \in \mathbb{R}^p : A_{-S} x \leq b_{-S},\, A_S x = b_S\}$. First, it is a standard linear algebra fact that any $x \in \mathbb{R}^p$ such that $A_S x = b_S$ can be uniquely identified with the low-rank representation $x = \xi + Vx'$ for some $x' \in \mathbb{R}^{\dim(\mathcal{P})}$. Then, we have the first claim by simple algebra that

$$A_{-S} x - b_{-S} = A' x' - b'. \tag{D.7}$$

We now show that $\mathcal{P}'$ has a non-empty interior. Since $\{x \in \mathbb{R}^p : A_{-S} x \leq b_{-S}\}$ has a non-empty interior by Lemma D.1.2, there exists $x \in \mathcal{P}$ such that $A_{-S} x < b_{-S}$ by Lemma D.1.1. By (D.7), we must have that $A' x' < b'$. Another application of Lemma D.1.1 shows that $\text{int}(\mathcal{P}')$ is non-empty. □

D.2 Proof of Lemma 5.3.1

Proof. Without loss of generality, assume every constraint is non-zero since otherwise we may remove these constraints without changing $\mathcal{P}$ or $\text{Null}(A)$. If $p \leq n$, the result is trivial. Hence, we assume without loss of generality that $p > n$.

We first prove the theorem when $p = n + 1$. Suppose x is in the $\mathcal{P}$-hull of Q so that $x = Qy$ for some $y \in \mathcal{P}$. If y binds at least one constraint of A, we are done. Suppose then y binds no constraint of A so that it lies in the interior of $\mathcal{P}$ by Lemma D.1.1. Since there are more columns than rows in Q, we necessarily have that $\text{Null}(Q)$ is non-trivial. Hence, there exists $v \neq \mathbf{0}$ such that $Qv = \mathbf{0}$. Since $v \in \text{Null}(Q) \setminus \{\mathbf{0}\} \subseteq \text{Null}(A)^c$, we have by Lemma D.1.4 that there exists $w = y + tv$ for some $t \neq 0$ such that w lies in $\mathcal{P}$ and binds at least one constraint of A. Finally, $Qw = x$ by construction. This proves the theorem when $p = n + 1$.

Now, suppose the theorem holds for $p > n$. We now prove it holds for $p + 1$. Suppose x is a $\mathcal{P}$-combination of $Q = \{q_i\}_{i=1}^{p+1}$. Consider the embedding

$$\forall i \in [p+1], \quad u_i := \begin{bmatrix} q_i \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^p, \quad v := \begin{bmatrix} x \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^p \tag{D.8}$$

where we pad our original points with zeros until the length is p. Define $U := \{u_i\}_{i=1}^{p+1}$. Then, v is a $\mathcal{P}$-combination of U. By the preceding argument, noting that $\text{Null}(Q) \equiv \text{Null}(U)$, v can then be written as a $\mathcal{P}$-combination of U with an element w that binds at least one constraint of A. Finally, we conclude that x can be written as a $\mathcal{P}$-combination of Q with (the same) w that binds at least one constraint of A. Without loss of generality, suppose the binding constraint is the first constraint a_1. Consider the polyhedron $\left\{z \in \mathbb{R}^{p+1} : A_{-1}z \leq b_{-1},\, a_1^\top z = b_1\right\}$ whose dimension is then p (since $a_1 \neq \mathbf{0}$). By Lemma D.1.5, it can be represented with the polyhedron $\mathcal{P}'$ defined by A' and b' in (D.6) with a non-empty interior. Let V and ξ be as in Lemma D.1.5 and define $Q' := QV$ and $x' := x - Q\xi$. Note that $x' \in \text{Hull}(Q', \mathcal{P}')$. We will apply the induction hypothesis on Q', $\mathcal{P}'$, and x'. Indeed, $\text{Null}(Q') \setminus \{\mathbf{0}\} \subseteq \text{Null}(A')^c$ since for any $y \in \text{Null}(Q') \setminus \{\mathbf{0}\}$, $Vy \in \text{Null}(Q)$, which implies that $AVy \neq \mathbf{0}$ by hypothesis. Since $a_1^\top V = \mathbf{0}^\top$ by definition of V, we have that $A'y = A_{-1}Vy \neq \mathbf{0}$ so $y \in \text{Null}(A')^c$. Hence, the assumptions of the theorem are satisfied. By the induction hypothesis, x' is a $\mathcal{P}'$-combination of Q' with an element w' that binds at least $\min(\text{rank}(A'), p-n)$ constraints of A' that forms a full row rank sub-matrix of A'. Letting $w := \xi + Vw'$ and noting that

$$A_{-1}w - b_{-1} = A'w' - b', \tag{D.9}$$

x is a $\mathcal{P}$-combination of Q with w that binds at least $\min(\text{rank}(A'), p-n)$ constraints of A_{-1}, which we denote as $T \subseteq [m-1]$. Hence, including the first binding constraint, w binds at least $\min(\text{rank}(A') + 1, p + 1 - n)$ constraints of A. It remains to show that $\text{rank}(A) \equiv \text{rank}(A') + 1$ and that appending a_1 as a row to $(A_{-1})_T$ keeps it full row rank. To show the first claim, observe that

$$A \begin{bmatrix} a_1 & V \end{bmatrix} = \begin{bmatrix} Aa_1 & AV \end{bmatrix}. \tag{D.10}$$

By definition of V, $\begin{bmatrix} a_1 & V \end{bmatrix}$ is invertible so that the rank of the left side of (D.10) is $\text{rank}(A)$.

Note that $Aa_1 \notin \text{span}(AV)$ since the first row of AV is $a_1^\top V \equiv \mathbf{0}^\top$ while that of Aa_1 is $\|a_1\|_2^2 \neq 0$. Again using that $a_1^\top V = \mathbf{0}^\top$, $\text{rank}(AV) = \text{rank}(A_{-1}V) = \text{rank}(A')$. This shows that $\text{rank}(A) = \text{rank}(A') + 1$. Next, we show the second claim. It suffices to show that a_1 is not in the row space of $(A_{-1})_T$. Otherwise, there exists $\gamma \neq \mathbf{0} \in \mathbb{R}^{|T|}$ such that $a_1 = (A_{-1})_T^\top \gamma$. Then,

$$\mathbf{0} = V^\top a_1 = V^\top (A_{-1})_T^\top \gamma = ((A_{-1})_T V)^\top \gamma = (A'_T)^\top \gamma \implies \gamma = \mathbf{0}, \tag{D.11}$$

which is a contradiction. This shows that including a_1 as another row in $(A_{-1})_T$ keeps it full row rank. By induction, this concludes the proof.

□

D.3 Proof of Theorem 5.3.2

Proof. First, consider the setting of Lemma 5.3.1. We argue that the conclusion can be improved from $\min(\text{rank}(A), (p-n)_+)$ to $\min(\text{rank}(A), (p-\text{rank}(Q))_+)$. We first perform a QR-decomposition on Q to get $Q = \tilde{Q}\tilde{R}$ with $\tilde{Q} \in \mathbb{R}^{n\times r}$, $\tilde{R} \in \mathbb{R}^{r\times p}$, and $r := \text{rank}(Q)$. Note that $\tilde{Q}$ has orthogonal columns so that $\text{Null}(Q) \equiv \text{Null}(\tilde{R})$. This shows that $\tilde{R}$ satisfies the assumptions of Lemma 5.3.1. Now, if $x \in \text{Hull}(Q, \mathcal{P})$, there exists $w \in \mathcal{P}$ such that $x = Qw = \tilde{Q}\tilde{R}w$. We then apply Lemma 5.3.1 for $\tilde{R}$, $\mathcal{P}$, and $\tilde{x} := \tilde{R}w$ to get that $\tilde{x}$ is a $\mathcal{P}$-combination of $\tilde{R}$ with an element $\tilde{w} \in \mathcal{P}$ that binds at least $\min(\text{rank}(A), (p-r)_+)$ constraints of A that form a full row rank sub-matrix. Since $\tilde{R}w = \tilde{R}\tilde{w}$, this immediately shows that $x = \tilde{Q}\tilde{R}w = \tilde{Q}\tilde{R}\tilde{w} = Q\tilde{w}$. Hence, we have our improved result.

If $\mathcal{P}$ has a non-empty interior, then Lemma 5.3.1 gives us the desired conclusion noting that $\dim(\mathcal{P}) \equiv p$. Suppose $\mathcal{P}$ has an empty interior. Then, $\text{rank}(A_S) \neq 0$ so that Lemma D.1.5 reduces our representation of $\mathcal{P}$ to a lower-dimensional polyhedron $\mathcal{P}'$ represented with A' and b' given in (D.6). Moreover $\mathcal{P}'$ has a non-empty interior. Then, with the following quantities

$$Q' := QV, \quad x' := x - Q\xi \tag{D.12}$$

using ξ as in Lemma D.1.5 (and V as in the hypothesis), we have

$$x \in \text{Hull}(Q, \mathcal{P}) \iff x' \in \text{Hull}(Q', \mathcal{P}'). \tag{D.13}$$

Note that the conditions of Lemma 5.3.1 are satisfied trivially for Q' and $\mathcal{P}'$ by definition. Moreover, $\dim(\mathcal{P}') \equiv \dim(\mathcal{P})$ by construction. Applying Lemma 5.3.1 for Q', $\mathcal{P}'$, and x', we have that x' is a $\mathcal{P}'$-combination of Q' with an element $w' \in \mathcal{P}'$ that binds at least $\min(\text{rank}(A_{-S}V), (\dim(\mathcal{P}) - \text{rank}(Q'))_+)$ constraints of A' that form a full row rank sub-matrix of A'. Then, letting $w := \xi + Vw'$ and using (D.7), w binds at least $\min(\text{rank}(A_{-S}V), (\dim(\mathcal{P}) - \text{rank}(Q'))_+)$ constraints of A_{-S}, which we denote as $T \subseteq [m - |S|]$. It remains to show that $\text{rank}(A_{-S}V) = \text{rank}(A) - \text{rank}(A_S)$ and that

the binding constraints form a full row rank sub-matrix of A_{-S}. We first show the first claim. Let $V^\perp \in \mathbb{R}^{p\times \mathrm{rank}(A_S)}$ denote a set of orthogonal vectors that span the row space of A_S. Then, $A[V^\perp \quad V]$ has the same rank as A and it is clear that $\mathrm{rank}(AV^\perp) = \mathrm{rank}(A_S)$. Moreover, since $A_S V = \mathbf{0}$, we have $\mathrm{rank}(AV) = \mathrm{rank}(A_{-S}V)$. Finally, every column of $AV^\perp$ is not in $\mathrm{span}(AV)$. Otherwise, $A_S V^\perp = A_S V M = \mathbf{0}$ for some matrix M, which is a contradiction since every column of $V^\perp$ lies in $\mathrm{span}(A_S)$. This proves that $\mathrm{rank}(A_{-S}V) = \mathrm{rank}(A) - \mathrm{rank}(A_S)$. Next, we show that $(A_{-S})_T$ is a full row rank sub-matrix. Otherwise, there exists $\gamma \neq \mathbf{0} \in \mathbb{R}^{|T|}$ such that

$$(A_{-S})_T^\top \gamma = \mathbf{0} \implies \mathbf{0} = V^\top (A_{-S})_T^\top \gamma = (A'_T)^\top \gamma \implies \gamma = \mathbf{0}, \tag{D.14}$$

which is a contradiction. This completes the proof. □

D.4 Proof of Theorem 5.3.3

Proof. We may assume without loss of generality $\mathcal{P}$ has a non-empty interior since otherwise we may first restrict to the lower-dimensional space removing the maximal affine constraints by Lemma D.1.5. Hence, we assume $p \equiv \dim(\mathcal{P})$. We may further assume Q is full row rank so that $n \equiv \mathrm{rank}(Q)$. We are now in the setting of Lemma 5.3.1 where Q is full row rank and $\mathcal{P}$ has a non-empty interior.

Consider the proof of Lemma 5.3.1. Recall that the proof procedure is to repeatedly find directions $v \in \mathrm{Null}(Q)$ such that $w = y + tv$ binds a constraint (where $x = Qy$), say a_1. Let $V \in \mathbb{R}^{p\times(p-1)}$ be as in Lemma D.1.5 that reduces the polyhedron by one dimension, namely, restricting to the affine space $\left\{w \in \mathbb{R}^p : a_1^\top w = b_1\right\}$. We claim that $\mathrm{rank}(QV) = \mathrm{rank}(Q)$. Indeed, it is equivalent to show that $\dim(\mathrm{Null}(QV)) = \dim(\mathrm{Null}(Q)) - 1$. This is easily seen by the fact that since $a_1^\top v \neq 0$ by Lemma D.1.4 and $V^\top a_1 = \mathbf{0}$, we must have $v \notin \mathrm{span}(V)$. Since the rank is preserved as we apply induction, we see that eventually Q will be reduced to an invertible matrix, in which case, the representation for x is unique. □

Bibliography

Alberto Abadie. Using synthetic controls: Feasibility, data requirements, and methodological aspects. *Journal of Economic Literature*, 59(2):391–425, June 2021. doi: 10.1257/jel.20191450. URL https://www.aeaweb.org/articles?id=10.1257/jel.20191450.

MOSEK ApS. *MOSEK Optimizer API for Python 10.2.5*, 2024. URL https://docs.mosek.com/latest/pythonapi/index.html.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017. URL http://jmlr.org/papers/v18/14-546.html.

Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Lagrangian Duality and Saddle Point Optimality Conditions*. 2006. ISBN 9780471787778. doi: https://doi.org/10.1002/0471787779.ch6. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/0471787779.ch6.

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.

Aharon Ben-Tal and Arkadi Nemirovski. Lecture notes optimization iii convex analysis nonlinear programming theory nonlinear programming algorithms. https://www2.isye.gatech.edu/~nemirovs/OPTIIILN2023Spring.pdf, 2023.

Alexander G. Bick, Ginger A. Metcalf, Kelsey R. Mayo, Lee Lichtenstein, Shimon Rura, Robert J. Carroll, Anjene Musick, Jodell E. Linder, I. King Jordan, Shashwat Deepali Nagar, Shivam Sharma, Robert Meller, Melissa Basford, Eric Boerwinkle, Mine S. Cicek, Kimberly F. Doheny, Evan E. Eichler, Stacey Gabriel, Richard A. Gibbs, David Glazer, Paul A. Harris, Gail P. Jarvik, Anthony Philippakis, Heidi L. Rehm, Dan M. Roden, Stephen N. Thibodeau, Scott Topper, Ashley L. Blegen, Samantha J. Wirkus, Victoria A. Wagner, Jeffrey G. Meyer, Mine S. Cicek, Donna M. Muzny, Eric Venner, Michelle Z. Mawhinney, Sean M. L. Griffith, Elvin Hsu, Hua Ling,

Marcia K. Adams, Kimberly Walker, Jianhong Hu, Harsha Doddapaneni, Christie L. Kovar, Mullai Murugan, Shannon Dugan, Ziad Khan, Eric Boerwinkle, Niall J. Lennon, Christina Austin-Tse, Eric Banks, Michael Gatzen, Namrata Gupta, Emma Henricks, Katie Larsson, Sheli McDonough, Steven M. Harrison, Christopher Kachulis, Matthew S. Lebo, Cynthia L. Neben, Marcie Steeves, Alicia Y. Zhou, Joshua D. Smith, Christian D. Frazar, Colleen P. Davis, Karynne E. Patterson, Marsha M. Wheeler, Sean McGee, Christina M. Lockwood, Brian H. Shirts, Colin C. Pritchard, Mitzi L. Murray, Valeria Vasta, Dru Leistritz, Matthew A. Richardson, Jillian G. Buchan, Aparna Radhakrishnan, Niklas Krumm, Brenna W. Ehmen, Sophie Schwartz, M. Morgan T. Aster, Kristian Cibulskis, Andrea Haessly, Rebecca Asch, Aurora Cremer, Kylee Degatano, Akum Shergill, Laura D. Gauthier, Samuel K. Lee, Aaron Hatcher, George B. Grant, Genevieve R. Brandt, Miguel Covarrubias, Eric Banks, Ashley Able, Ashley E. Green, Robert J. Carroll, Jennifer Zhang, Henry R. Condon, Yuanyuan Wang, Moira K. Dillon, C. H. Albach, Wail Baalawi, Seung Hoan Choi, Xin Wang, Elisabeth A. Rosenthal, Andrea H. Ramirez, Sokny Lim, Siddhartha Nambiar, Bradley Ozenberger, Anastasia L. Wise, Chris Lunt, Geoffrey S. Ginsburg, and Joshua C. Denny. Genomic data in the all of us research program. *Nature*, 627(8003):340–346, February 2024. ISSN 1476-4687. doi: 10.1038/s41586-023-06957-x. URL `http://dx.doi.org/10.1038/s41586-023-06957-x`.

Antoine Bonnefoy, Valentin Emiya, Liva Ralaivola, and Rémi Gribonval. A dynamic screening principle for the lasso. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pages 6–10, 2014.

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

Patrick Breheny and Jian Huang. Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Stat. Comput.*, 25(2):173–187, March 2015.

Richard P Brent. *Algorithms for Minimization without Derivatives*. Courier Corporation, 2013.

Rasmus Bro and Sijmen De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, 1997. doi: https://doi.org/10.1002/(SICI)1099-128X(199709/10)11:5⟨393::AID-CEM483⟩3.0.CO;2-L. URL `https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291099-128X%28199709/10%2911%3A5%3C393%3A%3AAID-CEM483%3E3.0.CO%3B2-L`.

Brian L. Browning, Xiaowen Tian, Ying Zhou, and Sharon R. Browning. Fast two-stage phasing of large-scale sequence data. *The American Journal of Human Genetics*, 108(10):1880–1890, October 2021. ISSN 0002-9297. doi: 10.1016/j.ajhg.2021.08.005. URL `http://dx.doi.org/10.1016/j.ajhg.2021.08.005`.

Alfred M. Bruckstein, Michael Elad, and Michael Zibulevsky. On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. *IEEE Transactions on Information Theory*, 54(11):4813–4820, 2008. doi: 10.1109/TIT.2008.929920.

Hung-Hsu Chou, Johannes Maly, and Claudio Mayrink Verdun. Non-negative least squares via overparametrization, 2022. URL `https://arxiv.org/abs/2207.08437`.

Benjamin B Chu, Eric M Sobel, Rory Wasiolek, Seyoon Ko, Janet S Sinsheimer, Hua Zhou, and Kenneth Lange. A fast data-driven method for genotype imputation, phasing and local ancestry inference: Mendelimpute.jl. *Bioinformatics*, 37(24):4756–4763, July 2021. ISSN 1367-4811. doi: 10.1093/bioinformatics/btab489. URL `http://dx.doi.org/10.1093/bioinformatics/btab489`.

Liang Dai and Kristiaan Pelckmans. An ellipsoid based, two-stage screening test for bpdn. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 654–658, 2012.

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

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Jelena Diakonikolas, Chenghui Li, Swati Padmanabhan, and Chaobing Song. A fast scale-invariant algorithm for non-negative least squares with non-negative data. *Advances in Neural Information Processing Systems*, 35:6264–6277, 2022.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2), April 2004. doi: 10.1214/009053604000000067.

Tolga Ergen and Mert Pilanci. The convex landscape of neural networks: Characterizing global optima and stationary points via lasso models, 2023. URL `https://arxiv.org/abs/2312.12657`.

Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 70(5):849–911, October 2008. doi: 10.1111/j.1467-9868.2008.00674.x. URL `https://doi.org/10.1111/j.1467-9868.2008.00674.x`.

Jianqing Fan, Richard Samworth, and Yichao Wu. Ultrahigh dimensional feature selection: Beyond the linear model. *J. Mach. Learn. Res.*, 10:2013–2038, dec 2009. ISSN 1532-4435.

Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Mind the duality gap: safer rules for the lasso, 2015.

Vojtěch Franc, Václav Hlaváč, and Mirko Navara. Sequential coordinate-wise algorithm for the non-negative least squares problem. In *International Conference on Computer Analysis of Images and Patterns*, pages 407–414. Springer, 2005.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010. ISSN 1548-7660. doi: 10.18637/jss.v033.i01. URL `http://dx.doi.org/10.18637/jss.v033.i01`.

Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems, 2011.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286 (5439):531–537, October 1999. ISSN 1095-9203. doi: 10.1126/science.286.5439.531. URL `http://dx.doi.org/10.1126/science.286.5439.531`.

David Harrison and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, March 1978. ISSN 0095-0696. doi: 10.1016/0095-0696(78)90006-2. URL `http://dx.doi.org/10.1016/0095-0696(78)90006-2`.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

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

Zihuai He, Linxi Liu, Michael E. Belloy, Yann Le Guen, Aaron Sossin, Xiaoxia Liu, Xinran Qi, Shiyang Ma, Prashnna K. Gyawali, Tony Wyss-Coray, Hua Tang, Chiara Sabatti, Emmanuel Candès, Michael D. Greicius, and Iuliana Ionita-Laza. Ghostknockoff inference empowers identification of putative causal variants in genome-wide association studies. *Nature Communications*, 13(1), November 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-34932-z. URL `http://dx.doi.org/10.1038/s41467-022-34932-z`.

Zihuai He, Benjamin Chu, James Yang, Jiaqi Gu, Zhaomeng Chen, Linxi Liu, Tim Morrison, Michael E. Belloy, Xinran Qi, Nima Hejazi, Maya Mathur, Yann Le Guen, Hua Tang, Trevor Hastie, Iuliana Ionita-laza, Chiara Sabatti, and Emmanuel Candès. In silico identification of putative causal genetic variants. March 2024. doi: 10.1101/2024.02.28.582621. URL `http://dx.doi.org/10.1101/2024.02.28.582621`.

Nathaniel E. Helwig. *grpnet: Group Elastic Net Regularized GLMs and GAMs*, 2024. URL `https://CRAN.R-project.org/package=grpnet`. R package version 0.3.

Joaqun Torres-Sospedra, Raul Montoliu, Adolfo Martnez-Us, Tomar Arnau, Joan Avariento. UJIIndoorLoc, 2014.

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.

Xin-Bing Kong, Zhi Liu, Yuan Yao, and Wang Zhou. Sure screening by ranking the canonical correlations. *TEST*, 26(1):46–70, June 2016. doi: 10.1007/s11749-016-0497-z. URL `https://doi.org/10.1007/s11749-016-0497-z`.

B. Krishnapuram, L. Carin, M.A.T. Figueiredo, and A.J. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, June 2005. ISSN 2160-9292. doi: 10.1109/tpami.2005.127. URL `http://dx.doi.org/10.1109/TPAMI.2005.127`.

Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, January 1995. ISBN 9781611971217. doi: 10.1137/1.9781611971217. URL `http://dx.doi.org/10.1137/1.9781611971217`.

Jason D. Lee, Yuekai Sun, and Michael A. Saunders. Proximal newton-type methods for minimizing composite functions. *SIAM Journal on Optimization*, 24(3):1420–1443, January 2014. ISSN 1095-7189. doi: 10.1137/130921428. URL `http://dx.doi.org/10.1137/130921428`.

Seunghak Lee, Nico Gornitz, Eric P. Xing, David Heckerman, and Christoph Lippert. Ensembles of lasso screening rules. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12): 2841–2852, December 2018. doi: 10.1109/tpami.2017.2765321. URL `https://doi.org/10.1109/tpami.2017.2765321`.

Lihua Lei and Timothy Sudijono. Inference for synthetic controls via refined placebo tests, 2024. URL `https://arxiv.org/abs/2401.07152`.

Lei Li and Terence P. Speed. Parametric deconvolution of positive spike trains. *The Annals of Statistics*, 28(5):1279 – 1301, 2000. doi: 10.1214/aos/1015957394. URL `https://doi.org/10.1214/aos/1015957394`.

Ruilin Li, Christopher Chang, Johanne M Justesen, Yosuke Tanigawa, Junyang Qian, Trevor Hastie, Manuel A Rivas, and Robert Tibshirani. Fast lasso method for large-scale and ultrahigh-dimensional cox model with applications to uk biobank. *Biostatistics*, 23(2):522–540, September 2020. ISSN 1468-4357. doi: 10.1093/biostatistics/kxaa038. URL `http://dx.doi.org/10.1093/biostatistics/kxaa038`.

Ruilin Li, Christopher Chang, Yosuke Tanigawa, Balasubramanian Narasimhan, Trevor Hastie, Robert Tibshirani, and Manuel A Rivas. Fast numerical optimization for genome sequencing data in population biobanks. *Bioinformatics*, 37(22):4148–4155, June 2021. ISSN 1367-4811. doi: 10.1093/bioinformatics/btab452. URL http://dx.doi.org/10.1093/bioinformatics/btab452.

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.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Springer US, 1989. ISBN 9781489932426. doi: 10.1007/978-1-4899-3242-6. URL http://dx.doi.org/10.1007/978-1-4899-3242-6.

Daniel J. McDonald, Xiaoxuan Liang, Anibal Solón Heinsfeld, and Aaron Cohen. *sparsegl: Sparse Group Lasso*, 2023. URL https://CRAN.R-project.org/package=sparsegl. R package version 1.0.2.

Lukas Meier. *grplasso: Fitting User-Specified Models with Group Lasso Penalty*, 2020. URL https://CRAN.R-project.org/package=grplasso. R package version 0.4-7.

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.

Melinda C. Mills and Charles Rahal. A scientometric review of genome-wide association studies. *Communications Biology*, 2(1), January 2019. ISSN 2399-3642. doi: 10.1038/s42003-018-0261-x. URL http://dx.doi.org/10.1038/s42003-018-0261-x.

J.M. Myre, E. Frahm, D.J. Lilja, and M.O. Saar. Tnt-nn: A fast active set method for solving large non-negative least squares problems. *Procedia Computer Science*, 108:755–764, 2017. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2017.05.194. URL https://www.sciencedirect.com/science/article/pii/S1877050917307858. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland.

Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort, and Joseph Salmon. Gap safe screening rules for sparsity enforcing penalties. *J. Mach. Learn. Res.*, 18(1):4671–4703, jan 2017. ISSN 1532-4435.

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.

Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7695–7705. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/pilanci20a.html`.

Junyang Qian, Yosuke Tanigawa, Wenfei Du, Matthew Aguirre, Chris Chang, Robert Tibshirani, Manuel A. Rivas, and Trevor Hastie. A fast and scalable framework for large-scale and ultrahigh-dimensional sparse regression with application to the uk biobank. *PLOS Genetics*, 16(10):e1009141, October 2020. ISSN 1553-7404. doi: 10.1371/journal.pgen.1009141. URL `http://dx.doi.org/10.1371/journal.pgen.1009141`.

Joseph D Romano, Trang T Le, William La Cava, John T Gregg, Daniel J Goldberg, Praneel Chakraborty, Natasha L Ray, Daniel Himmelstein, Weixuan Fu, and Jason H Moore. Pmlb v1.0: an open source dataset collection for benchmarking machine learning methods. *arXiv preprint arXiv:2012.00058v2*, 2021.

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`.

Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A Sparse-Group lasso. *J. Comput. Graph. Stat.*, 22(2):231–245, April 2013a.

Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. *SGL: Fit a GLM (or Cox Model) with a Combination of Lasso and Group Lasso Regularization*, 2019. URL `https://CRAN.R-project.org/package=SGL`. R package version 1.3.

Noah R. Simon, Jerome H. Friedman, and Trevor J. Hastie. A blockwise descent algorithm for group-penalized multiresponse and multinomial regression. *arXiv: Computation*, 2013b. URL `https://api.semanticscholar.org/CorpusID:18386302`.

Dinesh Singh, Phillip G Febbo, Kenneth Ross, Donald G Jackson, Judith Manola, Christine Ladd, Pablo Tamayo, Andrew A Renshaw, Anthony V D'Amico, Jerome P Richie, Eric S Lander, Massimo Loda, Philip W Kantoff, Todd R Golub, and William R Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1(2):203–209, March 2002.

Martin Slawski and Matthias Hein. Non-negative least squares for high-dimensional linear models: Consistency and sparse recovery without regularization. *Electronic Journal of Statistics*, 7(none): 3004 – 3056, 2013. doi: 10.1214/13-EJS868. URL `https://doi.org/10.1214/13-EJS868`.

Mashaal Sohail, María J. Palma-Martínez, Amanda Y. Chong, Consuelo D. Quinto-Cortés, Carmina Barberena-Jonas, Santiago G. Medina-Muñoz, Aaron Ragsdale, Guadalupe Delgado-Sánchez, Luis Pablo Cruz-Hervert, Leticia Ferreyra-Reyes, Elizabeth Ferreira-Guerrero, Norma Mongua-Rodríguez, Sergio Canizales-Quintero, Andrés Jimenez-Kaufmann, Hortensia Moreno-Macías, Carlos A. Aguilar-Salinas, Kathryn Auckland, Adrián Cortés, Víctor Acuña-Alonzo, Christopher R. Gignoux, Genevieve L. Wojcik, Alexander G. Ioannidis, Selene L. Fernández-Valverde, Adrian V. S. Hill, María Teresa Tusié-Luna, Alexander J. Mentzer, John Novembre, Lourdes García-García, and Andrés Moreno-Estrada. Mexican biobank advances population and medical genomics of diverse ancestries. *Nature*, 622(7984):775–783, October 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06560-0. URL `http://dx.doi.org/10.1038/s41586-023-06560-0`.

Philip B Stark and Robert L Parker. Bounded-variable least-squares: an algorithm and applications. *Computational Statistics*, 10:129–129, 1995.

Cathie Sudlow, John Gallacher, Naomi Allen, Valerie Beral, Paul Burton, John Danesh, Paul Downey, Paul Elliott, Jane Green, Martin Landray, Bette Liu, Paul Matthews, Giok Ong, Jill Pell, Alan Silman, Alan Young, Tim Sprosen, Tim Peakman, and Rory Collins. Uk biobank: An open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLOS Medicine*, 12(3):e1001779, March 2015. ISSN 1549-1676. doi: 10.1371/journal.pmed.1001779. URL `http://dx.doi.org/10.1371/journal.pmed.1001779`.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 00359246. URL `http://www.jstor.org/stable/2346178`.

Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor, and Ryan J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 74(2):245–266, 2012. ISSN 13697412, 14679868. URL `http://www.jstor.org/stable/41430939`.

Ryan J. Tibshirani. The lasso problem and uniqueness, 2012. URL `https://arxiv.org/abs/1206.0313`.

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.

Emil Uffelmann, Qin Qin Huang, Nchangwi Syntia Munung, Jantina de Vries, Yukinori Okada, Alicia R. Martin, Hilary C. Martin, Tuuli Lappalainen, and Danielle Posthuma. Genome-wide association studies. *Nature Reviews Methods Primers*, 1(1), August 2021. ISSN 2662-8449. doi: 10.1038/s43586-021-00056-9. URL `http://dx.doi.org/10.1038/s43586-021-00056-9`.

Mark H Van Benthem and Michael R Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(10):441–450, 2004.

Saverio Vito. Air Quality. UCI Machine Learning Repository, 2008. DOI: https://doi.org/10.24432/C59K5F.

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

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

Meng Wang and Ao Tang. Conditions for a unique non-negative solution to an underdetermined system. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 301–307, 2009. doi: 10.1109/ALLERTON.2009.5394815.

Meng Wang, Weiyu Xu, and Ao Tang. A unique "nonnegative" solution to an underdetermined system: From vectors to matrices. *IEEE Transactions on Signal Processing*, 59(3):1007–1016, 2011. doi: 10.1109/TSP.2010.2089624.

John S. Witte. Genome-wide association studies and beyond. *Annual Review of Public Health*, 31 (1):9–20, March 2010. ISSN 1545-2093. doi: 10.1146/annurev.publhealth.012809.103723. URL `http://dx.doi.org/10.1146/annurev.publhealth.012809.103723`.

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.

Zhen Xiang, Hao Xu, and Peter J Ramadge. Learning sparse representations of high dimensional data on large scale dictionaries. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL `https://proceedings.neurips.cc/paper_fEnsemblesiles/paper/2011/file/3ce3bd7d63a2c9c81983cc8e9bd02ae5-Paper.pdf`.

Zhen James Xiang and Peter J. Ramadge. Fast lasso screening tests based on correlations. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2137–2140, 2012. doi: 10.1109/ICASSP.2012.6288334.

James Yang and Trevor Hastie. A fast and scalable pathwise-solver for group lasso and elastic net penalized regression via block-coordinate descent, 2024a. URL `https://arxiv.org/abs/2405.08631`.

James Yang and Trevor Hastie. A fast and scalable pathwise-solver for group lasso and elastic net penalized regression via block-coordinate descent, 2024b. URL `https://arxiv.org/abs/2405.08631`.

Yi Yang and Hui Zou. A fast unified algorithm for solving group-lasso penalize learning problems. *Stat. Comput.*, 25(6):1129–1141, November 2015.

Yi Yang, Hui Zou, and Sahir Bhatnagar. *gglasso: Group Lasso Penalized Learning Using a Unified BMD Algorithm*, 2024. URL `https://CRAN.R-project.org/package=gglasso`. R package version 1.5.1.

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.

Yaohui Zeng and Patrick Breheny. The biglasso package: A memory- and computation-efficient solver for lasso model fitting with big data in r, 2018.

Yaohui Zeng, Tianbao Yang, and Patrick Breheny. Hybrid safe-strong rules for efficient optimization in lasso-type problems. *Computational Statistics & Data Analysis*, 153:107063, 2021. ISSN 0167-9473. doi: https://doi.org/10.1016/j.csda.2020.107063. URL `https://www.sciencedirect.com/science/article/pii/S0167947320301547`.

Barbara Żogała-Siudem and Szymon Jaroszewicz. Variable screening for lasso based on multi-dimensional indexing. *Data Mining and Knowledge Discovery*, August 2023. doi: 10.1007/s10618-023-00950-8. URL `https://doi.org/10.1007/s10618-023-00950-8`.

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`.